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.

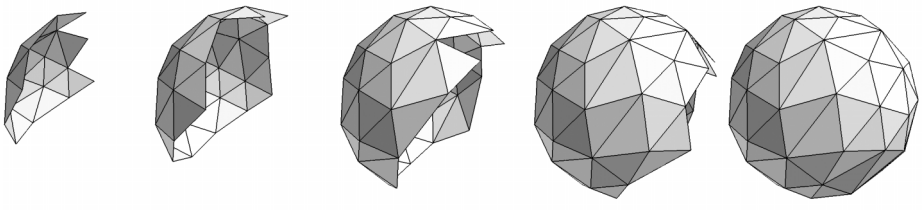

Fig. 1. Continuation scheme, new triangles are directly generated on an implicit surface.

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}.

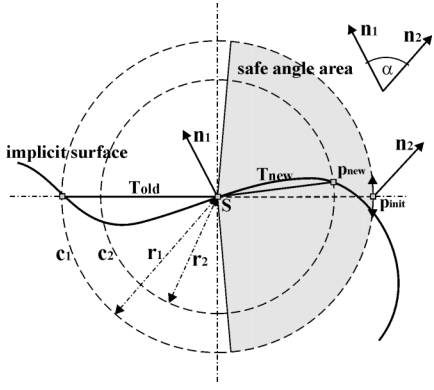

Fig. 2. The circle radius estimation.

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.