A Class of C^2 Interpolating Splines

CEM YUKSEL, University of Utah

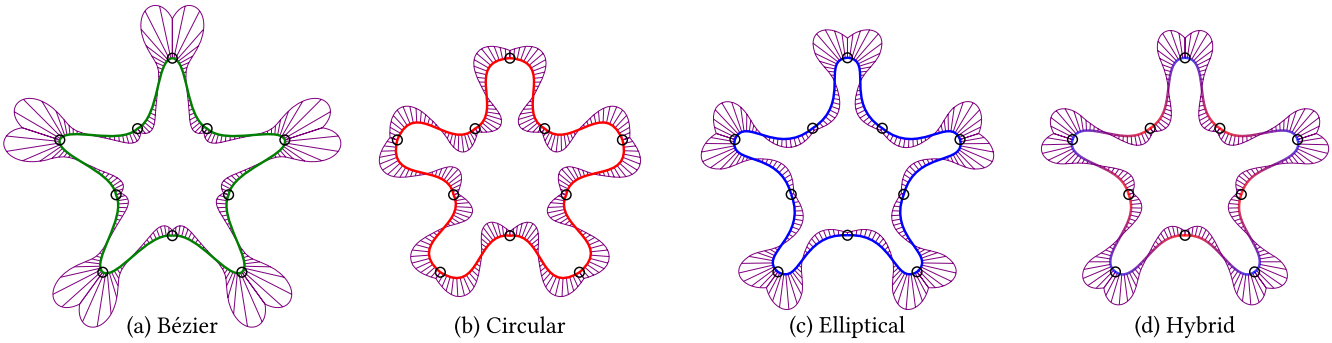

Fig. 1. Example curves generated from the same control points using our formulation with (a) Bézier interpolation function, (b) circular interpolation function, (c) elliptical interpolation function, and (d) hybrid (circular-elliptical) interpolation function. All curves have guaranteed C^2 continuity and local support, but they produce different shapes from the same control points. The purple lines indicate the curvature of the curves.

We present a class of non-polynomial parametric splines that interpolate the given control points and show that some curve types in this class have a set of highly desirable properties that were not previously demonstrated for interpolating curves before. In particular, the formulation of this class guarantees that the resulting curves have C^2 continuity everywhere and local support, such that only four control points define each curve segment between consecutive control points. These properties are achieved directly due to the mathematical formulation used for defining this class, without the need for a global numerical optimization step. We also provide four example spline types within this class. These examples show how guaranteed self-intersection-free curve segments can be achieved, regardless of the placement of control points, which has been a limitation of prior interpolating curve formulations. In addition, they present how perfect circular arcs and linear segments can be formed by splines within this class, which also have been challenging for prior methods of interpolating curves.

CCS Concepts: • **Computing methodologies → Parametric curve and surface models**;

Additional Key Words and Phrases: Splines, curves, curvature, circular arcs

ACM Reference format:
Cem Yuksel. 2020. A Class of C^2 Interpolating Splines. *ACM Trans. Graph.* 39, 5, Article 160 (August 2020), 14 pages.
https://doi.org/10.1145/3400301

Authors' address: C. Yuksel, University of Utah, 50 S. Central Campus Drive Room 3190, SLC UT 84112; email: cem@cemyuksel.com.

0730-0301/2020/08-ART160 $15.00
https://doi.org/10.1145/3400301

1 INTRODUCTION

Polynomial parametric splines have served us well in computer graphics. Among them, approximating splines (such as B-splines and NURBS) have been particularly popular, as they can provide important properties like C^2 continuity and local support, and they are not prone to producing unintended cusps and self-intersections. However, approximating splines do not go through the control points, which is a highly desirable property for some applications, and it is completely essential for others, such as the interpolation of animation keyframes.

Interpolating polynomial parametric splines, however, do go through the control points and thereby allow specifying exact positions on the curve. Yet, most interpolating splines are notoriously difficult to control, because the curve segments that connect consecutive control points can include unintended features like cusps and self-intersections that can be hard to avoid in practice. These problems are often exacerbated with C^2 continuous interpolation, which typically requires higher-order polynomials with each control point affecting a larger portion of the curve. Recent work addressed these long-standing problems that plagued interpolating splines by introducing costly numerical optimizations [Yan et al. 2017] with control points having global support (i.e., each control point affects the entire curve). Unfortunately, this approach not only leads to complex formulations for curves but also can be computationally expensive for some applications, and it is entirely impractical when a large number of curves are involved, such as hair modeling applications. Also, it makes it difficult to form linear segments, and, since these curves provide only G^2 continuity, they are not applicable when C^2 continuity is needed.

In this article, we present a class of interpolating splines with a different mathematical foundation. The splines in this class are formed by a trigonometric interpolation of an arbitrarily chosen *interpolation function* that connects three consecutive control points. The underlying formulation of this class provides a simple

mechanism for constructing custom curve types that can inherit the properties a given application needs by simply picking a suitable interpolation function. Most importantly, regardless of the chosen interpolation function, all curves in this class satisfy the following crucial properties:

- Guaranteed C^2 (thereby curvature) continuity everywhere,
- Local support with each curve segment being controlled by only four nearby control points, and
- Direct evaluation from control points without solving a global optimization problem.

We demonstrate the effectiveness of this formulation by presenting example interpolation functions (Figure 1). We show that by appropriately choosing the interpolation function, we can guarantee cusp-free and self-intersection-free curve segments between control points. Note that the only type of interpolating polynomial splines that can provide such a guarantee are the ones with only C^1 continuity or with global numerical optimizations (achieving G^2 continuity). We also show that it is possible to define an interpolation function such that the resulting curves can represent perfect circular (and elliptical) arcs, which has been a challenge for most interpolating splines. In addition, unlike polynomial curves with global support, it is easy to form perfectly linear segments with all example interpolation functions we present, since all curves in this class have local support. Furthermore, as the interpolation functions we present are continuous and do not involve solving an optimization problem that can lead to numerical instability, continuous control point motion leads to continuous curve deformation. Moreover, we show that it is possible to design an interpolation function with well-defined bounds (i.e., distance to the control polygon) and, while the convex hull property does not apply to interpolating splines,[1] we explain that bounds of the curve segments between control points can be defined using the bounds of the interpolation function.

Contributions: Unlike similar curve formulations in prior work, we show that our formulation can lead to curve types with important properties, such as guaranteed self-intersection-free segments, in addition to having local support, C^2 continuity everywhere, bounded distance to the control polygon, and the ability to form perfectly linear and circular/elliptical segments, all of which are satisfied in higher dimensions as well. To our knowledge, no prior curve formulation can satisfy this set of properties. As such, two of the curve types we represent in this article are strong alternatives to replace existing curve formulations for various graphics applications, providing interpolating curves with the set of properties mentioned above. Though the significance of some of these properties can be application-dependent, we do not specifically target a particular application for our evaluations in this article.

2 RELATED WORK

There is a large body of work in computer graphics on interpolating curve constructions [Hoschek and Lasser 1993]. Here we briefly discuss some of the more common representations.

Catmull–Rom splines [Barry and Goldman 1988; Catmull and Rom 1974], combining Lagrange interpolation with B-spline basis functions, are one of the most popular formulations for interpolating curves. Subdivision curves [Deslauriers and Dubuc 1989; Dyn et al. 1987] can be used for representing interpolating curves as well, and it is possible to approximate circles [Sabin and Dodgson 2005]. Interpolating B-splines can be formed by solving a tridiagonal system of equations [Farin 2002]. None of these formulations, however, can guarantee cusp-free curves with C^2 continuity, and cusps and self-intersections appear when the distances between control points have a significant-enough variation. In the case of C^1 Catmull–Rom curves, however, centripetal parameterization can uniquely guarantee no self-intersections within curve segments between consecutive control points [Yuksel et al. 2009a, 2011].

