2D Piecewise Algebraic Splines for Implicit Modeling

QINGDE LI
University of Hull
and
JIE TIAN
The Chinese Academy of Sciences

2D splines are a powerful tool for shape modeling, either parametrically or implicitly. However, compared with regular grid-based tensor-product splines, most of the high-dimensional spline techniques based on nonregular 2D polygons, such as box spline and simplex spline, are generally very expensive to evaluate. Though they have many desirable mathematical properties and have been proved theoretically to be powerful in graphics modeling, they are not a convenient graphics modeling technique in terms of practical implementation. In shape modeling practice, we still lack a simple and practical procedure in creating a set of bivariate spline basis functions from an arbitrarily specified 2D polygonal mesh. Solving this problem is of particular importance in using 2D algebraic splines for implicit modeling, as in this situation underlying implicit equations need to be solved quickly and accurately. In this article, a new type of bivariate spline function is introduced. This newly proposed type of bivariate spline function can be created from any given set of 2D polygons that partitions the 2D plane with any required degree of smoothness. In addition, the spline basis functions created with the proposed procedure are piecewise polynomials and can be described explicitly in analytical form. As a result, they can be evaluated efficiently and accurately. Furthermore, they have all the good properties of conventional 2D tensor-product-based B-spline basis functions, such as non-negativity, partition of unit, and convex-hull property. Apart from their obvious use in designing freeform parametric geometric shapes, the proposed 2D splines have been shown a powerful tool for implicit shape modeling.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*; *splines*

General Terms: Algorithm, Design

Additional Key Words and Phrases: Algebraic splines, CSG, isosurface, level set, function-based shape modeling, implicit curve, implicit modeling, implicit surface

ACM Reference Format:

Li, Q. and Tian, J. 2009. 2D piecewise algebraic splines for implicit modeling. ACM Trans. Graph. 28, 2, Article 13 (April 2009), 19 pages.
DOI = 10.1145/1516522.1516524 http://doi.acm.org/10.1145/1516522.1516524

1. INTRODUCTION

Implicit modeling has become increasingly popular in recent years with the significant advance in computer graphics hardware. Today's graphics hardware is not only good at processing polygonal meshes, but can also be programmed and used as a general-purpose computing device. The ever increasing processing power in modern GPUs is gradually removing the bottleneck of displaying implicitly represented geometric shapes [Purcell et al. 2005; Loop and Blinn 2006; Tatarchuk et al. 2007] implicitly.

So far, implicit shape design has been dominated by the ideas of CSG and blobs. With CSG, a geometric object is created by performing a sequence of set-theoretic operations on a collection of simple geometric primitives. Unlike CSG, the blob-based technique models geometric shapes as the accumulation of a set of particle field functions. Though the two techniques are powerful in modeling certain types of implicit shapes, they are not very flexible in creating implicitly represented freeform geometric objects, one of the most frequently used type of geometric objects in computer graphics, CAGD, and computer games. Although some conventional nontensor-product-based bivariate splines are available for freeform implicit geometric modeling, such as box splines and simplex splines, they are generally very expensive to evaluate, especially when the degree of smoothness required is high. In the situation of implicit modeling, it is becoming even more impractical to use these kinds of splines for implicit geometric shape design, due to the fact that their explicit mathematical expressions are generally not known. In this article, we aim to develop a technique for freeform implicit

This article is partly supported by the Project for the National Basic Research Program of China (973) under Grant No. 2006CB705700, Changjiang Scholars and Innovative Research Team in University (PCSIRT) under Grant No. IRT0645, CAS Hundred Talents Program, the National Natural Science Foundation of China under Grant No. 60532050, Beijing Natural Science Fund under Grant No. 4071003.
Authors' addresses: Q. Li, Department of Computer Science, University of Hull, Hull, HU6 7RX, UK; email: Q.Li@hull.ac.uk; J. Tian, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China; email: tian@ieee.com.

DOI 10.1145/1516522.1516524 http://doi.acm.org/10.1145/1516522.1516524

shape design similar to the conventional parametric splines for parametric freeform shape design. To understand the importance of this technique in designing implicit freeform geometric shapes, we need only look at how parametric spline-based curve and surface design techniques have revolutionized the shape design engineering.

The main challenge to be tackled in the development of such a technique is, for any given set of 2D polygons or 3D polyhedra, how to construct a set of analytically represented bivariate (for 2D polygons) or trivariate (for 3D polyhedra) spline basis functions that have all the good properties possessed by ordinary tensor product B-spline functions.

More precisely, let $\{\Delta_k : \Delta_k \subseteq \mathcal{R}^n\}$, n=2, 3, be a set of polygons (n=2) or polyhedra (n=3) that partition the space $\mathcal{R}^n$, such that

$$\bigcup \Delta_k = \mathcal{R}^n, \qquad area(\Delta_i \bigcap \Delta_j) = 0 \text{ when } i \neq j. \tag{1}$$

For each Δ_k, we wish to construct a function

$$B_{\Delta_k}(\mathbf{P}) : \mathcal{R}^n \longrightarrow \mathcal{R}, \tag{2}$$

such that it has the following properties:

(1) Each $B_{\Delta_k}(\mathbf{P})$ is a piecewise polynomial function and can be built to have any required degree of smoothness.

(2) $B_{\Delta_k}(\mathbf{P})$ is non-negative.

(3) Partition of unit:

$$\sum_k B_{\Delta_k}(\mathbf{P}) = 1, \qquad \mathbf{P} \in \mathcal{R}^n. \tag{3}$$

(4) $\{B_{\Delta_k}(\mathbf{P})\}$ are additive. Specifically,

$$B_{\cup_k \Delta_k}(\mathbf{P}) = \sum_k B_{\Delta_k}(\mathbf{P}). \tag{4}$$

In order to make our study more focused, this article will only consider the construction of this type of spline functions from a set of 2D polygons and their applications in implicit shape modeling. Our findings on how to construct trivariate spline functions with regard to 3D polyhedra will be presented in a separate paper.

The Figures 1(b) to (d) demonstrate some C^2-smooth spline basis functions built using the technique presented in this article upon the set of partition polygons shown in Figure 1(a). All these three sets of spline basis functions are differed by the values of a so called polygon smoothing parameter δ used in the process of constructing these bivariate spline basis functions. They have all the properties listed previously.

In the rest of the article, we first give a brief review on related work about implicit modeling using implicit splines in Section 2. In Section 3, the process of constructing the proposed spline basis functions will be presented. In Sections 4 and 5, some applications of the proposed binary algebraic splines in designing implicit shapes will be demonstrated. As will be seen later, the proposed design scheme is both theoretically elegant and can be implemented easily in practice.

2. BACKGROUND AND RELATED WORK

2.1 Implicit Modeling

Implicit geometric shapes can be described in several different ways. The most intuitive way is to specify an implicit shape as a collection of points contained within a geometric object. This view of a solid geometric object leads directly to one of the most popular implicit shape modeling techniques, known as Constructive Solid Geometry (CSG) [Ricci 1973]. By CSG, a general geometric object can be considered the result of a sequence of set-theoretic operations performed on a set of simple geometric primitives. The second way of describing an implicit geometric object is to view a geometric shape as the contour or the level set of a certain field function, and the corresponding implicit modeling technique is frequently referred to as level-set method, function-based shape modeling, or isosurface modeling. Blinn's popular blob technique [Blinn 1982] is a typical example developed from this idea. In general, an implicit shape can be viewed as a real function [Pasko et al. 1995] $F(\mathbf{P})$ with $\mathbf{P} \in \mathcal{R}^n$, $(n = 2, 3)$. A real function naturally associates with a geometric shape in $\mathcal{R}^n$ either by representing its surface as the real roots of the equation $F(\mathbf{P}) = 0$ or as a solid defined by the set of points $\{\mathbf{P} : F(\mathbf{P}) \geq 0\}$ or $\{\mathbf{P} : F(\mathbf{P}) \leq 0\}$. Representing a geometric object implicitly as a real function is of particular importance for certain computational tasks, such as calculating when a ray hits an implicitly represented geometric object, finding when a point in space is inside or outside of a given shape, or performing objects' collision detection.

Implicitly represented geometric objects have several advantages over parametric shapes. Firstly, different implicit shapes are inherently easy to combine in various ways [Pasko et al. 2005; Li 2007; Barthe et al. 2003; Hsu and Lee 2002], which makes implicit function-based shape design more intuitive. Secondly, ray tracing implicit surfaces is much less expensive than ray tracing a polygonal mesh. Implicit shapes are also a favorite kind of geometric object when performing collision detection. Implicit modeling is particularly preferred in those situations when accurate simulation of certain phenomena occurring in our physical world is required.

2.2 Implicit Multivariate Spline

Although various implicit curve and surface modeling techniques proposed so far can be used to design freeform implicit shapes, most of them are not freeform specific. They are either computationally expensive, such as distance field-based [Payne and Toga 1992] and approximation-and-interpolation-based techniques [Turk and O'Brien 2002, 1999; Li et al. 2004; Shen et al. 2005], or lack the convenience and flexibility in designing freeform implicit shapes like the blob-based method. Some considerations for modeling freeform implicit shapes have been given similarly to the conventional parametric spline techniques by defining an implicit freeform surface as the weighted sum of multivariate splines [Li and Phillips 2004]. The main idea in this approach is to partition the object space into a set of small areas according to the required shape features. In this approach, an implicit function with local support is built for each subarea. The overall required implicit shape is then described implicitly as

$$F(\mathbf{P}) = \sum_{k=0}^{m} P_k(\mathbf{P}) B_{\Delta_k}(\mathbf{P}), \tag{5}$$

where each $P_k(\mathbf{P})$ is in general an implicit function representing a local feature of the wanted shape, and each $B_{\Delta_k}(\mathbf{P})$ is a kind of multivariate spline basis function satisfying condition (3). Modeling implicit shape in this way has been referred to as implicit spline in Li [2005]. Freeform implicit surfaces can also be designed using the A-patch introduced in Bajaj et al. [1994], where the overall shape is described piece by piece locally as algebraic surfaces across the given polyhedral net. However, this technique does not provide us a globally defined implicit function explicitly, which is fundamentally different from the idea presented in this article.

As can be seen, the key to the freeform implicit modeling in the form presented in Eq. (5) is the construction of a set of bivariate functions $B_{\Delta_k}(\mathbf{P})$, $k = 0, 1, 2, \ldots, m$, from any given set of polygons

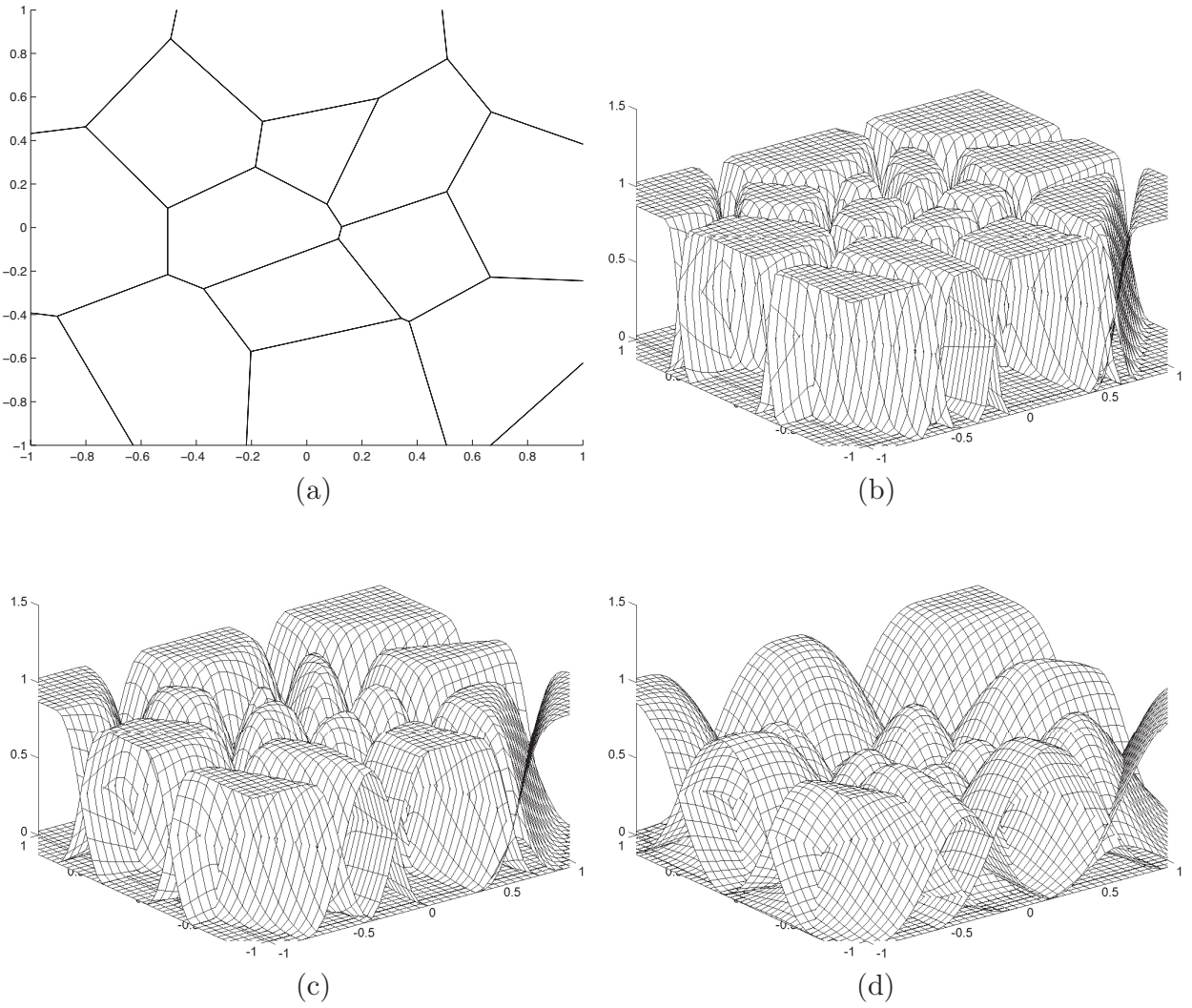


Fig. 1. The C^2-smooth implicit spline basis functions created from an arbitrarily specified partition net of the 2D plane, with different values of polygon smoothing parameter δ. (a) a set of partition polygons; (b) through (d) the spline basis functions constructed from the polygons shown in (a) with the polygon smoothing parameter δ as 0.05, 0.1, and 0.2, respectively.

