The Visual Computer manuscript No.
(will be inserted by the editor)

$\epsilon\kappa$-curves: Controlled Local Curvature Extrema

Kenjiro T. Miura · R.U. Gobithaasan · Péter Salvi · Dan Wang · Tadatoshi Sekine · Shin Usuki · Jun-ichi Inoguchi · Kenji Kajiwara

Revised: January 30, 2021

Abstract The κ-curve is a recently published interpolating spline which consists of quadratic Bézier segments passing through input points at the loci of local curvature extrema. We extend this representation to control the magnitudes of local maximum curvature in a new scheme called *extended-* or $\epsilon\kappa$-curves.

κ-curves have been implemented as the curvature tool in Adobe Illustrator® and Photoshop®, and are highly valued by professional designers. However, because of the limited degrees of freedom of quadratic Bézier curves, it provides no control over the curvature distribution.

We propose new methods that enable the modification of local curvature at the interpolation points by degree elevation of the Bernstein basis as well as application of generalized trigonometric basis functions. By using $\epsilon\kappa$-curves, designers acquire much more ability to produce a variety of expressions, as illustrated by our examples.

K. T. Miura
Shizuoka University

R. U. Gobithaasan
University Malaysia Terengganu

Péter Salvi
Budapest University of Technology and Economics

Dan Wang
Shizuoka University

Tadatoshi Sekine
Shizuoka University

Shin Usuki
Shizuoka University

Jun-ichi Inoguchi
University of Tsukuba

Kenji Kajiwara
Kyushu University

Keywords Interpolatory curves · Curvature continuity · Control of curvature magnitude · Degree elevation

1 Introduction

The κ-curve, proposed recently by [27], is an interpolating spline which is curvature-continuous almost everywhere and passes through input points at the local curvature extrema. It has been implemented as the curvature tool in Adobe Illustrator® and Photoshop® and is accepted as a favored curve design tool by many designers (see e.g. [4,6]).

We consider the reasons for the success of κ-curve to be:

1. Information along contours is concentrated at local maxima of curvature.
2. Curves of low degree have smooth distribution of curvature.
3. G^2-continuous curves tend to look fairer than only G^1-continuous ones.

Attneave [1] suggested, based on his empirical study, that information along contours is concentrated in regions of high magnitude of curvature, as opposed to being distributed uniformly along the contour, and it is further concentrated at local maxima of curvature (see also [32]). Although Attneave never published the details of his methods, [19] conducted a similar experiment, and obtained the same results. Levien and Séquin [11] argue similarly, and assert that points of maximal curvature are salient features.

The curvature of a polynomial curve is given by a relatively complicated rational function [7], and its distribution might not be globally smooth. However, if the

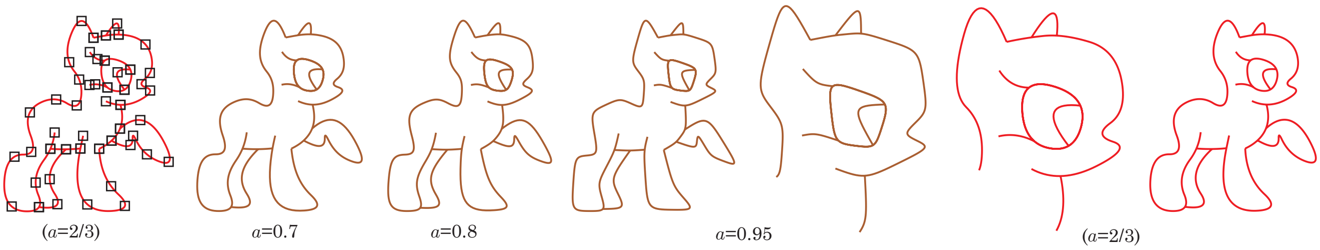

Fig. 1: The left- and rightmost curves are κ-curves; the others are $\epsilon\kappa$-curves with gradual changes in the global shape parameter a. The face part (for $a = 0.95$ and $a = 2/3$) is zoomed in for better comparison. When a=2/3, κ- and $\epsilon\kappa$-curves are identical.

curve is of a low degree, the curvature distribution is more uniform and the curve is fairer, thus more suitable for illustration. The quadratic polynomial curve has the nice property that its curvature has only one local maximum, and its location is easily computable [27], which makes the handling of curvature extrema much easier.

Graphic designers often accept G^1 continuity as good enough for illustration. However, discontinuity remains; for example, if you join a straight line and a circular arc with G^1 continuity, the rhythm of the curve will be broken at the joint. For this reason we give pereference to G^2-continuous curves.

Nonetheless, κ-curves are not perfect, and their further investigation is necessary [28]. The following are two important shortcomings of κ-curves:

1. They are not curvature-continuous everywhere: at inflection points only G^1 continuity is guaranteed.
2. Since the degree of freedom (DoF) of the quadratic segments is limited, it is impossible to control the magnitudes of local maximum curvature at the input points.

For the first shortcoming, Wang et al. [25] provided a solution through the use of log-aesthetic curves [14, 16] instead of polynomial curves. Log-aesthetic curves have a shape parameter (α), which can be utilised to control the curvature distribution as shown in Figure 2. Their method guarantees G^2 continuity everywhere, including inflection points, since log-aesthetic curves with negative α values can represent S-shaped curves with G^2 continuity (note that these cannot be represented by quadratic Bézier curves).

These curves, however, are defined by a Cesàro equation, and thus take extra time to evaluate, making the interpolation method impractical for real-time design purposes.

Because of the second shortcoming, if the designer wants to increase or decrease the magnitude of the curvature extremum, she needs to add extra input points, as shown in Fig. 3. Yan et al. [26] proposed a piecewise rational, quadratic, interpolatory curve that is able to reproduce circles and other elliptical or hyperbolic shapes. Although their main intention was to reproduce circles, their method could also control the magnitude of local maximum curvature. However, only rational quadratic curves are applicable, and it is not possible to extend their method for other types.

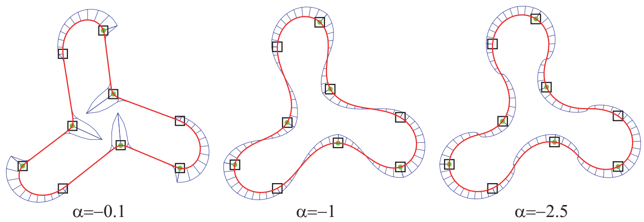

Fig. 2: Log-aesthetic curves with various α values [25]. Input points are depicted by black boxes and green points correspond to positions of local maximum curvature. The blue curves show the normal curvature. These curves are G^2-continuous everywhere.

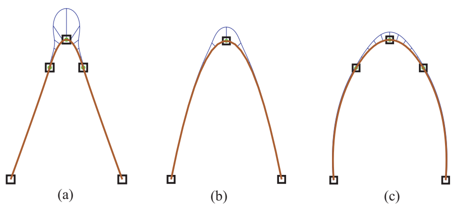

Fig. 3: Addition of extra input points to control the magnitude of curvature extrema: (b) is the original κ-curve, (a) and (c) shows deformed curves with large and small curvature extrema by adding two extra input points.

In this paper we propose a new method to solve the second shortcoming by degree elevation of the Bernstein basis functions, giving an extra DoF to each quadratic curve segment, providing control over the magnitudes of local maximum curvature at input points. In order to increase the designers' possible choices, we also introduce a new trigonometric basis for which we can perform degree elevation. In addition, we propose a general method for bases with extra shape parameters. By adding one more parameter to each of the curve segments, the designers obtain more expressive power for their illustration. The family of this new curve is denoted as $\epsilon\kappa$-curves.

$\epsilon\kappa$-curves preserve all of the appealing properties of κ-curves, i.e., point interpolation, G^2 continuity (except at inflection points), continuous modification (changes smoothly when the input points move), local influence, and real-time generation. With a small processing overhead, $\epsilon\kappa$-curves offer the ability to control the magnitude of local maximum curvature.

We have implemented $\epsilon\kappa$-curves in MATLAB® and Julia [10]. The source of the Julia code is available online [21].

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents a method to control the magnitudes of local maximum curvature by degree elevation of the Bernstein basis functions. Section 4 introduces a new trigonometric basis and proposes a method with degree elevation similar to the one proposed in the previous section. Finally, we end with conclusions and discussion of future work.

2 Related Work

In this section, we first review [27] and their underlying strategy developing κ-curves. Next, we discuss related researches on various kinds of basis function formulations for curve design.

2.1 κ-curve

The basic framework of our method is adopted from [27], generating curves controlled by interpolation points. κ-curves have stimulated the field of interpolatory curve generation, resulting in works such as [5], which proposes a method for good control over the location and type of geometric feature points (e.g. cusps and loops).

Yan et al. [27] create a sequence of quadratic curves with G^2 continuity almost everywhere. They derived explicit formulae for the point between two quadratic segments to guarantee G^2 continuity, and for the additional condition that input points should be interpolated at maximum curvature magnitude positions. κ-curves are determined by the locations of the middle control points of quadratic Bézier segments – the rest is easily derived from the continuity constraints. Hence the variables are the locations of these middle control points.

Their basic strategy to determine these locations is to adopt a local/global approach [12,22]: G^2-continuous connection is performed locally, while the interpolation at maximum curvature magnitude positions is done globally.

