A Field Interpolated Texture Mapping Algorithm for Skeletal Implicit Surfaces

Mark Tigges Brian Wyvill

Dept. of Computing Science
University of Calgary, Calgary, Alberta, Canada
mtigges@cpsc.ucalgary.ca blob@cpsc.ucalgary.ca

January 25, 1999

Abstract

We address the problem of texture mapping implicit surfaces. Recent work in providing 2D mappings for implicit surfaces maps points from a auxiliary parametric surface to the implicit surface. In this work we extend this method by observing that all commonly used skeletal elements are easily parametrized and can be their own support surfaces. In areas of a surface which are due to multiple skeletal primitives we compute a mapping through relations found with the zero level surface of a primitive by using the gradient field.

Keywords: texture mapping, implicit surfaces.

1 Introduction

Surfaces in computer graphics are typically described either with a set of parametric functions or a single implicit equation $f(\mathbf{p}) = 0$. The point sets resulting from the latter case are often called implicit surfaces [4]. The former parametric case enjoys the feature that a two dimensional mapping exists for the surface by the virtue of the parametric functions. Computing a mapping for an implicit surface is not as trivial, there is no associated two dimensional space defined on the surface.

In this paper we describe an algorithm for extending a method introduced recently for the texture mapping of implicit surfaces [19], to allow its use within commonly used implicit surface modeling practices. Previously the method required a substantial load on the user to specify parameters and place a parametric surface around the implicit surface. Moreover, the technique is best used to texture an implicit surface globally. For instance, in [13] the method was extended to allow constructive solid geometry and warps, and in [21] it was extended to accomodate articulated jointed blends. Our goal in this work is to define the texture directly on the surface, associated with each individual primitive skeletal element. This is possible when one notes that in a skeletal implicit surface system the individual elements often have a natural parametrization defined on their surface. In this work we restrict ourselves to such skeletal primitives. Implicit surfaces built with skeletal elements typically sum the scalar fields of several individual elements built using spheres, cylinders, cones etc. Each of these types has a natural parametric definition. The remaining work is to ensure that this mapping can be distorted properly to provide a continuous two dimensional mapping for a skeletal element even in the presence of blending with other elements.

In section 2 we describe relevant previous work. In section 3 we review the basic implicit surfaces techniques for skeletal implicit surfaces and describe the methods for using field interpolated surface attributes. In section 4 we define mappings for some commonly used primitive skeletal elements and describe the modification to the existing algorithm necessary to allow their use. Results and comparisons are presented in section 5 and conclusions are drawn in section 6.

2 Previous Work

Implicit surfaces are a point set defined by a single implicit function. For instance a unit radius sphere can be represented implicitly with $f(x, y, z) = x^2 + y^2 + z^2 - 1$. Modern use of this class of surfaces computes a scalar field based on a function of distance from a skeleton of primitives. The primitives are generally lines, points, polygons etc. These techniques were introduced by Blinn [3], and refined by Wyvill et al. [17] and Nishimura et al. [10]. For the purposes of this paper we restrict the class of implicit surfaces to skeleton based surfaces in accordance to those defined by separately by Wyvill. The definition of Blinn [3] uses an exponential function to define the scalar field. This function does not have finite support, which introduces complications to the algorithm presented here.

Texture mapping is the definition of surface attributes through the application of a mapping function from object space to texture space [9]. A great deal of work has been done in this area, the introduction of the technique was by Catmull [5] and Blinn [2], an excellent survey of the basics of texture mapping has been written by Heckbert [9].

Texture mapping implicit surfaces is a non-trivial problem since there does not exist a two dimensional space defined on the surface as with a parametrically defined surface. The first work in texturing implicit surfaces mapped the three dimensional object space to a three dimensional texture space [18]. This solid texturing approach was very succesfull, but it remains desireable to be able to apply the two dimensional space of a raster image onto the surface.

