Subdivision-Curve Primitives: a New Solution for Interactive Implicit Modeling

Marie-Paule Cani and Samuel Hornus

iMAGIS[†]-GRAVIR, INRIA Rhône-Alpes
655 avenue de l'Europe, 38330 Montbonnot, France
Marie-Paule.Cani@imag.fr
http://www-imagis.imag.fr/Membres/Marie-Paule.Cani/

Abstract

To remain an attractive model, skeleton-based implicit surfaces have to allow the design and display of shapes at interactive rates. This paper focuses on surfaces whose skeletons are graphs of interconnected curves. We present subdivision-curve primitives that rely on convolution for generating bulge-free and crease-free implicit surfaces. These surfaces are efficiently yet correctly displayed using local meshes around each curve that locally overlap in blending regions. Subdivision-curve primitives offer a practical solution to the unwanted-blending problem that ensures C^1 continuity everywhere. Moreover, they can be used to generate representations at different levels of detail, enabling the interactive display of at least a coarse version of the objects, whatever the performance of the workstation.

Categories and subject descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling–Curve, surface, solid and object representations;

Additional Key Words and Phrases: Convolution Surfaces, Levels of Detail, Implicit Surfaces.

1 Introduction

Implicit surfaces are defined as the set of points P verifying an equation $f(P) = c$, where f is a scalar "field function", and c is a given iso-value. The advantages of this representation are well-known [3]: these surfaces surround a closed volume, whose in-out function eases the computation of intersections along rays or with other objects. Shapes of any topology can be easily created using several primitives that blend their field function contributions. Moreover, local and global deformations can be incorporated in the construction tree.

However, although they have been introduced in Computer Graphics for many years now [2, 12, 21], implicit surfaces are not as popular as parametric surfaces for performing modeling tasks. One of the main problems is the lack of parameterization which makes the interaction with implicit shapes very difficult. The second problem is the high cost, in the general case, of the calculation of the field function that generates the surface. This can be easily understood: the theory tells us that any 3D shape can be represented by a skeleton, defined as the locus of the centers of the maximal spheres included inside the object [1]. This skeleton is a graph of interconnected curves and surface patches. Generating implicit surfaces without unwanted creases and bulges from such complex skeletons requires the use of convolution, which is known to be computationally expensive [4, 16].

This paper presents a solution to the interactive modeling and display of implicit shapes whose skeletons are graphs of branching curves. As shown in [19], this restricted set of skeletons still covers a number of useful cases. Our solution relies on convolution surfaces and on levels of detail (LOD) for defining subdivision-curve implicit primitives that can be displayed at interactive rates.

Section 2 discusses related works. Our first contribution, described in Section 3, is the introduction of the subdivision-curve primitive, which uses convolution for allowing the definition of implicit shapes at different levels of detail. Section 4 presents a new closed-form convolution kernel that allows the definition of implicit surfaces of a varying radius along a skeleton curve. The third contribution, described in Section 5 is a simple method for the

[†]iMAGIS is a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

interactive display of blended subdivision-curve primitives, based on locally-overlapping meshes. Section 6 shows that the subdivision-curve representation offers a practical solution to the unwanted blending problem, which ensures C^1 continuity everywhere. Sections 7 and 8 discuss results and possible extensions of this work.

2 Related works

2.1 Modeling complex shapes with implicit surfaces:

A recurring question in implicit modeling: What is the most efficient way to model a given 3D shape? Should we blend a few, complex primitives, or many simple ones? Should we, rather, define the object using a sampled-field representation?

Blending simple primitives: Among the practical solutions used in implicit shape design, the most common one consists in blending a large number of very simple primitives such as point skeletons, which sum their field contributions. Controlling these points is quite intuitive. However, the number of primitives needed for modeling a given shape shape can increase to an arbitrarily high value (think of modeling a locally planar surface, for instance). As a consequence, computing the field function at a given point may become very expensive.

Sampled-fields: A practical solution for avoiding this cost consists in sampling the field function over a 3D grid. Then, linear interpolation can be used for computing field values anywhere in constant-time. This approach was recently used in a real-time sculpting system [8]. The additional field contributions created by simple tool-primitives are progressively incorporated to the sampled-field. However, this representation is not well suited to deformations and animation: no structure is stored and the object would have to be re-sampled through the grid if it is moved or deformed.