Regarding control of the magnitude of local curvature, the most common technique is to change the weight of a control point of a rational curve [7]. A larger weight attracts the curve to its control point, which makes local curvature larger. However, this technique is not applicable for interpolatory curves. Another technique is to introduce extra parameters called bias and tension to the B-spline formulation for controlling local curvature [2], but this is also not applicable to interpolatory curves.

2.2 Basis Functions

As mentioned in [27], curve modeling has a long history, especially in computer-aided geometric design, as well as computer graphics. In CAGD and applied mathematics, to extend the expressive power of curves, many researchers have been trying to develop new bases with extra shape parameters. The following is a (nonexhaustive) list of such bases:

1. C-Bézier curve [30]
2. Cubic alternative curve [9]
3. Cubic trigonometric Bézier curve (T-Bézier basis) [8]
4. $\alpha\beta$-Bernstein-like basis [31]
5. Quasi-cubic trigonometric Bernstein basis [29]
6. Trigonometric cubic Bernstein-like basis [23]

Our method proposed in the next section can be applied for curves based not only on polynomials, but also other bases such as a trigonometric basis with degree elevation property which we will introduce in Section 4. All of the representations listed above have extra parameters for shape control, and we can utilize these parameters to control the magnitudes of local maximum curvature (not all curve types are applicable, however, as explained in Appendix C).

It is interesting to note that most researchers to date have attempted to develop new bases using four control points, based on cubic polynomials or quadratic trigonometric functions. They prefer using four control points out of concern for connections at both ends of the curve. In order to control the magnitude of curvature at the two ends independently, at least two control points are necessary at each end. This differs fundamentally from Yan et al.'s (and our) approach, which uses only three control points.

The importance of [27] is their proposed paradigm shift for curve generation, by considering the local maximum curvature in the middle, instead of focusing on the endpoints. If we can assume that the curvature has just one local maximum in each segment, then only one extra parameter per segment is adequate to control the magnitude of its local maximum curvature.

To our best knowledge, no trigonometric basis family for arbitrary degree has been published yet. Our novel *generalized trigonometric basis functions* range from linear, using three control points, to any higher degree n, using $2n+1$ control points. The curve can be evaluated by a recursive method, similar to de Casteljau's algorithm, as explained in Appendix B. Since the curve uses trigonometric functions as blending functions, it can represent a circular arc exactly, without using a rational form.

3 Cubic Bernstein Polynomials

In this section we extend κ-curves in a direct manner, by elevating the degree of quadratic Bézier segments to cubic. $\epsilon\kappa$-curves retain the following properties of κ-curves:

1. Interpolate all input points (control points).
2. All local maximum curvature points are the same as the input points.
3. G^2 continuity is guaranteed almost everywhere (except for inflection points).

In the following we discuss only closed curves, but it is straightforward to extend our methods to open curves as has been demonstrated for κ-curves.

If we elevate the degree of a planar Bézier curve, we obtain an additional control point, which has two DoFs (the x and y coordinates). To reduce these to one, we add a geometric constraint on the location of the second and third control points of the cubic Bézier curve, as shown in Figure 4. Here a is an internal division ratio, where the larger a is, the closer the control points P_1 and P_2 are to the control point Q_1. We make the restriction $2/3 \leq a < 1$ because the curve should not have a complicated curvature distribution. Using Q_i, the curve $C(t; a)$ is expressed by

$$C(t;a) = (1-t)^3 Q_0 + 3(1-t)^2 t\,[(1-a)Q_0 + aQ_1] + 3(1-t)t^2\,[aQ_1 + (1-a)Q_2] + t^3 Q_2. \quad (1)$$

Note that if $a = 2/3$, the curve degenerates to quadratic.

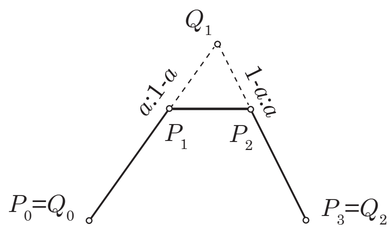

Fig. 4: Constrained cubic Bézier curve. If $a = 2/3$, the curve becomes quadratic.

We have proved that by constraining the construction of the cubic polynomial curve as in Eq. (1), using only three control points instead of four, the curvature in one curve segment has at most one local maximum for $2/3 \leq a < 1$; see details in [15], as well as Appendix A for a general discussion on the curvature extrema of cubic polynomial curves, and a high-level summary of the proof. Hence we can safely assume that the curvature in one curve segment has at most one local maximum, and a single extra parameter for each segment is enough to control the magnitudes of local maximum curvature.

3.1 Geometric Constraints

We assume that $\epsilon\kappa$-curves consist of a sequence of constrained cubic polynomial curves

$$c_i(t;a_i) = (1-t)^3 c_{i,0} + 3(1-t)^2 t\,[(1-a_i)c_{i,0} + a_i c_{i,1}] + 3(1-t)t^2\,[a_i c_{i,1} + (1-a_i)c_{i,2}] + t^3 c_{i,2}, \quad (2)$$

parameterized by t and also a_i, which is an extra shape parameter. The control points are given by $c_{i,0}$, $c_{i,1}$ and $c_{i,2} \in \mathbb{R}^2$, corresponding to Q_i, $i = 0, 1, 2$ in Figure 4. The a_i's are reserved for designers and can be manipulated independently.

The curvature of this curve $c_i(t; a_i)$ is given by

$$\kappa_i(t;a_i) = \det\left(\frac{\partial c_i(t;a_i)}{\partial t}, \frac{\partial^2 c_i(t;a_i)}{\partial t^2}\right) \Big/ \left\|\frac{\partial\, c_i(t;a_i)}{\partial t}\right\|^3$$
$$= \frac{4}{3} \cdot \frac{\triangle(c_{i,0}, c_{i,1}, c_{i,2})\,[a_i^2(1-t)t + a_i(1-a_i)\,((1-t)^2+t^2)]}{\|(1-t)^2 a_i r_i + 2(1-t)t(1-a_i)(r_i+s_i) + t^2 a_i s_i\|^3}, \quad (3)$$

where $\triangle$ indicates the area of the triangle specified by its arguments, and $r_i = c_{i,1} - c_{i,0}$, $s_i = c_{i,2} - c_{i,1}$.

In the quadratic case (i.e., κ-curves), as the curvature has such a simple formula, we can express the parameter t_i at the point of maximal curvature explicitly, in terms of the Bézier coefficients of the i^{th} quadratic Bézier curve as

$$t_i = \frac{\langle r_i, r_i - s_i\rangle}{\|r_i - s_i\|^2}, \tag{4}$$

where $\langle a, b\rangle$ means the scalar product of vectors a and b. Then we add the condition

$$c_i(t_i) = p_i, \tag{5}$$

where p_i is the i^{th} input point. Solving for $c_{i,1}$ and substituting into Eq. (4), we get a cubic equation in t_i that depends only on the endpoints $c_{i,0}$ and $c_{i,2}$, and the input points p_i.

Unfortunately, we cannot obtain an explicit formula like Eq. (4) for the parameter t_i in the cubic case, because of its high degree (see details in Appendix A.1), but this is not a problem. Solving Eq. (5) for $c_{i,1}$, we arrive at

$$\begin{aligned} c_{i,1} = \big[&p_i - (1-t_i)^3 c_{i,0} \\ &- 3(1-t_i)t_i(1-a_i)((1-t_i)c_{i,0} + t_i c_{i,2}) \\ &- t_i^3 c_{i,2}\big] / (3a_i(1-t_i)t_i). \end{aligned} \tag{6}$$

Substituting this into the derivative of Eq. (3), and letting it equal 0, we obtain (after some simplification) a polynomial equation of degree 9 in t_i. This equation can be derived by the Maxima [13] code in Figure 13 (Appendix A). We solve this equation and select a real root in $[0, 1]$. Note that we have proved that there is one and only one solution for the polynomial equation of degree 9 in $t_i \in [0, 1]$ as in the case of κ-curves; see details in [15], as well as Appendix A.2.

However, when we use other types of curves with more complicated representations (see examples in Appendix C), this kind of formula may be hard to derive. In these cases, we can use the relaxed Newton's method to compute the maximum curvature. For this, we need to be able to compute the curvature and its derivative; we do this using a quadratic Taylor series approximation around the last value of t_i.

We introduce the constant λ_i ($0 < \lambda_i < 1$) according to the construction method of κ-curves and set

$$c_{i,2} = c_{i+1,0} = (1-\lambda_i)c_{i,1} + \lambda_i c_{i+1,1}. \tag{7}$$

Let the curvatures at the endpoints of the curve segment be denoted by $\kappa_i(0; a_i)$ and $\kappa_i(1; a_i)$, then from Eq. (3)

$$\begin{aligned} \kappa_i(1; a_i) &= \frac{4}{3} \cdot \frac{(1-a_i)\triangle_i^+}{a_i^2 \lambda_i^2 \|c_{i+1,1} - c_{i,1}\|^3}, \\ \kappa_{i+1}(0; a_{i+1}) &= \frac{4}{3} \cdot \frac{(1-a_{i+1})\triangle_{i+1}^-}{a_{i+1}^2 (1-\lambda_i)^2 \|c_{i+1,1} - c_{i,1}\|^3}, \end{aligned} \tag{8}$$

introducing the notations $\triangle_i^+ = \triangle(c_{i,0}, c_{i,1}, c_{i+1,1})$ and $\triangle_i^- = \triangle(c_{i-1,1}, c_{i,1}, c_{i,2})$.

