Computer-Aided Design 43 (2011) 394–409

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Partial shape-preserving splines

Qingde Li[a,*], Jie Tian[b]

[a] Department of Computer Science, University of Hull, Hull, HU6 7RX, UK
[b] Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

ARTICLE INFO

Article history:
Received 4 August 2010
Accepted 6 January 2011

Keywords:
CAGD
Spline basis functions
NURBS
Shape parameters
Tension control
Polygon smoothing

ABSTRACT

A complex geometric shape is often a composition of a set of simple ones, which may differ from each other in terms of their mathematical representations and the ways in which they are constructed. One of the necessary requirements in combining these simple shapes is that their original shapes can be preserved as much as possible. In this paper, a set of partial shape-preserving (PSP) spline basis functions is introduced to smoothly combine a collection of shape primitives with flexible blending range control. These spline basis functions can be considered as a kind of generalization of traditional *B*-spline basis functions, where the shape primitives used are control points or control polygons. The PSP-spline basis functions have all the advantages of the conventional *B*-spline technique in the sense that they are nonnegative, piecewise polynomial and of property of partition of unity. However, PSP-spline is a more powerful freeform geometric shape design technique in the sense that it is also a kind of shape-preserving spline. In addition, the PSP-spline technique implicitly integrates the weights of shape control primitives into its basis functions, which allows users to design a required geometric shape based on weighted control primitives. Though its basis functions are simply piecewise polynomial functions, it has the same shape design strengths as the rational piecewise polynomial based spline techniques such as NURBS. In particular, when control shape primitives are specified as a set of control points, PSP-spline behaves like a polygon smoother, with which a shape can be designed to approximate the specified control polygon or control mesh smoothly with any required precision. Consequently, a richer set of geometric shapes can be built using a relatively smaller set of control points.

1. Introduction

Representing a complex geometric shape as a composition of a set of simple geometric primitives is a widely used shape modeling method in computer graphics and computer aided geometric design. The CSG (constructive solid geometry) technique, a solid shape modeling technique used to combine simple implicit geometric primitives into much more complex ones, is directly developed from this idea. One typical shape design feature offered by CSG is that it is a kind of shape-preserving technique, which allows shape designers to maintain partially the original geometric primitives. However, we still lack an efficient and effective constructive method similar to CSG for building parametrically represented shapes. One approach is to extend the conventional control polygon based spline technique into a general shape blender to allow it to smoothly combine a set of control primitive shapes, rather than just control points. One of the necessary requirements for such a kind of shape blender is its flexibility in controlling to what extent the original shape features specified by the control primitives should be maintained. Obviously, conventional spline basis functions, when used to weight a set of control primitives, do not meet this requirement.

In this paper, a new type of parametric spline technique is proposed based on the spline basis functions introduced in [1]. These basis functions are partially shape preserving in the sense that the shapes of the primitive control geometries can be maintained to any required extent. This partial shape-preserving feature of the proposed spline basis functions allows one to design a complex shape following the idea of divide-and-conquer, which works by recursively subdividing a shape into two or more simple shapes, until these sub-shapes become simple enough to be designed directly using certain parametric geometric primitives. In addition, the geometric primitives used in this technique to build a required geometric shape can be of different mathematical forms depending on the convenience and effectiveness of representing these shapes.

The proposed PSP-spline is particularly efficient in designing geometric shapes which have multiple flat parts. Although a variety of freeform CAGD techniques and tools are available for designing virtually any kind of geometric shape, most of them are not very efficient and effective in designing a freeform geometric

* Corresponding author. Tel.: +44 1482 465212.
E-mail addresses: q.li@hull.ac.uk (Q. Li), tian@ieee.org (J. Tian).

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.01.007

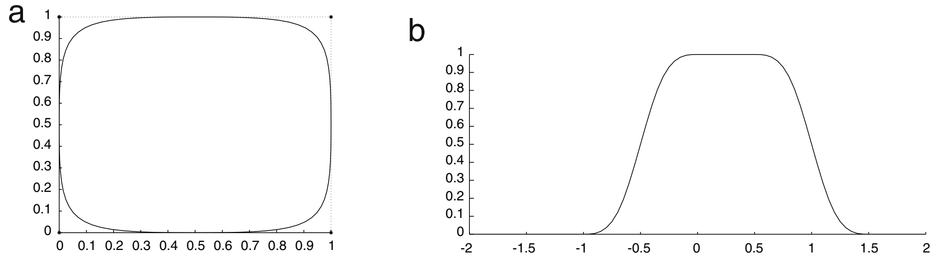


Fig. 1. (a) The shape of a rounded square can be well captured using just four control points. (b) A spline basis function can be designed to take value one flexibly in a subregion of its support.

shape with parts of its surface being flat. Let us consider the problem of designing the curves shown in Fig. 1(a). If these curves are designed using *B*-spline or NURBS, more than four control points are required to achieve the smoothness for the shape near the vertices of the specified control polygon, even though the control polygon defined by the four corner vertices is sufficient enough to capture the main feature of the rounded square. This is because, when conventional spline techniques are used, more than one control points are required to achieve a certain smoothness at each corner of the square. In addition, all these control points have to be carefully coordinated, where delicate attention is required. However, with the shape design technique proposed in this paper, this design task can be easily accomplished.

As will be seen later, the proposed PSP-spline basis functions have similar properties to the conventional *B*-spline basis functions. For instance, all these spline basis functions are piecewise polynomial and can be built to any required degree of smoothness. They are all nonnegative and have the property of partition of unity. In addition, these basis functions offer flexibility for users to associate a weight implicitly to each individual control primitive and to design a required shape using weighted control primitives. Furthermore, these spline basis functions are also of the simplicity and elegancy of the conventional *B*-spline basis functions. They are easy to implement and can be evaluated efficiently.

The development of these spline basis functions begins with a practical observation on the energy required to bend a real world curved wire, such as iron bars of different radii. To bend a highly rigid thick iron wire to a required shape, either the wire should be sufficiently long or the force used to bend the wire should be very large. However, for a very pliable wire, it can be easily bent to have any required curvature due to the fact that the force applied to the wire is only effective in a small range for a highly pliable wire.

Consider a control primitive curve represented in parametric form as $\mathcal{C}(t), t \in [a, b]$, and observe in what a way one can accommodate the pliability of the curve by introducing a weight function $B(t), 0 \leq B(t) \leq 1$, when modifying the curve $\mathcal{C}(t)$ as $\mathcal{D}(t) = B(t)\mathcal{C}(t)$. If we assume that the curve $\mathcal{C}(t)$ represents a quite rigid shape and we do not want it to be changed significantly when it is multiplied by $B(t)$, then the values of $B(t)$ must be close to 1. In order to be able to bend a rigid curve to achieve a required curvature, the length of the curve segment must be sufficiently long. With these two considerations in mind, a much clearer picture of the required spline basis function emerges. $B(t)$ should take the shape shown in Fig. 1(b). Suppose a curve is designed using a sequence of control curves with different degrees of pliability and let $\{B_i(t)\}_{i=1}^n$ be the sequence of spline basis functions developed to meet the above mentioned requirements. Then a curve can be described in the following way in parametric form:

$$\mathbf{P}(t) = \sum_{i=1}^{n} B_i(t)\mathcal{C}_i(t),$$

where the blending basis function $B_i(t)$ is designed according to the pliability of each specified primitive curve $\mathcal{C}_i(t)$. In other words, a curve primitive $\mathcal{C}_i(t)$ with high pliability should be associated with a basis function $B_i(t)$ with a relatively longer support.

To design a parametric curve using PSP-spline technique, one need only to specify a sequence of intervals according to the set of control shapes used in the design process. Then, for each interval, a spline basis function $B_i(t)$ can be built directly as the difference of two smooth unit step functions built according to the left and right ends of the interval. The novel feature of the newly proposed spline design scheme is that they can be tuned to approximate a given control polygon, or more generally, a given set of control boundary curves, to any required precision. Therefore, the design technique can also be referred to as a kind of polygon smoothing technique. Furthermore, the proposed design technique can also be used to approximate certain types of quadratic curves such as circle and ellipse, though it is impossible to provide an exact representation for these kinds of curves using piecewise polynomial curves.

Another feature of the proposed spline technique is that the weights considered in the conventional NURBS are interpreted as the length of the support of the PSP-spline basis functions and are implicitly built into the spline technique. Therefore, although the PSP-spline basis functions are simply represented in piecewise polynomials, it is even more powerful in designing freeform parametric shapes than the conventional NURBS.

In the rest of the paper, we will first briefly discuss some related work and the basic properties of the required spline basis functions. Then we consider how to construct smooth piecewise polynomial unit step functions, the building blocks of the PSP-spline technique. This is then followed by the construction of the PSP-spline basis functions and how they can be used to design smooth freeform curves. In Section 6, the generalization of PSP-spline to 2D is introduced, together with some surface design examples.

2. Related work

Spline based shape design techniques, such as *B*-spline and NURBS curves and surfaces, are very powerful in generating geometric shapes used in computer graphics, computer games and computer aided geometric design [2,3]. However, one still finds that these techniques are not very efficient and effective in designing certain complex geometric objects in terms of their mathematical representation and convenience in specifying the underlying control polygons, though it is possible in practice to design any required shapes using just *B*-spline or NURBS. It has been observed that the design flexibility and capability of a spline based geometric design scheme can be enhanced by introducing new shape parameters. Several techniques based on this observation have been proposed so far, such as Beta-spline [4,5], rational Beta-splines [6], and α-spline [7]. In Beta-spline, Barsky and Beatty introduced two shape parameters, known as bias and tension, into conventional uniform cubic *B*-spline modeling scheme by considering the first and the second order geometric continuity, respectively. Later, Barsky proposed further the rational Beta-spline by introducing weights to the Beta-spline. As the Beta-splines are the extension of conventional B-splines, rational Beta-splines can be considered as a kind of generalization of NURBS. More recently, Tai and Loe proposed another way of generalizing conventional NURBS and proposed α-spline by blending a sequence of singular reparameterized line segments. However, none of these splines offer the design feature of partial shape preserving when the control points or control polygons are extended to a set of general control geometric primitives. In [8], a kind of partial shape-preserving NURBS was introduced to enhance the design capability of conventional spline techniques. However, the technique presented in this paper is piecewise polynomial based, which is more appealing in terms of its theoretical simplicity and computational efficiency.

3. Partial shape-preserving spline basis functions

In spline geometry, there does not exist a commonly accepted definition on what a basis function is [9]. In shape design practice, different types of spline basis functions have been introduced to deal with different practical shape design problems, such as Bernstein polynomial, the B-spline basis functions, as well as the rational B-spline basis functions.

In general, a basis function in k-dimensional parametric space R^k can be understood as a mapping

$$B : R^k \to R$$

satisfying the following properties:

(1) $0 \le B(X) \le 1$, for all $X \in R^k$.
(2) For any real number $\alpha \in R$, $\{X | X \in R^k, B(X) \ge \alpha\}$ is a connected set.
(3) $B(X)$ should be piecewise polynomial and has a certain degree of smoothness, such as C^2-continuity.
(4) Each $B(X)$ should be easy to compute and numerically stable.
(5) For a given partition $\{\Delta_m\}_{m=0}^{M}$ of domain $D \subseteq R^k$, that is,

$$\bigcup_{m=0}^{M} \Delta_m = D, \qquad \text{Area}(\Delta_i \cap \Delta_j) = 0, \quad \text{when } i \neq j,$$

all the spline basis functions $B_{\Delta_m}(X)$, $m = 0, 1, \ldots, M$, built upon the partition should sum to one. That is

$$\sum_{m=0}^{M} B_{\Delta_m}(X) = 1.$$

However, to implement the idea addressed in Section 1, $B(X)$ need to have such a distinctive feature that it not only has similar shape design capability to the conventional B-spline technique for designing smooth freeform curves and surfaces, but can also be used to design those kinds of freeform curves and surfaces which are a composition of some premade shapes. That is, when these spline basis functions are used as a kind of blender to combine a set of primitive control shapes, users are able to specify to what extent they would like to maintain their original shapes of these primitive control shapes. Thus, in addition to the above mentioned requirements, the following partial shape-preserving condition should also be met by the spline basis function $B(X)$:

(6) Each $B_{\Delta_m}(X)$ should be built according to the pliability of the corresponding primitive shape and can be built to take value one in a subarea of Δ_m.

