Computers & Graphics 115 (2023) 96–106

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on SMI 2023

A skinning technique for modeling artistic disk B-spline shapes

Kinga Kruppa [a,*], Roland Kunkli [a], Miklós Hoffmann [a,b]

[a] Faculty of Informatics, University of Debrecen, Kassai út 26, Debrecen, 4028, Hungary
[b] Faculty of Informatics, Eszterházy Károly Catholic University, Leányka utca 4, Eger, 3300, Hungary

ARTICLE INFO

Article history:
Received 18 May 2023
Received in revised form 15 June 2023
Accepted 25 June 2023
Available online 28 June 2023

Keywords:
Disk B-spline curve
Circle skinning
Approximation
Self-intersection
Cusps
Brushstroke representation

ABSTRACT

Disk B-spline curve provides an alternative shape modeling tool to standard control point based modeling techniques by extending control points to control disks, resulting in an easy and straightforward way of modeling regions or shapes with adjustable thickness. Due to this feature, disk B-spline curves are frequently applied in artistic shape modeling applications, such as brushstroke representation, calligraphy, and animation, but they are also suitable for engineering purposes. However, the boundary curves of the described shapes tend to have unintentional self-intersections or cusps, negatively affecting the usability and artistic value of the resulting shape and texture. In this paper, we introduce an iterative algorithm to give a solution to this problematic issue with the help of an approximating circle skinning method, while preserving the advantageous properties of the disk B-spline curves. Our method also results in smoother texturing of the shape around the more sharply curved sections.

1. Introduction

In the last couple of decades, in addition to the industry standard free-form curves defined by control points, several alternative curve representation and shape control methods have been developed in computer graphics for specific tasks.

One of these methods is the disk-type generalization of classical representation tools, such as disk Bézier curve [1] and its natural extension, the disk B-spline curve (sometimes referred to as DBSC) [2], and disk Wang–Ball curve [3]. For these curves, a traditional control point is replaced by a disk, i.e., a combination of a control point and a radius. These shape modeling tools preserve many of the good properties of the original control point based representation methods, including standard algorithms for adjusting the shape [4], or calculating intersections [5], merging [6] and degree reduction of these curves [7]. Various further generalizations of these methods, such as dynamic DBSC [8], expressive DBSC [9], or disk ball curve with shape parameters [10] have been recently developed to further enhance the flexibility of these shape modeling tools.

In fact, instead of a simple 1D curve, planar disk Bézier curves and DBSC represent a 2D region, practically a band with adjustable "thickness". This feature makes these shape modeling tools ideal for artistic applications such as brushstroke representation [2,11], calligraphy [12–14], animation [15], and modeling [16,17], but also for engineering purposes [3].

However, these disk-defined shapes tend to have unintentional self-intersections or cusps on their boundary curves at the sharper bends of the shape. These artefacts may negatively affect the usability of these shape modeling tools in the artistic applications. The aim of this paper is to present a robust technique that can be used to detect and overcome these problems, meanwhile reproducing the original shape as much as possible without self-intersections and cusps.

Our solution is based on a circle skinning technique developed in [18], and further improved in [19,20]. The fundamental idea is to pick circles from the disk B-spline shape at the knots and compute a skin of them, that is compute a pair of curves touching each circle. Of course it is possible that this preliminary skin does not closely follow the original shape in some places. Therefore, applying an iterative approach, further knots are inserted to the original disk B-spline shape, and the circles at these knots are included in the skinning method. This way the skins will better and better approximate the boundary curves of the original shape (in terms of Hausdorff distance), meanwhile—due to the specific approach of the skinning technique—keeping the boundary free of cusps and self-intersections.

Our skinning curves, like the disk B-spline shape itself, are locally defined, therefore the elimination of one cusp does not affect the shape of the model in farther parts of the curve. Our method also does not affect the interactive design of the shape. Altering the control disks of the original model, the skinning algorithm will recompute the approximating curves, still automatically detecting and eliminating the artefacts.

The main contribution of our work can be summarized as follows. We provide

* Corresponding author.
E-mail address: kruppa.kinga@inf.unideb.hu (K. Kruppa).

https://doi.org/10.1016/j.cag.2023.06.030

- a computational approach to detect the self-intersections of the boundary curve of a disk B-spline shape;
- an iterative method for approximating a disk B-spline shape with a pair of intersection-free skinning curves by appropriately selecting circles from the shape;
- an application of the new method to improve the quality of artistic textured or contoured shapes designed by disk B-spline curves.

The structure of the paper is as follows. After providing the basic definitions and notions in Section 2, we present the computation of potential self-intersections of envelope curves in Section 3. In Section 4 the method of computing the approximating skinning curves is discussed. Results are presented in Section 5, while usability, limitations, and potential extension to 3D are considered in Section 6.

2. Related work and basic definitions

Disk Bézier and disk B-spline curves have been introduced in shape modeling by Lin and Rokne [1]. The basic idea is to extend the role of the usual control point in free-form shape design by adding a thickness parameter (i.e., a positive real number) to each control point, creating together a control disk with the control point as the center, and the radius as the parameter. With these data, the shape can be defined and adjusted as a normal free-form curve, but the shape will also have a variable thickness, depending on the radii of the disks.

This tool is not the only approach of extending the usual one-dimensional free-form curve in shape design by adding some thickness of the shape. The idea of using curved regions, that is, curves with nonzero width in geometric shape modeling is originated in the notion of fat curves (for a good overview see [21]), which also had artistic applications [16,22]. The notion fat Bézier curve and wide Bézier curve [23] also appears in the literature. The idea has been further developed to spatial curves under the name of three-dimensional wide curves by Zhou and Ting [24], naturally replacing the control disks by control spheres.

The boundary curves of these disk-based splines are essentially the envelope curves of a family of circles. That way, these splines can be regarded as curves in the Minkowski space $\mathbb{R}^{2,1}$, however, the envelope curves are usually not rational. In the case of Minkowski Pythagorean hodograph (MPH) curves [25–27], the envelopes are not only rational, but also Pythagorean hodograph curves, providing many advantageous properties in computer-aided geometric design. In addition, the possibility of the use of MPH curves for modeling purposes has also recently been the subject of discussion in the literature [28,29].

One common crucial issue of these shape modeling tools is the unintentional self-intersections of the boundary curves. Meanwhile these shapes preserve many important properties of the original free-form curves, these artefacts deeply affect the usability of them, especially in artistic applications, where these self-intersections may destroy the smoothness of the texture or the homogeneity of the coloring. Zhou and Ting also studied the unintentional cusps and self-intersections of the boundary curves of the disk Bézier shapes, but with limited success and validity [23].

Creating artistic shapes, brushstroke representations, and font design are among the most important applications of these shape modeling tools from the very beginning. Several authors have applied or studied these shapes [2,11,12,30], but the existence of cusps and self-intersections has always been an issue in this field.

The aim of this paper is to overcome this problem by providing a mechanism to eliminate intersections. The core idea is to approximate the disk-based shape by a circle skinning algorithm which was introduced in [18]. In this approach the input data consist of a series of circles, and the algorithm computes a "skin", that is a pair of envelope curves touching each circle. The touching points and tangent vectors of the skin curves are computed from circle to circle by using the solution of the Apollonian problem of computing the touching circle to three given circles (for the details of the method see [18]). The method of skinning has been further developed by several authors [19,29], and it has also been extended to 3D by skinning of spheres [20,31].

For the sake of clarity, we briefly describe the basic definitions and notations of disk B-spline curves based on the work of Lin and Rokne [1].

Definition 2.1 (*Disk*)**.** A disk with center $\mathbf{c}$ and radius r is defined and denoted as

$$\langle \mathbf{c}; r\rangle = \left\{ \mathbf{x} \in \mathbb{R}^2 \;\middle|\; \|\mathbf{x}-\mathbf{c}\| \le r, \mathbf{c} \in \mathbb{R}^2, r \in \mathbb{R}^+ \right\}. \tag{1}$$

For a given disk $\langle \mathbf{c}, r\rangle$, let us define addition and multiplication as follows:

- $a\langle \mathbf{c}; r\rangle = \langle a\mathbf{c}; |a|r\rangle$, where $a \in \mathbb{R}$;
- $\langle \mathbf{c}_1; r_1\rangle + \langle \mathbf{c}_2; r_2\rangle = \langle \mathbf{c}_1 + \mathbf{c}_2; r_1 + r_2\rangle$.

A disk B-spline curve of degree p can be defined in the following way.

Definition 2.2 (*Disk B-Spline Curve*)**.** Given $n+1$ points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ and radii $r_0, r_1, \ldots, r_n$, a disk B-spline curve (DBSC) of degree p with knot vector $U = \{u_0, u_1, \ldots, u_m\}$, $m = n+p+1$, is defined and denoted as follows:

$$\begin{aligned}\langle D\rangle(t) &= \sum_{i=0}^{n} N_{i,p}(t)\,\langle \mathbf{p}_i; r_i\rangle \\ &= \left\langle \sum_{i=0}^{n} N_{i,p}(t)\mathbf{p}_i;\ \sum_{i=0}^{n} N_{i,p}(t) r_i \right\rangle,\end{aligned} \tag{2}$$

where $N_{i,p}(t)$ is the ith B-spline basis function of degree p.

