Lecture Notes in Computer Science, ISSN 3037, pp. 412-418, Springer Verlag, 2004

Surface Curvature Estimation for Edge Spinning Algorithm*

Martin Cermak and Vaclav Skala

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Czech Republic
{cermakm|skala}@kiv.zcu.cz

Abstract. This paper presents an adaptive modification of the Edge spinning method for polygonization of implicit surfaces. The method insists on the shape of triangles and the accuracy of resulting approximation as well. The main advantages of the triangulation presented are simplicity and the stable features that can be used for next expanding. The implementation is not complicated and only the standard data structures are used. The presented algorithm is based on the surface tracking scheme and it is compared with the other algorithms which are based on the similar principle, such as the Marching cubes and the Marching triangles algorithms.

1 Introduction

Implicit surfaces seem to be one of the most appealing concepts for building complex shapes and surfaces. They have become widely used in several applications in computer graphics and visualization.

An implicit surface is mathematically defined as a set of points in space **x** that satisfy the equation $f(\mathbf{x}) = 0$. Thus, visualizing implicit surfaces typically consists in finding the zero set of *f*, which may be performed either by polygonizing the surface or by direct ray tracing. There are two different definitions of implicit surfaces. The first one [2], [3] defines an implicit object as $f(\mathbf{x}) < 0$ and the second one, F-rep [8], [10], [11], [12], defines it as $f(x) \geq 0$. In our implementation, we use the F-rep definition of implicit objects.

Existing polygonization techniques may be classified into three categories. Spatial sampling techniques that regularly or adaptively sample the space to find the cells that straddle the implicit surface [2], [4]. Surface tracking approaches (also known as continuation methods) iteratively create a triangulation from a seed element by marching along the surface [1], [2], [5], [7], [9], [15]. Surface fitting techniques progressively adapt and deform an initial mesh to converge to the implicit surface, [10].

* This work was supported by the Ministry of Education of the Czech Republic - project MSM 235200002

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 412–418, 2004.

Lecture Notes in Computer Science, ISSN 3037, pp. 412-418, Springer Verlag, 2004

2 Principle of the Edge Spinning Algorithm

Our algorithm is based on the surface tracking scheme (also known as the continuation scheme, see Fig. 1) and therefore, there are several limitations. A starting point must be determined and only one separated implicit surface is polygonized for such point. Several disjoint surfaces can be polygonized from a starting point for each of them.

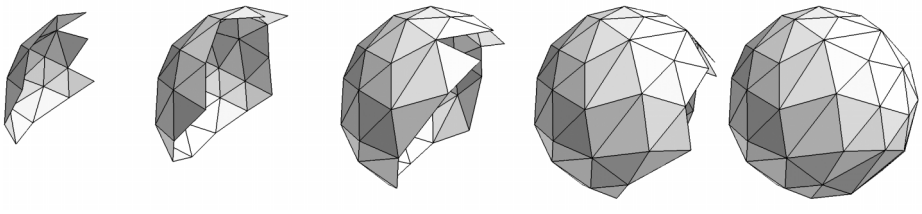

Fig. 1. Continuation scheme, new triangles are directly generated on an implicit surface.

The algorithm uses only the standard data structures used in computer graphics. The main data structure is an edge that is used as a basic building block for polygonization. If a triangle's edge lies on the triangulation border, it is contained in the *active edges list* (*AEL*) and it is called as an *active edge*. Each point, which is contained in an active edge, contains two pointers to its left and right active edge (left and right directions are in active edges' orientation).

The whole algorithm consists of the following steps (more detailed description in [5]):

1. Initialize the polygonization:
   a. Find the starting point $\mathbf{p}_0$ and create the first triangle T_0.
   b. Include the edges $(e_0,e_1,e_2,)$ of the first triangle T_0 into the active edges list.
2. Polygonize the first active edge *e* from the active edges list.
3. Update the *AEL*; delete the currently polygonized active edge *e* and include the new generated active edge/s at the end of the list.
4. If the active edges list is not empty return to step 2.

