



## Technical section

## Subdivision interpolating implicit surfaces

Xiaogang Jin<sup>a,\*</sup>, Hanqiu Sun<sup>b</sup>, Qunsheng Peng<sup>a</sup><sup>a</sup>State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, People's Republic of China<sup>b</sup>Department of Computer Science and Engineering, CUHK, Hong Kong

---

**Abstract**

Interpolating implicit surfaces using radial basis functions can directly specify surface points and surface normals with closed form solutions, so they are elegantly used in surface reconstruction and shape morphing. This paper presents subdivision interpolating implicit surfaces, a new progressive subdivision tessellation scheme for interpolating implicit surfaces controlled by a triangular mesh with arbitrary topology. We use a recursive polyhedral subdivision scheme to subdivide the control triangular mesh, and the new generated vertices are mapped to the implicit surface using Newton iteration. A multiresolution surface representation is automatically built with the proposed approach. Based on this approach, a newly surface modeling tool with more flexible control is developed by blending the interpolating subdivision surfaces with the subdivision interpolating implicit surfaces.

© 2003 Elsevier Ltd. All rights reserved.

*Keywords:* Interpolating implicit surface; Tessellation; Radial basis function; Multiresolution

---

**1. Introduction**

Smooth and deformable surfaces, especially those enclosing a volume, are usually difficult and inefficient to model and represent using traditional parametric surfaces. An increasingly popular modeling approach in recent years is the field-based implicit surfaces [1–6]. An implicit surface is defined as a locus in space that satisfies an equation  $f(x, y, z) = T$ , where  $T$  is the threshold of an iso-surface. The implicit representations make them convenient for modeling and animating complex smooth objects, such as liquids [1], clouds [3], plants, sea-life forms, and other organic shapes [5,6]. In addition to object modeling, implicit surfaces have also been widely used in other applications including constrained space deformations [4], surface reconstruction [7], and shape morphing [8]. In the entertainment industry, implicit surfaces have become an important tool for creating visually striking special effects.

Implicit surfaces based on radial basis functions, which are relatively new in implicit surface modeling, have recently attracted attention because they can easily represent complex shapes with arbitrary topology. Savchenko et al. first proposed an interpolating implicit surface based on radial basis functions to reconstruct solids from scattered surface points and contours [7]. Later, Turk et al. used an interpolating implicit surface as a shape transformation tool [8], both shape interpolation and feature specification are elegantly incorporated in one concise scattered data interpolation model. In addition to shape blending and surface reconstruction [8–10], interpolating implicit surfaces are also gaining acceptance in computer assisted surgical planning [11], complex surface modeling [12], mesh repairing [13] and sketching [14].

There are mainly two approaches for rendering interpolating implicit surfaces, *ray-tracing* and *polygonization*. Ray-tracing incorporates reflection, transparency, refraction and shadows concisely in one illumination model, and is a good method for generating high quality images. Turk et al. use an implicit surface ray-tracing method called sphere tracing, introduced by

---

\*Corresponding author. Tel.: +86-571-879-51045; fax: +86-571-879-51780.

E-mail address: jin@cad.zju.edu.cn (X. Jin).

Hart [15], to produce ray-traced images. The key idea of sphere tracing is to march along the ray toward the intersection point in steps small enough not to penetrate the implicit surface. However, as the interpolating implicit surface is usually complex, ray-tracing has the limitation of low speed because of the large computational cost involved in the ray/surface intersections.

Another indirect way to extract the iso-surface of the interpolating implicit surface is to convert the surface into polygons. This approach is popular as most of the graphics acceleration cards and commercial animation software packages support high performance polygon-based rendering, hence, it is a convenient way to incorporate implicit surface modeling into existing graphics systems. Bloomenthal's continuation polygonizer [16,17], which produces better results than the marching-cubes algorithm [18], is a well-known iso-surface extraction algorithm. This technique first partitions the space enclosed by the implicit surface using an octree, then samples the implicit function at the octree cube corners, and finally calculates and connects the surface vertices to form the resulting polygon mesh using a look-up table. However, for rendering the interpolating implicit surface with a triangular mesh as input, both the continuation method and the marching-cubes algorithm have the following limitations. (1) The fact that the triangular mesh is already a coarse polygonal approximation of the implicit surface is not exploited. (2) Marching-cubes type methods may produce thin and elongated triangles as illustrated in Fig. 1(a), which will lead to poor shading effects. (3) A surface region of high curvature is liable to be under-sampled.

The proposed approach also tessellates the interpolating implicit surface into polygons, but it differs from the marching-cubes or surface following methods in that it uses a polyhedral subdivision scheme as a polygon

conversion tool. In our approach we try to make full use of the interpolating vertices and the connectivity information provided by the input triangular mesh. The control mesh is recursively subdivided to a desired level using a polyhedral subdivision scheme. As the new added vertices usually do not lie on the interpolating implicit surface, we map them to the implicit surface using Newton's iteration method. By iteratively moving each added vertex along the gradient direction, we arrive at a point that lies on the implicit surface and take this point as the correspondence point or projection point of the added vertex. As the tessellated surface is quite similar to the interpolating subdivision surface [19,20], we call the resulting surface as *subdivision interpolating implicit surface*. This method can produce the same fine scalable triangles as the interpolating subdivision surface (see Fig. 1(b)), and users can choose a coarse or detailed mesh according to the requirements of their applications. Experimental results show that our method generates better aspect ratio triangles than the continuation method.

