ACM Symposium on Solid Modeling and Applications (2004)
G. Elber, N. Patrikalakis, P. Brunet (Editors)

Implicit curve and surface design using smooth unit step functions

Qingde Li and Roger Phillips

Department of Computer Science, University of Hull, Hull, HU6 7RX, UK

Abstract

This paper presents an implicit curve and surface design technique that uses smooth unit step functions. With the proposed method, an implicit curve or surface can be generated by inputting a sequence of points together with the normals at these points of the curve or surface to be designed. By choosing appropriate smooth unit step functions, these curves and surfaces can be designed to any required degree of smoothness.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

1. Introduction

A geometric shape can be represented either parametrically or implicitly. Both have their advantages and disadvantages. However, in the area of computer aided geometric design and computer graphics modelling, parametric curve and surface design techniques, such as B-spline curves and surfaces, have long been used as the primary shape modelling tools. The reasons why parametric shape design techniques are so popular in engineering object design and computer graphics modelling are obvious. Firstly, parametric curves and surfaces are much easier to manipulate when drawing, tessellating, subdividing and bounding. Secondly, the most popular parametric curve and surface design techniques have a simple and elegant description and are very easy to implement. Thirdly, the spline curves and surfaces are represented by piecewise polynomials and are fast to compute. Furthermore, parametric surfaces provide direct support to the polygonal mesh technique for displaying objects graphically. But parametric curves and surfaces have their drawbacks. One main drawback is that it is more computationally expensive to determine the distance from a point to a parametrically represented geometric shape. In general for a parametric curve or a parametric surface, it is impossible to know directly whether a given point is on the shape or not. In addition, for a parametrically represented closed curve or surface, it is hard to know whether a point lies inside or outside the shape. Unlike parametric geometric objects, an implicit shape in $\mathbb{R}^n$ is represented by a mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}$ as the 0-contour of f, or as the set $A = \{\mathbf{P} : f(\mathbf{P}) = 0, \mathbf{P} \in \mathbb{R}^n\}$. Implicitly represented shapes have advantages over parametric shapes in several aspects. Firstly, when objects are modelled as implicit surfaces, one knows directly whether a point lies inside or outside the shape and the problem of boundary detection can be easily solved. Secondly, the surface normals are easy to compute, and two implicit shapes can be easily blended. Thirdly, the most commonly used geometric shapes, such as spheres, cylinders, and ellipsoids, take very simple forms. In addition, the value of an implicit function at a point can be used to approximate the signed distance from the point to the surface.

One of the most active implicit modelling techniques has been the CSG [Ric73] [MS85] [Roc89] [BW90] [Hof93] [PASS95] [PPIK02] [HL02]. With this technique, a complex geometric object can be regarded as the result of a series set-theoretic operations acting on a set of primitive geometric solids. Given that two objects A, B are represented by implicit functions $F_A(\mathbf{P}) \geq 0$ and $F_B(\mathbf{P}) \geq 0$ respectively. Then, the union $A \cup B$ of sets A and B can be represented by the function $F(\mathbf{P}) = \max\{F_A(\mathbf{P}), F_B(\mathbf{P})\}$, the intersection $A \cap B$ by $F(\mathbf{P}) = -\max\{-F_A(\mathbf{P}), -F_B(\mathbf{P})\}$, and the subtraction $A - B$ by $F(\mathbf{P}) = -\max\{-F_A(\mathbf{P}), F_B(\mathbf{P})\}$.

One major problem with the blending operation $\max(x, y)$ is that it is not smooth. As a bivariate function, the value of $\max(x, y)$ changes sharply along the line $y = x$. The blended shapes based on this function always have a sharp edge at the join. Many solutions have been proposed to cope with the problem [MS85] [Roc89] [BW90] [Hof93] [PASS95] [PPIK02]. However, two problems still remain. First of all, most of the blending operations presented so far are not shape preserving and the blending range controllable operations given in [HL02] and [BDS*03] have quite complex mathematical forms. Secondly, with present approaches, it is hard to construct a highly smooth, say C^2 or C^3-smooth,

range controllable blending operations. Recently, a novel solution to these problems has been proposed in [Li04] by using what are known as smooth unit step functions.

The aim of this paper is to develop algorithms similar to spline parametric curve and surface design techniques which allow CAD engineers and computer graphics practitioners to design a rich variety of implicit shapes by simply laying down some points together with normals at these points to the designed shape.