An interactive method for defining a two dimensional mapping for an implicit surface was presented by Pedersen [11]. In Pedersens method the user is assisted in placing a set of patches over the implicit surface. However it is desireable to have an automatic technique for computing a mapping. Such a technique was introduced by Zonenschein et al [19] based on previous work by Figueredo [7]. Zonenscheins method bounds the implicit surface with an auxiliary surface which does have a natural parametrization. Both cylinders and spheres have been used but in principle any surface could be used. The parametrization for the implicit surface is found by mapping points between it and the auxiliary surface. This mapping is found through an interpolation of the gradient fields of the two surfaces. This is similar in principle to the two stage texture mapping due to Bier and Sloan [1]. The parametrization produced using the Zonenschein algorithm provides a global mapping for an implicit surface built from the blending of multiple individual skeletal implicit surfaces. Adaptation of the method to skeletal models built using CSG and warping, and to composite deformable models were addressed in two separate papers [13, 21]. Producing mappings at individual locations in the model is possible but requires considerable amounts of parametrization, control over local mappings was presented in [20].

The work presented in this paper deals with defining texture spaces on each individual primitive in a composite implicit surface and blending the attribute values in a continuous fashion at geometric blending points. Sherstyuk [12] describes a simple scheme based on methods described in [18] to combine the parametrizations defined on the skeletal elements. His technique does not succeed well in the areas of blending. The method in this paper presents an algorithm which finds a mapping at the blended locations of the surface by modification of the gradient mapping algorithm to suit the mapping of the individual elements.

3 Implicit Surfaces

In this section we review the definition of an implicit surface from a skeleton of primitives. Each primitive can be a point, or a line, or a line with varying radius or any other type of object. We also review the method for defining an attribute at any point on the surface by forming a linear combination of the attribute values defined on the primitives.

3.1 Surface Definition

The surface definition of an implicit surface is made by selecting an iso-value of a scalar field, the resulting surface is called an *iso-surface* or a *level set*. In an implicit surface defined with a skeleton of primitives the scalar field is generated with relation to a distance from the skeleton. The field value for an implicit surface $\mathcal{S}$ with child primitive skeletons $\mathcal{S}_i$ is

generated with the following function:

$$F(\mathbf{m}, \mathcal{S}) = \sum_{i=1}^{n} c_i f_i(r_i, \mathcal{S}_i) \tag{1}$$

where each r_i is the distance between $\mathbf{m}$ and the i^{th} child skeleton and each f_i is some monotonically decreasing sigmoid function ranging from 1 to 0 over it's domain. This will cause a smooth blend between the primitive skeletons when the f_i results are summed. A common function to use, equation 2, is due to Wyvill et al. [17], its graph is shown in figure 1. Implicit surfaces built with this forumation are called *Soft Objects*, the iso-value chosen is typically 0.5.

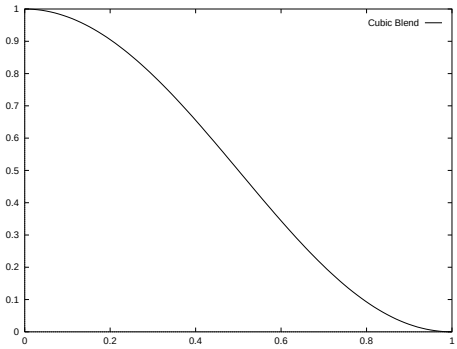

Figure 1: Field function of a *Soft Object*.

$$fr = \left(1 - \frac{4}{9}\frac{r^2}{R^2}\right)\left(1 - \frac{r^2}{R^2}\right)^2 \tag{2}$$

The summation in equation 1 can be replaced with some other equation to enable a variety of blending effects. See works by Ricci [?] and Pasko [?] for generalized blending functions.

3.2 Surface Attributes

Modeling implicit surfaces involves the interpolation of surface attributes defined on the individual primitives [17]. For instance the colour of an implicit surface is found by interpolating colour defined on each of the contributing skeletons. The field value of the skeletal element contributes to the surface is used as its weight in the interpolation. For example, figure 2 shows the blending of two point skeletons and interpolated colour between yellow and blue.

