Implicit Surfaces: Appearance, Blending and Consistency

Agata Opalach
Steve Maddock

Department of Computer Science, The University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield, S1 4DP, UK
Tel: +44 742 825579 Fax: +44 742 780972
E-mail: a.opalach@dcs.shef.ac.uk

Abstract

The implicit surface modelling technique can be used to model “soft” objects, objects whose surface or topology changes over time. It offers the opportunity of modelling and animating relatively complex shapes using only a few primitives. The final object is constructed by blending the primitives, and as the primitives are moved the blended surface changes shape. This can cause problems with object appearance and consistency and unwanted blending of object parts. In this paper we describe work in progress which addresses these problems.

1. Introduction

Implicit surfaces are often overlooked in computer graphics and animation because other techniques such as polygonal or parametric surfaces offer more established ways of visualising modelled objects. On the other hand objects being modelled often change their shape over time. The piece-wise nature of polygonal or parametric surfaces imposes the need for tedious polygon or patch fitting to maintain surface continuity and smooth blending. Implicit surfaces are always continuous and smoothly blended, thus more suitable for modelling such “soft” objects.

An implicit surface can be defined as all points P where

$$F(P) - Iso = 0 \qquad \text{(EQ 1)}$$

F(P) is called the *scalar field function* (or the *density distribution function*) and *Iso* is the *iso-value* (or the *threshold value*) at which an iso-surface is extracted (Figure 1). A simple example of a scalar field function is an implicit definition of a sphere $x^2 + y^2 + z^2 - r^2 = 0$. In this case, the scalar field function $F(P) = x^2 + y^2 + z^2$ where $P = (x, y, z)$ and the iso-value corresponds to the radius squared (r^2) of the sphere. We will call an implicit surface defined by a particular scalar field function a *primitive*. If we want an implicit surface consisting of multiple primitives we have to combine their scalar field functions. This process is called *blending*. It defines a method for calculating the overall scalar field value (contributed from all primitives) at a given point P, e.g. summation of contributions from individual primitives. If we define a distance R at which the scalar field function of a primitive is zero (*radius of influence*) we can discard the influence of primitives at a distance larger than R from P. By choosing various field functions and blending techniques we can model relatively complex implicit surfaces with varying levels of smoothness.

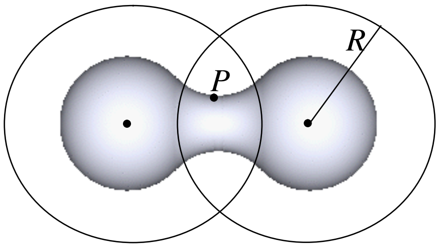

FIGURE 1 Two blended primitives

The computer graphics work on using implicit surfaces for modelling and animation includes [Blinn, 82] (blobby molecules), [Nishimura et al, 85] (metaballs), [Wyvill et al, 86], [Wyvill and Wyvill, 89] (soft objects), [Bloomenthal and Shoemake, 91] (convolution surfaces), [Bloomenthal and Wyvill, 90], [McPheeters, 90], [Gascuel, 93].

The smooth blending of primitives is a powerful property of the technique yet the constant surface change causes problems with animating implicit models. In order to preserve an object's identity during animation, primitives have to be moved in a controlled way. [Bloomenthal and Shoemake, 91] clothe a skeleton of curves and polygons with an implicit surface. An implicit surface always fits the shape of a skeleton. [Wyvill and Wyvill, 89] build a hierarchy in an object to define its appearance. They use it to metamorphose between two objects by matching their hierarchies.

Preventing blending between primitives is also recognised as a problem with the technique. [Wyvill and Wyvill, 89] solves it by "hierarchical clothing". Primitives which do not blend with each other are rendered separately and then composed to produce the final image. If scalar fields of unblendable objects overlap the objects will intersect. The metaballs technique solves undesired blending by introducing eccentric metaballs (as opposed to concentric spheres) [MetaEditor]. Two metaballs will blend (or fuse) only if their centres of density are close enough. Eccentric metaballs have their centres of density shifted which helps to avoid blending (called meta-fusion). [Gascuel, 93] uses physically based modelling to achieve a precise contact between unblendable implicit solids. Her algorithm detects collision between flexible objects and simulates forces causing their local deformation.

