Chapter 1

Introduction

1.1. A BRIEF HISTORY OF IMPLICIT SURFACES AND A LITERATURE REVIEW

It has been almost two decades since James F. Blinn put two Gaussian functions together and produced one dumbbell-shaped object that he called a *blobby molecule*. The first implicit blend of surfaces in computer graphics was born [7].

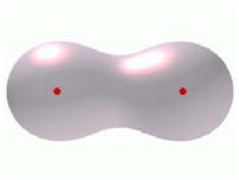

Figure 1.1: Blinn's Blob.

Almost immediately, the blobby molecule grew to an impressive size of over 4,000 atoms to present the DNA molecule in Carl Sagan's COSMOS TV Series.

At approximately the same time, Japanese researchers at Osaka University and Matsushita Electronic Industrial Co. developed a system for modeling objects with density distributions approximated by piecewise quadratic polynomials, better known as *metaballs* [48, 49, 50]. Unlike Blinn's blobby molecules, which were invented to represent blending shapes, the development of metaballs was largely motivated by the need of making smooth metamorphoses in animations. Computer graphics artist Yoichiro Kawaguchi used the metaballs system, running on a parallel computer LINKS-1 to produce the animated film *Growth: Mysterious Galaxy* [39]. *Growth* became a big success at the SIGGRAPH film show in 1983 and in years to follow.

> ...It was completely different from previous computer imagery in that the objects were comprised not of inorganic lines, but of living, curved surfaces. When the reaction to *Growth* was one of raised goosepimples, as though reacting to a nightmare, I knew that *Growth* was a success!
> *Yoichiro Kawaguchi* [41] [1] .

What makes implicit modeling so successful? Why do objects, produced with this method look like '...living, curved surfaces'? Why are they called implicit in the first place?

Implicit surfaces and implicit modeling received their name after the method the surfaces are defined. A surface S may be defined as a set of points $\mathbf{p} = (x, y, z)$

$$\begin{aligned} \text{parametrically:} \quad & S = \mathbf{p} \mid x = x(t), y = y(t), z = z(t)\}, \\ \text{explicitly:} \quad & S = \mathbf{p} \mid z = F(x, y)\}, \\ \text{implicitly:} \quad & S = \mathbf{p} \mid F(x, y, z) = 0\}. \end{aligned}$$

It is convenient to think of an implicit surface as an isosurface, formed in a scalar field $F(x, y, z)$ at certain threshold T:

$$S = \{(x, y, z) \mid F(x, y, z) = T\}$$

The field function F is often presented as

$$F(x, y, z) = \sum_{i=1}^{N} f_i(x, y, z) \tag{1.1}$$

The summation is performed over all constituent functions $f_i(x, y, z)$, each of which defines a density distribution in 3D-space. For example, the object pictured in Figure 1.1 is modeled by a sum of two Gaussian distributions

$$e^{-a_1 r_1^2(x,y,z)} + e^{-a_2 r_2^2(x,y,z)} = T,$$

where r_1 and r_2 are distances from the centers of the first and the second component and a_i are blending factors. If the modeling functions $f_i(x, y, z)$ are continuous and smooth, the resulting surface will exhibit seamless blends between its constituent parts. Changes in number, position or properties of the modeling components will cause changes in the resulting shape.

By adjusting these parameters, a designer controls all regions of the surface.

Density distribution functions $f_i(x, y, z)$, used in the main modeling equation (1.1), may be defined in a variety of ways. Blinn used exponentially decaying

potentials [7]. The metaballs system employed piecewise quadratic polynomials [50]. In the *soft objects* model, introduced by Wyvill et al. [73], the density function is given by a polynomial of degree 6. Another popular function is a quartic polynomial, used in many public domain ray-tracing programs [56, 58]. The exact formulae for all these functions are given in Appendix A. In their basic implementations, all of these modeling functions are spherically symmetric and sigmoidal with respect to the center of the density distribution.

It has been acknowledged that spherically symmetric modeling functions are not adequate for all modeling tasks. For example, flat or cylindrical surfaces cannot be represented by a collection of spherically symmetric components. Similarly, elongated objects like limbs, branches and tentacles also require a better representation. The search for a wider range of implicit shapes took two conceptually different directions [2].

- *Point primitives with anisotropic distance functions.*

  This approach was proposed by Blinn in his seminal paper [7]. He suggested using general quadrics in order to compute the distance to the center of the primitive. Essentially, this method changes the space metric around spherical field functions and makes them look like quadrics: ellipsoids, cones, cylinders, etc.