Given a point $\mathbf{m}$ on the $\mathcal{L}$ level surface of a compound implicit surface $\mathcal{S}$ which is due to the

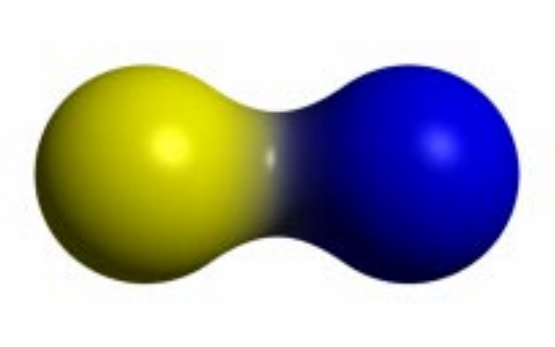

Figure 2: Interpolation of the attribute of colour across blends.

contribution of n primitives, the attribute a for the surface at $\mathbf{m}$ is found as follows:

$$a = A(\mathbf{m}, \mathcal{S}) = \frac{1}{\mathcal{L}} \sum_{i=1}^{n} f(\mathbf{m}, \mathcal{S}_i)\ a_i \tag{3}$$

where $\mathcal{S}_i$ is the i^{th} child and a_i is the value of the attribute defined for the i^{th} child.

This technique can be used for interpolating attributes of any kind across the surface of an implicitly defined model (eg transparency, specularity, etc.), as long as the type of the data can be computed through a linear combination. In the next section we describe how to extend this method to allow for the interpolation of attributes which are dependent on a two dimensional mapping.

4 Texture as Attribute

In the previous section field dependent interpolation of surface defined attributes was described. In his comprehensive survey Paul Heckbert enumerated the uses of texture mapping [9]. Chief among them was the generation of more visually satisfying images. To extend the technique described in section 3.2 we will provide one method for implicit surfaces to access these uses of texture mapping

4.1 Mappings for Single Skeletons

The extension of interpolating surface attributes to mapped attributes necessitates the definition of the

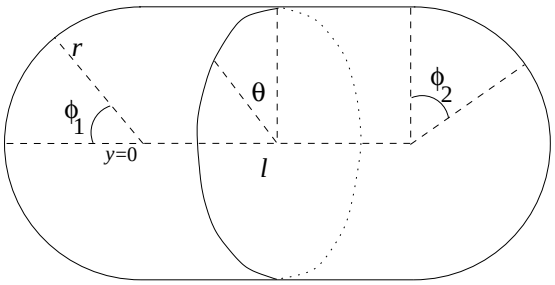

Figure 3: Parameteric representation for the surface of a line skeleton.

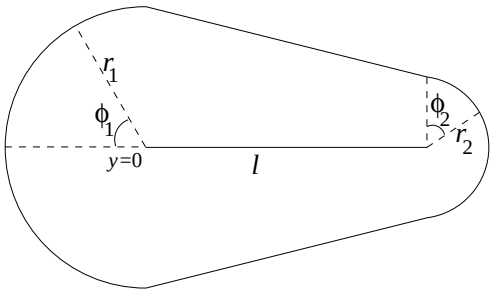

Figure 4: Parameteric representation for the surface of a cone skeleton.

mappings for each of the skeletal elements of a model. We therefore begin by briefly describing the definition of the two dimensional mappings for the standard skeleton types. In some cases it is trivial, in others there are slight wrinkles.

4.1.1 Point

A point skeleton generates a spherical field, the resulting surface of which is a sphere. Any standard mapping for a sphere can be used (see [15] for such a mapping).