Hereafter, we denote the spine curve of a DBSC by $\mathbf{q}(t) = \sum_{i=0}^{n} N_{i,p}(t)\mathbf{p}_i$ and the radius function by $r(t) = \sum_{i=0}^{n} N_{i,p}(t) r_i$. It follows from Definition 2.2 that a DBSC describes a family of circles for which the boundary curves can be determined and computed. If the family of circles is written in an implicit form of $F(x, y, t)$ with family parameter t, one can obtain the envelope by solving the usual system of equations

$$\begin{cases} F(x,y,t) = 0, \\ \dfrac{\partial F}{\partial t}(x,y,t) = 0. \end{cases} \tag{3}$$

This envelope is the boundary of the region represented by the DBSC. Also, $\langle D\rangle(t)$ can be regarded as a medial axis transform as

$$\overline{\mathbf{q}}\colon [a,b] \to \mathbb{R}^{2,1}, \quad \overline{\mathbf{q}}(t) = (\mathbf{q}(t), r(t)) = (x(t), y(t), r(t)). \tag{4}$$

Then, by applying the envelope formula given by Choi et al. [26], we can acquire the envelope curves $\mathbf{q}^{\pm}\colon [a,b] \to \mathbb{R}^2$ as:

$$\mathbf{q}^{\pm}(t) = \mathbf{q}(t) - r(t)\frac{r'(t)\,\mathbf{q}'(t) \pm \mathbf{q}'^{\perp}(t)\sqrt{\|\mathbf{q}'(t)\|^2 - r'(t)^2}}{\|\mathbf{q}'(t)\|^2}, \tag{5}$$

where $\mathbf{q}'^{\perp}(t)$ denotes the vector which is obtained by rotating $\mathbf{q}'(t)$ counter-clockwise. Eq. (5) can be rewritten in the form

$$\mathbf{q}^{\pm} = \mathbf{p}(t) + r(t)\frac{\mathbf{q}'^{\perp}(t)}{\|\mathbf{q}'^{\perp}(t)\|}, \tag{6}$$

where

$$\mathbf{p}(t) = \mathbf{q}(t) \pm \frac{\mathbf{q}'(t)}{\|\mathbf{q}'(t)\|}\frac{r(t)r'(t)}{\|\mathbf{q}'(t)\|}. \tag{7}$$

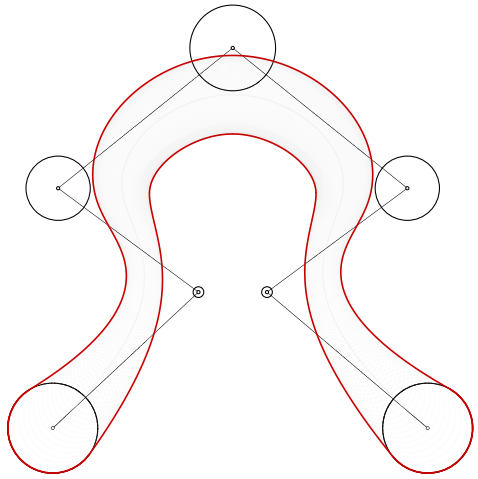

Fig. 1. A clamped disk B-spline curve of degree 3 with its boundary curves.

As a consequence, we can regard the envelope curves as special functional offsets of the spine curve $\mathbf{q}(t)$. Fig. 1 shows a DBSC with its envelope curves.

3. Self-intersections of the DBSC

Disk B-spline curves together with their boundary curves are widely used for modeling purposes as they have several advantageous properties. As we have shown in Section 1, they are particularly suitable for representing regions in an artistic sense, as well as for displaying textured shapes or, for example, brushstrokes. However, after creating the DBSC and its envelope curves, unintentional self-intersections or cusps might appear at the sharper bends of the shape.

In this section, we show how one can detect self-intersections of the disk B-spline curves, specifically, cusps on the curve. Cusps appear on the envelope curves when there is a discontinuity in the tangent vector [32], which results in a sudden reverse of the tangent vector along the curve. Fig. 2 shows such an example when the tangent vector of the envelope curve flips its direction, and two cusps are formed. These cusps can either be located by finding the roots of $\mathbf{q}^{\pm\prime}(t) = 0$ or by using a more robust method introduced by Elber and Cohen [33], who described such cusps (with regard to offsets) as local loops. We will use this method to detect these self-intersections. Let $\mathbf{T}(t)$ and $\mathcal{T}^{\pm}(t)$ denote the tangent vectors at $\mathbf{q}(t)$ and at $\mathbf{q}^{\pm}(t)$, respectively. One can locate the cusps on the envelope by finding the zero set of

$$\tau^{\pm}(t) = \mathcal{T}^{\pm}(t) \cdot \mathbf{T}(t). \tag{8}$$

Then the local loop is bounded by two cusps, for which region the condition

$$\tau^{\pm}(t) < 0 \tag{9}$$

holds. For further details and proofs, see [33]. Using this idea, if we would like to ensure that no cusps appear on the envelope curves,

$$\mathbf{q}^{\pm\prime}(t) \cdot \mathbf{q}'(t) > 0, \quad \forall t \in [a, b] \tag{10}$$

has to be satisfied. If we evaluate Eq. (10), we can further reduce it to

$$\left\|\overline{\mathbf{q}}'(t)\right\|_M^2 + \frac{r(t)\,\zeta(t)}{\|\mathbf{q}'(t)\|^2} > 0, \quad \forall t \in [a, b], \tag{11}$$

where

$$\zeta(t) = \mathbf{q}'(t) \cdot \mathbf{q}''(t)\, r'(t) - \left\|\mathbf{q}'(t)\right\|^2 r''(t) \pm \left\|\overline{\mathbf{q}}'(t)\right\|_M \left(\mathbf{q}'(t) \times \mathbf{q}''(t)\right)_z, \tag{12}$$

and

$$\left\|\overline{\mathbf{q}}'(t)\right\|_M = \sqrt{\|\mathbf{q}'(t)\|^2 - r'(t)^2} \tag{13}$$

is the Minkowski norm of $\overline{\mathbf{q}}$, and $\mathbf{a}_z$ denotes the third coordinate of an arbitrary vector $\mathbf{a} \in \mathbb{R}^3$. Since $\left\|\mathbf{q}'(t)\right\|^2 > 0$, $\forall t \in [a, b]$, Eq. (11) can be written in the form

$$\left\|\mathbf{q}'(t)\right\|^2 \left\|\overline{\mathbf{q}}'(t)\right\|_M^2 + r(t)\,\zeta(t) > 0. \tag{14}$$

In Eq. (14), both $\left\|\mathbf{q}'(t)\right\|^2 > 0$ and $\left\|\overline{\mathbf{q}}'(t)\right\|_M > 0$ holds if real envelopes exist, thus, Lemma 1 shows a general condition on the cusps of envelope curves.

Lemma 1. *Given the medial axis transform* $\overline{\mathbf{q}}\colon [a, b] \to \mathbb{R}^{2,1}$, $\overline{\mathbf{q}}(t) = (\mathbf{q}(t), r(t))$, *let us suppose that* $\mathbf{q}(t)$ *is regular and real envelope curves exist. Then its associated envelope curves* $\mathbf{q}^{+}(t)$ *and* $\mathbf{q}^{-}(t)$ *do not contain cusps if*

$$\left\|\overline{\mathbf{q}}'(t)\right\|_M + r(t)\,\Gamma(t) > 0, \quad \forall t \in [a, b], \tag{15}$$

where

$$\Gamma(t) = \frac{\mathbf{q}'(t) \cdot \mathbf{q}''(t)}{\|\mathbf{q}'(t)\|^2 \left\|\overline{\mathbf{q}}'(t)\right\|_M} r'(t) - r''(t) \pm \left(\mathbf{q}'(t) \times \mathbf{q}''(t)\right)_z. \tag{16}$$

A special case arises if the radius function of the medial axis transform is constant, i.e., $r(t) = r$. In this case, the derivatives of $r(t)$ disappear, thus, the envelope curves $\mathbf{q}^{\pm}(t)$ in forms (6) and (7) will be offsets of $\mathbf{q}(t)$ with distance r. This greatly simplifies the detection of cusps [33]. These points of singularities, therefore self-intersecting loops, can be detected when

$$\frac{1}{\kappa^{\pm}(t)} < r, \tag{17}$$

where $1/\kappa^{\pm}(t)$ denotes the radius of curvature of the envelope curves $\mathbf{q}^{+}(t)$ and $\mathbf{q}^{-}(t)$, respectively.

If we strive for a shape without self-intersections, a trimming process can be applied after detecting these loops: the part of the curve between the cusps should be removed, and the intersection point of the remaining two parts of the curve has to be calculated. To find the intersection point, numerical methods can be used, such as general curve–curve intersection techniques based on implicitization or clipping (however, clipping methods are most often applied after approximating the envelopes in Bernstein–Bézier form) [34–36]. After trimming, the envelope curves no longer contain intersecting loops, but this process can be quite costly. Also, it would result in sharp turnarounds, so the results may differ from what the modeler envisioned, both in terms of the final shape and in the textured result.

In the case of DBSCs, another solution to obtain an intersection-free result would be restricting the position or the radii of the control circles. However, this would result in a different shape than the intended one, as we can see in Fig. 3. Furthermore, the exact calculations of how one could modify the positions and/or the radii would be a difficult question of numerical optimization.

As we have discussed, such intersecting loops can occur during the modeling process with a DBSC, even though the modeler would not expect cusps and self-intersections. We may attempt to construct envelope curves without such singularities, but it would be exceedingly difficult to always guarantee that the conditions (15) and (16) are satisfied for all parameter values. In this sense, the modeling process with disk B-spline curves can be troublesome. As a solution, instead of using envelope curves, we suggest computing skinning curves for well-chosen circles of the DBSC, which will approximate the desired shape well without self-intersections.