4. Piecewise polynomial smooth unit step functions

One way to build the univariate shape-preserving spline basis functions is to use piecewise polynomial smooth unit step functions. Piecewise polynomial smooth unit step function is first introduced for blending implicit shapes in [10]. The function is iteratively defined starting with the standard Heaviside unit step function and can be defined in slightly different ways. The following form is the one used in [11].

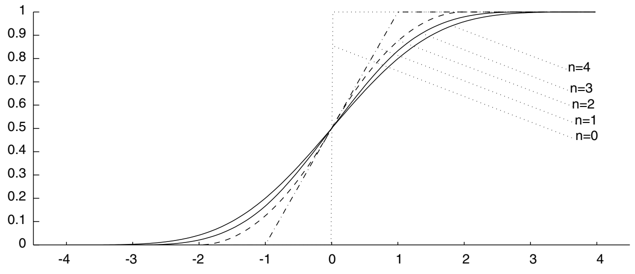


Fig. 2. Piecewise polynomial smooth unit step functions of degree 1–4.

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

$$H_n(x) = \frac{1}{2}\left(\left(1 + \frac{x}{n}\right) H_{n-1}(x+1) + \left(1 - \frac{x}{n}\right) H_{n-1}(x-1)\right), \quad n = 1, 2, 3, \ldots. \tag{2}$$

$H_n(x)$ can be considered as a generalization of the Heaviside unit step function and is called the degree n smooth unit step function. It can be shown that $H_n(x)$ has the following properties:

Proposition 4.1. *For each function $H_n(x)$, we have*

(1) *$H_n(x)$ is C^{n-1}-continuous for $n > 1$;*
(2) *$H_n(x)$ is a piecewise polynomial function of degree n;*
(3) *$H_n(x)$ is monotonically increasing and takes value 1 when $x \ge n$, and 0 when $x \le -n$;*
(4) *$H_n(x) + H_n(-x) = 1$, $H_n(0) = \frac{1}{2}$;*
(5) *$H_n(x) \ge H_{n-1}(x)$ when $x < 0$ and $H_n(x) \le H_{n-1}(x)$ when $x \ge 0$, $n = 1, 2, \ldots$;*
(6) *$x(2H_n(x) - 1) \le x(2H_{n-1}(x) - 1)$, $n = 1, 2, \ldots$.*

The proof of Proposition 4.1(1)–(5) can be obtained directly using the Principle of Mathematical Induction. As for Proposition 4.1(6), it can be obtained from the fact that $2H_n(x) < 1$ when $x < 0$ and $2H_n(x) \ge 1$ when $x \ge 0$ and Proposition 4.1(5).

Following the definition of $H_n(t)$ given in Eq. (2), the functions $H_1(x), H_2(x)$, and $H_3(x)$ can be written out explicitly. Note that $H_n(x) = 1 - H_n(-x)$, we need only write out these functions for $x \le 0$.

$$H_1(x) = \begin{cases} 0, & x < -1; \\ \dfrac{1+x}{2}, & -1 \le x \le 0. \end{cases} \tag{3}$$

$$H_2(x) = \begin{cases} 0, & x < -2; \\ \dfrac{1}{2}\left(1 + \dfrac{x}{2}\right)^2, & -2 \le x < 0. \end{cases} \tag{4}$$

$$H_3(x) = \begin{cases} 0, & x < -3; \\ \dfrac{1}{48}(3+x)^3, & -3 \le x < -1; \\ \dfrac{1}{24}(12 + 9x - x^3), & -1 \le x < 0. \end{cases} \tag{5}$$

Fig. 2 presents a plot of the smooth unit step functions of degree 1–4.

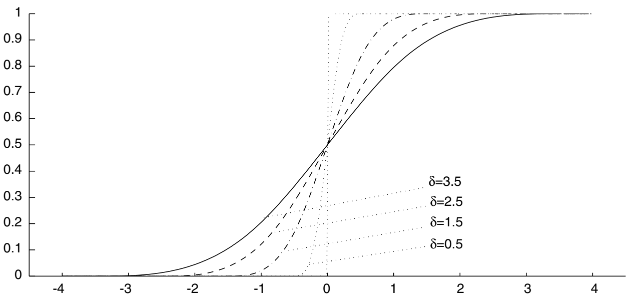


Fig. 3. Piecewise polynomial smooth unit step function $H_{3,\delta}(x)$ with different rising ranges specified using δ.

The smooth unit step function $H_n(x)$ defined above can also be written explicitly using the Heaviside unit step function in the following way:

$$H_n(x) = \frac{1}{n!2^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x+(n-2k))^n H_0(x+(n-2k)). \quad (6)$$

From (6), the generalized smooth unit step functions of degree 1–3 can also be expressed in the following form:

$$H_1(x) = \frac{1}{2}((x+1)H_0(x+1) - (x-1)H_0(x-1)) \quad (7)$$

$$H_2(x) = \frac{1}{8}((x+2)^2H_0(x+2) - 2x^2H_0(x) + (x-2)^2H_0(x-2)) \quad (8)$$

$$H_3(x) = \frac{1}{48}((x+3)^3H_0(x+3) - 3(x+1)^3H_0(x+1) \quad (9)$$

$$+\,3(x-1)^3H_0(x-1) - (x-3)^3H_0(x-3)). \quad (10)$$

As can be observed directly, the degree n smooth unit step function $H_n(x)$ is strictly monotone increasing over the interval $[-n, n]$. We call this interval the rising range of a smooth unit step function. The smooth unit step function with any specified rising range can be defined easily by introducing a nonnegative number $\delta > 0$ as follows:

$$H_{n,\delta}(x) = H_n(nx/\delta). \quad (11)$$

Obviously, $H_{n,\delta}(x) = 1$ when $x \geq \delta$, and $H_{n,\delta}(x) = 0$ when $x < -\delta$.

Fig. 3 shows some C^2-continuous cubic unit step functions $H_{3,\delta}(x)$ with different values of rising range parameter δ.

The derivatives of smooth unit step functions can be found easily for $n > 1$. From (6), it can be seen that the derivatives of $H_n(x)$ can be directly expressed explicitly using the Heaviside unit step function. When $n > 1$,

$$H_n'(x) = \frac{1}{(n-1)!2^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x+(n-2k))^{n-1} \times H_0(x+(n-2k)). \quad (12)$$

In general, for $i < n$, the ith order derivative of degree n smooth unit step function $H_n(x)$ can be expressed explicitly as

$$H_n^{(i)}(x) = \frac{1}{(n-i)!2^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x+(n-2k))^{n-i} \times H_0(x+(n-2k)). \quad (13)$$

With (13), the relevant derivatives for $H_2(x)$ and $H_3(x)$ can immediately be obtained as

$$H_2'(x) = \frac{1}{4}((2+x)H_0(x+2) - 2xH_0(x) + (x-2)H_0(x-2)) \quad (14)$$

$$H_3'(x) = \frac{1}{16}((x+3)^2H_0(x+3) - 3(x+1)^2H_0(x+1) + 3(x-1)^2H_0(x-1) - (x-3)^2H_0(x-3)) \quad (15)$$

$$H_3''(x) = \frac{1}{8}((x+3)H_0(x+3) - 3(x+1)H_0(x+1) + 3(x-1)H_0(x-1) - (x-3)H_0(x-3)). \quad (16)$$

Fig. 4 shows the shapes of the derivatives of $H_5(x)$.

5. PSP-spline basis functions

With smooth unit step function $H_{n,\delta}(x)$, a new type of spline basis function can be introduced immediately. Let $[a, b]$ be an interval with $a \leq b$. We define

$$B^{(n)}_{[a,b],\delta}(x) = H_{n,\delta}(x-a) - H_{n,\delta}(x-b), \quad (17)$$

where δ is a parameter used for controlling the blending range when $B^{(n)}_{[a,b],\delta}(x)$ is used as a shape blending function.

Fig. 5 shows the shapes of the cubic PSP-spline basis function $B^{(3)}_{[2,6],\delta}(x)$ defined over the interval $[2, 6]$ with different δ values.

With the properties of $H_{n,\delta}(x)$, it can be seen that $B^{(n)}_{[a,b],\delta}(x)$ has the following properties:

1. *Nonnegativity*. $0 \leq B^{(n)}_{[a,b],\delta}(x) \leq 1$.
2. *Smoothness*. $B^{(n)}_{[a,b],\delta}(x)$ is C^{n-1}-continuous.
3. *Convexity*. For any level values, the level set of $B^{(n)}_{[a,b],\delta}(x)$ is an interval.
4. *Additivity*. For any $c \in [a, b]$,
   $B^{(n)}_{[a,b],\delta}(x) = B^{(n)}_{[a,c],\delta}(x) + B^{(n)}_{[c,b],\delta}(x)$.
5. *Partition of unity*. Let $t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$ be a sequence of numbers. Then the sequence of PSP-spline basis functions built upon the sequence of intervals $(-\infty, t_0], [t_0, t_1], [t_1, t_2], \cdots [t_n, \infty)$ has the property of partition of unit. That is,
   $$\sum_{i=0}^{n+1} B^{(n)}_{[t_{i-1},t_i],\delta}(x) = 1, \quad (18)$$
   where t_{-1} and t_{n+1} are assumed to be $-\infty$ and ∞, respectively.

Fig. 6 shows the shapes of cubic PSP-spline basis functions $B^{(3)}_{[a,b],\delta}(x)$ defined with the sequence of intervals specified corresponding to the following knot sequence: −5, −4, 0, 2, 5.

The PSP-spline basis functions introduced in (17) are defined based on a single smoothing parameter δ. A non-symmetric PSP-spline basis function can be defined over a given interval $[a, b]$ in the following way by associating different interval ends with different smoothing parameters δ_a, δ_b:

$$B^{(n)}_{[a,b],\delta_a,\delta_b}(x) = H_{n,\delta_a}(x-a) - H_{n,\delta_b}(x-b). \quad (19)$$

In general, $B_{[a,b],\delta_a,\delta_b}$ may not necessarily be nonnegative as $H_{n,\delta_a}(x-a)$ may be smaller than $H_{n,\delta_b}(x-b)$ for some x. This happens when $0 \leq b-a < \delta_b - \delta_a$ or $0 \leq b-a < \delta_a - \delta_b$. As long as the difference between δ_a and δ_b is smaller than the interval length $b-a$, $B_{[a,b],\delta_a,\delta_b}$ will be nonnegative. Some examples of non-symmetric PSP-spline basis functions are plotted in Fig. 7. However, in this paper we will only investigate the properties of the PSP-spline basis functions constructed with one common smoothing parameter δ.