The basic idea behind the design approach presented in this paper is implicit shape subdivision and blending. Although the implicit shapes can be designed by performing a sequence of set theoretic operations on a set of geometric primitives, the shape constructed in this way can be very time consuming as shape complexity increases. One approach to cope with complex implicit shape is to subdivide the construction of the object. To construct a complex object, we can first partition the object into a number of simpler components and then design each component separately. These individually designed objects are then merged together. Usually, a partitioning procedure will result in a binary space partition tree, where each node represents a partitioned object.

The rest of the paper is organized in the following way. In Section 2, we introduce piecewise smooth unit step functions, which will allow us to construct piecewise polynomial shape preserving blending operations up to any required degree of smoothness. One way of constructing this type of blending operation and the use of smooth unit step functions to realize a soft partitioning of the design space is presented in Section 3. The designing strategies are demonstrated in Sections 4 and 5.

2. Piecewise polynomial smooth unit step functions

Definition 1 A real function $\mu : R \to [0, 1]$ is said to be a smooth unit step function, if it satisfies the following conditions:

(1) $\mu(t) = 0$, when $t < -1$;
(2) $\mu(t) = 1$, when $t > 1$;
(3) $\mu(t)$ is continuous and nondecreasing over real line $\mathbb{R}$.

A smooth unit step function can be regarded as a smooth approximation to the Heaviside unit step function, which is defined in the following way:

$$H_0(t) = \begin{cases} 0, & t < 0; \\ \frac{1}{2}, & t = 0; \\ 1, & t > 0. \end{cases} \tag{1}$$

Smooth unit step functions can be constructed in a number of ways and represented by different types of functions [Li04]. One particular simple and elegant technique has been a recursive procedure for constructing piecewise polynomials smooth unit step functions.

Let $H_0(t)$ be the Heaviside unit step function, and let

$$\begin{aligned} f_0(t) &= H_0(t), \\ f_n(t) &= \frac{t}{n} f_{n-1}(t) + (1 - \frac{t}{n}) f_{n-1}(t-1), \\ & \quad n = 1, 2, 3, \cdots . \end{aligned} \tag{2}$$

Set

$$\begin{aligned} H_n(t) &= f_n(\frac{n(t+1)}{2}), \\ & \quad n = 1, 2, 3, \cdots . \end{aligned} \tag{3}$$

Then it can be shown that all of these $H_n(t)$, $n = 0, 1, 2, \ldots$, are nonnegative monotonic increasing function with derivatives up to $n - 1$ order. They take values 0 when $t \leq -1$, and 1 when $t \geq 1$. The smooth unit step functions defined in

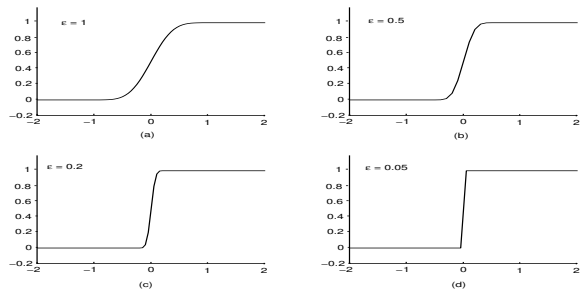

Figure 1: *A C^3 smooth unit step function with different rising ranges. (a) $\epsilon = 1$;(b) $\epsilon = 0.5$;(c) $\epsilon = 0.2$;(d) $\epsilon = 0.05$.*

Definition 1 will always have a support $[-1, 1]$. These functions can be modified to have an arbitrary support.

Definition 2 Let $\mu(t)$ be a smooth unit step function with support $[-1, 1]$, and let ϵ be a positive real number. The real function $\mu(\frac{t}{\epsilon})$, denoted by $\mu_\epsilon(t)$, is called a smooth unit step function with rising range parameter ϵ.

Obviously, a smooth unit step function with rising range parameter ϵ has a support $[-\epsilon, \epsilon]$.

A C^3 piecewise polynomial smooth unit step function constructed from the first method is displayed in figure 1 with different rising parameters. As can be seen later, the rising range parameter can be used to control the size of the transition area between two blending objects.

