Automatic Mesh Generation from 3D Internal Sketch and Topological Constraints

Jean-Luc Mari
Information and System Science Laboratory (LSIS) - Computer Graphics Dept. (I&M),
University of Marseille II - Faculté des Sciences de Luminy,
Marseille, France
mari@univmed.fr

Abstract

In this paper, we introduce a new method to generate a mesh from a 3D sketch. The latter can be considered as a *skeleton* that defines the global shape of the object to model. To induce a natural adjacency, for further topological control, this inner structure is a set of voxels connected using the 26-neighborhood. Two important features are supported by the resulting surface: it is a multiresolution model thanks to the use of subdivision surfaces and its topology is equivalent to that of the skeleton. The main problem is to construct a basic polyhedron that surrounds the 3D sketch and complies with its topology. The idea is then to use this rough mesh as the *control polyhedron* of a subdivision surface to model a smooth multiresolution object, well suited for further modifications in the frame of a 3D modeling software.

Keywords: *Geometrical Modeling, Mesh, Topological Skeleton, Sketch, Control Polyhedron, Subdivision Surfaces.*

1. INTRODUCTION

The aim of our approach is to compute a surface that preserves the initial topology of a 3D sketch designed interactively. To do that, a skeleton is characterized from the points of the sketch. Its role is to define the connection relations using edges and triangles, and to represent the global shape of the object to model.

From this shape descriptor, the approach we develop generates a polyhedron *M* topologically equivalent to the skeleton. To do this, points are first created around vertices of the skeleton. The location of these points is the key issue of the method. Subsequently, they are linked to form a rough triangulation. No edges must cross, as well as no triangles must intersect. The closed resulting mesh is adequate to initialize a subdivision surface process.

One asset of subdivision surfaces is to handle arbitrary topologies. Thereby using the aforementioned generated mesh *M* as the *control polyhedron* for subdivision surfaces ensures to produce a sequence of meshes of equivalent topology. If *M* is homotopic to the skeleton, so are the refined surfaces.

Another advantage of subdivision surfaces is the natural support of *multiresolution*. Once the control polyhedron is generated through our method, access to several *levels of detail* (LoDs) within a single model is permitted. Thus depending on the context, a more or less refined mesh can be used. For example, low resolution meshes are adapted to preview, whereas high-res models of the same object can be employed for final rendering.

A direct application of such multiresolution meshes, generated from a simple 3D sketch, is the use in geometrical modeling softwares. Image synthesis and animation can naturally be done using the provided intuitive shape design.

To define the global appearance of an object, the designer is asked to draw a 3D sketch in a regular cubic grid. This mean to create a shape is fast and simple as it refers to *voxels* modeling (see below) to build an entity made with "cubes". The resulting form is rough but the method we will develop has topological guarantees of the final mesh. The sketch, converted into a skeleton, gives a suitable shape descriptor for further finer geometrical modifications.

Voxels (for volume elements) are the basic units in the frame of *digital volumes* (i.e. discrete objects with coordinates in $\mathbf{Z}^3$). One particularity of the voxels is to have an easy way to define their connectedness relation: two voxels are *26-adjacent* if they share at least one vertex, *18-adjacent* if they share an edge and *6-adjacent* if they only have a face in common. The most natural neighborhood is the 26 one, because as soon as two voxels touch, they are said to be adjacent. Thus this adjacency is used to characterize the future surface: if two voxels are not connected this way, the related surrounding surface does not have to be connected, to preserve the topology of the 3D sketch. Figure 1 shows these three kinds of adjacency.

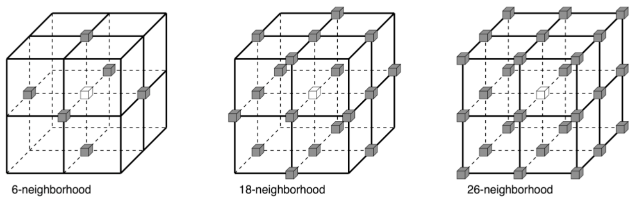

Figure 1: Three usual adjacencies in $\mathbf{Z}^3$: 6, 18 and 26.

This paper is divided in five major parts: an overview of related techniques is briefly developed in Section 2. The Section 3 is dedicated to the description of our approach. Starting from a discrete 3D sketch, a topological skeleton is created and followed by a basic surrounding polyhedron. The latter is suitable for multiresolution using subdivision surfaces, as explained in Section 4. We validate our method in Section 5 and finally we give tracks for future work in Section 6.

2. RELATED WORK

The most prominent offset surfaces based on skeletons are undoubtedly *implicit surfaces* [1]. Implicit modeling provides an intuitive mean to generate surrounding shapes, starting with a skeleton composed of simple primitives: points, segments, curves, triangles or patches. The resulting surfaces are continuous and naturally smooth. Moreover, they can be expressed in a mathematical way.

Usually, implicit surfaces are defined by a skeletal shape and a potential function. The choice of a scalar value associated to that function determines an isosurface around the skeleton. Many approaches have been elaborated for design [2], reconstruction [3] or animation [4].

Among implicit techniques, convolution approaches are particularly adapted to interactive modeling [5]. Thanks to a different distance definition, such surfaces do not present bumps on junction areas, which is a problem with classical implicit surfaces [6].

In the same way, another intuitive mean to model shapes interactively is implicit *virtual sculpture*. The object is manipulated like clay thanks to specific tools instead of skeleton primitives [7].

Variational implicit surfaces are used in the *BlobMaker project* to model free-form shapes using sketches. The main operations are based on inflations to create 3D forms from a 2D stroke [8].

