Mecánica Computacional Vol. XXI, pp. 3020–3036
S. R. Idelsohn, V. E. Sonzogni and A. Cardona (Eds.)
Santa Fe–Paraná, Argentina, October 2002

GENERATION OF A PRE-OPTIMIZED SURFACE MESH FROM A CSG MODEL

Y. Gardan [*], F. Heschung [*], and C. Minich [†]

[*] CMCAO Team, IFTS
7 boulevard Jean Delautre, 08000 Charleville-Mezieres, France
e-mail: gardan@infonie.fr, fheschung@netcourrier.com
web page: http://www.univ-reims.fr/UFR/IFTS/cmcao.html

[†] CMCAO Team, Metz University
Ile du Saulcy, 57045 Metz cedex 01, France
e-mail: minich@sciences.univ-metz.fr
web page: http://www.mim.univ-metz.fr/~minich

Key words: Mesh generation, CSG representation, pre-optimized mesh, advancing-front method.

Abstract. *Mesh generation remains a tricky and time-consuming step between design and finite element analysis. This paper proposes a new method to automatically build an optimized triangular mesh where density is adapted to finite element analysis. To achieve this goal, the method makes the most of both geometric data and information provided by the functional analysis of the product. This information determines where the mesh must be coarse or fine and, combined with a geometric model, allows to directly build an optimized triangular mesh meeting simulation requirements. For the geometric model, a CSG representation is used instead of a BRep because maintaining geometrical and topological consistency in a BRep raises important and sometimes unresolved issues. Moreover, it is possible to assign an importance regarding finite element analysis to each node of the CSG, which allows to determine the mesh density and nodes blending strategy. The mesh of Boolean nodes in the CSG is directly computed from the meshes of the operands. The paper highlights and solves the problems induced by this approach, that is triangles quality where meshes merge and errors introduced by approximating surfaces intersection with meshes intersection. A new meshing technique based on the advancing front method and taking into account triangles size variation between areas having very different densities, is presented.*

1. INTRODUCTION

Interfacing design and simulation is a key step in product life cycle as simulation must guarantee product quality before the first prototype is even manufactured and tested. For example, Aerospatiale Matra has been showing for several years its will to increase the quality of its simulations to such a level that the company will manufacture only one prototype of each plane and that this plane will even be marketed if it passes all in flight tests. Now, construction of the digital model that supports analysis, generally a mesh, remains a time-consuming task, which goes against ever increasing demands concerning the time to market. This paper proposes a new approach for surface mesh generation of solids, with a view to simulation. Unlike current methods which deduce the mesh from geometric information only, this new technique makes the most of both the functional information attached to the product and its shape description in a Constructive Solid Geometry representation. The paper is organized as follows. The second section recalls the reasons why mesh generation remains a tedious task, requiring a human intervention. This leads to the specifications of the new method. The next two sections introduce the method itself and solutions to the problems it raises. The final section summarizes the paper and lists some future work.

2. CURRENT APPROACHES AND GOALS

2.1 Current design-analysis interface

Most of the simulation systems require a mesh of the solids to be analyzed. But neither the mesh type – triangular, quadrangular …– nor, generally, the density are *automatically* adapted to the simulation needs. Consider, for example, a triangular mesh of the filling system of a foundry mould, the filling of which has to be simulated. The areas where the filling system opens onto the mould cavities are called the in-gates. These are small size pipes, compared with the whole filling system dimensions but they are very important for the flow simulation of the molten metal, as constraints are very high, there. If it was decided to mesh the filling system in a coarse way, to shorten the simulation time, it may happen that the in-gates disappear or that their geometry is highly damaged; in both cases, simulation results are no longer significant. If, on the contrary, the mesh is fine, in order to guarantee the in-gate presence in the mesh, the simulation time becomes disproportionate. To avoid these drawbacks, the operator(s) responsible for preparing the simulation have to provide the system with a large amount of information about the mesh. In most cases, this information was already provided during previous stages such as functional analysis, preliminary or detailed design. Such pieces of information are the functional model, business rules, form features, connections etc. For example, knowing the functions achieved by some parts of a solid would allow to set their importance for simulation and the mesh density in these areas; knowing that a shell has been used during detailed design allows to choose elements dedicated to shells; it also often happens that the mesh dimension (1D, 2D or 3D) or the interpolation functions have been previously given, during design or other analysis, implicitly or explicitly.

Now, the mesh tools do not make the most of these pieces of information, they just work out a mesh from geometric information. Thus, the generated mesh does not take into account the importance of such and such part of the solid. This need to provide the system with data that were already provided earlier, often in an implicit way, turns analysis into a real bottleneck in the design-simulation cycle.