3. The application of smooth unit step function in blending implicit objects

The two distinctive features of parametric spline curves and surfaces are that they are piecewise polynomial and that they can be designed up to any required degree of smoothness. To develop similar techniques for implicit curve and surface design, two desirable properties of the designed implicit shapes are that they are piecewise algebraic and that they exhibit the necessary smoothness characteristics. Obviously, the implicit shape designed should be a blend of a set of implicit geometric primitives, which are lower degree algebraic curves or surfaces. As with spline basis functions, this will require, firstly, that the blending operations used

for blending these primitive algebraic shapes should be polynomial and controllable over a blending range. Secondly, the blending operation can be designed to the required degree of smoothness. As pointed out in [Li04], the implicit shape blending operation constructed using piecewise smooth unit step functions has both these properties. Next let us investigate how piecewise smooth unit step functions can be used for implicit shape design.

3.1. Smooth piecewise polynomial blending operations

Piecewise unit step functions can be used to construct smooth piecewise polynomial shapes that preserve blending operations. Let $\delta > 0$ and let $\mu_\epsilon(t)$ be a piecewise polynomial smooth unit step function with rising range parameter ϵ, and let

$$\begin{aligned} f(t) &= \frac{x^2}{2\delta} + \frac{\delta}{2}. \\ g_0(t) &= 1 - \mu_\epsilon(t+\delta) \\ g_1(t) &= \mu_\epsilon(t+\delta)(1-\mu_\epsilon(t-\delta)) \\ g_2(t) &= \mu_\epsilon(t-\delta). \end{aligned} \tag{4}$$

Then $g_0(t), g_1(t), g_2(t)$ will be polynomials of the same order when $\epsilon \leq \delta$ and they satisfy

$$g_0(t) + g_1(t) + g_2(t) = 1.$$

The shapes of the three functions are plotted in figure 2. Using these three functions, the function $abs(t) = |t|$ can

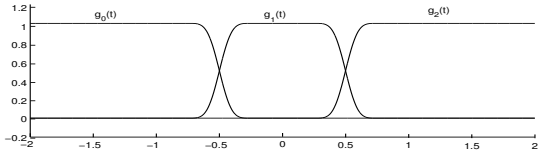

Figure 2: *The functions $g_0(t), g_1(t), g_2(t)$ for $\delta = 0.5$ and $\epsilon = 0.2$.*

be approximated with the following piecewise polynomial smooth function:

$$S_{abs}(t, \delta, \epsilon) = -g_0(t)t + g_1(t)f(t) + g_2(t)t. \tag{5}$$

This function approximates the properties: $S_{abs}(t) = -t$ when $t \in (-\infty, \delta]$; $S_{abs}(t) = t$ when $t \in (\delta, \infty]$; $S_{abs}(t) = f(t)$ when $t \in (-\delta, \delta]$, where $f(t)$ is part of a parabola tangent to both $s(t) = -t$ and $s(t) = t$. It can be shown that S_{abs} has the same degree of smoothness as the piecewise smoothness unit step function μ_ϵ. S_{abs} can be regarded as a smooth approximation to the absolute function $s(t) = |t|$, where δ is used to control the range of smoothness of the function. In practice, ϵ can be set to any positive number smaller than δ, say, 0.5δ. Figure 3 shows the shapes of such a function with different δ. With function S_{abs}, the following smooth function can be defined to approximate $\max(x, y)$:

$$S_{max}(x, y, \delta, \epsilon) = \frac{x + y - S_{abs}(x - y, \delta, \epsilon)}{2}. \tag{6}$$

For two implicit shapes $A : F_A(\mathbf{P}) \geq 0$ and $B : F_B(\mathbf{P}) \geq 0$, the union, intersection, and subtraction of the two

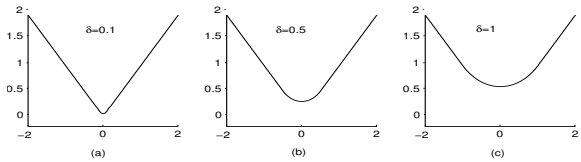

Figure 3: *The shapes of $S_{abs}(t)$ with different δ. (a)$\delta = 0.1$; (b)$\delta = 0.5$; (c)$\delta = 1.0$.*

implicit shapes can be defined by $S_{\max}(F_A(\mathbf{P}), F_B(\mathbf{P}))$, $-S_{\max}(-F_A(\mathbf{P}), -F_B(\mathbf{P}))$, and $-S_{\max}(-F_A(\mathbf{P}), F_B(\mathbf{P}))$ respectively.