The approach based on *Skins* [9] deals with skeleton implicit modeling and subdivision surfaces. It is a surface representation that uses particles to sculpt an object in an interactive way. Nevertheless, the computation time remains long and auto-intersection issues can appear when handling the shape as the topology of the skeleton is not connected with the surrounding subdivision mesh.

However, implicit modeling techniques suffer the same classical problem: it is difficult to control the topology of the final surface because *unwanted blendings* can appear when two yet unconnected skeletons elements are close. To sort out this point, *blending graphs* are often used to allow skeleton primitives to be jointed or not. This process slows down interactive handling and makes it less intuitive. Moreover, the second drawback of implicit surfaces is the real time rendering that is hard to obtain. Indeed, the computation time of the all potential functions over the isosurface increases when the number of primitives is high.

In the world of 3D modelers, to generate shapes interactively, the designer uses primitive objects like cubes, spheres or cones to get the rough appearance of the final form. Modifiers like *extrusion* or *spinning* (surfaces of revolution) allow to model more complex entities, but the use of shape descriptors like skeletons is rare. Most of the time, it is the contrary: the shape is designed first and a skeleton is added afterwards for animation.

The idea to use the skeleton as both a shape descriptor and, above all, a topological structure appears in [10]. Then developments and validation are made in [11] in the aim to get a model composed by three entities. An *inner skeleton* characterizes the structure, the global appearance and the topology. An *external layer* defines the boundary of the shape and all the geometrical details. Finally, a *transition layer* connects the two previous entities to maintain topological features between them. In the frame of interactive shape design, the surrounding layer is a mesh obtained thanks to a specific implicit surface. The latter is generated from the inner skeleton automatically not by using blending graphs but by setting skeletal points in a cubic grid to avoid surface intersection and topological degenerations. Thus implicit parameters are dependent on the grid resolution. However, to get the vertices of the mesh, we compute sample points of the implicit surface. They are then connected to obtain a polyhedron that defines the external layer of the model. These two steps (sampling and meshing) can fail depending on the chosen number of vertices: if the cloud of points is not dense enough, the triangulation module obviously collapses.

Therefore one asset of the above multilayer approach is the control of the topology by using the natural neighborhood of digital volumes: the surrounding layer is topologically equivalent to the inner skeleton. To follow this idea, we propose to define the skeleton from a 3D sketch designed in a cubic grid on one hand, and to mesh specific surrounding points according to topological rules on the other hand. The resulting polyhedron is then an appropriate starting point for further subdivisions and multiresolution design.

3. MESH GENERATION PROCESS

In this section, we develop the mesh generation process. In a first step, a 3D sketch is drawn in a cubic grid by a designer. Then this sketch is converted to a skeleton composed by vertices, edges and triangles that characterize the adjacency between voxels. Finally, a mesh is automatically generated around the skeleton structure. This polyhedron gives good representation of the global shape and a valid starting point for multiresolution via subdivision surfaces. All the operations on the generated mesh are made with respect to the topology of the sketch.

In the first subsection, we spell out the overview of the process, going from the sketch to the automatically generated polyhedron. In the second subsection, we explain the construction of the 3D skeleton. In the third subsection, we describe the computation of the surrounding polyhedron according to particular topological rules.

3.1 Overview of the Process: from the Sketch to the Polyhedron

During the mesh generation process, three entities are concerned: first a discrete sketch made of voxels, then the related skeleton that describes the topology of the shape, and finally the surrounding meshed surface (Figure 2).

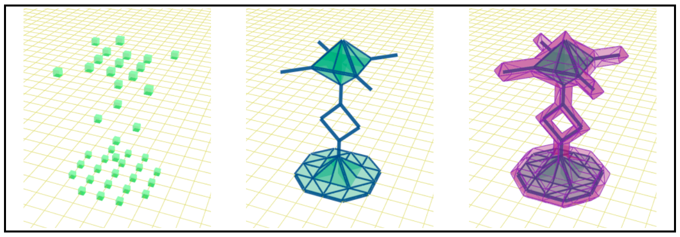

Figure 2: Sketch, skeleton and basic polyhedron.

We start from a set of voxels interactively set in a 3D grid by a designer. He first chooses the grid resolution (for example a subset 32^3 of the discrete space $\mathbf{Z}^3$). Then the designer draws voxels, knowing that the 26-adjacency will be used to determine the connected voxels.

During the voxels-edition step, the algorithm constructs a skeleton: segments and triangles are generated to emphasize the connection relations between voxels.

Once the skeleton is computed, based on the discrete sketch, a specific algorithm is used to generate a basic shape with the same topology as the skeleton. This is done due to the fact that the vertices of the surface are set up in safe locations, where further meshing cannot perturb the topology.

The resulting mesh is a good geometrical descriptor of the object. It gives an idea of the global shape, at low resolution level. This is why the mesh can be considered as a *control polyhedron* for subdivision. Using an approximating scheme like LOOP's will produce a surface slightly *inside* the volume of the initial polyhedron. This leads to a topological preservation, even at high resolution level.

3.2 Construction of the 3D Skeleton

We call *skeleton* the inner structure of the shape to be designed. It is made of three kinds of elements: the vertices (edited interactively by a manipulator), the edges and the triangles (computed to define the adjacency relation between the vertices). In other words, the skeleton is composed of the initial sketch plus segments and triangles.

3.2.1 *Interactive Edition of the Sketch*

The 3D sketch is edited interactively in the discrete space $\mathbf{Z}^3$: each voxel is located according to a regular grid (Figure 3).