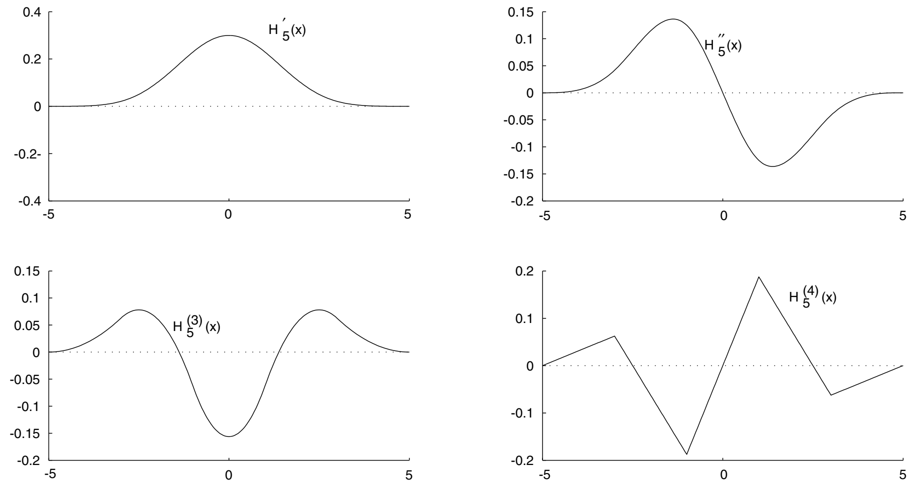


Fig. 4. The derivatives of the C^4 smooth unit step functions $H_5(x)$.

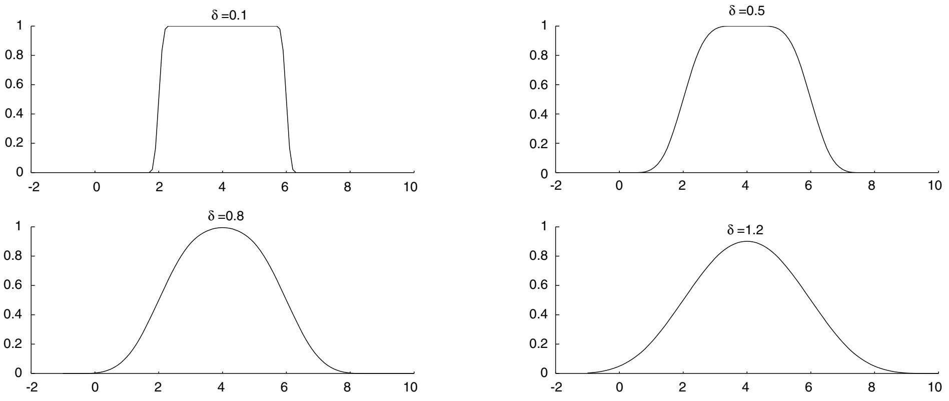


Fig. 5. Cubic PSP-spline basis function $B^{(3)}_{[2,6],\delta}(x)$ defined over the interval [2, 6] with different values of δ.

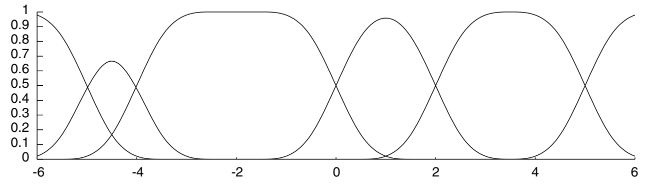


Fig. 6. The cubic PSP-spline basis functions $B^{(3)}_{[a,b],\delta}(x)$ built upon intervals $(-\infty, -5]$, $[-5, -4]$, $[-4, 0]$, $[0, 2]$, $[2, 5]$, $[5, \infty)$ with $\delta = 0.5$.

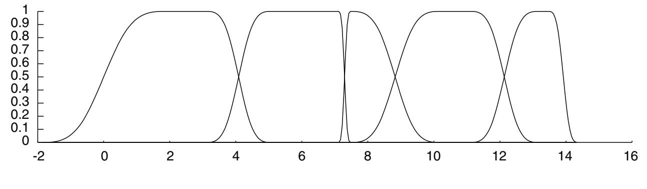


Fig. 7. The cubic non-symmetric PSP-spline basis functions.

When equal spaced knots are used, the conventional *B*-spline basis functions built upon an equal spaced knot sequence can be considered as a special case of the PSP-spline basis functions corresponding to a specific polygon smooth parameter δ used in building the PSP-spline basis functions. For instance, the cubic *B*-spline basis functions constructed using knots 0, 1, 2, ... will be the same as the cubic PSP-spline basis functions built with $\delta = 1.5$ corresponding to the intervals [1.5, 2.5], [2.5, 3.5], ... In general, for $n \geq 1$, the nth degree *B*-spline basis functions built with knots 0, 1, 2, ... are identical with PSP-spline basis functions $B^{(n)}_{[a_i,a_{i+1}],\delta}(t)$ built from intervals $[a_i, a_{i+1}]$, $a_i = i + n/2, i = 0, 1, 2, \ldots$, using polygon smooth parameter $\delta = n/2$.

For nonequal spaced knots, PSP-spline basis functions are in general different from the *B*-spline basis functions. This is because, when the order of a *B*-spline basis function is greater than two, the shape of *B*-spline basis functions depends on the lengths of all the knot spaces covered by its support, while the shape of a PSP-spline basis function depends only on the smoothing parameters associated with the ends of the interval upon which it is constructed.

6. Curve design using PSP-spline basis functions

In this section, we discuss the strengths of the PSP-spline in designing freeform parametric curves. As will be seen from the investigation, this newly proposed spline technique offers more design flexibility and power than traditional spline techniques.

6.1. Control polygon based curve design

Let $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n$ be the $n + 1$ vertices of a shape control polygon. For each control point $\mathbf{P}_i$, a univariate PSP-spline basis function $B_i(t)$ can be designed to specify the influence range of the control point. If all these points are treated equally, an equal spaced knot set can be used to build PSP-spline basis functions $B_i(t)$, $i = 0, 1, \ldots, n$. For instance, for a given $\delta > 0$, the following

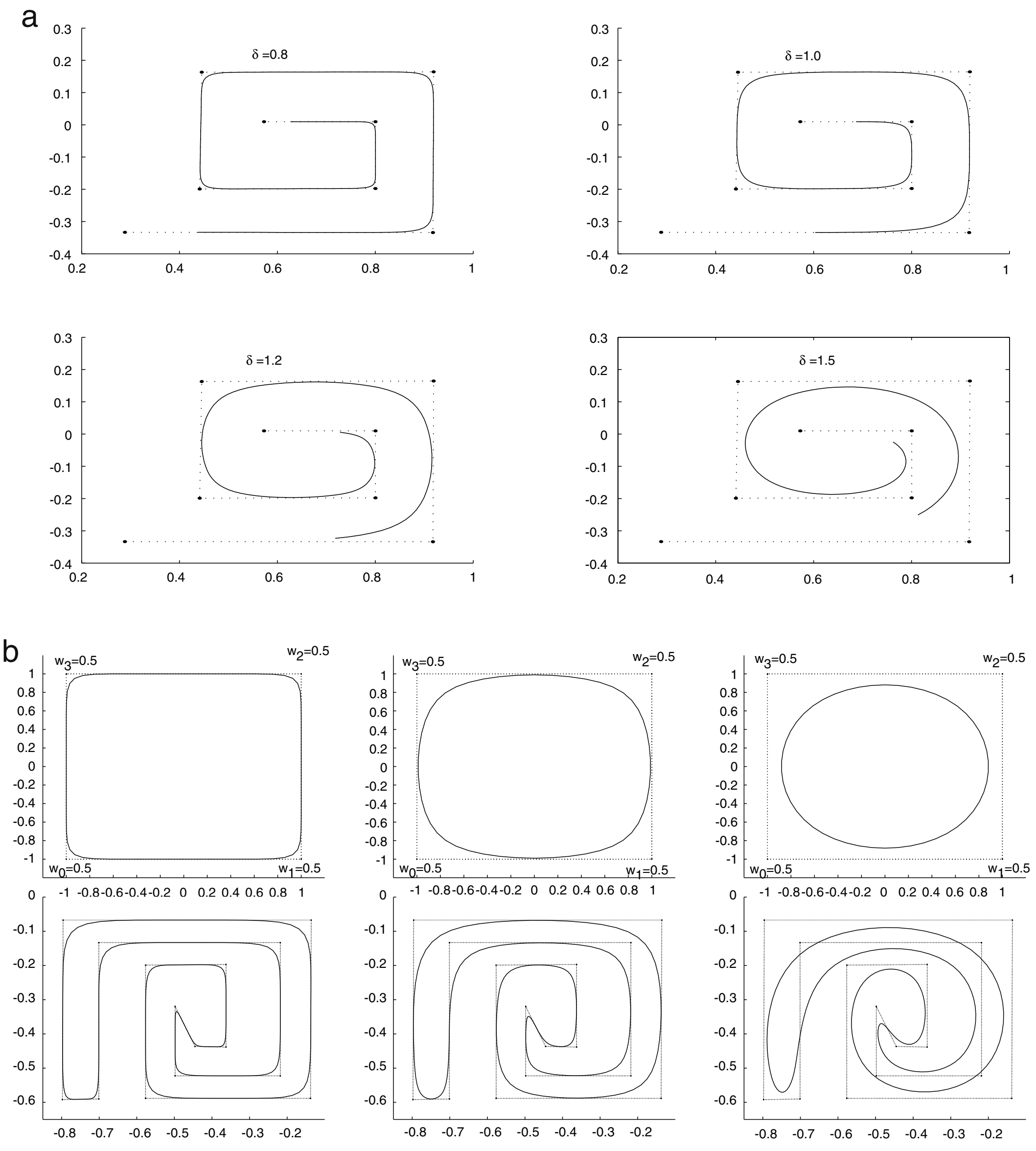


Fig. 8. Curves designed using cubic PSP-spline basis functions built with equal spaced intervals.

set of C^2-continuous PSP-spline basis functions built upon knots $0, 1, 2, \ldots, n$ can be used to blend these control points:

$B^{(3)}_{[0,1],\delta}(t),\ B^{(3)}_{[1,2],\delta}(t),\ B^{(3)}_{[2,3],\delta}(t),\ \cdots,\ B^{(3)}_{[n-1,n],\delta}(t),\ B^{(3)}_{[n,n+1],\delta}(t).$

With this set of spline basis functions, the designed shape can be described parametrically as

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B^{(3)}_{[i,i+1],\delta}(t). \tag{20}$$

Curves displayed in Fig. 8 are designed in this way.

In practice, it is often required that the designed curve interpolates certain control points' choosing by curve designers. In the case of *B*-spline based curve design, this is achieved basically in two ways, either using repeated control points or using duplicated knots. For a PSP-spline based curve, to let the designed curve interpolate some chosen control points is even simpler. In fact, we can interpret the length of the interval upon which the PSP-spline basis function is built as a kind of weight associated with a control point or a kind of rigidity of the designed curve around a control point. The longer the interval, the bigger is the weight associated with a control point. To let the designed curve interpolate a given control point, one need only to make sure that the length of the interval used to construct the PSP-spline basis function associated with the control point is longer than 2δ, this is because in this situation, the spline basis function will take value 1 in the middle of the interval.

To let the designed curve interpolate the first and the last control point, we need only use a relatively smaller knot for the left end of the first interval and a relatively bigger knot for the last interval.

To illustrate the strengths of the proposed spline technique, we describe here step by step how to generate a rich set of curves using six control points $\mathbf{P}_1(-3, 0)$, $\mathbf{P}_2(-1, 0)$, $\mathbf{P}_3(-3, 5)$, $\mathbf{P}_4(3, 5)$, $\mathbf{P}_5(1, 0)$, $\mathbf{P}_6(3, 0)$.