2.2 Goals

The above survey shows that current mesh generation tools are generally not able to produce a pre-optimized mesh. So a mesh is generally calculated, a simulation is performed, the mesh is refined where necessary and so on. Our goal is to produce a pre-optimized mesh that highly reduces the number of these iterations. Other studies aiming at providing a pre-optimized mesh have already been published but they only apply to limited contexts: Vexo's software[1] generates a mesh of the filling system of a mould but the rules to optimize the mesh are embedded in the code; Salgado et al.[2] and Dolsack et al.[3] have designed a learning system, but it is dedicated to shapes that are mainly cylindrical. Yamada et al.[4] developed a framework to adapt the mesh of a part to another part, but the parts have to have similar shapes. Cuilliere's study[5] deals with general parts but assumes that feature recognition is possible, whereas this is not yet fully solved, especially for features that are not machining features.

Current meshing techniques mostly rely on a boundary representation (BRep) of the solid to be meshed, as this representation is the closest to the meshed model[6]. However, several factors prompt us not to use this representation. 1) a current trend in CAD is to reduce the importance of boundary representations in favor of higher semantics level models (feature-based models[7], functional models[8,9,10,11], requirement-function-behavior-structure models[12,13] …). 2) A BRep cannot easily keep track of joints, that is to say the areas where the entities that were combined to build the solid, merge. Yet, joints are places where constraints generally concentrate and that should be explicitly known so that it is possible to mesh them finely. 3) Besides joints, there are other areas on a solid's boundary that sometimes have to be meshed more finely (friction zones, heavily loaded zones…) and a BRep does not allow to bound these zones, unless dummy edges and faces are built. 4) Whatever the way to get the BRep, feature-based modeling or combination of primitives and half spaces bounded by free-form surfaces, Boolean operators remain basic operations in geometric modeling. They raise the major problem of computing intersections between surfaces, which is not yet solved for parametric surfaces. The most commonly admitted solution consists in approximating free-form surfaces intersection by broken lines, which of course leads to errors.

For these reasons, our goal is to generate the mesh not from the BRep but from the CSG description of the solid. The CSG tree is used simultaneously with the functional model and this paper defends the idea that this association reduces the number of human interventions, the computation times and produces a mesh the density of which is adapted to the simulation needs.

3. PROPOSAL

There are various quality criteria for meshes: chord error, shape and size of triangles and mesh pertinence. Several methods dealing with some of these criteria[14,15,16] were published but none is able to take all criteria into account. In particular, the last factor has been less investigated than others so the method proposed in this paper focuses on the last three criteria; chord error will be integrated in future work.

3.1 Mesh generation strategy

Usually meshing an object described by a CSG tree first involves the evaluation of its boundaries, the resulting faces being then triangulated. For the reasons listed above, we decided not to base our method on a BRep but, instead, to build the mesh directly from the CSG tree: each primitive is triangulated first, then the resulting meshes are combined by Boolean operations, as requested in the CSG tree. Boender and Bronsvoort[17] also developed a method to build a mesh from a CSG tree, but they didn't address the problem of pre-optimization. In the next two sub-sections, the advantages and drawbacks of our approach are listed and the main difficulties it raises are presented.

3.2 Advantages and drawbacks

The design process of a product generally starts with a functional analysis. This analysis consists in a recursive decomposition of the product's main functions and continues until it becomes possible to assign to each elementary function a simple shape. This decomposition produces the so-called functional description. The simple shapes may be surfaces or primitive solids – spheres, cylinders, boxes… – so they can be described by small CSG trees. The aggregation of these small trees comprises a description of the entire product, often called design history, although this name is a little bit restrictive: the functional description might legitimately belong to this design history. As the present study is about finite element analysis of parts, the product is a single solid and we make the not very restrictive hypothesis that the design history is a CSG tree.

The big advantage of the method is that, as the CSG tree is built, the functional description is still very present, wherever in a data structure or in the designer's mind. And we consider this makes it possible to assign to each leaf of the tree and then to each non-terminal node, its importance with regard to simulation. Consider for example the design of a mould. Its filling system is built by combining several basic components such as pipes, a funnel, feeders, in-gates, cavities that are negatives of the part to be manufactured… When an in-gate is added as a basic component, that is as a leaf in the tree, it is known that it is a high constraints area and that the mesh should be very thin there.

For a leaf, this extra information that can be deduced from the functional description consists in a so-called *pertinence coefficient* and a *nodal distance*. The latter sets the maximal permitted distance between two vertices of a triangle of the mesh. So it has a strong influence on the mesh density, although other criteria, such as chord error, also matter. In the current version of this study, the nodal distance is the only factor impacting on the density. The

pertinence coefficient is used when two meshes are combined: during a combination, it may happen that the two involved meshes have very different densities and that they have to be adapted in the zones where they merge, to ensure a reasonable density variation. The pertinence coefficient is used to quantify the adaptation on the two meshes: the larger the pertinence coefficient, the more accurate the simulation should be in the corresponding area and the more important it is to reduce the size variation between two adjacent triangles of the mesh. If a zone has a small pertinence coefficient, a stronger size variation can be accepted, which allows to link it to a zone with a very different density.