By adopting the local/global approach, we treat $c_{i,0}$ as fixed for the computation of $\kappa_i(1; a_i)$, although it depends on λ_{i-1} (similarly for $c_{i+1,2}$).

In order to guarantee G^2 continuity at the joint of two consecutive segments, the following equations should be satisfied:

$$\kappa_i(1; a_i) = \kappa_{i+1}(0; a_{i+1}). \tag{9}$$

Hence

$$\lambda_i = \frac{\sqrt{(1-a_i)\triangle_i^+}}{\sqrt{(1-a_i)\triangle_i^+} + \frac{a_i}{a_{i+1}}\sqrt{(1-a_{i+1})\triangle_{i+1}^-}}. \tag{10}$$

Since $0 < a_i, a_{i+1} < 1$, λ_i is real and $0 < \lambda_i < 1$.

3.2 Optimization

In the global phase, we calculate the positions of the middle control points $c_{i,1}$ by solving a linear system of equations. We treat the current values of λ_i (internal division ratios of $c_{i,1}$ and $c_{i+1,1}$) and t_i (parameters of local maximum curvature) as fixed.

Substituting Eq. (7) into Eq. (5), we get

$$\begin{aligned} p_i = {}&(1-t_i)^3 \left[(1-\lambda_{i-1})c_{i-1,1} + \lambda_{i-1}c_{i,1}\right] \\ &+ 3(1-t_i)^2 t_i \left[(1-a_i)\left[(1-\lambda_{i-1})c_{i-1,1} + \lambda_{i-1}c_{i,1}\right] \right. \\ &\qquad \left. + a_i c_{i,1}\right] \\ &+ 3(1-t_i)t_i^2 \left[(1-a_i)\left[(1-\lambda_i)c_{i,1} + \lambda_i c_{i+1,1}\right] \right. \\ &\qquad \left. + a_i c_{i,1}\right] \\ &+ t_i^3 \left[(1-\lambda_i)c_{i,1} + \lambda_i c_{i+1,1}\right], \end{aligned} \tag{11}$$

which can be solved for $c_{i,1}$.

The optimization process is summarized in Algorithm 1.

Algorithm 1: The optimization process.

Result: control points $c_{i,k}$ and parameters of maximal curvature t_i
Set all λ_i to 0.5;
Compute all $c_{i,0}$ and $c_{i,2}$ by Eq. (7);
while *not convergent* **do**
    Compute all λ_i by Eq. (10);
    Compute all $c_{i,0}$ and $c_{i,2}$ by Eq. (7);
    Compute all t_i by polynomial root finding;
    Compute all $c_{i,1}$ by Eq. (11);
end
Compute all $c_{i,0}$ and $c_{i,2}$ by Eq. (7);

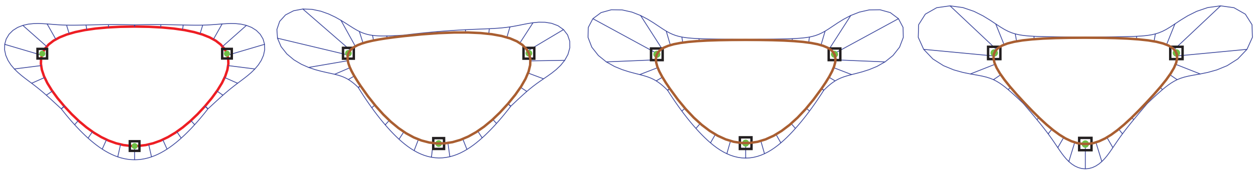

Fig. 5: The leftmost curve is a κ-curve. The other curves are $\epsilon\kappa$-curves: the a_i values are equal to 2/3 except for one, two and three input points, respectively, where $a_i = 0.85$.

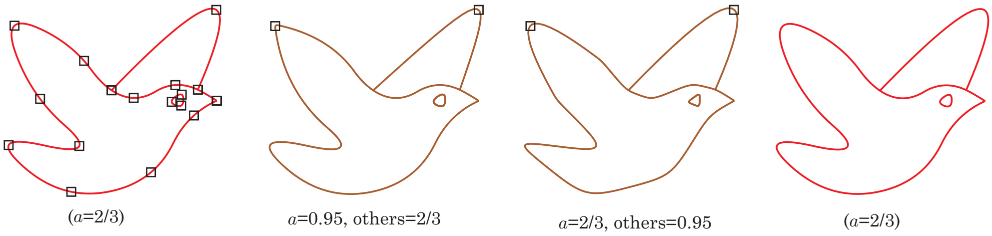

Fig. 6: The a values of the two wing tips in the second figure from the left are 0.95, and those of the other input points are 2/3; note the sharpening of the wings. In the third figure, the roles are reversed. The left- and rightmost curves in red are κ-curves with the same input points.

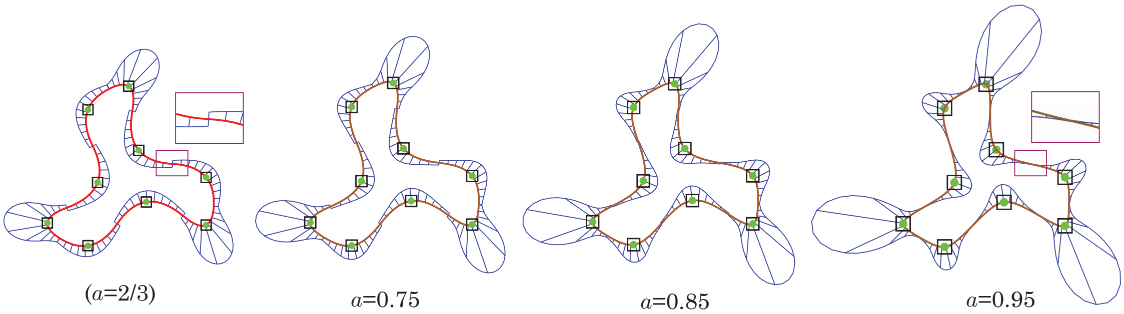

Fig. 7: The leftmost curve is a κ-curve; the other curves are $\epsilon\kappa$-curves: a is equal to 0.75, 0.85 and 0.95, respectively. Note the curvature at the inflection points.

3.3 Results

Figure 5 shows examples of closed $\epsilon\kappa$-curves along with the original κ-curve. The input points are located at the same positions. The a_i values of these curves are equal to 2/3, except for one, two and three input points, respectively, where a_i is set to 0.85. If the a_i of all input points are 2/3, the κ-curve on the left is generated. Since we specify a larger value for some input points, the magnitudes of the corresponding local maximum curvature increase, as we expected.

Figure 6 shows another example of local curvature control. From the left to right, the first drawing shows a bird using κ-curves. In the second we set a at the wing tips to 0.95, while leaving all others at the default 2/3. This has the effect of sharpening the wing tips. In the third figure, we reversed the role of the input points, giving $a = 2/3$ to those at the wing tips, and 0.95 to all other points. Here the wings are rounded, while other parts of the bird get sharper.

Notice that the bird's beak resembles a cusp, but is actually the start and end points of an open curve located at the same position. In our implementation, we limit $2/3 \leq a \leq 1$ to make a curve with smooth curvature distribution, which disallows the generation of a cusp even at $a = 1$. In cases where the designer wants to use a cusp, the curve should be cut in two, or the input points should be relocated to form a cusp as explained in [27].

Figure 7 shows examples of global curvature control. There are three $\epsilon\kappa$-curves, with a set to 0.75, 0.85 and 0.95, respectively, along with the original κ-curve ($a = 2/3$) for comparison. By increasing a, the magnitudes of local maximum curvature increase. As the close-up windows indicate, at the inflection point G^2 continuity is violated for κ-curves, and only G^1 continuity is guaranteed. However for $\epsilon\kappa$-curves with a larger a, the magnitude of curvature at inflection points, and consequently the G^2 error, become smaller. Note that although these curves are almost G^2-continuous everywhere, they are quite different from those in Figure 2 (generated using the same input points).

Figures 1, 8 and 9 show the effect of changes of the global shape parameter a on various designs. As discussed above, larger a values generally induce larger local curvature extrema and steeper curvature variation. The resulting curves look more sharp at the input points and more flat between them.

4 Generalized Trigonometric Basis

In this section, we describe our new *generalized trigonometric basis*. This is based on the trigonometric cubic Bernstein-like basis [23], which we are going to review first.