Implicit surfaces have already found practical applications. For example [Tatsumi et al., 90] presents the use of metaballs in the process of modelling a cod's liver, [Muraki, 91] uses implicit surfaces for volumetric shape description (including human face modelling), [Max and Wyvill, 91] renders corals as "soft objects" and [Payne and Toga, 92] models a rat's brain using "distance fields". The growing popularity of the technique suggests that we should keep improving solutions for the many problems that occur in using it.

2. Problems To Be Solved

The problems we want to focus on are related to the animation of implicitly defined objects which do not metamorphose during motion. They still change their shape but their topology remains invariable. We can categorise these problems into appearance, blending and consistency, although they are interrelated. For appearance, we wish an object to retain its general shape when moving between two key positions. Providing automatic appearance maintenance allows for complex deformations to be performed on an object while it remains recognisable. For instance, a human arm should not become a “blob” when it is moving from one position to another. Although smooth blending is an advantage of the technique we can see that in this case it is not desirable.

This leads us to the problem of unwanted blending. Consider our human arm again. We want the elbow to blend with both the lower arm and the upper arm but we want to avoid the upper arm blending with the lower arm. It is worth pointing out that such unwanted blending may be required in some cases as we may want an object to melt or metamorphose into another object. But unless we explicitly request the shape change, we want our object to preserve its character throughout the animation.

For consistency, we would prefer that our objects did not break into pieces during an animation sequence. This can easily happen if the object is composed of multiple primitives. If one primitive is moved too far apart from the others the radius of influence is breached and the iso-surface will consist of separate parts.

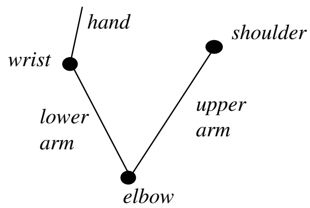

FIGURE 2 A skeleton which defines an arm

3. Some Solutions

3.1 Appearance

In order to preserve the appearance of an object composed from multiple blended primitives, one solution is to impose a structure on the primitives. Our models are built according to a graph which defines a skeleton within an object in terms of joints and connecting links. A sample skeleton specifying a simplified arm is shown in Figure 2. The upper arm, lower arm and hand are links and the shoulder, elbow and wrist are joints. Each joint and each link has a set of primitives assigned to it and at present the primitives are not allowed to break apart, although they

a)

b)

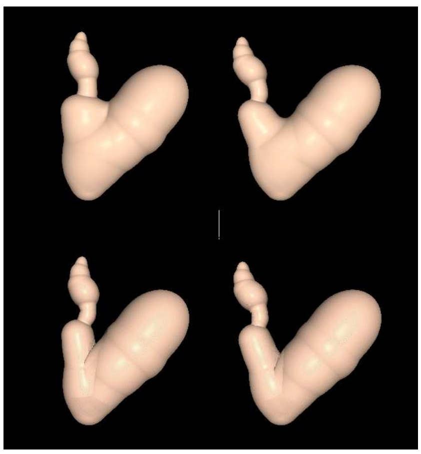

FIGURE 3 A model of an arm being bent
a. Unwanted blending - the upper arm blends with the lower arm
b. Unwanted blending is prevented

can stretch (provided the consistency is preserved). For the arm in Figure 2, the shoulder is assigned one primitive, the upper arm two, the elbow one, the lower arm two, the wrist one and the hand four.

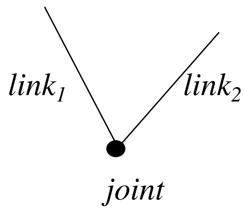

FIGURE 4 The skeleton defining the objects in Figure 5

The movement of an object is specified in terms of transforming joints and links. The primitives attached to a relevant joint or link all undergo the required transformation (but not necessarily at the same time) resulting in the specified movement. In the skeleton of Figure 2, the lower arm, wrist and hand are rotated around the elbow to give the animation shown in Figure 3.

The idea of using a skeleton is not new [Wyvill and Wyvill, 89, Bloomenthal and Shoemake, 91] but the novelty of our approach is that it does not restrict the spatial relationship of joints and links. For example, Figure 4 shows a simple skeleton which was used to model two objects in Figure 5. The link $link_1$ and joint *joint* are in both cases assigned one primitive. The link $link_2$ is assigned one primitive in the top row and three primitives in the bottom row. Introducing new primitives to an object may be useful for modelling extreme stretching, e.g. extending an arm to infinity - an essential cartoon effect!