For a non-terminal CSG node, the only extra information is a nodal distance. It defines the maximal permitted distance between consecutive vertices on the intersection curves of the two meshes that are combined by the CSG node. The need for this nodal distance results from the fact that constraint concentrations often happen nearby intersection curves, making it necessary to mesh them and their neighborhood with a given nodal distance. The value of this nodal distance is less likely to be automatically determined from the functional decomposition. So, if the user knows it, he can give it to the system at the time when he builds the CSG node, otherwise, a heuristic is applied to determine the nodal distance as the meshes are combined. This heuristic simply consists in computing the average of the distances between vertices of the intersection curves. The vertices are then moved on the intersection curves so that their spacing meets the nodal distance.

The main drawback of the method is that the whole mesh has to be reevaluated when a leaf is modified. This happens for example when an in-gate nodal distance is reduced, for a more accurate simulation. This could be partially overcome with techniques similar to incremental BReps but has not been considered in the current version.

3.3 Combining meshes

A conventional Boolean combination module cannot be used to combine two meshes. First, this would generate non-triangular elements where the meshes join. Second, when the combined meshes have different densities, the triangles have to be adapted so that the size variation between adjacent triangles complies with the simulation needs. So a mesh combination technique has been developed. It is detailed for the union operator; intersection and difference use similar principles.

The first step consists in computing the intersection curves between the two meshes. Actually, these curves are polygons and are called *intersection loops* in what follows. It can be noticed that the nodal distance on the intersection loops is generally smaller than on the operands. Then, all triangles that do not entirely belong to the resulting mesh are removed. For the union operation, the concerned triangles are those which are totally or partially included in the solid bounded by the other mesh. When removing a triangle, a particular attention has to be paid to so-called initial edges: a side of a triangle is an *initial edge* if it is carried by an actual edge of the solid to be meshed. When deleting a triangle, its initial edges are removed only if they entirely lie inside the other solid. Otherwise, they are trimmed so that only the part inside the other solid is deleted.

At this stage, unmeshed zones remain that must be filled in by linking together the intersection curves to the boundaries of the "open" meshes. Figure 1 shows an example where the intersection loop has to be linked together to two meshes.

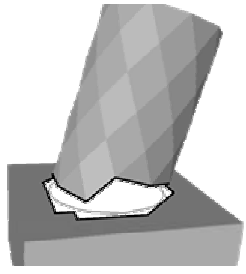

Figure 1. Union of a box and a cylinder

The final stage of the combination is the re-meshing of the gaps. For any gap, the main problem is to know whether it is possible to build new triangles from the intersection loops to the boundary of the "open" mesh so that the triangle sizes do not vary excessively. The various aspects of this stage are detailed in the next section.

4. REMESHING

4.1 Issues

The problem is to mesh a three dimensional area bounded by intersection loops and the boundary of an "open" mesh. This mesh corresponds to one son of the Boolean operation so the re-meshing operation has to be performed twice.

Nodal distances can be sharply smaller on intersection loops than on mesh boundary. It is then important to manage size variation over the new mesh so that two neighbor elements – triangles in this case – always have consistent dimensions, to maintain triangles shape and size quality. Current meshing algorithms can triangulate most of the parametric surfaces while reducing approximation and stretching errors through metrics, but do not care about mesh density variations.

Our strategy first consists in computing the maximal enlargement value (Δ_{max}) in the area that has to be re-meshed. This value is directly deduced from the pertinence coefficient when the "open" mesh is a leaf of the CSG tree. For example, if the pertinence coefficient is high, the simulation has to be very accurate so Δ_{max} has to be very low. If the "open" mesh corresponds to a sub-tree, then Δ_{max} is simply the smallest Δ_{max} of all the leaves of this sub-tree.

Then, new elements – or triangles – are created, from the intersection loops to the boundary of the "open" mesh. These new elements are built while taking into account Δ_{max}; however, it

may happen that the gap between the intersection loop and the boundary of the "open" mesh is not large enough and it is not possible to perform this operation. In that case, instead of increasing the triangles' size faster, other triangles are removed on the "open" mesh, until it becomes possible to connect the new mesh to the "open" mesh. As it is difficult to determine a priori how much space is required to cope with the nodal distance variation, the un-meshing operation is performed dynamically, while the new mesh is being constructed.

The end of this section (from 4.2 to 4.9) details the re-meshing operation in a plane domain. Section 5 explains how to extend it to parametric surfaces.

4.2 Plane re-meshing

The problem is to mesh a plane area bounded by loops with different nodal distances. For example, in figure 1, one of the two areas to be re-meshed is plane because it lies on the cube's top face. It is bounded by an intersection loop, which is approximately an ellipse, and by the boundary of the "open" mesh, which is a broken line. The meshing algorithm developed within this project is based on an advancing front method. So we briefly summarize this type of method in the next section.

