Shape Design Using Convolution Surfaces

Andrei Sherstyuk

School of Computer Science and Software Engineering
Monash University, Victoria 3168, Australia
E-mail: ash@csse.monash.edu.au

Abstract

This paper presents a complete design environment that explores the modeling capabilities of convolution surfaces and puts them into practice. The design system includes a number of implicit primitives and modeling techniques, that allow objects to be created by sculpturing their surfaces at all levels of detail. All techniques are illustrated using models of marine life forms and other objects of organic origin. Implementation and efficiency issues are discussed.

1 Introduction

Convolution surfaces were introduced by Bloomenthal and Shoemake [8] as a generalization of the 'blobby models' developed for computer graphics in [4], [19], [28]. Being a superset of these models, convolution surfaces inherit their valuable properties, such as an ability to form smooth blends and free-form shapes. At the same time, convolution surfaces demonstrate much greater modeling flexibility, allowing a designer to create objects using skeletal elements of various shapes and sizes. This advantage was crisply stated in the original paper on convolution surfaces:

> Convolution surfaces incorporate the smooth blending power and easy manipulability of potential surfaces while expanding the skeletons from points to lines, polygons, planar curves and regions, and in principle, any geometric primitive.
> *Bloomenthal and Shoemake* [8].

Such qualities make convolution surfaces particularly attractive for the design of articulated objects, especially those of organic origin. The modeling practices developed over nearly two decades for 'classic implicits' [4, 19, 28] cannot, however, be carried over to convolution surfaces 'as is', without doing an injustice to the convolution surface model. As a more versatile and more powerful tool, convolution needs a better 'manual of operation' to reveal its potential fully.

To date, practical examples of modeling with implicit surfaces in general, and convolution surfaces in particular, mainly exercise their blending abilities. Consequently, the majority of objects designed with implicit surfaces demonstrate various branching structures, such as trees [5, 14], hands [6], paws [22] and chromosomes [9]; or exhibit certain softness or fluidity, either in animation or as perceived from static images. Examples are: human lips [12] and faces [17], molecular shapes [4], boiling liquids [27] and rubbery-looking objects [26]. Few attempts have been made to employ implicit surfaces for modeling rough objects. One of them is presented in [15].

Starting from our previous work [16, 23], we show that the modeling capabilities of convolution surfaces extend beyond traditional blends. We introduce a set of primitives and a number of techniques that allow us to sculpture objects, manipulating their shape at all levels of detail, including fine textures. We demonstrate that convolution surfaces can be successfully used to represent not only soft and pliable substances, but also objects that are hard and fragile. For that reason, most examples show marine life forms – there are plenty of such creatures of all types.

We also outline a system architecture for interactive design using convolution surfaces. The system allows objects to be modified at interactive rates, which helps the design to converge quickly to the desired shape.

2 Definitions

The following are the basic concepts and equations used in modeling with implicit surfaces.

2.1 Implicit surfaces

An *implicit surface S* is defined as an isosurface at level T in some scalar field $f(\mathbf{p})$:

$$S = \{\mathbf{p} \in R^3 \mid f(\mathbf{p}) - T = 0\} \qquad (1)$$

Figure 1 gives an example of a simple implicit surface.

2.2 Convolution surfaces

A *convolution surface* is the implicit surface based on a function, $f(\mathbf{p})$, obtained via the convolution of a kernel

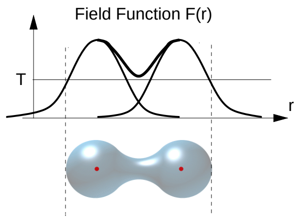


Figure 1. An implicit surface generated by two point sources.

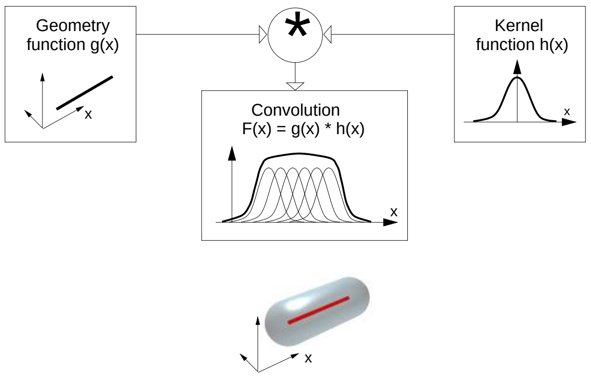


Figure 2. Components of a convolution surface: geometry function $g(\mathbf{p})$ (top left), kernel function $h(\mathbf{p})$ (top right), convolution field function $f(\mathbf{p})$ (center), convolution surface (bottom).

function h with a geometry function g:

$$f(\mathbf{p}) = g(\mathbf{p}) \star h(\mathbf{p}) = \int_{R^3} g(\mathbf{r}) h(\mathbf{p} - \mathbf{r})\, d\mathbf{r} \qquad (2)$$

The geometry function $g(\mathbf{p})$ defines the shape of an object and its position in 3D space. The kernel function $h(\mathbf{p})$ defines the distribution of some potential that is produced by each point on the object. Convolved together, these two functions produce a tri-variate scalar function $f(\mathbf{p})$ that defines the convolution surface. Figure 2 gives an example of convolution with a Gaussian-like kernel. Here the geometry function $g(\mathbf{p})$ describes a line segment aligned along the x-axis.

2.3 Implicit primitives

In the metaballs model of implicit surfaces [19], the point potential sources were introduced as "meta-primitives defined by their distribution function, that together form a meta-surface". With respect to convolution surfaces, we say that the distribution function (2) defines an *implicit primitive* with geometry g.

Since most modeling primitives form closed compact sets (points, line segments, triangles, etc), the integration (2) over 3D space may be conveniently replaced with integration over the volume **V** of the modeling primitive:

$$f(\mathbf{p}) = \int_{\mathbf{V}} h(\mathbf{p} - \mathbf{r})\, d\mathbf{r} \qquad (3)$$

2.4 Skeletons

A *skeleton* is a collection of geometric primitives that outline the inner structure of an object being modeled. With respect to the convolution surface model, a skeleton is a sum (union) of geometry functions $g = \sum_{i=1}^{N} g_i$ and $f = h \star \sum_{i=1}^{N} g_i$ which also equals $\sum_{i=1}^{N} h \star g_i$ because convolution is a linear operator. Visually, such a skeleton is represented by a union of corresponding primitives. Convolved with a kernel function, the skeleton yields a field function $f(\mathbf{p})$ and a convolution surface S.