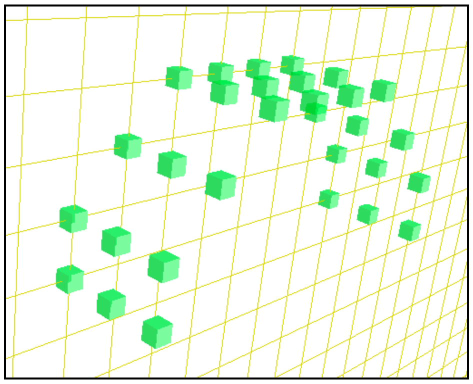
Figure 3: Edition of the sketch in a 3D grid.

Following this representation of the 3D sketch, an *adjacency list* in constructed for each voxel. The maximal number of neighbors is 26, due to the chosen connection relation. This set of voxels defines the points of the skeleton.

3.2.2 *Adjacency Representation of the Voxels*

For each voxel of all adjacency lists, an edge is added to the skeleton structure. The edges materialize pairs of connected voxels according to the 26-adjacency.

During the voxel edition phase, when an edge about to be added crosses another edge, its incorporation is canceled. This happens only for the 2×2 configuration (Figure 4).

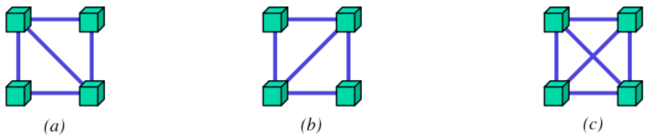

Figure 4: Good edge construction *(a,b)*; bad construction *(c)*

A list of triangles is made from the list of edges, by scanning 3-cycles of edges.

Figure 5 illustrates an elementary edition sequence: 3 points form a triangle (cycle of 3 edges) and a 4th point is added to constitute a second triangle, without edge crossing thanks to the middle edge removal routine seen previously.

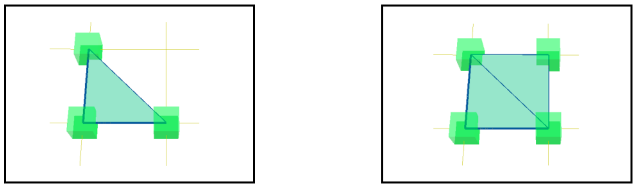
Figure 5: Edition sequence on a plane.

Considering the voxels of Figure 3, Figure 6 shows the edges of the final skeleton (on the left) and its triangles (on the right).

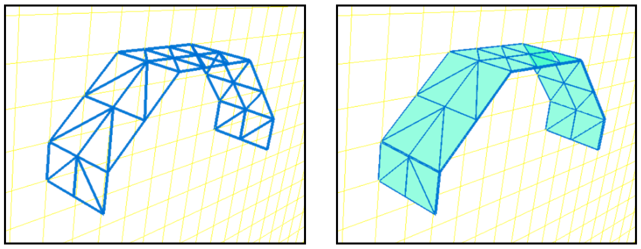
Figure 6: Edges and triangles of the skeleton.

The resulting skeleton structure, designed interactively in a three dimensional environment, characterizes the starting entity from which the surrounding polyhedron is computed.

3.3 Computing the Surrounding Polyhedron

The mesh to be computed has to surround the skeleton as well as preserving its topology. In particular, no auto-intersection must appear. Moreover, if the skeleton has a cycle or a hull, then the generated polyhedron must include a hole or a cavity. To this end, specific rules are elaborated to create vertices and to mesh them.

3.3.1 *Creating the Vertices*

The vertices of the basic polyhedron are created as follows. For each voxel of the skeleton, up to 6 vertices of the mesh can be generated. Depending on adjacency rules (supported by the edges and triangles of the skeleton), some points of the polyhedron are not created in order to get an actual surrounding mesh, with no points inside the wanted shape.

To simplify the problem, let us consider the size of a voxel as 10×10×10. The distance between the centers of two 6-adjacent voxels is then 10 *units*.

Let $S(x,y,z)$ be a voxel of the skeleton. For a single voxel *S*, six points of the mesh are generated: $P_1(x-4,y,z)$, $P_2(x+4,y,z)$, $P_3(x,y-4,z)$, $P_4(x,y+4,z)$, $P_5(x,y,z-4)$ and $P_6(x,y,z+4)$. This avoids interaction between non-adjacent voxels. The points of the mesh associated to a voxel of the skeleton will never be located farther than 4 units, to prevent auto-intersection during the meshing step (Figure 7, on the left).

However, when two voxels are 6-connected, only 5 points per voxel have to be generated, instead of 6. This is done thanks to *face-connection flags*, constructed during the edition of the sketch. Six boolean values are allocated to each voxel (*up*, *down*, *left*, *right*, *front* and *back*): when the latter has face neighbors (6-adjacent connected voxels), the related values are set to '*true*'. Therefore no points of the mesh are created in the specific directions of 6-connected voxels (Figure 7, on the right).

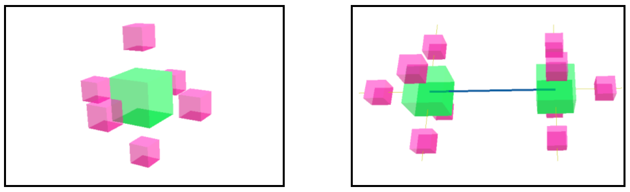
Figure 7: Surrounding points of a single voxel (on the left) and of two 6-adjacent voxels (on the right).

Naturally, for an inner voxel of the skeleton having 6 face-neighbors, no points of the mesh are generated at all.

3.3.2 Meshing the Vertices

Starting from the generated points, a critical step remains: meshing these vertices with the aim of getting a surrounding polyhedron that reflects the topology of the designed sketch.

The idea is first to generate all possible edges between the points P_i of the mesh, following specific linking rules, and then in a second phase to build the triangles of the mesh to get the final polyhedral crust of the object.