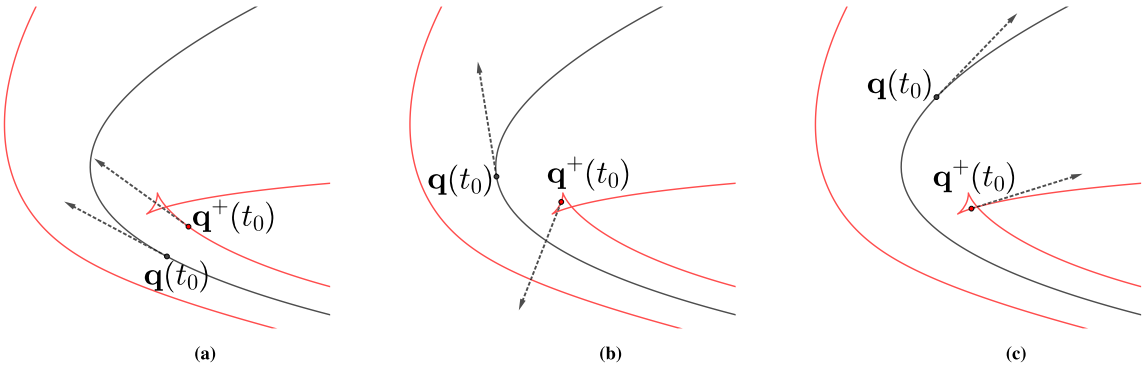

Fig. 2. Two cusps and a loop appear on the envelope curve when its tangent vector flips its direction.

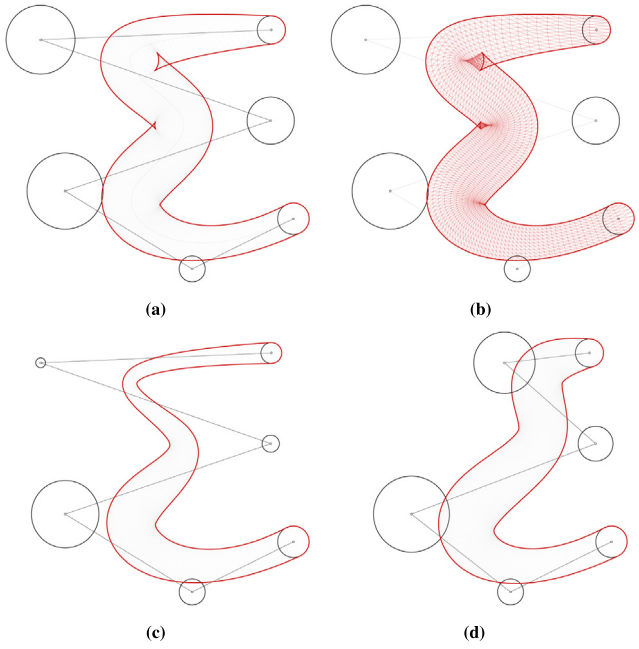

Fig. 3. DBSC shapes with inadvertent self-intersections or cusps on their boundary curves will also yield intersections in the tessellated region (see (a) and (b)). Although we can get rid of the intersections by modifying the positions and/or radii of the control circles, the resulting shape may significantly differ from the intended one (as shown in (c) and (d)).

4. Approximation of DBSC with skinning curves

Although DBSCs have proven to be an efficient modeling technique, they also suffer from some drawbacks as we showed in Section 3. The appearance of cusps and self-intersecting loops on the boundary curves is not only visually disturbing but can cause further problems if the created model as a region is to be textured. Fig. 3(b) shows an example of this problem.

In this section, we present a solution that accurately approximates the desired shape described by a DBSC—but without cusps and self-intersecting loops. The idea is simple, yet effective: we select appropriate circles from the family of circles defined by the DBSC, on which we apply a modified skinning technique to obtain two splines that bound the sequence of circles. Each of these splines will approximate the left and right envelope curves with a given precision. The method that we present consists of the following parts:

- Initialization: defining circles based on the knot values of DBSC.
- Phase 1: an iterative process of skinning the circles and inserting new ones, ensuring a balanced shape. If the resulting shape does not approximate well the original shape and no more insertion is possible, then we proceed to the second phase.
- Phase 2: if the approximation is not good enough, we iteratively insert knot values, but without checking for any further symmetry. When no more insertion is possible, we proceed to the third phase.
- Phase 3: if there are skinning segments where the approximation error is large for both the left and the right sides, we use "two-sided skinning" for a final refinement of the shape.

Throughout the rest of this study, the control disks and polylines will be omitted from some of the figures to give a better view of the actual shapes.

4.1. Defining circles on the DBSC

Let us consider a disk B-spline curve $\langle D \rangle(t)$ of degree p with $n+1$ control points $\mathbf{p}_i$ and radii r_i. Let $m+1$ be the size of knot vector U where $m = n+p+1$, i.e., $U = \{u_0, u_1, \ldots, u_m\}$. By control disks, we refer to the disks

$$\langle \mathbf{p}_i; r_i \rangle, \quad i = 0, 1, \ldots, n. \tag{18}$$

In what follows, we denote circle c with center $\mathbf{p}$ and radius r as $c = \langle \mathbf{p}; r \rangle_C$ by directly applying Definition 2.1. For our approximation algorithm, the aim is to define the sequence of circles $\mathcal{C} = \{c_0, c_1, \ldots, c_n\}$ which is an admissible input for skinning.

Definition 4.1 (*Admissible Configuration*)**.** Let us consider a sequence of circles $\mathcal{C} = \{c_0, c_1, \ldots, c_n\}$ and their corresponding disks $\mathcal{D} = \{d_0, d_1, \ldots, d_n\}$. We call $\mathcal{C}$ an admissible input, if the following criteria are satisfied:

1. $d_i \not\subset \bigcup_{j=0, j\neq i}^{n} d_j, \quad i = 0, 1, \ldots, n;$
2. if $d_{i-1} \cap d_{i+1} \neq \emptyset$, then $d_{i-1} \cap d_{i+1} \subset d_i, \quad i = 2, 3, \ldots, n-1;$
3. if $\mathbf{s}_i$ denotes the radical point of circles c_{i-1}, c_i, and c_{i+1}, then $\mathbf{s}_i \notin \bigcup_{j=i-1}^{i+1} d_j$, $i = 1, 2, \ldots, n-1$.

We have modified the admission criteria defined by Kunkli and Hoffmann [18] as we did not include the condition $d_i \cap d_j = \emptyset$ $(i, j = 1, 2, \ldots, n,\ j \notin \{i-2, i-1, i, i+1, i+2\})$ in Definition 4.1. In order to obtain an accurate approximation, it is sometimes necessary to use many more circles than it is usual in traditional skinning approaches, and some of these circles may overlap, causing this condition to fail. Therefore, we have chosen only those criteria that are indeed necessary to find touching points on the given circles.

The initialization step of the method is selecting those circles that correspond to the knots in the knot vector U. That is, we populate $\mathcal{C}$ as follows:

$$\mathcal{C} = \left\{ \langle \mathbf{q}(u_i); r(u_i) \rangle_C \;\middle|\; u_i \in U, \; i = p, p+1, \ldots, m-p \right\} \tag{19}$$

For this initial step, $\mathcal{C}$ is required to be an admissible configuration. In the following, we refer to the resulting circles in $\mathcal{C}$ as knot circles.

4.2. Approximation and iterative error measurement

Once the initial configuration is obtained, we choose a skinning technique [18–20] with which we compute the splines $\mathbf{s}^+(t)$ and $\mathbf{s}^-(t)$ as the so-called left and right skins. In the present work, the figures were created by applying the method of Kunkli and Hoffmann [18] to $\mathcal{C}$. In this way, we acquired the left and right touching points on each knot circle c_i, and constructed the curves as G^1 continuous cubic Bézier splines.

After we have skinned the circles of $\mathcal{C}$, we need to measure the accuracy of the approximation by calculating the Hausdorff distance on both the left and right envelope and skinning curve pairs, respectively. After the initial selection of the circles in $\mathcal{C}$, the skins will evidently differ from the envelope curves. However, depending on the shape being defined, some parts would differ more than others. Since the left and right skinning curves are both splines constructed from Bézier curves between the two knot circles c_i and c_{i+1} ($i = 0, 1, \ldots, n-1$), we compare them to their corresponding parts of the envelope curves. Thus, we divide the envelope curves into segments at their knot values. The envelope curves $\mathbf{q}^\pm$ are then considered as splines of $\mathbf{q}_i^\pm(u)$, $u \in \left[u_{p+i}, u_{p+i+1}\right]$, $i = \{0, 1, \ldots, n-p\}$ segments. To measure the error of the approximating skinning curves, we need to evaluate

$$D_{H_i}^\pm = D_{H_i}\left(\mathbf{q}_i^\pm(u), \mathbf{s}_i^\pm(t)\right) \quad u \in \left[u_{p+i}, u_{p+i+1}\right], \; t \in [0,1], \; i = \{0, 1, \ldots, n-p\}, \tag{20}$$

where $D_H(\mathbf{C}_1, \mathbf{C}_2)$ denotes the Hausdorff distance between two given curves $\mathbf{C}_1$ and $\mathbf{C}_2$ defined as

$$D_H(\mathbf{C}_1, \mathbf{C}_2) = \max\left\{ \max_{\mathbf{p}\in\mathbf{C}_1} \min_{\mathbf{q}\in\mathbf{C}_2} \|\mathbf{p}-\mathbf{q}\|, \; \max_{\mathbf{q}\in\mathbf{C}_2} \min_{\mathbf{p}\in\mathbf{C}_1} \|\mathbf{p}-\mathbf{q}\| \right\}. \tag{21}$$

The computation of the Hausdorff distance between two parametric curves is not a straightforward problem and has been the subject of several papers. There are existing methods for estimation and exact calculation of the distance as well [37–40]. After obtaining the $D_{H_i}^\pm$ values, we decide whether the error of the ith skinning segment is within a tolerance ϵ. If not, we need to refine the approximation to follow the shape of the envelope curves more closely. As it was proved by Kruppa et al. [20], the skinning curves locally converge to the envelope curves. Therefore, if we choose more circles while keeping the admissibility criteria intact, then we can get the shape approximated within the chosen tolerance.