$\{\Delta_k\}_{k=0}^m$. Naturally, we expect that the set of spline basis functions constructed from the set of the polygons satisfies the properties 1 to 4 listed in the first section.

Compared with the construction of 1D spline basis functions, constructing 2D and 3D smooth piecewise polynomial spline basis functions having similar geometric properties to the univariate spline basis functions is an extremely tough task. In the 1D case, a point is the only type of geometrical object that separates a real line into two parts, and an interval is the only type of connected set that serves as the support of a univariate spline basis function. However, in 2D and 3D or higher dimensions, the situation is much more complicated. For example, in the case of 2D, there are infinitely many different types of geometric objects that can separate a plane into two simply connected areas, and there is a variety of different kinds of simply connected sets. Consequently, in practice, how to create the required bivariate spline basis functions depends on how the space has been partitioned. If the space is partitioned with a regular grid, a set of multivariate splines can be built easily as the tensor product of univariate B-splines. However, when the space is partitioned using an irregular grid, we still do not have a theoretically elegant and practically usable technique to construct the set of spline basis functions from the given partition set. Though simplex and box splines can be used to create multivariate splines, they are very expensive to evaluate in general [Fong and Seidel 1992], which makes them quite impractical to use in shape design practice. Some less expensive splines have been proposed. For instance, in Li and Phillips [2004] and Li [2005], line-based piecewise polynomial splines were used to build polygon-based spline basis functions. However, these functions are not in general additive in the sense of Eq. (4), and the corresponding shape design schemes do not in general possess some good geometric properties observed in traditional spline-based shape design techniques.

As will be seen later, the type of bivariate spline basis functions proposed in this article meet all the requirements described

in Section 1. In particular, this type of spline is piecewise polynomial and is expressed explicitly in analytical form, which can be evaluated extremely efficiently.

3. BIVARIATE PIECEWISE POLYNOMIAL SPLINES

The main idea used here to construct the required bivariate spline basis functions is to find a general solution to the following integral convolution. Let $\square \subset \mathcal{R}^2$ be a square of size $2\delta \times 2\delta$ centered at the coordinate origin with $\delta > 0$. For an arbitrarily given polygon $\Delta \subset \mathcal{R}^2$, we define a sequence of functions as

$$B^{(0)}_{\Delta,\delta}(x, y) = \chi_\Delta(x, y) = \begin{cases} 1, & (x, y) \in \Delta; \\ 0, & (x, y) \notin \Delta. \end{cases},$$

$$B^{(n)}_{\Delta,\delta}(x, y) = \frac{1}{4\delta^2} \int\int_{\mathcal{R}^2} B^{(n-1)}_{\Delta,\delta}(s, t)\chi_\square(s - x, t - y)dsdt, \quad (n > 0). \tag{6}$$

The parameter δ in the integral serves as a solid polygon softening parameter and will be referred to as a polygon smoothing parameter in this article. With the properties of integration it can be seen clearly that each $B^{(n)}_{\Delta,\delta}(x, y)$ has the following two properties. First, $B^{(n)}_{\Delta,\delta}(x, y)$ is a piecewise polynomial function. Second, $B^{(n)}_{\Delta,\delta}(x, y)$ will be C^{n-1} continuous. Though this idea of constructing the required spline basis functions is simple, the evaluation of these functions can be quite expensive if they cannot be expressed explicitly in analytical form. Fortunately, we have in this research found a way to express these functions explicitly. To avoid diverting readers' attention, we will first describe directly in what a way these functions can be expressed explicitly and how they can be applied to design implicit shapes, while the detailed reasoning process in finding these explicit forms will be outlined in the Appendix.

To express function (6) explicitly, we first introduce a bivariate function $A^{(n)}_{\alpha,\beta}(x, y)$ for each positive integer n and a pair of positive number $\alpha > 0, \beta > 0$. We have

$$A^{(n)}_{\alpha,\beta}(x, y) = \begin{cases} 0, & y \geq \min\left\{\frac{\beta}{\alpha}x, 0\right\}; \\ \frac{1}{(2n)!\alpha^n\beta^n}(\beta x - \alpha y)^{2n}, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x \leq 0; \\ \sum_{k=1}^{n} \frac{(-1)^{n+k}\alpha^k}{(n-k)!(n+k)!\beta^k} x^{n-k} y^{n+k}, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x > 0. \end{cases} \tag{7}$$

With function $A^{(n)}_{\alpha,\beta}(x, y)$, the following function can be defined with a number $\delta > 0$. We have

$$\Omega^{(n)}_{\alpha,\beta,\delta}(x, y) = \frac{1}{(4\delta^2)^n} \sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{i+j} C^i_n C^j_n F_{i,j}(x, y), \tag{8}$$

where

$$F_{i,j}(x, y) = A^{(n)}_{\alpha,\beta}(x + (n - 2i)\delta,\ y - (n - 2j)\delta).$$

It is obvious that $\Omega^{(n)}_{\alpha,\beta,\delta}(x, y)$ is piecewise polynomial. It can also be seen directly that $\Omega^{(n)}_{\alpha,\beta,\delta}(x, y)$ is C^{n-1} continuous.

As can be seen later, the spline basis function $B^{(n)}_{\Delta,\delta}(x, y)$ can, in general, be expressed immediately using $\Omega^{(n)}_{\alpha,\beta,\delta}(x, y)$. As a result, the bivariate function $\Omega^{(n)}_{\alpha,\beta,\delta}(x, y)$ plays a central role in building the required spline basis functions. Since C^1 and C^2 smooth bivariate splines are most frequently used in practice, as an illustration, we describe here more explicitly how $\Omega^{(n)}_{\alpha,\beta,\delta}(x, y)$ is defined for the cases of $n = 1, 2, 3$.

Case 1. When n=1, we have

$$A^{(1)}_{\alpha,\beta}(x, y) = \begin{cases} 0, & y \geq \min\left\{\frac{\beta}{\alpha}x, 0\right\}; \\ \frac{1}{2\alpha\beta}(\beta x - \alpha y)^2, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x \leq 0; \\ \frac{\alpha}{2\beta}y^2, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x > 0. \end{cases} \tag{9}$$

From (8), we can see that the value of $\Omega^{(1)}_{\alpha,\beta,\delta}(x, y)$ at $\mathbf{P}(x, y)$ is a linear combination of the values of the function $A^{(1)}_{\alpha,\beta}(x, y)$ at positions $\mathbf{P}_{00}(x - \delta, y + \delta)$, $\mathbf{P}_{10}(x - \delta, y - \delta)$, $\mathbf{P}_{01}(x + \delta, y + \delta)$, and $\mathbf{P}_{11}(x + \delta, y - \delta)$. In other words,

$$\Omega^{(1)}_{\alpha,\beta,\delta}(x, y) = \frac{1}{4\delta^2}\Big[A^{(1)}_{\alpha,\beta}(\mathbf{P}_{00}) - A^{(1)}_{\alpha,\beta}(\mathbf{P}_{10}) - A^{(1)}_{\alpha,\beta}(\mathbf{P}_{01}) + A^{(1)}_{\alpha,\beta}(\mathbf{P}_{11})\Big],$$

which can be illustrated using the following mask (see Figure 2), where the numbers in the circles are the coefficients used to combine the values of $A^{(1)}_{\alpha,\beta}(x, y)$ evaluated at the corresponding positions.

Case 2. In the case of $n = 2$, $\Omega^{(2)}_{\alpha,\beta,\delta}(x, y)$ is defined using $A^{(2)}_{\alpha,\beta}(x, y)$, whose nonzero part is described by two pieces of polynomials of degree 4. Specifically,

$$A^{(2)}_{\alpha,\beta}(x, y) = \begin{cases} 0, & y \geq \min\left\{\frac{\beta}{\alpha}x, 0\right\}; \\ \frac{1}{4!(\alpha\beta)^2}(\beta x - \alpha y)^4, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x \leq 0; \\ \frac{1}{4!}(\frac{\alpha}{\beta})^2 y^4 - \frac{1}{3!}(\frac{\alpha}{\beta})xy^3, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x > 0. \end{cases} \tag{10}$$

From (8), $\Omega^{(2)}_{\alpha,\beta,\delta}(x, y)$ can be calculated directly from $A^{(2)}_{\alpha,\beta}(x, y)$ in the following way.

$$\Omega^{(2)}_{\alpha,\beta,\delta}(x, y) = \tfrac{1}{(4\delta^2)^2}\Big[A^{(2)}_{\alpha,\beta}(\mathbf{P}_{00}) - 2A^{(1)}_{\alpha,\beta}(\mathbf{P}_{10}) + A^{(2)}_{\alpha,\beta}(\mathbf{P}_{20}) - 2A^{(2)}_{\alpha,\beta}(\mathbf{P}_{01}) + 4A^{(2)}_{\alpha,\beta}(\mathbf{P}_{11}) - 2A^{(2)}_{\alpha,\beta}(\mathbf{P}_{21}) + A^{(2)}_{\alpha,\beta}(\mathbf{P}_{02}) - 2A^{(1)}_{\alpha,\beta}(\mathbf{P}_{12}) + A^{(2)}_{\alpha,\beta}(\mathbf{P}_{22})\Big]$$

The corresponding mask for finding $\Omega^{(2)}_{\alpha,\beta,\delta}(x, y)$ is shown in Figure 3.

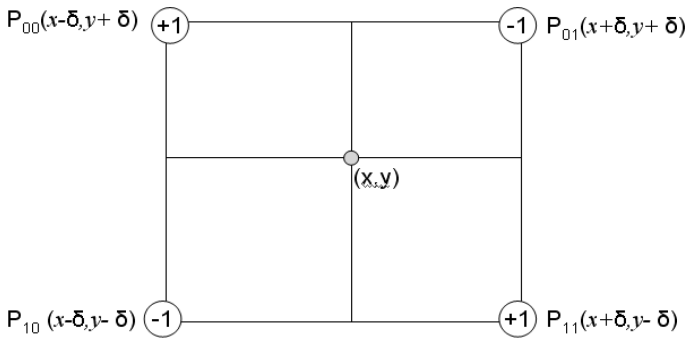


Fig. 2.

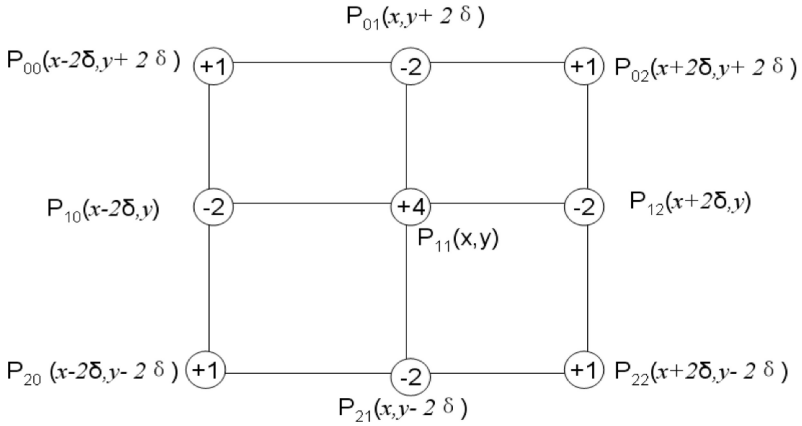


Fig. 3.

Case 3. For $n = 3$,

$$A^{(3)}_{\alpha,\beta}(x, y) = \begin{cases} 0, & y \geq \min\left\{\frac{\beta}{\alpha}x, 0\right\}; \\ \frac{1}{6!(\alpha\beta)^3}(\beta x - \alpha y)^6, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x \leq 0; \\ \frac{1}{6!}(\frac{\alpha}{\beta})^3 y^6 - \frac{1}{5!}(\frac{\alpha}{\beta})^2 xy^5 + \frac{1}{2!4!}(\frac{\alpha}{\beta})x^2y^4, & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x > 0, \end{cases} \tag{11}$$

and $\Omega^{(3)}_{\alpha,\beta,\delta}(x, y)$ is the weighted sum of the values of $A^{(3)}_{\alpha,\beta}(x, y)$ at 16 grid positions shown in Figure 4.

Specifically,

$$\begin{aligned} \Omega^{(3)}_{\alpha,\beta,\delta}(x, y) = \tfrac{1}{(4\delta^2)^3}\Big[& \\ & A^{(3)}_{\alpha,\beta}(\mathbf{P}_{00}) - 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{10}) + 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{20}) - A^{(3)}_{\alpha,\beta}(\mathbf{P}_{30}) \\ & - 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{01}) + 9A^{(3)}_{\alpha,\beta}(\mathbf{P}_{11}) - 9A^{(3)}_{\alpha,\beta}(\mathbf{P}_{21}) + 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{31}) \\ & + 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{02}) - 9A^{(3)}_{\alpha,\beta}(\mathbf{P}_{12}) + 9A^{(3)}_{\alpha,\beta}(\mathbf{P}_{22}) - 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{32}) \\ & - A^{(3)}_{\alpha,\beta}(\mathbf{P}_{03}) + 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{13}) - 3A^{(3)}_{\alpha,\beta}(\mathbf{P}_{23}) + A^{(3)}_{\alpha,\beta}(\mathbf{P}_{33})\Big]. \end{aligned}$$

To express the integral defined in (6) explicitly, we also need the following smooth unit step functions introduced in Li et al. [2006].