1. First specify the weights associated with each control point:

   $w_1, w_2, w_3, w_4, w_5, w_6.$

   Usually, we assume that each $w_i > 0$ and all the weights sum to unity, but this is not essential. These weights can be any positive numbers.

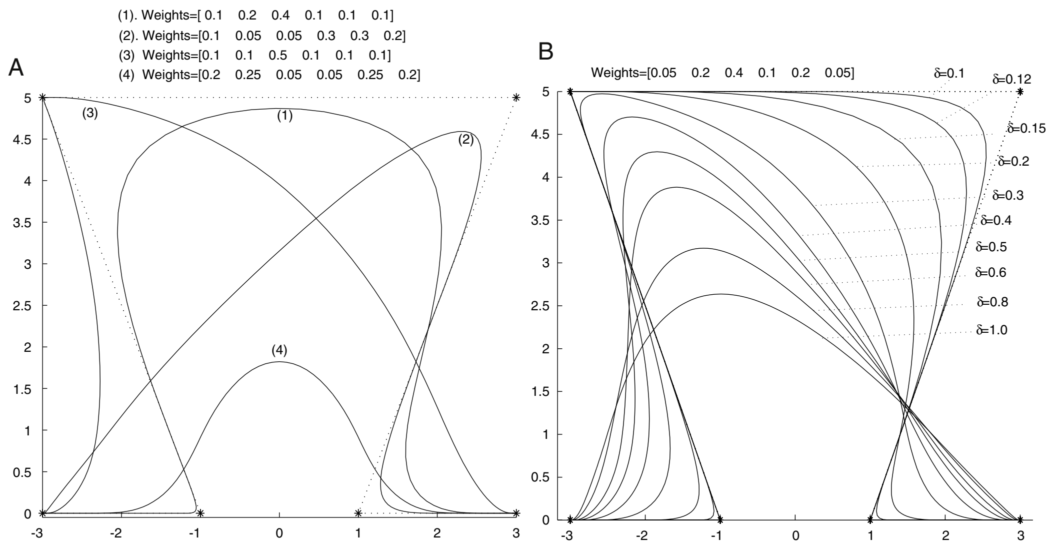
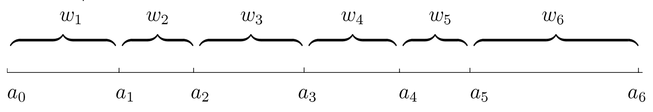


Fig. 9. (a) Four PSP-spline curves designed using four sets of weights and the same smoothing parameter value δ ($\delta = 0.25$). (b) Different PSP-spline curves produced using the same set of weights but different values for smoothing parameter δ.

2. Create a set of knots $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ based on the given weights, such that

$$w_i = a_{i+1} - a_i, \quad i = 0, 1, 2, \ldots, 5.$$



3. For each interval $[a_i, a_{i+1}]$ defined above, build spline basis functions $B^{(n)}_{[a_i,a_{i+1}],\delta}(t)$ according to (17).
4. Representing the designed curve parametrically as

$$\mathbf{P}(t) = \sum_{i=1}^{6} \mathbf{P}_i B^{(n)}_{[a_i,a_{i+1}],\delta}(t). \tag{21}$$

Some curves shown in Fig. 9 were designed based on the above process. Curves shown in Fig. 9(a) were designed using different sets of weights but the same δ value. As can be observed from the figure, this way of designing curves using PSP-splines has the same advantage as NURBS. However, the introduction of parameter δ into the design scheme provides an additional flexibility and dimension in shape design. As have been shown in Fig. 9(b), with the use of different δ values in the basis functions, various curves can be generated.

Some more example curves designed using nonequal weights are demonstrated in Figs. 10 and 11.

Since PSP-spline basis functions have exactly the same properties as *B*-spline basis functions, curves designed using PSP-spline basis functions are of similar properties to *B*-spline curves, such as convex-hull property, linear precision.

6.2. Curve blending

A required curve can also be described as a blend of a sequence of simple premade geometric primitives, such as line segments, quadratic and other mathematically defined parametric curves. In (20), instead of using a sequence of control points, a sequence of shape primitives represented in parametric form can be used to specify the geometric features of a required curve. That is, a curve can be designed to take the following form:

$$\sum_{i=0}^{n} \mathbf{P}_i(t) B^{(m)}_{[t_i,t_{i+1}],\delta}(t), \tag{22}$$

where $\mathbf{P}_i(t)$, $i = 0, 1, \ldots, n$, is a set of locally defined parametric curves.

The curve displayed in Fig. 12 is designed in this way, which is a blending of a helix curve with a circle. This idea can be very useful in the area of re-engineering, where parts of the curve to be designed are reconstructed using data sampled from some real world objects. With this curve design approach, a curve can be designed part by part individually. These individually designed curves do not have to take the same mathematical form. Depending on the design convenience, they may be expressed in different ways. For instance, some parts of the designed curve may be represented using sine and cosine functions, and some other parts are designed as polynomial or piecewise polynomial curves. When these curves are combined into one piece, their major original shape features can be maintained by choosing appropriate values for blending range control parameter δ and lengths of intervals on which the associated PSP-spline basis functions are built.

By using the shape-preserving feature of the PSP-spline basis functions, various design ideas used in curve design practice can be implemented easily in a much more intuitive way as special cases of this generalized curve design technique. For instance, a design scheme similar to the Hermite curve can be directly expressed in a form given in (22). Assume that the path of a moving particle is described by the sequence of data $\mathbf{P}_i, \mathbf{v}_i$, $i = 0, 1, 2, \ldots, n$, representing the positions and velocities at different moments of the particle. Then the path can be represented either using the idea of data interpolation or approximation in slightly different ways. For instance, when a curve is required to interpolate both the specified positions and velocities, the path can be expressed in general in the following form using the quadratic PSP-spline basis function $B^{(2)}_{[a_i,a_{i+1}],\delta}(t)$:

$$\mathbf{P}(t) = \sum_{i=0}^{n} (\mathbf{P}_i + (t - t_i)\mathbf{v}_i) B^{(2)}_{[a_i,a_{i+1}],\delta}(t) \tag{23}$$

where $t_i \in [a_i, a_{i+1}]$ and $B^{(2)}_{[a_i,a_{i+1}],\delta}(t_i) = 1$. It can be seen directly that, at the moment t_i, the designed curve will interpolate not only the position P_i, but also the velocity $\mathbf{v}_i$. That is, when $B^{(2)}_{[a_i,a_{i+1}],\delta}(t_i) = 1$,

$$\mathbf{P}(t_i) = \mathbf{P}_i, \qquad \dot{\mathbf{P}}(t_i) = \mathbf{v}_i.$$

Fig. 13 shows an example curve designed in this way.

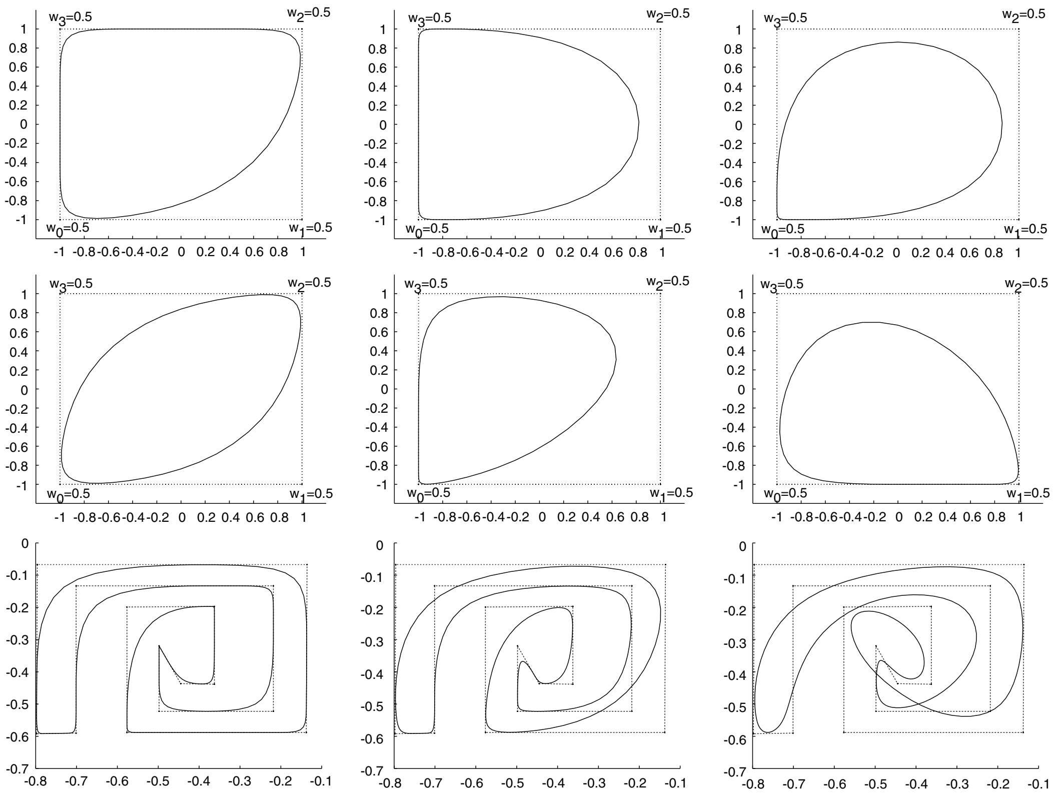
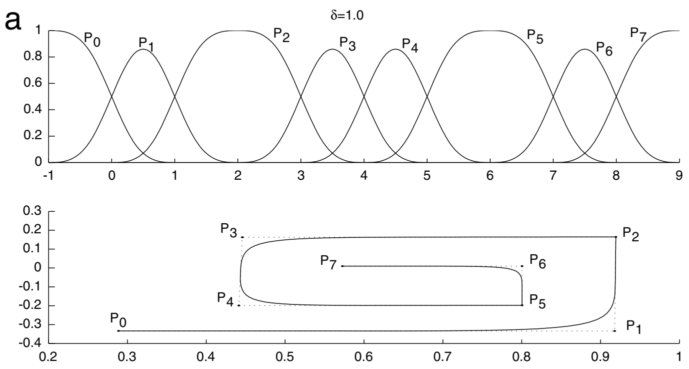


Fig. 10. Curves designed using cubic PSP-spline basis functions built from nonequal spaced intervals.



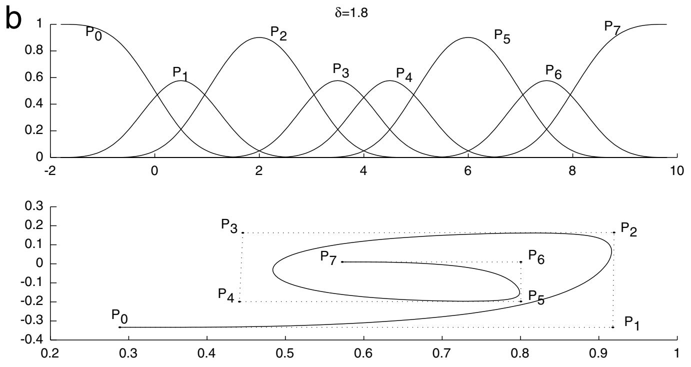


Fig. 11. Freeform curves designed using cubic PSP-spline basis functions built with nonequal spaced intervals. The upper diagram shows to which control point a spline basis function is associated with. (a) P_2 and P_5 are interpolated, in addition to the interpolation of P_0 and P_7. (b) Except for P_0 and P_7, no control points are interpolated but the curve is drawn closer to P_2 and P_5 owing to the longer intervals used to build the spline basis functions associated with the two control points.

