Generating sketch based adaptive meshes

Leonardo Carvalho
impa
Rio de Janeiro, Brazil
leo1984@impa.br

Luiz Velho
impa
Rio de Janeiro, Brazil
lvelho@impa.br

***Abstract*—This work presents a new technique for generating the base mesh for an adaptive mesh generated by a sketch based modeling system. The method uses topology information of the region inside a closed curve to create a good quality base mesh, with controlled vertex valence, and few irregular vertices. The base mesh is refined using an inflation of the region inside the curve to form the final mesh. The user can control how to do the refinement operation including more details where it is necessary.**

***Keywords*-Base mesh generation; Adaptive meshes; Sketch based modeling**

I. Introduction

Sketch-based modeling techniques use information provided by hand-drawn curves in order to make some kind of 3D modeling operation. They can be done in various ways [1]. They can be used as an intuitive tool for CAD systems, very important to the design of 3D models for industry. One of the well known applications of Sketch-based modeling is the generation of a surface whose silhouette is the user-defined curve. Techniques that implement this operation generally represent the surface using a polygonal mesh, which usually is not adaptive. Adaptive meshes are more flexible data structures that provide mechanisms for changing the mesh resolution. The mesh can be more refined in the regions where the user finds appropriate. For example, when one wants to render a smooth surface it is enough to have more details in the region near the mesh silhouette, and the refinement may change as the mesh changes its position relative to the observer.

This work presents an algorithm for generating an adaptive mesh from user-defined curves. It is a simple method, uses a multiresolution structure, generating good quality meshes. The mesh is generated by a subdivision of the polygons of a carefully constructed base mesh.

II. Related work

The generation of surfaces from curves is a very intuitive operation, as one can deduce the geometry of an object just by looking at its silhouette. The seminal work that uses this concept is [2], where a 3D model is constructed and inflated based on the edges of an object in an image.

One of the most significant works in sketch-based modeling is TEDDY [3], where the user can create and edit meshes from hand-drawn curves. As it was one of the first works in this area, there are some problems with the quality of the generated mesh, and the results are not adaptive.

Other works, like FiberMesh [4], improve the generation and editing of the mesh, but still generate non-adaptive structures.

In Gridmesh [5] an already regular base is adjusted to the curve and then inflated. The triangles in this base must be small enough in order to approximate all features of the region defined by the curve, but this forces all the mesh to be more refined just to represent small parts of the shape. In our work, the mesh resolution is adapted to the size of each part of the shape, so there are smaller or bigger polygons where it is necessary, without compromising the quality of the mesh.

In [6] images are used to help the construction of the mesh, and they make some treatments to improve the quality of the mesh. In our method the meshes seem to be better adapted to the shape of the surfaces (comparing the most similar results).

Base meshes can be used in a lot of contexts. Works like [7], [8] discuss base mesh generation from 3D meshes. They use the base meshes mainly to remeshing. The work in [9] presents B-Mesh, a modeling system for generating base meshes of 3D articulated shapes. A base mesh is created from a user-defined skeleton with key balls at its nodes, and it is refined to create the final mesh. This is similar to our work, but in our case the skeleton is inferred by a 2 dimensional user-defined curve that represents the silhouette of the shape, and the inflation of the mesh is based on this curve while they work directly with the skeleton in 3 dimensional space.

The construction of base meshes in the context of sketch based modeling was also used in [10], where the base mesh is used for a Catmull-Clark subdivision surface. They used a similar approach to our work, but with less control on the valence of the irregular vertices and less treatment on size-varying shapes.

There are some mechanisms to work with adaptive structures, like the RGB triangulation [11] and the semi-regular A48 scheme [12], that builds a mesh based on stellar operators. Some properties of the A48 scheme made it convenient to be used in this work, as will be discussed. With this kind of structure it is possible to change adaptively the mesh resolution according to user-defined criteria, always keeping a conforming triangulation even when the resolution is changing. The user just needs to provide an initial mesh, the rules for surface sampling, and a set of adaptation criteria. The quality of the refined mesh, however, depends heavily on the quality of the base mesh.

III. Basic concepts

In the construction of an adaptive mesh, there must be a careful work to certificate that the resulting mesh has a good quality in terms of the shape of the triangles and the regularity of the vertices.

In the A48 scheme one works with a triquad structure, i.e. a triangulated quadrangulation, where each quadrilateral is divided into a pair of triangles. In this case, it is desirable that the quadrilaterals be nearly squares, with smooth variations on the length of the edges.

The refinement in the A48 scheme is done by using a bi-section operator, which divides the diagonal of a quadrilateral. Figure 1 shows this operation applied to one quadrilateral.