Before adding an edge E_i to the mesh structure, a series of tests is done.

Let P_1 and P_2 be the points that define E_i. The latter is added to the list of edges if the coordinates satisfy to one of the following conditions (the coordinates *x*, *y* and *z* are equivalent, thus permutations have to be done on the three axis to obtain all conditions):

- $P_1x = P_2x \pm 10$ and $P_1y = P_2y$ and $P_1z = P_2z$
- $P_1x = P_2x \pm 10$ and $P_1y = P_2y \pm 10$ and $P_1z = P_2z$
- $P_1x = P_2x \pm 10$ and $P_1y = P_2y \pm 6$ and $P_1z = P_2z \pm 6$
- $P_1x = P_2x \pm 10$ and $P_1y = P_2y \pm 4$ and $P_1z = P_2z \pm 4$ (to connect diagonal edges as shown in Figure 8 on the left).
- $P_1x = P_2x \pm 10$ and $P_1y = P_2y \pm 6$ and $P_1z = P_2z \pm 4$ (Figure 8 on the right).

These rules link all the points related to adjacent voxels that must be connected to form a surrounding mesh.

In addition, an edge E_i is not added to the list if:

- The new edge E_i intersects an existing edge in the list.
- E_i intersects an edge or a triangle of the skeleton.
- The length of E_i equals 2. That means the two extremities of E_i have only a gap of 2 units, i.e. the related voxels are not adjacent and should not be connected via the mesh.

For testing intersections of an edge with a triangle of the skeleton, we use the approach described in [12].

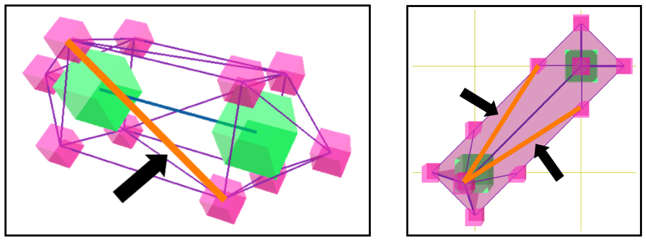
Figure 8: Linking vertices of the mesh according to specific rules.

The final step consists in constructing the triangles of the polyhedron: this computation is similar to the one detailed in section 3.2.2: the list of edges is scanned and all 3-cycles produce triangles. However one more test is done: if an edge of the skeleton intersects a triangle of the mesh, the latter is not stored as part of the resulting polyhedron.

Figure 9 illustrates the mesh related to the simplest skeleton composed of one single vertex.

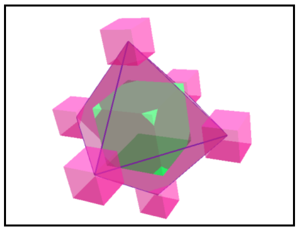
Figure 9: A single voxel produces an octahedron.

On Figure 10, the surrounding polyhedra related to skeletons composed of 2 and 3 voxels are presented.

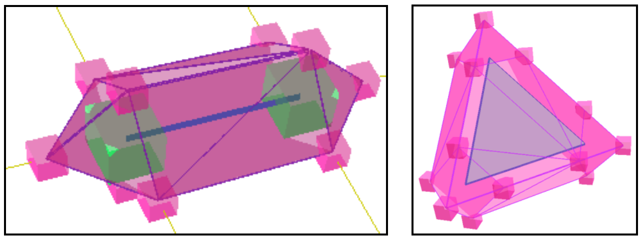
Figure 10: Polyhedra related to simple skeletons.

Figure 11 shows the entire triangulation process around a simple holed skeleton: a sketch is firstly edited in a cubic grid, then segments and triangles are computed to define the skeleton, and finally surrounding points, edges and triangles of the mesh are generated.

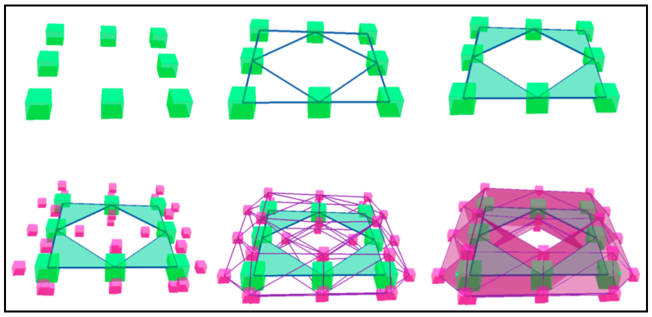
Figure 11: Overview of the process.

These generated meshes are by definition simple and rough. They define the global shape of the object to be designed and thus are well suited to be used as *control polyhedra* for subdivision surfaces. Thereby further multiresolution editing can still be done on resulting meshes, according to the desired level of resolution. Among numerous subdivision schemes, we naturally focus on the ones dedicated to triangulated meshes, and more particularly to LOOP scheme.

4. MULTIRESOLUTION

In geometrical modeling, the multiresolution feature is the ability for a model to enclose several *levels of detail* (called *LoD* in literature) to characterize an object. Depending on the context, the object can be represented either as a rough shape (a low resolution mesh within the LoD pyramid) or as a refined shape (a detailed and high model). Therefore the choice of the LoD can save computation time when the model is rendered as a small entity in a 3D scene: only few polygons need to be displayed to get a satisfying visual aspect.

4.1 Subdivision Surfaces

In this frame, s*ubdivision surfaces* for shape modeling emerged about twenty years ago, following works of CATMULL and CLARK [13] and DOO and SABIN [14]. This approach has been developed only recently in a large panel of applications in computer graphics and CAD/CAM [15]. One reason of this development is the rise of multiresolution techniques, which focus on surfaces whose geometry is more and more complex.