The curve expressed in the form shown in (23) can also be used to design curves containing multiple flat parts by partially interpolating each line segment $\mathbf{P}_i + (t - t_i)\mathbf{v}_i$, which can be considered as the trajectory of an object in motion with a uniform rectilinear translation. The difference between position based interpolation and line segment based interpolation is in the shape

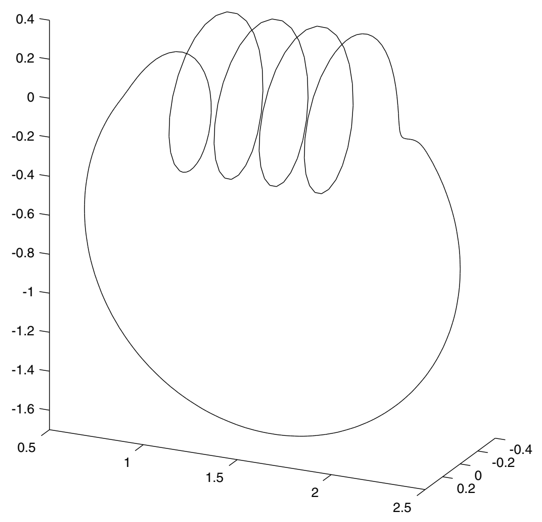


Fig. 12. A smooth curve designed by blending a helix and a circle.

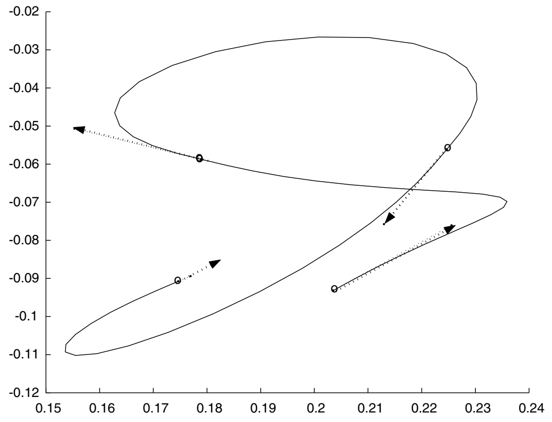


Fig. 13. Curve designed by interpolating the position and velocity data, similar to the Hermite curve design scheme.

of the spline basis function $B^{(2)}_{[a_i,a_{i+1}],\delta}(t)$ associated with $\mathbf{P}_i + (t - t_i)\mathbf{v}_i$. For line segment based interpolation, flat-topped spline basis function should be used, where the function takes value one within a subinterval of $[a_i, a_{i+1}]$.

This idea immediately leads to a simple interpolation scheme. Let $\mathbf{P}_i$, $i = 0, 1, 2, \ldots, n$, be a sequence of data points to be interpolated. For each pair of control points $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$, let $\mathcal{C}_i(t)$ be a local curve interpolating the two points at $t = t_i$ and $t = t_{i+1}$, respectively. Then a curve interpolating all the data points can be represented as a blending of these locally designed curves. One simple way to do this is to use a straight line defined by each pair of control points. For any sequence of knots t_i, $i = 0, 1, 2, \ldots, n$, when $\delta \le t_{i+1} - t_i$, the following parametric curve will interpolate all the control points:

$$\mathbf{P}(t) = \sum_{i=0}^{n} ((1 - T_i)\mathbf{P}_i + T_i\mathbf{P}_{i+1}) B^{(2)}_{[t_i,t_{i+1}],\delta}(t), \tag{24}$$

where $T_i = \frac{t-t_i}{t_{i+1}-t_i}$.

Curve shown in Fig. 14 is designed in this way.

Similar idea can also be applied to design a curve which partially interpolate the control polygon edges. Let ϵ be a positive number for specifying to what extent one would like to maintain the edge of a given control polygon as parts of the designed curve. Then the

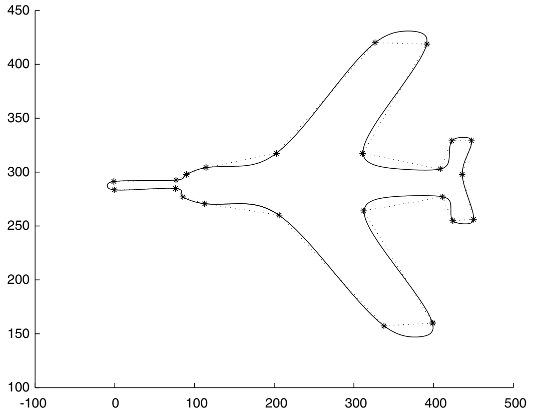


Fig. 14. A piecewise cubic curve interpolating the given data points designed following the idea expressed in (24) using equal spaced intervals.

required curve can be designed in the following way:

$$\mathbf{P}(t) = \sum_{i=0}^{n} ((1 - T_{i,\epsilon})\mathbf{P}_i + T_{i,\epsilon}\mathbf{P}_{i+1}) B^{(2)}_{[a_i,a_{i+1}],\delta}(t), \tag{25}$$

where $T_{i,\epsilon} = \frac{t-t_i+\epsilon}{t_{i+1}-t_i+2\epsilon}$.

Curves shown in Fig. 15 is designed in this way using different δ and ϵ values.

As it is commonly known that fitting a *B*-spline curve to a data set in general involves solving a linear equation system, which can be quite expensive when a very large set of data points is to be interpolated. However, with the above proposed approach, the curve fitting problem turns into a task of blending a set of locally specified curves, which is not only simple to implement and very efficient to compute, but also much more flexible.

Similar to the second order Hermite curve design, a curve can be designed to interpolate the positions, velocities and accelerations using locally defined quadratic curves. Suppose $\mathbf{P}_i, \mathbf{v}_i, \mathbf{a}_i$ are the positions, velocities and accelerations of a particle in motion corresponding to the moments t_i, $i = 0, 1, 2, \ldots, n$. Then the track of the particle can be approximated by a curve represented in the following form using cubic PSP-spline basis function $B^{(3)}_{[a_i,a_{i+1}],\delta}(t)$:

$$\mathbf{P}(t) = \sum_{i=0}^{n} \left(\mathbf{P}_i + (t - t_i)\mathbf{v}_i + \frac{1}{2}(t - t_i)^2 \mathbf{a}_i \right) B^{(3)}_{[a_i,a_{i+1}],\delta}(t). \tag{26}$$

Obviously, when $B^{(3)}_{[a_i,a_{i+1}],\delta}(t_i) = 1$, we will have

$$\mathbf{P}(t_i) = \mathbf{P}_i, \qquad \dot{\mathbf{P}}(t_i) = \mathbf{v}_i, \qquad \ddot{\mathbf{P}}(t_i) = \mathbf{a}_i.$$

This is because at $t = t_i$, $\dot{B}^{(3)}_{[a_k,a_{k+1}],\delta}(t_i) = \ddot{B}^{(3)}_{[a_k,a_{k+1}],\delta}(t_i) = 0$ when $B^{(3)}_{[a_i,a_{i+1}],\delta}(t_i) = 1$.

It is commonly known that certain quadratic curves like circles and ellipses cannot be expressed as polynomial curves precisely. However, they can be approximated quite accurately using PSP-spline curves. Fig. 16 shows visually how an ellipse can be approximated using four control points based PSP-spline curves. Only a close-up of the figure can reveal the difference between the two curves.

7. Freeform surface design using PSP-spline curves

As with conventional spline surfaces, various freeform surfaces can be generated using PSP-spline curves, such as ruled surfaces and rotated surfaces. Some example surfaces of revolution are shown in Fig. 17. The profile curves used to generate these surfaces are PSP-spline curves constructed based on the same set of control points.

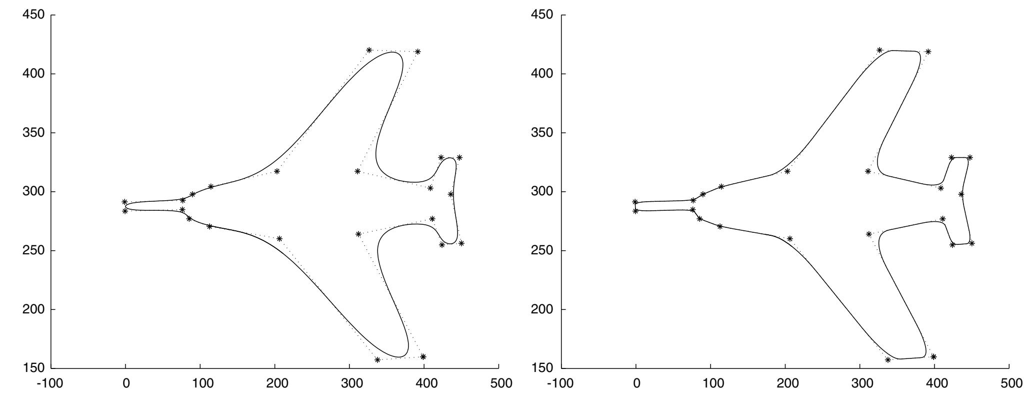


Fig. 15. PSP-spline curves designed based on partial polygon edge interpolation using different δ and ϵ values. Left: $\delta = 1$ and $\epsilon = 0.6$; Right: $\delta = 0.3$ and $\epsilon = 0.2$.

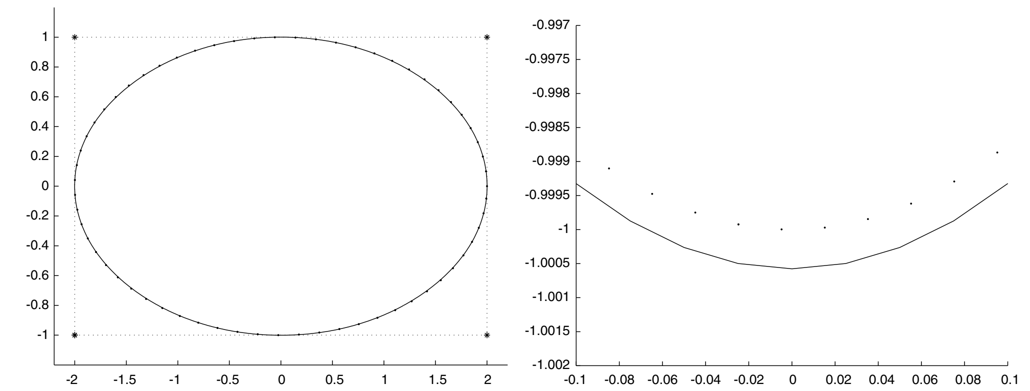


Fig. 16. The cubic PSP-spline curve designed using four control points can well approximate an ellipse with an appropriately chosen δ value. The dotted line refers to the plot of the ellipse $\frac{1}{4}x^2 + y^2 = 1$ and the solid line refers to the cubic PSP-spline curve designed using $\delta = 0.582$ with spline basis functions constructed based on the knot vector [0, 0.25, 0.5, 0.75, 1.0]. Left: the two curves are plotted with a normal view. Right: a close-up view of the two curves to reveal the difference between the two curves.