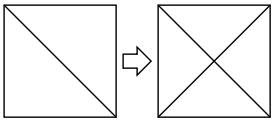
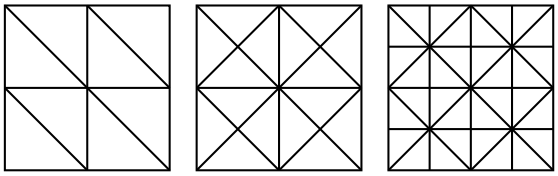


Fig. 1. Refinement operation.

The regular vertices in a quadrangulation are those whose valence is 4. The valence of these vertices can increase up to 8, when the quadrilaterals are triangulated (forming a triquad). The refinement operation includes only regular vertices, and do not increase the valence of old vertices (from last level) more than once. After refining uniformly all the faces, there will be only vertices with valence 4 or 8. This can be seen in Figure 2, where two refinement steps are uniformly done from a triquad structure.

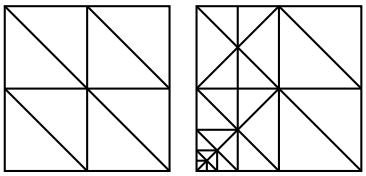


(a) Level 0. (b) Level 1. (c) Level 2.

Fig. 2. Two steps of uniform refinement operation.

Notice that each refinement operation changes the orientation of the quadrilaterals, the edges of old quadrilaterals become the diagonals of the new ones (which may be subdivided in the next refinement step).

The refinement can also be done adaptively according to user-defined criteria. Figure 3 shows an example.

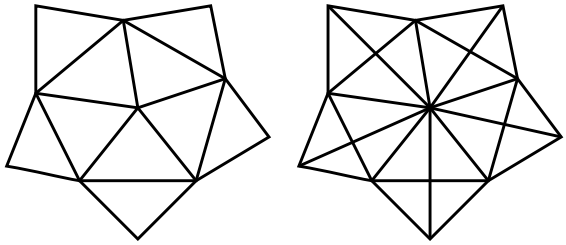


(a) Base mesh. (b) Refined mesh.

Fig. 3. Adaptive refinement.

Even using this irregular refinement, the valence of the vertices is not greater than 8, so the maximum valence in a mesh is controlled.

The maximum valence can be increased when there is an irregular vertex in the base mesh. A vertex whose valence is over 4 (in the base quadrangulation) can have its valence increased to more than 8 when the triquad is created or after a refinement operation. Figure 4 shows an example, the base mesh has a vertex whose valence increases from 5 to 10 after one refinement step.



(a) Base mesh. (b) After refinement.

Fig. 4. The valence of this irregular vertex (in the center) increases from 5 to 10 after refinement.

We want to minimize the number of irregular vertices, and limit their valence to an upper bound. To achieve this goal, it is necessary to create a good base mesh, i.e. a simple mesh that will represent the desired surface in the lowest level of resolution. If this base mesh has good quality triangles, then the final triangulation (after refinement) will also have good quality triangles.

IV. Algorithm

Given a user-defined curve we want to generate a surface whose silhouette is that curve. The generated surface is similar to the one obtained by using TEDDY algorithm [3], where an approximation of the medial axis is calculated, elevated, and joined to the curve using circular arcs.

The input curve is a polygonal closed curve without self-intersections. In the first step of the algorithm, the curve is smoothed and uniformly resampled, the length of the edges after the resampling must be small enough to capture the details of the curve shape.

The CDT (Constrained Delaunay Triangulation) restricted by the edges of the polygonal curve is calculated, it will be used to get an approximation of the medial axis, which provides topological information about the shape of the region inside the curve.

The triangles of the CDT inside the region defined by the curve are then segmented in three types: *sleeve* triangles have only one edge of the polygonal curve; *terminal* triangles have two edges of the curve; and *joint* triangles do not have edges of the curve. An example of this segmentation is shown in Figure 5.

Some triangles in the CDT can be changed to improve the shape of the polygons in the final result. For each edge ab of a joint triangle we get the triangles connecting it to a terminal

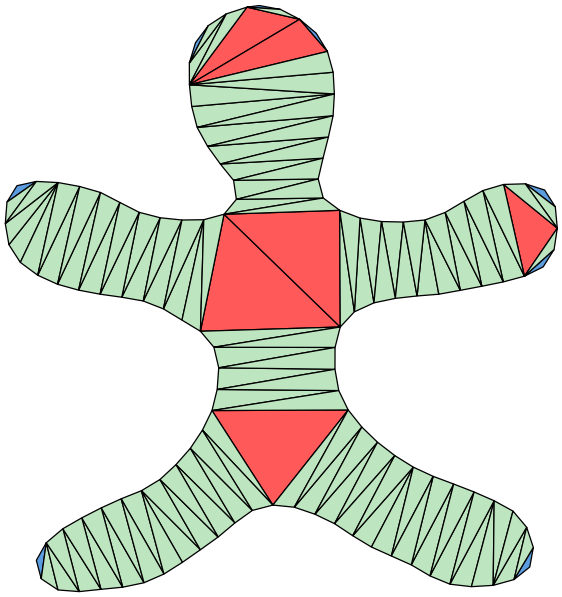


Fig. 5. The CDT with the three types of triangles. The joint triangles are in red; the sleeves in green; the terminals in blue.