The concept of a skeletal design with convolution surfaces was introduced to computer graphics by Bloomenthal and Shoemake in [8]. They observed that the additive property of convolution allows us to build complex skeletons out of simple primitives and, most importantly, allows us to evaluate them individually. That makes convolution surfaces computationally practical.

Bloomenthal [6] applied convolution surfaces to generate smooth shapes, resembling various organic objects. In what follows, we will use implicit primitives obtained via convolution technique, to produce a wide variety of surfaces, including rough, prickly and wrinkled surfaces.

3 The design system

In this section we present our system for modeling with convolution surfaces.

3.1 New modeling primitives as skeletal elements

In principle, any geometric primitive can be used as a skeletal element for the convolution surface model by means of the generic integral (2). In practice, the choice of such primitives is often limited by technical difficulties in evaluating the convolution integral.

As most implementations of the convolution surface model demonstrate [8, 6, 22], such computations require point-sampling of the field in the model space and storage of intermediate results for polygonization and rendering. As for techniques that employ point-sampling, special care must be taken to ensure that small features of the object being modeled are not missed. This task may be especially difficult for models that contain primitives of widely varying characteristic sizes.

A closed-form solution for the convolution integral (2) provides such a care-free modeling environment. Based

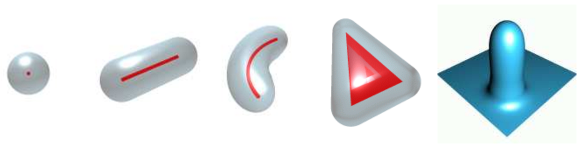


Figure 3. A set of implicit primitives: point, line segment, arc, triangle and plane.

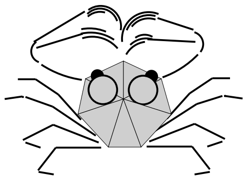


Figure 4. The skeleton of this coral crab is made of 7 triangles, 24 arcs and 24 segments.

on our prior results [16], we suggest the following implicit primitives as skeletal elements for convolution surfaces: points, line segments, arcs, triangles, planes. All these modeling primitives are presented as closed-form functions, i.e. they are built up as compositions of elementary functions that return the field strength generated by the primitive at an arbitrary point, calculated to a machine precision. This makes it possible to visualize convolution surfaces using direct rendering algorithms, such as ray-tracing. Thus, neither preprocessing nor intermediate storage is required to render the surface. This is particularly convenient for interactive design. Figure 3 shows the implicit primitives, rendered as semi-transparent surfaces with the underlying 'bare' geometric primitives inside. The actual field functions are presented in [16].

Although most of these primitives have been used for implicit modeling before (points in [4, 17, 19, 28], line segments in [2, 5, 22] and triangles in [6, 22]), the closed form formulation of their field functions given in [16] is still unexplored. These functions constitute the core of our design system.

With a multitude of modeling primitives available, the concept of a skeleton becomes especially life-like, because now a designer may think of skeletons as if they consist of solid 'bones', each of unique shape and size: point, line segments, curves, triangular pieces. In addition, arcs may be combined into circles and spirals; triangles – into polygons; line segments – into polylines, etc. This allows a designer to work with the skeleton, creating and using those elements that fit best for each particular part of the object. An example of such a multi-primitive skeleton is shown in Figure 4 which depicts a model of a crab. Note that even this simple sketch gives a good idea of what the resulting shape of the crab is going to look like. The completed model of the crab will appear later in section 5.

3.2 Local properties of implicit primitives

As defined by equation (2), a convolution surface is the product of a geometry function g and a kernel function h. A geometry function $g = \sum_{i=1}^{N} g_i$ forms a skeleton of N pieces, which provides a global description of the object: its general layout, location and the basic shapes of its parts. To complete the description of the object, the local properties of each element g_i of the skeleton must be specified.

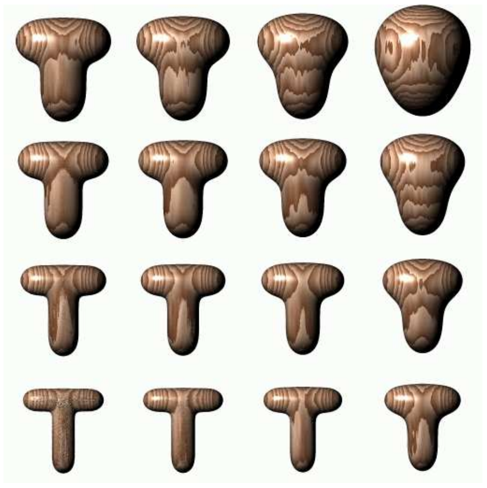


Figure 5. A T-shaped convolution surface, formed by two line segments. Increasing blobbiness (left to right) and radius in isolation (from bottom to top) allows to achieve a variety of forms.

Such properties may be conveniently described in terms of the radius in isolation R and blobbiness B, as was suggested by Blinn [4]. A *radius in isolation* is a characteristic distance between a geometric primitive and the implicit surface that it generates. For non-point primitives, the radius may vary along the surface. A *blobbiness* parameter associated with the skeleton and passed to a kernel h controls blending of the object's parts. Very blobby objects, when brought together, tend to form sphere-like shapes with very little or no distinguishable detail preserved. Skeletal elements with low blobbiness retain their original shape. Figure 5 shows a series of convolution surfaces created with various values of blobbiness and radius in isolation. Blinn used these two parameters to define the appearance of his point-based modeling primitives. We extend them to all the modeling primitives pictured in Figure 3.

The skeleton, radius in isolation and blobbiness of all skeletal elements completely define an implicit surface. The threshold T, used in the implicit surfaces equation (1), may be conveniently set to some canonical value, as described in [4].