The key features of the blending operations defined using $S_{\max}(x, y)$ are that: (1) The blending of algebraic shapes will still be algebraic; (2) The blending of smooth algebraic shapes will still be smooth if the smooth unit step function used is smooth enough.

3.2. Building soft space partitioning

Smooth unit step functions can also be used to construct implicit shapes based on soft space partitioning.

Definition 3 A mapping $b : \mathbb{R}^n \longrightarrow [0, 1]$ is called a base function with respect to a region D in Euclidean space $\mathbb{R}^n$ if $b(t)$ takes value 1 within the region D and the value 0 outside the region, except for those points close to the boundary of the region. A base function $b(\mathbf{P})$ is said to be simple if, for each number $\alpha \in [0, 1]$, the α-level set b_α defined in the following way is a simply connected set in $\mathbb{R}^n$:

$$b_\alpha = \{\mathbf{P} : b(\mathbf{P}) \geq \alpha,\ P \in \mathbb{R}^n\}.$$

For a general base function, all its α-level sets should be topologically identical. The characteristic function of a region is a special type of base function.

Definition 4 Let $\mathcal{G} = \{b_i : i = 0, 1, \cdots, m-1\}$ be a set of base functions on real Euclidean space $\mathbb{R}^n$. $\mathcal{G}$ is said to be a partitioning of $\mathbb{R}^n$, if for each $P \in \mathbb{R}^n$, we have

$$\sum_{i=0}^{m-1} b_i(\mathbf{P}) = 1.$$

For a given partitioning of space $\mathbb{R}$, a smooth soft partitioning can be defined directly using the smooth unit step functions.

For example, if we have three primitives described in figure 4, where the arrows correspond to the outside of each partitioning. Suppose each of these lines is described implicitly as $G_1(\mathbf{P}) = 0$, $G_2(\mathbf{P}) = 0$, $G_3(\mathbf{P}) = 0$ respectively, then a partition can be defined using the following three functions:

$$\begin{aligned} b_1(\mathbf{P}) &= \mu(G_1(\mathbf{P})), \quad b_2(\mathbf{P}) = \mu(G_2(\mathbf{P})), \\ b_3(\mathbf{P}) &= \mu(G_3(\mathbf{P})) \end{aligned} \tag{7}$$

where μ is a piecewise polynomial smooth unit step function. The four base functions corresponding to the four areas are:

$$1 - b_1, \quad b_1(1 - b_2), \quad b_1 b_2(1 - b_3) \quad b_1 b_2 b_3. \tag{8}$$

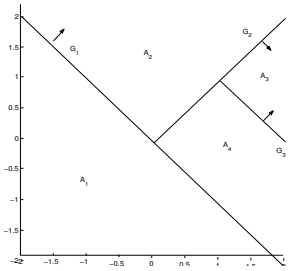

Figure 4: *The 2D plane is partitioned into four areas using three lines G_1, G_2, G_3.*

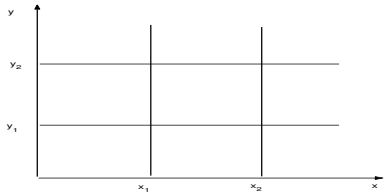

Figure 5: *The 2D plane is partitioned into nine areas using four lines.*

Another more direct way of partitioning a space is to divide the space into a relatively regular grid using straight lines (2D) and planes (3D). In figure 5, 2D space is partitioned into nine areas using four straight lines parallel to the coordinate axes. In the figure, if we define x_0 and y_0 to be $-\infty$, and x_3 and y_3 to be ∞ and define $\mu(\infty) = 1$, then the corresponding soft partitioning can be represented by the following nine base functions.

$$b_{i,j}(x, y, \epsilon) = b_{1D}(x, x_i, x_{i+1}, \epsilon) b_{1D}(y, y_j, y_{j+1}, \epsilon), \qquad (9)$$

where $(i = 0, 1, 2; j = 0, 1, 2)$, and $b_{1D}(t, t_0, t_1, \epsilon)$ is a function defined by a smooth unit step function μ as

$$b_{1D}(t, t_0, t_1, \epsilon) = \mu((t - t_0)/\epsilon)\mu((t_1 - t)/\epsilon). \qquad (10)$$

If a piecewise polynomial smooth unit function is used in (10), then $b_{1D}(t, t_0, t_1, \epsilon)$ will also be a piecewise polynomial as will each base function in (9). If a cubic smooth unit step function is used, then each $b_{i,j}(x, y, \epsilon)$ will be a bi-cubic piecewise polynomial.