4.1.2 Line

A line skeleton generates a cylinder with hemispherical caps. A suitable two dimensional mapping can be found as shown in the following equations.

$$u = \frac{\theta}{2\pi} \tag{4}$$

$$v = \begin{cases} \phi_1 \frac{d}{\pi/2} & y \in [-r \ldots 0] \\ d + \frac{y}{l}(1-2d) & y \in [0 \ldots l] \\ 1 - d + \phi_2 \frac{d}{\pi/2} & y > l \end{cases} \tag{5}$$

$$d = \frac{\pi/2 \ r}{1 + \pi \ r}$$

Figure 3 shows the setup of the l, r, θ and ϕ_1, ϕ_2 variables. The line of the skeleton is assumed to be $y \in [0 \ldots l]$, which implies y coordinates for points on the surface are $y \in [-r \ldots (l + r)]$.

4.1.3 Cone

The parametric definition for a cone is very similar to that for a line. The surface generated for a cone skeleton is shown in figure 4, the only difference from the line skeleton is that there are differing radii for the hemispherical caps. This is due to the fact that the skeleton is not really a cone at all, rather it is a line with a radius which varies linearly along the length of the line. The axis of the cone is assumed to be $y \in [0 \ldots l]$, which implies y coordinates for points on the surface are $y \in [-r_1 \ldots (l + r_2)]$. This changes the formulation for the v coordinate to that shown in equation 6.

$$v = \begin{cases} \phi_1 \frac{d_1}{\pi/2} & y \in [-r_1 \ldots 0] \\ d_1 + \frac{y}{l}(1 - d_1 - d_2) & y \in [0 \ldots l] \\ 1 - d_2 + \phi_2 \frac{d_2}{\pi/2} & y > l \end{cases} \tag{6}$$

$$d_i = \frac{\pi/2 \ r_i}{l + \pi \ r_i}$$

4.1.4 Poly-Line

A poly-line skeleton is a set of n-lines defined by an ordered set of $n + 1$ vertices. Please see appendix **??** for the definition of the field generated by a poly-line.

4.1.5 Circle

A circular skeleton, produces a toroidal surface, which can be described parametrically. With this surface we see an advantage of local definition of the mapping over previous methods of placing a parametric surface around the implicit surface. With a skeletal element which generates an implicit surface

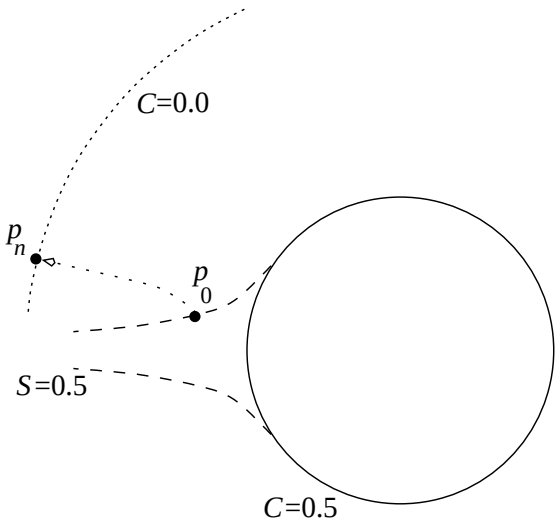

Figure 5: Trajectory illustration to map a point p_0 on the $\mathcal{S}$ iso-surface to p_n to allow computation of a mapping for the $\mathcal{C}$ skeleton.

whose topology does not match the parametric surface which bounds it, it is difficult to find the *appropriate* placement of the parametric surface. The parametrization in our method can be defined in any way desired. We choose u to be the radial angle in the plane the circle lies in and v to be the radial angle of the point in the cross section of the torus.

4.2 Mapping Blends

The remaining task is to define how to find texture coordinates in an area where blending is taking place. We describe how to map values from points p on an $\mathcal{L} > 0$ level surface of a skeletal element to points on the $\mathcal{L} = 0$ level surface of the same skeleton.