The trigonometric cubic Bernstein-like basis functions have an extra shape parameter α, and are defined by

$$\begin{aligned}
f_0 &= \alpha S^2 - \alpha S + C^2 = 1 + (\alpha - 1)S^2 - \alpha S, \\
f_1 &= \alpha S(1 - S), \\
f_2 &= \alpha(S^2 + C - 1) = \alpha C(1 - C), \\
f_3 &= (1 - \alpha)S^2 - \alpha C + \alpha = 1 + (\alpha - 1)C^2 - \alpha C,
\end{aligned} \tag{12}$$

where $S = \sin \frac{\pi t}{2}$, $C = \cos \frac{\pi t}{2}$, for $\alpha \in (0, 2)$, $t \in [0, 1]$. Note that these functions satisfy partition of unity, i.e., $\sum_{i=0}^{3} f_i(t) = 1$ for any α. When $\alpha = 1$, the above functions are simplified to

$$\begin{aligned}
f_0 &= 1 - S, \\
f_1 &= S(1 - S), \\
f_2 &= C(1 - C), \\
f_3 &= 1 - C.
\end{aligned} \tag{13}$$

If we add the second and third functions together and rename them to u, v and w, we obtain blending functions $\{u, v, w\}$ as follows:

$$\begin{aligned}
u &= 1 - S, \\
v &= S(1 - S) + C(1 - C) = S + C - 1, \\
w &= 1 - C.
\end{aligned} \tag{14}$$

It is straightforward to define a curve by these blending functions with three control points, which we can regard as a "linear" trigonometric curve since the highest degree the trigonometric functions are in is one.

One interesting relationship among these functions is

$$v^2 = 2uw, \tag{15}$$

which enables

$$(u + v + w)^2 = u^2 + 2uv + 4uw + 2vw + w^2, \tag{16}$$

and yields the five blending functions $\{u^2, 2uv, 4uw, 2vw, w^2\}$, associated with five control points. We can define a curve using these blending functions and regard it as a "quadratic" trigonometric curve since the highest power of each blending function is now degree two.

In a similar way, we can extend blending functions of "degree" n with $2n+1$ control points. As explained in Appendix B, we can perform a recursive procedure to evaluate a curve of any degree similar to de Casteljau's algorithm avoiding the overhead of trigonometric function evaluation. This means that it is not necessary to calculate the coefficients of blending functions, or keep a coefficient table.

We formulate the $\epsilon\kappa$-curve in this basis using a strategy similar to that in the previous section, i.e., using a sequence of quadratic trigonometric curves with a constraint on the positions of their control points, as shown in Figure 10. Note the location of the control point $P_2 = [(1 - a)Q_0 + 2aQ_1 + (1 - a)Q_2]/2$. The curve $c(t; a)$ is defined by

$$\begin{aligned}
c(t; a) =& u^2 Q_0 + 2uv [(1 - a)Q_0 + aQ_1] \\
&+ 2uw [(1 - a)(Q_0 + Q_2) + 2aQ_1] \\
&+ 2vw [aQ_1 + (1 - a)Q_2] + w^2 Q_2.
\end{aligned} \tag{17}$$

When a is equal to $1/2$, the curve degenerates to a linear trigonometric curve.

4.1 Geometric Constraints and Optimization

First, we analyze the linear trigonometric curve since it corresponds to the original κ-curve. Let $c_i(t)$ be a linear trigonometric curve with control points $c_{i,0}$, $c_{i,1}$ and $c_{i,2}$ and defined by

$$c_i(t) = (1 - S)c_{i,0} + (S + C - 1)c_{i,1} + (1 - C)c_{i,2}, \tag{18}$$

where $S = \sin \frac{\pi t}{2}$, $C = \cos \frac{\pi t}{2}$ and $t \in [0, 1]$. Its curvature is given by

$$\kappa_i(t) = \frac{2\Delta(c_{i,0}, c_{i,1}, c_{i,2})}{(C^2\|r_i\|^2 + 2CS\langle r_i, s_i\rangle + S^2\|s_i\|^2)^{\frac{3}{2}}}, \tag{19}$$

where $r_i = c_{i,1} - c_{i,0}$ and $s_i = c_{i,2} - c_{i,1}$. The numerator of the above formula does not depend on t, so the extrema of the following $f_i(t)$ corresponds to those of $\kappa_i(t)$:

$$f_i(t) = C^2\|r_i\|^2 + 2CS\langle r_i, s_i\rangle + S^2\|s_i\|^2, \tag{20}$$

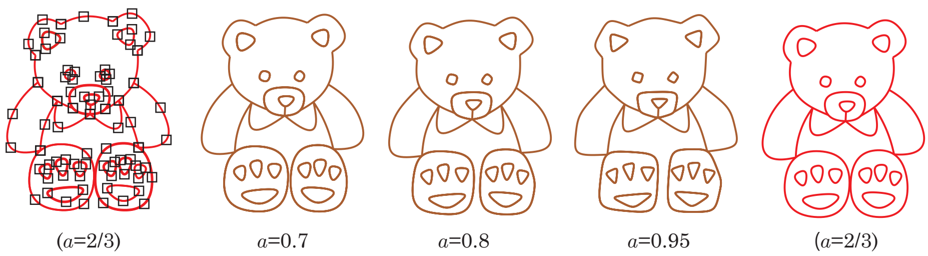

Fig. 8: Changing the global shape parameter in the bear model.

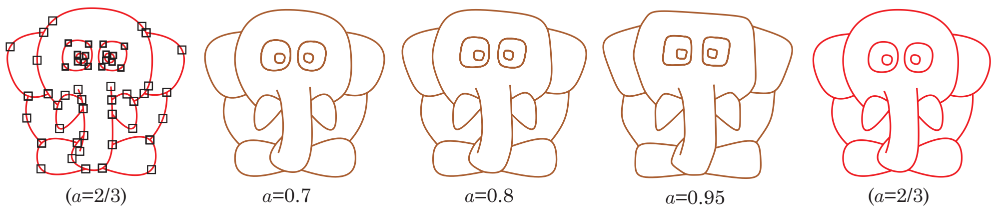

Fig. 9: Changing the global shape parameter in the elephant model.

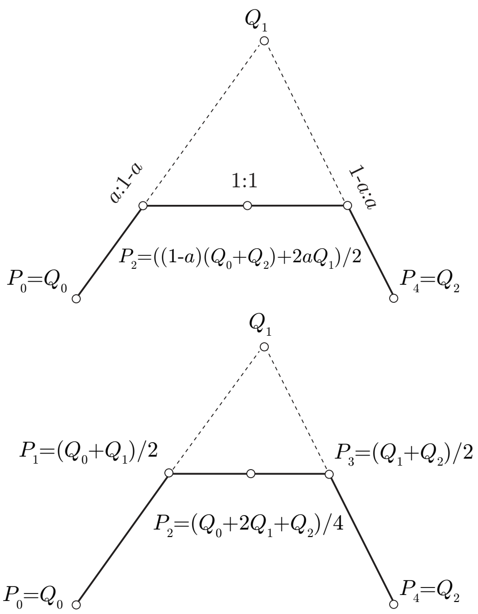

Fig. 10: Constrained quadratic trigonometric curve. When $a = 1/2$ (bottom), the curve becomes linear.

and its derivative with respect to t is given by

$$\frac{df_i(t)}{dt} = \pi(-CS\|r_i\|^2 + (-S^2 + C^2)\langle r_i, s_i\rangle + SC\|s_i\|^2). \tag{21}$$

By assuming $df_i(t)/dt = 0$ with $S, C \neq 0$, we obtain

$$S^2 - \gamma SC - C^2 = 0, \tag{22}$$

where $\gamma = (\|s_i\|^2 - \|r_i\|^2)/\langle r_i, s_i\rangle$. We can solve the above equation and obtain

$$C = \frac{-\gamma + \sqrt{\gamma^2 + 4}}{2} S = \beta S. \tag{23}$$

Since $0 \leq S, C \leq 1$, we have the unique solution

$$S = \frac{1}{\sqrt{\beta^2 + 1}}. \tag{24}$$

Hence

$$t = \frac{2}{\pi} \arcsin \frac{1}{\beta^2 + 1}. \tag{25}$$

Note that when r_i and s_i are perpendicular to each other, if $\|r_i\| = \|s_i\|$, then the curve becomes a circular arc, and no local maximum curvature exists. If $\|r_i\| > \|s_i\|$, then the curvature at $t = 1$ will be maximum and if $\|r_i\| < \|s_i\|$, the curvature at $t = 0$ will be maximum in this curve segment.