We assume that the input is a triangular mesh with normal information at each vertex. If there is no normal information in the input model, we estimate the normal of a vertex by averaging the normals of its neighboring triangles. Non-triangular meshes can be easily converted into triangular ones by triangulation. Based on the information in the input polygonal model, an interpolating implicit surface is created using radial basis functions. The resulting tessellated surface is then obtained by employing a 1-to-4 subdivision scheme. The rest of the paper is organized as follows. In Section 2, we introduce the radial basis function based interpolating implicit surfaces. In Section 3, we present our subdivision scheme and mapping. The blending between the interpolating subdivision surface and the subdivision interpolating implicit surface is described in Section 4. In Section 5, we present our results.

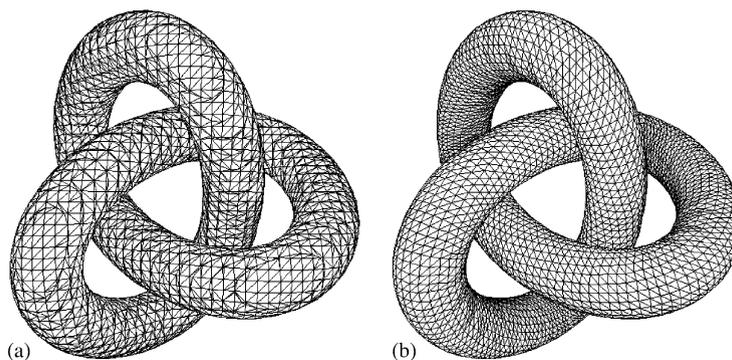


Fig. 1. The tessellation of the implicit surface that interpolates a knot: (a) continuation method generates thin and elongated triangles and (b) subdivision tessellation.

Discussions and conclusions are presented in Section 6 and Section 7, respectively.

**2. Interpolating implicit surface based on radial basis functions**

An interpolating implicit surface is a generalization of a thin-plate spline surface for scattered data interpolation. In this paper, we will adopt the notations presented by Turk et al. [12]. The scattered data interpolation problem in 3D can be stated as follows: given  $n$  distinct points  $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_n\}$  in  $R^3$  and their associated real values  $\{h_1, h_2, \dots, h_n\}$ , find a smooth unknown function  $f(\mathbf{r})$  such that  $f(\mathbf{c}_i) = h_i, i = 1, 2, \dots, n$ . The smoothness of  $f(\mathbf{r})$  is measured by minimizing the energy functional

$$E(f) = \int_{R^3} \left| \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} + 2 \frac{\partial^2 f}{\partial x \partial y} + 2 \frac{\partial^2 f}{\partial x \partial z} + 2 \frac{\partial^2 f}{\partial y \partial z} \right| dx dy dz. \tag{1}$$

The above energy minimization problem can be solved using radial basis functions, a well-established mathematical tool for solving scattered data interpolation problems. The analytical interpolant can be expressed as

$$f(\mathbf{r}) = \sum_{j=1}^n \omega_j \phi(\mathbf{r} - \mathbf{c}_j) + P(\mathbf{r}), \tag{2}$$

where  $\omega_j$  are real-valued weights,  $\phi$  are radial basis functions, and  $P(\mathbf{r})$  is a one-degree polynomial accounting for the linear and constant portions of  $f$  [12]. Choosing  $P(\mathbf{r})$  as a linear polynomial can ensure that the resulting surface is affine invariant to the input scattered points.

Let  $\mathbf{c}_i = (c_i^x, c_i^y, c_i^z)$ , then the unknown weights  $\omega_j$  and polynomial coefficients  $p_0, p_1, p_2, p_3$  can be determined by the interpolation conditions [13]

$$f(\mathbf{c}_i) = \sum_{j=1}^n \omega_j \phi(\mathbf{c}_i - \mathbf{c}_j) + P(\mathbf{c}_i), \quad i = 1, 2, \dots, n \tag{3}$$

and the orthogonality conditions [13]

$$\sum_{j=1}^n \omega_j = \sum_{j=1}^n \omega_j c_j^x = \sum_{j=1}^n \omega_j c_j^y = \sum_{j=1}^n \omega_j c_j^z = 0. \tag{4}$$

Let  $\phi_{ij} = \phi(\mathbf{c}_i - \mathbf{c}_j), \mathbf{r} = (x, y, z), P(\mathbf{r}) = p_0 + p_1x + p_2y + p_3z$ , then Eqs. (3) and (4) will lead to the

following linear system:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{2n} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} & 1 & c_n^x & c_n^y & c_n^z \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \dots & c_n^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_n^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_n^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5}$$

The matrix of the system is symmetric and positive semi-definite, and the unique solution can be solved by direct LU decomposition.