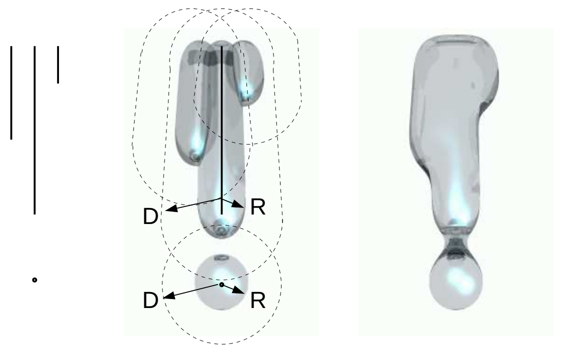


Figure 6. Elements on an implicit icicle, from left to right: skeleton made of ‘bare’ geometric primitives (three segments and one point); their implicit surfaces, rendered as stand-alone objects; the final convolution surface. In the middle image, blending between implicit surfaces is disabled to show the radius in isolation R and depth of influence D better.

As an alternative to radius in isolation, an implicit primitive may be characterized by its *region of influence*. A region of influence is the region in 3D space, where the field function f of the primitive has non-zero values.

An important property of regions of influence is that implicit surfaces are always located inside the boundaries of these regions. Regardless of the values of blobbiness and radius in isolation, the convolution surface of the icicle in Figure 6 (middle and right), always covers its ‘bare’ skeleton (solid lines) and is always covered by the boundaries of the regions of influence of its components (dotted lines). This property is very valuable for estimating the extents of a convolution surfaces.

Thus, the depth of the region of influence (or simply depth of influence) may serve as a modeling parameter, which is an alternative to radius in isolation. Figure 6 demonstrates the basic elements and parameters of modeling with implicit surfaces.

3.3 Materials and elements

Having formulated the properties of skeletal elements (which are radius in isolation R and blobbiness B), we must provide a way to associate these parameters with the skeletal elements. This may be done by

- sharing them among a large number of elements;
- assigning them to each skeletal element individually.

Practice shows that both ways have their uses, so we incorporated them both into our modeling system.

3.3.1 Materials

The first approach is more convenient for modeling objects that have large areas with the same properties, because it allows us to employ the notion of a material. Normally, a description of a material includes conventional photometric characteristics, dependent on a lighting model: diffuse color, specular reflectivity, transparency, etc. Such descriptions may easily be enhanced by adding values of blobbiness and radius in isolation, which resemble softness and thickness of the material, respectively. This fits well with the idea of skeletal design: now each ‘bone’ in the skeleton is covered with some ‘soft tissue’, which is shared between skeletal elements. As an example, consider the material of an implicit icicle in Figure 6:

```
material Ice {
    body          SummerSky,
    ambient       1%,
    diffuse       1%,
    specular      SummerSky, shine 30,
    transparent   40%, index 1.33,
    reflective    50%,
    radius 0.25,  blobbiness -0.30
}
```

In this example, material `Ice` is defined as an almost colorless, reflective and transparent substance. Values of `radius` and `blobbiness` dictate the geometric aspects of the convolution surface, that is based upon the skeleton, defined as

```
object ICICLE
    line Ice, <0 0 0>, <0 0 4.5>
    line Ice, <0 0.5 3.0>, <0 0.5 4.5>
    line Ice, <0 -0.5 1.5>, <0 -0.5 4.5>
    dot  Ice, <0 0 -0.85>
close
```

The material-based description of convolution surfaces has two advantages.

First, for design purposes, it is more convenient to think of skeletons and ‘soft tissues’ as separate entities that can be modified independently of each other. Keeping the material descriptions separate from the skeleton often allows materials of the objects to be changed interactively. This helps many surface parameters, both photometric and geometric, to be adjusted without reloading the model into memory for rendering.

Secondly, the local properties of the skeleton are not duplicated for each skeletal element, but are shared via a description of a material. (Note the simplicity of the syntax for the `dot` and `line` modeling primitives in the example above.) This arrangement is memory-friendly, which may be important when the number of skeletal elements is large.

3.3.2 Individual elements

The individual assignment of parameters R and B is needed when the model requires each primitive to be strictly different from its neighbors. Such models often come from procedural methods, especially in the simulation of growth [13, 20]. For example, consider a sequence of spheres, pictured in Figure 7, that is characteristic for solid-based models of seashells [20]. In the domain of implicit modeling, each sphere depicts a region of influence of a point primitive, so the whole object must be described as

```
object SHELL
    sphere Plaster, r1, x1 y1 z1
    sphere Plaster, r2, x2 y2 z2
    ...
close
```

Here $r_1, r_2, ..., r_i$ denote the depth of influence of each point, triples (x, y, z) give the locations of their centers. The value of the blobbiness is stored in the specifications for the material `Plaster`, defined elsewhere, as was done for the material `Ice`. All seashell models that will be discussed below are modeled by setting individual depths of influence for each element.

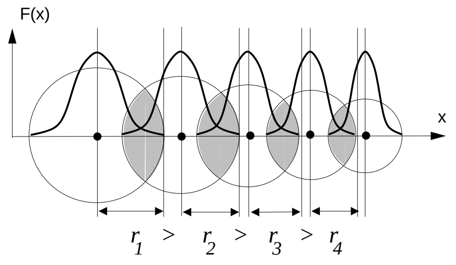


Figure 7. Strictly decreasing regions of influence of these implicit point primitives call for individual assignment of their values.

3.4 Profiling as a means for achieving variety

Figure 3 shows the basic primitives in their canonical, undeformed shapes, which may not be the best for a particular modeling task. Rather, these shapes provide a platform for variations and experiments. There are infinitely many ways to modify the field function of an implicit primitive.

As shown in Figure 8, an undeformed line segment (A) easily assumes various shapes (B, C, D), following the *profiling functions*. Profiling functions scale the values of the field, causing changes in the resulting implicit surface. Since line segments are axially symmetric, each profiling function, pictured in Figure 8, defines a surface of revolution around the axis.

To continue a parallel with skeletons, profiling functions provide a mechanism for simulating changes in thickness and/or softness of the tissues along the bones, e.g., line segments. Profiling functions are used with nearly all examples of modeling with convolution surfaces presented in this paper. At present our modeling system includes over a dozen predefined profiling functions similar to those shown in Figure 8.