3 Edge Spinning

The main goal of this work is a numerical stability of a surface point coordinates' computation for objects defined by implicit functions. In general, a surface vertex position is searched in direction of a gradient vector $\nabla\mathbf{f}$ of an implicit function *f*, e.g. as in [7]. In many cases, the computation of gradient of the function *f* is influenced by a major error that depends on modeling techniques used [8], [9], [10], [11], [13], [14]. Because of these reasons, in our approach, we have defined these restrictions for finding a new surface point $\mathbf{p}_{new}$:

Lecture Notes in Computer Science, ISSN 3037, pp. 412-418, Springer Verlag, 2004

- The new point $\mathbf{p}_{new}$ is sought on a circle; therefore, each new generated triangle preserves the desired accuracy of polygonization. The circle radius is proportional to the estimated surface curvature.
- The circle lies in the plane that is defined by the normal vector of triangle T_{old} and axis *o* of the current edge *e*, see Fig. 3; this guarantees that the new generated triangle is well shaped (isosceles).

3.1 Determination of the Circle Radius

The circle radius is proportional to the estimated surface curvature. The surface curvature in front of current active edge is determined in according to angle α between the surface normals $\mathbf{n}_1$, $\mathbf{n}_2$, see Fig. 2. The normal vector $\mathbf{n}_1$ is computed at point **S** that lies in the middle of the current active edge *e* and the vector $\mathbf{n}_2$ is taken at initial point $\mathbf{p}_{init}$ that is a point of intersection of the circle c_1 with the plane defined by the triangle T_{old}.

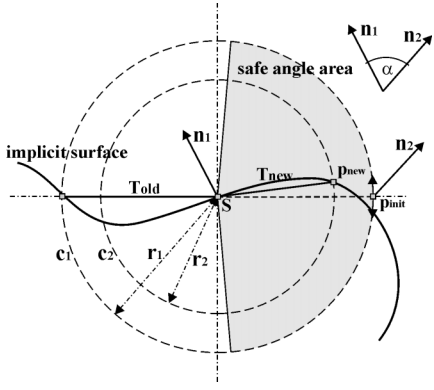

Fig. 2. The circle radius estimation.

Note that the initial radius r_1 of the circle c_1 is always the same and it is set at beginning of polygonization as the lowest desired level of detail (LOD).
The new circle radius r_2 is computed as follows.

$$r_2 = r_1 \cdot k, k \in \langle 0,1 \rangle; \qquad k = \left(\frac{\alpha_{\lim} - \alpha \cdot c}{\alpha_{\lim}} \right), \tag{1}$$

where α_{lim} is a limit angle and the constant *c* represents a speed of "shrinking" of the radius according to the angle α. To preserve well shaped triangles, we use a constant k_{min} that represents a minimal multiplier. In our implementation we used $\alpha_{min} = \pi/2$, $k_{min} = 0.2$ and $c = 1.2$.

Correction notes:

if $(\alpha > \alpha_{min})$ then $k = k_{min}$
if $(k < k_{min})$ then $k = k_{min}$

These parameters affect a shape of triangles of the polygonal mesh generated.

Lecture Notes in Computer Science, ISSN 3037, pp. 412-418, Springer Verlag, 2004

3.2 Root Finding

If the algorithm knows the circle radius, the process continues as follows.

1. Set the point $\mathbf{p}_{new}$ to its initial position; the initial position is on the triangle's T_{old} plane on the other side of the edge *e*, see Fig. 3. Let the angle of the initial position be $\alpha=0$.

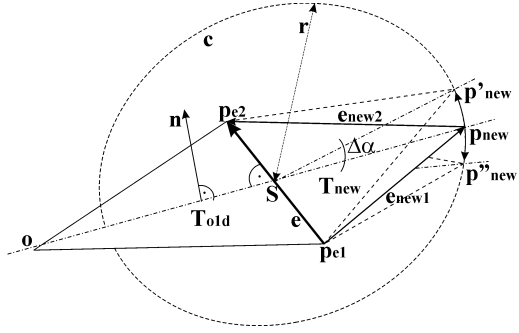

Fig. 3. The principle of root finding algorithm.

2. Compute the function values $f(\mathbf{p}_{new}) = f(\alpha)$,
   $f(\mathbf{p'}_{new}) = f(\alpha + \Delta\alpha)$ – initial position rotated by the angle $+\Delta\alpha$,
   $f(p''_{new}) = f(\alpha - \Delta\alpha)$ - initial position rotated by the angle $-\Delta\alpha$; Note that the rotation axis is the edge *e*.
3. Determine the right direction of rotation; if $|f(\alpha + \Delta\alpha)| < |f(\alpha)|$ then $+\Delta\alpha$ else $-\Delta\alpha$.
4. Let the function values $f_1 = f(\alpha)$ and $f_2 = f(\alpha \pm \Delta\alpha)$; update the angle $\alpha = \alpha \pm \Delta\alpha$.
5. Check which of following case appeared:
   a) If $(f_1 \cdot f_2)<0$ then compute the accurate coordinates of the new point $\mathbf{p}_{new}$ by the binary subdivision between the last two points which correspond to the function values f_1 and f_2;
   b) If the angle $|\alpha|$ is less than α_{safe} (see *safe angle area* in Fig. 2) return to step 4.
   c) If the angle $|\alpha|$ is greater than α_{safe} then there is a possibility that both triangles T_{old} and T_{new} could cross each other; the point $\mathbf{p}_{new}$ is rejected and it is marked as *not found*.