triangle. If all the vertices of these triangles are inside the circle centered in the middle of ab and whose diameter is equal to the length of ab, then these triangles (including the joint and the terminal) are rearranged, such that all of them become sleeve triangles. When there is more than one edge satisfying this property in one triangle, the rearrangement is done using the one with less triangles. This operation is exemplified in Figure 6. By doing this, joint and terminal triangles that are too close together are transformed into sleeve triangles. These changings are equivalent to prune small branches of the medial axis.

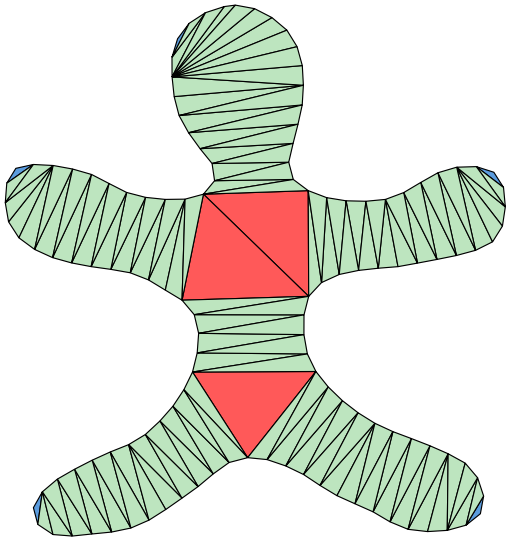


Fig. 6. The improved CDT.

After that, the region inside the curve is divided into polygonal blocks using the CDT triangles as guides.

Quadrilateral blocks are generated from each sequence of sleeve triangles. We start with the edge ab between a sleeve and a non-sleeve triangle. We traverse the triangles in the sequence of sleeves until we find an edge cd such that $0.5(length(ab) + length(cd)) < length(ad) + length(bc)$, so the quadrilateral $abcd$ forms two blocks that are nearly squares. This operation continues from the edge cd, stopping when the sequence of sleeves is over. The last quadrilateral is merged with the one before it, avoiding the formation of very thin blocks.

The vertices of the blocks do not need to be vertices of the polygonal curve, it is possible to move them along the curve in order to improve the shape of the blocks. We create a parametric smooth curve from the polygonal curve using the vertices of the polygonal curve as control points of a closed B-Spline curve. The vertices of the blocks are then adjusted by moving along this smooth curve. Each vertex is located on the curve by its parameter t, we check the shape of the blocks using this vertex, and change t to put the vertex to a position that makes the blocks more similar to squares.

This operation is applied to all vertices and repeated some times (3 times is generally sufficient). Figure 7 shows an example of the blocks generated from a sequence of sleeve triangles.

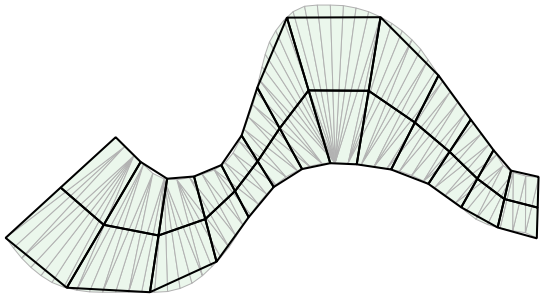


Fig. 7. Blocks generated from a sequence of sleeve triangles.

Each block that is next to a terminal triangle is extended to use the outer vertex (the one shared by the two edges from the input curve in this terminal), as illustrated in Figure 8. By doing this, one of the edges of the block will collapse into one vertex, so each block like this will be a degenerated quadrilateral, but this won't be a problem for the mesh generation, because it can be easily detected and handled in the next steps of the algorithm.

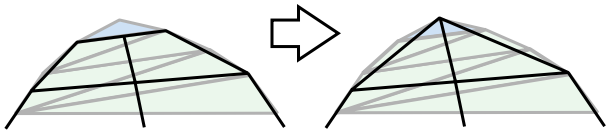
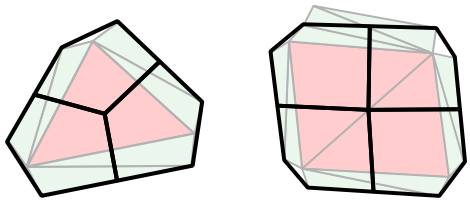


Fig. 8. Blocks next to a terminal triangle have edges collapsed into the outer vertex.