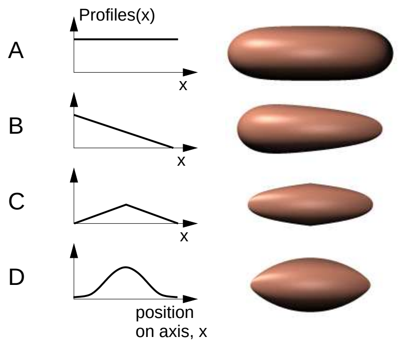


Figure 8. Line segments with modified field functions: constant (A), linear (B), hat (C), $sin^2(x\pi)$ (D).

3.5 Offset surfaces as a visual aid and the data structures

As Figure 3 shows, the unblended implicit primitives based upon points, line segments, arcs, triangles and planes can be sufficiently approximated by an offset surface [8, 7] based upon the same primitives. These are, respectively: spheres, cylinders, arc tubes, prisms and infinite slabs. The extrusion distance may be set to radius in isolation R or depth of influence D (see Figure 6, middle), according to the current modeling task.

Such similarity suggests a convenient modeling strategy: first an object is approximated by an offset surface and then refined as a convolution surface. The important feature of this approach is that most of the hard modeling work related to building the skeleton may be done with easy-to-render offset surfaces.

To support this correspondence, we use the same syntax and internal data structures both for implicit primitives and their offset counterparts. For instance, the command line

```
sphere Plaster, r0, 0 0 0
```

creates an object which may appear as an ordinary sphere, with radius `r0`, made of `Plaster` and positioned at the origin, but this object may also appear as an implicit point with a radius in isolation `r0`. As described above, its blobbiness is stored in the description of the material `Plaster`.

Cylinders, arcs, triangles and planes also have reusable syntax and data structures. When the set of primitives is loaded into memory it is ready for rendering in either mode: as an offset surface or as a convolution surface. In the former case, the implicit nature of all the modeling primitives is ignored and they are rendered as a union of offset surfaces. In the latter case, the primitives are treated as functions $f_i(\mathbf{p})$ that form the composite field of a convolution surface. To switch between the two rendering modes, a simple on/off flag is used; the model does not have to be reloaded into memory.

Such duality of data structures has proven itself very useful. Computationally inexpensive offset drafts allow fine tuning of models at interactive rates. Since ray-tracing is used as a rendering method, all camera settings, lights and photometric properties of the objects are also adjusted in the draft mode.

3.6 Variables

For better interactivity in our design environment the following data types are allowed to be defined as variables: scalars, vectors, colors, textures, materials. Variables may be created and modified 'on the fly' during a design session. They may also be assigned to each other, observing type conversion. With the use of variables it is possible to modify practically all elements of the model while it is still in computer memory. These elements include: positions and shapes of all skeletal elements, blobbiness and thickness, all photometric characteristics. In addition, variables may be declared as auto-incremental and/or auto-multiplicable, which helps specifying motion of objects in animations.

4 The main modeling loop

4.1 Datasets

In the previous section, we mentioned elements of what we call a *dataset*, which is a complete collection of parameters that defines a particular geometric model. Every dataset contains descriptions of all materials and skeletal elements that together form an offset or convolution surface. In addition, a dataset describes all modifications of the local properties of skeletal elements that are needed in order to add a desired shape to the final surface.

With respect to design and rendering, a dataset may be regarded as an output of a design process and an input for a rendering program.

4.2 The main modeling strategy

We use a method of progressive refinement of the dataset as the modeling strategy. An initial dataset is acquired first using some foreign geometric model, a special-purpose program or a user's inspiration with a paper-and-pencil method. Then the dataset undergoes a series of iterations, during which a designer modifies existing elements of the dataset, adds new ones and deletes those that fail to fit the model.

The global dataflow chart of a typical design session is depicted in Figure 9. For this particular example, the skeletal model was produced by a tree-growing program, based upon Eric Haines' implementation [13] of Aono and Kunii's tree-generation method [1]. By applying various values of thickness to the skeletal elements, several drafts have been made. These are pictured as a stack of images on the left-hand side of Figure 9. Since offset surfaces, used for

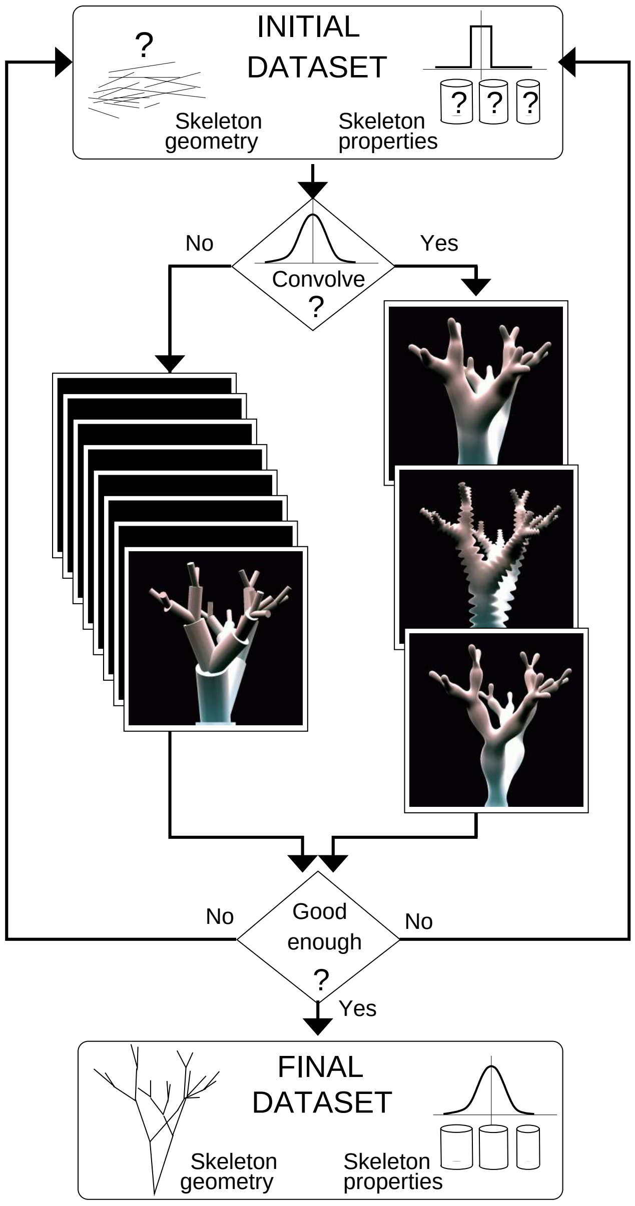


Figure 9. The main modeling loop.

the drafts, are easy to render, the process can easily be reiterated as many times as needed to find the esthetically desired values for thickness of all branches.