Convolution surfaces: In theory, any 3D object can be entirely defined from a geometric skeleton, i.e. a graph of interconnected curves and surface patches, provided with radius information that describe the object's local thickness. This skeleton, or approximated versions of it, can be computed from sampled-points on the surface [1, 19]. Directly generating implicit surfaces from these continuous skeletons requires the use of convolution surfaces [4], otherwise creases and bulges would be generated. Defined using integrals of field contributions along primitives, convolution surfaces were long considered as very expensive models. To avoid this cost, Ferley [9] defined a relatively complex solution for generating smooth branching between skeleton curves. However, this method only worked for modeling the branching between three curves, which was quite limiting.

The first implementation of convolution surfaces [4] relied on the precomputation and storage of sampled field values. This increases efficiency but requires large volumes of memory. An alternative is to find closed-form solutions for integrals. Such solutions were recently given for a number of classical kernels, convolved with a restricted number of simple primitives such as line-segments [16]. However, no kernel resulted in a closed-form solution when integrated along a free-form curve.

The method studied in this paper belongs to structured modeling based on convolution surfaces. Our aim is to generate a surface from a graph of free-form curves, provided with a varying radius parameter. Such curves offer a compact yet intuitive definition of an object's shape, well suited to deformations and animation.

2.2 Interactive display and levels of detail

Solutions for interactive display: The lack of parameterization has long been an obstacle to the interactive display of implicit surfaces. When only local editing of the shape is performed, real-time visualization can be obtained through incremental polygonisation [8]. However, this yields high memory costs since a huge data-structure has to be stored.

In constructive approaches, alternative methods have been used for displaying very quickly approximated representations of the designed shape. The simplest solution is to display non-blended versions of the primitives [14] or offsets of the skeletal elements [15]. Witkin [20] uses particles, rendered as discs oriented along the normal, for displaying sampling points on the designed surface. Desbrun [7] rather displays a piece-wise polygonisation, defined by closed-meshes surrounding each implicit primitive. Each of the meshes samples the region of the implicit volume where the contribution of the associated primitive is the highest (this region is called the primitive's *territory*). Unfortunately, this results in local "cracks" between parts of the surface sampled by different meshes, as shown in Figure 1. Moreover, the number of polygons linearly increases with the number of primitives, which is unfortunate since some of them may be hidden inside the implicit volume. For instance the polygons generated between two blended point-primitives in Figure 1 are hidden, and serve no purpose since lips do not use to break into pieces.

The LOD paradigm: Whatever the representation, very complex objects will need an arbitrarily large number of

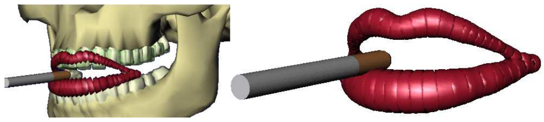

Figure 1. This figure, extracted from [5], illustrates the use of the interactive display method in [7], in the context of implicit lips made of a series of point-primitives positioned along a curve. Even with fine sampling rates (right), small cracks between local meshes are still visible.

polygons to be displayed. Parametric models cope with this problem using the level of detail (LOD) paradigm: a region of interest, or an object which is close to the camera, will be displayed at a fine scale, while coarser representations will be used elsewhere.

To become well suited to interactive modeling, implicit surfaces have to obey this paradigm: at least a coarse version of a shape should be available for interactive display, and refined versions should be generated where and when needed.

Although already studied in the sampled-field representation [18, 11, 10], the LOD paradigm has never been used, to the authors knowledge, in the constructive implicit modeling framework. There, the intuitive way to achieve this would be to progressively simplify (respectively refine) the skeleton that defines the surface, yielding a more efficient field evaluation (respectively, a more detailed shape). However, finding implicit primitives that allow a continuous alteration of the shape when their skeletons are simplified or refined is not easy. In particular, the classical distance surfaces that achieve blending by summing the skeleton contributions, would result in a bulged version of themselves if a skeleton segment refines into two sub-segments.

Our new subdivision-curve primitive, introduced next, is a first solution to this problem, since it provides several approximations of the desired shapes at different LODs.

3 The Subdivision-Curve Implicit Primitive

3.1 A subdivision-curve skeleton

In the remainder of this paper, we consider implicit surfaces generated by skeletons which are graphs of interconnected curves, described with a varying radius along each curve segment.