Based on various applications, the radial basis functions can be chosen as thin-plate spline, Gaussian and multiquadric [13]. In this paper, we adopt triharmonic spline  $\phi(\mathbf{r}) = |\mathbf{r}|^3$  for 3D interpolation as in [12]. The interpolant then becomes:

$$f(x, y, z) = \sum_{j=1}^n \omega_j \left( \sqrt{(x - c_j^x)^2 + (y - c_j^y)^2 + (z - c_j^z)^2} \right)^3 + p_0 + p_1x + p_2y + p_3z. \tag{6}$$

The interpolating implicit surface can be created using the concept of 3D scattered data interpolation. We use two types of constraints: boundary constraints and normal constraints, as introduced by Turk et al. [12]. The vertices of the mesh form a set of boundary constraints where the implicit function takes on zero, and the normals form a set of normal constraints where the function takes on one. The level set of the implicit surface forms a surface interpolating the vertices of the input triangular mesh. As the resulting surface is obtained through a variational interpolation problem, Turk and O'Brien also call the surface a variational implicit surface.

**3. Subdivision scheme and mapping**

To obtain the polygonized mesh of the interpolating implicit surface, we use a polyhedral subdivision scheme to subdivide each triangle of the control triangular mesh recursively. By adding a new vertex at the midpoint of each edge, each edge is divided in two, and each triangle of the control net is subdivided into four smaller triangles. This scheme is shown in Fig. 2. However, as the new added vertices usually do not lie on the interpolating implicit surface, we must find a way to map them onto the corresponding points in the implicit

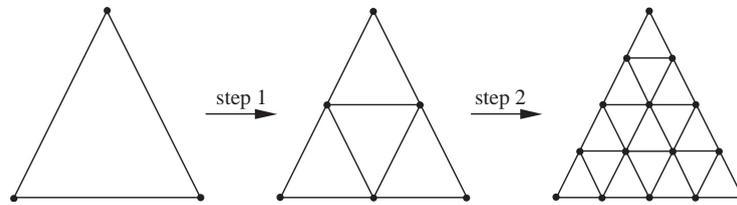


Fig. 2. Subdivision scheme for a triangle.

surface. Our solution is to follow a new added vertex along its gradient until it reaches a point on the implicit surface, and take this point as the mapped point of the vertex. This procedure is performed by Newton’s iteration method, as used by several researchers [21,22]. The gradient of the function  $f(x, y, z)$  is  $\nabla f = (\partial f / \partial x, \partial f / \partial y, \partial f / \partial z)$ , where

$$\frac{\partial f}{\partial x} = 3 \sum_{j=1}^n \omega_j (x - c_j^x) \left( \sqrt{(x - c_j^x)^2 + (y - c_j^y)^2 + (z - c_j^z)^2} \right) + p_1, \tag{7}$$

$$\frac{\partial f}{\partial y} = 3 \sum_{j=1}^n \omega_j (y - c_j^y) \left( \sqrt{(x - c_j^x)^2 + (y - c_j^y)^2 + (z - c_j^z)^2} \right) + p_2, \tag{8}$$

$$\frac{\partial f}{\partial z} = 3 \sum_{j=1}^n \omega_j (z - c_j^z) \left( \sqrt{(x - c_j^x)^2 + (y - c_j^y)^2 + (z - c_j^z)^2} \right) + p_3. \tag{9}$$

The mapped point on implicit surface for a new added vertex  $\mathbf{r}_0$  can then be calculated iteratively by

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \frac{f(\mathbf{r}_k)}{\|\nabla f(\mathbf{r}_k)\|^2} \nabla f(\mathbf{r}_k). \tag{10}$$

The above iteration continues until  $\|\mathbf{r}_{k+1} - \mathbf{r}_k\|$  is less than a specified tolerance  $\epsilon$ . The iteration procedure can be illustrated in 2D, in this case the subdivision scheme subdivides each edge into two by adding the midpoint as a new vertex. In Fig. 3,  $\mathbf{c}_1$  and  $\mathbf{c}_2$  are two vertices,  $\mathbf{r}_0$  is the midpoint of  $\mathbf{c}_1\mathbf{c}_2$ . At the first subdivision level,  $\mathbf{r}_0$  is mapped to  $\mathbf{c}$ , which lies on the implicit curve.