At this point the designer may switch to using convolution surfaces and try to enhance the resulting shape by applying different profiling functions. Several images, depicting such iterations, are shown on the right-hand side of Figure 9. After the third attempt, the desired shape was obtained – the dataset is now complete and the process terminates.

Note that this modeling strategy is a winning strategy by construction: the process terminates when the designer is satisfied with the result. Visual control over the quality of the current state of the model guarantees that the design process moves in the right direction. Backup copies of the dataset help to fix irreversible changes.

5 Practical examples of implicit design

In this section, a number of practical modeling examples are discussed. They are grouped by methods that have been used for their design. With respect to the main modeling loop, pictured in Figure 9, these methods are roughly divided into those that deal with the skeletons (left-hand side of the chart) and methods that refine the local properties of the convolved surfaces (right-hand side).

5.1 Global skeleton manipulations

In principle, the modeling system allows us to build a new skeleton from scratch interactively. However, it is more convenient to prepare an initial skeleton using one of the methods listed below.

5.1.1 Procedural methods

Two skeletons have been produced using purely procedural methods: the flat spiral shell in Figure 13 and the seaweed in Figure 9. Both are generated using modified utilities `shell` and `tree`, respectively, from the Standard Procedural Database of Eric Haines [13]. Their skeletons are fairly simple.

5.1.2 Hand-crafting

The skeleton of a seahorse in Figure 17 is 100% handcrafted. The initial paper-and-pencil drawing was created first. Then it was duplicated with the *xfig* drawing tool (Figure 17, left), which produced the coordinates of the skeletal elements, in this case, arcs. In most cases, the joints between the arcs in the line drawing are C^1 and C^2 continuous; the cracks in the offset surface (Figure 17, middle) are introduced intentionally to show the skeletal elements better. The final convolved surface (Figure 17, right) seals these cracks. Note, that the thickness of each skeletal element is set individually. The central arc in the body of the seahorse is made especially thick. The resulting inflation fills the empty space inside the body and, perhaps, indicates the presence of internal organs.

5.1.3 Mixed methods

Procedurally generated models may be significantly improved by manual editing. For instance, the oval seashell, pictured in Figure 14, is derived from the previous flat sphere-based model (Figure 13) by the simple addition of one point which was then connected with all points in the initial dataset. Thus, all spheres become cylinders with a common base. By moving this common base, the shape of the new shell may be controlled easily. This is similar to rubber-band techniques used with a wire-frame representation of a solid object. Using a vector variable for coordinates of the common base point makes such manipulations especially convenient.

The skeleton of the next shell was also created procedurally and then updated manually. This time two skeletons were created as described above. Then they were superimposed to produce a combined skeleton for the spiked shell, shown in Figure 15.

While manual intervention often improves procedural skeletons, the opposite is also true. Hand-crafted skeletons can be made much more interesting and visually appealing with the help of special purpose program generators. Examples of such skeletons are: a spindle cowrie shell (Figure 16) and coral crab (Figure 18). Both skeletons were hand-copied from the actual photographs [18], using the *xfig* drawing tool. Then, small details were added procedurally.

Usually, skeletons built using mixed techniques yield the most interesting shapes.

5.1.4 Imported models

One fully imported polygonal model of Nefertiti is pictured in Figure 11. It has been used as a polygonal skeleton for the convolution surfaces without any modification done to its initial mesh of triangles.

5.2 Local modeling techniques

After the global skeleton is complete a designer may switch to more subtle modeling techniques and refine the model to their taste. With respect to the modeling loop (Figure 9), these operations happen on the right-hand side of the loop, because most techniques that are described below apply to convolved versions of the object.

5.2.1 Shaping techniques

Shaping is a straight-forward application of profiling functions, that creates local variations of radius in isolation of modeling primitives (or the thickness of the material). Examples of shaping are presented in Figure 9, where a normal tree is turned into seaweed by applying a linear shaping function to its branches. Less obvious examples of shaping may be found in the model of a coral crab (Figure 18). The end segments of the crab's legs are also linearly shaped to make them look sharper.

5.2.2 Carving techniques

Carving is a combination of using depth of influence as a modeling parameter with some profiling functions. The depth of influence around modeling primitives defines a volume of 'matter' to carve from; the profiling function defines the shape of the 'cutting tool'.

Technically, profiling functions operate on each modeling primitive individually, as shown in Figure 8. When the modeling primitives interpenetrate, however, which is true for the spiral seashells (Figures 13, 14, 15), the result of carving is as if the cutting tool had been revolved in the model space following the spiral curve of the shell. In this way, profiling functions are similar to the *generating curves* used for modeling seashells in [10]. The carving technique was used with all the models of spiral shells pictured in Figures 13, 14 and 15.

6 Volumetric detail

Defined as isosurfaces in a scalar field, convolution surfaces are sensitive to variations of that field. Large-scale variations cause global changes in the appearance of the object. Small-scale variations modify the geometry of the surface locally, adding more detail to the surface. Such detail enhances the appearance of the object and produces various texturing effects. Since these details are based upon implicit functions, which are volumetric by their nature, we call this *volumetric detail*.

Most models presented in this paper contain volumetric details. These may be grouped in three major ways: structural, functional and procedural.

6.1 Structural detail

This method involves adding more modeling primitives, which may be done manually or by using some auxiliary utilities. Although it seems like a brute-force solution, structural detail can be very effective in modeling various irregular and rough-looking surfaces, such as the body, claws and pincer grip of the coral crab shown in Figure 18. Similarly, the shell of a spindle cowrie (Figure 16) is enhanced by cylindrical spikes. Other examples of using spikes with convolution surfaces are presented in [6, 7].