Each joint triangle forms three blocks, using edges connecting the midpoints of its edges to the triangle barycenter. These

blocks are extended, becoming pentagonal (see Figure 9(a)). This extension is an important mechanism to control the valence of the vertices.

When two joint triangles are connected by an edge, they are treated as only one big quadrilateral, so four blocks are created, instead of 6 that would be made normally, this simplifies the base mesh, and reduces the number of irregular vertices that will be formed. Figure 9(b) shows this case. When there are more than two joint triangles connected, this coupling operation is made using pairs of triangles whose shapes are closer to squares.



(a) Blocks at a joint. (b) Blocks at 2 joints.

Fig. 9. Blocks from joint triangles.

Figure 10 shows with thick lines one example of the blocks created by using the described rules.

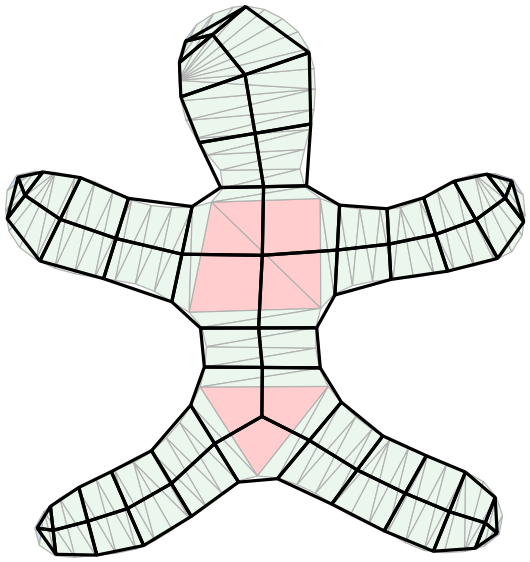


Fig. 10. Blocks.

The vertices on the approximated axis are elevated according to their distances to the curve, which are calculated as the length of the edges connecting them to the curve. These blocks are then copied and mirrored in the z-direction, forming a closed mesh.

For the construction of the base mesh, it is necessary the creation of a triquad structure from these blocks. A way of doing this without generating too many faces and high valence vertices is by the addition of one vertex at the center of each block, and edges connecting this vertex to each block vertex. This way each edge of the blocks will be a diagonal of a quadrilateral.

Figure 11 shows an example of a base mesh generated this way, the edges of quadrilaterals can be seen with thick lines, the dotted lines are the diagonals of the quadrilaterals (the block edges).

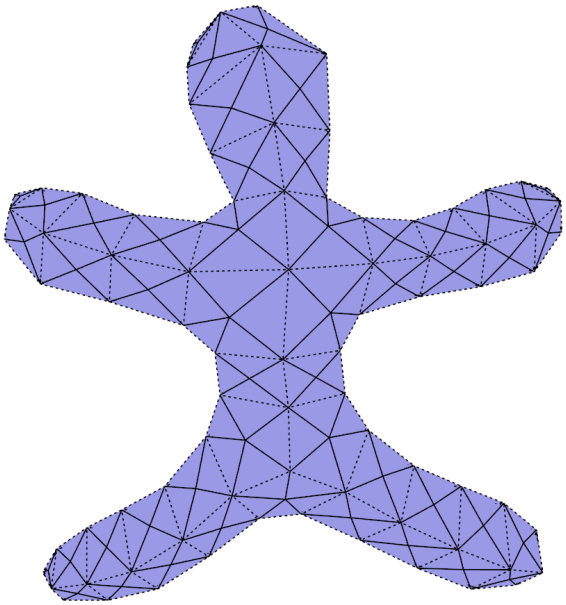


Fig. 11. The base mesh.

The only irregular vertices in this base mesh are created from the triangular blocks, where there are vertices of valence 3 (will be 6 after one refinement step), and from pentagonal blocks, where there are vertices of valence 5 (10 after one refinement step), and vertices of valence 6 when only three pentagonal blocks are connected. All other vertices will be regular (valence 4 or 8 in the triangular mesh).

Now we need to define how to inflate the surface that will form the shape of the mesh. This can be done in several ways. One option is to use parametrical surface patches. We can form a patch from a block by connecting the curve to the approximated axis using circular arcs. Figure 12 shows an example of a surface patch from a quadrilateral block.

The triangular blocks will create surface patches with a singular point located at the position of the point corresponding to the collapsed edge of the degenerated quadrilateral block. This singular point can be easily handled during the subdivision process. Figure 13 illustrates this kind of surface patch.

Each patch is easily parameterized, just using a parametrization of the input curve (the B-Spline), the axis (calculated as an average of two parts of the input curve), and an arc profile (a quarter of a circle) The union of these patches forms the surface which defines the geometry of our mesh.

The vertices of the base mesh that are on the curve or on the axis are already on this surface. Each one of the other vertices can be repositioned on the patch originated from the block containing this vertex, using parameters in the middle of the patch.