Due to the difficulties of avoiding unintended self-intersections with interpolating curves, Bézier curves, which only interpolate the first and the last control points, are more commonly used in practice. In particular, cubic Bézier curves with four control points are ubiquitous in computer graphics applications. While most applications allow the user to specify all Bézier control points, C^2 splines with monotonic curvature can be formed by restricting the locations of internal control points [Higashi et al. 1988]. Class A Bézier curves [Farin 2006; Mineur et al. 1998] achieve monotonic curvature by limiting the degree of freedom. Log-aesthetic curves [Miura and Gobithaasan 2014; Miura et al. 2013; Yoshida et al. 2009; Yoshida and Saito 2017] and clothoids [Havemann et al. 2013; McCrae and Singh 2009; Schneider and Kobbelt 2000] produce curves with monotonic curvature, which is desirable in some design applications, but their generation involves iterative optimization processes for placing the internal Bézier control points. Moreover, depending on the configuration of the interpolated control points, minor variations in control point positions can lead to discontinuous changes in the generated curve. Recently, κ-curves [Yan et al. 2017] were introduced for generating piecewise-quadratic Bézier curves with automatically-placed Bézier control points using an iterative global optimization process. κ-curves have G^1 continuity at inflection points and G^2 continuity everywhere else, and the local curvature maxima are at control points. While control points have global support with κ-curves, they tend to have more significant influence in a smaller region. Nonetheless, global support makes it difficult to form perfectly linear segments with these curves. Though κ-curves appear to be effective in simple two-dimensional (2D) drawing applications, they are not as suitable for some other applications, since they cannot provide C^2 continuity and require a relatively expensive global optimization. An extension of this approach uses a rational quadratic Bézier formulation to form circular segments [Yan et al. 2019]. In comparison, the curve formulation we present in this article ensures C^2 continuity everywhere, provides local support, and allows direct evaluation from the control points without solving a global optimization problem. Other properties of the curves in this new class depend on the chosen interpolation function.

Our curve construction is similar in spirit to the formulation of Overhauser [1968] that linearly blends two parabolas with G^1 continuity. This approach was also used for blending circular arcs and lines [Pobegailo 1992; Wenz 1996] and higher-degree

[1] Any formulation for interpolating splines must produce curves that are not bounded by the convex hull of the control polygon, unless the curve has only C^0 continuity.

polynomial curves to achieve C^2 continuity [Wiltsche 2005]. An alternative formulation uses B-spline basis functions for blending Lagrange [Röschel 1997] or Hermite [Gfrerrer and Röschel 2001] interpolants. More recently, Juhász and Róth [2014] extended the blending formulation for inserting additional control points to produce a trigonometric curve that interpolates the given control points with the desired degree of continuity. More similar to our formulation, Szilvási-Nagy and Vendel [2000] and Séquin et al. [2005] independently proposed interpolating circular arcs with a trigonometric blending function, identical to ours, and Sun and Zhao [2009] presented a similar formulation with rational quadratic Bézier interpolation that can form conic sections. Pobegailo [2013] proposed an alternative blending formulation using Bernstein polynomials. Unfortunately, none of these curve types that provide C^2 continuity prevent self-intersecting curve segments. Thus, they inherit this crucial limitation that plagued interpolating curve formulations and hindered their use in practice. The curve types we present in this article show that these ideas can be extended to form interpolating curves with self-intersection-free segment by carefully choosing an interpolation function.

3 THE SPLINE FORMULATION

The interpolating spline formulation we describe in this article relies on *interpolation functions*. The role of an interpolation function is to define a curve that goes through three consecutive control points. Let $\mathbf{F}_i$ be an interpolation function defining a curve that goes through control points $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, and $\mathbf{p}_{i+1}$. The interpolation functions are constrained so that

$$\mathbf{F}_i(0) = \mathbf{p}_{i-1}, \qquad \mathbf{F}_i(\tfrac{\pi}{2}) = \mathbf{p}_i, \qquad \mathbf{F}_i(\pi) = \mathbf{p}_{i+1}. \tag{1}$$

The curve segment $\mathbf{C}_i$ that interpolates the control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ is constructed by *blending* these two interpolation functions $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$. We use trigonometric interpolation for our blending function, such that

$$\mathbf{C}_i(\theta) = \cos^2\theta\, \mathbf{F}_i(\theta + \tfrac{\pi}{2}) + \sin^2\theta\, \mathbf{F}_{i+1}(\theta), \tag{2}$$

where $\theta \in [0, \frac{\pi}{2}]$ is a normalized parameter value.

We show that this curve formulation guarantees C^2 continuity regardless of the chosen interpolation function, starting with C^0. By definition, $\mathbf{F}_{i+1}$ interpolates control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ as well. Therefore, $\mathbf{C}_i$ provides an interpolating curve formulation with C^0 continuity, since $\mathbf{C}_i(\frac{\pi}{2}) = \mathbf{C}_{i+1}(0) = \mathbf{p}_{i+1}$.

For C^1 continuity, we must test the derivative of this curve

$$\begin{aligned}\mathbf{C}_i'(\theta) = {} & 2\cos\theta\sin\theta\, \left(\mathbf{F}_{i+1}(\theta) - \mathbf{F}_i(\theta + \tfrac{\pi}{2})\right) \\ & + \cos^2\theta\, \mathbf{F}_i'(\theta + \tfrac{\pi}{2}) + \sin^2\theta\, \mathbf{F}_{i+1}'(\theta),\end{aligned} \tag{3}$$

where $\mathbf{C}_i' = \mathrm{d}\mathbf{C}_i/\mathrm{d}\theta$ and $\mathbf{F}_i' = \mathrm{d}\mathbf{F}_i/\mathrm{d}\theta$ denote the first derivatives. Note that the first term is zero at the end points of the curve segment and the derivatives at the control points only depend on the derivative of one interpolation function, such that $\mathbf{C}_i'(0) = \mathbf{F}_i'(\frac{\pi}{2})$ and $\mathbf{C}_i'(\frac{\pi}{2}) = \mathbf{F}_{i+1}'(\frac{\pi}{2})$. Thus, $\mathbf{C}_i'(\frac{\pi}{2}) = \mathbf{C}_{i+1}'(0) = \mathbf{F}_{i+1}'(\frac{\pi}{2})$.

For C^2 continuity, we must also check the second derivative of the curve,

$$\begin{aligned}\mathbf{C}_i''(\theta) = {} & 2\left(\cos^2\theta - \sin^2\theta\right)\, \left(\mathbf{F}_{i+1}(\theta) - \mathbf{F}_i(\theta + \tfrac{\pi}{2})\right) \\ & + 4\cos\theta\sin\theta\, \left(\mathbf{F}_{i+1}'(\theta) - \mathbf{F}_i'(\theta + \tfrac{\pi}{2})\right) \\ & + \cos^2\theta\, \mathbf{F}_i''(\theta + \tfrac{\pi}{2}) + \sin^2\theta\, \mathbf{F}_{i+1}''(\theta).\end{aligned} \tag{4}$$

At the control points the first term of this equation is zero by definition (since $\mathbf{F}_i(\frac{\pi}{2}) = \mathbf{F}_{i+1}(0)$ and $\mathbf{F}_i(\pi) = \mathbf{F}_{i+1}(\frac{\pi}{2})$), and the second term is zero as well. Therefore, the second derivative at the ends of the curve segment depends on only one of the interpolation functions, such that $\mathbf{C}_i''(0) = \mathbf{F}_i''(\frac{\pi}{2})$ and $\mathbf{C}_i''(\frac{\pi}{2}) = \mathbf{F}_{i+1}''(\frac{\pi}{2})$. Thus, as long as the interpolation functions within the range $\theta \in [0, \frac{\pi}{2}]$ have C^2 continuity, the resulting curve is also C^2 continuous.