For a quadratic trigonometric curve, the curvatures $\kappa_i(1; a_i)$ and $\kappa_{i+1}(0; a_i)$ at the endpoints of the constrained quadratic trigonometric curve c_i are given by

$$\kappa_i(1; a_i) = \frac{1 - a_i}{a_i^2} \cdot \frac{\triangle_i^+}{\lambda_i^2 \|c_{i+1,1} - c_{i,1}\|^3},$$
$$\kappa_{i+1}(0; a_{i+1}) = \frac{1 - a_{i+1}}{a_{i+1}^2} \cdot \frac{\triangle_{i+1}^-}{(1 - \lambda_i)^2 \|c_{i+1,1} - c_{i,1}\|^3}. \tag{26}$$

We can calculate λ_i by guaranteeing G^2 continuity at the joint of $c_i(1; a_i)$ and $c_{i+1}(0; a_{i+1})$:

$$\lambda_i = \frac{\sqrt{(1 - a_i)\triangle_i^-}}{\sqrt{(1 - a_i)\triangle_i^-} + \frac{a_i}{a_{i+1}}\sqrt{(1 - a_{i+1})\triangle_{i+1}^+}}. \tag{27}$$

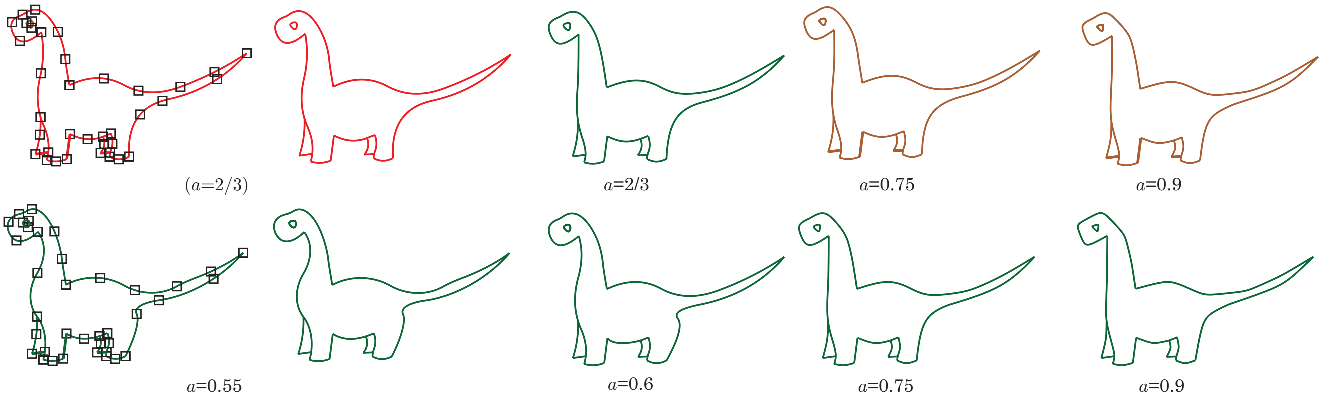

Fig. 11: The two curves on the top left (red) are κ-curves; the two on the top right (brown) are $\epsilon\kappa$-curves using cubic Bernstein basis functions with $a = 0.75$ and 0.9. The bottom row shows $\epsilon\kappa$-curves using quadratic trigonometric basis functions with $a = 0.55$, 0.6, 0.75 and 0.9.

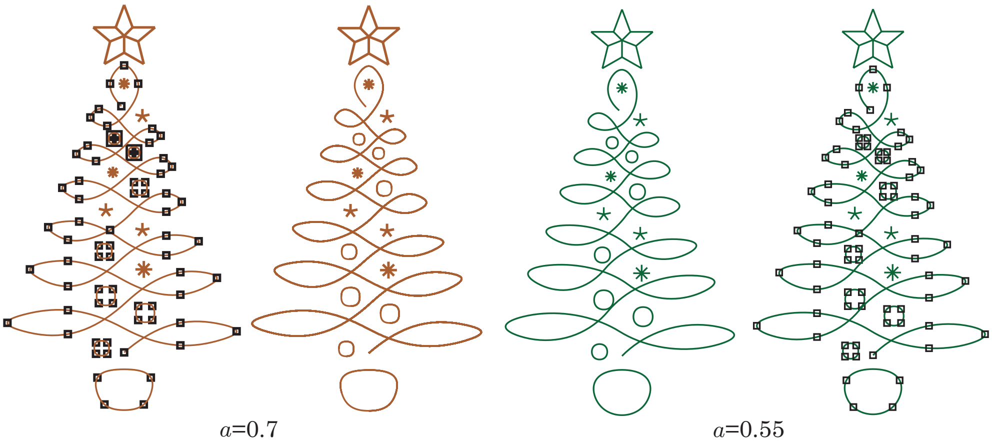

Fig. 12: A Christmas tree drawn with $\epsilon\kappa$-curves using the cubic Bernstein basis functions (left) and the quadratic trigonometric basis functions (right). Note that the latter has more rounded forms and—in this case—preferable.

As before, we get a linear system of equations for $c_{i,1}$:

$$\begin{aligned}
p_i &= u_i^2((1-\lambda_{i-1})c_{i-1,1} + \lambda_{i-1}c_{i,1}) \\
&+ 2u_i v_i((1-a_i)(1-\lambda_{i-1})c_{i-1,1} + ((1-a_i)\lambda_{i-1} + a_i)c_{i,1}) \\
&+ 2u_i w_i((1-a_i)(1-\lambda_{i-1})c_{i-1,1} + ((1-a_i)\lambda_{i-1} \\
&+ 2a_i + (1-a_i)(1-\lambda_i))c_{i,1} + (1-a_i)\lambda_i c_{i+1,1}) \\
&+ 2v_i w_i((a_i + (1-a_i)(1-\lambda_i))c_{i,1} + (1-a_i)\lambda_i c_{i+1,1}) \\
&+ w_i^2((1-\lambda_i)c_{i,1} + \lambda_i c_{i+1,1}),
\end{aligned} \tag{28}$$

where $u_i = 1 - \sin\frac{\pi t_i}{2}$, $v_i = \sin\frac{\pi t_i}{2} + \cos\frac{\pi t_i}{2} - 1$ and $w_i = 1 - \cos\frac{\pi t_i}{2}$.

4.2 Results

Figures 11 and 12 show examples of $\epsilon\kappa$-curves using the quadratic trigonometric basis functions explained in this section. In the first figure, the top left two curves (red) are κ-curves. The top right two curves (brown) are $\epsilon\kappa$-curves, using cubic Bernstein basis functions with $a = 0.75$ and 0.9. The bottom row (green) shows $\epsilon\kappa$-curves using quadratic trigonometric basis functions with $a = 0.55$, 0.6, 0.75 and 0.9. The curves in the bottom row are more rounded than those of the Bernstein basis. By increasing a, the differences between the two types of curves become smaller as they approach a polyline generated by connecting the input points.

The second figure shows a case where these more rounded forms are clearly preferable.

4.3 Use of Built-in Shape Parameters

So far our strategy made use of degree elevation – from quadratic to cubic in the polynomial case, and from linear to quadratic in the trigonometric case. Yet another strategy is to use extra shape parameters built into the basis functions.

As an illustrative example, take the trigonometric cubic Bernstein-like basis functions reviewed in the previous section. In the framework of $\epsilon\kappa$-curves, we have to degenerate the curve by relocating the positions of its control points, essentially reducing its degree.

The trigonometric cubic Bernstein-like basis functions need four control points to define a curve. To construct "quadratic" curves corresponding to the quadratic Bézier segments of κ-curves, we make the second and third control points collocate. Hence the blending functions become

$$\begin{aligned} b_0(t;\alpha) &= 1 + (\alpha-1)S^2 - \alpha S, \\ b_1(t;\alpha) &= \alpha(S + C - 1), \\ b_2(t;\alpha) &= 1 + (\alpha-1)C^2 - \alpha C, \end{aligned} \tag{29}$$

where $S = \sin\frac{\pi t}{2}$, $C = \cos\frac{\pi t}{2}$, for $\alpha \in (0,2)$, $t \in [0,1]$. Incidentally, this will result in the same curve as Eq. (17), if we substitute $2a$ for α.

Note that several curve types – such as the cubic alternative curve and the $\alpha\beta$-Bernstein-like basis functions, for specific extra parameters – have zero curvature at the endpoints, so we cannot obtain the internal division ratio λ_i. For these curves, the method shown in this section cannot be applied. See details in Appendix C.

5 Conclusions and Future Work

We have proposed two types of $\epsilon\kappa$-curves as extensions of κ-curves, for controlling the magnitudes of local maximum curvature. Our methods use degree elevation of the Bernstein basis functions, and a new family of trigonometric basis functions.

In line with Yan et al.'s paradigm shift for curve generation we consider the local maximum curvature in the middle, instead of focusing on the endpoints. The new curves preserve all of the nice properties of κ-curves, i.e., point interpolation, G^2 continuity (except at inflection points), continuous modification (changes smoothly when the input points move), and local influence. Computing $\epsilon\kappa$-curves is quite fast, and designers can manipulate them interactively, acquiring much more expressive power for curve design, as illustrated by our examples. The processing time for generating $\epsilon\kappa$-curves is similar to that of κ-curves, especially for $\epsilon\kappa$-curves with cubic Bernstein basis. For example, κ-curves consume 0.07 sec to draw Figure 8 (bear). Under similar conditions, cubic Bézier $\epsilon\kappa$-curves consume 0.08 sec and generalized trigonometric $\epsilon\kappa$-curves takes 0.37 sec on average.

Future work includes the development of $\epsilon\kappa$-curve plug-ins for Adobe Illustrator® and Photoshop®. Another possible research direction is to apply the proposed method to different types of aesthetic curves, e.g., log-aesthetic curves [25], σ-curves [17] or τ-curves [18].

Acknowledgement

This work was supported by JST CREST (No. JPMJCR1911); JSPS Grant-in-Aid for Scientific Research (B, No. 19H02048); JSPS Grant-in-Aid for Challenging Exploratory Research (No. 26630038); Solutions and Foundation Integrated Research Program; ImPACT Program of the Council for Science, Technology and Innovation; and the Hungarian Scientific Research Fund (OTKA, No. 124727). The authors acknowledge the support by 2016, 2018 and 2019 IMI Joint Use Program Short-term Joint Research "Differential Geometry and Discrete Differential Geometry for Industrial Design" (September 2016, September 2018 and September 2019). The second author acknowledges University Malaysia Terengganu for approving sabbatical leave which was utilized to work on emerging researches, including this work.