  Superquadric implicit primitives can also be created this way, as described by Wyvill and Wyvill [78] and Kalra and Barr [38]. Blanc and Schlick [6] presented ratioquadrics, as a computational improvement of the superquadric model. Ratioquadrics and superquadrics are also based on the notion of anisotropic distance function. Many other useful field functions are reviewed in [5] and [78].

  Perhaps the most advanced modeling function in this class is an *implicit sweep object*, described by Crespin et al. [20] [3]. Swept objects are also described by Sourin and Pasko [68] in the context of more general modeling concept, the Function Representation [1, 43, 52].

- *Non-point primitives with isotropic distance functions.*

  With this approach, the modeling implicit functions are built around some non-point geometric objects, typically, a collection of lines, splines and polygons. Each point on such an object is assumed to carry a certain potential charge which decays symmetrically in all directions. To compute the value of a modeling function f_i at some point of interest (x, y, z), the potential may be measured either from the nearest point on the underlying object, or it may be summed from all points that belong to the object. Surfaces, modeled with the former approach are known as *distance surfaces*. They are surveyed in [11], [15] and [16]. Surfaces, modeled with the latter approach are called *convolution surfaces* [12, 13, 19, 42, 60].

Of all the multitude of modeling methods that utilize implicit functions, we have mentioned only those that are often referred to as 'classic implicits'. Essentially, these methods provided the environment in which convolution surfaces evolved, which will be the topic of this dissertation.

1.2. GENERALIZATION OF CLASSIC IMPLICIT MODELS

Convolution surfaces were introduced into computer graphics by Bloomenthal and Shoemake [12] as a generalization of point-based implicit models [7, 50, 73], extended to multi-dimensional modeling primitives. In short, a convolution surface is the implicit surface, based upon distribution functions f_i, each of which is obtained via a 3D-integration

$$f(\mathbf{r}) = \int_{\mathbf{V}} g(\mathbf{p}) h(\mathbf{r} - \mathbf{p})\, d\mathbf{p}, \qquad (1.2)$$

where h and g are *potential* and *geometrical* functions of the modeling primitive, respectively. The integration is performed over the volume $\mathbf{V}$ of the primitive. Equation (1.2) is commonly referred to as the convolution integral; the function f($\mathbf{r}$) is called a convolution field function. In Chapter 2, we will describe their properties in more detail.

[2] It must be noted, that the brief description given below by no means should be taken as a complete classification of modeling functions. To date, implicit modeling have gathered under its roof an immense variety of tools and concepts. For a broader view, the reader is invited to consult [52], where a unified representation of many methods is suggested, called *function representation* or *F-rep*. To demonstrate the width of this representation, consider a modeling task of creating a hair-style for a virtual actor. The F-rep solution for this task involves "...procedurally defined real functions with the use of solid noise, sweep-like technique, offsetting and set-theoretic operations, and non-linear deformations. More details on hair modeling can be found in the paper [67]" (quoted from [61]). The resulting hair looks like real hair (not shown) and nothing like a 'characteristic' implicit object pictured in Figure 1.1. More on function representation may also be found in [1, 43].

[3] An implicit sweep primitive may also be classified as a distance surface (see [11], [15] and [16]), based on some curve in 3D-space. However, the authors [20] presented it as a point source with an anisotropic field function. The modeling complexity is achieved by specifying a 3D-trajectory the point follows and the transformations that its field function undergoes.

Convolution surfaces received a thorough study in a doctoral dissertation by Jules Bloomenthal, "Skeletal Design of Natural Forms" [13] and other works [8, 10, 12, 16] with special attention to modeling and rendering issues. Using convolution as a mathematical apparatus, Bloomenthal developed a technique for creating the smooth, continuous forms that are ubiquitously present in nature.

An alternative formulation and motivation for convolution surfaces is described by Sealy and Wyvill [60]. They examined how convolution technique may be applied for smoothing models that are common in engineering applications.

1.3. OPEN QUESTIONS

There are many aspects of the convolution model that may be significantly improved. These problems fall into a framework, that is characteristic for most modeling techniques used in computer graphics: Formulation, Modeling, Rendering.

- *Formulation.*
  Questions in this category relate to how the method is defined mathematically and applied algorithmically. Topics of interest are: how rigorously the model is developed, is it consistent within to its own parts, does it allow extensions, are there special cases when the model may break down, etc.