While C^2 continuity is important in practice for defining smooth curves, a higher degree of continuity is seldom needed. Nonetheless, it is possible to extend this formulation to achieve C^3 continuity if desired, but this requires additional restrictions on how the interpolation functions can be defined. The third derivative of the curve can be written as

$$\begin{aligned}\mathbf{C}_i'''(\theta) = {} & 8\cos\theta\sin\theta\, \left(\mathbf{F}_i(\theta + \tfrac{\pi}{2}) - \mathbf{F}_{i+1}(\theta)\right) \\ & + 6\left(\cos^2\theta - \sin^2\theta\right)\, \left(\mathbf{F}_{i+1}'(\theta) - \mathbf{F}_i'(\theta + \tfrac{\pi}{2})\right) \\ & + 6\cos\theta\sin\theta\, \left(\mathbf{F}_{i+1}''(\theta) - \mathbf{F}_i''(\theta + \tfrac{\pi}{2})\right) \\ & + \cos^2\theta\, \mathbf{F}_i'''(\theta + \tfrac{\pi}{2}) + \sin^2\theta\, \mathbf{F}_{i+1}'''(\theta),\end{aligned} \tag{5}$$

The first and the third terms of this equation are zero at the ends of the curve segment. However, for the second term to be zero, the first derivatives of the interpolation functions must align at the two ends of the curve. The interpolation function examples we discuss in the next section do not enforce this additional condition, so they do not provide C^3 continuity.

Note that the blending formulation in Equation (2) is simply a linear interpolation (using trigonometric weights) of two functions. Therefore, if the representation of the interpolation function is affine invariant, then the resulting curve is affine invariant as well. Furthermore, if the functions $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ have well-defined bounds, then $\mathbf{C}_i$ is bounded by their combination (i.e., the bounding box of the two bounding boxes, the convex hull of the two convex hulls, or the maximum of the two distances to the control polygon). The bounds for $\mathbf{C}_i$ can also be computed for a specific θ value by simply interpolating the bounds of the interpolation functions using Equation (2).

This formulation leads to a class of interpolating splines with each spline formulation in this class using a different interpolation function. Regardless of the chosen interpolation function, the resulting curves are guaranteed to be C^2 continuous. Since the interpolation functions only consider three consecutive control points, the curves have local support, and they do not require a global optimization step. Other properties of the curve types within this class depend on the chosen interpolation function.

4 EXAMPLE INTERPOLATION FUNCTIONS

It is easy to imagine various types of interpolation functions that can be used with the formulation described above. Indeed, since the only requirement on the interpolation function is that it goes

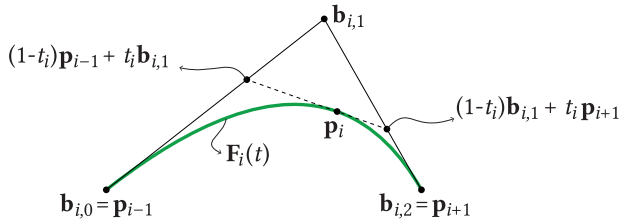

Fig. 2. The construction of the Bézier interpolation function.

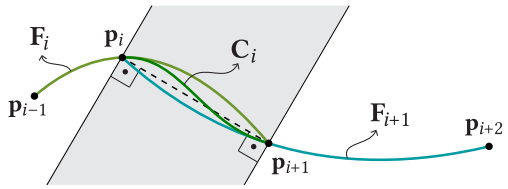

Fig. 3. The curve generated with Bézier interpolation function remains between two control points (the shaded region).

through three consecutive control points, unusual formulations that have not been previously used for defining splines can be utilized as interpolation functions. In this section, we provide some examples that demonstrate the effectiveness of our spline formulation. Note that since three points in space define a plane, the interpolation functions can be formulated in 2D, but the control points and the curve segments formed by Equation (2) can have any dimensions.

4.1 Bézier Interpolation Function

The first interpolation function $\mathbf{F}_i$ we present is a typical spline formulation, using a quadratic Bézier curve that interpolates three consecutive control points $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, and $\mathbf{p}_{i+1}$. A quadratic Bézier is defined by three control points $\mathbf{b}_{i,0}$, $\mathbf{b}_{i,1}$, and $\mathbf{b}_{i,2}$, and it goes through the first and the last one. Therefore, the placement of these two control points is trivial, such that $\mathbf{b}_{i,0} = \mathbf{p}_{i-1}$ and $\mathbf{b}_{i,2} = \mathbf{p}_{i+1}$, as shown in Figure 2. The middle control point $\mathbf{b}_{i,1}$ must be chosen such that the curve goes through $\mathbf{p}_i$. Let $t_i \in [0,1]$ be the normalized parameter value for the Bézier curve at $\mathbf{p}_i$. Then, using quadratic Bézier formulation, we can write

$$\mathbf{b}_{i,1} = \frac{\mathbf{p}_i - (1-t_i)^2\,\mathbf{b}_{i,0} - t_i^2\,\mathbf{b}_{i,2}}{2(1-t_i)t_i}. \tag{6}$$

Note that any choice for t_i forms a valid interpolation function; therefore, t_i can be chosen arbitrarily.

We favor picking the t_i that places the local maximum of curvature at $\mathbf{p}_i$. The corresponding t_i value can be calculated by solving the cubic equation

$$\begin{aligned}&\|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|^2\, t_i^3 + 3\,(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) \cdot (\mathbf{p}_{i-1} - \mathbf{p}_i)\, t_i^2 \\ &+ (3\mathbf{p}_{i-1} - 2\mathbf{p}_i - \mathbf{p}_{i+1}) \cdot (\mathbf{p}_{i-1} - \mathbf{p}_i)\, t_i - \|\mathbf{p}_{i-1} - \mathbf{p}_i\|^2 = 0,\end{aligned} \tag{7}$$

which has a single root in $[0,1]$ for any placement of the control point positions $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, and $\mathbf{p}_{i+1}$ [Yan et al. 2017].

This particular choice for t_i bounds the distance between the curve and the control polygon, such that the distance between curve segment $\mathbf{C}_i$ and the line that connects its end points h_i is bounded by $h_i/d_i \le 1/8$, where $d_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|$. A proof of this property is provided under Theorem A.1 in the appendix.

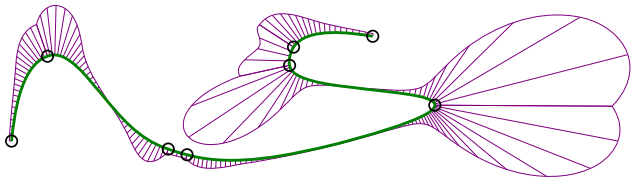

Fig. 4. An example curve generated with Bézier interpolation function, showing that the local curvature maxima appear near control points. The purple lines indicate the curvature of the curve.