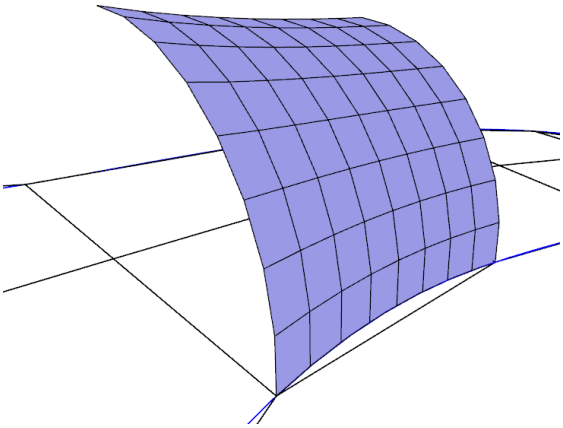


Fig. 12. Surface patch.

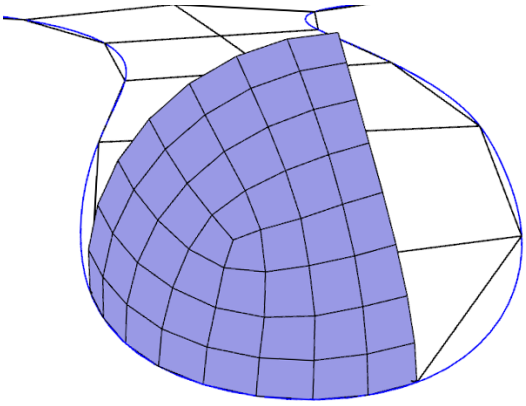


Fig. 13. Surface patch from a block using a terminal triangle.

The surface generated by this method is not smooth everywhere (it is not smooth between blocks generated by joint triangles), but it is good enough for our purposes, because we are more interested in the quality of the polygons in the generated mesh.

For the refinement operation we need to specify how to divide an edge of the diagonal of a quadrilateral in the triquad structure. To divide an edge, we can simply calculate the average of the parameters of its extreme vertices as the parameters of a point on the patch, this point is then used as a new vertex of the mesh.

After the mesh is created and subdivided, the shape of the polygons can still be improved by using a smoothing operation on the vertices. A very effective way of doing this is using an intrinsic Laplacian mesh smoothing [13].

V. Results

The method discussed here was implemented in C++. For the calculation of the CDT we used the Triangle library [14]. All the examples were hand-drawn.

On Figure 14 we can see the mesh of Figure 11 after 3 uniform subdivision steps.

Because we used a uniform subdivision, all the faces are in the same level, so we can show the mesh using only the quadrilaterals of the triquad, which is better for visualization of the mesh.

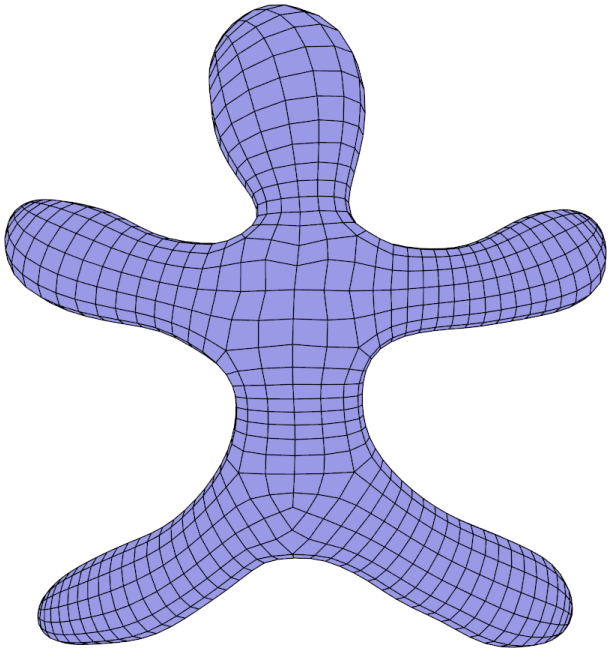


Fig. 14. Example mesh.

Another example, this one representing a frog is shown in Figure 15.

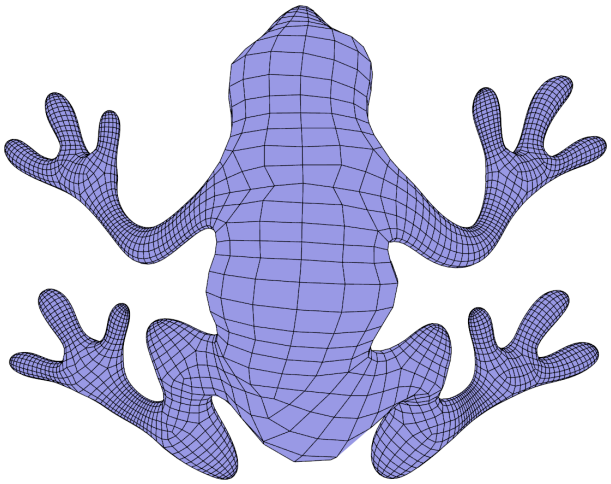


Fig. 15. Frog, 3 subdivision steps.