- *Modeling.*
  These questions are of a more applied nature. Once a modeling method is developed mathematically, one has to find out how this method can be used for practical tasks. How wide is the application base of the method? How flexible and how intuitive is the process of creating new shapes?

- *Rendering.*
  What algorithms may be used for visualizing the model? What are the trade-offs between rendering speed and accuracy? Is an intermediate representation (e.g., polygonization) required? Are there any additional requirements in storage, etc.

For the convolution surfaces as a modeling concept, all these questions translate into the following concerns.

1.3.1. Formulation

The field functions f_i used in the convolution surface model are defined via spatial integration (1.2). For most modeling primitives (with the exception of points and line segments), the convolution integral (1.2) does not yield closed form solutions [4] easily and requires special methods for its evaluation. These methods normally involve approximation and storage of intermediate results. We will give two examples how the problem of evaluating (1.2) was approached by previous researchers.

Bloomenthal and Shoemake [12] described how to evaluate the convolution integral (1.2) for polygonal primitives. For a potential function h, they used a separable Gaussian function

$$h(x, y, z) = e^{-(x^2+y^2+z^2)/2}$$

which allowed them to compute the convolution integral (1.2) as a product of two convolutions, planar $e^{-(x^2+y^2)/2}$ and perpendicular $e^{-z^2/2}$. However, the intermediate planar convolution (in the polygon's plane $z = 0$) appeared to be too difficult to perform analytically and required a raster approximation, which is performed as follows. First, a polygon is scan-converted into its digital image in the $z = 0$ plane and then convolved with a Gaussian two-dimensional filter, using well-established algorithms from digital signal processing [23]. Then, the point of interest (x, y, z) is projected onto this plane and a planar component of the convolution is obtained from the filtered image. Finally, this value is multiplied by a perpendicular component of the Gaussian function, $e^{-z^2/2}$. The results of the intermediate planar convolution were originally stored as eight-bit integers [12] and were later replaced by real-valued tables to reduce quantization effects [13]. The demands on memory are $O(n^2)$ with respect to image resolution.

Sealy and Wyvill [60] suggested computing the convolution integral (1.2) using volume sampling techniques and tri-linear interpolation between the nodes. They deliberately replaced the actual integration (1.2) by a discrete summation over the model space, aiming for a more general approach. The trade-off for this generality is the usual 'accuracy-vs-storage' problem. In a brute-force uniform voxel representation memory demands are $O(n^3)$, therefore, Sealy and Wyvill used an adaptive octree representation.

> *We would like to emphasize, that at present, the lack of closed form methods of evaluating the convolution integral (1.2) is the major drawback of the convolution surface model. In general, the model requires a raster representation for two dimensional modeling primitives and volume*

[4] When saying 'closed form solution', we mean a function that can be expressed via elementary functions and algebraic operations and its evaluation does not require procedural techniques.

representation for three dimensional primitives. Both representations use point sampling evaluation techniques that may cause undersampling artifacts. Also, they require large volumes of memory for storage, which may become prohibitive for complex scenes.

The necessity of point sampling and storage makes the convolution surface model conceptually discrete, even though its mathematical foundation requires and implies continuity.

1.3.2. Modeling

Due to difficulties in formulation and actual implementation of convolution surfaces, the practical applications of this method did not receive full attention and development. Yet, modeling capabilities of this method are immense.

The convolution surface model provides a continuous description of a volume of space around the object. This information may be used for various modeling tasks, including those related to the objects' appearance and tasks of controlling the objects' behavior as well. A short excursion in school physics will help to illustrate this point.

According to its definition, the value of a convolution integral (1.2) is in fact a total potential $f(\mathbf{r})$, generated by a 3D object at point $\mathbf{r}$. Sometimes, it is convenient to think of $f(\mathbf{r})$ as of an electrostatic potential, because then one can say that the object that generates this potential is "charged" electrically. Alternatively, $f(\mathbf{r})$ may be associated with a gravitational potential field produced by a massive body, or some other force-generating potential field for that matter [5]. In any case, in order to render the object, represented as an isosurface in $f(\mathbf{r})$, the values of the normal vector

$$\mathbf{N}(\mathbf{r}) = -\nabla f(\mathbf{r})$$

must be computed at every point on the surface of the object. Notably, the gradient vector, as shown above (i.e., without normalizing), gives the vector value of the physical force that is associated with the potential. Thus, if one is creating a physically-based animation of interacting solids that are represented by convolution surfaces, one gets the forces at each point of each object for free! That may help greatly in computing trajectories, collisions and even mutual deformations of the objects.