The basic idea is to consider that every polygon of a mesh can be subdivided according to a specific scheme. The result of this sequence successively refined is a smooth surface when the number of iterations tends to infinity. Yet subdivision surfaces are visually satisfactory after few iterations.

Subdivision algorithms are recursive in nature. The generic process starts with a given polygonal mesh, called the *control polyhedron*. A refinement scheme is then applied to this mesh, creating new vertices and new faces.

4.1.1 Approximation vs. Interpolation

Subdivision schemes can be classified into two categories: interpolating and approximating. Interpolating schemes are required to match the original position of vertices in the control polyhedron (the positions of the new vertices in the mesh are computed based on the positions of nearby old vertices). Concerning approximating schemes, they adjust original positions as needed (the positions of old vertices might also be altered).

There is another division in subdivision surface schemes: the type of polygon that they operate on. Some function for quadrilaterals (quads), while others operate on triangles.

The following table presents a classification of the main subdivision techniques, depending on the type of generated mesh (triangular or quadrilateral) and on whether the scheme is approximating or interpolating.

	Triangular meshes	*Quadrilateral meshes*
Approximating	Loop	Doo-Sabin Midedge Catmull-Clark
Interpolating	Butterfly	Kobbelt

When using interpolation schemes, after subdivision, the control points of the original mesh and the new generated control points are interpolated on the limit surface.

Approximating means that the limit surfaces approximate the initial meshes and that after subdivision, the newly generated control points are not in the limit surfaces.

In our approach, to use subdivision surfaces for its multiresolution asset, we choose approximating schemes to ensure the topology preservation of the control polyhedron. With such techniques, the resulting refined mesh is always *inside* the volume of the original one. On the contrary, interpolating subdivision meshes encircle control polyhedra: this can lead to topological degenerations due to surface auto-intersections.

4.1.2 LOOP Subdivision Scheme

The LOOP scheme [16] is dedicated to triangulated meshes, thus it can be applied to arbitrary polygonal meshes, after the mesh is converted into a triangular mesh. The technique corresponds to an approximating vertex insertion scheme. It is based on the three-directional box spline, which produces C^2-continuous surfaces on the regular meshes (the scheme produces surfaces that are C^2-continuous everywhere except at extraordinary vertices, where they are C^1-continuous). Three iterations of the subdivision scheme are shown on Figure 12.

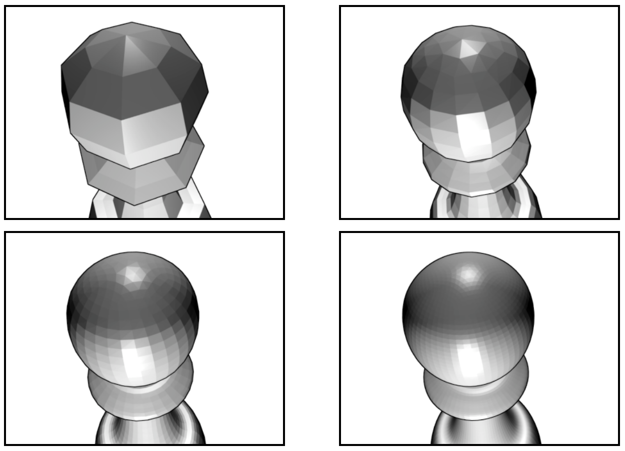

Figure 12: Four iterations of LOOP subdivision scheme.

The principle of LOOP subdivision scheme is the same as for all other approximating schemes. Each triangle is divided into four. Old and new vertices are displaced according to neighbor vertices. Each new location corresponds to the weighted barycenter of neighbors (the weights are set by specific *masks*).

4.2 Using Subdivision Surfaces in Our Frame

In our frame, we settle on LOOP subdivision scheme for three reasons: it is a multiresolution technique to get several levels of detail, it is an approximating scheme and it is dedicated to triangulations.

Approximation mesh ensures that the multiresolution surface is inside the volume of the control polyhedron computed in Section 3. In this way, no topological degenerations can appear on the different LoDs because no surface auto-intersections can happen.

As the generated mesh is only made of triangles, LOOP scheme is the most appropriated subdivision technique.

Thus on the one hand we have a basic polyhedron that is automatically generated from a sketch. Its topology is the same as the skeleton. On the other hand, we can get several levels of detail of a triangulated model by using subdivision surfaces thanks to

LOOP subdivision scheme. With the aim of getting a multiresolution model of arbitrary topology, we combine our mesh construction approach with subdivision surfaces. At high resolution levels, it is then easy to edit details and local features on a sketched model.

5. VALIDATION

In this section, we show an example of shape design, from the sketch to the final subdivision surface, via the generated mesh.

5.1 Example: the Spider with a Hole

We describe step by step the construction of a *spider with a hole*, to illustrate the topology preservation (it is a surface of genus 1).

5.1.1 *Sketching the Spider*

First global features of the shape to design are drawn in a cubic grid, as seen on Figure 13. This is done by using a 3D pointer and by displacing an *active edition plane*.

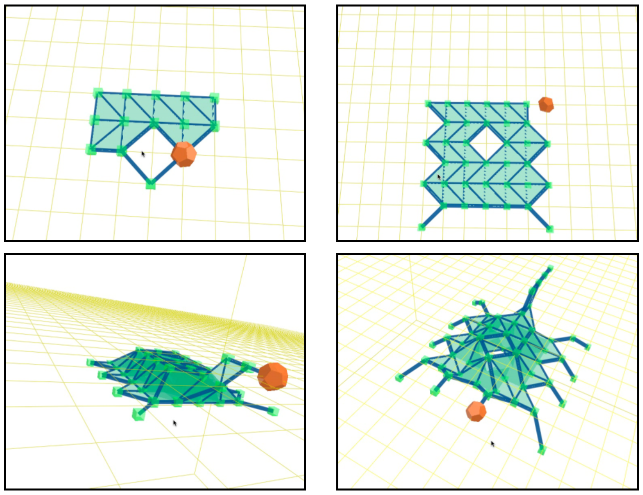
Figure 13: Sketching in a 3D grid.