Let $\mathcal{C}$ be the field due to a single skeletal element composite implicit surface $\mathcal{S}$. Then we can compute a texture coordinate for a point p_0 on $\mathcal{S}$ in the field of $\mathcal{C}$ by stepping through a vector field from p_0 to a point p_n on the $\mathcal{L} = 0$ level surface of $\mathcal{C}$. Figure 4.2 shows the location of a p_n and hence an associated (u, v) coordinate for p_0. The vector field is an interpolation of the gradient of $\mathcal{S}$ and the gradient of $\mathcal{C}$, the formulation for stepping through this vector field is given by equation 7.

$$\begin{aligned} k &= f(\mathcal{C}, p_i) \\ p_{i+1} &= p_i + \\ &\quad \delta\ (1.0 - k)\ \nabla f(\mathcal{C}, p_i) + \\ &\quad \delta\ k\ \nabla f(S, p_i) \end{aligned} \tag{7}$$

Equation 7 describes a sequence of points p_i which together form a trajectory through the level surfaces of the skeletal element C. Iteration of this formula should procede until the $\mathcal{L} = 0$ level surface is found. Standard numerical techniques can be used to find an adaptive step size for δ [6]. A description of implementation of adaptive stepping in a very similar situation is given in [14]. Once p_n on the $\mathcal{L} = 0$ level surface is found it can be used to find (u, v) coordinates for p_0, and hence an associated surface attribute value for $\mathcal{C}$ can be found by lookup in a two dimensional texture map.

An example of this mapping for a two dimensional composite implicit surface is shown in figure 6. Notice in this figure that for many of the trajectories the path is simply a straight line. This is the case for all types of primitives for points on $\mathcal{S}$ which are

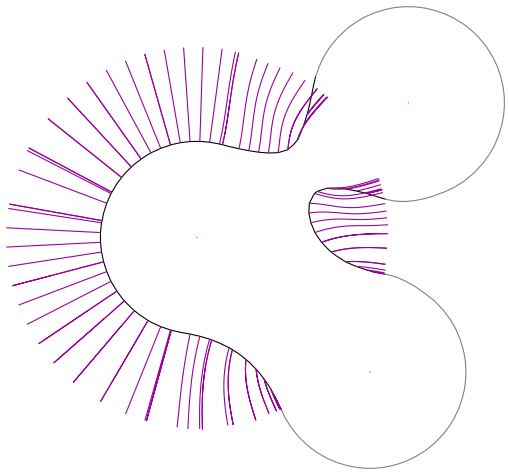

Figure 6: Mapping for an individual of a composite implicit surface. The surface $\mathcal{S}$ is the outline of the black curve surrounding the point skeletons. The element $\mathcal{C}$ for which a mapping is computed is the point skeleton surrounded by the trajectories

due solely to $\mathcal{C}$. These trajectories are shown in

figure 6 for illustration only. They are not generally computed as they are not needed.

This formulation has much in common with techniques described in [13, 19]. The difference with these methods is that they were designed to provide a global mapping for a single composite implicit surface. The basic method was extended in [21] to include surfaces which contain articulation of jointed skeletons. The method described here allows these techniques to be more easily applied to a single element of a composite skeletal implicit surface. In fact in [13] an extension was introduced that allows the method to be applied to anywhere in a hierarchy of primitives and blends in a coherent fashion. The definition of a surrounding parametric surface is required to associate a mapping with an individual of the hiearchy. This is difficult to define for the use and can be computationally expensive. In the method presented here the mapping is associated directly with the elements making up the composite surface. The attributes (texture, bump, etc.) due to the mappings are blended in exactly the same framework as already exists with the modelling system. This makes the modeling task required for the application of texture maps consistent with methods for other kinds of attributes.

The method presented in this section is also advantageous for efficiency. The mapping from points on the composite implicit surface to the $\mathcal{L} = 0$ surface only needs to be performed if necessary. For a comparison of efficiencies see the results in section 5. This is effectively done by checking the field of other primitives along the straight line trajectory to the $\mathcal{L} = 0$ surface, an effecient method for this is to use interval analysis. Compute the field value for the sum of the other skeletal elements for the surface over the interval of the default trajectory, if it contains zero then we need to perform the blend mapping. If the point is due to multiple primitives then iteration of equation 7 is required.