One convenient feature of structural details is that, unlike other methods of adding detail, most iterations on the dataset may be done with the offset representation of the

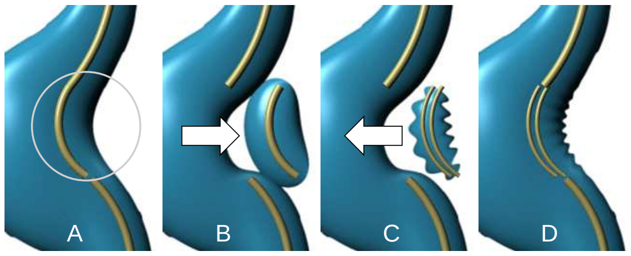


Figure 10. How to make volumetric wrinkles: (A) The place for wrinkles is chosen. (B) The closest arc is pulled out. (C) The arc is split into halves and one half is modulated by a sine wave. Together, they form a wrinkled volumetric implant, ready for re-insertion. (D) The wrinkled implant is put back in place.

surfaces. This corresponds to the left-hand side of the main modeling loop (Figure 9), where a designer may allow a large number of takes.

6.2 Functional detail

Functional detail is produced by applying profiling functions to small areas of surfaces. As an example consider wrinkles on the back and the tail of a seahorse (Figure 17). Figure 10 illustrates how the wrinkles were implanted in the originally smooth convolution surface. This figure is an exact close-up of Figure 17 with more anatomical parts shown (only participating skeletal elements and close neighbors are displayed).

Note that the field function of the wrinkled volumetric implant, as depicted in Figure 10, could have been represented by a single arc, modified by an elevated and scaled sine wave $a_1(1 + a_2\sin(x))$, instead of two, as shown above. In this case, it is a matter of choice whether to add another primitive into the dataset, or define one more profiling function in the modeling system. A similar use of high-frequency functions for adding volumetric details is reported in [3] and in [7].

6.3 Procedural detail

Perhaps, the most unusual examples of volumetric detail are given in Figure 11. These images present the results of experiments with triangular meshes used as skeletons for convolution surfaces.

Triangular meshes are normally employed to approximate surfaces, as shown in Figure 11, left. Convolving triangles with a potential kernel function introduces remarkable changes into the appearance of the object being modeled, as illustrated in Figure 11, middle. However, simple convolution makes the skin of Nefertiti look swollen.

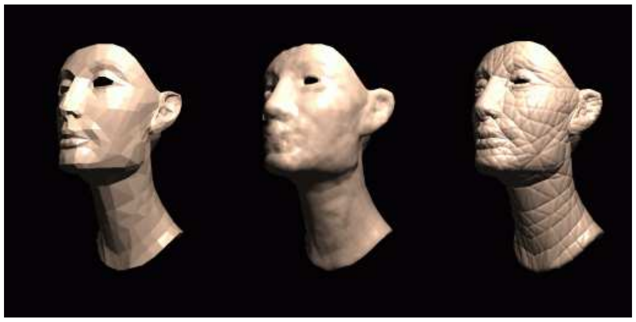


Figure 11. Nefertiti, before (left) and after (middle, right) convolution. Restricted blending near the edges produces the wrinkles (right). Total 1,242 implicit triangles.

All fine features are missing because high frequencies have been removed by a low-pass convolution filter.

To reduce the swelling effect, a restricted mechanism is used which artificially reduces the amount of field generated by the triangular primitives near the edges. Thus, the surface exhibits hills over the central parts of each triangle and valleys along their edges. The maximal difference in elevation of these hills and valleys depends on the thickness of the material of the face (i.e., skin) and the value of the restriction factor. The former tends to elevate the convolution surface above the 'bare' polygonal skeleton, especially in the central areas of each polygon. The latter pulls the surface closer to edges of triangles, producing wrinkles along the edges. The roughness of the resulting implicit surface may be effectively controlled by varying both parameters: thickness and restriction scaling factor.

This mimics the effects of aging. The wrinkled and dried-up skin makes the face of Nefertiti arguably more interesting and definitely more realistic — the prototype of this model is 5,500 years old.

The restriction mechanism may be combined with an additional UV-mapping to produce even more wrinkles inside each triangle. Alternatively, procedural or noisy functions may be used to modify the values of the field functions. In all cases the visible complexity of the resulting convolution surface will be increased without increasing the number of elements in the skeleton.

Note that the types of volumetric details presented above are not shading tricks, but true geometric details. They participate in a full range of graphics manipulations, such as visibility calculations, shadows, transformations, etc. By modifying the local geometry of the surface, volumetric detail creates additional features that may be very informative. For instance, spikes and horns imply firm and rigid surfaces; wrinkles often suggest that the object is flexible, allows deformations, and is currently deformed from its usual state; wrinkled skin suggests old age. These details enhance the visual realism of the model and have a strong potential for complex modeling with implicit surfaces.

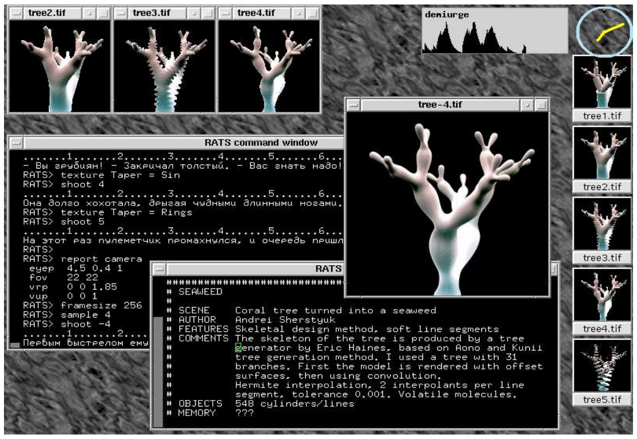
Model	Skeleton	Techniques
Shell I	Procedural	Carving
Shell II	Procedural, manual	Carving, shaping
Shell III	Procedural, manual	Carving, shaping
Seaweed	Procedural	Shaping
Seahorse	Manual	Functional wrinkles
Cowrie	Manual, procedural	Structural detail
Crab	Manual, procedural	Structural detail, shaping
Nefertiti	Imported	Procedural wrinkles

Table 1. Summary of modeling techniques.

7 Implementation details and timing results

All the models discussed above were designed and ray-traced using the in-house modeling/rendering system *RATS Version 7.31* running under Linux OS on a 90 MHz Pentium processor. More about *RATS* may be found in [21] and [24]. The algorithm for ray-tracing convolution surfaces, used in the *RATS* system, is described in [23]. Offset surfaces are also ray-traced; the relevant algorithms may be found in [11] and [25]. Table 2 gives details about each dataset and the rendering times. Image resolution: 512 x 512. Anti-aliasing method: shooting at most 16 primary rays per pixel. Of course, during interactive design sessions, much smaller images were used, typically 128 x 128. Shooting 1 ray per pixel allowed us to render all offset surfaces under 10 seconds per iteration.