This choice for t_i also ensures that the two parts of the Bézier curve that interpolate consecutive control points remain between the control points, as shown in Figure 3. Consider the shaded area between two parallel lines that are perpendicular to the line that connects the two consecutive control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and each passing through one of the control points. The curve segment between these control points is contained within this area (see Theorem A.2 in the appendix for a proof). This property holds in higher dimensions as well, where $\mathbf{C}_i$ is bounded by two planes or hyperplanes that are perpendicular to $\overline{\mathbf{p}_i\mathbf{p}_{i+1}}$.

Furthermore, this particular choice for t_i also guarantees that the resulting curve segment $\mathbf{C}_i$ cannot contain self-intersections. We provide a proof of this property in the appendix under Theorem A.3. Cusps can appear only at control points and only in the singular case when three consecutive control points are colinear and out of order (i.e., the one in the middle is not between the other two).

Note that while we pick the t_i value that places $\mathbf{p}_i$ at the local maximum of curvature for the interpolation function, the final curve $\mathbf{C}_i$ that combines two interpolation functions can have its maximum local curvature at a slightly different position. Our experiments with this particular interpolation function revealed that the local curvature maxima appear very close to control points (Figure 4).

Constructing a global parameterization s with this interpolation function can be achieved by assigning global parameter values s_i to each control point, such that

$$s_i = \alpha_i t_i + s_{i-1}, \text{ and} \tag{8}$$

$$s_{i+1} = \alpha_{i+1} t_{i+1} + s_i = \alpha_i + s_{i-1}, \tag{9}$$

where $\alpha_i = s_{i+1} - s_{i-1}$ is a scaling factor for the normalized local parameter of $\mathbf{F}_i$. The first values of this recursive definition can be chosen arbitrarily, such as $s_0 = 0$ and $\alpha_0 = 1$. For each curve segment, where $s \in [s_i, s_{i+1}]$, the local parameter θ can be defined as $\theta = (\pi/2)(s - s_i)/(s_{i+1} - s_i)$. Note that constructing a global parameterization is not required for evaluating $\mathbf{C}_i$, and it can be computed directly using the local parameters.

Bézier curves remain within the convex hull of their control points. Therefore, regardless of the choice for t_i, the convex hull of the curve segment $\mathbf{C}_i$ can be defined using the convex hulls of the parts of the two interpolation functions between control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. Since these two points are shared by the interpolation functions, the convex hull of $\mathbf{C}_i$ is formed by four

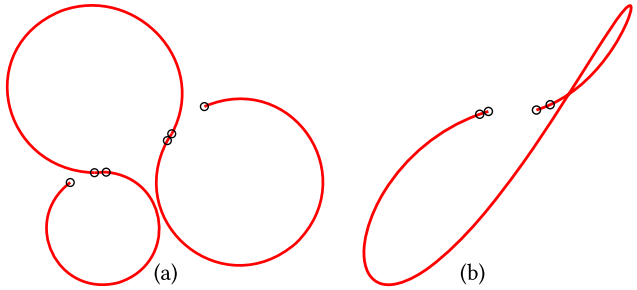

Fig. 5. Interpolating splines generated using circular interpolation functions (a) can be far from the control polygon and (b) lead to cusps and self-intersections within curve segments.

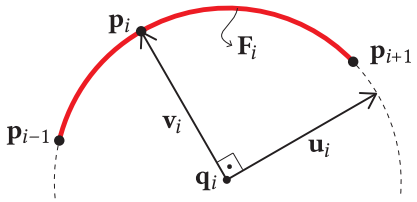

Fig. 6. The affine-invariant representation of the circular interpolation function.

points in total, two of which are $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, and the other two are $(1-t_i)\mathbf{b}_{i,1} + t_i\mathbf{p}_{i+1}$ and $(1-t_{i+1})\mathbf{p}_i + t_{i+1}\mathbf{b}_{i+1,1}$ (see Figure 2).

Overall, this quadratic Bézier interpolation function leads to C^2 curves with each segment defined by only four control points, guarantees no self-intersections within curve segments, and ensures that the resulting curve always moves toward the next control point. However, just like polynomial interpolating curves, it cannot represent perfect circular arcs.

4.2 Circular Interpolation Function

The second interpolation function we present is a circle that goes through three consecutive control points. This is not a typical formulation for splines, as a circle cannot handle more than three control points. Nonetheless, a circle can be used as an interpolation function with our spline formulation. Indeed, this particular interpolation function also appears in prior blending curve formulations [Séquin et al. 2005; Szilvási-Nagy and Vendel 2000; Wenz 1996]. Regardless of dimensions and control points positions, there is always a circular arc that goes from $\mathbf{p}_{i-1}$ to $\mathbf{p}_{i+1}$ and passes through $\mathbf{p}_i$. In the singular case when control points are colinear, leading to a circle with infinite radius, we simply use a line segment instead.

This interpolation function produces perfect circular arcs whenever four consecutive control points are on the same circle. The resulting curves address a fundamental limitation of polynomial spline formulations.

Yet, other properties of splines formed by this circular interpolation function are not as desirable. First, curves can be substantially far from the control polygon, as shown in Figure 5(a). Furthermore, it is possible to get cusps and self-intersections with this interpolation function, as shown in Figure 5(b).

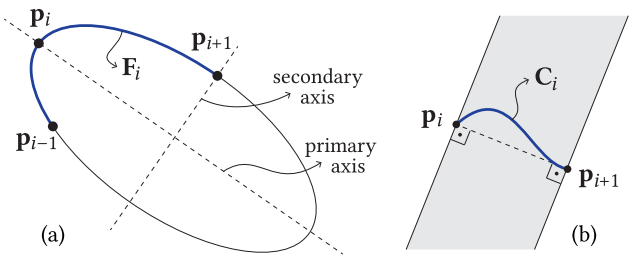

Fig. 7. Elliptical interpolation function: (a) The construction of the ellipse and (b) the resulting curves contained in the shaded area.

Another difficulty with this interpolation function is handling affine transformations. Translation, rotation, and uniform scale applied to the control points or the resulting curve produces the same result, but this is not true for non-uniform scale (and thereby shear). This comes as no surprise, because circles that undergo non-uniform scale are no longer circles, they are ellipses. Yet, supporting non-uniform scale is particularly important for rendering operations with rasterization. We can achieve this by converting our circle function to a representation that can handle *all* affine transformations. The representation we use for $\mathbf{F}_i$ consists of the center of the circle $\mathbf{q}_i$ and two perpendicular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ that are on the same plane as the circle and have the same length as the radius of the circle, as shown in Figure 6. We arbitrarily pick $\mathbf{v}_i = \mathbf{p}_i - \mathbf{q}_i$ and the resulting interpolation function $\tilde{\mathbf{F}}_i$ representing the circular arc using a normalized local parameter $t \in [0,1]$ can be written as

$$\tilde{\mathbf{F}}_i(t) = \cos(\alpha_i t + \delta_i)\,\mathbf{u}_i + \sin(\alpha_i t + \delta_i)\,\mathbf{v}_i + \mathbf{q}_i, \qquad (10)$$

where α_i and δ_i are chosen such that $\tilde{\mathbf{F}}_i(0) = \mathbf{p}_{i-1}$ and $\tilde{\mathbf{F}}_i(1) = \mathbf{p}_{i+1}$. A global parameterization can be formed similar to the Bézier interpolation function described above, using $t_i = -\delta_i/\alpha_i$ that leads to $\tilde{\mathbf{F}}_i(t_i) = \mathbf{p}_i$. Obviously, this representation must be computed prior to applying non-uniform scale. Note that this representation can properly handle perspective transformations as well.