4.3 Advancing front method

The area to be meshed is bounded by one or several fronts. A front is a sorted list of oriented edges. At each iteration, an edge $[S_iS_{i+1}]$ from one front is chosen (figure 2). A vertex Q is then computed so that $(S_iS_{i+1}Q)$ is an ideally shaped triangle. In the case of an isotropic mesh in the plane, the ideal shape is typically an equilateral triangle.

Then if one vertex V of any front lies in the neighborhood of Q, V is chosen to build a new element in the triangulation and Q is neglected. Otherwise Q is chosen to build the new triangle. Of course, in both cases, some validation tests, such as the absence of intersection with other triangles, must be performed. Fronts are then updated, which may consist in splitting a front, adding a line segment in it or in merging two fronts. This process is applied until all fronts are empty.

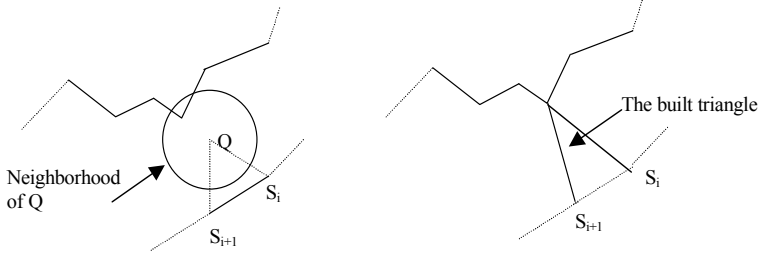

Figure 2. An advancing front method

Next section describes how to build the next triangle based on $[S_iS_{i+1}]$ in our context. It

must be clearly underlined that, although the proposed method takes several successive triangles into account to make a better forecast, only one triangle is built at each iteration.

4.4 Maximal enlargement control

We assume for the explanation that the nodal distance on intersection loops is smaller than on the "open" mesh boundary. This means that the triangles built from the intersection loops to the mesh boundary have to have increasing dimensions. Otherwise, all calculations are symmetric and the triangles decrease. In what follows, *current front* denotes the front in which new triangles are added. So, at the beginning of the re-meshing, the current front is the intersection loop. The opposite front will denote the front towards which new triangles are built.

Because of the need for an enlargement, the ideal triangle does not have an equilateral shape, in our case. The ideal shape has to take into account the best way to pass gradually from one nodal distance to another (from (S_iS_{i+1}) to (P_iP_{i+1}) in figure 3).

First the minimal number of enlargements that would be required if the maximal enlargement were used to pass from the current front nodal distance to the opposite front nodal distance, is computed. Let's call this number NE. Then, NE *virtual* isosceles triangles are built and stacked so that their third vertex stays on the perpendicular bisector of $[S_iS_{i+1}]$. (S_iS_{i+1}) is the basis of the first triangle and their heights increase by Δ_{max} at each time. The furthest vertex on the last triangle is the furthest point that can be reached by NE triangles that respect the maximal enlargement constraint (Q_3 in figure 3). If this vertex lies beyond the opposite front, that is to be beyond $[P_iP_{i+1}]$ in figure 3, then it is assumed that the triangulation is not possible and that the opposite front has to be modified (see 4.8). Otherwise, the sufficient size variation is computed from the actual distance between the two fronts and a new triangle is built based on $[S_iS_{i+1}]$, using the sufficient enlargement.

The algorithm to build a new triangle can be summarized as follows:

- Evaluate the distance between $[S_iS_{i+1}]$ and the opposite front (see 4.5). Let $[P_iP_{i+1}]$ be the first edge in the opposite front cut by the perpendicular bisector of $[S_iS_{i+1}]$.
- Given Δ_{max}, compute the distance required to absorb the nodal distances between both fronts (see 4.6).
- Test the position of the vertex corresponding to this distance along m with regard to $[P_iP_{i+1}]$.
- If this vertex lies before the opposite front, that is between $[S_iS_{i+1}]$ and $[P_iP_{i+1}]$, then calculate the sufficient enlargement (see 4.7), construct one new vertex on $[S_iS_{i+1}]$'s perpendicular bisector and build a new triangle with S_i, S_{i+1} and the new vertex. The new vertex is located at a distance corresponding to S_iS_{i+1} increased by the sufficient enlargement.
- Otherwise modify the opposite front (enlarge the area to be re-meshed or decrease the nodal distances of the opposite front – see 4.8).

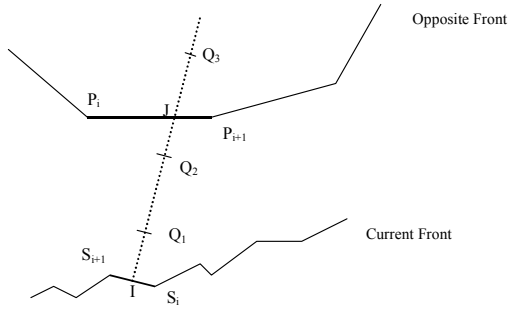

Figure 3. Distance between two fronts

4.5 Distance between the two fronts

To determine the distance between the two fronts, we proceed as follows. Let I be the middle of $[S_iS_{i+1}]$ and J the projection of I along m on the opposite front. We define the distance as the length $\|IJ\|$. The distance is chosen this way because the perpendicular bisector of $[S_iS_{i+1}]$ gives the direction for building the new vertex. A comment about this convention is given in 4.9.