4. Implicit curve design

In this section, we introduce two implicit curve design techniques using the smooth unit step functions. With these techniques, designers need only place a sequence of shape control points and shape control normals. All these techniques can be directly implemented to produce an interactive implicit shape design package.

4.1. Design of implicit curves using blending operations

For the shape to be designed, let $\{\mathbf{P}_i, \mathbf{n}_i\}_{i=0}^m$ be a set of $m+1$ control points and control normals at these points. To design an implicit curve using this information, we attach

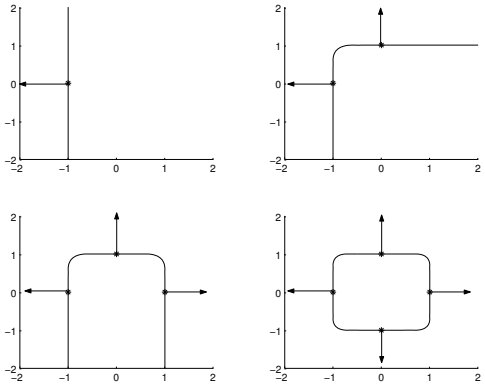

Figure 6: *Interactive curve design by sequentially inputting of a control point and a control normal.*

to each of these control points $\mathbf{P}_i$ a primitive implicit curve which passes through the control point and has a normal $\mathbf{n}_i$. In the simplest situation, a straight line can be used. Using a straight line as the shape primitive to each point will result in a convex shape. To increase the flexibility in designing a rich set of implicit shapes, more general primitives can be used. For example, instead of using straight lines as the primitive shape, parabolas with parameters controlling parameters concerning changing rate and orientation can be used. The algorithm is described as follows.

$F = Shape(\mathbf{P}_0, \mathbf{n}_0, shapeParameters);$
$for(i = 1; i \leq m; m++)\{$
$\quad F = blend(F, Shape(\mathbf{P}_i, \mathbf{n}_i, shapeParameters))\};$

To avoid an unbounded curve, we can begin with a simple base shape such as an ellipse. Figure 6 shows how the design proceeds. The designer first lays down the first control point and the control normal for the curve using a mouse. The primitive shape used here is a parabola and the length of the normal to the parabola is used to control its bending degree. A shorter normal corresponds to a relatively shallow parabola. The length of the normal can be defined to vary within a certain range. For instance, when the length is 1, the primitive corresponds to a straight line and a length larger than 1 corresponds to a convex shape and a length smaller than 1 corresponds to a concave shape (see figure 7).

4.2. Design of implicit curves by space partitioning

Another way of designing implicit curves is based on space partitioning. To design an implicit pattern with this method, we first partition the space into several areas. The implicit shape can then be designed area by area. In each area, we can apply the technique provided in the previous section to design the part of the implicit curve lying in this area. For example, with the partitioning given in figure 4, we can design the shape presented in figure 9 in the following way. We design the shape corresponding to the area of A_1 as a circle, and design the shape in the area of A_2 as a line, and so on. In general, let us assume that we have a soft parti-

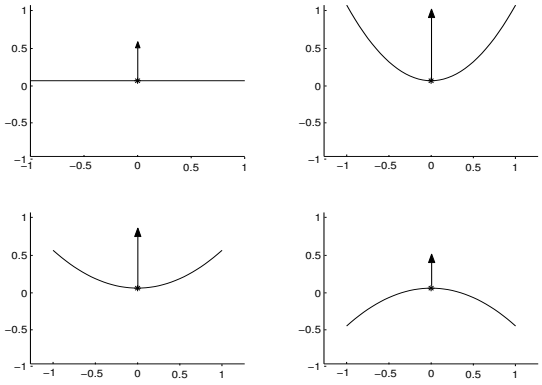

Figure 7: *Design primitives with controllable location, orientation and blending degree.*