Of course if the point on the surface is not on the iso-surface used to determine the total surface then the interval analysis step can be ignored, as it is obviously part of a blend.

5 Results

In this section we present some visual results. Comparisons with previous methods are shown.

The first example shows the result of blending two mappings from different geometries. In figure 7(c) we show the result of blending a line skeleton to a point skeleton, the result shows the smooth blending of the mappings across the geometry of the blend. In figure 7 we show two different mappings produced by the previous algorithm from [19, 13]. Figure 7(a) uses a cylindrical parametric surface which surrounds the non zero portion of the scalar field, the orientation of the cylinder is with the axis of the blend between the line and the sphere. Figure 7(b) uses two parametric surfaces. Each of the primitives has its surface surrounding its scalar field. Both of the parametric surfaces are cylinders.

Figure 7 shows the same model textured using three different methods. The results indicate that the new algorithm presented in section 4 produces very similar mappings. The advantage is the efficiency. Table 1 outlines the relative time spent producing the mappings for the images. This indicates the amount of time the renderer spent iterating equation 7 in relation to the amount of time it spent rendering the image. It should be noted that the value of $\triangle$ was reduced for these comparisons to an extremely small number. This caused the trajectory calculations to be extremely slow. This was done to ensure that the time taken to compute the trajectories was longer than the resolution reported by the `times` standard C library function. Also we did not adaptive steps for these experiments.

Figure	$\frac{t_{texturing}}{t_{rendering}}$	$\frac{t_{textures}}{t_{notextures}}$
7(a)	0.95	25.78
7(b)	0.82	6.21
7(c)	0.27	1.52

Table 1: Comparison of efficiencies. The table shows for each image in figure 7 the ratio of the amount of time to perform texturing versus the time to render the image, and the time to render the image versus the time to render the same image with no texturing.

The results in table 1 seem at first glance wrong. In the case where a single parametric surface, figure 7(a), was used the texturing algorithm took the

(a) Previous algorithm with a single parametric surface.

(b) Previous algorithm with separate parametric surfaces.

(c) New algorithm with parametrization defined on the primitives.

Figure 7: Visual comparison of the results. It can be seen that the result from the new algorithm is visually very similar to using separate parametric surfaces for the primitives.

majority of the time of the image generation. With two parametric surfaces, figure 7(b), the texturing time vs. rendering time ratio is greatly reduced. This is because of the nature of the model, the parametric surface did not fit the model very well in figure 7(a), and hence a great many trajectory calculations took a very long time. Each of the two parametric surfaces used in figure 7(b) fit their respective primitives very well and so the trajectories were much shorter, yielding a faster image rendering despite more trajectory calculations. However, the main point of the results is that the algorithm presented in this paper greatly speeds image generation.

The second example, figure 8, shows the use of the techniques presented here in a more practical scene. The coffee cup is constructed through a hierarchy of CSG, warping and blending according to algorithms described in [16]. The texture map of the Great Train Rubbery star is applied to a line skeleton.

The third example, figure 5 shows the combination of both techniques for texture mapping implicit surfaces. The seashell is built through controlled blending [8] of a sequence of blends simulating the spiral growth of the shell. Each of these areas are textured with a single parametric surface surrounding it. The spikes and bumps are textured with the algorithm presented in this paper. The combination of these techniques succeeds in providing a visually appealing texture mapping over the entire surface. All of the texture maps used are very subtle but they make the surface more visually interesting.

These two models were both built using the *Blob-Tree*hierarchical data structure as described in [16].

Figure 9: Combination of both methods of texturing implicit surfaces in the model of a Cabrit's Murex seashell.

6 Conclusion

In this paper an algorithm has been presented which unifies previous work on texture mapping implicit surfaces with a classical technique used often during modeling with implicit surfaces. The work provides a framework for the incorporation of gradient trajectory calculation methods for computing mappings

Figure 8: Texture mapping an implicit coffee cup.

into standard interpolation methods of defined skeletal surface attributes.