While editing the vertices of the sketch, the edges and the triangles are constructed to characterize the topology of the object.

5.1.2 *Surrounding Polyhedron*

In a second step, a surrounding polyhedron is automatically generated following the rules defined in Section 3.3. Figure 14 shows the created vertices before (on the left) and after (on the right) the meshing phase.

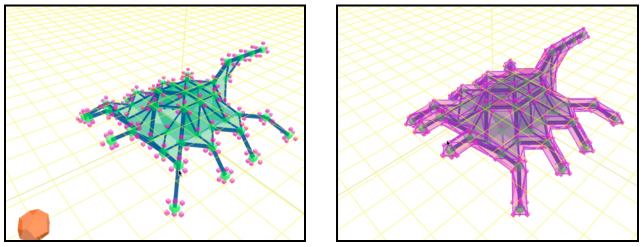
Figure 14: Generation of the mesh.

The resulting polyhedron surrounds the sketch, and the hole of the spider, initially defined by a cycle, is properly present.

5.1.3 *Levels of Detail*

Starting from the surrounding polyhedron of the sketch, levels of detail are generated using LOOP subdivision scheme. From left to right, and from top to bottom, Figure 15 shows the initial control polyhedron M, three levels of subdivision and the superimposition of M and a high level of subdivision. M surrounds the refined mesh, thanks to the approximating scheme.

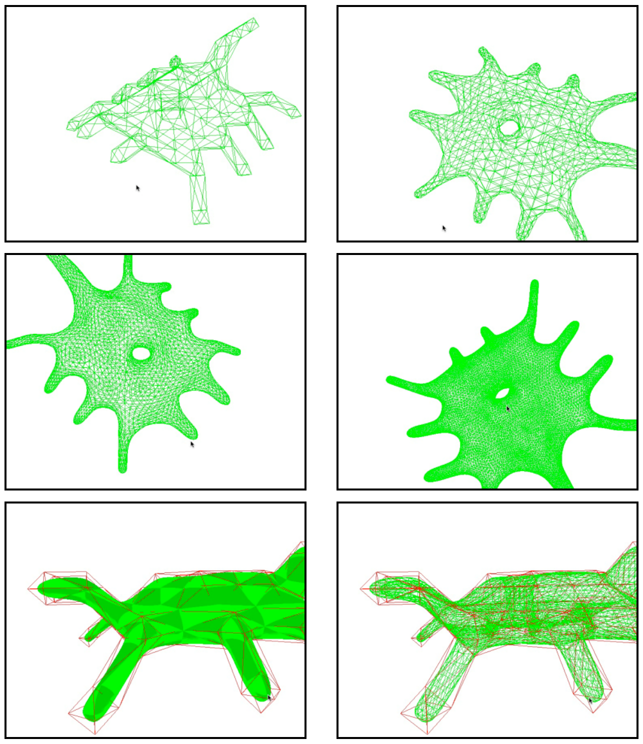
Figure 15: Some subdivision levels.

5.2 Topological Guarantees

At this stage, the topology preservation is guaranteed because:

- the voxels of the sketch are connected according to the 26-adjacency to form the skeleton;
- the surrounding polyhedron preserves holes and cavities of the skeleton, without creating unwanted local connections between close areas of the mesh;
- the levels of detail obtained thanks to the approximating subdivision algorithm are slightly internal to the control polyhedron.

5.3 Further Deformations

On Figure 16, further mesh edition (after two subdivision iterations) is performed using the free software BLENDER [17]. Local and global deformations are done to the spider using the proportional editing tool.

5.4 Other Examples

Figures 17 and 18 illustrate the whole process for two other objects: a table with 4 holes (a surface of genus 4) and a mug plus its bottle-opener (two 1-holed surfaces). For each figure are shown (a) the interactive discrete sketch and skeleton, (b) the generated polyhedron, (c) three levels of subdivision and (d) the final smooth surface.

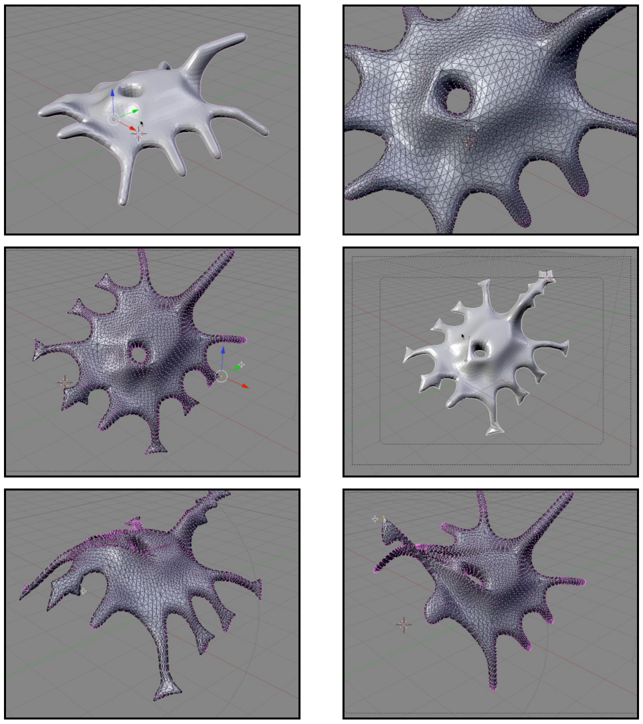

Figure 16: Mesh editing using a 3D modeling software.