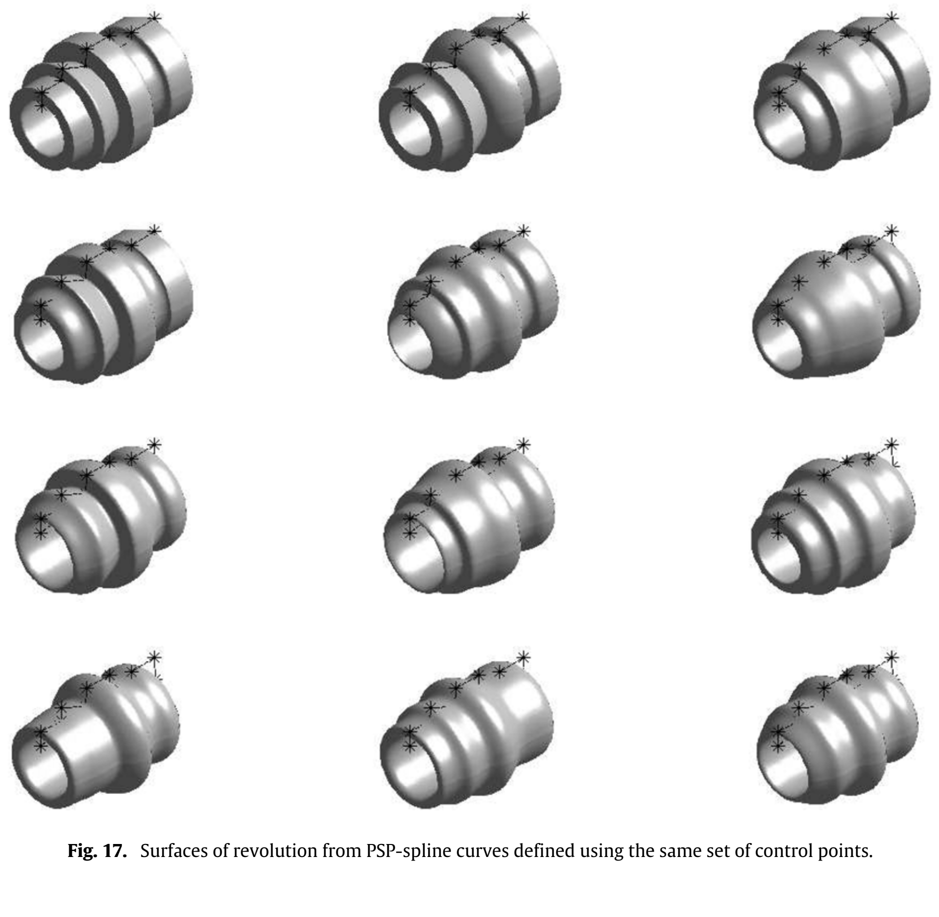


Fig. 17. Surfaces of revolution from PSP-spline curves defined using the same set of control points.

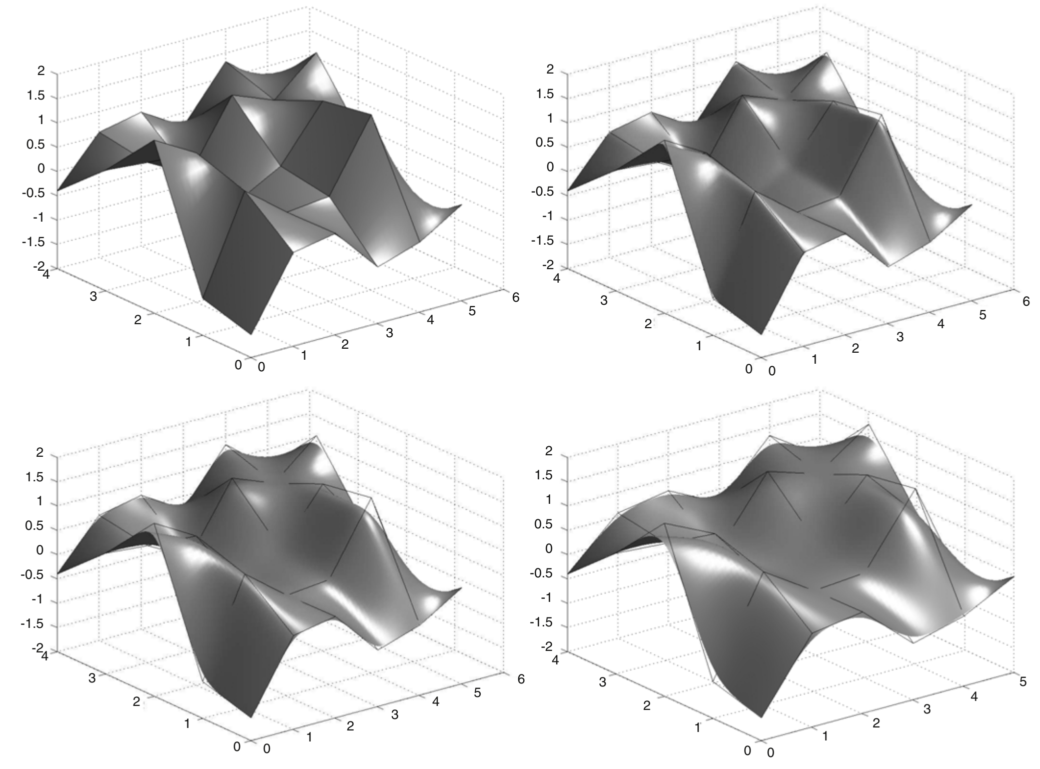


Fig. 18. Tensor-product PSP-spline surfaces designed using the same set of 6×5 control points with $\delta_1 = \delta_2 = 0.5, 0.8, 1.0, 1.2$, respectively.

Similar to the conventional tensor-product *B*-spline surface, a surface patch can also be generated as the tensor product of the PSP-spline basis functions in the following way:

$$S(u, v) = \sum_{i,j=1}^{n} \mathbf{P}_{i,j} B^{(m)}_{[a_i,a_{i+1}],\delta_1}(u) B^{(m)}_{[b_j,b_{j+1}],\delta_2}(v). \tag{27}$$

The features of tensor-product PSP-spline surfaces are illustrated in Fig. 18. Unlike conventional tensor-product *B*-spline surfaces, a set of PSP-spline surface patches can be generated corresponding to different values of shape-preserving parameters δ_1 and δ_2 in (27) and different sizes of the supports of PSP-spline basis functions.

When the PSP-spline basis functions are designed to have different sizes in their supports, a rich set of geometric shapes can be generated using a much smaller set of control points by intuitively tuning the shape-preserving parameter δ_1 and δ_2 used in (27). Fig. 19 shows some surfaces built in this way.

As has been demonstrated in Fig. 19, the key advantage of PSP-spline surfaces is its versatility and efficiency in designing freeform geometric shapes.

8. 2D PSP-spline basis functions and their application in freeform surface modeling

Tensor-product based freeform surfaces are designed using spline basis functions built from rectangular grid, which offers a simple way to build freeform surfaces when the specified control points are relatively regularly distributed. However, many data sets are quite irregular, especially those data sampled from the surfaces of real world objects. In many cases, spline basis functions corresponding to arbitrarily specified polygons may need to be constructed.

Compared with the construction of 1D spline basis functions, constructing 2D smooth piecewise polynomial spline basis functions that are having similar geometric properties to the univariate spline basis functions is an extremely tough task. In the 1D case, point is the only type of geometrical object that separates a real line into two parts, and interval is the only type of connected set that serves as the support of a univariate spline basis function. However, in higher dimension, 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 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 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 partitioning polygons. Though various techniques have been proposed to build bivariate spline basis functions, such as box and simplex splines, they are in general very expensive to evaluate, especially when evaluating a bivariate spline basis functions with high degree of smoothness [12]. Some less expensive splines have been proposed. For instance, in [13], edge based piecewise polynomial functions were introduced to build spline basis functions from an arbitrarily specified set of polygons. However, these functions are not in general additive. In addition, the corresponding shape design method does not in general possess some good geometric properties observed in traditional spline based shape design techniques. In this paper, we demonstrate how to use the bivariate spline basis functions introduced in [1] for the purpose of parametric surface design.

8.1. Bivariate PSP-spline basis functions

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 in the following way

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

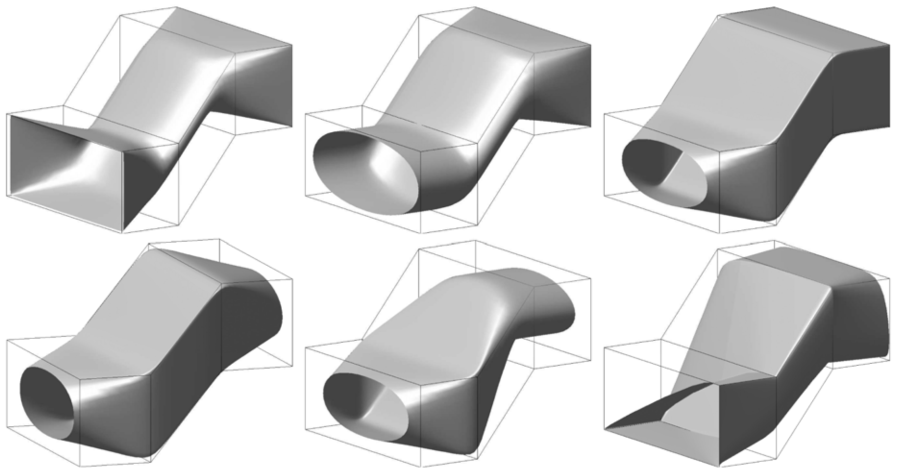


Fig. 19. Tensor-product PSP-spline surfaces designed using the same set of 4×4 control points, where the PSP-spline basis functions are of different support lengths.

$$B^{(n)}_{\Delta,\delta}(u,v) = \frac{1}{4\delta^2}\iint_{\mathcal{R}^2} B^{(n-1)}_{\Delta,\delta}(s,t)\chi_{\square}(s-u,t-v)\mathrm{d}s\mathrm{d}t, \quad (n>0), \tag{28}$$

where

$$\chi_{\square}(u,v) = \begin{cases} 1, & (u,v) \in [-\delta,\delta]\times[-\delta,\delta]; \\ 0, & \text{otherwise.} \end{cases}$$

The parameter δ in the integral plays a role similar to the one used in (11) and can be used to specify to what extent one wants the contour curve $B^{(0)}_{\Delta,\delta}(u,v) = 0.5$ to approximate the control polygon.

As have been shown in [1], the integration defined above can be expressed in an explicit form.

Let $\mathbf{v} = (\alpha,\beta), \alpha > 0, \beta > 0$, be a vector representing the orientation of an edge of a polygon. Then for any integer $n > 0$ a bivariate function expressed in piecewise polynomial can be defined as follows:

$$A^{(n)}_{\alpha,\beta}(u,v) = \begin{cases} 0, \\ \quad v \geq \min\left\{\frac{\beta}{\alpha}u, 0\right\}; \\ \frac{1}{(2n)!\alpha^n\beta^n}(\beta u - \alpha v)^{2n}, \\ \quad v < \min\left\{\frac{\beta}{\alpha}u, 0\right\}, u \leq 0; \\ \sum_{k=1}^{n} \frac{(-1)^{n+k}\alpha^k}{(n-k)!(n+k)!\beta^k} u^{n-k}v^{n+k}, \\ \quad v < \min\left\{\frac{\beta}{\alpha}u, 0\right\}, u > 0. \end{cases} \tag{29}$$

It can be shown directly that the piecewise polynomial function $A^{(n)}_{\alpha,\beta}(u,v)$ is nonnegative and C^{n-1}-continuous.