4.6 Distance required between the two fronts

The nodal distance in J is defined as the distance between P_i and P_{i+1} ($\|P_iP_{i+1}\|$). We are searching for the distance required along m to reach the nodal distance of the opposite front while respecting Δ_{max}. To do this, as explained in 4.4, we build virtual triangles the heights of which increase by Δ_{max} at each time. The height of the n^{th} triangle is given by the recursively defined sequence $(u_n)_n$:

$$u_n = \Delta_{\max}^n \cdot \frac{\sqrt{3}}{2} \left\| S_i S_{i+1} \right\| \tag{1}$$

Let the series $(S_n)_n$ be the sum of the n first terms of $(u_n)_n$. $(S_n)_n$ describes the distance covered along m after n successive enlargements. Let Q_n be a point on m so that $\|IQ_n\| = S_n$. We can compute the nodal distance ND (Q_n) for any Q_n:

$$ND(Q_n) = \frac{2\Delta_{\max}}{\sqrt{3}} u_{n-1} = \Delta_{\max}^n \left\| S_i S_{i+1} \right\| \tag{2}$$

We now search for p so that, for all $n > p$, the nodal distance in Q_n is equal to or greater than the nodal distance in J, that is:

$$ND(Q_p) \geq \left\| P_i P_{i+1} \right\| \tag{3}$$

This leads to

$$p \geq \log_{\Delta \max} \left(\frac{\left\| P_i P_{i+1} \right\|}{\left\| S_i S_{i+1} \right\|} \right) \tag{4}$$

We choose p as the smaller integer value satisfying this inequality. Then, three cases may occur:

- p is null, which means both fronts have similar nodal distances.
- p is negative. The nodal distance on the opposite front is less than the one on the current front. This situation can be detected before. In this case, the computations are the same except for the determination of the sufficient enlargement, which is symmetric.
- p is positive. This is the most frequent situation, where the nodal distance on the opposite front is greater than the one on the current front.

We assume in the following that p is positive or null. To know whether it is possible to add a new triangle based on $[S_iS_{i+1}]$ while meeting the size variation constraint, we test the position of Q_p with regard to line (P_iP_{i+1}). If $\|IQ_p\| > \|IJ\|$ (i.e. Q_p lies beyond the opposite front) then the distance between the fronts is not sufficient: even if the triangles increase as fast as permitted, it is not possible to pass from the nodal distance on the current front to the nodal distance on the opposite front. In this case, the opposite front must be altered (see 4.8), otherwise, a new vertex is build based on the sufficient enlargement which is computed as follows.

4.7 Sufficient enlargement

Here, we assume the steps described above have determined that it is possible to link the two fronts while respecting Δ_{max}. However the nodal distance difference between the fronts may not require that the triangles grow according to the maximal enlargement. To compute the sufficient enlargement, the number of necessary steps is calculated first.

$$r = \frac{\left\| IJ \right\| - \left\| IQ_p \right\|}{u_p} + p \tag{5}$$

Then we compute Δ, the sufficient enlargement, so that

$$ND(Q_r) = \left\| P_i P_{i+1} \right\| \tag{6}$$

This gives

$$\Delta = \sqrt[r+1]{\frac{\|P_i P_{i+1}\|}{\|S_i S_{i+1}\|}} \tag{7}$$

Depending whether the nodal space on the current front is greater or less than the nodal space on the opposite front, the sufficient size variation is set to Δ or $1/\Delta$. Then a new vertex V is built on m at a distance equal to $\|S_iS_{i+1}\|$, increased or reduced by the sufficient size variation. As explained in 4.3, a new triangle based on S_i, S_{i+1} and V or a vertex close to V is finally added to the triangulation.

4.8 Opposite front modification

In the case when it is not possible to build a triangle based on segment $[S_iS_{i+1}]$, there are two possibilities, depending whether $[P_iP_{i+1}]$ is a floating edge or not. $[P_iP_{i+1}]$ is a floating edge if and only if it is neither an initial edge nor a line segment belonging to an intersection curve.

If $[P_iP_{i+1}]$ is a floating edge, then it is just deleted, which means that a triangle is removed in the mesh that has to be reached. This corresponds to the un-meshing operation mentioned in the overall approach described in 4.1.