4 Experimental Results

The Edge spinning algorithm (ES) is based on the surface tracking scheme and therefore, we have compared it with other methods based on the same principle – the Marching triangles algorithm (MTR, introduced in [7]) and the Marching cubes method (MC, Bloomenthal's polygonizer, introduced in [2]).
As a testing function, we have chosen the implicit object Genus 3 that is defined as follows.

Lecture Notes in Computer Science, ISSN 3037, pp. 412-418, Springer Verlag, 2004

$$f(\mathbf{x}) = r_z^4 \cdot z^2 - \left[1 - (x/r_x)^2 - (y/r_y)^2\right] \cdot \left[(x - x_1)^2 + y^2 - r_1^2\right] \cdot \left[(x + x_1)^2 + y^2 - r_1^2\right] = 0$$

where the parameters are: $\mathbf{x} = [x,y,z]^T$, r_x=6, r_y=3.5, r_z=4, r_1=1.2, x_1=3.9.

The values in Table 1 have been achieved with the desired lowest level of detail (LOD) equal 0.8. It means that maximal length of triangles' edges is 0.8. Note that there is not defined a unit of length, so that number could be for example in centimeters as well as the parameters of the function Genus 3 described above.

Table 1. Values of the object Genus 3 with the lowest level of detail LOD = 0.8

	ES	MTR	MC
# Triangles	4886	947	1056
# Vertices	2439	473	516
Avg dev.	10,99	56,80	73,28
Angle crit.	0,65	0,67	0,38
Elength crit.	0,77	0,78	0,54

The table contains the number of triangles and vertices generated. The value *Avg dev.* means the average deviation of each triangle from the real implicit surface. It is measured as an algebraic distance of a gravity centre of a triangle from an implicit surface, i.e. the function value at the centre of gravity of the triangle. Note that the algebraic distance strongly depends on the concrete implicit function; in our test, the Genus 3 object is used for all methods, so the value has its usefulness. The value *Angle crit.* means the criterion of the ratio of the smallest angle to the largest angle in a triangle and the value *Elength crit.* means the criterion of the ratio of the shortest edge to the longest edge of a triangle. The value *Avg dev.* shows the accuracy of an implicit object approximation and the adaptive ES algorithm is logically the best of tested methods. The criterions of angles and length of edges in triangles are similar for the ES and the MTR algorithms, so the both approaches generate well-shaped triangular meshes.

For visual comparison, the resulting pictures of the Genus 3 object generated in the test are in figures below. Fig. 4a shows the object generated by the adaptive algorithm, so the number of triangles generated is higher in dependence on the surface curvature. In Fig. 4b, some parts of the object are lost because the algorithm just connect nearest parts by large triangles depending of the lowest level of detail. The resulting image generated by the Marching cubes algorithm is shown in Fig. 4c. This algorithm produces badly-shaped triangles but it is fast and also stable for complex implicit surfaces with C^0 continuity, only.

Lecture Notes in Computer Science, ISSN 3037, pp. 412-418, Springer Verlag, 2004

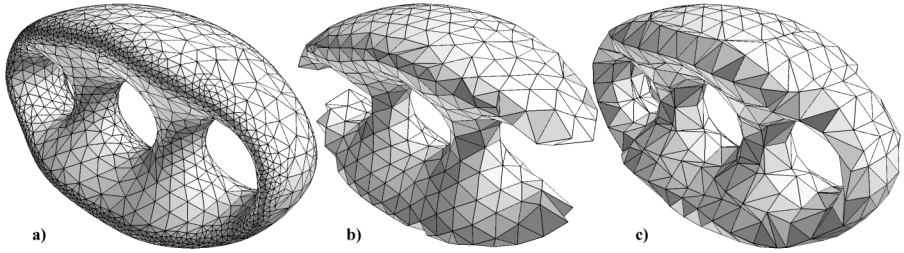

Fig. 4. The Genus 3 object generated by the a) Adaptive Edge spinning algorithm; b) Marching triangles algorithm; c) Marching cubes algorithm.