4.3 Elliptical Interpolation Function

A significantly better formulation can be achieved using ellipses, instead of circles. However, three points are not enough to uniquely define an ellipse. To narrow down the solution space into a unique ellipse, we introduce additional constraints, resulting in a novel formulation for constructing splines with ellipses.

First, we make sure that $\mathbf{p}_i$ is along one of the two major axes of the ellipse, as shown in Figure 7(a). We refer to this axis as the *primary axis*. Second, we pick one of the other control points, either $\mathbf{p}_{i-1}$ or $\mathbf{p}_{i+1}$, whichever one is further away from $\mathbf{p}_i$, and place it along the other major axis, referred to as the *secondary axis*. We impose no restrictions on which axis would be the shorter one (i.e., the semi-major axis). Finally, we pick the ellipse that places the third control point on the other side of the primary axis. Since it is closer to $\mathbf{p}_i$ than the other control point, it resides somewhere between two the vertices of the ellipse (i.e., the points on the ellipse that intersect with one of the axes). These restrictions uniquely define an ellipse. Let $\mathbf{p}_{i+1}$ be further away from $\mathbf{p}_i$ than $\mathbf{p}_{i-1}$ (as in Figure 7). The set of ellipses that have $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ as vertices on

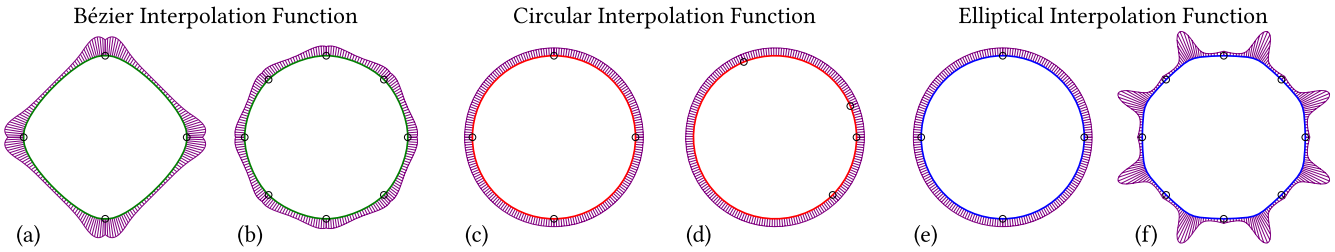

Fig. 8. Control points along a circle: (a) The Bézier interpolation function cannot produce circular arcs, though (b) adding more control points on a circle brings the resulting curve closer to a circle; (c) circular interpolation function can easily form perfect circles, (d) even when control points are not regularly placed; (e) elliptical interpolation function can form perfect circular arcs in special cases, but (f) more than four control points on a circle leads to a non-circular curve with curvature spikes away from control points. The purple lines indicate the curvature of the curves.

the primary and secondary axes have centers along the circle with diameter $\overline{\mathbf{p}_i\mathbf{p}_{i+1}}$. In our implementation, we find the center of the ellipse that passes through $\mathbf{p}_{i-1}$ using bisection and formulate $\mathbf{F}_i$ as in Equation (10).

The resulting splines with this interpolation function have some interesting properties that are highly desirable for an interpolating spline formulation. First, the curves generated with this formulation are close to the control polygon. Let d_i be the distance between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and h_i be the maximum distance between the curve segment $\mathbf{C}_i$ and the line that connects its end points. h_i/d_i is maximized when the curve forms a circular arc of angle $\frac{\pi}{2}$ and h/d_i becomes $(\sqrt{2}-1)/2$, which is less than 21% (see the proof of Theorem B.1 in the appendix).

Second, since we constrain the pair of consecutive control points with the largest separation to lie on the two major axes of the ellipse, the tangent of the curve segment between two control points is always pointed toward the next control point, except for the singular case when the control points are colinear and the secondary axis collapses to a point. In the case of this singularity, the derivative at $\mathbf{p}_i$ becomes zero, but the rest of the curve segment still maintains tangents toward the next control point, as shown in the proof of Theorem B.2 in the appendix.

Consequently, the curve is contained between the two consecutive control points, as shown in Figure 7(b) (see Theorem B.3 in the appendix). Like our Bézier interpolation function, this property holds in higher dimensions as well.

Another consequence of this is that the curve segments cannot contain self-intersections (see Theorem B.4 in the appendix). Cusps can only appear at control points in the singular case when the secondary axis collapses.

Similarly to the circular interpolation function, this formulation can also represent perfect circles (in addition to perfect elliptical arcs). However, perfect circular (and elliptical) arcs only appear in special cases,[2] such as the example in Figure 8(e). Also, the maximum curvature of the interpolation function $\mathbf{F}_i$ is at $\mathbf{p}_i$ if the primary axis is longer than the secondary axis of the ellipse; otherwise, it is at one of the other two control points (the one on the secondary axis). Therefore, the resulting curve segments $\mathbf{C}_i$ can have sharp curvature peaks close to their centers, such as the example in Figure 8(f), which can be undesirable for some applications.

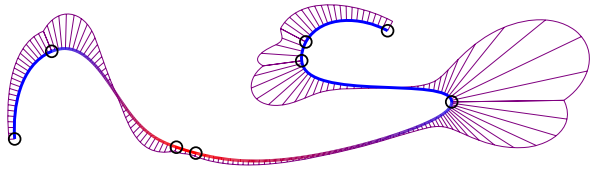

Fig. 9. An example curve generated with hybrid interpolation function, showing that the local curvature maxima appear near control points. The purple lines indicate the curvature of the curve.

4.4 Hybrid (Circular-Elliptical) Interpolation Function

The class of splines we present in this article does not require using the same formulation for $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$. Therefore, each interpolation function of the curve can be defined independently. We demonstrate this feature with a hybrid interpolation function that alternates between circular and elliptical interpolation functions.

The hybrid interpolation function we describe here combines the benefits of circular and elliptical interpolation functions and avoids their undesirable properties. The advantage of the circular interpolation function is that it makes it easy to define circular arcs, but when the angle of the arc between two consecutive control points is larger than π, the tangents of the interpolation function on control points no longer point toward the next control point, resulting in curves that can be arbitrarily far from the control polygon (Figure 5(a)) and can contain self-intersections (Figure 5(b)). The elliptical interpolation function guarantees that the tangent is always toward the next control point (thereby avoiding self-intersections), but when the primary axis is shorter than the secondary axis, the maximum curvature of the curve segment appears near its center (Figure 8(f)), which might be undesirable for some applications. Therefore, our hybrid interpolation function uses circular interpolation when the angle of the arc is small and switches to elliptical interpolation for larger arcs, which produce ellipses with longer primary axes.

The threshold we set for switching between the two functions is $\frac{\pi}{2}$. If the angle of the arc that corresponds to one of the two circular arcs $\widehat{\mathbf{p}_{i-1}\mathbf{p}_i}$ and $\widehat{\mathbf{p}_i\mathbf{p}_{i+1}}$ is above this threshold, then we use the elliptical interpolation function. If both angles are smaller than the threshold, then we use the circular interpolation function. The reason behind the choice of this particular threshold is that in

[2]The elliptical interpolation function produces a circular arc of $\pi/2$ when four consecutive control points are on a circle, the middle two have $\pi/2$ separation, and the others have $\pi/2$ or less separation.