6. CONCLUSION AND FUTURE WORK

The approach we propose in this paper consists in generating a mesh automatically from a shape descriptor. The latter is designed interactively in a 3D grid. It gives the global appearance of the object to model. In addition to that, we combine subdivision surfaces with the resulting control polyhedron to get a multiresolution feature. The refined meshes can then be manipulated through a 3D modeling software.

Our two contributions are thus the elaboration of a new method to generate surfaces from a topological mesh and the coupling with subdivision surfaces to obtain several levels of detail of the model.

Future work can be done in two directions. First as the surrounding surface has a multiresolution feature, the sketch could be designed in the same way, by editing large or small *cubes* depending on the size of the global morphological details to characterize. A second perspective is to use the interactive sketch like a skeleton for animation, as *bones* or *armatures* like 3D modelers would do.

7. REFERENCES

[1] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill and G. Wyvill. Introduction to Implicit Surfaces. *Computer Graphics and Geometric Modeling series*, Morgan Kaufmann Publisher, 1997.

[2] J. Bloomenthal and B. Wyvill. Interactives techniques for implicit modeling. *Computer Graphics*, 24 (2), pp 109-116, 1990.

[3] E. Bittar, N. Tsingos and M.-P. Gascuel. Automatic reconstruction of unstructured 3D data: Combining a medial axis and implicit surfaces. *Computer Graphics Forum (Eurographics'95 Proc.)*, 14, pp 457-468, 1995.

[4] B. Wyvill, C. McPheeters and G. Wyvill. Animating soft objects. *The Visual Computer*, 2 (4), pp 235-242, 1986.

[5] A. Sherstyuk. Interactive shape design with convolution surfaces. *Shape Modeling International '99, International Conference on Shape Modeling and Applications*, Aizu-Wakamatsu, Japan, March 1-4, 1999.

[6] A. Alexe, L. Barthe, M.-P. Cani and V. Gaildrat. Shape modelling by sketching using convolution surfaces, *Pacific Graphics (Short Papers)*, Macau, China, October 2005.

[7] E. Ferley, M.-P. Cani and J.-D. Gascuel. Resolution Adaptive Volume Sculpting. *Graphical Models (GMOD), Special Issue on Volume Modelling*, 63, pp 459-478, 2002.

[8] B. De Araujo and J. Jorge. Blobmaker: Free-form modelling with variational implicit surfaces. *Proceedings of "12° Encontro Português de Computação Grafica" (12° EPCG)*, pp 17-26, Porto, Portugal, 2003.

[9] L. Markosian, J. M. Cohen, T. Crulli and J. Hugues. Skin: a constructive approach to modeling free-form shapes. *Computer Graphics Proceedings (SIGGRAPH'99)*, pp 393-400, 1999.

[10] J.-L. Mari and J. Sequeira. A new modeling approach by global and local characterization. In *11th International Conference on Computer Graphics. GraphiCon'2001*, pp 126-131, Nizhny Novgorod, Russia, September 2001.

[11] J.-L. Mari and J. Sequeira. Closed Free-Form Surface Geometrical Modeling – A New Approach with Global and Local Characterization. *International Journal of Image and Graphics (IJIG)*, 4 (2), pp 241-262, World Scientific Publishing, 2004.

[12] M. Cyrus and J. Beck. Generalized Two- and Three-Dimensional Clipping. *Computers and Graphics* 3(1), pp 23-28, 1978.

[13] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design* 10 (6), pp 350 – 355, 1978.

[14] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design* 10 (6), pp 356-360, 1978.

[15] P. Schröder and D. Zorin. Subdivision for modeling and animation. *Course notes of Siggraph 98*. ACM SIGGRAPH, 1998.

[16] C. Loop. Smooth subdivision surfaces based on triangles. University of Utah, Department of Mathematics, 1987.

[17] Blender. Free open source 3D content creation suite, available for all major operating systems under the GNU General Public License. http://www.blender.org/

About the author

Jean-Luc Mari is an Associate Professor at the University of Marseille. His research interests include geometrical modeling, algebraic topology and shape description. His contact address is: Information and System Science Laboratory (LSIS), Computer Graphics Dept. (LXAO), University of Marseille II - ESIL, Campus de Luminy, case 925, 13288 Marseille cedex 9, France. His contact phone number is: +33 4 91 82 85 26. His contact email is mari@univmed.fr. His webpage is http://jean-luc.mari.perso.esil.univmed.fr/.