We have

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

$$\begin{aligned} H_n(x) = \frac{1}{2}\Big(&\Big(1 + \frac{x}{n}\Big) H_{n-1}(x + 1) \\ &+ \Big(1 - \frac{x}{n}\Big) H_{n-1}(x - 1)\Big), \\ & n = 1, 2, 3, \ldots. \end{aligned} \tag{12}$$

It has been shown that all these generalized unit step functions $H_n(t)$, $n = 1, 2, \ldots$, are non-negative monotonically increasing functions with derivatives everywhere up to $n - 1$ order. They take value 0 when $t \leq -n$, and 1 when $t \geq n$.

To define implicit spline, similar to the concepts of 2D vertex, 2D polygon edge, and 2D polygon, we introduce the concepts of implicit vertex, implicit edge, and implicit polygon in $\mathcal{R}^2$ using the bivariate function $\Omega^{(n)}_{\alpha,\beta,\delta}(x, y)$ and univariate function $H_n(x)$ defined in (8) and (12). These concepts play an important role in the process of constructing our spline basis functions. In fact, for a given polygonal net that partitions $\mathcal{R}^2$, the underlying bivariate spline basis functions built according to (6) are essentially a set of implicit polygons.

Definition 3.1 (*Implicit Vertex*). Let δ be a real number with $\delta > 0$. The order n directed implicit vertex positioned at $\mathbf{P}(x_0, y_0)$ with

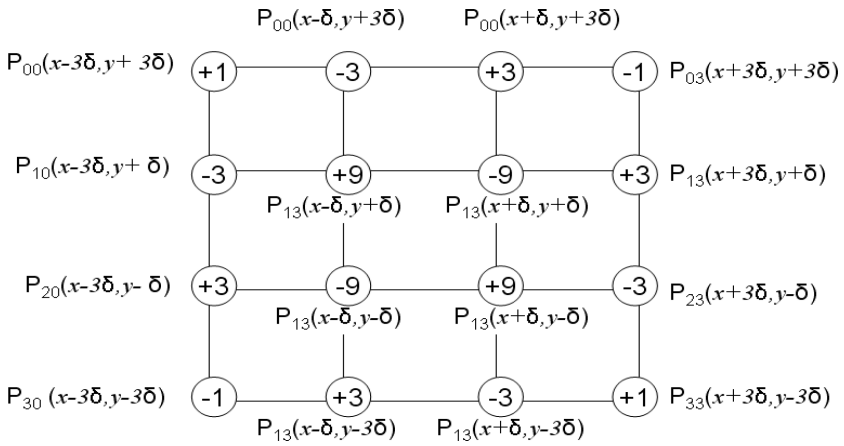


Fig. 4.

orientation specified by a vector $\mathbf{v} = (\alpha, \beta)$ in $\mathcal{R}^2$, where $\alpha \geq 0$, is defined as a bivariate function in the following form. We have

$$\mathbb{P}^{(n)}_{\alpha,\beta,\delta}(x, y; \mathbf{P}_0) = \begin{cases} 0, & \alpha = 0; \\ H_n\left(\frac{1}{\delta}(x_0 - x)\right) H_n\left(\frac{1}{\delta}(y_0 - y)\right), & \alpha > 0, \beta = 0; \\ \Omega^{(n)}_{\alpha,\beta,\delta}(x - x_0, y - y_0), & \alpha > 0, \beta > 0; \\ \Omega^{(n)}_{\alpha,|\beta|,\delta}(-(x - x_0), y - y_0), & \alpha > 0, \beta < 0. \end{cases} \tag{13}$$

With the concept of implicit vertex introduced before, the concept of implicit edge can be defined as the difference of two implicit vertices.

Definition 3.2 *(Implicit Edge)*. Let $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$ be two points in $\mathcal{R}^2$, and let α, β be defined as

$$\alpha = \begin{cases} 0, & x_1 = x_0; \\ 1, & \text{otherwise.} \end{cases}, \qquad \beta = \begin{cases} 1, & x_1 = x_0; \\ \frac{y_1 - y_0}{x_1 - x_0}, & \text{otherwise.} \end{cases}$$

Then, the order n implicit edge corresponding to points $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$ is defined as the difference between the two implicit vertices corresponding to positions $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$.

$$\mathbb{L}^{(n)}_{\delta}(x, y; \mathbf{P}_0, \mathbf{P}_1) = \begin{cases} \mathbb{P}^{(n)}_{\alpha,\beta,\delta}(x, y; \mathbf{P}_1) - \mathbb{P}^{(n)}_{\alpha,\beta,\delta}(x, y; \mathbf{P}_0), \\ \qquad y_1 > y_0 \ \ or \ \ (y_1 = y_0, x_1 > x_0); \\ \mathbb{P}^{(n)}_{\alpha,\beta,\delta}(x, y; \mathbf{P}_0) - \mathbb{P}^{(n)}_{\alpha,\beta,\delta}(x, y; \mathbf{P}_1), \\ \qquad y_1 < y_0 \ \ or \ \ (y_1 = y_0, x_1 < x_0). \end{cases} \tag{14}$$

Now we define the concept of implicit polygon.

Definition 3.3 (*Implicit Polygon*). Let $\mathbf{P}_0(x_0, y_0), \mathbf{P}_1(x_1, y_1), \ldots, \mathbf{P}_m(x_m, y_m)$ be the $m + 1$ vertices of a polygon specified in counter-clockwise order. Then the order n implicit polygon defined by the $m + 1$ points is an implicit function defined in the form

$$\mathbb{Q}^{(n)}_{\delta}(x, y; \mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_m) = \sum_{k=0}^{m} sign(x_k - x_{k+1}) \ \mathbb{L}^{(n)}_{\delta}(x, y; \mathbf{P}_k, \mathbf{P}_{k+1}), \tag{15}$$

where $\mathbf{P}_{m+1} = \mathbf{P}_0$ and $\mathbb{L}^{(n)}_{\delta}(x, y; \mathbf{P}_i, \mathbf{P}_j)$ is the implicit edge defined in Definition 3.2.

It can be shown that for a given polygon $\Delta \subset \mathcal{R}^2$, the implicit polygon built according to Definition 3.3 is identical to the spline basis function built upon the polygon using the iterative integration process described in (6).

THEOREM 3.4. *For each polygon, we have*

$$B^{(n)}_{\Delta,\delta}(x, y) = \mathbb{Q}^{(n)}_{\delta}(x, y; \mathbf{V}(\Delta)), \tag{16}$$

where $\mathbf{V}(\Delta)$ *represents the vertices of polygon* Δ.

As the explicit form of the bivariate function $B^{(n)}_{\Delta,\delta}(x, y)$ is known, it can be evaluated quickly with any specified degree of smoothness and polygon smoothing parameter δ. Figure 1 shows a set of C^2-continuous bivariate functions $B^{(3)}_{\Delta,\delta}(x, y)$ generated from a set of mutually connected polygons of different shapes. Figure 5 displays the shapes of the implicit curve defined by $B^{(3)}_{\Delta,\delta}(x, y) = 0.5$ with different δ values. As can be seen from the figure, the smaller the value of δ, the closer the implicit curve is to the underlying polygon and this is why δ is referred to as polygon smoothing parameter.

The following properties about the bivariate spline basis function $B^{(n)}_{\Delta,\delta}(x, y)$ can be shown immediately from the properties of convolution integration.

(1) *Nonnegativity.* $0 \leq B^{(n)}_{\Delta,\delta}(x, y) \leq 1$.

(2) *Smoothness.* $B^{(n)}_{\Delta,\delta}(x, y)$ has a C^{n-1} continuity.

(3) *Piecewise Polynomial.* $B^{(n)}_{\Delta,\delta}(x, y)$ is piecewise polynomial.

(4) *Local Support.* $B^{(n)}_{\Delta,\delta}(x, y)$ has a finite support if Δ is finite.

(5) *Additivity.* $B^{(n)}_{\Delta,\delta}(x, y)$ is additive; that is, if two polygons Δ_1 and Δ_2 do not intersect or they only intersect at their edges, then

$$B^{(n)}_{\Delta_1 \cup \Delta_2,\delta}(x, y) = B^{(n)}_{\Delta_1,\delta}(x, y) + B^{(n)}_{\Delta_2,\delta}(x, y).$$

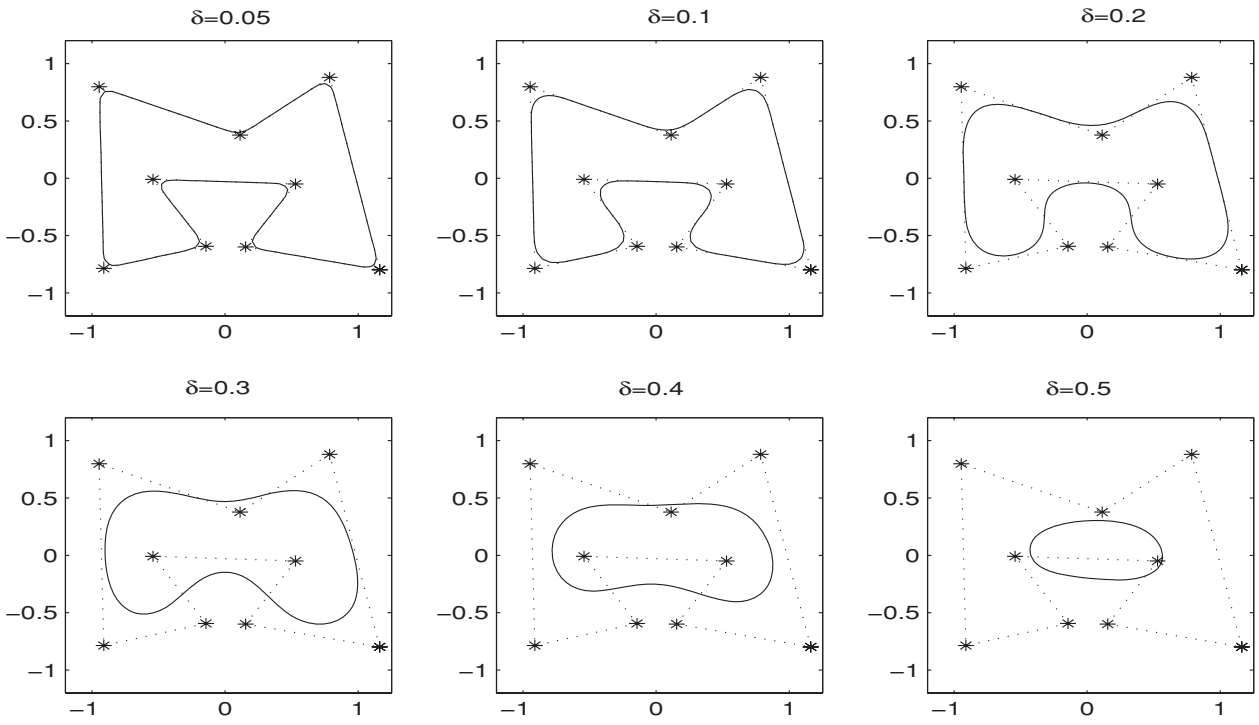


Fig. 5. The shapes of the implicit curve $B^{(n)}_{\Delta,\delta}(x, y) = 0.5$ corresponding to a control polygon specified using nine vertices $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_8$ with different δ values.

(6) *Partition of Unit*. $B^{(n)}_{\Delta,\delta}(x, y)$ takes value in [0, 1] and if

$$\bigcup_k \Delta_k = \mathcal{R}^2, \quad area\left(\Delta_i \bigcap_{i \neq j} \Delta_j\right) = 0,$$

then

$$\sum_k B^{(n)}_{\Delta_k,\delta}(x, y) = 1.$$

It might have been noticed that the spline basis function $B^{(n)}_{\Delta,\delta}(x, y)$ is defined recursively through integral convolution using the characteristic function $\chi_\square(x, y)$ of a square. Since the square is anisotropic, the result of the convolution will depend on the orientation of the polygon edges. As one way to understand in what way the orientation of a polygon edge will influence the result of the convolution, we can consider the convolution of an infinite polygon defined by a single infinite line $\mathcal{L}$ with $\chi_\square(x, y)$ and observe how quickly the values of function $B^{(n)}_{\mathcal{L},\delta}(x, y)$ change from 0 to 1 with the square moves from the outside to the inside of the polygon along a line normal to the line $\mathcal{L}$. Let the distance from position $\mathbf{P}(x, y)$ to the boundary line of the region be $D(\mathbf{P}, \mathcal{L})$. Then it is obvious that, when $\mathcal{L}$ is parallel to the coordinate axes and $D(\mathbf{P}, \mathcal{L}) > n\delta$, $B^{(n)}_{\mathcal{L},\delta}(x, y) = 0$ if $\mathbf{P}(x, y)$ is outside the region and $B^{(n)}_{\mathcal{L},\delta}(x, y) = 1$ if $\mathbf{P}(x, y)$ is inside the region. However, when the line $\mathcal{L}$ is not parallel to the coordinate axes, it takes a relatively longer distance for $B^{(n)}_{\mathcal{L},\delta}(x, y)$ to change its values from 0 to 1. Let the smaller angle between the line $\mathcal{L}$ and the x-axis be θ, $0 \leq \theta \leq \frac{\pi}{2}$. Then, it can be shown that the width of the strip along the line $\mathcal{L}$, in which $B^{(n)}_{\mathbf{L},\delta}(x, y)$ increases from 0 to 1, is

$$d = 2n\delta\ (cos(\theta) + sin(\theta)).$$

When $\theta = \frac{\pi}{4}$, such a distance gets its maximum $2\sqrt{2}n\delta$. Thus, $B^{(n)}_{\Delta,\delta}(x, y)$ will have a much steeper gradient for a polygonal edge whose orientation is close to the x-axis or y-axis. This observation also leads to an explanation on the behavior of $B^{(n)}_{\Delta,\delta}(x, y)$ around a vertex of the corresponding polygon Δ. For a given position $\mathbf{P}(x, y)$, when the convolution involves a vertex of the given polygon, the orientations of the two polygonal edges connected to the vertex will influence the value of $B^{(n)}_{\Delta,\delta}(x, y)$, not just the angle formed by the two edges.