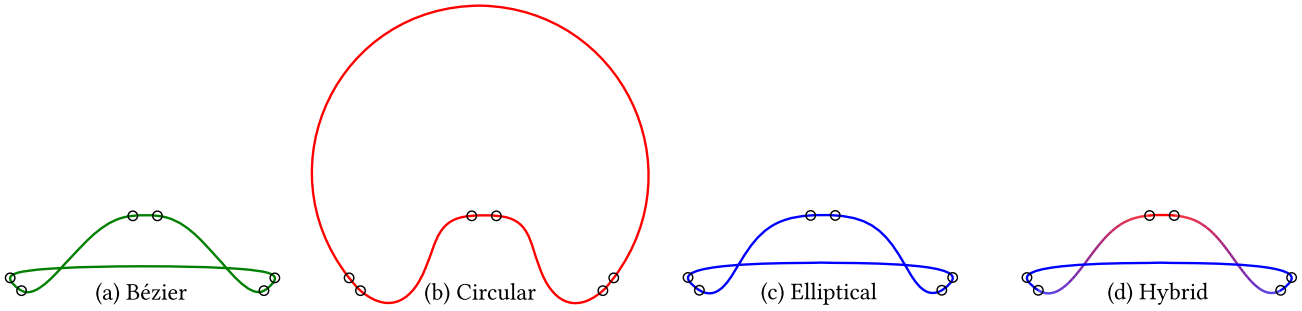

Fig. 10. A comparison of curves generated with our formulation using our example interpolation functions from the same control points.

Table 1. Comparison of Curvature-continuous Spline Formulations

	Catmull-Rom Splines [Catmull and Rom 1974]	Interpolating B-Splines [Farin 2002]	κ-curves [Yan et al. 2017]	**Ours with Bézier Int. Func.**	**Ours with Circular Int. Func.**	**Ours with Elliptical Int. Func.**	**Ours with Hybrid Int. Func.**
Continuity	C^2	C^2	mostly G^2	C^2	C^2	C^2	C^2
Local Support	yes (6 points)	no	no	yes (4 points)	yes (4 points)	yes (4 points)	yes (4 points)
Global Optimization	no	no	yes	no	no	no	no
Self-Intersections	yes	yes	no	no	yes	no	no
Distance to Polygon	unbounded	unbounded	bounded	bounded	unbounded	bounded	bounded
Max. Curvature	anywhere	anywhere	at control points	near control points	anywhere	anywhere	near control points
Linear Segments	simple	hard	hard	simple	simple	simple	simple
Circular Arcs	no	no	no	no	yes	yes	yes

the case when one of the angles is exactly $\frac{\pi}{2}$, and the other one is equal or smaller, the two interpolation functions become identical (i.e., elliptical interpolation function produces a circle). Therefore, using $\frac{\pi}{2}$ as the threshold, we can seamlessly transition between the two functions. Thus, continuous motion of control points leads to continuous changes in the shapes of the resulting curves.

This simple combination of the two interpolation functions inherits all benefits of the elliptical interpolation function. Also, it makes it easier to form circular arcs and avoids the high-curvature peaks of the elliptical interpolation near the center of curve segments. $\mathbf{F}_i$ either has constant curvature (forming a circle) or its maximum curvature is at $\mathbf{p}_i$ (forming an ellipse with a longer primary axis). Our experiments revealed that, just like our Bézier interpolation function, the resulting curves $\mathbf{C}_i$ have local curvature maxima close to control points, as shown in Figure 9.

5 RESULTS AND DISCUSSION

We include a self-contained example implementation of our curve formulations (with full source code) in a single-file webpage (using HTML, JavaScript, and WebGL) as a supplementary document, including all four example interpolation functions.

5.1 Comparison of Interpolation Functions

Obviously, the example interpolation functions we describe produce different curves from the same control points. Still, we provide comparisons using the same set of control points for discussing their similarities and differences.

Two simple examples comparing the four example interpolation functions are provided in Figure 1 and Figure 10, and more complex examples are shown in Figure 11. The curves with hybrid (circular-elliptical) interpolation function are colored to indicate which function is used for which curve segment (red indicates circular and blue indicate elliptical). The one that visually stands out among these four functions is the circular interpolation function, forming curves with no sharp features. More importantly, curves with circular interpolation function can significantly deviate from the control polygon (Figure 10(b)). In fact, other than producing perfect circles, the curves with the circular interpolation function do not have many desirable features, besides the common features of this class of curves (i.e., C^2 continuity and local support). The elliptical interpolation function can produce sharper features near isolated control points, but curve segments can contain curvature peaks away from control points (Figure 11(c)). The curves with both Bézier and hybrid interpolation functions can produce sharper features near isolated control points and avoid curvature spikes away from control points. The resulting curves are also remarkably close to the control polygon.

The main advantage of the hybrid interpolation function (Figure 11(d)) over Bézier (Figure 11(a)) is its ability to easily produce perfect circular arcs. This is presented in Figure 8, showing curves generated with control points on a circle. Notice that Bézier interpolation function produces relatively sharp features near control points (Figure 8(a) and (b)). While this might be desirable in some cases, approximating a circle with these curves becomes difficult. Circular interpolation function produces a circle regardless of how the control points are oriented around the circle (Figure 8(c) and (d)). Elliptical interpolation function can produce perfect circular arcs (Figure 8(e)) but can deviate from a circle when representing smaller arcs (Figure 8(f)), which is avoided by the hybrid interpolation function.

5.2 Curve Properties

Table 1 provides a comparison of the four example curve types we present in this article to popular interpolating spline

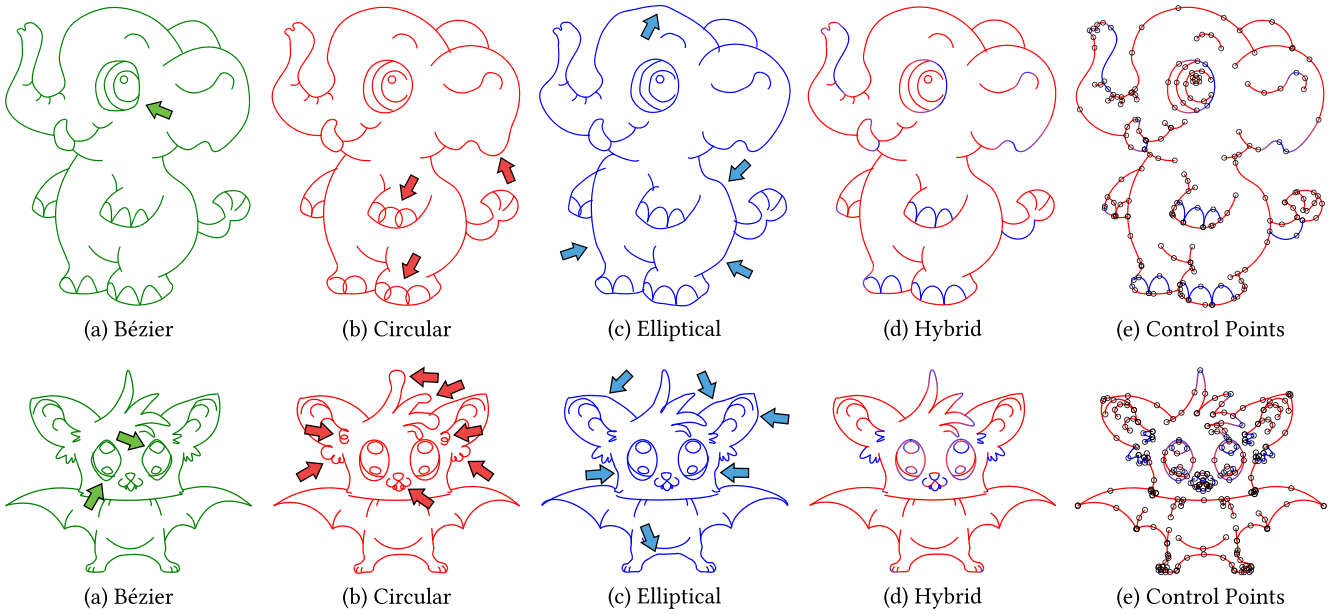

Fig. 11. A comparison of curves generated with our formulation using different interpolation functions. The arrows highlight parts of the curves that exhibit the characteristics of the interpolation functions: (a) Bézier interpolation function can have difficulty forming near-circular segments, (b) circular interpolation function can deviate from the control polygon and always forms circular shapes, and (c) elliptical interpolation function can form high-curvature peaks between control points, negatively impacting the visual smoothness of the curve segments. (d) Hybrid interpolation functions avoids these problems. Artwork by Ozum Yuksel.