As stated earlier, generating a representation at several LODs can be done by refining or simplifying the skeleton. This requires the use of convolution for generating the surface. Indeed, thanks to the properties of the integral, convolution surfaces maintain a constant shape when a skeleton primitive breaks into pieces. This allows for a continuous variation of the shape during refinements and simplifications. However, we have mentioned that although there are closed-form solutions for convolution along line-segments, there is no solution yet for convolution along a free-form curve.

Our basic idea consists of using a subdivision-curve as a skeleton. Subdivision-curves (see for instance [17]) are limit curves of a series of polylines, that are recursively computed using "corner cutting" techniques. These polylines provide the LOD representations of the curve we are looking for.

For instance, let us consider curves defined using Chaikin's subdivision mask: The user gives a control polyline $(c^0) = (c^0_0, \ldots, c^0_{m-1})$. This polyline can be used as a skeleton to generate a first, coarse approximation of the shape. If a finer representation is required, the line is recursively subdivided. This is done by using:

$$c^j_i = r_{-1}\,\dot{c}^j_{i-1} + r_0\,\dot{c}^j_i + r_1\,\dot{c}^j_{i+1}$$

where $\dot{c}^j_i = c^{j-1}_k$ if $i = 2k$, and $\dot{c}^j_i = \frac{1}{2}(c^{j-1}_k + c^{j-1}_{k+1})$ if $i = 2k+1$.

If $(r_{-1}, r_0, r_1) = (0, \frac{1}{2}, \frac{1}{2})$, the limit curve is an uniform quadratic Bspline. If $(r_{-1}, r_0, r_1) = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$, the limit curve is the well known cubic uniform B-spline.

In practice, the user may like to slightly modify this model by defining two kinds of joints in the initial polylines: those that should be smoothed, and those, called "sticking joints", that should remain un-changed. In consequence we are using a modified version of Chaikin's mask, where a control point is replaced by two points only when it is not marked as a sticking joint. See figure 2.

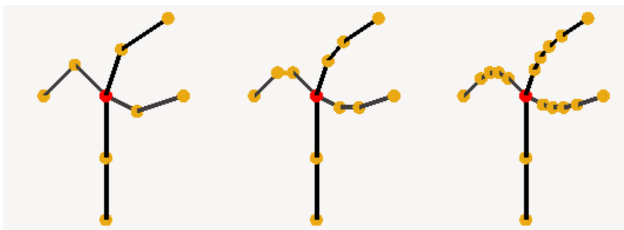

Figure 2. A subdivision-curve skeleton. The user defines a coarse version of the graph of curves (left). During refinement, the curves are smoothed (right), except at the "sticking joints", displayed in darker grey, that the user has specified.

Subdivision-curve implicit primitives are computed by convolving a polyline, which is the representation of the subdivision curve at a given LOD, with a well chosen kernel. We will come back to the choice of an adequate kernel in Section 4.

3.2 Specifying a varying radius along subdivision-curve

Associating a non-constant radius parameter along a subdivision-curve is easy: the user defines radius values R_i at the joints of the coarse control polygon that is used for defining the subdivision-curve. These values are interpolated for defining a radius $R(u)$ at any parameter u along the curve. When the skeleton curve subdivides, new radii are associated to the new vertices using the same subdivision mask as for vertex positions.

However, taking a local radius value into account during convolution can be a problem. Classical methods for generating surfaces of varying radius modify the distance $d(P)$ used in the field function computation by either dividing it by $R(u)$, or subtracting $R(u)$ from it, where u is the parameter of the curve point that is the closest to P (see [9]). This approach cannot directly be used in a convolution surface scheme: the field value integrates contributions along the curve rather than using the distance to the closest point. Thus, taking a varying radius into account is not easy. Incorporating it inside the convolution integral would certainly spoil the chance of finding any closed-form solution.

We will see that the new convolution kernel presented below solves the problem, since its closed-form solution for convolution along a line-segment is based on the distance to the closest point on this line.

4 Choice of a Convolution Kernel

4.1 An efficient infinite-support kernel

Our solution is based on the use of $h(r) = 1/r^3$ as convolution kernel, r being the distance from a given point to the curve. The field function that results from this new kernel along a line-segment primitive $[C1, C2]$ is:

$$f(P) = \frac{(\sin(\alpha 1) - \sin(\alpha 2))}{d^2(P,H)} \tag{1}$$

where $d(P, H)$ is the distance between P and its projection point H on the line-segment support, and where the angles α_1 and α_2 are the signed angles $([PH], [PC_1])$ and $([PH], [PC_2])$, respectively (see Figure 3).