If $[P_iP_{i+1}]$ is not a floating edge, it is split by inserting a node in its middle and the process described above starts again until the nodal distance on the opposite front is sufficiently close to the one on the current front. Precautions must be taken after each node insertion: $[P_iP_{i+1}]$ belongs to an existing triangulation, because operands are meshed before Boolean calculation. So Δ_{max} could no longer be respected in the neighborhood of $[P_iP_{i+1}]$ because of the node insertion. To compensate this Δ_{max} irregularity, node insertions are propagated throughout the existing triangulation beyond $[P_iP_{i+1}]$ using the longest edge bisection[18]. This method consists in bisecting an edge and in recursively propagating the edge splitting to the adjacent triangle sharing the longest edge of the split triangle.

4.9 Remarks

There is no guarantee that this method gives the better triangulation. Particularly, the distance between $[S_iS_{i+1}]$ and $[P_iP_{i+1}]$ might not reflect the actual distance between the fronts (Figure 4). In this situation, some vertices – A and B – are very close to the current line segment. These vertices prevent the mesh from being generated while respecting Δ_{max} without modifying the opposite front (vertex insertion or triangle suppression). But the algorithm does not detect this situation in the current iteration.

However, the distance between the two fronts is computed for each line segment of the current front so if the fronts can not be linked, the algorithm as it is proposed will detect it later, although, in the meanwhile, some triangles will have been built with the maximal enlargement while it was not required. Even so, the technique we use to evaluate the distance between the fronts gives the direction where the new vertex should be created and allows to

generate a mesh satisfying Δ_{max}.

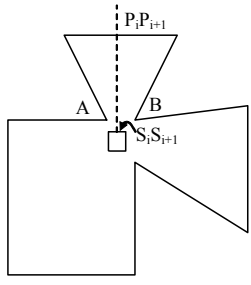

Figure 4. Evaluation of the distance between two fronts

Another important point concerns the convergence of the algorithm. This question takes place for two reasons. First, while the current front moves forward, the opposite may move backwards and they might never meet. Second, edges of the opposite front could be split with Rivara's edge bisection method[18] and it must be checked that the modifications this causes, end.

In the case of a backwards progress of the opposite front, we ensure convergence by never removing edges of the pre-existing mesh boundary. This ensures that the mesh always progresses from the current front up to the pre-existing mesh boundary. So, even if the opposite front (the pre-existing mesh boundary) progresses backwards, the algorithm does not loop infinitely. In the worst case, all triangles of the pre-existing mesh are removed except initial edges and the mesh is entirely re-generated. So, even if the opposite front progresses backwards, the process converges.

Rivara's edge bisection method defines the longest side propagation path, which is built with all the triangles that share their longest edge. To ensure the convergence, only triangles in the pre-existing triangulation are split, otherwise edges of the current front could be split and the size variation between triangles of the two fronts would not decrease.

This method allows to mesh areas having strong nodal densities variations. It proceeds by stretching triangles as earlier as possible and by front advance and backing up. However, in most cases, the area to be re-meshed is not plane but skew. Next section shows that the method can be adapted to parametric surfaces under some conditions.

5. ADAPTATION TO PARAMETRIC SURFACES

The method described above relies on the notion of distance along a perpendicular bisector. It uses 2D geometric operations like line-line intersections to mesh a plane area and it is difficult to adapt these operations to mesh a surface in the three dimensional space. So, to adapt the algorithm to surfaces, the mesh is generated in the parametric space, which reduces the problem to a 2D triangulation. In what follows, we'll call P: (u,v) → (x,y,z) the regular

mapping from parametric space to real space R^3.

The preliminary step of the re-meshing is to convert all points of both fronts in the parametric space. In what follows, all points are supposed to be in the parametric space. If S_i and S_{i+1} are two consecutive points of the current front, the goal is to compute a third vertex V so that (P (S_i), P (S_{i+1}), P (V)) is a good element, that is to be a good triangle.

To control the shape and size of the elements that make up the mesh, the method requires distances, perpendicular bisectors and projections. But the mapping from parametric space to real space does not preserve distances and angles. For example, the images of two line segments that are perpendicular in the parametric space are not perpendicular line segments of the 3D space; they are even not line segments. In our context, the two basic operations that have to be adapted are the construction of the perpendicular bisector of a line segment of the front and the distance between two points. Do Carmo[19] and Cuilliere[20] propose solutions to similar issues in the case of regular parametric surfaces, that is to say it is possible to convert any point from one space to the other and there exists a tangent plane in any point of the surface. The next two paragraphs summarize and adapt them to our context.

5.1 Length of a line segment

Let [A,B] be the line segment whose length is required. We define the distance between A and B as the length of the geodesic linking these two points. We assume that this length is correctly approximated by the length of the 3D curve corresponding to the segment [A,B] in the parametric space (P ([A,B])). So a parameterization (u(t),v(t)) of [A,B] is first determined; a(t)=P(u(t),v(t)) is a curve lying on P and its integration over t gives the required length.