Fig. 5 shows the object modeled as intersection of two spheres. The left picture is polygonized without detection of sharp edges, and the right picture is polygonized with the edge detection principle applied to the ES method, see [6]. This object complies only the C^0 continuity and it is correctly polygonized by our method.

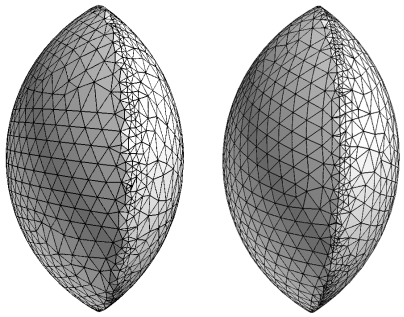

Fig. 5. Intersection of two spheres generated by the Adaptive Edge spinning algorithm

5 Conclusion

This paper presents the new adaptive approach for polygonization of implicit surfaces. The algorithm marches over the object's surface and computes the accurate coordinates of new points by spinning the edges of already generated triangles. Coordinates of the new points depend on surface curvature estimation. We used the estimation by deviation of angles of adjacent points because it is simple and fast for computation. The similar measurement has been used as curvature estimation in [16] as well. Our experiments also proved its functionality.

The algorithm can polygonize implicit surfaces which comply C^1 continuity, thin objects and some non-complex objects of C^0 continuity (an object should have only sharp edges, no sharp corners or more complex shapes). In future work, we want to modify the current algorithm for more complex implicit functions of the C^0 continuity, only.

Acknowledgement. The authors of this paper would like to thank all those who contributed to development of this new approach, especially to colleagues MSc. and PhD. students at the University of West Bohemia in Plzen.

References

1. Akkouche, S., Galin, E.: Adaptive Implicit Surface Polygonization using Marching Triangles, Computer Graphic Forum, 20(2): 67-80, 2001.
2. Bloomenthal, J.: Graphics Gems IV, Academic Press, 1994.
3. Bloomenthal, J.: Skeletal Design of Natural Forms, Ph.D. Thesis, 1995.
4. Bloomenthal, J., Bajaj, Ch., Blinn, J., Cani-Gascuel, M-P., Rockwood, A., Wyvill, B., Wyvill, G.: Introduction to implicit surfaces, Morgan Kaufmann, 1997.
5. Čermák, M., Skala, V.: Polygonization by the Edge Spinning, 16th Conference on Scientific Computing, Algoritmy 2002, Slovakia, ISBN 80-227-1750-9, September 8-13.
6. Čermák, M., Skala, V.: Detection of Sharp Edges during Polygonization of Implicit Surfaces by the Edge Spinning. Geometry and graphics in teaching contemporary engineer, Szczyrk 2003, Poland, June 12-14.
7. Hartmann, E.: A Marching Method for the Triangulation of Surfaces, The Visual Computer (14), pp.95-108, 1998.
8. "Hyperfun: Language for F-Rep Geometric Modeling", http://cis.k.hosei.ac.jp/~F-rep/
9. Karkanis, T., Stewart, A.J.: Curvature-Dependent Triangulation of Implicit Surfaces, IEEE Computer Graphics and Applications, Volume 21, Issue 2, March 2001.
10. Ohtake, Y., Belyaev, A., Pasko, A.: Dynamic Mesh Optimization for Polygonized Implicit Surfaces with Sharp Features, The Visual Computer, 2002.
11. Pasko, A., Adzhiev, V., Karakov, M., Savchenko,V.: Hybrid system architecture for volume modeling, Computer & Graphics 24 (67-68), 2000.
12. Rvachov, A.M.: Definition of R-functions, http://www.mit.edu/~maratr/rvachev/p1.htm
13. Shapiro, V., Tsukanov, I.: Implicit Functions with Guaranteed Differential Properties, Solid Modeling, Ann Arbor, Michigan, 1999.
14. Taubin, G.: Distance Approximations for Rasterizing Implicit Curves, ACM Transactions on Graphics, January 1994.
15. Triquet, F., Meseure, F., Chaillou, Ch.: Fast Polygonization of Implicit Surfaces, WSCG'2001 Int.Conf., pp. 162, University of West Bohemia in Pilsen, 2001.
16. Velho,L.: Simple and Efficient Polygonization of Implicit Surfaces, Journal of Graphics Tools, 1(2):5-25, 1996.