With function $A^{(n)}_{\alpha,\beta}(u,v)$, the following function can be defined for a given number $\delta > 0$:

$$\Omega^{(n)}_{\alpha,\beta,\delta}(u,v) = \frac{1}{(4\delta^2)^n}\sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{i+j}\binom{n}{i}\binom{n}{j}F_{i,j}(u,v), \tag{30}$$

where

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

As has been pointed out in [1], $\Omega^{(n)}_{\alpha,\beta,\delta}(u,v)$ is piecewise polynomial and C^{n-1}-continuous. In addition, it is also nonnegative and only takes value from the interval [0, 1].

Now for an arbitrarily given 2D polygon, a 2D field function can be directly defined using $\Omega^{(n)}_{\alpha,\beta,\delta}(u,v)$. First, for each vertex of the polygon $\mathbf{V}_0$, the following function can be defined for a nonnegative numbers α and a number β:

$$\mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_0) = \begin{cases} 0, \quad \alpha = 0; \\ H_n\left(\frac{1}{\delta}(u_0-u)\right)H_n\left(\frac{1}{\delta}(v_0-v)\right), \\ \quad \alpha > 0,\ \beta = 0; \\ \Omega^{(n)}_{\alpha,\beta,\delta}(u-u_0, v-v_0), \\ \quad \alpha > 0,\ \beta > 0; \\ \Omega^{(n)}_{\alpha,|\beta|,\delta}(-(u-u_0), v-v_0), \\ \quad \alpha > 0,\ \beta < 0. \end{cases} \tag{31}$$

Now consider two vertices $\mathbf{V}_0(u_0,v_0)$ and $\mathbf{V}_1(u_1,v_1)$ associated with an edge of the given polygon. A bivariate function can be defined using $\mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_0)$ and $\mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_1)$ introduced in (31):

$$\mathbb{L}^{(n)}_{\delta}(u,v;\mathbf{V}_0,\mathbf{V}_1) = \begin{cases} \mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_1) - \mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_0) \\ \quad v_1 > v_0 \text{ or } (v_1 = v_0, u_1 \geq u_0); \\ \mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_0) - \mathbb{V}^{(n)}_{\alpha,\beta,\delta}(u,v;\mathbf{V}_1) \\ \quad v_1 < v_0 \text{ or } (v_1 = v_0, u_1 < u_0), \end{cases} \tag{32}$$

where

$$\alpha = \begin{cases} 0, & u_1 = u_0; \\ 1, & \text{otherwise} \end{cases}, \qquad \beta = \begin{cases} 1, & u_1 = u_0; \\ \frac{v_1-v_0}{u_1-u_0}, & \text{otherwise.} \end{cases}$$

Now let $\mathbf{V}_0(u_0,v_0), \mathbf{V}_1(u_1,v_1), \ldots, \mathbf{V}_m(u_m,v_m)$ be the $m+1$ vertices of the given polygon Δ specified in counter-clockwise order. Then it can be shown that the bivariate function defined in (28) can be written explicitly in the following form:

$$B^{(n)}_{\Delta,\delta}(x) = \sum_{k=0}^{m}\operatorname{sign}(x_k - x_{k+1})\mathbb{L}^{(n)}_{\delta}(u,v;\mathbf{V}_k,\mathbf{V}_{k+1}), \tag{33}$$

where $\mathbf{V}_{m+1} = \mathbf{V}_0$ and $\mathbb{L}^{(n)}_{\delta}(u,v;\mathbf{V}_i,\mathbf{V}_j)$ is defined according to (32).

With the properties of integration it can be shown that $B^{(n)}_{\Delta,\delta}(u,v)$ has the following properties:

1. $0 \leq B^{(n)}_{\Delta,\delta}(u,v) \leq 1$.
2. $B^{(n)}_{\Delta,\delta}(u,v)$ has a C^{n-1}-continuity.
3. $B^{(n)}_{\Delta,\delta}(u,v)$ is a piecewise polynomial.
4. $B^{(n)}_{\Delta,\delta}(u,v)$ has a finite support if polygon Δ is finite.

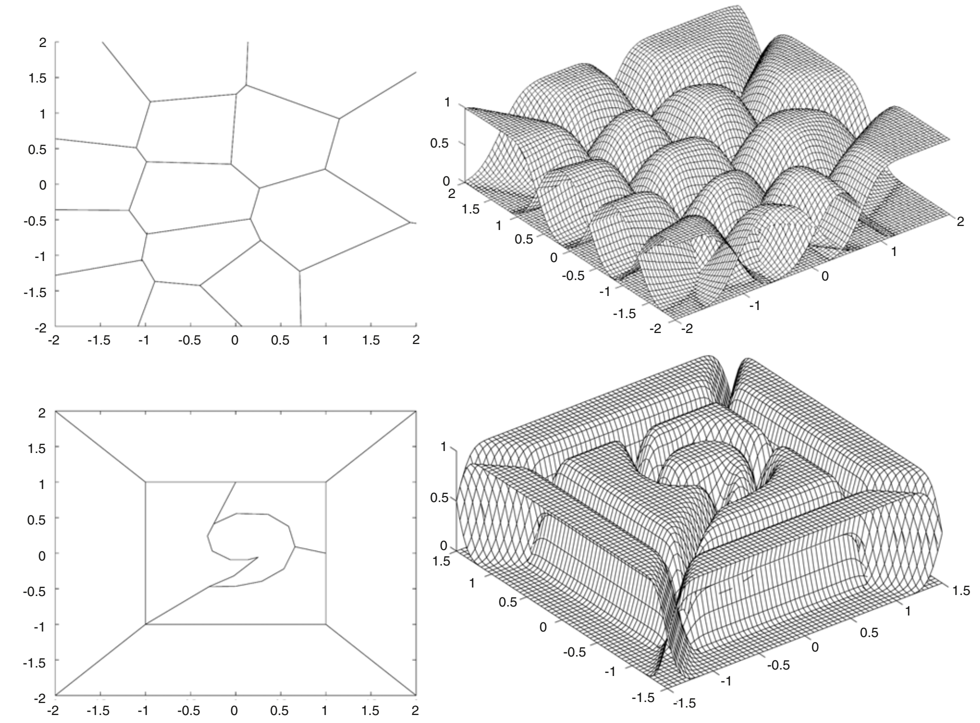


Fig. 20. Left column. A set of 2D polygons. The polygons used to construct a spline basis function can be convex (upper row) but can also be non-convex (some polygons in lower row). Right column. A set of 2D spline basis functions built upon the polygons shown on the left hand side.

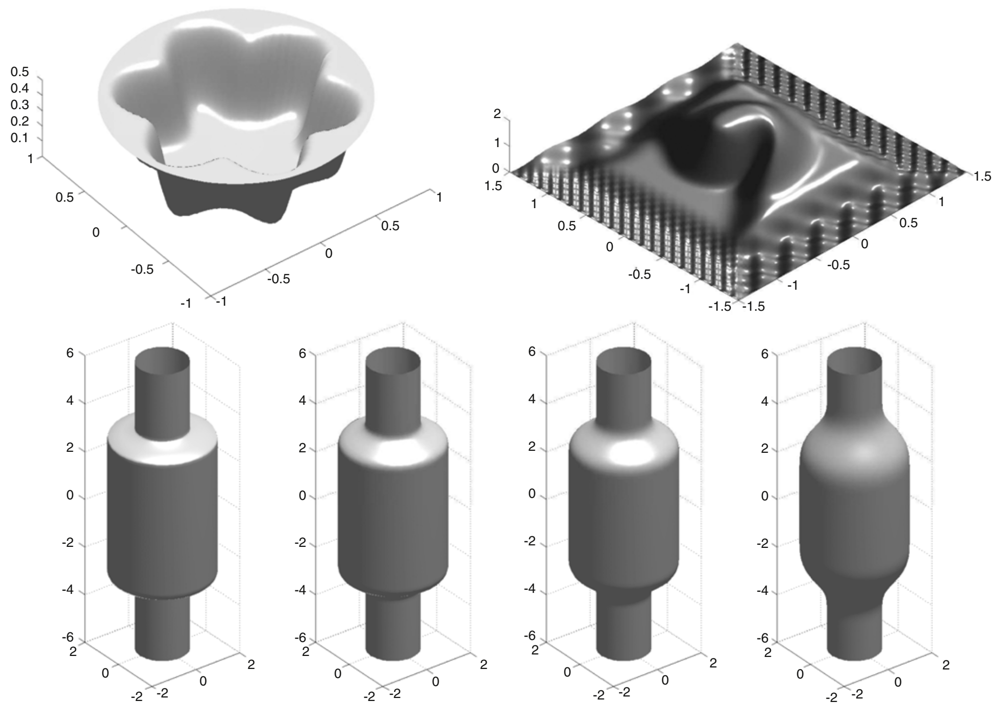


Fig. 21. Surface designed based on the idea shown in (34). Row 1: Simple geometric shapes designed by blending a set of parametric patches. Row 2: Blending range can be controlled using different values of δ in (34). The surfaces, from left to right, are generated using $\delta = 0.2, 0.5, 0.8, 1.5$, respectively.

5. $B^{(n)}_{\Delta,\delta}(u, v)$ is additive. That is, if two polygons Δ_1 and Δ_2 do not intersect or they only intersect at their edges, then

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

6. Partition of unity. If

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

then

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

Fig. 20 shows two sets of 2D PSP-spline basis functions built according to (33) from two given sets of polygons. The polygon used to construct a 2D spline basis function can either be convex or non-convex.

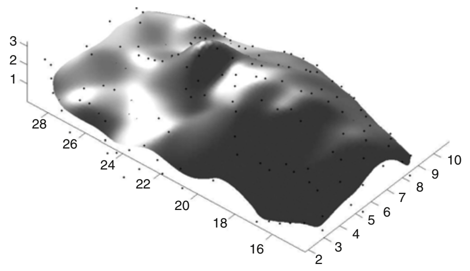


Fig. 22. PSP-spline surface constructed using a real human facial data set.

8.2. Design freeform surfaces using bivariate PSP-splines

8.2.1. Control surface patch based surface design

Suppose a required surface can be described by a set of surface patches $P_i(u, v), i = 0, 1, 2, \dots, m$, each of which is defined locally on polygons $\Delta_i, i = 0, 1, \dots, m$. Then a surface can be designed in the following way:

$$S(u, v) = \sum_{i=0}^{m} P_i(u, v) B^{(n)}_{\Delta_i,\delta}(u, v). \tag{34}$$

With this shape design method, one can first decompose a required shape into a set of locally specified simple geometric shapes over a given parametric space. For each domain of these locally defined shapes, a 2D PSP-spline basis function can be built. The required shape can then be blended directly in the way shown in (34), where δ can be used as a parameter to control the blending range of the shape composition technique. As an illustration, the surfaces presented in Fig. 21 are all generated in this way.