Computing this field value does not take too many operations, since sine values are easily computed using scalar products. In comparison with the number of operations given for the finite support polynomial kernel in [16], we get:

Number of operations	*	/	+ or -	sqrt
New kernel	28	3	22	2
Polynomial kernel	33	3	36	1

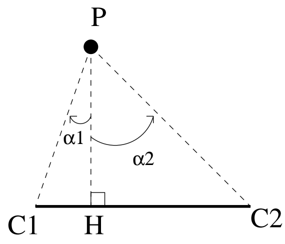

Figure 3. The field function that results from the integration of $1/r^3$ along a line-segment primitive is based on the distance between P and H, and on the signed angles α_1 and α_2.

Numerical experiments showed that the new kernel has about the same performance as the polynomial one.

In practice, the convolution surfaces we use are generated by taking the iso-surface of iso-value $iso = 1$ of the convolved field.

4.2 Modified field for generating surfaces of non-constant radius

The closed-form field value at point P given in Equation (1) relies on the distance d to the closest point H on the segment skeleton. This allows us to generate a surface of a non-constant value along the skeleton. As stated earlier, the user defines radii at each control point of the skeleton curve, so a radius $R(H)$ at point H can be interpolated from the radii at the segment extremities.

The surface is set to a varying radius by replacing $d^2(P, H)$ in Equation (1) by $D^2(P, H)$, where:

$$D(P, H) = \sqrt{2}\frac{d(P, H)}{R(H)}$$

With this expression, the surface will tend to have the radius $R(H)$ if P is near the center of a very long primitive, since sine values in Equation (1) then tend to 1. As in conventional convolution models, the local surface thickness is smaller than the specified value near the extremities of a primitive. Moreover, if the user has specified different radius values along a subdivision-curve primitive, the surface thickness will change accordingly along the primitive, since $R(H)$ changes with the projection point H.

5 Interactive Display

The method we use for the interactive display of subdivision-curve primitives is an extension of the adaptive sampling method introduced by Desbrun [7]: Sample-points are sent along given axes by each skeletal element in order to sample the element's "territory", i.e. the region where its field contribution is the highest. Sample-points may start from a previous position when they converge to the iso-surface, enabling the use of benefits gained from temporal coherence. This feature is essential for increasing efficiency during interactive modeling or animation sessions. This idea of temporal coherence is also very interesting in our subdivision/refinement framework: it will allow sample points to migrate from their previous position when a skeleton refines. However, two problems have to be solved for generating an efficient yet good quality display:

- Associating individual meshes with each skeleton primitive is inefficient for neighboring primitives (see Figure 1).

- Gaps between local meshes appear in Desbrun's method, since sample points are stopped at the limit of their associated primitive's territory.

Our solutions to these two problems are the following: First of all, we generate a single mesh around each subdivision-curve primitive. Mesh nodes are attached to specific segments of the line, and converge to the iso-surface along fixed axes, as was done in [7]. When the subdivision-curve refines, nodes are attached to smaller line-segments and their associated axes are modified accordingly. This increases the quality of results, as shown in Figure 4, and reduces the number of generated polygons compared to the use of several disconnected meshes as in Figure 1.

Figure 4. Interactive display of a lips model using the subdivision-curve methodology. The skeleton is made of two user-defined control-polylines that do not blend (left). Our display method associates a closed mesh to each of the curves (center and right). A non-constant surface radius is specified along the lines. The subdivided subdivision-curves are displayed on the right.

Secondly, we avoid discontinuities when several subdivision-curve primitives blend together by extending the region sampled by each mesh: instead of only sampling its primitive's territory, a mesh is set to sample the region of the implicit surface where the parent primitive's field value *plus a given constant* (0.5 in our implementation) is higher that any other contribution. This creates local overlapping regions between neighboring meshes, as shown in Figure 5, thus increasing the visual quality of the interactive display.