formulations that provide curvature-continuity. All curve types within the class of splines we present in this article have C^2 continuity, which is essential for defining smooth curves. When the curves are used for representing hair or cloth fibers, C^2 continuity provides smoothly varying specular reflections. When interpolating animation keyframes, C^2 continuity means continuous acceleration. G^2 continuity of κ-curves [Yan et al. 2017] is sufficient for curvature continuity. Though G^2 does not mean continuous second derivative, it is possible to define a global C^2 parameterization for a G^2 curve. Yet, κ-curves are only G^1 at infliction points, due to the underlying limitation of the quadratic polynomial representation they use.

Note that constructing a global parameterization with our formulation is no different than parameterizing Catmull–Rom curves. However, unlike the Catmull–Rom formulation, we cannot achieve C^2 continuity with user-specified parameter values per control point.

Local support is important to make sure that modifying a single control point does not have a global effect on the curve. Neither interpolating B-Splines [Farin 2002] nor κ-curves [Yan et al. 2017] provide local support. While the effect of a single control point is mostly local with κ-curves, they require a global optimization step for recomputing the entire curve whenever a control point is modified. C^2 Catmull–Rom splines [Catmull and Rom 1974] do provide local support, but they form piecewise fourth-degree polynomials defined by six control points each. In comparison, all curve formulations in the class of splines we present need only four control points to define each curve segment.

Three of the example interpolation functions we describe (all but circular interpolation function) guarantee self-intersection-free curve segments. This property is also supported by κ-curves [Yan et al. 2017]. This is a crucial property, and the unpopularity of prior interpolating C^2 curve formulations in practice can be attributed to the fact that they are prone to producing unintended self-intersections.

The same three interpolation functions (all but circular interpolation function) also produce curves that are close to the control polygon. However, C^2 Catmull–Rom splines [Catmull and Rom 1974] and interpolating B-Splines [Farin 2002] can form curve segments that are arbitrarily far from the control polygon, and their distance is not bounded as a function of the distance between the two control points they interpolate. Indeed, this has been another important flaw of earlier interpolating spline formulations that certainly contributes to their unpopularity in practice.

κ-curves [Yan et al. 2017] place the local curvature maxima at control points. Arguably, this property can make the resulting curves easier to control for design applications. Curves with our Bézier and hybrid interpolation functions place the local curvature maxima in close proximity of the control points. This is because these two interpolation functions guarantee that the maximum curvature of $\mathbf{F}_i$ is at $\mathbf{p}_i$. Blending the interpolation functions $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ using the trigonometric interpolation in Equation (2), however, can slightly shift the locations of the curvature maxima.

However, forming linear curve segments with κ-curves [Yan et al. 2017] and interpolating B-splines [Farin 2002] can be challenging, due to global support. Even when multiple control points are placed on a line, the resulting curve can bend because of the

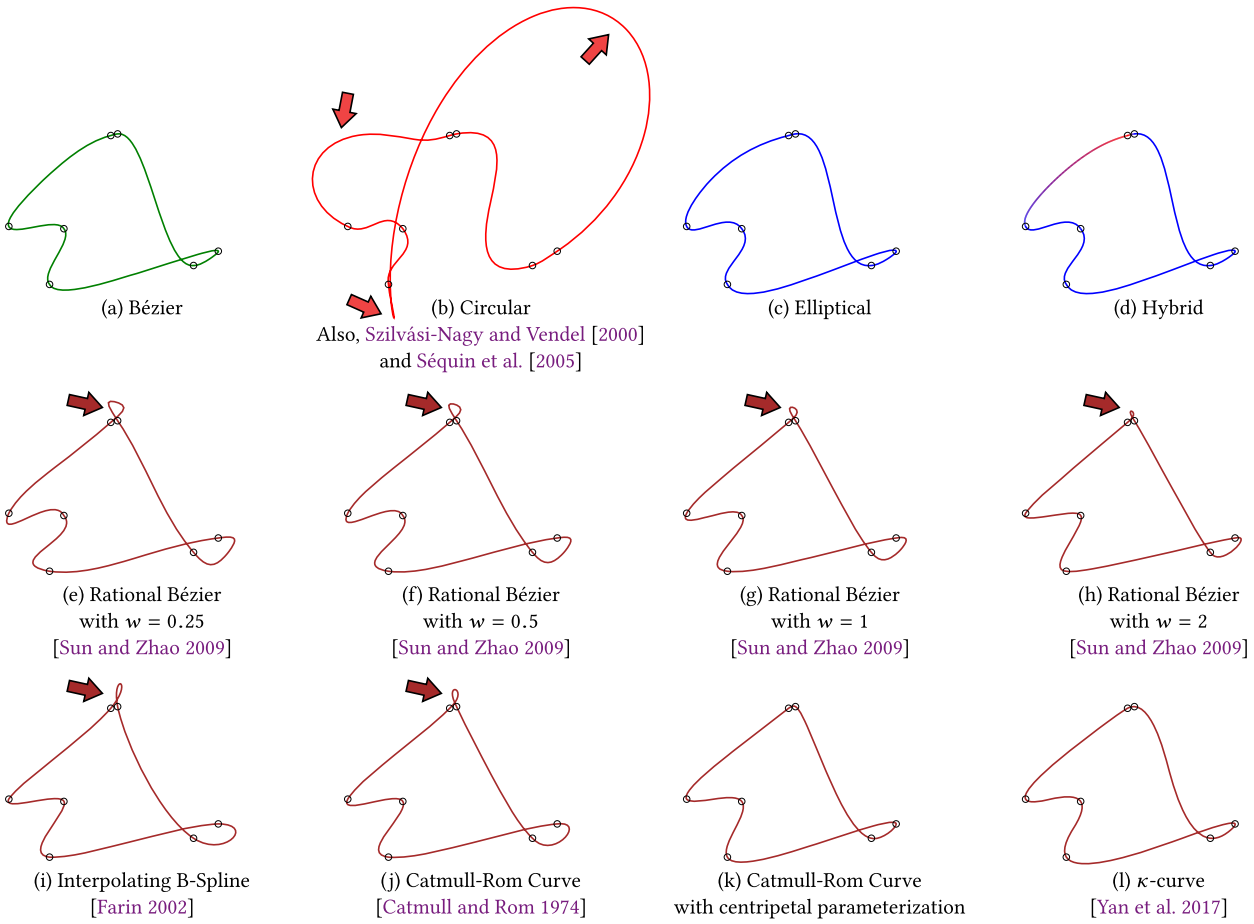

Fig. 12. Comparison of curves generated from the same set of control points using different curve formulations. The top row shows our example interpolation function; the middle row shows the formulation of Sun and Zhao [2009], which is similar to our curve construction, with different weight parameters w; and the bottom row shows other curve types. Though in this example C^2 Catmull–Rom curve with centripetal parameterization does not present self-intersections, only three of our example interpolation functions and κ-curves guarantee self-intersection-free curves.

other control points. Also, linear segments conflict with the goal of placing local curvature maxima at control points that κ-curves enforce, as line segments have zero curvature. C^2 Catmull–Rom splines [Catmull and Rom 1974] can produce linear segments, but they require six consecutive control points to be on a line. In comparison, all of our interpolation functions can produce perfect linear segments when four consecutive control points are placed linearly.