The new vertices can be mapped onto the implicit surface in two ways. The first way is to subdivide the control mesh to the desired level first before any vertex mapping, then map all the newly generated vertices onto the interpolating implicit surface. The second way is to perform the mapping after each level of the subdivision. For the latter case, it is obvious that all the vertices of the new control mesh after each subdivision level lie on the interpolating implicit surface, and therefore the vertex mapping procedure can be performed faster. Let  $\mathbf{s}_1$  and  $\mathbf{s}_2$  be the midpoints of  $\mathbf{c}_1\mathbf{c}$  and  $\mathbf{c}\mathbf{c}_2$ ,  $\mathbf{q}_1$  and  $\mathbf{q}_2$  be the midpoints of  $\mathbf{c}_1\mathbf{r}_0$  and  $\mathbf{r}_0\mathbf{c}_2$ , respectively. As

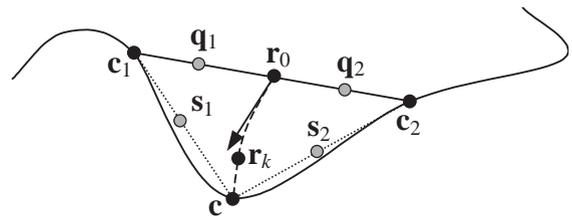


Fig. 3. Illustration of the trajectory that maps a new added vertex to the implicit surface.

illustrated in Fig. 3, vertices  $\mathbf{s}_1$  and  $\mathbf{s}_2$  are closer to the implicit curve than vertices  $\mathbf{q}_1$  and  $\mathbf{q}_2$ , and hence they converge faster to the curve than  $\mathbf{q}_1$  and  $\mathbf{q}_2$  in the mapping iteration. Another observation reveals that the second way generates more uniform triangles than the first one. Hence, we chose the second way in our implementation as it can produce faster and better result.

Let  $m$  be the number of triangles in the input triangular mesh. After the first level of subdivision, the original mesh is replaced by a new triangular mesh with  $4m$  triangles. The new mesh is a refined approximation of the interpolating implicit surface. In Fig. 4, the input mesh is an icosahedron with 20 triangles. By solving the scattered data interpolation problem, we obtain a sphere-like shaped implicit surface. We generate a new mesh with 80 triangles after the level 1 subdivision. Repeating the subdivision procedure, we can in turn generate progressively refined meshes with 320, 1280, 5120, and 20,480 triangles. The subdivision procedure automatically builds a multiresolution model of the interpolating implicit surface, and can be used in time-critical rendering.

#### 4. Blending the interpolating subdivision surface with the subdivision interpolating implicit surface

Subdivision surfaces that can retain the original vertices are very useful to the game community, as this makes it possible to display geometric models to suit the hardware capabilities. Dyn et al. first proposed an

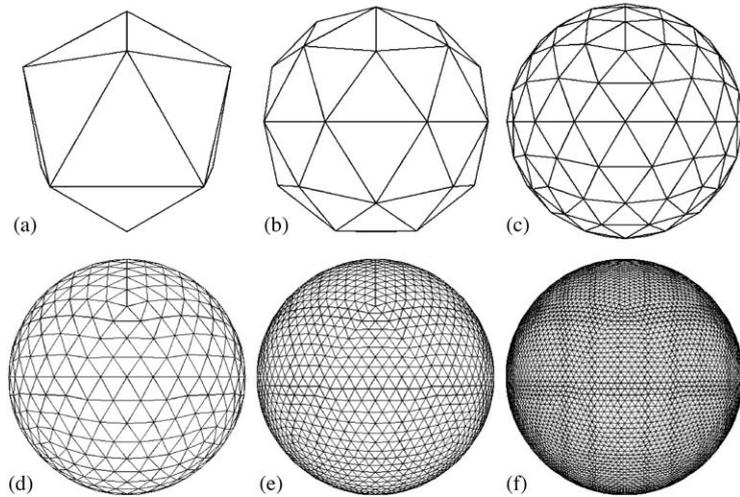


Fig. 4. Subdivision of the implicit surface interpolating an icosahedron: (a) level 0 (input mesh); (b) level 1; (c) level 2; (d) level 3; (e) level 4; (f) level 5.

interpolating subdivision surface model based on a butterfly scheme for a regular topology [19]. This scheme was later improved by Zorin to accommodate topologically irregular triangular meshes [20]. Zorin’s improved scheme can deal with vertices with degree different than 6 and boundary vertices nicely. The interpolating subdivision surface is especially useful for 3D character modeling in 3D games. The two main stencils for the modified butterfly scheme are shown in Fig. 5. For the 10-point stencil, the weights for the vertices are:

$$a:\frac{1}{2} - w, \quad b:\frac{1}{8} + 2w, \quad c:-\frac{1}{16} - w, \quad d:w,$$

where  $w$  is a global tension control parameter.

As the subdivision interpolating implicit surface and the interpolating subdivision surface share the same subdivision scheme, we can blend them to provide a new surface modeling tool. Let  $f$  and  $g$  be the subdivision interpolating implicit surface and the interpolating subdivision surface with the same subdivision levels, respectively. The vertex correspondences for these two surfaces have been automatically established as they have the same vertex/edge/triangle network, thus we can blend them

$$s = (1 - \lambda)f \oplus \lambda g, \quad 0 \leq \lambda \leq 1$$

to create a new geometric model. Using this model, we can use the parameter set

$$\Omega = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_n, \mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_n, w, \lambda\}$$

to control the resulting shape, where  $\mathbf{n}_i$  is the normal at point  $\mathbf{c}_i$ ,  $i = 1, 2, \dots, n$ . Because the energy functional is minimized, the interpolating implicit surfaces are everywhere smooth [12]. For appropriate values of  $w$ , the

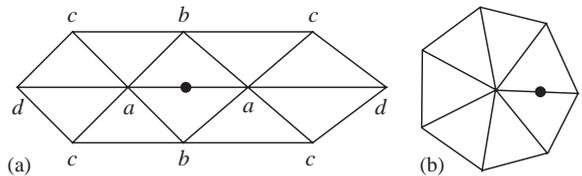


Fig. 5. The stencils from the modified butterfly scheme: (a) 10-points stencil, (b) stencil for a vertex in the 1-neighborhood of an extraordinary vertex.

interpolating subdivision surface is  $C^1$  smooth. Therefore, the surface  $s$  is also  $C^1$  smooth if  $w$  is appropriately chosen.