3.2 Blending

Figure 3a shows an example of unwanted blending. As the arm is bent, the upper arm blends with the lower arm. In order to prevent this, we use the concept of blending types (b-types). A primitive is assigned a b-type and two primitives with the same b-type will blend. Primitives with different b- types may or may not blend. Using the skeletons described above we assign each joint and link (and therefore each primitive they consist of) a different b-type and let joints blend with relevant links but do not allow links or joints to blend between each other. This information can be automatically extracted from a skeleton but we also provide a means of specifying blending properties explicitly in order not to restrict modelling possibilities (since we may well want to model a creature whose limbs actually blend with its body).

Again the idea of various types of primitives is not new [Wyvill and Wyvill, 89]. However, whereas Wyvill solves the problem by overlapping in the intersection areas, we want to achieve contact between unblended primitives. Rather than overlapping, the primitives “squeeze” against each other (see Figure 6). We also deal with the problem of an object which has both blendable and unblendable parts. This is different from the case of separate interacting objects when they either completely blend with each other or not at all (as in [Gascuel, 93]).

Assigning blending properties obviously does not solve the unwanted blending problem but only helps to specify it. Having identified the b-types of primitives (either explicitly or implicitly using the skeleton) we still have to find a way of using this information during the extraction of an iso- surface.

The algorithm for extracting a correct iso-surface is similar to the one presented in [Gascuel, 93] yet it serves cases when both blendable and unblendable primitives influence a particular point.

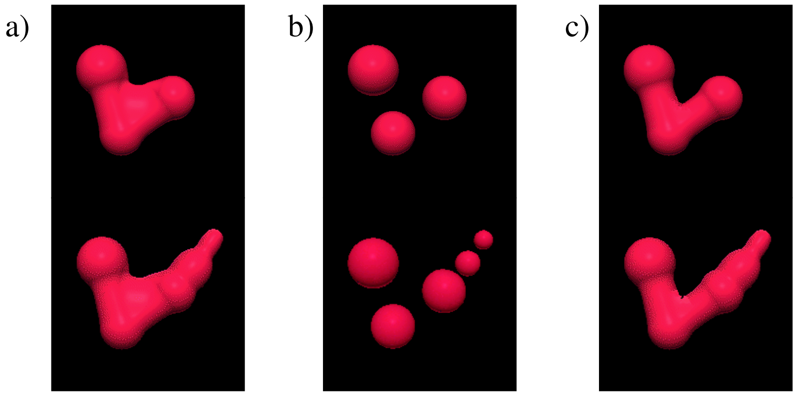

FIGURE 5 Two objects built on the skeleton from Figure 4 with various blending properties
a. all primitives are blendable
b. all primitives are unblendable
c. the joint blends with the links but the links do not blend with each other

3.2.1 All primitives blendable

If all primitives influencing a point P are blendable the overall field function value at P, $F(P)$, is calculated by summing the contributions from all influencing primitives.

$$F(P) = \sum_{i \in Influencing} f_i(P) \qquad \text{(EQ 2)}$$

The result is a blended surface as shown in Figure 5a.

3.2.2 All primitives unblendable

When two primitives do not blend with each other, we want them to approach each other without blending until they get in contact and start to deform. The contact and deformation of primitives will not occur immediately when their scalar fields overlap. The distance at which contact takes place (a radius of contact) depends on the chosen iso-value and is less than or equal to the radius of influence. It can be calculated from the field function by finding the distance at which the scalar field value equals Iso.

In order to calculate the field value at a point P we first take the field function value from the primitive whose contribution is highest at P. For all other influencing primitives, if P is within their radius of contact we need to consider a deformation caused by them. We define it by the contraction function c_i. It has to be negative to “squeeze” the other primitive. We use

$$c_i(P) = Iso - f_i(P) \qquad \text{(EQ 3)}$$

where Iso is the iso-value at which an iso-surface is to be extracted. $c_i(P)$ is negative within the radius of contact, hence the field value at P decreases causing deformation.

The final field value at P is

$$F(P) = f_{i_0}(P) + \sum_{\substack{i \in Contracting \\ i \neq i_0}} c_i(P) \qquad \text{(EQ 4)}$$

$$f_{i_0}(P) = max_{i \in Influencing} f_i(P)$$

Figure 5b shows an example of interaction between unblendable primitives. For two unblendable primitives in contact (Figure 6) we have

$$F(P) = f_1(P) + Iso - f_2(P) \qquad \text{(EQ 5)}$$

Considering the iso-surface definition (EQ 1) we get an exact contact surface at points *P* where

$$f_1(P) = f_2(P) \qquad \text{(EQ 6)}$$

which justifies our choice for the contraction function (3).