The idea is to check all the curve segments $\mathbf{s}_i^\pm$, and if they do not approximate $\mathbf{q}_i^\pm(u)$ well, i.e., $H_i^\pm > \epsilon$, then we need to choose a new circle from the family of circles described by the DBSC to reduce the error. For this purpose, we use knot insertion on the DBSC to easily keep track of the envelope and skin segments and the approximation error. The most crucial question is which parameter value should be chosen as a new knot. When choosing the new knot circle, we have to keep in mind that $\mathcal{C}$ should remain an admissible configuration according to Definition 4.1. Let us assume that the ith segment of the left skin is not yet approximating the ith segment of the left envelope well, so that $D_{H_i}^+ > \epsilon$ holds. An obvious choice is to insert a new knot at the middle of the ith knot span, that is, at

$$u^* = \frac{u_{p+i} + u_{p+i+1}}{2}. \tag{22}$$

This results in splitting the segment in two and in inserting a new knot circle into $\mathcal{C}$. If the modified sequence of circles is still admissible, then the resulting skinning curve would have less error than before. This approach behaves very well in general scenarios, but it can cause problems when the envelope curves have cusps and self-intersecting loops. Since obtaining shapes without these singularities is the main reason why we would like to use skinning curves instead of the actual envelope, we need to find a solution to these issues.

For a given DBSC with knot circles $\mathcal{C}$ already determined, let us consider the following example. Looking at the error estimates, we would like to insert a knot at the ith curve segment, i.e., at the $(p+i)$th knot span between circles c_i and c_{i+1}. Let us assume that the insertion can be completed successfully because the knot circles form an admissible input, which we now denote as $\hat{\mathcal{C}}$. After the insertion, one of the skins may become distorted and quite asymmetric, since the touching point at $c_{i+1} = \hat{c}_{i+2}$ would move closer to $c_{i+2} = \hat{c}_{i+3}$. To ensure a more symmetrical shape, we would like to insert a new knot circle between $\hat{c}_{i+2}$ and $\hat{c}_{i+3}$, but we are unable to do so because this would result in a non-admissible configuration: the radical point of the circle to be inserted and its left and right neighbors would be inside the circles. Subsequently, it is not possible to complete the insertion, and the result obtained so far is still asymmetric, which is usually not a desirable outcome. Fig. 4 shows an example of this scenario, where the insertion of the knot circle is not possible, yielding an unsatisfactory result seen in Figs. 4(c) and 4(d). We can overcome these problems by requiring a condition to be fulfilled upon insertion regarding the neighboring circles.

Specifically, if we would like to insert knot u^* into the $(p+i)$th knot span, that is, between c_i and c_{i+1}, we will not only check the admissibility of circle $\langle \mathbf{q}(u^*); r(u^*) \rangle_C$, but also its symmetric pair $\langle \mathbf{q}(u^{**}); r(u^{**}) \rangle_C$ as follows. For this purpose, we create the modified knot vector $\hat{U}$ as

$$\hat{U} = \left\{ u_0, u_1, \ldots, u_{p+i}, u^*, u_{p+i+1}, u^{**}, u_{p+i+2}, \ldots, u_m \right\}, \tag{23}$$

where

$$u^* = \frac{u_{p+i} + u_{p+i+1}}{2} \quad \text{and} \quad u^{**} = \frac{u_{p+i+1} + u_{p+i+2}}{2}. \tag{24}$$

Consequently, we create $\hat{\mathcal{C}} = \left\{ c_0, c_1, \ldots, c_i, c^*, c_{i+1}, c^{**}, c_{i+2}, c_{i+3}, \ldots, c_{m-p} \right\}$ where c^* and c^{**} are the knot circles corresponding to u^* and u^{**}, respectively. If $\hat{\mathcal{C}}$ is an admissible configuration, then we can proceed with the insertion. If not, we need to modify the values of u^* and u^{**} appropriately. For the sake of a more symmetrical shape, we modify them in pair as follows. Because we wish to push c^* and c^{**} towards c_i and c_{i+1}, respectively, we introduce weights $W = \{w_0, w_1, \ldots, w_{m-1}\}$ so that the knot values u^* and u^{**} are chosen as:

$$\begin{aligned} u^* &= (1 - w_{p+i})\, u_{p+i} + w_{p+i}\, u_{p+i+1}, \\ u^{**} &= w_{p+i+1}\, u_{p+i+1} + (1 - w_{p+i+1})\, u_{p+i+2}. \end{aligned} \tag{25}$$

The values of w_i and w_{i+1} would be iteratively modified to $w_j = w_i/2$ $(j = p+i, p+i+1)$, if the insertion is still not possible. This modification of the weights can be done until a solution is found or until u^* or u^{**} is in a too close proximity to u_{p+i} or u_{p+i+1}, respectively. In this latter case, the process is aborted with no possible insertion.

We can take a look at our previous example of Fig. 4, starting with the initial shape in Fig. 4(b). Before inserting a circle at knot

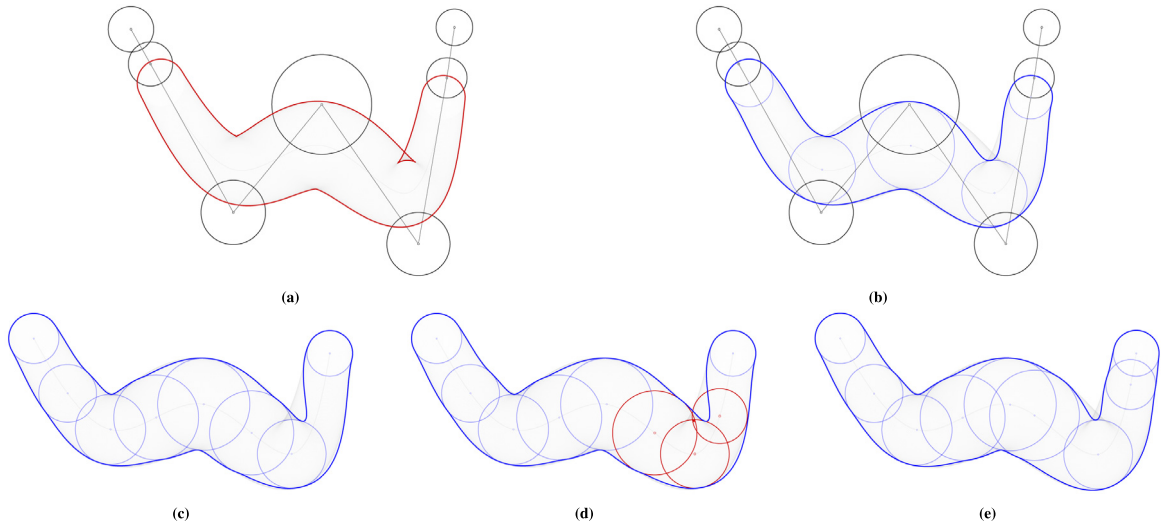

Fig. 4. The first phase of our DBSC shape approximation algorithm. (a) The shape created with an open disk B-spline curve with a visible cusp and self-intersection. (b) Initial configuration of the knot circles for our DBSC-based skinning method. (c) The first iteration of the insertions results in an asymmetric shape. (d) Insertion at the last segment is not possible due to the position of the radical point. (e) The result of the insertion after introducing weights.

span $p+i$, we check whether it would be also possible to insert a knot circle at the next span too. In this example, it is not possible (shown with red in Fig. 4(d)), so we fine-tune the weights of the insertion. Fig. 4(e) shows the result, where we can see that we had to modify the weights of the insertion to obtain proper knot circles.

It is also important to note that the knot does not necessarily have to be inserted at the $(i+1)$th span, we are only checking the possibility of the insertion. If the $(i+1)$th segment also needs refinement, then evidently the modified value will be used to insert the knot circle.

While we iterate through the curve segments to reduce the approximation error, as we discussed earlier, we do not have to insert a knot exactly at the middle of the knot span, but at a different ratio. For example, it is possible that we modified the weights of w_{p+i} and w_{p+i+1} so that we could insert a knot circle at the ith curve segment. Let us assume that the next curve segment also needs refinement. So we need to check knot circles with weights w_{p+i+1} and w_{p+i+2} and their respective u^* and u^{**} values. If the resulting circle sequence is not admissible, we have to modify these weights. However, if we changed w_{p+i+1} and consequently pushed the knot circle closer to c_i, then it would be possible to break the admissibility of the previous insertion. So we need to push w_{p+i} backwards as well to obtain an admissible sequence, if it is possible. Therefore modifying a weight in an insertion can affect previously selected weights and knot circles as well.