A kind of edge-orientation-independent bivariate spline basis function can be defined by replacing the square region with a circle region in (6). However, we choose not to use a circular region in the convolution for two very simple reasons. First, circle-based convolutions cannot be explicitly expressed as piecewise polynomials, though it may be possible to express them explicitly in some other forms. Second, the slope of a polygonal edge is relatively neutral when the implicit shape is described by the level set $B^{(n)}_{\Delta,\delta}(x, y) = 0.5$, since for each individual polygon edge the contour curve corresponding to the level value 0.5 always corresponds to the center line of the strip on which $B^{(n)}_{\Delta,\delta}(x, y)$ strictly increases from 0 to 1.

An interesting fact about the bivariate spline basis function $B^{(n)}_{\Delta,\delta}(x, y)$ is that when the underlying polygon Δ is simply a rectangle, $B^{(n)}_{\Delta,\delta}(x, y)$ will be the tensor product of univariate spline basis functions defined using smooth unit step function $H_n(x)$. Let $[a, b]$ be an interval with $a \leq b$ and let

$$B^{(n)}_{[a,b],\delta}(x) = H_n\left(\frac{b - x}{\delta}\right) - H_n\left(\frac{a - x}{\delta}\right). \tag{17}$$

From the property of $H_n(x)$, it can be seen directly that $B^{(n)}_{[a,b],\delta}(x)$ is nonnegative piecewise polynomial of degree n with derivatives up

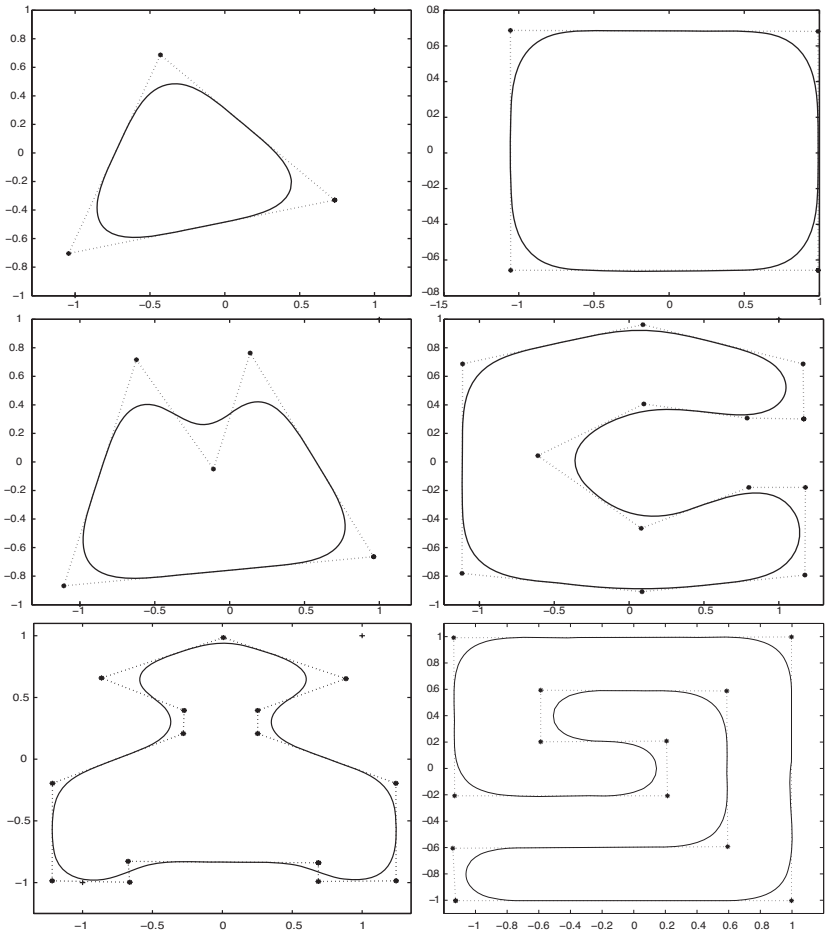


Fig. 6. Freeform implicit curves designed based on some simple polygons. All the underlying implicit functions $B^{(n)}_{\Delta,\delta}(x, y)$ are C^2 continuous.

to $n-1$ order. In addition, if $-\infty = t_0 \le t_1 \le t_2 \le t_3 \cdots \le t_n \le t_{n+1} = \infty$ is a sequence of numbers, then we have

$$\sum_{i=0}^{n} B^{(n)}_{[t_i,t_{i+1}],\delta}(x) = 1, \qquad x \in \mathcal{R}. \tag{18}$$

Figure 7 shows the shapes of C^2-smooth cubic spline basis function $B^{(3)}_{[2,6],\delta}(x)$ defined over interval [2, 6] for δ to be 0.1, 0.5, 0.8, and 1.2.

$B^{(n)}_{[a,b],\delta}(x)$ can be used as an alternative to the conventional B-spline basis functions for parametric curve and surface design. In fact, the spline curves designed using $B^{(n)}_{[a,b],\delta}(x)$ behave very similarly to the conventional B-spline curves in terms of their geometric properties. However, $B^{(n)}_{[a,b],\delta}(x)$ is more flexible and convenient to use than the conventional B-spline basis functions, especially its flexibility in controlling the blending range when it is used to blend locally specified control curves. The application of spline basis functions $B^{(n)}_{[a,b],\delta}(x)$ and $B^{(n)}_{\Delta,\delta}(x, y)$ in modeling parametric shapes will be investigated in a separate paper. Figure 8 shows an example of C^2-smooth cubic spline curves designed using spline basis function $B^{(3)}_{[a,b],\delta}(x)$ with the same set of control points. All these curves differ only by the δ values used in the design. As can be seen from the figure, the smaller the δ value, the more accurately the designed curve approximates the control polygon.

Suppose the lower-left and the top-right vertices of a rectangle Δ are $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$, respectively. Then, it can be shown from the definition of $B^{(n)}_{\Delta,\delta}(x, y)$ that $B^{(n)}_{\Delta,\delta}(x, y)$ is just the tensor product of two univariate spline basis functions $B^{(n)}_{[x_0,x_1],\delta}(x)$ and $B^{(n)}_{[y_0,y_1],\delta}(y)$. In other words, we have

$$B^{(n)}_{[x_0,x_1]\times[y_0,y_1],\delta}(x, y) = B^{(n)}_{[x_0,x_1],\delta}(x) B^{(n)}_{[y_0,y_1],\delta}(y). \tag{19}$$

As pointed out previously, the spline basis functions presented here are completely different from the implicit splines defined using barycentric coordinates [Bajaj and Xu 2001, 1999; Xu et al. 2000a, 2000b]. Firstly, our bivariate spline basis functions are a kind of geometric generalization of univariate spline basis functions in the sense that they possess similar geometric properties to their univariate counterparts, such as nonnegativity and partition of unit. In contrast, the implicit splines defined using barycentric coordinates can only be considered as a kind of formal generalization to the conventional univariate splines. They are referred to as a kind of implicit spline simply because they can formally be expressed in

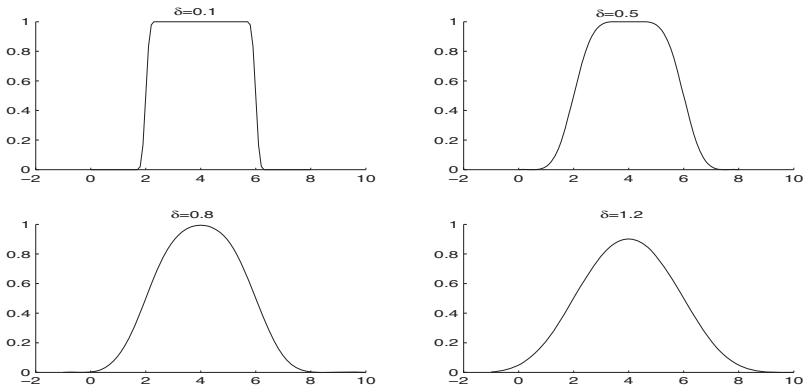


Fig. 7. The shapes of C^2-smooth cubic spline basis function $B^{(3)}_{[2,6],\delta}(x)$ defined over interval [2, 6] with different δ values.

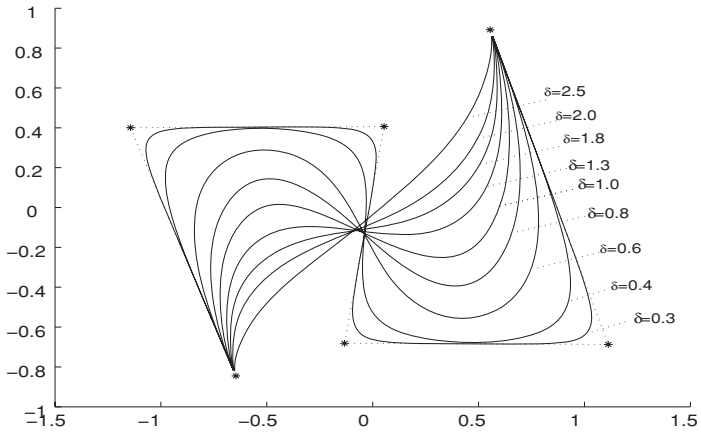


Fig. 8. A set of C^2-smooth cubic spline curves designed using spline basis functions $B^{(3)}_{[a,b],\delta}(x)$ with the same set of control points. All these curves are differed by the δ values used in the design.

the form of tensor product of univariate spline basis functions in terms of barycentric coordinates. However, geometrically, they do not behave in a similar way with the univariate spline basis functions. The second difference between the implicit spline basis functions presented in this article and those defined directly using barycentric coordinates is that our spline functions are directly defined globally in $\mathcal{R}^2$, whereas the latter form is defined locally since, in practice, only those pieces corresponding to the points contained within the corresponding triangles are meaningful. Thirdly, all the level sets of $B^{(n)}_{\Delta,\delta}(x, y)$, when Δ is a finite polygon, are finite connected regions in $\mathcal{R}^2$ if the value of δ is properly chosen, whereas the barycentric coordinates based spline basis functions do not have such property in general. Thus, we can refer to the bivariate spline basis functions $B^{(n)}_{\Delta,\delta}(x, y)$ as a kind of solid spline basis function, and refer to the splines defined using barycentric coordinates as a kind of algebraic patch or segment. Due to all these differences, the two kinds of implicit splines are used in practice in completely different ways.

4. DESIGN IMPLICIT PLANE CURVE USING THE BIVARIATE SPLINE $B^{(N)}_{\Delta,\delta}(X, Y)$

With the proposed bivariate splines, a freeform implicit curve can be generated intuitively by laying out a sequence of control points that specify the features of the required shapes.

4.1 Implicit Curve Corresponding to Simple Control Polygons

For a simple control polygon Δ, where the control polygons do not have any holes inside the polygon and no two polygon edges intersect other than at vertices, the corresponding implicit curve can be defined directly as $B^{(n)}_{\Delta,\delta}(x, y) = 0.5$. From the definition of implicit polygon described in Definition 3.3, to find $B^{(n)}_{\Delta,\delta}(x, y)$, designers need only specify the control points in counterclockwise order and choose a proper degree of smoothness of the required bivariate function, as well as the polygon smooth parameter δ. Figure 6 demonstrates the implicit curves designed in this way based on some simple polygons. Each of these curves is the contour curve of a C^2-continuous bivariate function $B^{(n)}_{\Delta,\delta}(x, y)$ with height level 0.5.

4.2 Implicit Curve Corresponding to Nonsimple Control Polygons

A polygon with holes in it can be considered as a simple polygon by adding some auxiliary polygon edges and the proposed implicit polygon function described in Definition 3.3 can then be applied directly to design implicit curves using this kind of polygon. Figure 9 demonstrates some examples of freeform implicit curves designed using polygons that contain holes. It should be noted that the same

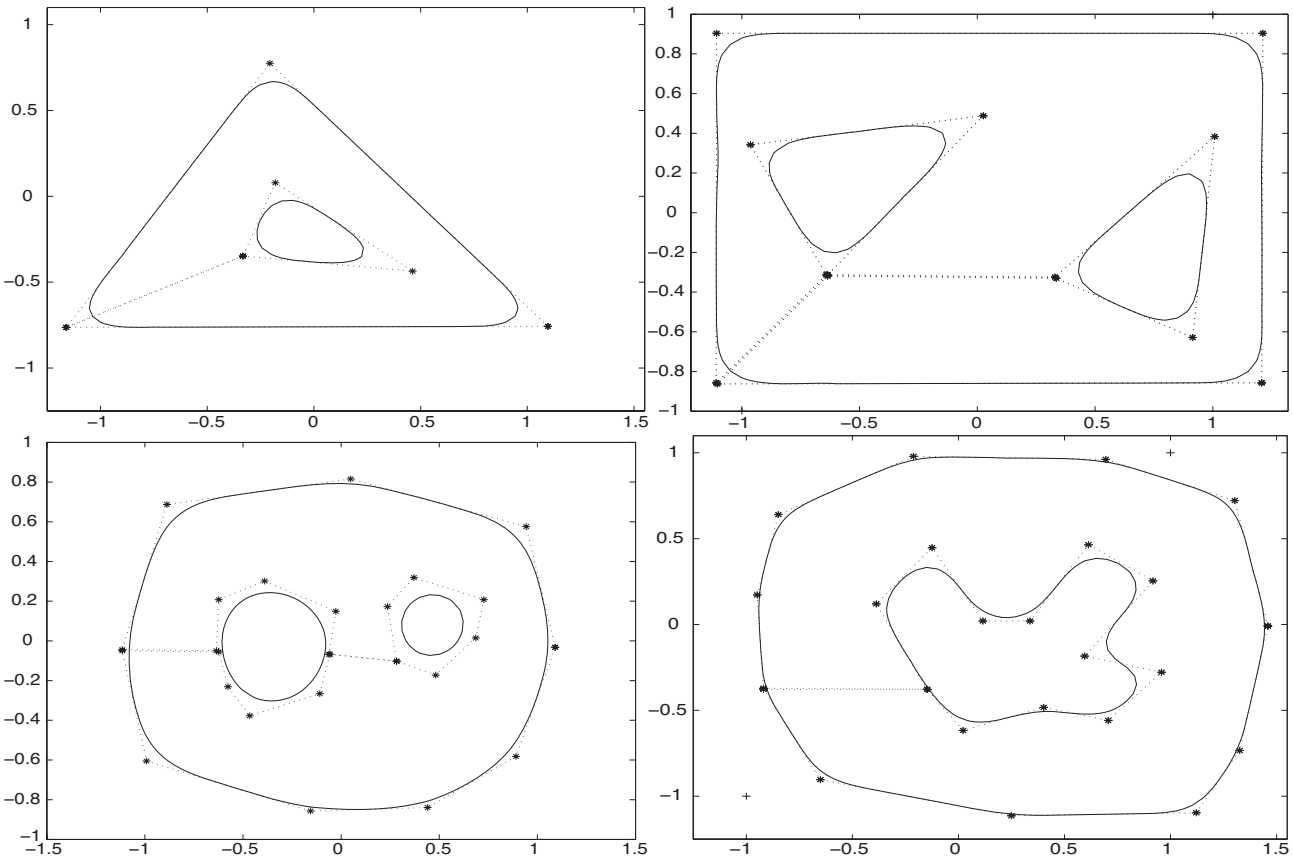


Fig. 9. Design implicit curves using polygons with inside holes as ordinary simple polygons by adding some auxiliary polygon edges.

shape can be obtained following the idea of CSG by describing a nonsimple polygon-based implicit spline as the difference between the implicit spline constructed from the exterior simple polygon and the implicit splines based on the interior simple polygons corresponding to the holes.