These timing data give a somewhat distorted picture of the actual situation. For instance, some offset surfaces contain a large number of edges that had to be anti-aliased by



```
.......1.......2.......3.......4.......5.......6...
- Вы грубиян! - Закричал толстый. - Вас знать надо!
RATS> texture Taper = Sin
RATS> shoot 4
.......1.......2.......3.......4.......5.......6...
Она долго хохотала, дрыгая чудными длинными ногами.
RATS> texture Taper = Rings
RATS> shoot 5
.......1.......2.......3.......4.......5.......6...
На этот раз пулеметчик промахнулся, и очередь пришл
RATS>
RATS> report camera
 eyep  4.5 0.4 1
 fov   22 22
 vrp   0 0 1.85
 vup   0 0 1
RATS> framesize 256
RATS> sample 4
RATS> shoot -4
.......1.......2....
Первым выстрелом ему
```

```
################################
# SEAWEED
#
# SCENE    Coral tree turned into a seaweed
# AUTHOR   Andrei Sherstyuk
# FEATURES Skeletal design method, soft line segments
# COMMENTS The skeleton of the tree is produced by a tree
#          generator by Eric Haines, based on Aono and Kunii
#          tree generation method. I used a tree with 31
#          branches. First the model is rendered with offset
#          surfaces, then using convolution.
#          Hermite interpolation, 2 interpolants per line
#          segment, tolerance 0.001. Volatile molecules.
# OBJECTS  548 cylinders/lines
# MEMORY   ???
```

Figure 12. A screen shot of a design session: input window (left), dataset file (bottom), output images (top) and thumbnail icons (right).

Model	**Contains**		t_1	t_2	t_2/t_1
Shell I	361	points	2:52	22:46	7.81
Shell II	42	lines	20:45	32:51	1.58
Shell III	70	lines	12:54	41:16	3.20
Seaweed	31	lines	3:34	4:46	1.34
Seahorse	43	arcs	5:25	11:52	2.19
Cowrie	103	total	7:34	24:40	3.26
Crab	641	total	14:03	46:13	3.29

Table 2. Rendering time (min:sec) for offset surfaces t_1 and convolution surfaces t_2.

shooting more rays. Convolved shapes tend to be smoother, thus fewer pixels had to be supersampled. This explains an unusually low time increase of 1.58 for rendering the second seashell as a convolution surface: it has a very sharp offset-surface counterpart.

The seahorse dataset presents another special case of low time ratio. Since arc tubes, used as offset surfaces, are already computationally expensive (they require solving quartic polynomials for each intersection test), switching to convolution surfaces does not introduce dramatic changes in rendering time.

The fastest implicit primitive, according to Table 2, is a line segment (see the seaweed model in Figure 9). It required only 34% more rendering time than its explicit counterpart. The average time factor penalty for using convolution surfaces compared to offset surfaces is about 3.23. The crab model, which contains all types of new implicit primitives, shows a similar slow-down of 3.29.

8 Conclusion

A number of implicit primitives and a number of modeling techniques for convolution surfaces have been presented. Points, lines, arcs and triangles, when used as elements of convolution surfaces, provide a rich palette of unusual variations of this modeling concept. Methods of adding details to convolution surfaces are discussed, which enhance the visual realism of the surfaces and expand the application base of implicit modeling.

9 Acknowledgments

Many thanks to Peter Tischer and Jon McCormack for their advice and discussions and to John Crossley for his patience in critical readings and re-readings of this material. Very special thanks to Jules Bloomenthal for his personal support and numerous suggestions which helped to focus the paper.

References

[1] M. Aono and T. L. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, 4(5):10–29, 32–34, May 1984.

[2] T. Beier. Practical uses for implicit surfaces in animation. In *Modeling and Animating with Implicit Surfaces*, pages 20.1–20.11, 1990. SIGGRAPH Course Notes 23.

[3] H. B. Bidasaria. Defining and rendering of textured objects through the use of exponential functions. *Graphical Models and Image Processing*, 54(2):97–102, Mar. 1992.

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

[5] J. Bloomenthal. Modeling the mighty maple. *Computer Graphics*, 19(3):305–311, July 1985.

[6] J. Bloomenthal. *Skeletal Design of Natural Forms*. Doctoral dissertation, University of Calgary, Deptartment of Computer Science, 1995.

[7] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann Inc, 1997.

[8] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991.

[9] E. Ferley, M.-P. Cani-Gascuel, and D. Attali. Skeletal reconstruction of branching shapes. *Computer Graphics Forum*, 16(5):283–293, Dec. 1997.

[10] D. R. Fowler, H. Meinhardt, and P. Prusinkiewicz. Modeling seashells. *Computer Graphics*, 26(2):379–387, July 1992.

[11] A. S. Glassner, editor. *An Introduction to Ray Tracing*. Academic Press, 1989.

[12] T. Guirard-Marigny, N. Tsingos, A. Adjoudani, C. Benoit, and M.-P. Gascuel. 3d models of the lips for realistic speech animation. In *Computer Animation'96*, pages 80–89, 1996.

[13] E. A. Haines. A proposal for standard graphics environments. *IEEE Computer Graphics and Applications*, 7(11):3–5, Nov. 1987. ftp.princeton.edu:/pub/Graphics.

[14] J. Hart and B. Baker. Implicit modeling of tree surfaces. In *Implicit Surfaces '96*, pages 143–152, 1996.

[15] J. C. Hart. Implicit formulations of rough surfaces. *Computer Graphics Forum*, 16(2), June 1997.

[16] J. McCormack and A. Sherstyuk. Creating and rendering convolution surfaces. *Computer Graphics Forum*, 1998, to appear.

[17] S. Muraki. Volumetric shape description of range data using "blobby model". *Computer Graphics*, 25(4):227–235, July 1991.

[18] F. Nichols and M. Stachels, editors. *Marine life of the Indo-Pacific region*. Periplus Editions, Singapore, 1996.