There is a big variation in the size of each piece of this surface, it is possible to see that the generated mesh follows this variation, with small quadrilaterals in narrow areas, as can be seen in a detail in Figure 16. It is also possible to see that the directions of the edges mostly follow the principal directions of the shape.

As we are using an adaptive structure, it is possible to use a non-uniform subdivision, for example we can subdivide according to the length of the edges. One can define adaptation rules to simplify a vertex of each edge whose length is less

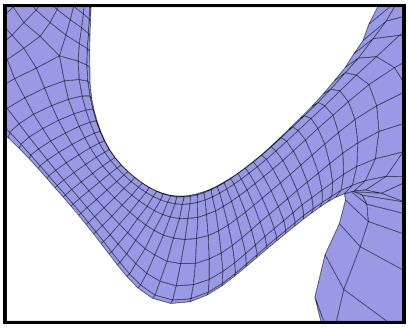


Fig. 16. Detail of a leg of the frog.

than a value a, and to refine every edge whose length is bigger then a fixed value b. This can be seen in Figure 17.

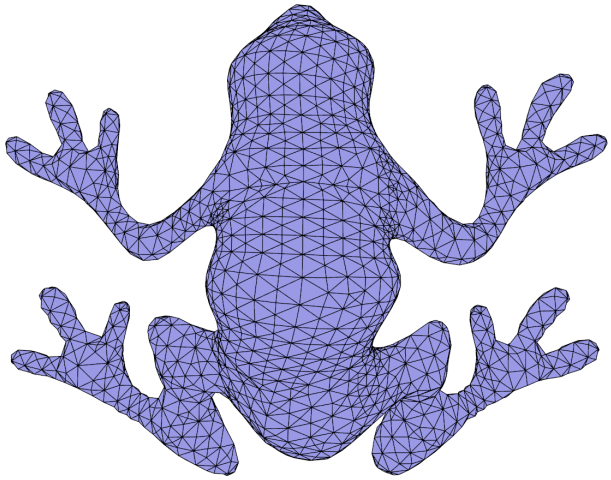


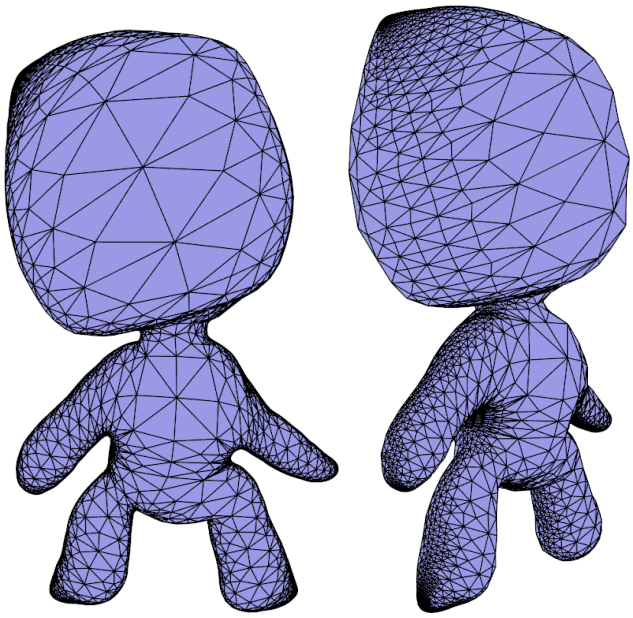
Fig. 17. Using adaptive subdivision.

Another way to adaptively refine the mesh is to subdivide more in silhouette areas than in internal regions, as can be seen in Figure 18. The adaptation rules in this case use target levels for vertices and edges, subdividing edges that are bellow their target levels, and simplifying vertices that are above their levels. The target level of a vertex is defined based on the angle between its normal and the viewer direction. For each edge, we use the angle between the direction from its midpoint to the viewer and the average of the normals of the two vertices. Using a bigger target level when this angle is near to 90°, one can clearly see the silhouette of the mesh without having to subdivide all the triangles uniformly.

As other examples we have a mesh representing an ant in Figure 19, and a gecko in Figure 20, both using uniform subdivision after three levels of subdivisions. One can see the edges follow the principal directions of each object, except in regions where there is an asymmetry in the drawing, like between the forelegs (in both figures).

VI. Conclusion

In this work we discussed a new way of creating the base mesh to be used in an adaptive structure in a sketch-based modeling system.



(a) Front view. (b) Side view.

Fig. 18. Using view dependent subdivision.