Fig. 6 shows the blending between the interpolating subdivision surface and the subdivision interpolating implicit surface. The control mesh is a triangulated cube (see Fig. 6(a)), Fig. 6(b) is the interpolating subdivision surface with parameter  $w = 0$ , and Fig. 6(d) is the interpolating implicit surface. Fig. 6(c) is the blended surface with blending parameter  $\lambda = 0.5$ .

### 5. Experimental results

We have implemented our algorithm on an Intel Pentium IV 2.0 GHz PC with 512M main memory under the Windows 2000 operating system. The algorithm consists of the following steps:

- Read the triangular mesh.
- Construct the linear system using the boundary and normal constraints.

- Solve the linear system and compute the analytical solution of the variational surface.
- Progressively apply the subdivision scheme to obtain the tessellated mesh.

Figs. 7–9 are examples generated by the proposed algorithm with hidden line removal. In each figure, on the left is the original triangular mesh, in the middle is the result after one subdivision step, and on the right is

the result after two subdivision steps. These examples indicate that our approach can generate progressively uniform triangular meshes.

Figs. 10–15 illustrate the shading results of our approach and Bloomenthal's polygonizer. Flat shading and Phong shading are used to demonstrate the shading effects in each example. For Phong shading, the vertex normal is calculated by normal averaging. As our algorithm generates more uniform triangles, it produces smoother shading results.

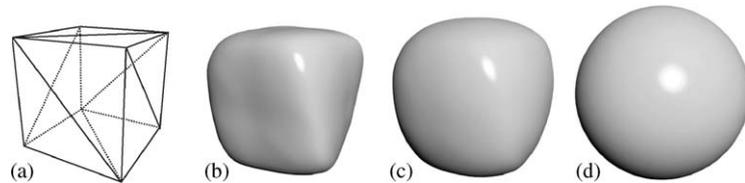


Fig. 6. Blending between the two interpolating surfaces: (a) control mesh, (b) interpolating subdivision surface, (c) blended surface, and (d) variational surface.

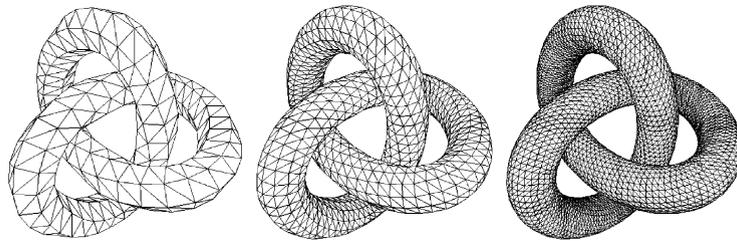


Fig. 7. Subdivision of the implicit surface interpolating a knot.

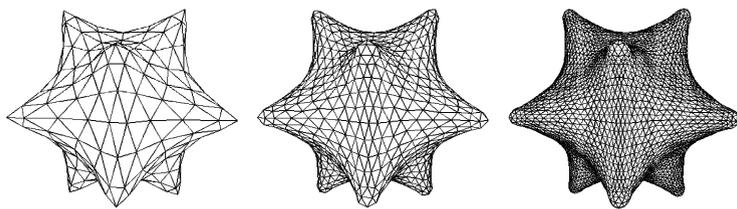


Fig. 8. Subdivision of the implicit surface interpolating a star-shaped mesh.

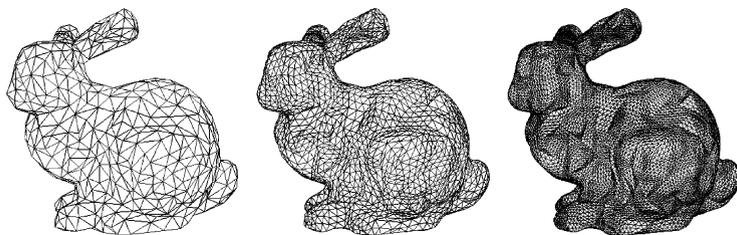


Fig. 9. Subdivision of the implicit surface interpolating a bunny.

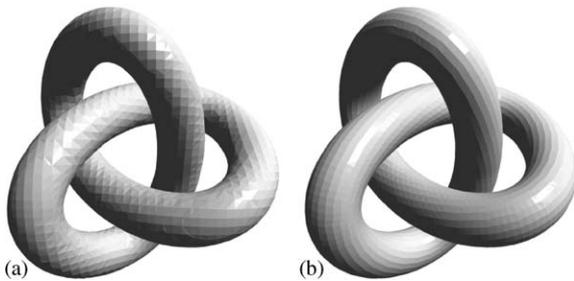


Fig. 10. Flat shading for the knot model: (a) continuation polygonizer and (b) subdivision tessellation.