If inserting a new knot circle is not possible for a given curve segment because it would result in a non-admissible sequence of knot circles, then we mark this segment so as not to try to insert on it again. After we checked all curve segments $\mathbf{s}_i^{\pm}$ whether the approximation is good enough, i.e., $H_i^{\pm} < \epsilon$, and if needed, we inserted new knot circles, we can start a new iteration of the knot insertion method. This iterative process will halt if one of the following conditions is true for all curve segments $\mathbf{s}_i^{\pm}$, $i = \{0, 1, \ldots, n-p\}$:

- $D_{H_i}^{\pm} < \epsilon$;
- new knot value u^* cannot be inserted.

The following steps summarize the first phase of our method:

1. Initialize $\mathcal{C}$ with knot circles created for each knot value of the DBSC.
2. Compute the skinning curves for $\mathcal{C}$.
3. For each segment:
   - If $D_H^{\pm} < \epsilon$, mark the segment as FINISHED.
   - Otherwise, try to insert a new knot value:
     - Determine values for u^* and u^{**} to ensure a balanced shape.
     - If the input is admissible, insert the new knot value u^* and store u^{**}. Otherwise, the segment is marked as STOPPED.
4. Loop through these steps up until all segments are FINISHED or STOPPED.

After the possible iterations of the insertions, we reach the second phase of our algorithm, where we solve a problematic issue that might arise. An example is shown in Fig. 5. Starting from the initial shape (see Fig. 5(b)), the insertion algorithm halts in the state shown in Fig. 5(c). We can see that the shape is not approximating the envelope well enough, but no more knot circles can be inserted. In this example, the first curve segment is problematic: we cannot insert a circle because of its neighboring segment, regardless of the weights used. This happens simply because the next segment is not the region where the symmetry should be checked, and actually, in this phase of the algorithm the asymmetric shape of turnarounds are already taken care of. Thus, after the first phase of insertions, if there are still problematic segments, we insert the knot value u^* if the resulting circle sequence would be an admissible configuration—without any further examination of the neighboring circles. This second phase of the algorithm is being iterated until the halting conditions are satisfied. The result of the above example can be seen in Fig. 5(d).

This second phase of the method can be very similarly summarized in steps as the first phase has been. The main difference between them would be omitting the usage of u^{**} in the insertion procedure.

If the envelopes of the DBSC do not have cusps, then eventually the method will achieve that $D_{H_i}^{\pm} < \epsilon$, $i = \{0, 1, \ldots, n-p\}$ because the skinning curves locally converge to the envelope. However, if there are self-intersections, then depending on ϵ, we might not reach this state. Since our aim is to provide skinning curves without intersections, it is quite natural that the difference between the envelope curves and the skins will be relatively large

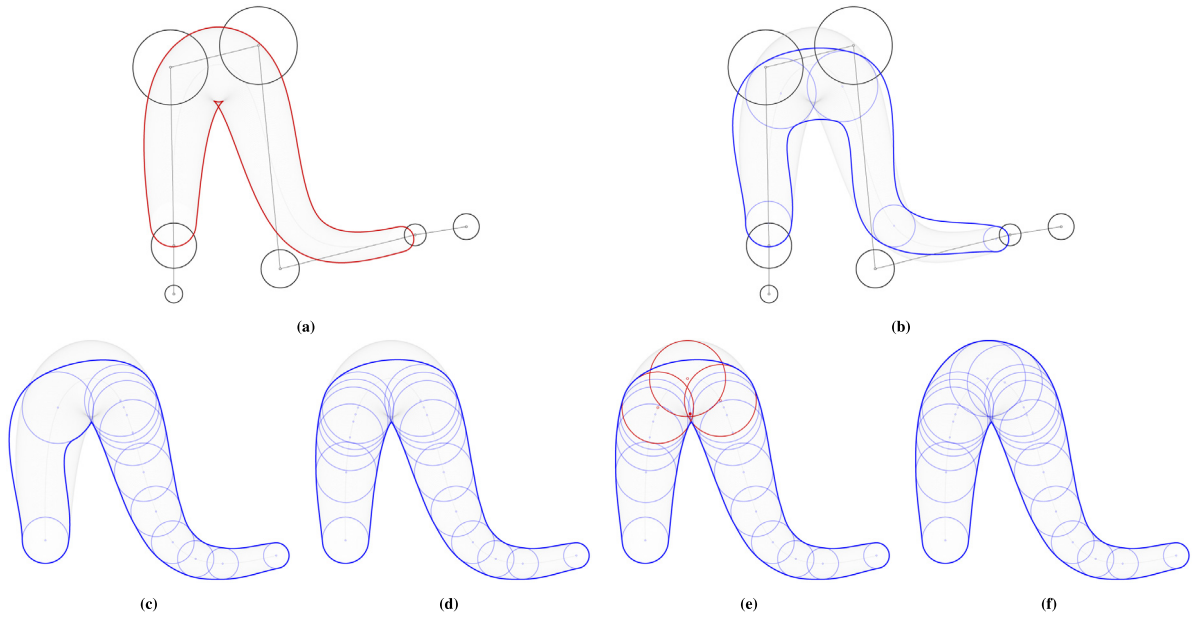

Fig. 5. (a) A shape created with an open DBSC. (b) Initial configuration of knot circles with our method. (c) Starting position of the second phase of our insertion algorithm. Insertion to the first segment is not possible if we consider the neighboring segment. (d) As the asymmetric shape is already avoided and the insertion at the first segment would not directly lead to a non-admissible configuration, there is no need to check the neighboring segment anymore. Thus, the insertion can be completed and iterated. (e) Starting position of the third, refinement step of the algorithm. The shape is not approximating the envelope well, but insertion is not possible. (f) By introducing two-sided skinning, we can insert the necessary knot circles to obtain the shape.

at the affected segments. In such cases, knot circles are inserted up until no more new knot values can be inserted, namely, up until $\mathcal{C}$ is admissible. As a result, the skinning curve will closely follow that segments which do not have intersections, and more loosely at that parts where there are cusps on the envelope.

In a sense, our approximation method is capable of detecting whether the envelope curves have self-intersecting loops: if they do not have, then our approximation will terminate with condition $D^{\pm}_{H_i} < \epsilon$, $i = \{0, 1, \ldots, n-p\}$. If there are self-intersections, then after halting the algorithm, the corresponding envelope curve intersects itself at those segments where $D^{\pm}_{H_i} > \epsilon$ stands.

4.3. Final refinement of the skinning curves

We can extend the algorithm with a third phase by providing a further possibility to reduce the error of the approximation at those segments where the envelope curves have self-intersecting loops. When an envelope segment $\mathbf{q}^{\pm}_i$ has cusps, the self-intersecting loop is only on one side of the envelope, for instance on $\mathbf{q}^{+}_i$, while the other side $\mathbf{q}^{-}_i$ has no singularities. This means that after the second phase of our algorithm, usually only one of the error estimates (e.g., $D^{+}_{H_i}$) is larger than the tolerance. Thus, such a scenario where neither the left, nor the right side of a skinning segment approximate the envelope well, suggests that we face a special case where a new knot circle has to be inserted—even though the second phase has already stopped, meaning that it is not possible to insert knots anymore.

Let us continue the examination of the input in Fig. 5. The second phase of the algorithm ends at Fig. 5(d), where the knot insertion halted. However, we can see that not only $D^{+}_{H_i} > \epsilon$, but also $D^{-}_{H_i} > \epsilon$ stands for a specific curve segment. In this situation, a new knot circle should be inserted at the problematic segment, but this is not possible because of the position of the radical point (see Fig. 5(e)).

To solve this situation, we can use the idea originally shown by Bastl et al. [19]. They introduced the idea of skinning branched systems of circles by separating which circles should be included in the left skin and which ones in the right skin. Even though the use case is different, the construction can work very well in our situation. We will refer to this technique as two-sided skinning. We separate the knot circles whether they are included in the construction of the left or right skins, respectively. Let us denote the new knot circle as c^* that we would like to insert, so we have circles $c_{p+i-1}, c_{p+i}, c^*, c_{p+i+1}, c_{p+i+2}$ from $\mathcal{C}$. We compute the left and right touching points on c^*, and we denote them as $\mathbf{t}^+$ and $\mathbf{t}^-$. When the radical point $\mathbf{s}^*$ of circles c_{p+i}, c^*, c_{p+i+1} is inside c^*, one of the touching points $\mathbf{t}^{\pm}$ lies inside c^* as well. However, the other touching point is usable, therefore we can include c^* to create that side of the skin, but not the other side. Consequently, one of the skins will be constructed by skinning the original sequence $\ldots, c_{p+i-1}, c_{p+i}, c_{p+i+1}, c_{p+i+2}, \ldots$, and the other skinning curve is obtained by skinning $\ldots, c_{p+i-1}, c_{p+i}, c^*, c_{p+i+1}, c_{p+i+2}, \ldots$. In the example of Fig. 5(e), the right touching point is inside circle c^*, so c^* is used for the creation of the left skin, but not in the right one's. It is an interesting property of the approach that the middle red circle in Fig. 5(f) is an obstacle in the second phase of the insertion but then part of the solution in the last step of the approximation process. The result of the two-sided skinning solution can be seen in Fig. 5(f), where we inserted two more knot circles to the left skin. Another example is shown in Fig. 6 where we compare our method with the original DBSC shape.