When there are intersected polygon edges, the proposed freeform implicit plane curve design technique can also be applied. However, it will ignore those subpolygons whose vertices are in clockwise order.

4.3 Constructive Implicit Curve Design

One typical feature of a spline-based parametric curve is that it is expressed as a weighted sum of local control shapes, where the weight associated to each local control shape is a spline basis function. A similar idea can be introduced to design implicit curves. We first break the required shape into some much simpler shapes, such that each of these simple shapes can be designed easily using a simple implicit shape. Suppose $F_k(x, y), k = 0, 1, \ldots, m$ are the locally specified implicit functions with their main features defined on polygons $\Delta_k, k = 0, 1, \ldots, m$, respectively. Then these $m + 1$ implicit functions can be combined as a weighted sum of the $m + 1$ implicit polygons, as described in

$$F(x, y) = \sum_{k=0}^{m} F_k(x, y) B^{(n)}_{\Delta_k,\delta}(x, y), \tag{20}$$

where $B^{(n)}_{\Delta_k,\delta}(x, y)$ is the implicit spline basis function constructed from polygon Δ_k.

Representing an implicit shape as a sum of weighted implicit spline basis functions offers an alternative to the conventional CSG technique for combining implicit shapes of different kinds. Figure 10(a) shows an example of designing an implicit shape in this way, where the overall implicit function is a weighted sum of a circle, two implicit lines, and several sine curves. One obvious advantage of this way of combining a set of implicit shapes is that the blending range can be controlled easily by selecting a proper value for the polygon smoothing parameter δ. This blending feature becomes more important when parts of a designed curve come from some actual geometric objects, such as the implicit curves obtained by fitting real datasets. Obviously, the shapes of these fitted curves should be kept unchanged as much as possible. Figure 10(b) shows an implicit shape designed in this way, where the top part of the curve is obtained by fitting real data.

Note that when the value of polygon smoothing parameter is small, the shapes of implicit polygons may look like mesas. These kinds of bivariate spline basis functions have such a property that they take value 1 for points well within their supports and 0 for those well outside their supports. Consider a function $B(x, y)F(x, y)$ for a flat-topped spline basis function $B(x, y)$ and an arbitrary function $F(x, y)$. The new function $B(x, y)F(x, y)$ will have exactly the same shape as $F(x, y)$ when the point $\mathbf{P}(x, y)$ is well within the support of $B(x, y)$, while the shape of $F(x, y)$ well outside the

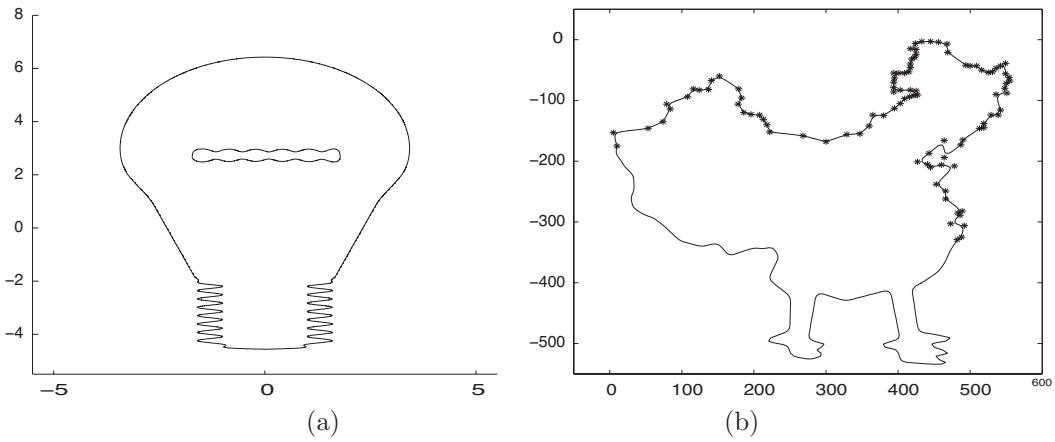


Fig. 10. Implicit curves designed as a weighed sum of a set of implicit shapes of different types.

support of $B(x, y)$ is removed as $B(x, y)F(x, y)$ will be nearly zero. Therefore, functions $B(x, y)$ can be referred to as a kind of shape-preserving spline basis function. This partial shape-preserving feature of the introduced spline is particularly useful in fitting a large dataset with an implicit curve [Li et al. 2006]. We can first break a large dataset into some small datasets such that the shape corresponding to each of these subdatasets is simple and can be easily fitted. Due to the features of partial shape-preserving and the property of partition of unit of binary spline functions $\{B^{(n)}_{\Delta_k,\delta}(x, y)\}$, these individually fitted implicit functions can then be combined in form (20).

Just as a parametric B-spline curve may generate nonsmooth curves if control points are not carefully laid out, the proposed curve design technique may also generate nonsmooth implicit curves if control points are not placed properly. Figure 13 illustrates how the problem may occur and how to avoid this. As can be seen from the figure, if the underlying control polygon has a narrow strip, then it can happen that the constructed function may take 0.5 at all points on the central axis of the strip. As a result, part of the contour curve will be a straight line, which corresponds to the ridge of the constructed function. This problem can be solved by either individually designing each particularly thin part or using a much smaller δ value. Due to the easy blending nature of implicit shapes, a complex implicit curve can be considered as the combination of several simple implicit curves designed using different polygon smoothing parameter values. Figure 13(b) demonstrates an implicit curve designed in this way.

It is commonly known that parametric B-spline curves exhibit various good geometric properties, such as convex hull, linear precision, variation diminishing, affine invariant. Will the implicit curves designed using the ideas described in Sections 4.1 and 4.2 possess these properties? To answer this question, we have first to understand the difference between implicitly represented curves and parametric curves. As an implicit curve is defined as the boundary of a level set of an implicit function for a given isovalue, its shape depends on the choice of isovalue. As the implicit function $B^{(n)}_{\Delta,\delta}(x, y)$ built from a control polygon is always a mapping from $\mathcal{R}^2$ to interval [0, 1], its isovalue can vary from 0 to 1. In general, the size of the level set $\{(x, y) : B^{(n)}_{\Delta,\delta}(x, y) \geq \alpha\}$ corresponding to isovalue α will increase with the decrease of the isovalue α. Obviously, the level set is the entire plane when the isovalue is 0. Thus, it is meaningless to talk about the curve design property of convex hull in general. However, it can be shown directly that an implicit curve designed using the implicit polygon corresponding to isovalue 0.5 does have the convex hull property. As the proposed implicit spline is a kind of solid spline, it is not appropriate to discuss those curve design properties concerning open curve segments, such as the property of linear precision. Obviously, when a polygon Δ degenerates into a single line or line segment, the corresponding implicit polygon $B^{(n)}_{\Delta,\delta}(x, y)$ will be zero everywhere. As for the property of variation diminishing, whether an implicit polygon constructed from a given polygon has this property depends on the choice of parameter δ used in (6). In general, this property is not possessed by curves designed using implicit polygons. As can be seen from Figure 11, this property is corrupted when the internal part of a given design polygon is relatively too narrow with respect to the δ value used in (6). However, the implicit curves designed using implicit polygons will perserve the curve design property of variation diminishing, as long as the value of the polygon smoothing parameter δ used in $B^{(n)}_{\Delta,\delta}(x, y)$ is sufficiently small. It can be shown that when the δ value is set properly such that the value of $B^{(n)}_{\Delta,\delta}(x, y)$ is greater or equal to 0.5 for any point in the narrowest interior regions of the design polygon, the property of variation diminishing will still be held by the proposed implicit curve design technique. As an illustration, an implicit curve design using a control polygon with a relatively complex branching structure is presented in Figure 12.

As have been pointed out previously, the proposed implicit curve design technique using an implicit polygon is polygon orientation dependent. Thus, it is obvious that it does not possess the property of affine invariant. An affine invariant implicit polygon can be developed by replacing the square region with a circle region in the convolution defined in (6). But we choose not to use a circular region in (6) due to the two obvious reasons described previously. Nevertheless, we want to point out here that the property of rotational transformation invariant is an important issue in terms of computational efficiency in manipulating and displaying the designed shape, but it is not a necessary requirement with regard to the efficiency and effectiveness in designing a geometric shape.

In this section, we demonstrated how 2D implicit curves can be designed using the proposed implicit spline basis functions, especially how to design freeform implicit curves as an implicit polygon by specifying a sequence of control points. Compared with the techniques presented in Bajaj and Xu [2001, 1999], the implicit curve

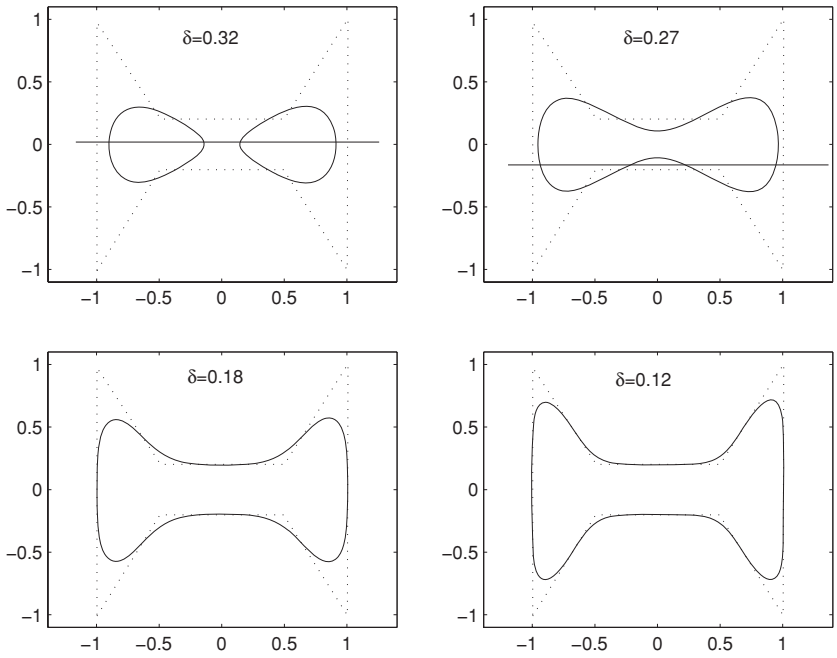


Fig. 11. When the δ value used in (6) for constructing the implicit function is too big, the property of variation diminishing will not be kept by implicit curve designed using implicit polygon, as can be seen from the two figures shown in the first row, where a line intersects the implicit curves designed with $\delta = 0.32$ and $\delta = 0.27$ four times, but only intersects control polygon twice. However, as can be seen from the figures in the second row, when appropriate smaller δ values are used, this design property will be preserved by the designed implicit curves.

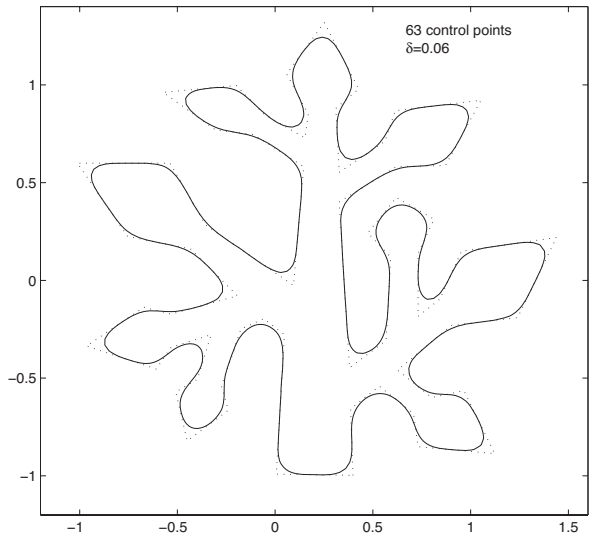


Fig. 12. A C^2-smooth implicit curve $B^{(3)}_{\Delta,\delta}(x, y) = 0.5$ designed using 63 control points. In general, implicit curves designed using an implicit polygons will have the curve design property of variation diminishing, as long as the δ value used in $B^{(n)}_{\Delta,\delta}(x, y)$ is sufficiently small.