The main advantage of this model is the elimination of user interaction and the elimination of parameters necessary to compute the mapping. Also advantageous is the realization that both previous methods and the algorithm described in this paper can easily co-exist in the same system as used in the sea shell model, figure 5. The default method in our system is the algorithm described here, if the user wishes to apply a global mapping to a set of blended skeletal elements the techniques presented in [13, 21] can be used.

Acknowledgements

The authors are indebted to Callum Galbraith for providing the model of the seashell. The authors would like to express gratitude for the support and use of the MACI cluster of workstations for high performance computing. We also thank Gladimir Baranoski and the referees for their careful reading and helpful suggestions. This work is partially sponsored by the Natural Sciences and Engineering Research Council.

References

[1] Eric Bier and Kenneth R. Sloan. Two part texture mappings for ray tracing. *IEEE Computer Graphics and Applications*, 6(9):40–53, September 1986.

[2] James F. Blinn. Texture and reflection in computer generated images. *CACM*, 19(10):542–547, October 1976.

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

[4] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.

[5] Ed Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Dept. of CS, Univ. of Utah, 1974.

[6] Stephen Nash David Kahaner, Cleve Moler. *Numerical Methods and Software*. Prentice-Hall, 1989.

[7] Luiz Henrique de Figueredo and Jonas Gomes. Sampling implicit objects with physically-based particle systems. *Computer & Graphics*, 20(3):365–375, 1996.

[8] Andrew Guy and Brian Wyvil. Controlled blending for implicit surfaces. In *Implicit Surfaces '95*, April 1995.

[9] Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[10] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Trans. IECE Japan, Part D*, J68-D(4):718–725, 1985.

[11] Hans Køhling Pedersen. Decorating implicit surfaces. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 291–300. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[12] Andrei Sherstyuk. Fast ray tracing of implicit surfaces. In *Proceedings of the Third Eurographics Workshop on Implicit Surfaces*, pages 145–154, Seattle, June 1998.

[13] Mark Tigges and Brian Wyvill. Texture mapping the blobtree. In *Proceedings of the Third Eurographics Workshop on Implicit Surfaces*, pages 123–130, Seattle, June 1998.

[14] Kees van Overveld and Brian Wyvill. Potentials, Polygons and Penguins. An efficient adaptive algorithm for triangulating an equi-potential surface . In *Proc. 5th Annual Western Computer Graphics Symposium (SKIGRAPH 93)*, pages 31–62, 1993.

[15] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. ACM Press, 1992.

[16] Brian Wyvill, Eric Galin, and Andrew Guy. Extending the csg tree - warping, blending and boolean operations in an implicit surface modeling system. In *Proceedings of the Third Eurographics Workshop on Implicit Surfaces*, June 1998.

[17] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[18] G. Wyvill, B. Wyvill, and C. McPheeters. Solid texturing of soft objects. *IEEE Computer Graphics and Applications*, 7(12):20–26, December 1987.

[19] R. Zonenschein, J. Gomes, L. Velho, and L. H. Figueiredo. Texturing implicit surfaces with particle systems. In *Technical Sketch in Visual Proceedings SIGGRAPH 1997*, page 172, August 1997.

[20] Ruben Zonenschein, Jonas Gomes, Luiz Velho, and Luiz Henrique de Figueredo. Controlling texture mapping onto implicit surfaces with particle systems. In *Proceedings of the Third Eurographics Workshop on Implicit Surfaces*, pages 131–138, Seattle, June 1998.

[21] Ruben Zonenschein, Jonas Gomes, Luiz Velho, Luiz Henrique de Figueredo, Mark Tigges, and Brian Wyvill. Texturing composite deformable implicit objects. In *Proceedings of the XI International Symposium on Computer Graphics, Image Processing and Vision*, pages 346–353, Rio de Janeiro, October 1998.