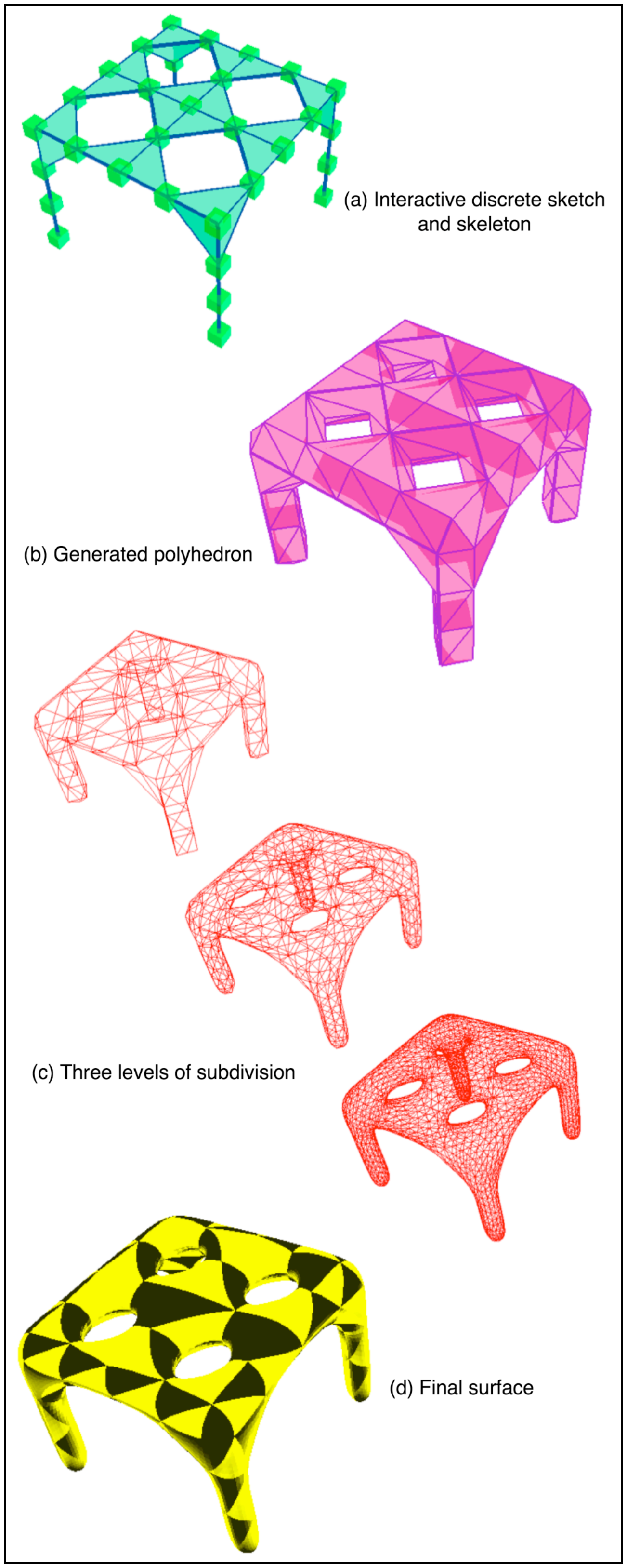
(a) Interactive discrete sketch and skeleton
(b) Generated polyhedron
(c) Three levels of subdivision
(d) Final surface

Figure 17: Mesh generation of a table with 4 holes.

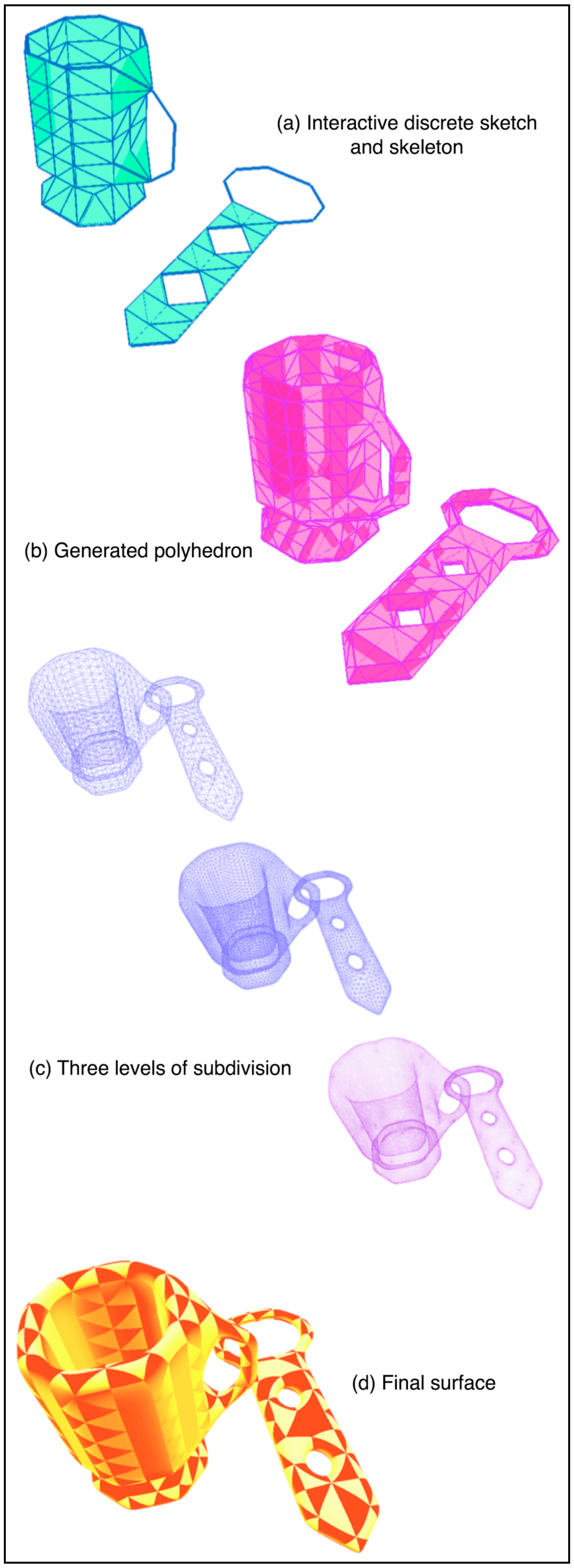
(a) Interactive discrete sketch and skeleton
(b) Generated polyhedron
(c) Three levels of subdivision
(d) Final surface

Figure 18: Mesh generation of a mug and a bottle-opener.