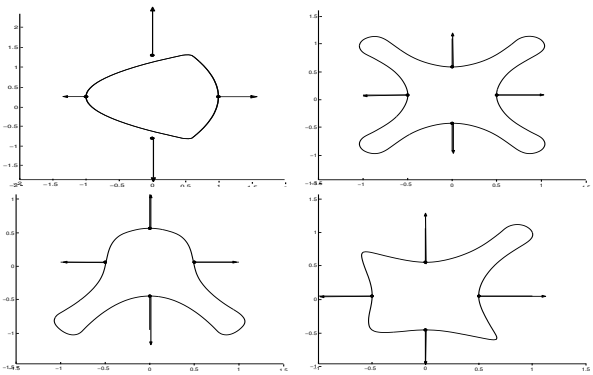

Figure 8: *Implicit curve designed with four pairs of control points and normals.*

tion $\{b_i(\mathbf{P})\}_{i=0}^{m-1}$ for a 2D plane, where $b_i(\mathbf{P})$ corresponds to region D_i. Let the implicitly designed shape within the region D_i be $F_i(\mathbf{P})$, then the overall design can be expressed implicitly as $F(\mathbf{P}) = 0$, where

$$F(\mathbf{P}) = \sum_{i=0}^{m-1} b_i(\mathbf{P}) F_i(\mathbf{P}). \tag{11}$$

If we regard each $b_i(\mathbf{P})$ as a blending basis function and each $F_i(\mathbf{P})$ as a control design primitives, we have obtained a design method very similar to the parametric spline curve and surface design.

With the presented techniques, the designer can decompose a complex shape into some simpler geometric primitives. When the type of primitive has been selected, the designer need only specify the location and orientation of the selected implicit shape by placing a control point and a control normal. The geometric primitive is then localized by multiplying the base function corresponding to the design region.

Figure 10 presents some more examples for this techniques. The whole 2D design area has been sub-divided into nine region using four straight lines. The geometric primitive

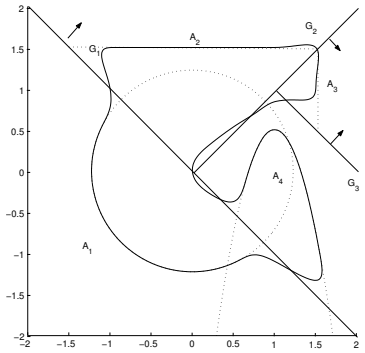

Figure 9: *Different shapes can be freely laid in the four areas (represented as dotted lines). These shapes are combined using corresponding base functions (shown as solid line).*

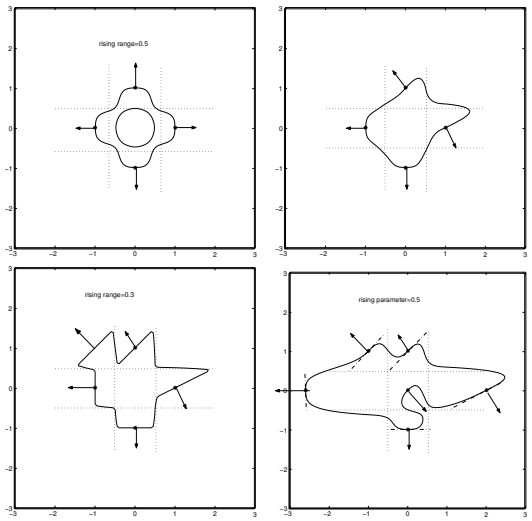

Figure 10: *Design implicit shapes by laying down a few points and the normals to the designed shape*

used in each region is just a straight line. The rising range parameter ϵ can be used to control the smooth transition among those locally designed shapes. Figure 11 shows that with the increase in the rising range parameter, the shape transition from one to another becomes smoother. However, smaller rising parameters are required if we do not want a big change in the locally designed shapes.

4.3. Implicit surface design

Techniques similar to the implicit curve can be directly generalized for implicit surface design. Here we just provide a simple example to demonstrate how to design an implicit surface by inputting a few points plus the normals at these points to the designed surface. The implicit objects presented in figure 13 are obtained by just partitioning the space into eighteen parts (see figure 12). Cubic piecewise polynomial smooth unit step functions have been used to construct the blending basis functions and implicit planes are used for each subspace. Therefore all the surfaces shown