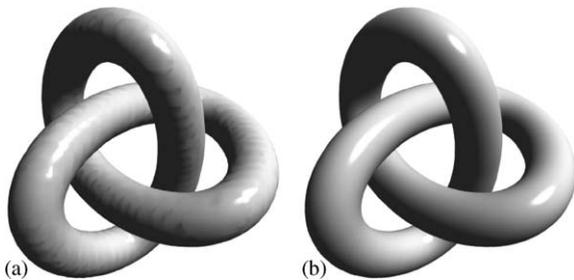


Fig. 11. Phong shading for the knot model: (a) continuation polygonizer and (b) subdivision tessellation.

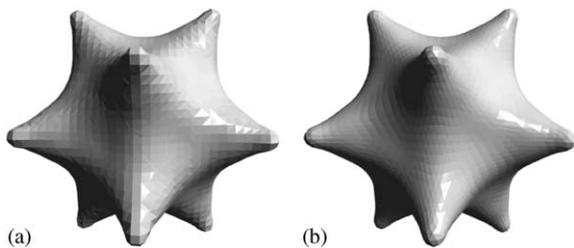


Fig. 12. Flat shading for the star model: (a) continuation polygonizer and (b) subdivision tessellation.

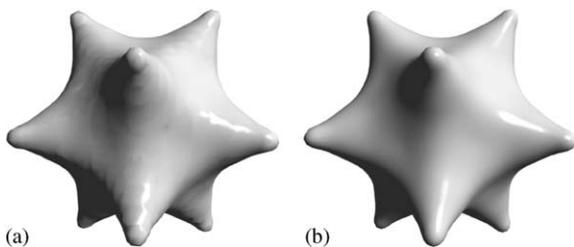


Fig. 13. Phong shading for the star model: (a) continuation polygonizer and (b) subdivision tessellation.

To compare the computational efficiency between Bloomenthal's tessellation method and our subdivision scheme, we interactively adjust the width of the partitioning cube in Bloomenthal's polygonizer until it

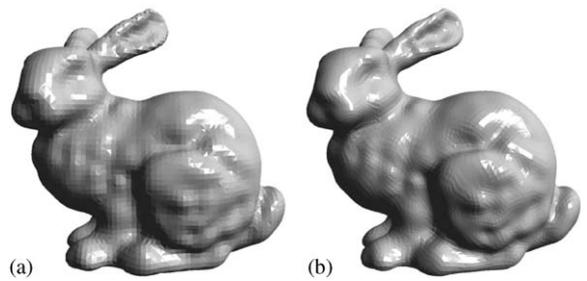


Fig. 14. Flat shading for the bunny model: (a) continuation polygonizer and (b) subdivision tessellation.

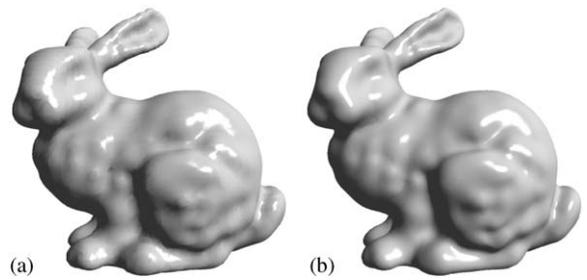


Fig. 15. Phong shading for the bunny model: (a) continuation polygonizer and (b) subdivision tessellation.

generates almost the same number of triangles with our subdivision method. The number of generated triangles and running time for the knot, the star and the bunny models are listed in Table 1. For the subdivision tessellation, the running time  $0.70 + 2.94 = 3.64$  stands for 0.70 s for the first-level subdivision, 2.94 s for the second-level subdivision, and 3.64 s in total. The statistics show our approach has a better performance than Bloomenthal's polygonizer. This is so for two reasons. First, the fact that the original triangular mesh is a coarse sample of the interpolating implicit surface is fully exploited in our algorithm. Second, the convergence speed to map a vertex to the surface is getting faster with each subdivision step, as the refined mesh is a closer approximation of the interpolating implicit surface.

The marching method proposed by Hartman produces coherent triangles [21], and the curvature-dependent triangulation method proposed by Karkanis et al. generates close-to-equilateral triangles with sizes dependent on the local surface curvature [22]. Our approach generates the same high-quality triangles as the interpolating subdivision scheme, because they both adopt the same subdivision scheme. If the triangles of the input mesh have good aspect ratios, our method can generate very good coherent triangles, this can be verified in the examples shown in Figs. 4, 7 and 8. However, if the triangles of the input mesh have very poor aspect ratios, the methods by Hartman and Karkanis et al. may

Table 1  
Comparison of running time

Model	Bloomenthal's polygonizer		Subdivision tessellation	
	Number of triangles	Running time (s)	Number of triangles	Running time (s)
Knot	11,528	13.84	11,520	0.70 + 2.94 = 3.64
Star	8644	8.56	8640	0.34 + 1.66 = 2.00
Bunny	25,048	66.75	24,928	3.02 + 13.25 = 16.27

produce better results than our method. In any case, our approach still has the advantage of a simple data structure, fast tessellation, and automatic level of detail representations. The curvature-dependent triangulation method is typically 20 times slower than continuation method [22], while our method is nearly 4 times faster than continuation method.