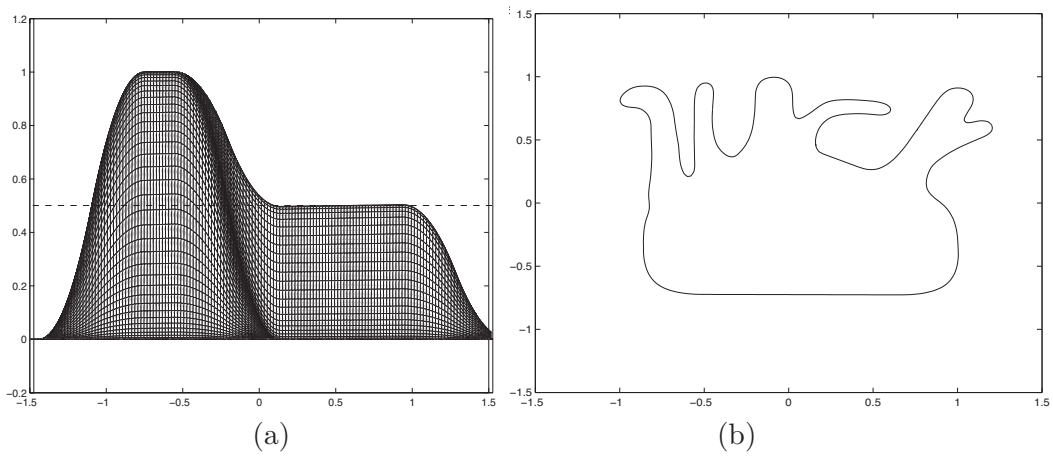


Fig. 13. Parts of some bivariate spline basis functions corresponding to certain control polygons may exhibit flat ridges, which may produce nonsmoothing implicit curves at certain contour values for a certain polygon smoothing parameter δ. This problem can be solved by splitting the required shape into several simpler control polygons, which allows shape designers to design an implicit shape using different values of polygon smoothing parameter δ: (a) a bivariate spline basis function having a flat ridge; (b) a combined implicit curve using several implicit spline curves designed with different δ values.

modeling techniques presented in this article are a kind of solid spline modeling technique which produces globally defined implicit functions directly in shape space. The main strengths of the proposed technique are its intuitiveness, simplicity, and computational efficiency. As has been seen, the curves designed using shape control polygons are described directly using implicit vertices, which are all calculated using the same function described in (8) and generalized smooth unit step functions described in (12). It can be seen from their definitions that all these functions can be computed efficiently due to their simple mathematical forms. Consequently, the proposed freeform implicit curve design scheme is particularly simple to implement. For instance, to generate a C^2-continuous implicit curve from a control polygon defined by five control points $\mathbf{V}_0$, $\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$, $\mathbf{V}_4$, it is only necessary to write out five implicit line segments corresponding to the five edges of the polygon using bivariate function $\Omega^{(3)}_{\alpha,\beta,\delta}(x, y)$, and the implicit function defined by the polygon is simply the signed sum of the five implicit edges, depending on whether an edge is a top one or a lower one. Furthermore, unlike the regular algebraic curve-segments-based techniques, where delicate considerations have to be given to combine two pieces of curves smoothly, it is not an issue at all for the freeform implicit curves design technique proposed in this article. The choice of smoothness of the designed curves using the proposed technique is simply a matter of choosing the degree of smoothness of the function $\Omega^{(n)}_{\alpha,\beta}(x, y)$ and the smooth unit step functions $H_n(x)$. However, as pointed out previously, our implicit splines are a kind of solid spline, which is not an appropriate technique for designing and describing a single curve segment locally.

5. DESIGN FREEFORM IMPLICIT SURFACES USING BIVARIATE SPLINES

The proposed piecewise polynomial binary spline basis functions can also be used to design freeform implicit surfaces in various ways. In this section we give a brief discussion on some of the implicit surface design techniques using 2D implicit curves, including implicit spline surface, implicit surface of revolution, and extrusion. As can be seen in the following: a rich set of implicit surfaces can be designed using implicit spline curves.

5.1 Design Freeform Implicit Surfaces Using Implicit Control Curves

3D parametric spline surfaces can be considered as the result of blending a set of shape control curves using spline basis functions. Similarly, freeform implicit surfaces can also be designed in this way as a weighted sum of a set of 2D implicit control curves, where each of these implicit curves specifies a cross-section profile of the designed shape. One simple and direct method is to slice the desired shape along a coordinate axis, say the z-axis, and describe the overall geometric shape of the object as a set of 2D contour curves distributed along the z-axis. Then the required shape can be described implicitly as

$$F(x, y, z) = \sum_{k=0}^{m} C_k(x, y)B_k(z) = 0, \tag{21}$$

where $\{B_k(z)\}_{k=0}^m$ are spline basis functions. The shapes presented in Figure 14 are generated in this way using C^2-smooth spline basis functions $B^{(3)}_{[a_k,b_k],\delta}(z)$, $k = 0, 1, 2, \ldots, m$, defined in (17).

In the figure, the shapes displayed in the first two rows are designed using just one cross-section control polygon. Those displayed in the third and fourth rows are designed with two and three cross-section control polygons, respectively. In (21), we can also use ordinary B-spline basis functions to combine implicit control curves corresponding to different cross-section profiles. The implicit shapes showed in Figure 15 are designed using fourth-order B-spline basis functions. To achieve various degrees of smoothness for the designed shapes, different values of the polygon smoothing parameter are used.

Implicit surfaces represented in this form can also be used to smooth real data. The implicit shapes shown in Figure 16 are created from data sampled from an actual human body.

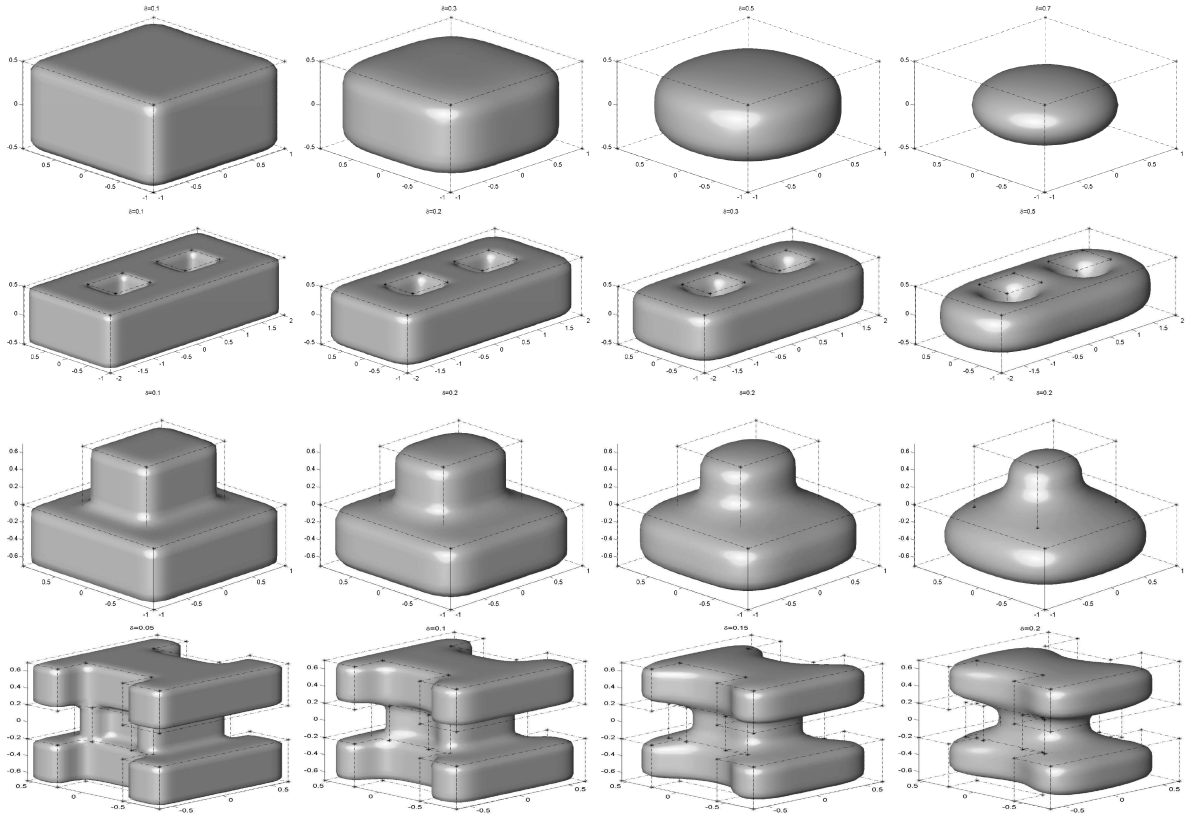


Fig. 14. Freeform implicit surfaces designed using spline basis functions $H_{[a,b],\delta}(z)$ defined in (17).

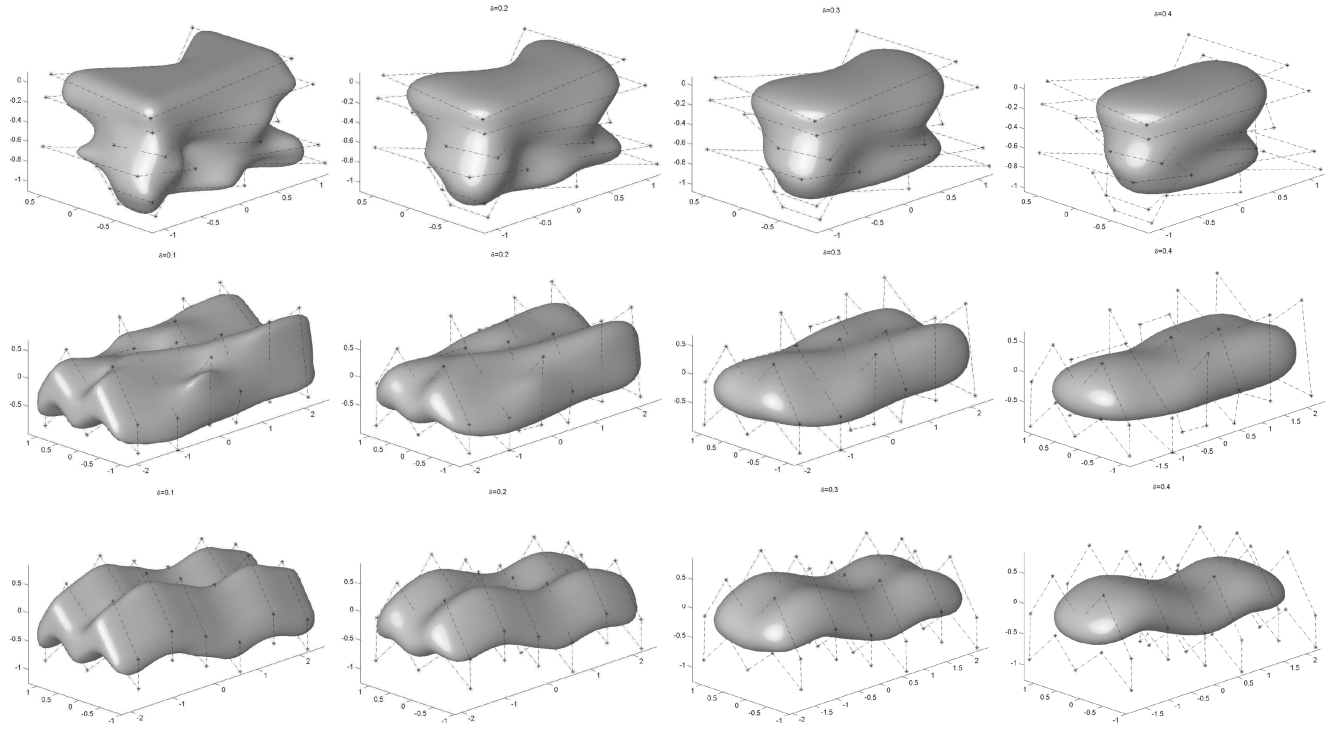


Fig. 15. Freeform implicit surfaces designed using B-spline basis functions of order four.

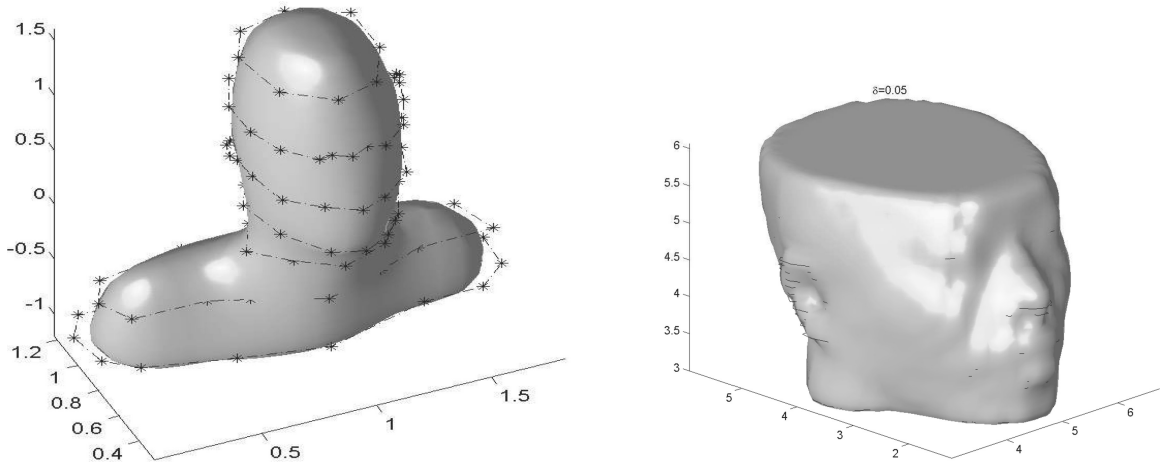


Fig. 16. Implicit surfaces built with data sampled from an actual human body.