5.2 Determination of a perpendicular bisector

To evaluate the distance between the two fronts, the perpendicular bisector to a line segment of the front is required. Let $[S_iS_{i+1}]$ be the segment the perpendicular bisector of which is required. Of course, the bisector B in the parametric space is not satisfactory as its image in R^3 is not perpendicular to [P (S_i),P (S_{i+1})]. So the 3D perpendicular bisector is calculated as follows: in the parametric space, the straight line m making an angle θ with B and passing through the middle I of $[S_iS_{i+1}]$ is determined (Figure 5). θ is then chosen so that the image of m in R^3 is perpendicular to the image of $[S_iS_{i+1}]$ in P (I) (see below). Then the distance between the two fronts can computed as the distance between I and the intersection of m and the opposite front.

To compute θ, $[S_iS_{i+1}]$ is parameterized in the same way as we did to measure the line segment length (see 5.1). Let a (t) be the corresponding curve in R^3; m is also parameterized and the corresponding curve is called b (w), which depends on θ. θ is then calculated so that the tangents to a(t) and b(w) are perpendicular in P (I), that is:

$$\frac{\partial a}{\partial \mathrm{t}} \cdot \frac{\partial b}{\partial \mathrm{w}} = 0 \tag{8}$$

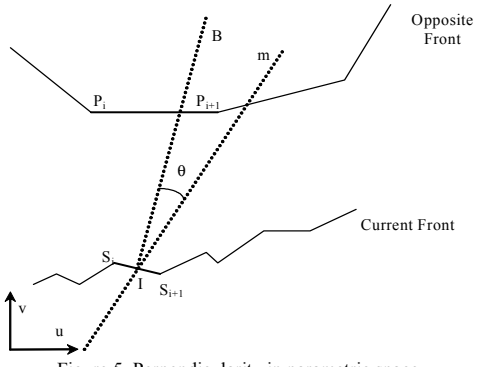

Figure 5. Perpendicularity in parametric space

This leads to[20]:

$$\tan\theta = \frac{(E-G)(u_1-u_0)(v_1-v_0)-F\left[(u_1-u_0)^2-(v_1-v_0)^2\right]}{E(u_1-u_0)^2+G(v_1-v_0)^2+2F(u_1-u_0)(v_1-v_0)} \tag{9}$$

where E, F and G are the coefficients of the first fundamental form of the surface and $S_0=(u_0, v_0)$ and $S_1=(u_1, v_1)$.

These relations enable to compute distances between fronts, triangles heights and sufficient enlargement for each new vertex construction. It is then possible to generate a mesh on a regular parametric surface having a strong nodal distance variation, while respecting Δ_{max} imposed by the functional reasoning.

5.3 Approximation error control

One point that has not been tackled yet concerns approximation errors on the intersection loops vertices. These vertices are obtained by intersecting triangles. This implies that they generally lie on none of the two (or more) surfaces. To correct this error, we apply the following process, which is based on a regular-parametric-surfaces intersection method[21].

Let S (u,v) and T (w,r) be two regular parametric surfaces:

$$\begin{aligned} S(u,v) &= x_S(u,v),\ y_S(u,v),\ z_S(u,v) \\ T(w,r) &= x_T(w,r),\ y_T(w,r),\ z_T(w,r) \end{aligned} \tag{10}$$

The intersection process gives a set of points, close to the exact intersection curve C. We are going to move these points to decrease the distance between these points and C. Let P_i be one of these points close to C and let Q_i (x_{Qi} ,$y_{Qi,}$ z_{Qi}) be the same point after it has been moved on C. If (u_q,v_q) and (w_q,r_q) are Q_i's coordinates in the parametric spaces of S and T, then

$$S(u_p,v_p) - T(w_p,r_p) = 0 \tag{11}$$

This linear system has 3 equations and 4 unknowns u_p, v_p, w_p and r_p. A fourth equation is required to solve it and find Q_i. To get this fourth equation, we request Q_i to lie on the plane containing P_i and the normal of which is P_iP_{i+1}. There are two possibilities to express this constraint, with S or T. Berroug[21] advises to choose the most regular mapping. Assuming S is the most regular surface, the system becomes:

$$\begin{cases} S(u_p,v_p)-T(w_p,r_p) = 0 \\ a.x_S(u_p,v_p) + b.y_S(u_p,v_p) + c.z_S(u_p,v_p) + d = 0 \end{cases} \tag{12}$$

where a, b, c, d are the coefficients of the plane.
To solve this, a Newton-Raphson method can be applied with P_i as a first solution.

This method allows to control approximation errors on the intersection points and to insert new nodes on the intersection loops. This makes it possible to define the nodal distance on intersection loops and, thus, to better take into account the functional directives.

6. CONCLUSIONS

This paper presents a mesh generation technique which consists in meshing the part's primitives and in combining these meshes according to the CSG description of the part. The goal was to develop a method that generates a mesh while reducing manual interventions, immediately adapting node densities to simulation needs and maintaining triangles size and shape quality.

The reducing of manual interventions stems from the use of the functional model. The functional model allows to mesh judiciously the CSG primitives. When two meshes are combined and a reasonable evolution of adjacent triangles sizes has to be achieved, it allows to choose what operand has to be re-meshed and to what extent. This operation also guarantees the triangles size and shape quality.