References

1. Attneave, F.: Some informational aspects of visual perception. Psychological Review **61**, 183–193 (1954)
2. Barsky, B.A., Beatty, J.C.: Local control of bias and tension in beta-splines. ACM Transactions on Graphics **2**(2), 109–134 (1983)
3. do Carmo, M.P.: Differential geometry of curves and surfaces. Prentice-Hall (1976)
4. Chelius, C.: Using the curvature tool in Adobe Illustrator (2016). URL https://web.archive.org/web/20161023130701/ https://creativepro.com/curvature-tool-adobe-illustrator/
5. Chen, Z., Huang, J., Cao, J.C., Zhang, Y.J.: Interpolatory curve modeling with feature points control. Computer-Aided Design **114**, 155–163 (2019)
6. Djudjic, D.: Photoshop CC officially gets curvature pen tool and other improvementsts (2017). URL https://web.archive.org/web/20200620025914/ https://www.diyphotography.net/photoshop-cc-officially-gets-curvature-pen-tool-improvements/
7. Farin, G.: Curves and surfaces for CAGD. Morgan-Kaufmann (2001)
8. Han, X.A., Ma, Y., Huang, X.: The cubic trigonometric Bézier curve with two shape parameters. Applied Mathematics Letters **22**(2), 226–231 (2009)
9. Jamaludin, M., Said, H., Majid, A.: Shape control of parametric cubic curves. In: Proceeding of the 4th International Conference on CAD & CG, pp. 161–167 (2001)
10. Julia: The Julia programming language (2020). URL https://julialang.org/

11. Levien, R., Séquin, C.H.: Interpolating splines: Which is the fairest of them all? Computer-Aided Design and Applications **6**(1), 91–102 (2009)
12. Liu, L., Zhang, L., Xu, Y., Gotsman, C., Gortler, S.J.: A local/global approach to mesh parameterization. In: Proceedings of the Symposium on Geometry Processing, pp. 1495–1504 (2008)
13. Maxima: A computer algebra system. version 5.44.0 (2020). URL http://maxima.sourceforge.net/
14. Miura, K.T.: A general equation of aesthetic curves and its self-affinity. Computer-Aided Design and Applications **3**(1-4), 457–464 (2006)
15. Miura, K.T., Salvi, P.: On the curvature extrema of special cubic Bézier curves (2021). URL https://arxiv.org/abs/2101.08138
16. Miura, K.T., Shibuya, D., Gobithaasan, R.U., Usuki, S.: Designing log-aesthetic splines with G^2 continuity. Computer-Aided Design and Applications **10**(6), 1021–1032 (2013)
17. Miura, K.T., Suzuki, S., Gobithaasan, R.U., Usuki, S., Inoguchi, J., Sato, M., Kajiwara, K., Shimizu, Y.: Fairness metric of plane curves defined with similarity geometry invariants. Computer-Aided Design and Applications **15**(2), 256–263 (2018)
18. Miura, K.T., Suzuki, S., Usuki, S., Gobithaasan, R.U.: τ-curve – Introduction of cusps to aesthetic curves. Journal of Computational Design and Engineering **7**(2), 155–164 (2020)
19. Norman, J.F., Phillips, F., Ross, H.E.: Information concentration along the boundary contours of naturally shaped solid objects. Perception **30**, 1285–1294 (2001)
20. Said, H.: Generalized ball curve and its recursive algorithm. ACM Transactions on Graphics **8**, 360–371 (1989)
21. Salvi, P.: Implementation of $\epsilon\kappa$-curves (2020). URL https://github.com/salvipeter/ekcurves/
22. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of the Symposium on Geometry Processing, pp. 109–116 (2007)
23. Usman, M.M., Abbas, M., Miura, K.T.: Some engineering applications of new trigonometric cubic Bézier-like curves to free-form complex curve modeling. Journal of Advanced Mechanical Design, Systems, and Manufacturing **14**(4), 1 (2020)
24. Walton, D.J., Meek, D.S.: Curvature extrema of planar parametric polynomial cubic curves. Journal of Computational and Applied Mathematics **134**, 69–83 (2001)
25. Wang, D., Gobithaasan, R.U., Sekine, T., Usuki, S., Miura, K.T.: Interpolation of point sequences with extremum of curvature by log-aesthetic curves with G^2 continuity. Computer-Aided Design and Applications **18**(2), 399–410 (2021)
26. Yan, Z., Schiller, S., Schaefer, S.: Circle reproduction with interpolatory curves at local maximal curvature points. Computer Aided Geometric Design **72**(6), 98–110 (2019)
27. Yan, Z., Schiller, S., Wilensky, G., Carr, N., Schaefer, S.: κ-curves: Interpolation at local maximum curvature. ACM Transactions on Graphics **36**(4), Article 129 (2017)
28. Yuksel, C.: A class of C^2 interpolating splines. ACM Transactions on Graphics **39**(5) (2020)
29. Zhang, G., Wang, K.: Quasi-cubic trigonometric curve and surface. Preprint (2019)
30. Zhang, J.: C-curves: An extension of cubic curves. Computer Aided Geometric Design **13**, 360–371 (1996)
31. Zhu, Y., Han, X., Liu, S.: Curve construction based on four $\alpha\beta$-Bernstein-like basis functions. Journal of Computational and Applied Mathematics **272**, 160–181 (2015)
32. Ziatdinov, R.: Visual perception, quantity of information function and the concept of the quantity of information continuous splines. Scientific Visualization **8**(1), 168–178 (2016)

A Local Maximum Curvature of Cubic Polynomial Curves

A.1 General Case

The signed curvature $\kappa(t)$ of a cubic polynomial curve $c(t) = (x(t), y(t))$ is given by [3] as

$$\kappa(t) = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{\frac{3}{2}}}, \tag{30}$$

where $x' = \mathrm{d}x(t)/\mathrm{d}t$, $x'' = \mathrm{d}^2x(t)/\mathrm{d}t^2$, and higher derivatives are expressed in a similar way.

At the extremum $\mathrm{d}\kappa(t)/\mathrm{d}t = 0$ and $\mathrm{d}\kappa(t)^2/\mathrm{d}t = 2\kappa(t) \cdot \mathrm{d}\kappa(t)/\mathrm{d}t$. Hence, if we exclude points where $\kappa(t) = 0$ we can obtain t values at the extrema.

By differentiating

$$\kappa(t)^2 = \frac{(x'(t)y''(t) - x''(t)y'(t))^2}{(x'(t)^2 + y'(t)^2)^3} \tag{31}$$

with respect to t, we obtain

$$\begin{aligned}\left(x'^2 + y'^2\right)^4 \kappa \frac{\mathrm{d}\kappa}{\mathrm{d}t} = (x''y' - x'y'')\big((x'''y' - x'y''')(x'^2 + y'^2) \\ - 3(x'x'' + y'y'')(x''y' - x'y'')\big),\end{aligned} \tag{32}$$

where $x'y'' - x''y' = 0$ means the curvature is equal to 0, and it corresponds to inflection points. Consequently,

$$(x'''y' - x'y''')(x'^2 + y'^2) - 3(x'x'' + y'y'')(x''y' - x'y'') = 0 \tag{33}$$

corresponds to curvature extrema. The above equation is regarded as (quadratic × quartic) − (cubic × cubic), but the coefficient of sextic terms vanishes and it becomes quintic. Therefore a cubic polynomial curve can have at most 5 curvature extrema. Please refer to [24] for more details. Since Eq. (33) is quintic, it is generally not possible to have analytical solutions, so we must use numerical approach to obtain the points on the curve where curvature extrema occurs.

A.2 Uniqueness

Here we give a high-level summary of the proof [15] that the curvature of a cubic curve of the form shown in Eq. (1) has at most one local extremum in the $(0, 1)$ parameter interval. As a consequence, the degree-9 polynomial in Section 3.1, computed by the Maxima [13] program in Fig. 13, has at most one real solution in the $(0, 1)$ interval. When it has none, this means that no local extremum is present in the curvature, so the extremum occurs at

$$t_0 = \operatorname*{arg\,max}_{t=0,1} |\kappa(t)|. \tag{34}$$

Without loss of generality, place the control points as follows:

$$Q_0 = (-1, 0), \qquad Q_1 = (b, h), \qquad Q_2 = (1, 0), \tag{35}$$

where $b \geq 0$ and $h > 0$. (The special cases where $Q_0 = Q_2$ or $h = 0$ are also handled in [15].)

Let $N(t, a)$ denote the left side of Eq. (33) applied to this curve. When $b \leq 3-2/a$, it is easy to see that $\partial N(t, a)/\partial t < 0$ for any $t \in (0, 1)$ and $a \in [2/3, 1]$, so N is decreasing. Since $N(0, a)$ is always positive, this means N has at most one 0-crossing.