To illustrate, consider the following example. With the help of the convolution surface model, it is easy to simulate, visualize and animate how things may evolve in a gravity field of a cigar-shaped, a cross-shaped or a donut-shaped planet. To compute the gravitational forces of such unusual configurations, it suffices to evaluate the convolution integral and its gradient for the corresponding geometric primitives that represent the basic shape of such planets (which would be a line, a cross and a circle, respectively). Observe, that in order to simulate conventional gravity, a Newtonian gravity potential $1/r$ must be used as the convolution kernel h. Nothing prevents the designer from creating and exploring worlds where gravity obeys a different potential instead, say, a Gaussian potential, or any other distribution that can be used as a kernel with the convolution integral (1.2) [6]. Appendix B shows the possible gravitational potentials of a cigar-shaped object, including Newtonian $1/r$. Antigravity in such imaginary worlds can be achieved by negating the resulting force vector [7].

Even in the domain of more traditional modeling tasks, i.e., for purely geometric shape modeling, convolution surfaces may be employed with more versatility. To date, convolution surfaces are still mostly used to represent smooth forms, either of natural [8, 13] or artificial [19, 60] origin, and are not commonly thought of as tools for creating shapes that exhibit high frequencies.

The application base of convolution surfaces is perceived as being limited to blending and branching structures which is an obvious underestimation of their modeling capabilities.

1.3.3. Rendering

In order to render a convolution surface even for a TV resolution image, the main modeling equation $F(x, y, z) = 0$ must be solved hundreds of thousands times. Solving this equation numerically requires, in turn, millions of evaluations of the convolution integral (1.2) per image. To avoid such computational costs, polygonization is usually employed to convert implicit models into their polygonal representations that can be rendered by any polygonal renderer. In

[5] Temperature is another good example of a scalar field. However, it is not related to a force field that can be imagined as easily as gravitational or electrostatic forces.

[6] As already mentioned, the vector field of a gravitational force may be obtained by computing a gradient of the potential distribution. If this distribution is "flat" at the center of mass, which is the case of the Gaussian function, the gravity forces diminish as the observer gets closer to the center of mass! This describes the situation that can be experienced during a journey towards the center of a planet, when the traveler eventually finds himself in a free-fall when he reaches the center.

[7] We remind that Newtonian gravity is a vector force field $\mathbf{F}$, with its magnitude proportional to $1/r^2$. Gravity potential is a scalar field P and $\mathbf{F} = -\nabla P(\mathbf{r})$, so the gravity potential is proportional to $1/r$.

fact, polygonal representation has become a standard in the computer graphics industry, including implicit modeling. For example, details about modeling and rendering practices in computer graphics production company Pacific Data Images are given in [3].

Two sets of problems are related to polygonal representation of implicit surfaces:

- traditional problems of polygonal surface representation;
- problems specific to implicit surfaces.

Problems of the first kind are common for all models that employ polygons for representing three dimensional objects. These problems are: geometric aliasing [8], high storage demands, the need for interpolative shading techniques, which, in turn, spawns a new subset of shading-related problems for ray-tracing algorithms [66]. Most of these common problems are well understood and can be solved with well-established methods developed for polygonal modeling and rendering. Detailed discussions on advantages and disadvantages of polygonal representation of 3D objects may be found in any text on computer graphics, for example, in [25] and [71].

Problems of the second kind are specific for implicit surfaces only and are less explored.

First, the polygonal representation of implicit surfaces is not something that can be taken for granted — this is a large area of research. For a comprehensive review and classification of polygonization methods, see [15], [47] and [80]. In the most general case of implicit surfaces, when nothing is assumed about the modeling function $F(x, y, z)$, there are no guaranteed methods to solve the isosurface equation $F(x, y, z) = 0$ reliably, and, therefore, to polygonize the surface correctly. Since the polygonization algorithms effectively flatten the isosurface between the sample points, small features may be missed if they slip between the samples.

Second, the surface has to be repolygonized each time as the modeling implicit equation $F(x, y, z) = 0$ changes, for instance, when components of the composite object, modeled with an implicit equation (1.1), move with respect to each other. In animated metamorphoses, this routinely happens with every new frame. A dramatic example of how serious the problem of repolygonization may get, is described by Hart et. al. [37]. "The Hollywood film industry issued a challenge to computer graphics community, which was iconified by the character *La Magra* in the vampire movie *Blade*. La Magra was described as a tornadic monster of blood and was expected to consist of as many as tens of thousands of metaballs [9] " (reiterated from [37]). La Magra was also expected to swirl across the scene at very high rates, so that motion blur has to be used to make her movements look realistic. To render motion blur efficiently, a polygon-based representation was required. Thus, the real challenge was how to polygonize the surface of such complexity for every single frame. To avoid the brute-force repolygonization, the authors proposed to maintain the polygon mesh dynamically, basing on the method developed in [69]. The idea was to watch for critical changes in the topology of the mesh and make the necessary adjustments locally. Although the preliminary results were promising, a number of unsolved technical problems prevented the successful completion of the project. La Magra was removed from the script.