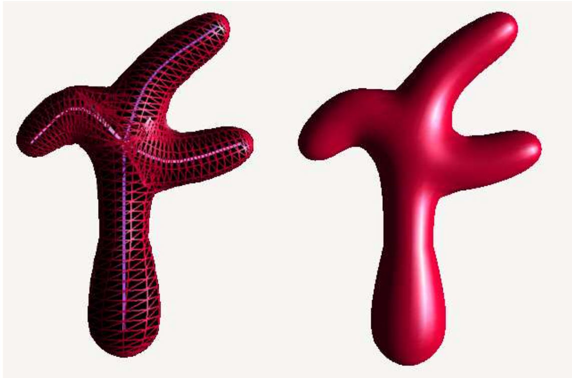

Figure 5. Implicit surfaces generated from the skeleton in Figure 2, made of two curves that sum their contributions. The local overlapping between the two closed meshes used for display yields quite a good quality visual result, without altering performances.

6 Avoiding Unwanted Blending with Preserved Smoothness

Modeling and animating complex shapes requires the ability to use a restricted blending graph. For instance, the loops of the snake's body in Figure 6 should not blend together.

Current solutions to the unwanted blending problem [13, 6], do not maintain the C^1 continuity of the shape everywhere, since the field at a given point is defined as the maximal contribution from groups of skeletons that blend together. This solution results in a union of volumes, possibly creating tangent discontinuities in regions where blending properties change.

Modeling with subdivision-curve primitives offers a practical solution to the problem: we use the skeleton, i.e. our graph of interconnected subdivision curves for defining blending properties, stating that a line-segment blends with its immediate neighbors, and that blending is locally transitive. Now, we have to avoid unwanted blending between segments that are very far away with respect to the topology of the graph, as the folds of the snake in Figure 6. To do so, we integrate the field at a point P by recursively adding the contribution of the segments that are nearest to P, and the contributions from their neighbors, but stopping along a curve as soon as a contribution can be neglected. Then, if a long curve loops and then folds back, the two folds will not blend together. This method yields C^1 continuity since a contribution is disregarded only when it is negligible. However, it should be noted that it only gives a partial solution to the problem: generating very close branches that do not blend cannot be done using our method.

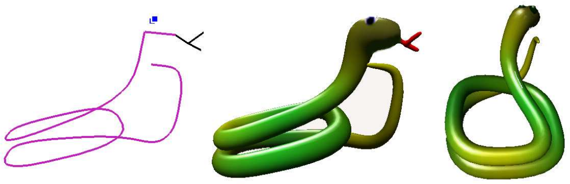

Figure 6. Unwanted blending is avoided while preserving C^1 continuity when this snake folds back onto itself.

7 Results

The stylized kangaroo example in Figure 7 illustrates the kind of shape we can generate with our approach. The object is displayed at two different levels of detail. The left hand-side model only relies on 27 line-segment primitives for approximating the subdivision-curves that constitute the skeleton, while the right hand-side model uses 216 line-segments. Knowing that computing the field contribution of a line-segment takes a constant time, computing the surface on the left is about 8 times faster than computing the one on the right, if the same number of points are used to sample them.

Figure 7. A stylized kangaroo modeled using subdivision curve primitives. Two different levels of detail are shown.

In practice, there is usually no need to compute a fine mesh along a coarse version of a subdivision-curve as only a coarse representation of the object will be displayed. Figure 8 shows different representations of the same shape obtained by adapting the mesh resolution according to the resolution of the underlying skeleton.

8 Conclusion

Subdivision-line primitives define free-form implicit surfaces that can be displayed in real-time by adjusting the LOD at which a surface is represented. This is done in a framework of structured modeling, where objects are edited by applying deformations to the skeleton curve. These two features make the model very well suited to interactive modeling systems where the user could first display a coarse version of a shape, and then adjust details by “zooming” (i.e. increasing the LOD) in the region where the skeleton curve is to be edited.

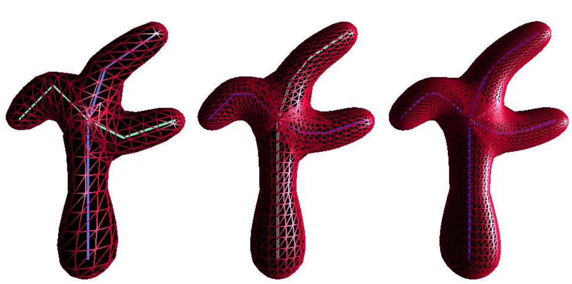

Figure 8. Representations of a surface at different LODs can be obtained by tuning both the resolution of the local meshes that sample the surface, and the resolution of its subdivision-curve skeletons. The left-hand-side model could be used when the objct is far away, and the two other representations when it gets closer.