In the following, let us also assume that $b > 3-2/a$. Then the following statements can also be proven:

$$N(1, a) < 0 \qquad \text{when } \partial N(1, a)/\partial t > 0, \tag{36}$$
$$\partial N(0, a)/\partial t < 0, \tag{37}$$
$$\partial N(t, a)/\partial t < 0 \qquad \text{when } \partial N(1, a)/\partial t < 0, \tag{38}$$
$$\partial^2 N(0, a)/\partial t^2 > 0, \tag{39}$$
$$\partial^3 N(t, a)/\partial t^3 < 0. \tag{40}$$

From the above it is easy to prove that $N(t, a) = 0$ has exactly one solution—and thus the curvature has at most one local extremum—in $(0, 1)$.

B Generalized Trigonometric Basis Functions

B.1 Recursive evaluation

For our new trigonometric basis, we can derive a recursive algorithm similar to de Casteljau's algorithm. For simplicity we explain only the quadratic case, but it can be extended to a general degree n by induction. To shorten expressions, we use $u = 1-S(t)$, $v = S(t)+C(t)-1$ and $w = 1-C(t)$, where $S(t) = \sin\frac{\pi t}{2}$ and $C(t) = \cos\frac{\pi t}{2}$. Note that $v^2 = 2uw$, and

$$\begin{aligned}(u + v + w)^2 &= \\ &u(u + v + w) + v(u + v + w) + w(u + v + w).\end{aligned} \tag{41}$$

For a quadratic curve with this basis, five control points P_i $(i = 0 \dots 4)$ are used, and the curve point at t is evaluated as

$$\begin{bmatrix} u & v & w \end{bmatrix} \begin{bmatrix} P_0 & P_1 & P_2 \\ P_1 & P_2 & P_3 \\ P_2 & P_3 & P_4 \end{bmatrix} \begin{bmatrix} u & & \\ & v & \\ & & w \end{bmatrix}. \tag{42}$$

Hence the algorithm repeats a simple blending of three points $uP_{i-1} + vP_i + wP_{i+1}$ to generate a point on the given curve.

B.2 Triangle method

We can also construct a triangle using the coefficients of trigonometric basis functions, similarly to Pascal's triangle. Below is a table of degree elevation, from the first row representing degree 1 to the sixth row representing degree 6:

$$\begin{array}{ccccccccccccc} & & & & & 1 & 1 & 1 & & & & & \\ & & & & 1 & 2 & 4 & 2 & 1 & & & & \\ & & & 1 & 3 & 9 & 8 & 9 & 3 & 1 & & & \\ & & 1 & 4 & 16 & 20 & 34 & 20 & 16 & 4 & 1 & & \\ & 1 & 5 & 25 & 40 & 90 & 74 & 90 & 40 & 25 & 5 & 1 & \\ 1 & 6 & 36 & 70 & 195 & 204 & 328 & 204 & 195 & 70 & 36 & 6 & 1 \end{array} \tag{43}$$

C Various Basis Functions

Here we check the applicability of the bases listed in Section 2, except for the trigonometric cubic Bernstein-like basis functions [23], since that was already discussed in the paper.

C.1 C-Bézier curve [30]

The basis of the C-Bézier curve is $\{\sin t, \cos t, t, 1\}$, and the curve is defined by the following formula:

$$\begin{aligned} B_\alpha(t) &= Z_0(t)q_0 + Z_1(t)q_1 + Z_2(t)q_2 + Z_3(t)q_3 \\ &= \frac{1}{\alpha - S} \begin{bmatrix} \sin t \\ \cos t \\ t \\ 1 \end{bmatrix}^\top \begin{bmatrix} C & 1-C-M & M & -1 \\ -S & (\alpha-K)M & -KM & 0 \\ -1 & M & -M & 1 \\ \alpha & -(\alpha-K)M & KM & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \end{aligned}$$

Here α is a built-in shape parameter satisfying $0 < \alpha \leq \pi$, and $S = \sin\alpha$, $C = \cos\alpha$. The parameter of the curve is $t \in [0, \alpha]$, and

$$K = \frac{\alpha - S}{1 - C},$$
$$M = \begin{cases} 1 & \text{if } \alpha = \pi, \\ \frac{S}{\alpha - 2K} = \frac{S(1-C)}{2S - \alpha - \alpha C} & \text{if } 0 < \alpha < \pi. \end{cases}$$

We degenerate the curve by adding its second and third basis functions, i.e., placing the second and third control points at the same position. The curve is then defined by three control points. However, even if we vary α from 0 to π, the blending functions do not vary much, as shown in Figure 14. Therefore this type of curve is not suitable for changing the magnitude of local maximum curvature.

C.2 Cubic alternative curve [9]

The cubic alternative curve is similar to the cubic Bézier curve, and is defined by

$$Z(t) = F_0(t)P_0 + F_1(t)P_1 + F_2(t)P_2 + F_3(t)P_3, \quad 0 \leq t \leq 1,$$

where the basis functions $F_i(t)$, $t = 0 \dots 3$ are

$$\begin{aligned} F_0(t) &= (1-t)^2(1 + (2-\alpha)t), \\ F_1(t) &= \alpha(1-t)^2 t, \\ F_2(t) &= \beta t^2(1-t), \\ F_3(t) &= t^2(1 + (2-\beta)(1-t)). \end{aligned}$$

When $\alpha = \beta = 2$, the curve becomes the cubic Ball curve; for $\alpha = \beta = 3$, the classical cubic Bézier curve; and for $\alpha = \beta = 4$, the cubic Timmer curve. The basis functions are nonnegative when $0 \leq \alpha \leq 3$.

We assume that $\beta = \alpha$, and the curve is degenerated by adding the second and third blending functions. Hence

$$\begin{aligned} A_0(t;\alpha) &= (1-t)^2(1 + (2-\alpha)t), \\ A_1(t;\alpha) &= \alpha(1-t)t \\ &= 1 - (1-t)^2(1 + (2-\alpha)t) - t^2(1 + (2-\alpha)(1-t)), \\ A_2(t;\alpha) &= t^2(1 + (2-\alpha)(1-t)). \end{aligned}$$

We define the curve $c(t;\alpha)$ by

$$\begin{aligned} c(t;\alpha) &= A_0(t;\alpha)P_0 + A_1(t;\alpha)P_1 + A_2(t;\alpha)P_2 \\ &= A_0(t;\alpha)(P_0 - P_1) + P_1 + A_2(t;\alpha)(P_2 - P_1). \end{aligned}$$

```
/*
 Input:
   (x0,y0) and (x2,y2) - the endpoints
   (px,py)             - the point to interpolate at the maximal curvature
   a                   - alpha, the ratio for conversion between quadratic/cubic
 Output:
   <a 9th-degree polynomial in t, having one real solution in [0,1]>
 */

/* (cx,cy) is the curve, (dx,dy) and (ddx,ddy) are the 1st and 2nd derivatives */
cx: (1-t)^3*x0+3*(1-t)^2*t*((1-a)*x0+a*x1)+3*(1-t)*t^2*((1-a)*x2+a*x1)+t^3*x2$
cy: (1-t)^3*y0+3*(1-t)^2*t*((1-a)*y0+a*y1)+3*(1-t)*t^2*((1-a)*y2+a*y1)+t^3*y2$
dx: diff(cx,t)$
dy: diff(cy,t)$
ddx: diff(dx,t)$
ddy: diff(dy,t)$

/* n and d are the numerator and denominator of the curvature, respectively */
n: dx*ddy-ddx*dy$
d: (dx^2+dy^2)^(3/2)$

/* The numerator of the curvature's derivative; we need to solve dk = 0 */
dk: diff(n,t)*d-n*diff(d,t)$
/* Looking at factor(dk), we can see that there is some room for simplification */
dk1: factor(dk/(162*a*(x1*y2-x0*y2-x2*y1+x0*y1+x2*y0-x1*y0)*sqrt(dx^2+dy^2)))$
solution: rhs(solve(dk1,t)[1])$

/* (x1,y1) is set s.t. the curve interpolates (px,py) */
x1: (px-((1-t)^3+3*(1-t)^2*t*(1-a))*x0-(3*(1-t)*t^2*(1-a)+t^3)*x2)/(3*(1-t)*t*a)$
y1: (py-((1-t)^3+3*(1-t)^2*t*(1-a))*y0-(3*(1-t)*t^2*(1-a)+t^3)*y2)/(3*(1-t)*t*a)$

/* Generate a string representation that can be inserted in a program */
display2d: false$ /* programming-friendly output */
collectterms(expand(num(xthru(ev(solution,x1=x1,y1=y1)))),t);
```

Fig. 13: Maxima [13] code to calculate the degree-9 polynomial in Section 3.1.

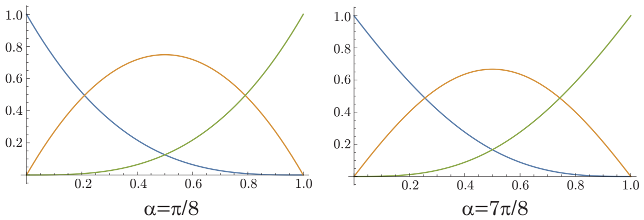

Fig. 14: Blending functions of the degenerated (quadratic) C-Bézier curve.