Whether the input mesh is open or closed, the interpolating implicit surface will always produce a smooth closed surface. This feature makes interpolating implicit surfaces useful for mesh repairing [13]. With the subdivision scheme, it becomes easy to show the interior or part of the implicit surface, not only the entire surface. To display a surface partially, we first edit the surface by deleting the triangles of the input mesh that represent the unwanted part of the surface, then use the original mesh as the input to the scattered data interpolation to obtain the analytical variational surface, the resulting surface is finally obtained by applying the subdivision scheme to the edited mesh. The knot example shown in Fig. 16 only displays part of the interpolating implicit surface, a result that is difficult to achieve using marching-cubes type tessellation methods.

## 6. Discussion

For the interpolating subdivision surface, the limit surface is determined by the coordinates of the control vertices, the connectivity of these vertices and the subdivision scheme. For the same vertices set, different connectivity and subdivision schemes will lead to different limit subdivision surfaces. On the other hand, different connectivity and subdivision schemes will converge to the same surface for the interpolating implicit surface. That is because the interpolating implicit surface is fully determined by the coordinates and the normals of the control triangular mesh. The problems with non-uniform and non-stationary schemes for subdivision surfaces do not exist in the proposed approach. Hence we can adopt any non-uniform and non-stationary subdivision schemes, such as curvature or distance to the camera based schemes, to adaptively subdivide the interpolating implicit surface.

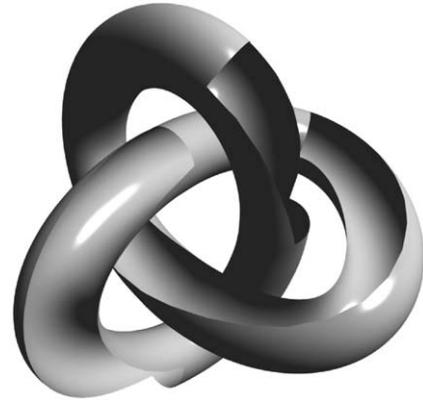


Fig. 16. Shows part of the interpolating implicit surface.

Both the interpolating subdivision surface and the subdivision interpolating implicit surfaces pass through all the vertices of the control mesh. They share the same subdivision scheme and produce similar resulting surfaces. Each method has its own advantages and limitations. The advantage of interpolating subdivision surfaces is its fast and stable computation. A new added vertex is simply the affine combination of nearby vertices. However, unlike Catmull–Clark type approximation schemes, interpolating subdivision surfaces may exhibit ripples and undulations over the surface, especially near tight joint areas. In contrast, such lack of fairness will not exist in interpolating implicit surfaces because of the minimized energy functional. Any such artifacts would result in a larger energy. The limitation of subdivision interpolating implicit surfaces is its slow computation, which involves two computational intensive steps: solving the linear system and the Newton iteration. If we take the input mesh as a polyhedral sample of a smooth surface, then the interpolating implicit surface and the interpolating subdivision surface can be regarded as two methods of recovering the original smooth surface. When fairness is critical, we can use a subdivision interpolating implicit surface as an alternative to an interpolating subdivision surface because of its smoothness.

For marching-cubes type tessellation methods, the resulting triangles will be different when the implicit surface is transformed. Any translation, scaling and rotation transformation may lead to potentially different triangulation result with different number of triangles. In contrast, the connectivity of the resulting triangles for the presented scheme will always be the same no matter how the implicit surface is transformed, even deformed. The user has full control of the number and connectivity of the resulting triangles in our approach.

In our implementation, we used a triharmonic spline as the radial basis function. These infinite-support basis functions have the following limitations: (1) Moving one vertex of the input mesh will globally affect the whole implicit shape, and this may be inappropriate in some applications. (2) They are not suitable for models with large constraints because of the computational and storage complexity involved. Fortunately, local control and computational reduction can be achieved by using compactly supported radial basis functions [23,24].

## 7. Conclusions

In this paper, we have proposed and developed a new progressive method to tessellate interpolating implicit surfaces controlled by a triangular mesh with an arbitrary topology using subdivision schemes. The information stored in the input triangular mesh is fully exploited in the later tessellation, which saves computational time. We have also developed a new smooth surface modeling tool by blending interpolating subdivision surfaces and subdivision interpolating implicit surfaces. By integrating the benefits of the interpolating implicit surface and the interpolating subdivision surface, our approach has the following features:

- *Analytical representation.* The whole surface can be represented using one analytical function. This feature is inherited from the interpolating implicit surface.
- *Smooth interpolation.* All of the vertices of the control mesh are interpolated and the newly generated vertices lie on the interpolating implicit surface. The resulting meshes are both uniform and smooth.
- *High performance.* The algorithm is simple to implement, and is nearly 4 times faster than the continuation method.
- *Progressive refinement and multiresolution.* The control mesh can be refined progressively until the user is satisfied with the result. The multi-resolution LOD model is automatically established and can be used in time critical rendering.
- *New surface modeling tool.* This is easily achieved by blending the two kinds of interpolating surfaces.