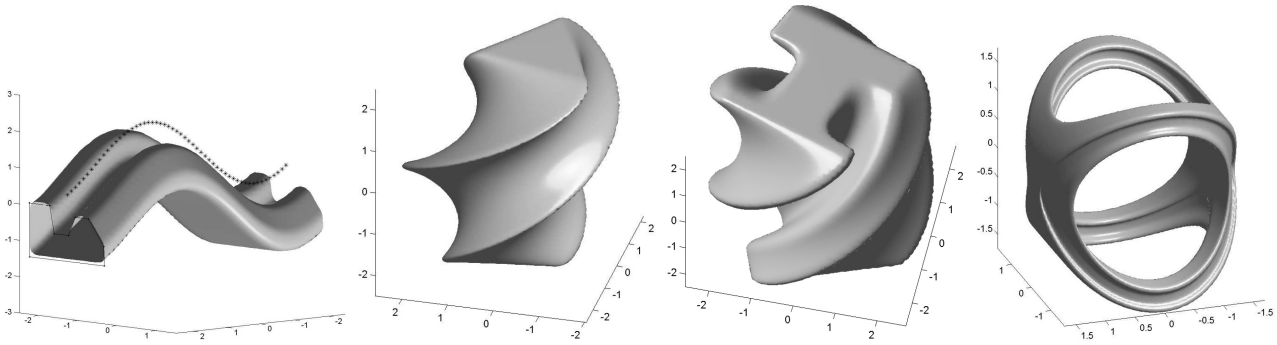


Fig. 17. Implicit extruded surfaces obtained using implicit polygonal profiles.

5.2 Extruded Implicit Surfaces

The shapes shown in Figures 14 and 15 can be regarded as the results of extruding a 2D implicit curve along a coordinate axis. This idea can be generalized by extruding an implicitly specified curve along an arbitrary curve, for example, using the technique proposed in Barthe et al. [2003] and Schmidt and Wyvill [2005]. In general, an extrusion of an implicit profile curve along an extrusion path corresponds to a mapping from $\mathcal{R}^3$ to $\mathcal{R}^2$: $X = X(x, y, z), Y = Y(x, y, z)$. This view of implicit extrusion directly leads to a simple extrusion method. Suppose an extrusion path is represented implicitly as the intersection of two implicit surfaces $F_1(x, y, z) = 0$ and $F_2(x, y, z) = 0$ and suppose the cross-section profile curve is defined as an implicit function $C(x, y) = 0$. Then an extruded implicit surface can be directly given by

$$C(F_1(x, y, z), F_2(x, y, z)) = 0,$$

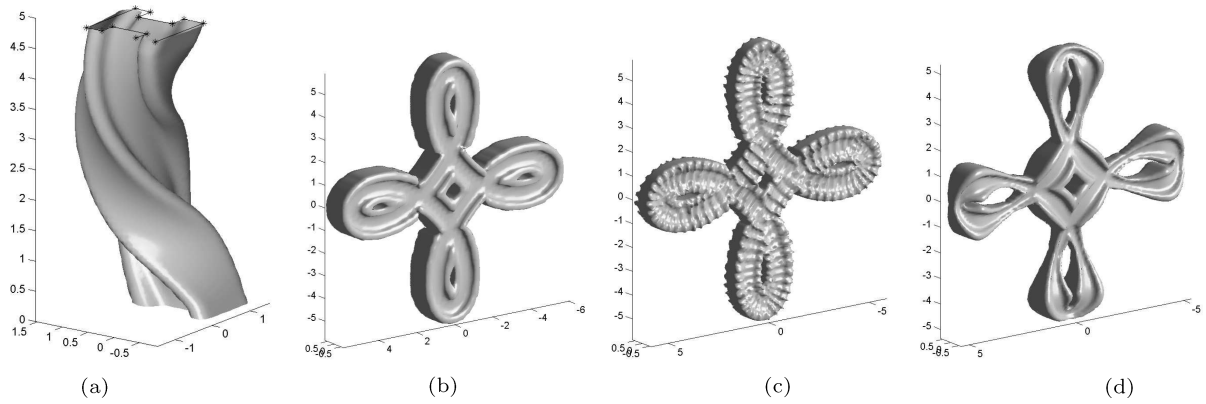
where the signed function values of the functions $F_1(x, y, z)$ and $F_2(x, y, z)$ have been used to approximate, respectively, the distances from point $\mathbf{P}(x, y, z)$ on the normal plane to the rectifying plane and the osculating plane of the Frenet-Serret frame of the extrusion path. A better way to extrude along an implicitly defined extrusion path can be defined using Taubin's approximate distance [Taubin 1991]. Specifically, we can define the extruded implicit surface as

$$C\left(\frac{F_1(x, y, z)}{|\nabla F_1(x, y, z)|}, \frac{F_2(x, y, z)}{|\nabla F_2(x, y, z)|}\right) = 0.$$

The implicit shapes shown in Figure 17 are generated in this way. However, the implicit surfaces generated thusly will in general be distorted in some way. This is because in general the surfaces $X = F_1(x, y, z)$ and $Y = F_2(x, y, z)$ used to define the mapping from $\mathcal{R}^3$ to $\mathcal{R}^2$ are not necessarily flat on one the hand and are not mutually orthogonal on the other. To produce a profile shape-preserving extruded implicit surface, a precise mapping from $\mathcal{R}^3$ to $\mathcal{R}^2$ needs to be established. In practice, such a mapping can be numerically represented as two signed distance maps corresponding to the rectifying plane and the osculating plane of the extrusion path. The implicit shapes shown in Figure 18(a) and (b) are created in this way. The mapping from $\mathcal{R}^3$ to $\mathcal{R}^2$ can also be followed by certain transformations such that the cross-section profile can be changed in different ways to meet the requirements of a desired shape. The implicit shapes shown in Figure18(c) and (d) are generated following this idea.

5.3 Implicit Surfaces of Revolution

Surface of revolution is an important kind of geometric shape. Similar to a parametrically represented surface of revolution, a surface



(a) (b) (c) (d)

Fig. 18. Implicit extruded surfaces generated using the signed distance maps of the rectifying plane and osculating plane of the extrusion path.

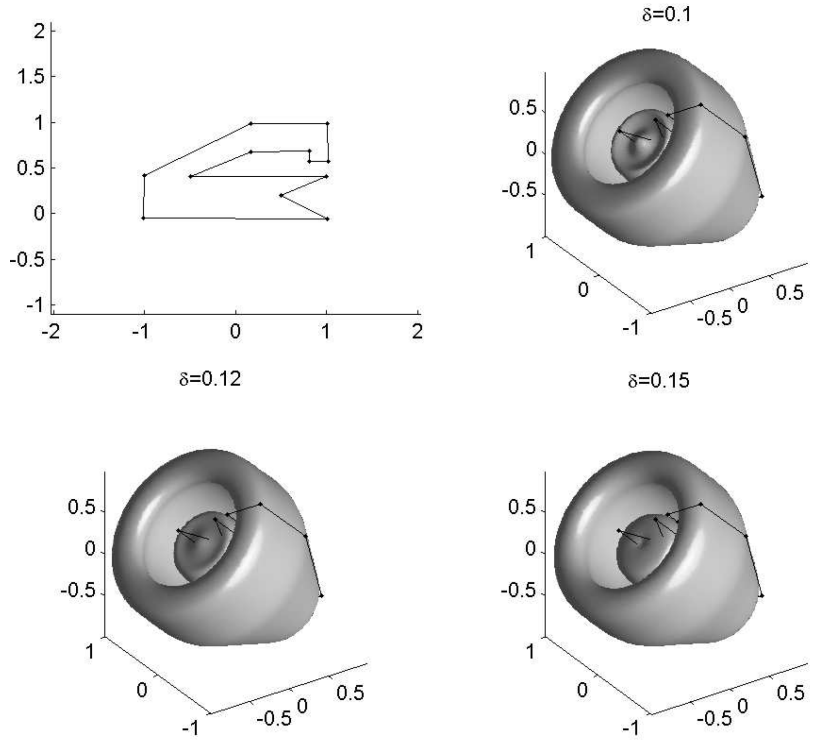


Fig. 19. Implicit surfaces of revolution obtained by rotating implicit polygonal profiles.

of revolution in implicit form can also be generated easily. It can be considered as a special case of extruding an implicitly defined 2D profile curve along a circle. Suppose the profile of a rotated surface is represented as an implicit curve $F(x, y) = 0$. Then the surface of revolution generated by rotating a profile curve about the y-axis can be described by $F(G(x^2 + z^2), y) = 0$, and the surface of revolution generated by rotating the profile curve about the x-axis can be described by $F(x, G(y^2 + z^2)) = 0$, where $G(t)$ is a certain real function. Figure 19 demonstrates an example of implicit surface of revolution obtained by rotating an implicit spline curve.

6. SUMMARY

We have presented in this article a bivariate spline technique for modeling freeform implicit curves from a given control polygon. Unlike some other binary spline basis functions used in practice, the explicit analytical expression of the proposed bivariate splines has been developed, which makes it possible to evaluate the value of this type of implicit spline basis functions quickly and accurately. Some design examples have been given to demonstrate the capability and flexibility of this newly developed spline technique for designing

implicit curves. Some examples in using this type of implicit curve for freeform implicit surface design have also been presented. As can be seen, a rich set of freeform implicit surfaces can be generated quickly by manipulating implicitly designed curves.

APPENDIX

PROOF OF THEOREM 3.4. The proof of Theorem 3.4 is straightforward once we view the value of the implicit edge $\mathbb{L}_{\delta}^{(n)}(x, y; \mathbf{P}_0, \mathbf{P}_1)$ at a point $\mathbf{P}(x, y)$ as the volume contained within all the following three surfaces: the $n-1$ order implicit edge $z = \mathbb{L}_{\delta}^{(n-1)}(x, y; \mathbf{P}_0, \mathbf{P}_1)$, the xy-plane $z = 0$, and function $z = \frac{1}{4\delta^2}\chi_{\square}(x, y)$, where $\chi_{\square}(x, y)$ is the characteristic function corresponding to the square $\square$ centered at point $\mathbf{P}(x, y)$ with base area of $4\delta^2$. Now consider a 2D polygon Δ. From the definition given in (6), the value of $B_{\Delta,\delta}^{(n)}(x, y)$ at a point $\mathbf{P}(x, y)$ can also be viewed as the volume contained within three surfaces: $z = B_{\Delta,\delta}^{(n-1)}(x, y)$, the xy-plane $z = 0$, and the surface defined by $z = \frac{1}{4\delta^2}\chi_{\square}(x, y)$. If we can show that that the implicit edge $\mathbb{L}_{\delta}^{(n)}(x, y; \mathbf{P}_0(x_0, y_0), \mathbf{P}_1(x_1, y_1))$ is identical to the binary spline function $B_{\sqcap,\delta}^{(n)}(x, y)$, where $\sqcap$ is the infinite polygon defined with vertices $\mathbf{P}_0$, $\mathbf{P}_1$, and vertices $\mathbf{Q}_0(x_0, -\infty)$ and $\mathbf{Q}_0(x_1, -\infty)$ at the infinity, then $B_{\Delta,\delta}^{(n)}(x, y)$ can immediately be expressed as the difference between the sum of all the implicit edges corresponding to the upper edges of the polygon Δ and the sum of all those implicit edges constructed from the lower edges of the polygon. Suppose the vertices of a given polygon are specified in a counterclockwise way as $\mathbf{P}_1(x_1, y_1), \ldots, \mathbf{P}_m(x_m, y_m)$, then edges of the polygon can be grouped according to the sign of $x_k - x_{k+1}$. An edge is an upper edge when $x_k - x_{k+1} > 0$, and an lower edge when $x_k - x_{k+1} < 0$. When $x_k - x_{k+1} = 0$, the polygon edge is vertical, it can be seen immediately that $B_{\sqcap,\delta}^{(n)}(x, y) = 0$. There is no need to consider it in (15).

Now to show Theorem 3.4, we need only show that an implicit edge corresponding to two points $\mathbf{P}_0(x_0, y_0), \mathbf{P}_1(x_1, y_1))$ defined according to (14) given in Definition 3.2 can actually be expressed as $B_{\sqcap,\delta}^{(n)}(x, y)$ for an infinite polygon corresponding to the edge of the polygon Δ.

Let us first consider a simple infinite angle polygon $\angle$ defined using two rays from the coordinate origin, with one of the ray being vertical along the negative direction of the y-axis and the other one along a line against the direction defined by a vector $\mathbf{v}(\alpha, \beta), \alpha \geq 0$, $\beta \geq 0$ and $\alpha + \beta > 0$. Let

$$\begin{aligned}\Lambda_{\angle}^{(0)}(x, y) &= \chi_{\angle}(x, y) = \begin{cases} 1, & (x, y) \in \angle; \\ 0, & (x, y) \notin \angle. \end{cases}, \\ \Lambda_{\angle}^{(n)}(x, y) &= \int\int_{\mathcal{R}^2} \Lambda_{\angle}^{(n-1)}(s, t)\chi_{\lrcorner}(s - x, t - y)dsdt, \\ & \qquad (n > 0), \quad (22)\end{aligned}$$

where $\lrcorner$ is an infinite polygon defined by a horizontal ray against the direction of x-axis and a vertical ray along the y-axis, both of which start from the position $\mathbf{P}(x, y)$.