Then

$$\frac{dc(t;\alpha)}{dt} = (1-t)(-3\alpha t+6t+\alpha)(P_1-P_0) + t(3\alpha t-6t-2\alpha+6)(P_2-P_1),$$

$$\frac{d^2c(t;\alpha)}{dt^2} = 2\big[(3\alpha t-6t-2\alpha+3)(P_1-P_0) + (3\alpha t-6t-\alpha+3)(P_2-P_1)\big].$$

Hence the curvature $\kappa_i(t;\alpha)$ for each curve segment is given by

$$\kappa_i(t;\alpha) = \frac{2\alpha_i(3(\alpha-2)(1-t)t-\alpha_i+3)(P_1-P_0)\times(P_2-P_1)}{\|(1-t)(-3\alpha_i t+6t+\alpha_i)(P_1-P_0)+t(3\alpha_i t-6t-2\alpha_i+6)(P_2-P_1)\|^3}.$$

When $\alpha = 3$, this degenerates to the classical cubic Bézier curve with collocated second and third control points, and at the endpoints we get $\kappa_i(0,3) = \kappa_i(1,3) = 0$, since the directions of the first and second derivatives are the same. Hence, the proposed method is not applicable in this case.

When $\alpha \neq 0$,

$$\kappa_i(1;\alpha_i) = \frac{4(3-\alpha_i)\triangle_i^+}{\alpha_i^2\lambda_i^2\|c_{i+1,1}-c_{i,1}\|^3},$$

$$\kappa_{i+1}(0;\alpha_i) = \frac{4(3-\alpha_{i+1})\triangle_{i+1}^-}{\alpha_{i+1}^2(1-\lambda_i)^2\|c_{i+1,1}-c_{i,1}\|^3}.$$

To guarantee G^2 continuity at the joint of the segments, when $0 < \alpha_i, \alpha_{i+1} < 3$, we can compute λ_i $(0 < \lambda_i < 1)$ by

$$\lambda_i = \frac{\sqrt{(3-\alpha_i)\triangle_i^+}}{\sqrt{(3-\alpha_i)\triangle_i^+} + \frac{\alpha_i}{\alpha_{i+1}}\sqrt{(3-\alpha_{i+1})\triangle_{i+1}^-}}.$$

When $\alpha_i, \alpha_{i+1} > 3$,

$$\lambda_i = \frac{\sqrt{(\alpha_i - 3)\triangle_i^+}}{\sqrt{(\alpha_i - 3)\triangle_i^+} + \frac{\alpha_i}{\alpha_{i+1}}\sqrt{(\alpha_{i+1} - 3)\triangle_{i+1}^-}}.$$

In other cases, such as $\alpha_i > 3$, $0 < \alpha_{i+1} < 3$ or $0 < \alpha_i < 3$, $\alpha_{i+1} > 3$, λ_i can be determined by careful handling of the signs of $3 - \alpha_i$ and $3 - \alpha_{i+1}$.

By assuming a local maximum curvature at t_i, we can express the input point p_i as

$$\begin{aligned} p_i = {} & (1-t_i)^2(1+(2-\alpha_i)t_i)\left[(1-\lambda_{i-1})c_{i-1,1} + \lambda_{i-1}c_{i,1}\right] \\ & + \alpha_i(1-t_i)t_i c_{i,1} \\ & + t_i^2(1+(2-\alpha_i)(1-t_i))\left[(1-\lambda_i)c_{i,1} + \lambda_i c_{i+1,1}\right]. \end{aligned}$$

C.3 Cubic trigonometric Bézier curve [8]

For λ, $\mu \in [-2, 1]$, $t \in [0, 1]$, the cubic trigonometric Bézier denoted as T-Bézier basis functions are defined by

$$\begin{aligned} b_0(t) &= (1-S)^2(1-\lambda S), \\ b_1(t) &= S(1-S)(2+\lambda-\lambda S), \\ b_2(t) &= C(1-C)(2+\mu-\mu C), \\ b_3(t) &= (1-C)^2(1-\mu C), \end{aligned}$$

where $S = \sin\frac{\pi t}{2}$ and $C = \cos\frac{\pi t}{2}$.

We assume $\lambda = \mu = \alpha$, and the curve is degenerated by adding the second and third blending functions. Hence

$$\begin{aligned} A_0(t;\alpha) &= (1-S)^2(1-\alpha S), \\ A_1(t;\alpha) &= 1-(1-S)^2(1-\alpha S)-(1-C)^2(1-\alpha C), \\ A_2(t;\alpha) &= (1-C)^2(1-\alpha C). \end{aligned}$$

We define the curve $c(t;\alpha)$ by

$$\begin{aligned} c(t;\alpha) &= (1-S)^2(1-\alpha S)P_0 \\ &\quad + (1-(1-S)^2(1-\alpha S)-(1-C)^2(1-\alpha C))P_1 \\ &\quad + (1-C)^2(1-\alpha C)P_2 \\ &= (1-S)^2(1-\alpha S)(P_0-P_1) + P_1 \\ &\quad + (1-C)^2(1-\alpha C)(P_2-P_1). \end{aligned}$$

Then

$$\begin{aligned} \frac{dc(t;\alpha)}{dt} &= \frac{\pi}{2}\big[(1-S)C(2+\alpha-3\alpha S)(P_1-P_0) \\ &\quad + (1-C)S(2+\alpha-3\alpha C)(P_2-P_1)\big], \\ \frac{d^2c(t;\alpha)}{dt^2} &= \frac{\pi^2}{4}\big[(S-1)((2S+1)(2+\alpha-3\alpha S)+3\alpha C^2)(P_0-P_1) \\ &\quad + (1-C)((2C+1)(2+\alpha-3\alpha C)+3\alpha S^2)(P_2-P_1)\big]. \end{aligned}$$

Hence $dc/dt \times d^2c/dt^2$ is given by

$$\begin{aligned} \frac{dc}{dt} \times \frac{d^2c}{dt^2} &= (1-S)(1-C) \\ &\quad \big[C(2+\alpha-3\alpha S)((2C+1)(2+\alpha-3\alpha C)+3\alpha S^2) \\ &\quad + S(2+\alpha-3\alpha C)((2S+1)(2+\alpha-3\alpha S+3\alpha C^2)\big] \\ &\quad (P_1-P_0)\times(P_1-P_0). \end{aligned}$$

When $\alpha = 0$, this curve is the same as the trigonometric cubic Bernstein-like curve with $\alpha = 2$, and the curvatures at the start- and endpoints $\kappa(0)$ and $\kappa(1)$ are generally not equal to zero. For other α values, $\kappa(0) = \kappa(1) = 0$, so our method is not applicable. This is because the directions of the first and second derivatives are the same.

C.4 $\alpha\beta$-Bernstein-like basis functions [31] and quasi-cubic trigonometric Bernstein basis curves [29]

For arbitrary $\alpha, \beta \in [2, +\infty]$, and $t \in [0, 1]$, the $\alpha\beta$-Bernstein-like basis functions are defined by

$$\begin{aligned} A_0(t;\alpha) &= (1-t)^\alpha, \\ A_1(t;\alpha) &= 1-3t^2+2t^3-(1-t)^\alpha, \\ A_2(t;\beta) &= 3t^2-2t^3-t^\beta, \\ A_3(t;\beta) &= t^\beta. \end{aligned}$$

When $\alpha = \beta = 2$, these degenerate to cubic Said-Ball basis functions [20]. When $\alpha = \beta = 3$, these become cubic Bernstein basis functions.

Let $\beta = \alpha$ and we degenerate the curve by adding the second and third blending functions. Hence

$$\begin{aligned} A_0(t;\alpha) &= (1-t)^\alpha, \\ A_1(t;\alpha) &= 1-(1-t)^\alpha-t^\alpha, \\ A_2(t;\alpha) &= t^\alpha. \end{aligned}$$

We define the curve $c(t)$ by

$$\begin{aligned} c(t;\alpha) &= (1-t)^\alpha P_0 + (1-(1-t)^\alpha - t^\alpha)P_1 + t^\alpha P_2 \\ &= (1-t)^\alpha(P_0-P_1) + P_1 + t^\alpha(P_2-P_1). \end{aligned}$$

Then

$$\begin{aligned} \frac{dc(t;\alpha)}{dt} &= \alpha\left[(1-t)^{\alpha-1}(P_1-P_0) + t^{\alpha-1}(P_2-P_1)\right], \\ \frac{d^2c(t;\alpha)}{dt^2} &= \alpha(\alpha-1)\left[(1-t)^{\alpha-2}(P_0-P_1) + t^{\alpha-2}(P_2-P_1)\right]. \end{aligned}$$

Hence its curvature $\kappa(t;\alpha)$ is given by

$$\kappa(t;\alpha) = \frac{\alpha-1}{\alpha} \cdot \frac{(1-t)^{\alpha-2}t^{\alpha-2}(P_1-P_0)\times(P_2-P_1)}{\|(1-t)^{\alpha-1}(P_1-P_0)+t^{\alpha-1}(P_2-P_1)\|^3}.$$

When $\alpha = 2$ (cubic Said-Ball curve) the curvatures at the start- and endpoints $\kappa(0;\alpha)$ and $\kappa(1;\alpha)$ are generally not zero. However, when $\alpha > 2$, we get $\kappa(0;\alpha) = \kappa(1;\alpha) = 0$. Therefore in this case our method is not applicable.

In the case of quasi-cubic trigonometric Bernstein basis curves, the directions of the first and second derivatives at the ends are the same, thus our method is not applicable.