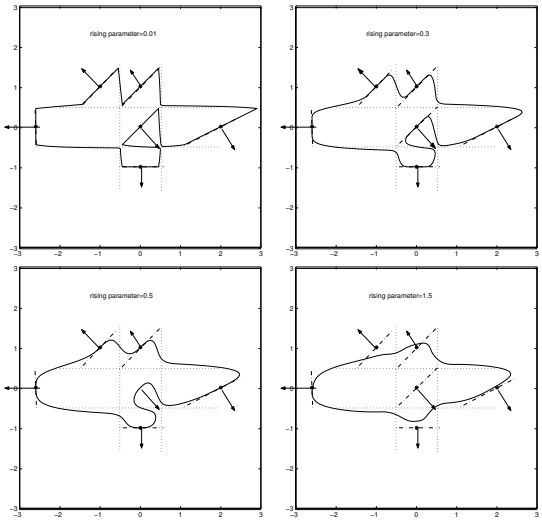

Figure 11: *Modification the smoothness in shape transition by using different rising range parameters.*

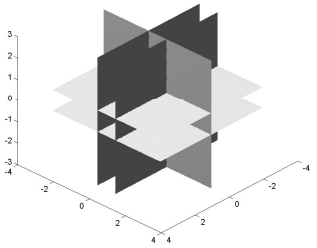

Figure 12: *An example of partitioning 3D space.*

in the figure are piecewise algebraic. As can be seen, a rich set of implicit surface can be directly generated using this technique when higher degree of algebraic surfaces are used as design primitives.

5. Conclusion

In this paper, we have presented a curve and surface design technique concerning space partition. The technique presented is quite similar to the parametric spline curve and surface technique. The potential applications of the presented technique include parallel computing in implicit design, implicit surface shading and implicit surface parameterization.

References

[BDS*03] Barthe L., Dodgson N. A., Sabin M. A., Wyvill B., Gaildrat V.: Two-dimentional potential fields for advanced implicit modeling operators. *Computer Graphics Forum 22*, 1 (2003), 23–33.

[BW90] Bloomenthal J., Wyvill B.: Interactive techniques for implicit modeling. *Symposium on Interactive 3D Graphics 24*, 2 (1990), 109–116.

[HL02] Hsu P. C., Lee C.: The scale method for blending operations in functionally-based constructive geometry. *Computer Graphics Forum 22*, 2 (2002), 143–158.

[Hof93] Hoffmann C. M.: Implicit curves and surfaces in cagd. *IEEE Computer Graphics and Appl 13* (Jan 1993), 79–88.

[Li04] Li Q.: Blending implicit shapes using smooth unit step functions. *Short Communication Papers Proceedings of WSCG'2004* (2004), 297–304.

[MS85] Middleditch A. E., Sears K. H.: Blend surfaces for set-theoretic volume modelling systems. *Proceedings of SIGGRAPH 85* (1985), 161–170.

[PASS95] Pasko A. A., Adzhiev V., Sourin A., Savchenko V. V.: Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11*, 8 (1995), 429–446.

[PPIK02] Pasko G., Pasko A. A., Ikeda M., Kunii T. L.: Bounded blending operations. *Proc. of Shape Modeling International* (2002), 95–104.

[Ric73] Ricci A.: A constructive geometry for computer graphics. *Computer Journal 16*, 3 (May 1973), 157–160.

[Roc89] Rockwood A. P.: The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics 8*, 4 (1989), 279–297.

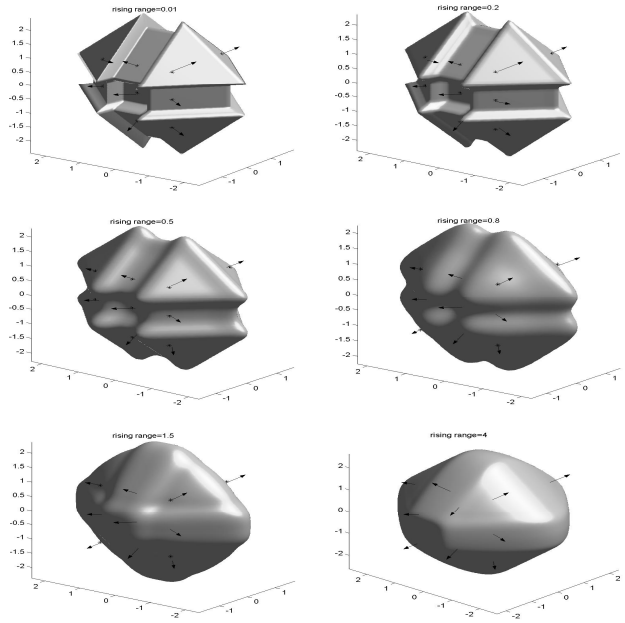

Figure 13: *Design implicit surface by placing a few control points and normals at these points to the designed shape.*