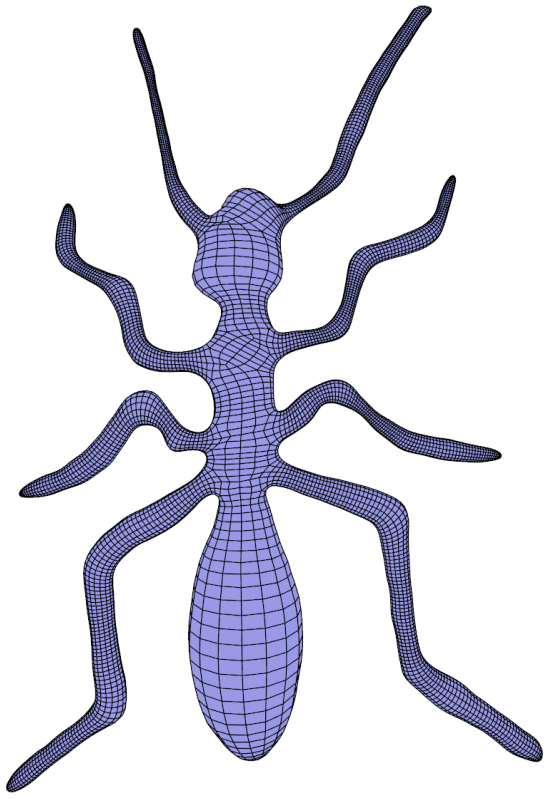


Fig. 19. Ant.

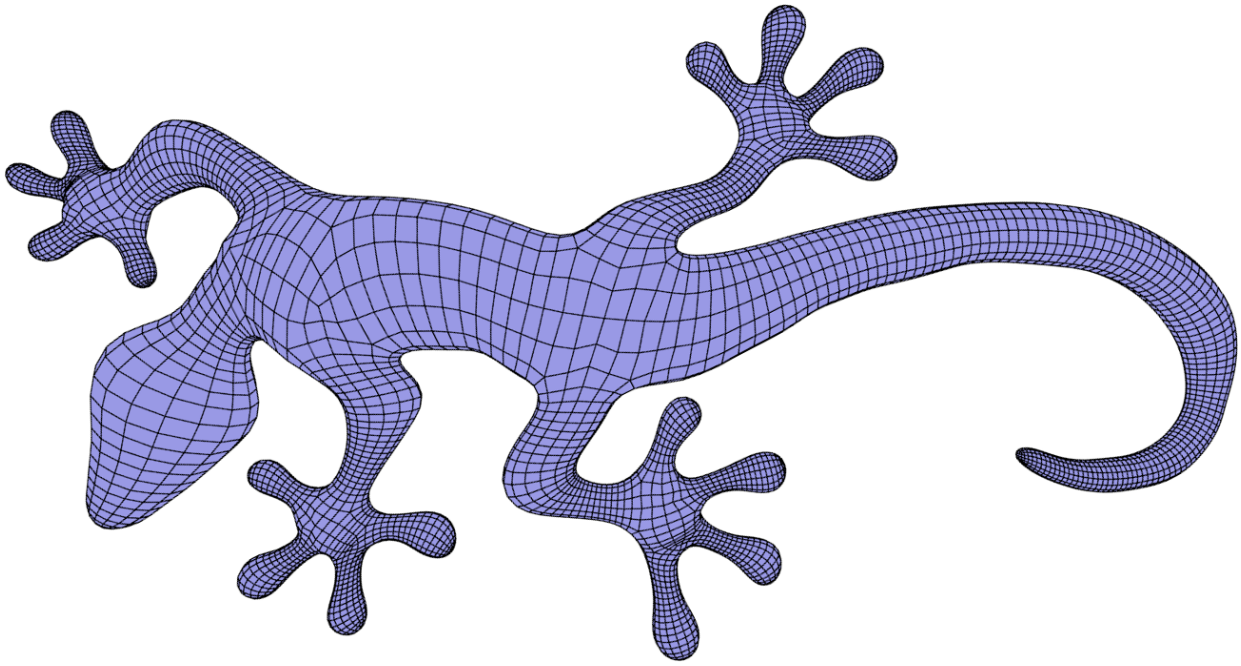


Fig. 20. Gecko.