Our approach has the following limitations. (1) It only applies to interpolating implicit surfaces. For other implicit surfaces, such as metaballs and convolution surfaces, the presented method cannot be applied directly. One solution is to first generate a coarse tessellation using the marching-cubes algorithm, and then refine the mesh using a subdivision scheme. (2) It will not improve the aspect ratio of the original mesh. If there are triangles with poor aspect ratios in the input mesh, the subdivision scheme will propagate these ratios in the result. (3) For an input triangular mesh with folds, the presented tessellation will create folds in the resulting mesh.

## Acknowledgements

The authors are grateful to Greg Turk for his help on providing the PLY models of the paper. We are also grateful to the anonymous reviewers for their constructive suggestions. This work was supported by RGC research grant of Hongkong (No. 4189/01E), RGC direct grant (No. 4450002), National Natural Science Foundation of China (No. 60273054, 60021201), National Key Basic Research Foundation of China (No. 2002CB312101) and Specialized Research Fund for the Doctoral Program of Higher Education (No. 20020335070).

## References

- [1] Bloomenthal J, Bajaj C, Blinn J, Cani M, Rockwood A, Wyvill B, Wyvill G. An introduction to implicit surfaces. Los Altos, CA: Morgan Kaufmann Publishers; 1997.
- [2] Cani-Gascuel M, Desbrun M. Animation of deformable models using implicit surfaces. IEEE Transactions on Visualization and Computer Graphics 1997;3(1): 39–50.
- [3] Dobashi Y, Kaneda K, Yamashita H, Okita T, Nishita T. A simple, efficient method for realistic animation of clouds. In: Proceedings of SIGGRAPH '00, New Orleans, 2000. p. 19–28.
- [4] Jin X, Li Y, Peng Q. General constrained deformation based on generalized metaballs. Computers & Graphics 2000;24(2):219–31.
- [5] Jin X, Tai C. Analytical methods for polynomial weighted convolution surfaces with various kernels. Computers & Graphics 2002;26(3):437–47.
- [6] Jin X, Tai C. Convolution surfaces for arcs and quadratic curves with a varying kernel. The Visual Computer 2002;18(8):530–46.
- [7] Savchenko V, Pasko A, Okunev O, Kunii T. Function representation of solids reconstructed from scattered surface points and contours. Computer Graphics Forum 1995;14(4):181–8.

- [8] Turk G, O'Brien J. Shape transformation using variational implicit functions. In: Proceedings of SIGGRAPH '99, Los Angeles, 1999. p. 335–42.
- [9] Yngve G, Turk G. Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics* 2002;8(4):346–59.
- [10] Dinh H, Turk G, Slabaugh G. Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2002;24(20):1358–71.
- [11] Carr J, Fright W, Beatson R. Surface interpolation with radial basis functions for medical imaging. *IEEE Transactions on Medical Imaging* 1997;16(1):96–107.
- [12] Turk G, O'Brien J. Modeling with implicit surfaces that interpolate. *ACM Transactions on Graphics* 2002;21(4):855–73.
- [13] Carr J, Beatson R, Cherri J, Mitchell T, Fright W, McCallum B. Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of SIGGRAPH '01, Los Angeles, 2001. p. 67–76.
- [14] Karpenko O, Hughes J, Raskar R. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum* 2002;21(3):585–94.
- [15] Hart J. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 1996;12(10):527–45.
- [16] Bloomenthal J. Polygonization of implicit surfaces. *Computer Aided Geometric Design* 1988;5(4):341–55.
- [17] Bloomenthal J. An implicit surface polygonizer. In: Heckbert PS, editor. *Graphics gems IV*. New York: Academic Press; 1994. p. 324–49.
- [18] Lorensen W, Cline E. Marching cubes: a high resolution 3-D surface construction algorithm. *Computer Graphics* 1987;21(4):163–9.
- [19] Dyn N, Gregory J, Levin D. Butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics* 1990;9(2):160–9.
- [20] Zorin D, Schröder P, Sweldens W. Interpolating subdivision for meshes with arbitrary topology. In: Proceedings of SIGGRAPH '96, New Orleans, 1996. p. 189–92.
- [21] Hartmann E. A marching method for the triangulation of surfaces. *The Visual Computer* 1998;14(3):95–108.
- [22] Karkanis T, Stewart J. Curvature dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications* 2001;21(2):60–9.
- [23] Morse B, Yoo T, Rheingans P, Chen D, Subramanian K. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In: *Shape Modeling International '01*, Genoa, 2001. p. 89–98.
- [24] Wendland H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics* 1995;4(4):389–96.