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

[20] C. A. Pickover. A short recipe for seashell synthesis. *IEEE Computer Graphics and Applications*, 9(6):8–11, Nov. 1989.

[21] *RATS*. http://www.csse.monash.edu.au/˜ash/rats/.

[22] G. Sealy and G. Wyvill. Smoothing of three dimensional models by convolution. In *Computer Graphics International '96*, pages 184–190, June 1996.

[23] A. Sherstyuk. Fast ray tracing of implicit surfaces. In *Implicit Surfaces '98*, pages 145–153, June 1998.

[24] A. Sherstyuk. *Convolution Surfaces in Computer Graphics*. Doctoral dissertation, Monash University, School of CSSE, 1998 (pending).

[25] A. Watt and M. Watt. *Advanced animation and rendering techniques - Theory and practice*. Addison-Wesley, New York, 1992.

[26] B. Wyvill. The great train rubbery. *SIGGRAPH Electronic Theater and Video Review*, 26, 1988.

[27] G. Wyvill, C. McPheeters, and B. Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, 1986.

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

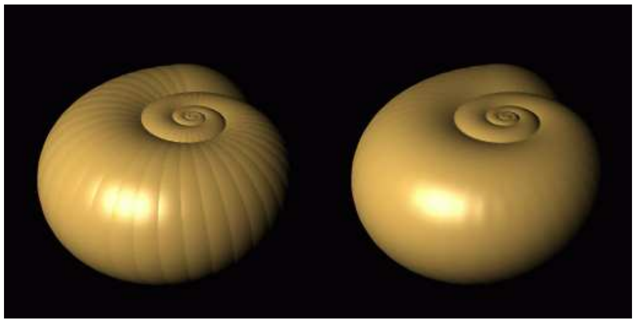


Figure 13. Shell I. The offset surface (left) and the convolved surface (right) are very similar. This dataset served as a base model for Shell II and Shell III. 361 elements.

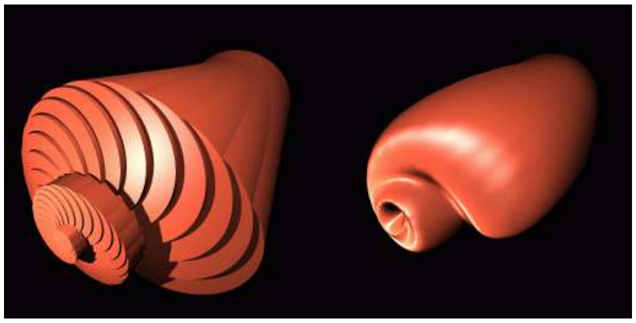


Figure 14. Shell II. The offset surface of this unidentified mollusk is pictured as a union of simple cylinders (instead of sphere-capped cylinders) to emphasize the spiral structure better. 42 elements.

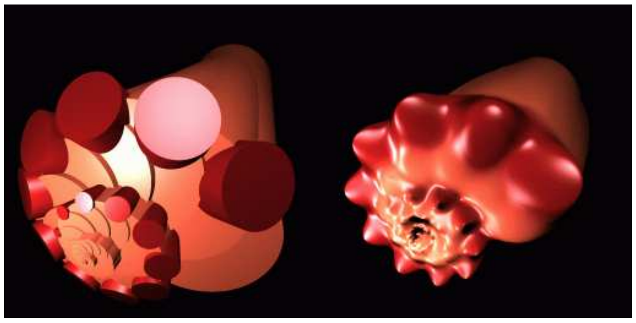


Figure 15. Shell III. Superposition of two skeletons: the orange base (55 line segments) blends with the spiral row of red horns (15 line segments) yielding a smooth convolved shape of a new seashell. Both skeletons are created procedurally.

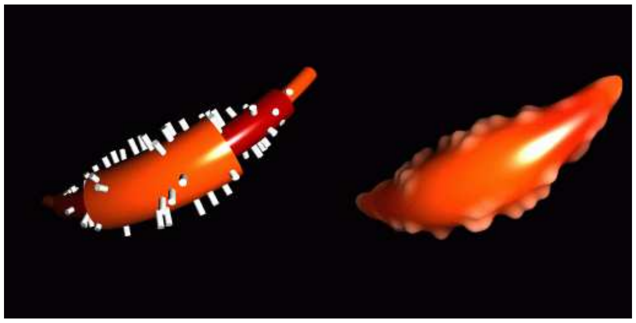


Figure 16. Spindle Cowrie. This very simple model is based on a croissant-like combination of 3 arcs. Additional 100 cylinders make the surface look more interesting.

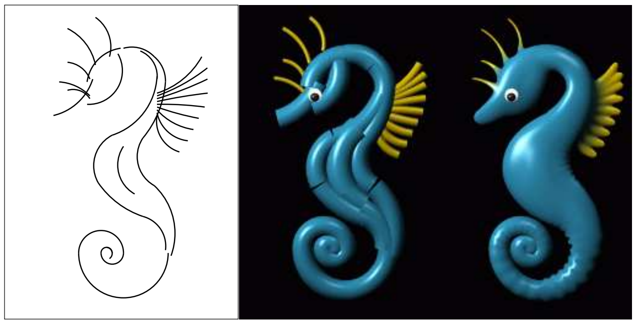


Figure 17. Seahorse. The hand drawing (bare skeleton), the offset surface and the convolved shape. Note how the wrinkles in the back and the tail indicate softness of the skin. 43 elements.

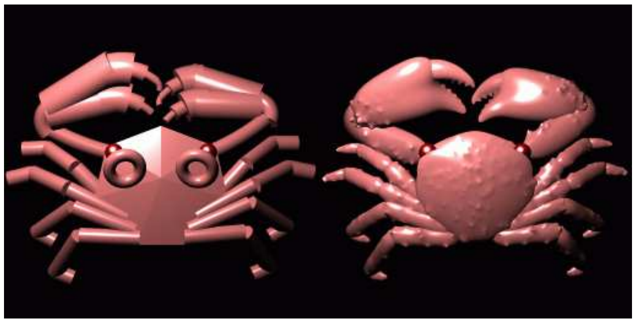


Figure 18. Coral crab. The simplified draft surface consists of 7 polygons (body), 26 arcs (claws, eye sockets), 28 cylinders (legs, claws) and 2 spheres (eyes). The convolved version has additional 608 spikes, scattered along the surface.