Subdivision-curve primitives would also be a good model for generating animation sequences, since the skeleton curves give an intuitive way to apply motion and deformations to objects. If these primitives were used in an interactive animation system, an automatic, error-driven, refinement/simplification procedure would have to be settled for automatically tuning LODs, using criteria such as the size of a primitive on the screen.

Lastly, interesting future work would consist in extending our methodology to surface primitives, thus offering the possibility to model any 3D shape. A closed-form convolution primitive already exists for triangles [16], and could be used for generating subdivision-surface primitives based on triangle subdivision schemes. Closed-form primitives could be searched for quadrilateral primitives, since many subdivision schemes such as Catmull-Clark scheme [17] rely on quadrilateral elements.

Acknowledgments: The authors would like to thank Andrei Sherstyuk for his helpful suggestions, and Pauline Jepp for carefully re-reading this paper.

References

[1] D. Attali and A. Montanvert. Computing and simplifying 2d and 3d semicontinuous sekeletons of 2d and 3d shapes. *Computer Vision and Image Understanding*, 67(3):261–273, Sept. 1997.

[2] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[3] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.

[4] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[5] M.-P. Cani-Gascuel. Layered deformable models with implicit surfaces. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, June 1998. Invited paper.

[6] M.-P. Cani-Gascuel and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, Mar. 1997.

[7] M. Desbrun, N. Tsingos, and M.-P. Cani-Gascuel. Adaptive sampling of implicit surfaces for interactive modeling and animation. *Computer Graphics Forum*, 15(5):319–327, Dec. 1996. A preliminary version of this paper appeared in *Implicit Surfaces'95*, Grenoble, France, may 1995.

[8] E. Ferley, M.-P. Cani, and J.-D. Gascuel. Practical volumetric sculpting. *To appear in the Visual Computer, 2000*, 2000. A preliminary version of this paper appeared in *Implicit Surfaces'99*, Bordeaux, France, sept 1999.

[9] E. Ferley, M.-P. Cani-Gascuel, and D. Attali. Skeletal reconstruction of branching shapes. *Computer Graphics Forum*, 16(5), Dec. 1997. An early version of this paper appeared in *Implicit Surfaces'96*, Eindhoven, The Netherlands, oct 1996.

[10] S. Frisken, R. Perry, A. Rockwood, and T. Jones. Adaptive sampled distance fields: A general representation of shape for computer graphics. *Computer Graphics*, July 2000. Proceedings of SIGGRAPH'00 (New Orleans, July 2000).

[11] L. Grisoni and C. Schlick. Multiresolution representation of implicit objects. In *Implicit Surfaces'98—Eurographics and ACM-Siggraph Workshop*, pages 1–10, Seattle, USA, June 1998.

[12] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Objects modeling by distribution function and a method of image generation (in japanese). *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):718–725, 1985.

[13] A. Opalach and S. Maddock. Implicit surfaces: Appearance, blending and consistency. In *Fourth Eurographics Workshop on Animation and Simulation*, pages 233–245, Barcelona, Spain, Sept. 1993.

[14] J. Shen and D. Thalmann. Interactive shape design using metaballs and splines. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 187–195, Grenoble, France, Apr. 1995.

[15] A. Sherstyuk. Interactive shape design with convolution surfaces. In *Shape Modeling International '99*, pages 56–65, Aizu-Wakamatsu, Japan, Mar. 1999.

[16] A. Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer*, 15(4), 1999.

[17] E. Stollnitz, T. Derose, and D. Salesin. *Wavelets for Computer Graphics*. Morgan Kaufmann, San Francisco, California.

[18] L. Velho and J. Gomez. Approximate conversion of parametric to implicit surfaces. *Computer Graphics Forum*, 15(5):327–337, Dec. 1996. A preliminary version of this paper appeared in *Implicit Surfaces'95*, Grenoble, France, may 1995.

[19] A. Verroust and F. Lazarus. Extracting skeletal curves from 3d scattered data. In *Shape Modeling International '99*, pages 194–201, Aizu-Wakamatsu, Japan, Mar. 1999.

[20] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94.

[21] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, Aug. 1986.