The third phase of our method can be formalized as follows:

1. Mark all circles in $\mathcal{C}$ as BOTH.
2. For each segment where $D^{+}_{H} > \epsilon$ and $D^{-}_{H} > \epsilon$ both hold:
   - Determine the value for u^* and then circle c^*.
   - If the radical point of c_{p+i}, c^*, and c_{p+i+1} is inside c^*:
     - Insert the new knot value.
     - Calculate $\mathbf{t}^+$ and $\mathbf{t}^-$, the left and right touching points on c^*.
     - If $\mathbf{t}^+$ is inside c^*, then mark c^* as RIGHT.
     - If $\mathbf{t}^-$ is inside c^*, then mark c^* as LEFT.
3. Compute the left skinning curve for circles marked as LEFT and BOTH.

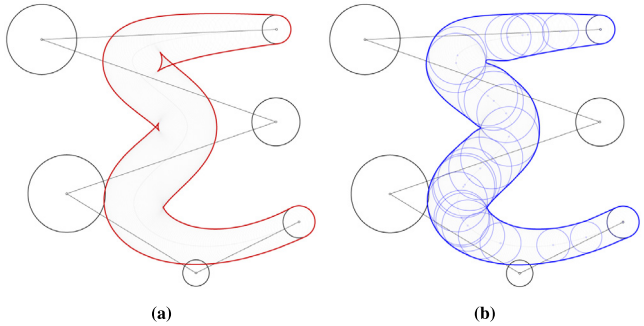

Fig. 6. For the same DBSC shape as in Fig. 3, we can see that (a) the shape created with the DBSC has cusps, but (b) the skinning shape that we provide does not have such self-intersections.

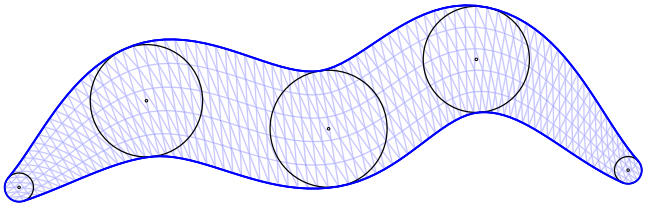

Fig. 7. Obtaining the inner points of the skinned region yields the tessellation of the model.

4. Compute the right skinning curve for circles marked as RIGHT and BOTH.
5. If needed, more "two-sided" knot circles can be inserted to reduce the approximation error.

This refinement phase has to be done after all iterations of the knot insertion are finished, as introducing two-sided skinning right at the beginning of the method would add unnecessary complexity to the approximation task. We discuss this issue in more detail in Section 6.

5. Results

As we previously mentioned in Section 3, if the envelope curves of the DBSC have self-intersections or cusps, then we also get intersections in the tessellated region as well. This causes a problem when we intend to apply texture to the region. By using our proposed method, we get shapes without such problems, so the textured region is flawless. In the literature, tessellation has not yet been performed on regions defined by skinning curves. For this purpose, we can parametrize the interior of the skinned region using linear interpolation, similarly as Wu et al. defined the inside of the DBSC [4]. Between circles c_i and c_{i+1}, the inner points of the region can be parametrized as

$$\mathbf{s}_i(t,v) = (1-v)\,\mathbf{s}_i^-(t) + v\,\mathbf{s}_i^+(t), \quad t,v \in [0,1]. \tag{26}$$

If we would like to directly assign the skinning curve to the DBSC, i.e., for a given parameter value $u \in [u_k, u_{k+1}[$ on the DBSC, the inner points of the corresponding skinning segment can be calculated as

$$\mathbf{s}_{k-p}(u,v) = (1-v)\;\mathbf{s}_{k-p}^-\left(\frac{u-u_k}{u_{k+1}-u_k}\right) + v\;\mathbf{s}_{k-p}^+\left(\frac{u-u_k}{u_{k+1}-u_k}\right). \tag{27}$$

Fig. 7 shows a tessellated region bounded by skinning curves. Fig. 8 shows a comparison between the textured DBSC and skinning shape. We can clearly see the distortion when the intersection happens at the model created with the DBSC. On the contrary, there are no such cusps on the shape constructed with our proposed approach. Two other examples can be seen in Figs. 9 and 10.

6. Discussion and conclusion

In this section we briefly discuss the usability of our method. As one can see, the proposed method provides a solution to avoid the self-intersections of the boundary curves of DBSCs. As we discussed, our technique will closely follow the shape described by the DBSC because the skinning curves locally converge to the envelope curves (as described in [20]). When there are self-intersecting loops and cusps on the envelope, we do not want to generate the same intersections but follow the shape in a slightly looser way. This will certainly result in a greater error in the approximation of the cusps, but this is in fact our intention. Therefore, our new technique is capable of generating an intersection-free boundary while still following the intended shape.

Moreover, since the chosen circles are from the family of circles described by the DBSC, the advantageous properties of the original shape hold as well, such as convex hull property, affine invariancy, and local modification. As a plus, the skinning curves are made of Bézier curves, thus several calculations get simplified, e.g., obtaining derivatives or intersection detection.

The new method combines the advantageous properties of DBSC-based design and skinning techniques. Compared to traditional skinning, this combination with DBSCs not only offers the above-mentioned inherited advantages, but our method can provide more flexibility in the modeling process with the use of control disks. Traditional skinning methods work very well with a relatively small number of circles, and the user can easily create a model and modify the resulting shape. In the case of more complex models, it is only possible to create the desired shape with many more circles. However, once the shape is created, modifying it would require changing the positions or radii of a larger number of circles that are likely to be very close to each other. Therefore it can be a challenging task to precisely control the shape. On the other hand, when we use our DBSC-based skinning technique, the user does not directly see the large number of knot circles, but they can simply modify the positions and radii of the control disks. This results in a much simpler and carefree user experience in the modeling process.

The proposed DBSC-based skinning technique can be used in those scenarios where our goal is to obtain intersection-free shapes. This includes tessellated and textured shapes, but also when the outline, i.e., the contour of the shape is important. This can be the case in the modeling and manufacturing process of, for example, carvings, engravings or LED signs. Figs. 11 and 12 show further examples of shapes created with our DBSC-based skinning technique.

6.1. Limitations of our method

Our proposed method has its limitations when it comes to the shape required by the user. As we mentioned in Section 4, the initial step of setting the knot circles requires that $\mathcal{C}$ must be an admissible configuration of our skinning procedure in terms of Definition 4.1. In those cases where this condition is not met, our algorithm would halt. This can be regarded as validated because even if we chose other circles as starting knot circles, after a few iterations we would indeed get a non-admissible sequence of circles. Then the algorithm would stop, and we would not be able to have proper skinning curves on such a shape. Fig. 13 shows an example where the second admissibility condition is violated of Definition 4.1, resulting in intersecting skinning curves. If the first

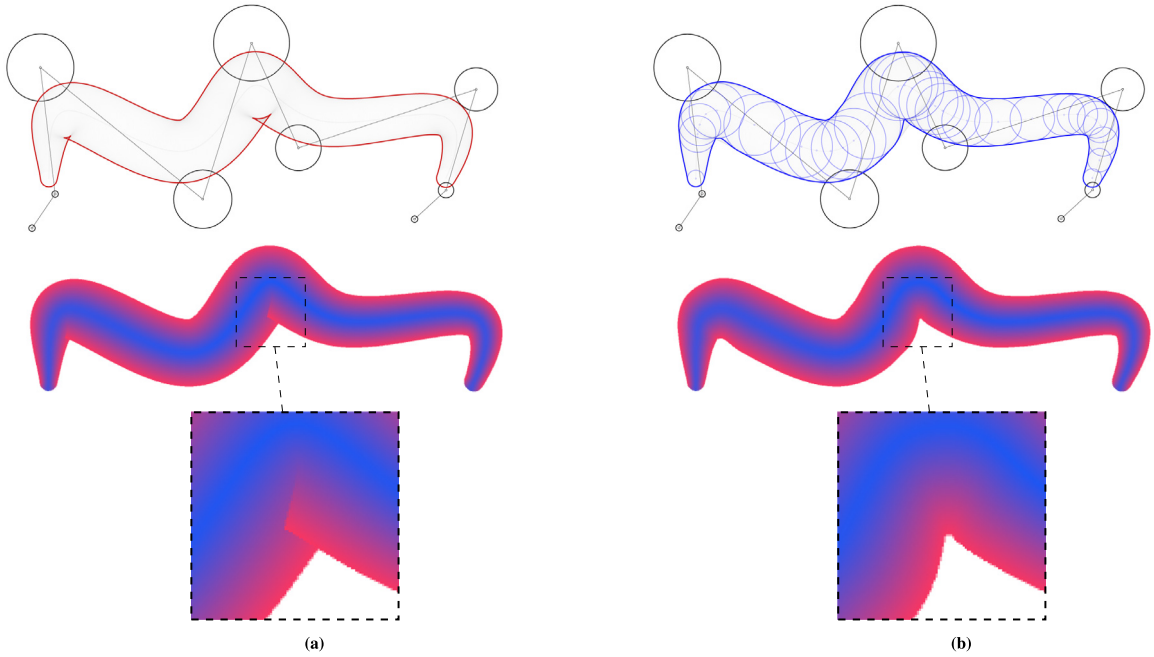

Fig. 8. Comparison of (a) the tessellated and textured open DBSC shape and (b) the results of our proposed method. The cusps on the boundary of the DBSC cause unintended intersection on the texture, while our DBSC-based skinning solution has no such artefacts.