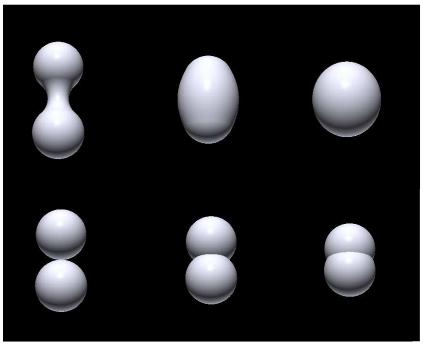

FIGURE 6 Two primitives approaching each other
a. when they blend
b. when they do not blend

3.2.3 Blendable and unblendable primitives

When a point *P* is influenced by primitives which blend with themselves and primitives which do not blend the principle of calculating the field value at *P* is similar to case (ii). This time we find a set of blendable primitives whose sum of contributions is highest at *P* (*MaxBlend*). The summed contribution is added to the field value at *P*. All other primitives are unblendable with the selected set. Among them we identify a set of primitives for which *P* is within their radius of contact (*Contracting*) and add the contraction function at *P* for primitives from this set. Thus the final field value at *P* is given by:

$$F(P) = \sum_{i \in MaxBlend} f_i(P) + \sum_{i \in Contracting} c_i(P) \qquad \text{(EQ 7)}$$

Figure 5c shows the case when an object is constructed according to the skeleton from Figure 4 and the joint blends with its links while the links do not blend with each other.

In the above algorithm (for cases in Section 3.2.2 and Section 3.2.3) volume is not conserved and there is no mass propagation outside the contact area. This makes squeezing between primitives look unnatural (plasticine appearance). We believe that it will be easier to describe, determine and solve this problem when we subject implicitly defined objects to the forces (e.g. gravitation, acceleration) in an environment containing other flexible and rigid objects. [Gascuel, 93] proposes an elegant method of calculating local deformations occurring during collisions between flexible objects which preserves their C^1-continuity.

3.3 Consistency

We have imposed consistency constraints on the spatial relationship of the primitives. We can make sure objects do not break apart but also that they do not appear rigid and unnatural. They stay consistent yet can change their shape slightly during motion. The maximum distance to which two primitives can be dragged apart can be calculated from their field functions. Two primitives are blended when

$$\exists P\,(f_1(P) + f_2(P) \geq Iso) \qquad \text{(EQ 8)}$$

Using this information and considering the scalar field function distribution along the line connecting the centres of two primitives we can detect when the primitives are moved too far from each other. Let us consider primitives with centres C_1 and C_2. A point P between C_1 and C_2 can be represented by

$$P = C_1 + t(C_2 + C_1), \qquad 0 \leq t \leq 1 \qquad \text{(EQ 9)}$$

Taking the overall field function for two blendable primitives

$$F(P) = f_1(P) + f_2(P) \qquad \text{(EQ 10)}$$

We can substitute P in $F(P)$ in the iso-surface definition (EQ 1) giving

$$f_1(C_1 + t(C_2 + C_1)) + f_2(C_1 + t(C_2 + C_1)) = Iso \qquad \text{(EQ 11)}$$

If (EQ 11) has a solution between C_1 and C_2 primitives cannot be moved any further (we assume that the considered model is consistent when motion starts).

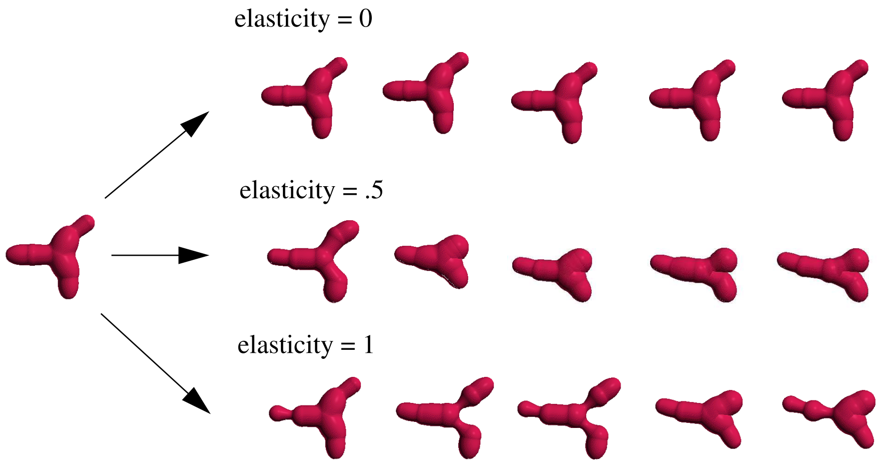

FIGURE 7 An object being dragged to the left hand side by one primitive for various values of the attribute elasticity a. elasticity = 0.0 b. elasticity = 0.5 c. elasticity = 1.0

Solving equation (EQ 11) each time a movement is detected is a very expensive operation. Therefore we approximate the maximum permitted distance between primitives before creating an animation sequence for a given model.