> *It seems apparent now, that the lack of efficient algorithms for visualizing implicit surfaces restrict their use in the computer graphics industry, especially, in the film production industry, where requirements on image resolution, modeling flexibility and complexity are very high.*

One of the core operations of visualizing implicit surfaces, with or without polygonizing, is solving the implicit equation $F(x, y, z) = 0$. Thus, it is highly desirable to have a method for solving $F(x, y, z) = 0$ rapidly, reliably and with a high degree of accuracy for a wide class of modeling implicit functions $F(x, y, z)$. If this task is achieved, implicit surfaces in general and convolution surfaces in particular could be rendered efficiently, ideally without intermediate polygonization step, i.e. directly from their model representation. The polygonization algorithms could also benefit from such method.

1.4. CONTRIBUTIONS

In this dissertation, we provide our solutions for all three sets of problems, described above.

In Chapter 2, a new mathematical formulation of convolution surfaces is developed. We present a new potential function $h(r)$ that yields closed-form solutions of the convolution integral (1.2) for a wide set of modeling primitives, including points, lines, curves, planes and polygons. These primitives form the basic

[8] 'Geometric aliasing' (also known as 'scaling problem') is an unwanted artifact in computer-generated images, which results in piecewise representation of curved surfaces by meshes of polygons. Geometric aliasing manifests itself during enlargement of an image, when straight edges of the polygons become obvious. This happens, for instance, when a camera zooms in on an object.

[9] 70,000 metaballs, as reported during the paper presentation at the 3rd International Workshop on Implicit Surfaces in Seattle, 1998.

building set of zero-, one- and two-dimensional skeletal elements for implicit modeling. Closed-form solutions eliminate the need for intermediate volumetric computation and storage, which were typical attributes for most implementations of the convolution surface models before [12, 13, 60]. Also, in this Chapter we provide a comparative analysis of alternative potential functions with respect to their compatibility with various modeling primitives and computational costs.

In Chapter 3, we explore the modeling capabilities of convolution surfaces built with the newly developed primitives. We introduce techniques that go beyond simple blends. Using implicit surfaces, we sculpt objects, manipulating their shape at all levels of detail, including fine textures. Methods for creating spikes, horns, wrinkles and other features that are common for many organic objects, are discussed.

Chapter 4 is devoted entirely to rendering. We present a ray-tracing algorithm, specifically designed for rendering convolution surfaces. As of this writing, this algorithm outperforms all other ray-tracing algorithms in its class. The algorithm is based upon a piecewise interpolation scheme, that allows the surface to be approximated as closely as required. Within this approximation, the algorithm is guaranteed to locate the surfaces to a machine-size floating-point precision. The algorithm does not employ numerical root-finding methods, therefore, it does not require iterations and multiple evaluations of the modeling functions. This permits the rapid and accurate rendering of scenes with very high complexity, and with constant computational cost with respect to the silhouette edges. Although the algorithm is implemented as part of a ray-tracing program, its core interpolating and surface-locating modules may be readily used for polygon-generating techniques. However, the high speed of our algorithm allows direct rendering with comfortable interactive rates.

Each chapter of the thesis is preceded with its own short introduction that supplies the reader with the context of the problems being addressed and provides a review of related work. Also, each chapter is concluded with a brief summary of the results presented and a discussion of their possible application and further improvements. In principle, all parts of this thesis may be read separately. For better readability, we allowed some repetitions of the key notions and definitions.

"In 10 years, all rendering will be volume rendering", predicted Jim Kajiya at SIGGRAPH'91. Ergo, all modeling has to become implicit modeling!

While this statement may be a slight exaggeration, we strongly believe that the time for implicit modeling methods to take their place in the Computer Graphics arsenal is long overdue.

We believe that material presented in this dissertation, reveals the true potential of convolution surfaces, a superset of 'classic implicits'. Consistent mathematical formulation, a wide application base and the efficient rendering algorithm make convolution surfaces more practical and powerful than ever.