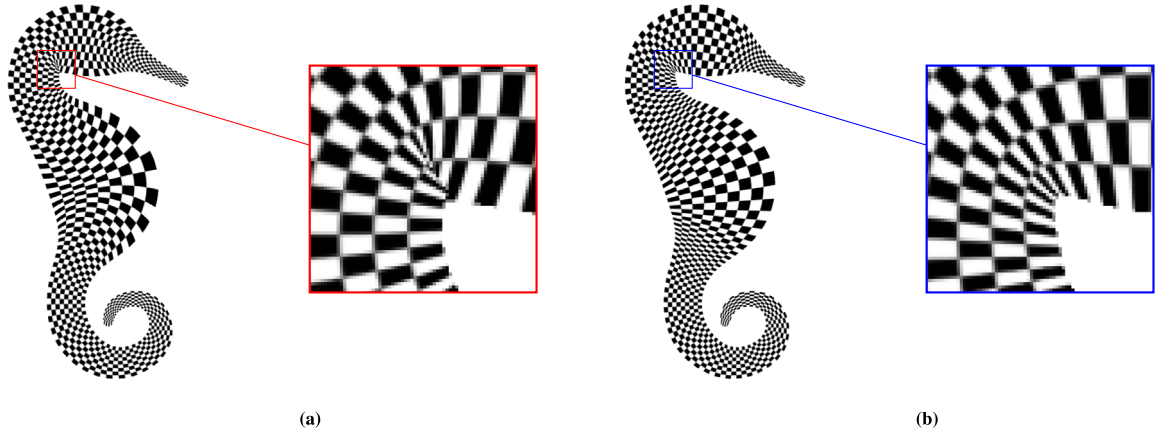

Fig. 9. Comparing the textured seahorse model generated by (a) a disk B-spline curve and (b) our proposed, DBSC-based skinning method.

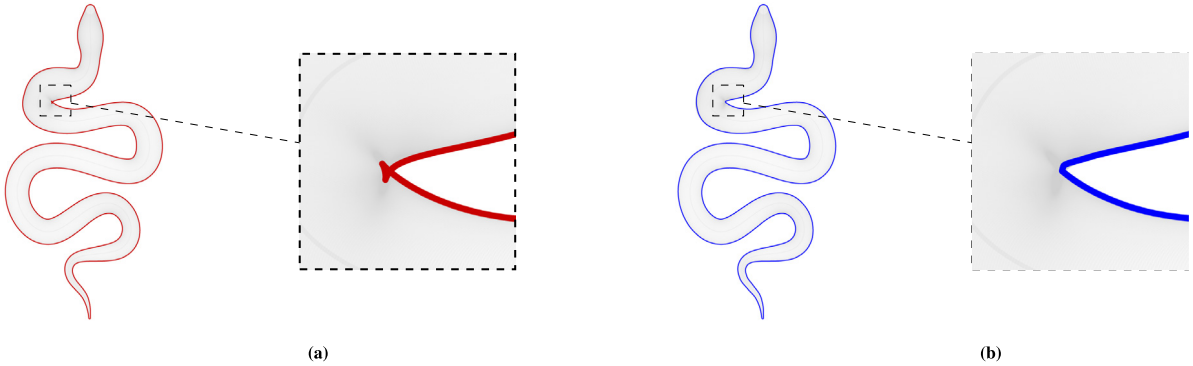

Fig. 10. A snake model created by using (a) the envelope curves of the DBSC, (b) the skinning curves constructed with our technique. As we can see in the zoomed part, the envelope has a self-intersecting loop, while our solution does not have any.

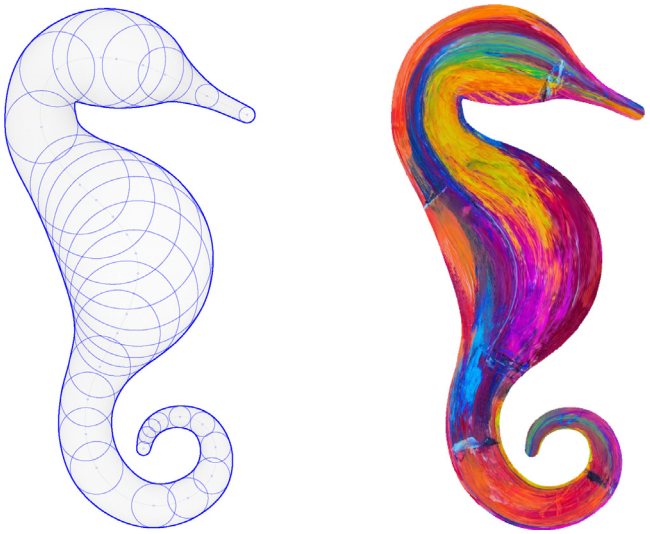

Fig. 11. The seahorse model created with the proposed DBSC-based skinning technique. The resulting model is tessellated and textured.

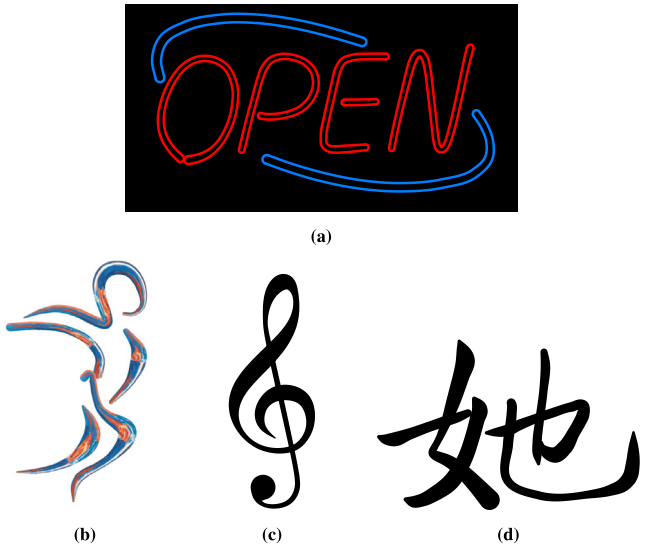

Fig. 12. Artistic shapes created with our proposed DBSC-based skinning method. (a) Design for manufacturing a neon sign, (b) remodeled version of Wu et al.'s [4] running human, (c) G-clef model, (d) Chinese character remodeled by our method, inspired by the Qi Gong calligraphy displayed in [12].

or third condition is not met, then it is not possible to create skins at all. We therefore restrict the method to admissible sequences of circles.

If we insist on using skinning curves in these cases (because the envelopes have intersections), then we could apply two-sided skinning or let the error tolerance be much higher for a more loosened approximation.

As we mentioned in Section 4, introducing two-sided skinning with separation of the left and right circle sequences gives a tremendous amount of freedom in creating a shape. However, it would become overwhelmingly complex to control this approach algorithmically. Therefore, we use this tool only to fine-tune the final iteration of the approximation.

6.2. Extension to 3D

A natural extension of disk B-spline curves to 3D is ball B-spline curves (BBSC), where instead of control disks, we have

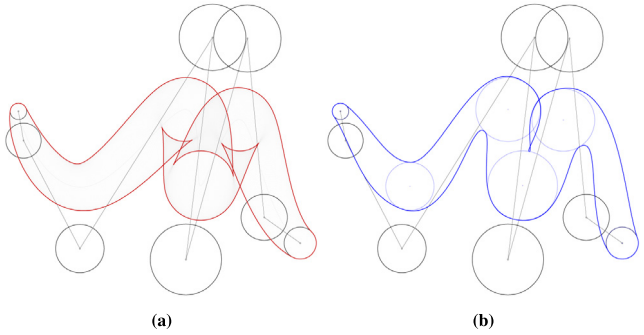

Fig. 13. Limitations of our method. (a) The control circles define such a shape that the envelope curves have extremely deep self-intersecting loops. (b) In this case, we obtain a non-admissible configuration of input circles of our method, resulting in a skinning curve that has self-intersection.

control balls. The idea was introduced by Wu et al. [41], after which several papers have been published on both the theory of BBSCs and their possible applications [42–44]. The boundary of a BBSC is a canal surface, on which—in a similar way as in 2D—self-intersections can occur. Since skinning of spheres is also developed (see [18–20]), extending our method to 3D can also be done in an analogous way to the algorithm in Section 4. We intend to further investigate this possible extension as future work.

CRediT authorship contribution statement

Kinga Kruppa: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Funding acquisition. **Roland Kunkli:** Conceptualization, Methodology, Validation, Investigation, Writing – review & editing. **Miklós Hoffmann:** Conceptualization, Methodology, Validation, Investigation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

Supported by the ÚNKP-22-4 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund, Hungary.

References

[1] Lin Q, Rokne JG. Disk Bézier curves. Comput Aided Geom Design 1998;15(7):721–37. http://dx.doi.org/10.1016/S0167-8396(98)00016-8.
[2] Seah HS, Wu Z, Tian F, Xiao X, Xie B. Artistic brushstroke representation and animation with disk B-spline curve. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology. 2005, p. 88–93. http://dx.doi.org/10.1145/1178477.1178489.
[3] Hu G, Yang R, Wei G. Hybrid chameleon swarm algorithm with multi-strategy: A case study of degree reduction for disk Wang–Ball curves. Math Comput Simulation 2023;206:709–69. http://dx.doi.org/10.1016/j.matcom.2022.12.001.