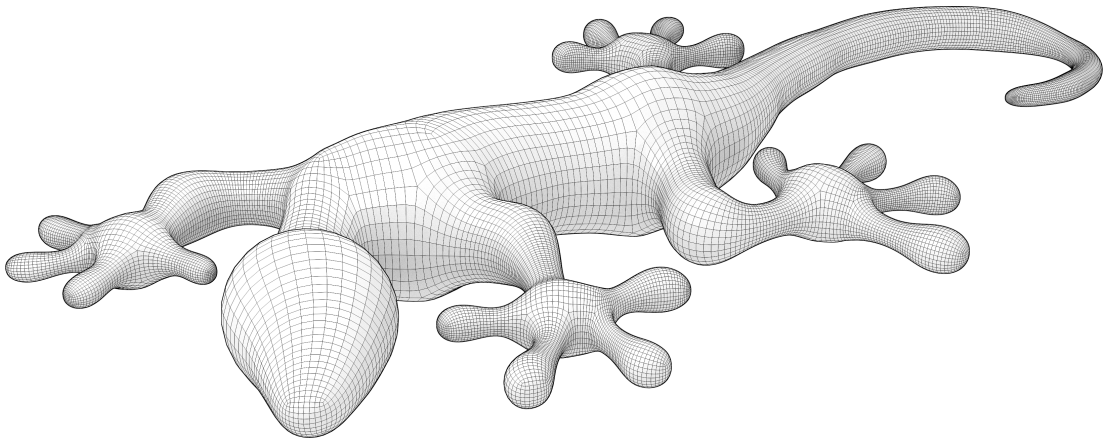


Fig. 21. Gecko using our method and Catmull-Clark subdivision.

The created mesh has a good quality in terms of the shape of triangles, the edges follow mostly the main directions of the surfaces, and there is a control over the valence of the vertices.

A limitation of the method is that some asymmetries in the drawing can affect the resulting mesh, so better results depend on the quality of the input curve.

This work can still be improved by implementing some editing operations on the mesh, as cutting, extrusion and deformation, like the ones used in TEDDY [3] and Fibermesh [4]. We can also use different inflation methods to improve the shape of the surface, in Figure 21 we applied two levels of uniform refinement as described above, and 3 levels of the Catmull-Clark subdvision method [15] adapted to work with the triquad mesh, resulting in a smooth surface with our high quality mesh.

REFERENCES

[1] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 85–103, Feb 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYG-4TPF4GK-2/2/bc4d1162dc5b79a3887fd969b736dd3e

[2] D. Terzopoulos, "On matching deformable models to images," in *Optical Society of America, Topical Meeting on Machine Vision*, 1980, pp. 160–163.

[3] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design," in *SIGGRAPH99*, ser. CGPACS. New York: ACM Press/ACM SIGGRAPH, 1999, pp. 409–416.

[4] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "FiberMesh: Designing freeform surfaces with 3D curves," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, vol. 26, no. 3, p. article no. 41, 2007.

[5] A. Nealen, J. Pett, M. Alexa, and T. Igarashi, "Gridmesh: Fast and high quality 2d mesh generation for interactive 3d shape modeling."

[6] L. Olsen and F. F. Samavati, "Image-assisted modeling from sketches," in *Proceedings of the Graphics Interface 2010 (GI'10)*, May 31-Jun 2 2010.

[7] J. Daniels, II, M. Lizier, M. Siqueira, C. T. Silva, and L. G. Nonato, "Smi 2011: Full paper: Template-based quadrilateral meshing," *Comput. Graph.*, vol. 35, pp. 471–482, June 2011. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2011.03.024

[8] N. Pietroni, M. Tarini, and P. Cignoni, "Almost isometric mesh parameterization through abstract domains," *IEEE Transaction on Visualization and Computer Graphics*, vol. 16, no. 4, pp. 621–635, July/August 2010. [Online]. Available: http://vcg.isti.cnr.it/Publications/2010/PTC10

[9] Z. Ji, L. Liu, and Y. Wang, "B-mesh: A modeling system for base meshes of 3d articulated shapes," *Computer Graphics Forum (Proceedings of Pacific Graphics)*, vol. 29, no. 7, pp. 2169–2178, 2010.

[10] A. Nasri, W. B. Karam, and F. Samavati, "Sketch-based subdivision models," in *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM'09)*. New York, NY, USA: ACM, Aug 1-2 2009, pp. 53–60.

[11] E. Puppo, "Selectively refinable subdivision meshes," in *Symposium on Geometry Processing*, 2006, pp. 153–162.

[12] L. Velho, "A dynamic adaptive mesh library based on stellar operators," *journal of graphics, gpu, and game tools*, vol. 9, no. 2, pp. 21–47, 2004.

[13] F. de Goes, F. P. G. Bergo, A. X. Falcao, and S. Goldenstein, "Adapted dynamic meshes for deformable surfaces," *Graphics, Patterns and Images, SIBGRAPI Conference on*, vol. 0, pp. 213–220, 2006.

[14] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," in *Applied Computational Geometry: Towards Geometric Engineering*, ser. Lecture Notes in Computer Science, M. C. Lin and D. Manocha, Eds. Springer-Verlag, May 1996, vol. 1148, pp. 203–222, from the First ACM Workshop on Applied Computational Geometry.

[15] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," *Computer-aided Design*, vol. 10, pp. 350–355, 1978.