A special case for surfaces represented in (34) is that each $P_i(u, v)$ is just a single control point, $i = 0, 1, \dots, m$, which can be regarded as a generalization to the conventional tensor-product spline surfaces. The main advantage of bivariate PSP-splines is that it allow us to design freeform surfaces using quite irregularly distributed control points. Fig. 22 shows a PSP-spline surface

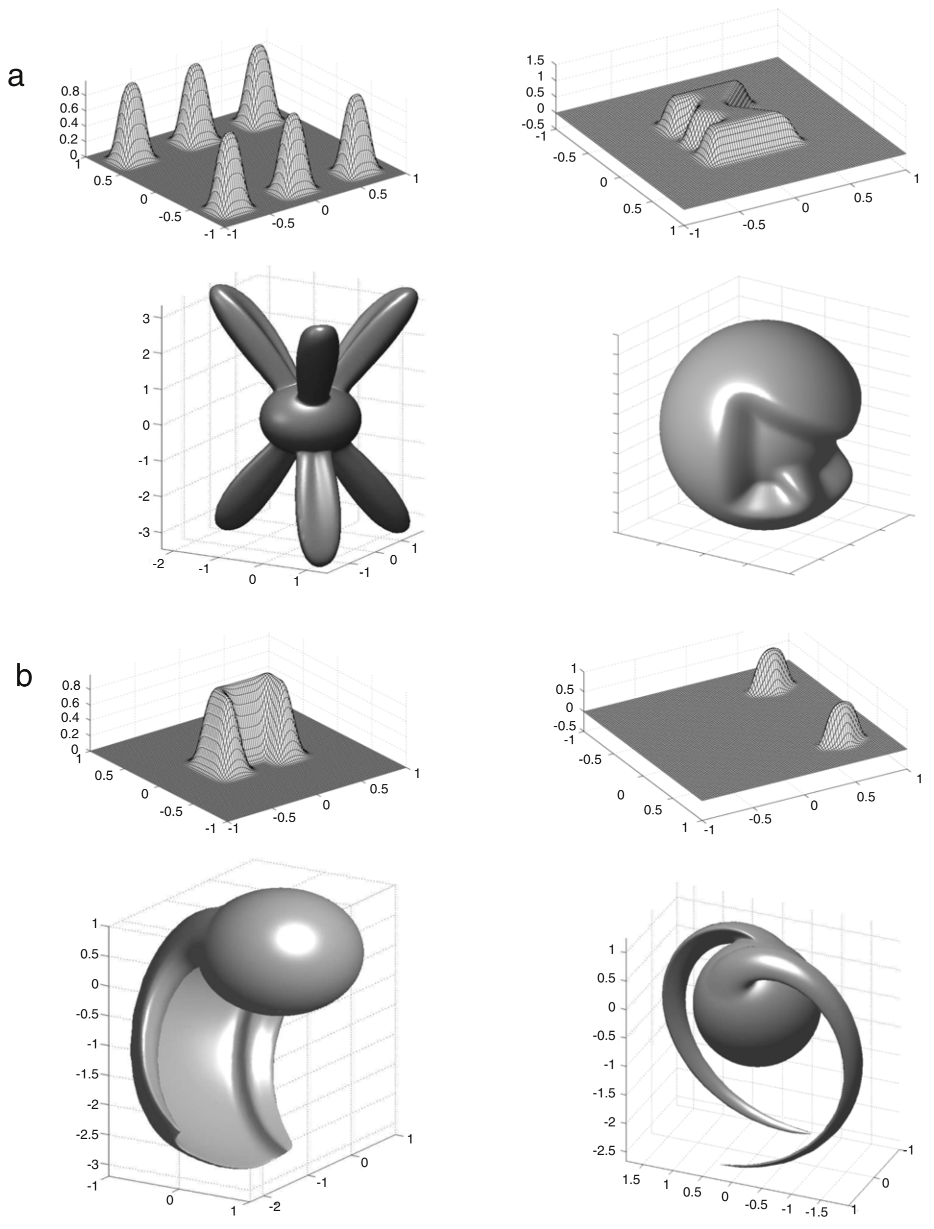


Fig. 23. Surfaces obtained by deforming a sphere using localized transformations. (a) Surfaces due to local scaling transformations constructed using the PSP-spline basis functions shown in the first row. (b) Surfaces due to local rotational transformations constructed using PSP-spline basis functions shown in the first row.

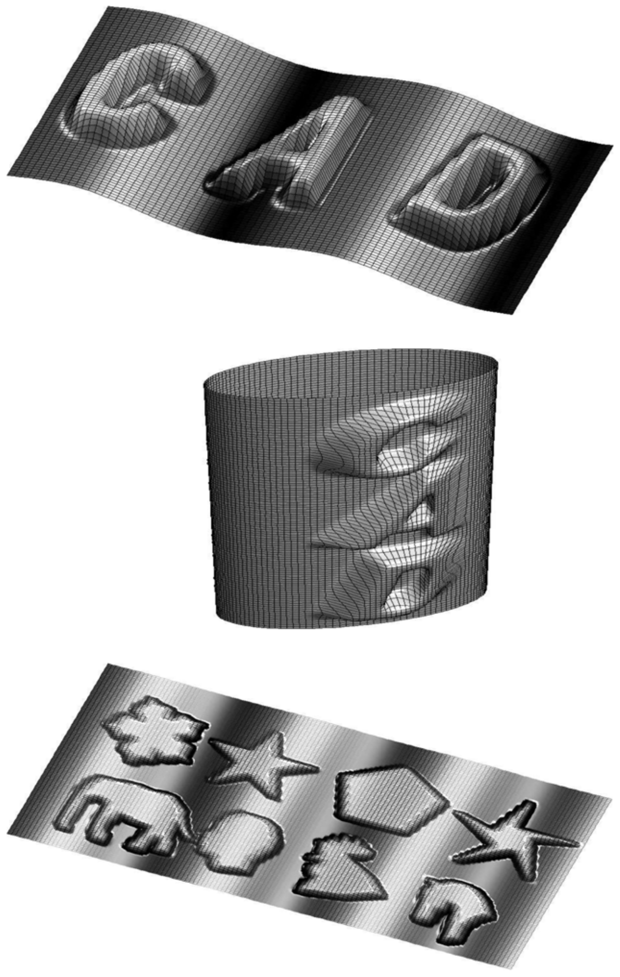


Fig. 24. Relief pattern design using 2D PSP-spline based on the idea shown in (35).

built using a set of control points sampled from a real human face.

8.2.2. Surface design using localized geometric transformations

In practice, local geometric features of a surface patch can also be specified by performing local transformations. The shape-preserving features of 2D PSP-spline basis functions can be used to control in which region a transformation is to be performed. Suppose a parametric surface is defined on parametric space $[a, b] \times [c, d]$. Let $\mathcal{T}$ be a parameter involved in a transformation, say, the rotation angle of a rotational transformation, and $B^{(n)}_{\Delta,\delta}(u, v)$ the PSP-spline basis function built on a polygon Δ, a subregion of $[a, b] \times [c, d]$, on which the transformation is to be performed. Then the transformation can be localized in the following way so that the transformation is effective only for the part of the surface corresponding to parametric region Δ:

$$\mathcal{T}_1(u, v) = \mathcal{T} \times B^{(n)}_{\Delta,\delta}(u, v) + \mathbb{I} \times (1 - B^{(n)}_{\Delta,\delta}(u, v)),$$

where $\mathbb{I}$ represents the parameter corresponding to the identity transformation. For instance, if $\mathcal{T}$ represents the rotation angle, then $\mathbb{I} = 0$.

Fig. 23 shows some surfaces generated in this way by performing localized transformations.

8.2.3. Surface editing using bivariate PSP-splines

The bivariate spline basis functions presented above is also a very useful means for shape editing. In real world, one typical feature exhibited by ordinary home and office facilities is that there are various patterns extruded from the surfaces of these objects, varying from product logos to diverse embossed ornamentations. With the introduction of the bivariate PSP-splines, the geometry of a relief pattern can be designed easily. One way to extrude the specified area of a surface is via surface blending. Suppose $S_1(u, v)$ and $S_2(u, v)$ are two surfaces, corresponding to the base surface from which a relief pattern is to be created and the out-layer surface to which the specified area on surface $S_1(u, v)$ is to be extruded. Let Δ be the polygon representing a 2D pattern and $B^{(n)}_{\Delta}(u, v)$ the corresponding bivariate spline basis function built from Δ. Then, a relief pattern designed by polygon Δ can be generated on the surface $S_1(u, v)$ in the following way:

$$S(u, v) = (1 - B^{(n)}_{\Delta}(u, v))S_1(u, v) + B^{(n)}_{\Delta}(u, v)S_2(u, v). \tag{35}$$

The geometric shapes demonstrated in Fig. 24 are all designed following this idea.

9. Summary

In this paper, a new type of spline technique is proposed based on the newly introduced PSP-spline basis functions. PSP-spline basis functions have several distinctive features to the conventional *B*-spline basis functions. For 1D spline basis functions, they are interval based. For any given interval, a degree *n* spline basis function can be built directly as the difference of two degree *n* smooth unit step functions corresponding to the two ends of the interval. Second, they can be expressed explicitly using the Heaviside unit step function. In addition, PSP-spline curve design technique has similar power to the conventional NURBS, which can weigh different control points differently by using nonequal spaced intervals to build the PSP-spline basis functions. The most important feature of the PSP-spline basis function is that it can be built to be partial shape preserving when it is used as a means to blend different premade shapes. In the case of 2D, a technique to build bivariate PSP-spline basis functions from any given set of 2D polygon net is also proposed, which can be seen as a kind of generalization of conventional tensor-product based spline basis function built from a regular polygonal net. In addition to their direct applications in designing freeform surfaces, the binary PSP-spline can be used as an effective tool for editing geometric surfaces. Compared with conventional *B*-spline shape design technique, shape design using PSP-splines is more flexible. It behaves, when the designed shape is specified as a polygon, as a kind of polygon edge smoother in the sense that a smooth curve or surface can be designed to approximate the control polygon or control polygonal mesh to any preset accuracy. As a result, a rich set of curves or surfaces can be generated from one single set of control points or a control polygonal mesh.

References

[1] Li Q, Tian J. 2D piecewise algebraic splines for implicit modeling. ACM Transactions on Graphics 2009;28(2):1–19.
[2] Farin G. Curves and surfaces for CAGD: a practical guide. Academic Press; 1997.
[3] Piegl L, Tiller W. The NURBS book. 2nd ed. New York (NY, USA): Springer-Verlag New York, Inc.; 1997.
[4] Barsky BA. Computer graphics and geometric modeling using Beta-splines. Springer-Verlag; 1988.
[5] Barsky BA, Beatty JC. Local control of bias and tension. Computer Graphics 1983;17(3):193–218.
[6] Barsky BA. Rational Beta-splines for representing curves and surfaces. IEEE Computer Graphics and Applications 1993;13(6):24–32.
[7] Tai CL, Loe KF. Alpha-spline: a c^2 continuous spline with weights and tension control. In: Shape modeling international'99. Aizu-Wakamatsu (Japan): IEEE Computer Society Press; 1999. p. 138–45.
[8] Li Q. Polygon smoothing NURBS curves and surfaces. In M. H. Hamza, (Ed.), Proceedings of the 14th IASTED international conference on applied simulation and modelling. 2005. p. 109–14.

[9] Neamtu M. What is the natural generalization of univariate splines to higher dimensions? In: Lyche T, Schumaker LL, editors. Mathematical methods for curves and surfaces. Nashville: Vanderbilt University Press; 2001. p. 355–92.
[10] Li Q, Griffiths JG, Ward J. Constructive implicit fitting. Computer Aided Geometric Design 2006;23(1):17–44.
[11] Li Q. Smooth piecewise polynomial blending operations for implicit shapes. Computer Graphics Forum 2007;26(2):157–71.
[12] Fong P, Seidel H-P. An implementation of multivariate *b*-spline surfaces over arbitrary triangulations. In: Proceedings of the conference on graphics interface'92. San Francisco (CA, USA): Morgan Kaufmann Publishers Inc.; 1992. p. 1–10.
[13] Li Q. Implicit spline curves and surfaces. In: Hu H, Zhu Z, Lang Z, editors. Proceedings of CACSCUK'05. Colchester (England): Pacilantic International Ltd.; 2005. p. 201–6