It can be shown using the principle of mathematical induction that, when , $\alpha > 0$, $\beta > 0$, $\Lambda_{\angle,\delta}^{(n)}(x, y)$ can be expressed explicitly in the form given in (7). We have

$$\begin{aligned}\Lambda_{\angle,\delta}^{(n)}(x, y) &= A_{\alpha,\beta}^{(n)}(x, y) \quad (23) \\ &= \begin{cases} 0, & \\ & y \geq \min\left\{\frac{\beta}{\alpha}x, 0\right\}; \\ \frac{1}{(2n)!\alpha^n\beta^n}(\beta x - \alpha y)^{2n}, & \\ & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x \leq 0; \\ \sum_{k=1}^{n} \frac{(-1)^{n+k}\alpha^k}{(n-k)!(n+k)!\beta^k}x^{n-k}y^{n+k}, & \\ & y < \min\left\{\frac{\beta}{\alpha}x, 0\right\}, x > 0. \end{cases}\end{aligned}$$

Let $\square \subset \mathcal{R}^2$ be a square of size $2\delta \times 2\delta$ centered at the coordinate origin with $\delta > 0$. Then, for any point $\mathbf{P}(x, y)$, if it is not on the boundary of $\square$, we have

$$\begin{aligned}\chi_{\square}(x, y) &= \chi_{\lrcorner}(x + \delta, y - \delta) - \chi_{\lrcorner}(x - \delta, y - \delta) \quad (24) \\ &\quad -\chi_{\lrcorner}(x + \delta, y + \delta) + \chi_{\lrcorner}(x - \delta, y + \delta).\end{aligned}$$

Thus, we have

$$\begin{aligned}B_{\angle,\delta}^{(1)}(x, y) &= \frac{1}{4\delta^2}\int\int_{\mathcal{R}^2} B_{\angle,\delta}^{(0)}(s, t)\chi_{\square}(s - x, t - y)dsdt \\ &= \frac{1}{4\delta^2}\int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(0)}(s, t)\chi_{\square}(s - x, t - y)dsdt \\ &= \frac{1}{4\delta^2}(A_{00} - A_{01} - A_{10} + A_{11}),\end{aligned}$$

where

$$\begin{aligned}A_{00} &= \int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(0)}(s, t)\chi_{\lrcorner}(s - (x + \delta), t - (y - \delta))dsdt \\ &= \Lambda_{\angle,\delta}^{(1)}(x + \delta, y - \delta).\end{aligned}$$

$$\begin{aligned}A_{01} &= \int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(0)}(s, t)\chi_{\lrcorner}(s - (x - \delta), t - (y - \delta))dsdt \\ &= \Lambda_{\angle,\delta}^{(1)}(x - \delta, y - \delta).\end{aligned}$$

$$\begin{aligned}A_{10} &= \int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(0)}(s, t)\chi_{\lrcorner}(s - (x + \delta), t - (y + \delta))dsdt \\ &= \Lambda_{\angle,\delta}^{(1)}(x + \delta, y + \delta).\end{aligned}$$

$$\begin{aligned}A_{11} &= \int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(0)}(s, t)\chi_{\lrcorner}(s - (x - \delta), t - (y + \delta))dsdt \\ &= \Lambda_{\angle,\delta}^{(1)}(x - \delta, y + \delta).\end{aligned}$$

Therefore,

$$\begin{aligned}B_{\angle,\delta}^{(1)}(x, y) &= \frac{1}{4\delta^2}[\Lambda_{\angle,\delta}^{(1)}(x + \delta, y - \delta) - \Lambda_{\angle,\delta}^{(1)}(x - \delta, y - \delta) \\ &\quad -\Lambda_{\angle,\delta}^{(1)}(x + \delta, y + \delta) + \Lambda_{\angle,\delta}^{(1)}(x - \delta, y + \delta)].\end{aligned}$$

Similarly,

$$\begin{aligned}B_{\angle,\delta}^{(2)}(x, y) &= \frac{1}{4\delta^2}\int\int_{\mathcal{R}^2} B_{\angle,\delta}^{(1)}(s, t)\chi_{\square}(s - x, t - y)dsdt \\ &= \frac{1}{(4\delta^2)^2}\int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(1)}(s + \delta, t - \delta)\chi_{\square}(s - x, t - y)dsdt \\ &\quad -\frac{1}{(4\delta^2)^2}\int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(1)}(s - \delta, t - \delta)\chi_{\square}(s - x, t - y)dsdt \\ &\quad -\frac{1}{(4\delta^2)^2}\int\int_{\mathcal{R}^2} \Lambda_{\angle,\delta}^{(1)}(s + \delta, t + \delta)\chi_{\square}(s - x, t - y)dsdt\end{aligned}$$

$$+\frac{1}{(4\delta^2)^2}\int\int_{\mathcal{R}^2}\Lambda^{(1)}_{\angle,\delta}(s-\delta,t+\delta)\chi_{\square}(s-x,t-y)dsdt. \quad (25)$$

As each term on the righthand side of the Eq. (25) can again be expressed using $\chi_{\perp}(x, y)$, it is not difficult to see that

$$\begin{aligned}B^{(2)}_{\angle,\delta}(x,y) &= \frac{1}{(4\delta^2)^2}[\Lambda^{(2)}_{\angle,\delta}(x-2\delta,y-2\delta)-2\Lambda^{(2)}_{\angle,\delta}(x,y-2\delta)\\ &\quad+\Lambda^{(2)}_{\angle,\delta}(x+2\delta,y-2\delta)-2[\Lambda^{(2)}_{\angle,\delta}(x-2\delta,y)\\ &\quad-2\Lambda^{(2)}_{\angle,\delta}(x,y)+\Lambda^{(2)}_{\angle,\delta}(x+2\delta,y)]\\ &\quad+\Lambda^{(2)}_{\angle,\delta}(x-2\delta,y+2\delta)-2\Lambda^{(2)}_{\angle,\delta}(x,y+2\delta)\\ &\quad+\Lambda^{(2)}_{\angle,\delta}(x+2\delta,y+2\delta)].\end{aligned}$$

In general, we can show that $B^{(n)}_{\angle,\delta}(x, y)$ is exactly the binary piecewise polynomial function defined in (8). We have

$$\begin{aligned}B^{(n)}_{\angle,\delta}(x,y) &= \Omega^{(n)}_{\alpha,\beta,\delta}(x,y)\\ &= \frac{1}{(4\delta^2)^n}\sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{i+j}C^i_nC^j_nF_{i,j}(x,y),\end{aligned}$$

where

$$F_{i,j}(x,y)=A^{(n)}_{\alpha,\beta}(x+(n-2i)\delta,\ y-(n-2j)\delta).$$

When $\alpha > 0$, $\beta = 0$, the infinite angle polygon $\angle$ becomes the third quadrant of the coordinate plane. In this case, it can be shown that, again with the use of the principle of mathematical induction, $B^{(n)}_{\angle,\delta}(x, y)$ can be expressed, using the smooth unit step function $H_n(x)$ given in (12), in the following form.

$$B^{(n)}_{\angle,\delta}(x,y)=H_n\left(-\frac{x}{\delta}\right)H_n\left(-\frac{y}{\delta}\right)$$

Note that when $\alpha = 0$, the infinite angle polygon $\angle$ degenerates into a single ray from the coordinate origin along the negative direction of the y-axis. In this situation, $B^{(n)}_{\angle,\delta}(x, y) = 0$.

Note that if $\angle^*$ is the mirrored angle polygon of $\angle$ about the y-axis, we have $B^{(n)}_{\angle^*,\delta}(x, y) = B^{(n)}_{\angle,\delta}(-x, y)$. As the nonvertical edge of the angle polygon $\angle^*$ has a negative slope, it can be considered as a special case of $\angle$ when we allow β to take a negative value. Therefore, for the infinite angle polygon $\angle$, we have the following result.

$$B^{(n)}_{\angle,\delta}(x,y)=\begin{cases}0, & \alpha=0\\ H_n\left(-\frac{x}{\delta}\right)H_n\left(-\frac{y}{\delta}\right), & \alpha>0,\beta=0\\ \Omega^{(n)}_{\alpha,\beta,\delta}(x,y), & \alpha>0,\beta>0\\ \Omega^{(n)}_{\alpha,-\beta,\delta}(-x,y), & \alpha>0,\beta<0\end{cases} \quad (26)$$

Now, it can be seen clearly that the implicit point $\mathbb{P}^{(n)}_{\alpha,\beta,\delta}(x, y; \mathbf{P}_0)$ introduced in Definition 3.1 at a point $\mathbf{P}_0(x_0, y_0)$ is simply a translation of the implicit function $B^{(n)}_{\angle,\delta}(x, y)$ from the coordinate origin to $\mathbf{P}_0(x_0, y_0)$. With this result at hand, it is comparatively a simple step to show that the implicit edge defined in Definition 3.2 is actually $B^{(n)}_{\sqcap,\delta}(x, y)$ corresponding a polygon specified by one finite edge with endpoints $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$ and two infinite rays along the negative direction of the y-axis from $\mathbf{P}_0$ and $\mathbf{P}_1$, respectively.

In fact, for any line segment defined by two points $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$, three cases may occur if it is not parallel to the y-axis: The slope of the line segment is positive, negative, or zero. In any of these cases, the binary piecewise polynomial $B^{(n)}_{\sqcap,\delta}(x, y)$ constructed over the infinite polygon corresponding to the line segment from $\mathbf{P}_0(x_0, y_0)$ to $\mathbf{P}_1(x_1, y_1)$ can obviously be expressed as the difference of the implicit vertices at $\mathbf{P}_0(x_0, y_0)$ and $\mathbf{P}_1(x_1, y_1)$ in the way described in Definition 3.3. □

REFERENCES

BAJAJ, C. L., CHEN, J., AND XU, G. 1994. Free form surface design with A-patches. In *Proceedings of Graphics Interface*. 174–181.

BAJAJ, C. L. AND XU, G. 1999. A-splines: Local interpolation and approximation using g^k-continuous piecewise real algebraic curves. *Comput. Aided Geom. Des. 16*, 6, 557–578.

BAJAJ, C. L. AND XU, G. 2001. Regular algebraic curve segments(III)-applications in interactive design and data fitting. *Comput. Aided Geom. Des. 18*, 3, 149–173.

BARTHE, L., DODGSON, N. A., SABIN, M. A., WYVILL, B., AND GAILDRAT, V. 2003. Two-Dimentional potential fields for advanced implicit modeling operators. *Comput. Graph. Forum 22*, 1, 23–33.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3, 235–256.

FONG, P. AND SEIDEL, H.-P. 1992. An implementation of multivariate B-spline surfaces over arbitrary triangulations. In *Proceedings of the Conference on Graphics Interface '92*. Morgan Kaufmann, San Francisco, CA, 1–10.

HSU, P. C. AND LEE, C. 2002. The scale method for blending operations in functionally-based constructive geometry. *Comput. Graph. Forum 22*, 2, 143–158.

LI, Q. 2005. Implicit spline curves and surfaces. In *Proceedings of the Chinese Automation and Computing Society Conference in the UK (CACSUK'05)*, H. Hu et al., Eds. Pacilantic International Ltd, Colchester, England, 201–206.

LI, Q. 2007. Smooth piecewise polynomial blending operations for implicit shapes. *Comput. Graph. Forum 26*, 2, 157–171.

LI, Q., GRIFFITHS, J. G., AND WARD, J. 2006. Constructive implicit fitting. *Comput. Aided Geom. Des. 23*, 1, 17–44.

LI, Q. AND PHILLIPS, R. 2004. Implicit curve and surface design using smooth unit step functions. In *Proceedings of the 9th ACM Symposium on Solid Modeling and Applications*, G. Elber et al., Eds. Eurographics Association, 237–242.

LI, Q., WILLS, D., PHILLIPS, R., VIANT, W. J., GRIFFITHS, J. G., AND WARD, J. 2004. Implicit fitting using radial basis functions with ellipsoid constraint. *Comput. Graph. Forum 23*, 1, 55–70.

LOOP, C. AND BLINN, J. 2006. Real-Time GPU rendering of piecewise algebraic surfaces. In *ACM SIGGRAPH Papers*. ACM, New York, 664–670.

PASKO, A. A., ADZHIEV, V., SOURIN, A., AND SAVCHENKO, V. V. 1995. Function representation in geometric modeling: Concepts, implementation and applications. *The Visual Comput. 11*, 8, 429–446.

PASKO, G. I., PASKO, A. A., AND KUNII, T. L. 2005. Bounded blending for function-based shape modeling. *IEEE Comput. Graph. Appl. 25*, 2, 36–45.

PAYNE, B. A. AND TOGA, A. W. 1992. Distance field manipulation of surface models. *IEEE Comput. Graph. Appl. 12*, 1, 65–71.

PURCELL, T. J., BUCK, I., MARK, W. R., AND HANRAHAN, P. 2005. Ray tracing on programmable graphics hardware. In *ACM SIGGRAPH Courses*. ACM, New York, 268.

RICCI, A. 1973. A constructive geometry for computer graphics. *Comput. J. 16*, 3, 157–160.

SCHMIDT, R. AND WYVILL, B. 2005. Generalized sweep templates for implicit modeling. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE'05)*. ACM, New York, 187–196.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2005. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH Courses*. ACM, New York, 204.

TATARCHUK, N., SHOPF, J., AND DECORO, C. 2007. Real-Time isosurface extraction using the GPU programmable geometry pipeline. In *ACM SIGGRAPH Courses*. ACM, New York, 122–137.

TAUBIN, G. 1991. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 13*, 11, 1115–1138.

TURK, G. AND O'BRIEN, J. F. 1999. Shape transformation using variational implicit functions. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley, New York, 335–342.

TURK, G. AND O'BRIEN, J. F. 2002. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph. 21*, 4, 855–873.

XU, G., BAJAJ, C. L., AND CHU, C. I. 2000a. Regular algebraic curve segments(II)-interpolation and approximation. *Comput. Aided Geom. Des. 17*, 6, 503–519.

XU, G., BAJAJ, C. L., AND XUE, W. 2000b. Regular algebraic curve segments(I)-definitions and characteristics. *Comput. Aided Geom. Des. 17*, 6, 485–501.

Received May 2008; revised October 2008; accepted February 2009