[4] Wu Z, Wang X, Liu S, Chen Q, Seah HS, Tian F. Skeleton-based parametric 2-D region representation: Disk B-Spline curves. IEEE Comput Graph Appl 2021;41(3):59–70. http://dx.doi.org/10.1109/MCG.2021.3069847.
[5] Ao X, Fu Q, Wu Z, Wang X, Zhou M, Chen Q, et al. An intersection algorithm for disk B-spline curves. Comput Graph 2018;70:99–107. http://dx.doi.org/10.1016/j.cag.2017.07.021.
[6] Yang W, Chen F. Merging a pair of disk Bézier curves. In: Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia. 2004, p. 65–70. http://dx.doi.org/10.1145/988834.988845.
[7] Chen F, Yang W. Degree reduction of disk Bézier curves. Comput Aided Geom Design 2004;21(3):263–80. http://dx.doi.org/10.1016/j.cagd.2003.10.004.
[8] Zhang Y, Wu Z, Wang X. Dynamic disk B-spline curves. Comput Animat Virtual Worlds 2020;31(4–5):e1955. http://dx.doi.org/10.1002/cav.1955.
[9] Seah HS, Tandianus B, Sui Y, Wu Z. Expressive B-spline curves: a pilot study on a flexible shape representation. In: Muramatsu S, Nakajima M, Kim J-G, Guo J-M, Kemao Q, editors. International Workshop on Advanced Imaging Technology (IWAIT) 2022. Hong Kong, China: SPIE; 2022, p. 87. http://dx.doi.org/10.1117/12.2626063.
[10] Du B, Hu G, Zhong J, Yuan X. Disk ball curve with shape parameters: construction and its degree reduction. In: International Conference on Advanced Manufacturing Technology and Manufacturing Systems. ICAMTMS 2022, vol. 12309, SPIE; 2022, p. 460–8. http://dx.doi.org/10.1117/12.2645626.
[11] Min C, Guojin W. Shape blending of artistic brushstroke represented by disk B-spline curves. Prog Nat Sci 2007;17(12):1501–7.
[12] Wang M, Fu Q, Wang X, Wu Z, Zhou M. Evaluation of Chinese calligraphy by using DBSC vectorization and ICP algorithm. Math Probl Eng 2016;2016:4845092. http://dx.doi.org/10.1155/2016/4845092.
[13] Fu Q, Wu Z, Ying X, Wang M, Zheng X, Zhou M. Generating Chinese calligraphy on freeform shapes. In: Gavrilova ML, Tan CK, Sourin A, editors. Transactions on Computational Science XXVIII. vol. 9590, Berlin, Heidelberg: Springer Berlin Heidelberg; 2016, p. 69–87. http://dx.doi.org/10.1007/978-3-662-53090-0_4.
[14] Klimenko S, Mestetskiy L, Semenov A. Imitation of Handwriting for Art and Heritage in Cyberspace. In: 2017 International conference on cyberworlds (CW). IEEE; 2017, p. 174–7. http://dx.doi.org/10.1109/CW.2017.57.
[15] Chen Q, Tian F, Seah H, Wu Z, Qiu J, Konstantin M. DBSC-based animation enhanced with feature and motion. Comput Animat Virtual Worlds 2006;17(3–4):189–98. http://dx.doi.org/10.1002/cav.122.
[16] Mestetskii LM. Fat curves and representation of planar figures. Comput Graph 2000;24(1):9–21. http://dx.doi.org/10.1016/S0097-8493(99)00133-8.
[17] Seah HS, Tandianus B, Sui Y, Wu Z, Zhang Z. Modeling and Rendering with eXpressive B-Spline Curves. In: 31. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. WSCG 2023, 2023, URL http://wscg.zcu.cz/wscg2023/papers/full/E31-full.pdf. [Accessed 18 May 2023].
[18] Kunkli R, Hoffmann M. Skinning of circles and spheres. Comput Aided Geom Design 2010;27(8):611–21. http://dx.doi.org/10.1016/j.cagd.2010.07.003.
[19] Bastl B, Kosinka J, Lávička M. Simple and branched skins of systems of circles and convex shapes. Graph Models 2015;78:1–9. http://dx.doi.org/10.1016/j.gmod.2014.12.001.
[20] Kruppa K, Kunkli R, Hoffmann M. An improved skinning algorithm for circles and spheres providing smooth transitions. Graph Models 2019;101:27–37. http://dx.doi.org/10.1016/j.gmod.2018.12.001.
[21] Yao C, Rokne J. Fat curves. Comput Graph Forum 1991;10(3):237–48. http://dx.doi.org/10.1111/1467-8659.1030237.
[22] Hobby JD. Rasterizing curves of constant width. J ACM 1989;36(2):209–29. http://dx.doi.org/10.1145/62044.62045.
[23] Zhou H, Ting K-L. Shape and size synthesis of compliant mechanisms using wide curve theory. J Mech Des 2006;128:551–8. http://dx.doi.org/10.1115/1.2180809.
[24] Zhou H, Ting K-L. Geometric optimization of spatial compliant mechanisms using three-dimensional wide curves. J Mech Des 2009;131(5):051002. http://dx.doi.org/10.1115/1.3086792.
[25] Moon HP. Minkowski Pythagorean hodographs. Comput Aided Geom Design 1999;16(8):739–53. http://dx.doi.org/10.1016/S0167-8396(99)00016-3.
[26] Choi HI, Choi SW, Moon HP. Mathematical theory of medial axis transform. Pacific J Math 1997;181(1):57–88. http://dx.doi.org/10.2140/pjm.1997.181.57.
[27] Kosinka J, Lávička M. On rational Minkowski Pythagorean hodograph curves. Comput Aided Geom Design 2010;27(7):514–24. http://dx.doi.org/10.1016/j.cagd.2010.06.003.
[28] Kruppa K, Kunkli R, Hoffmann M. Possibilities and advantages of rational envelope and Minkowski Pythagorean hodograph curves for circle skinning. Mathematics 2021;9(8):843. http://dx.doi.org/10.3390/math9080843.
[29] Bizzarri M, Lávička M. Interpolation of Hermite data by clamped Minkowski Pythagorean hodograph B-spline curves. J Comput Appl Math 2021;392:113469. http://dx.doi.org/10.1016/j.cam.2021.113469.
[30] Berio D, Leymarie FF, Asente P, Echevarria J. StrokeStyles: Stroke-based segmentation and stylization of fonts. ACM Trans Graph 2022;41(3):1–21. http://dx.doi.org/10.1145/3505246.
[31] Bizzarri M, Lávička M, Kosinka J. Skinning and blending with rational envelope surfaces. Comput Aided Des 2017;87:41–51. http://dx.doi.org/10.1016/j.cad.2017.02.002.
[32] Manocha D, Canny JF. Detecting cusps and inflection points in curves. Comput Aided Geom Design 1992;9(1):1–24. http://dx.doi.org/10.1016/0167-8396(92)90050-Y.
[33] Elber G, Cohen E. Error bounded variable distance offset operator for free form curves and surfaces. Internat J Comput Geom Appl 1991;1(1):67–78. http://dx.doi.org/10.1142/S0218195991000062.
[34] Sederberg TW, Parry SR. Comparison of three curve intersection algorithms. Comput Aided Des 1986;18(1):58–63. http://dx.doi.org/10.1016/S0010-4485(86)80013-6.
[35] Sederberg TW, Nishita T. Curve intersection using Bézier clipping. Comput Aided Des 1990;22(9):538–49. http://dx.doi.org/10.1016/0010-4485(90)90039-F.
[36] Lou Q, Liu L. Curve intersection using hybrid clipping. Comput Graph 2012;36(5):309–20. http://dx.doi.org/10.1016/j.cag.2012.03.021.
[37] Jüttler B. Bounding the Hausdorff distance of implicitly defined and/or parametric curves. In: Mathematical methods for curves and surfaces: Oslo 2000. USA: Vanderbilt University; 2000, p. 223–32.
[38] Alt H, Scharf L. Computing the Hausdorff distance between curved objects. Internat J Comput Geom Appl 2008;18(4):307–20. http://dx.doi.org/10.1142/S0218195908002647.
[39] Elber G, Grandine T. Hausdorff and minimal distances between parametric freeforms in $\mathbb{R}^2$ and $\mathbb{R}^3$. In: Chen F, Jüttler B, editors. Advances in Geometric Modeling and Processing. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer; 2008, p. 191–204. http://dx.doi.org/10.1007/978-3-540-79246-8_15.
[40] Kim Y-J, Oh Y-T, Yoon S-H, Kim M-S, Elber G. Precise Hausdorff distance computation for planar freeform curves using biarcs and depth buffer. Vis Comput 2010;26(6–8):1007–16. http://dx.doi.org/10.1007/978-3-540-79246-8_15.
[41] Wu Z, Seah HS, Zhou M. Skeleton based parametric solid models: Ball B-Spline curves. In: 2007 10th 10th IEEE International Conference on Computer-Aided Design and Computer Graphics. Beijing, China: IEEE; 2007, p. 421–4. http://dx.doi.org/10.1109/CADCG.2007.4407920.
[42] Tang Y, Wu Z, Zhou M. Sketching 3D plant based on Ball B-Spline curves and L-system. In: 2009 Third International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications. Beijing, China: IEEE; 2009, p. 425–32. http://dx.doi.org/10.1109/PMA.2009.41.
[43] Wu Z, Zhou M, Wang X. Interactive modeling of 3D tree with Ball B-Spline curves. Int J Virtual Real 2009;8(2):101–7. http://dx.doi.org/10.20870/IJVR.2009.8.2.2731.
[44] Wang X, Wu Z, Shen J, Zhang T, Mou X, Zhou M. Repairing the cerebral vascular through blending Ball B-Spline curves with G^2 continuity. Neurocomputing 2016;173:768–77. http://dx.doi.org/10.1016/j.neucom.2015.08.028.