Detecting a loss of consistency is used to trigger the other primitives to follow the one which is moving. To provide a user with control over the way an object preserves its consistency we introduce an attribute called elasticity. It can be assigned to a link or a joint and decides when primitives start to follow the motion. It can be varied from 0 (immediate following, rigid appearance) to 1 (following only if the maximum allowed distance is reached, fluid motion). In the future we plan to introduce forced stretching as a possible response to consistency loss (with adding new primitives if necessary).

Figure 7 shows an object being dragged by one primitive for various values of the elasticity attribute. In Figure 7a elasticity is 0, hence the object does not change its shape while it moves. Figure 7b and Figure 7c show the same object moving for elasticity values 0.5 and 1.0 which models more fluid motion.

4. Conclusions

We have presented a few solutions to some known problems of the implicit surfaces modelling technique. Our early ideas have produced some fairly pleasing results especially with solving the unwanted blending problem in the case of an object which has both blendable and unblendable parts. In the future, we plan to develop more realistic collision handling between unblendable primitives, natural ways of preserving consistency in the case of complex motion (e.g. dragging parts in opposite directions) and implementing automatic metamorphosis between objects to achieve a smooth appearance transfer from one to another.

5. References

1. Blinn, J. F., “A Generalisation of Algebraic Surface Drawing”, ACM Trans. Graphics, Vol. 1, No 3, July 1982, pp 135-256

2. Bloomenthal, J. and Shoemake, K., “Convolution Surfaces”, Computer Graphics, Vol. 25, No 4, July 1991, pp 251-256

3. Bloomenthal, J. and Wyvill, B. “Interactive Techniques for Implicit Modelling”, Computer Graphics, Vol. 24, No 2, 1990, pp 109-116

4. Gascuel, M.-P., “An Implicit Formulation for Precise Contact Modelling between Flexible Solids”, Siggraph 93, to appear

5. Max, N. L. and Wyvill, B., “Shapes and Textures for Rendering Coral”, Scientific Visualisation of Physical Phenomena, N. M. Patrikalakis (Ed.), Springer-Verlag 1991, pp 333-343

6. McPheeters, C. W., “Isosurface Modelling of Soft Objects in Computer Graphics”, 1990, National Centre for Computer Animation, Department of Communication and Media, Dorset Institute, PhD thesis

7. MetaEditor, “Operation guide”, personal communication from Bruno Tsuchiya, META Corporation, Japan, July 1993

8. Muraki, S., “Volumetric Shape Description of Range Data Using “Blobby Model””, Computer Graphics, Vol. 25, No 4, July 1991, pp 227-235

9. Nishimura, H., Hirai, M., Kawai, T., Kawata, T., Shirakawa, I. and Omura, K., “Object Modelling by Distribution Function and a Method of Image Generation”, The Transactions of the Institute of Electronics and Communication Engineers of Japan, 1985, Vol. J68-D, Part 4, pp. 718-725, in Japanese

10. Payne, B. A. and Toga, A. W., “Distance Field Manipulation of Surface Models”, IEEE Computer Graphics and Applications, January 1992, pp 65- 71

11. Tatsumi, H., Takaoki, E., Omura, K. and Fujito, H., “A new method for 3D reconstruction from serial sections by computer graphics using “meta-balls”: reconstruction of “Hepatoskeletal System” formed by Ito Cells in the Cod Liver”, Computers and Biomedical Research, 1990, Vol. 23, Part 1, pp 37- 45

12. Wyvill, G. and McPheeters, C. and Wyvill, B., “Data Structure for Soft Objects”, Visual Computer, Vol. 2, No 4, August 1986, pp 227-234 Wyvill, B. and Wyvill, G., “Field Functions for Implicit Surfaces”, Visual Computer, Vol. 5, 1989, pp 75-82