The mesh pertinence results from the fact that the mesh is not uniform and that it is coarser in the areas where a high density is useless. Simulation times are thus reduced as much as they can be, without affecting results accuracy.

The fact of combining meshes saves from evaluating the part's BRep model; this differentiates our approach from classical ones. As intersections apply to meshes, no assumption is made on the nature of involved surfaces. However, to cope with triangles size variation, surfaces have to be regular.

Future work includes the consideration of chord error while meshing and a better use of the various implicit and explicit pieces of information the CAD system can provide, leading to even better early design/analysis integration. In a number on contexts, such as design of a mould filling system, the technique as described in this paper succeeds in producing a pre-optimized mesh. In other contexts, our intent is to combine it with other known approaches such as feature extraction or medial axis and medial surfaces techniques.

7. REFERENCES

[1] Vexo F., *Contribution à l'Intégration de la Simulation avec la CAO: Application à la Construction du Système de Remplissage en Fonderie*, Thèse de doctorat, Université de Reims, 2000

[2] Salgado N.K., AliabadiM.H., Callan R.E., *Rule Inferencing and Object-Orientation for Boundary Elements Mesh Design*, Artificial Intelligence in Engineering, vol.11, n°2 1997, p. 183-190

[3] Dolsak B., Muggleton S., *The Application of Inductive Logic Programming to Finite Element Mesh Design*, in Inductive Logic Programming. Academic Press, London, 1992, p.453-472

[4] Yamada A., Inoue K., Itoh T., Shimada K., *An Approach for Generating Meshes Similar to a Reference Mesh*, Proceedings. of 9th International Meshing Roundtable, 2000, p. 101-109

[5] Cuilliere J-C., *Pre-Optimisation de Maillages Automatiques Tridimensionnels pour les Méthodes Numériques - Application à l'Ingénierie Simultanée*, Thèse de doctorat, Institut National Polytechnique de Lorraine, Nancy, 1993

[6] P-L. George, H. Borouchaki, *Triangulation de Delaunay et Maillage - Applications aux Eléments Finis*, Edition Hermès. (1997)

[7] J.J. Shah, M. Mäntilä, *Parametric and feature-based CAD/CAM,* John Wiley & sons, Inc, 1995

[8] M. Ranta, M. Mäntylä, Y. Umeda and T.Tomiyama, *Integration of Functional and Feature-based product modeling – the IMS/GNOSIS experience,* Computer Aided Design, 28(5):371-381, 1996

[9] C. X. Feng, C. C. Huang, A. Kusiak and P. G. Li, *Representation of functions and features in detail design,* Computer Aided Design, 28(12):961-971, 1996

[10] Y. Shimomura Y. Yoshioka, H. Takeda, Y. Umeda, T. Tomiyama, *Representation of design object based on the functional evolution process model,* Journal of Mechanical Design, (120):221-229, June 1998

[11] Y. Gardan, C. Minich, D. Pallez, E. Perrin, *Towards a specifications-to-shape translation tool* in Third International symposium on Tools and Methods of Competitive Engineering, Delft, The Netherlands, April 18-21 2000

[12] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura et T. Tomiyama, *Supporting*

conceptual design based on the function-behavior-state modeler, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 10, pages 275-288, 1996

[13] Simon Szykman, Christophe Bochenek, J. W. Racz and Ram Sriram, *Design Repositories: Next-Generation Engineering Design Databases*, IEEE Intelligent Systems and Their Applications, January, 2000

[14] S.J. Owen, *A Survey of Unstructured Mesh Generation Technology*, Proceeding of 7th International Meshing Roundtable, p. 239-267, 1998

[15] M. Bern and D. Eppstein, *Mesh Generation and Optimal Triangulation*, in Computing in Euclidean Geometry, D.-Z. Du and F.K. Hwang, eds., World Scientific, 1992, 2nd edition, 1995

[16] P. Pebay, T.J. Baker, *A Comparison of Triangle Quality Measures*, Proceedings of 10th International Meshing Roundtable, 2001

[17] E. Boender, W.F. Bronsvoort, F.H. Post, *Finite-Element Mesh Generation from Constructive-Solid-Geometry Models*, *Computer-Aided Design*, Vol.26, n°.5, pp. 379-392, 1994

[18] M-C. Rivara, *New Mathematical Tools and Techniques for the Refinement and/or Improvement of Unstructured Triangulations*, Proceedings of 5th International Meshing Roundtable, p. 77-86, 1996

[19] PM. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976

[20] J-C. Cuilliere., *An Adaptative Method for the Automatic Triangulation of 3D Parametric Surfaces*, Computer-Aided Design, Vol.30, n°2, pp. 139-149, 1998

[21] M. Berroug, *Contribution à la résolution du problème d'intersection de deux carreaux de surfaces*, Ph.D, Université de Metz, (1995)