Circular and hybrid interpolation functions form perfect circular arcs whenever any four consecutive control points are on a circle. Our elliptical interpolation function produces circular arcs only in special cases. Neither our Bézier interpolation function, nor any polynomial interpolating curve formulation can form perfect circles. However, a recent extension of κ-curves can produce circular an elliptical arcs by using a rational Bézier formulation with a more complicated global optimization procedure [Yan et al. 2019].

5.3 Comparisons to Prior Methods

Figure 12 shows a comparison of curvature-continuous curves generated from the same set of control points using different curve formulations. The top row shows our formulation with our example interpolation functions. Notice that circular interpolation function (Figure 12(c)), which produces identical curves to Szilvási-Nagy and Vendel [2000] and Séquin et al. [2005], highly deviates from the control points and forms a self-intersecting segment. The novel interpolation functions we introduced in this article (Figure 12(a), (c), and (d)) form reasonable interpolations and they guarantee self-intersection-free segments. The middle row shows curves generated using the rational quadratic Bézier formulation of Sun and Zhao [2009] with different weight parameters w (Figure 12(e)–(h)), ranging from conics to hyperbola, all of which lead to self-intersecting segments (and notable deviations from the control polygon for shorter segments). Indeed, this self-intersection problem is common with other prior interpolating C^2 curves as well, such as interpolating B-Splines [Farin 2002] (Figure 12(i)) and Catmull–Rom curves [Catmull and Rom 1974] (Figure 12(j)) with uniform parameterization. Using this particular set of control points, C^2 Catmull–Rom curves with chordal and centripetal parameterizations do not produce self-intersections (Figure 12(k)), but they are also prone to forming

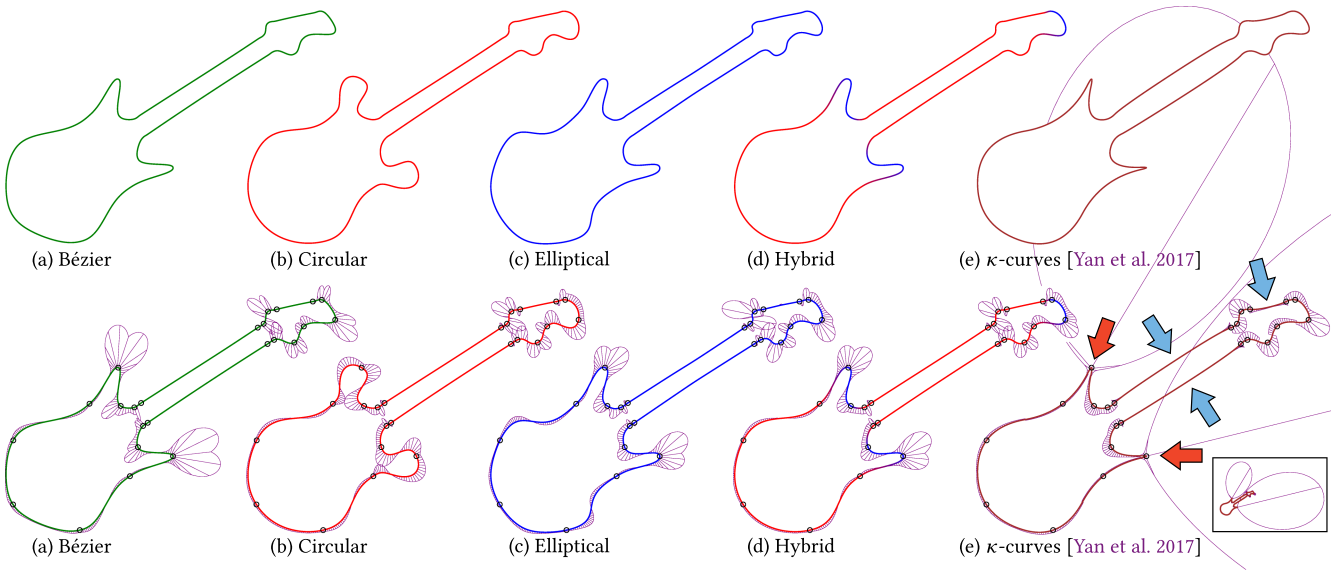

Fig. 13. Comparison of ((a)–(d)) curves generated using our method with four example interpolation functions and (e) κ-curves [Yan et al. 2017] from the same control points. The bottom row shows the control point positions and the curvature of the curves. The inset on the right bottom provides a zoomed-out view, showing the entire curvature visualization of κ-curves. The red arrows highlight the extremely sharp corners generated by κ-curves, and the blue arrows show the curve segments that are bent due to the global support of κ-curves.

self-intersecting segments with different control point positions [Yuksel et al. 2009b]. The κ-curves method [Yan et al. 2017] in this example leads to a reasonable interpolation (Figure 12(l)), similarly to our curves with Bézier, elliptical, and hybrid interpolation functions. However, the resulting κ-curve is mostly G^2 with G^1 at infliction points (where the curvature changes sign), unlike the curves with our formulation that have C^2 continuity everywhere.

Of the similar curve formulations in prior work, the method of Szilvási-Nagy and Vendel [2000] and Séquin et al. [2005] are identical to our curves with circular interpolation function (Figure 12(b)), and the rational Bézier formulation of Sun and Zhao [2009] forms curves within the class we present in this article (Figure 12(e)–(h)). Rational Béziers allow elliptical (with $0 < w < 1$), parabolic (with $w = 1$), and hyperbolic (with $w > 1$) interpolation. However, these curves cannot address the problems of typical interpolating curves with cusps and self-intersecting segments, unbounded distance to the control polygon, and curvature maxima appearing anywhere. In fact, besides their ability to form conics and somewhat improved local-support, they are not qualitatively superior to the popular alternative of Catmull–Rom curves [Catmull and Rom 1974] (Figure 12(j)), producing a visually similar interpolation. The advantages of our Bézier interpolation function stem from placing $\mathbf{p}_i$ at the local curvature maxima. The formulation of Sun and Zhao [2009], however, places $\mathbf{p}_i$ at the parametric center of a rational Bézier curve, which leads to the undesirable properties mentioned above that are common with prior interpolating curves. Note that it is possible to replace our Bézier interpolation function with rational Béziers, solving a different equation to find the curvature maxima [Yan et al. 2019].

We provide another comparison to κ-curves [Yan et al. 2017] in Figure 13. As compared to our curves, κ-curves (Figure 13(e)) can produce extremely sharp corners (highlighted with red arrows). More importantly, κ-curves fail to produce linear segments (highlighted with blue arrows in Figure 13(e)). Notice that all of our interpolation functions are able to produce linear segments (Figure 13(a)–(d)). Since κ-curves are quadratic polynomials, they cannot represent perfect circular arcs either. Indeed, the objective of placing the maximum curvature exactly at control points conflicts with lines and circles, which have constant curvature (i.e., no maximum point of curvature). That is why the implementation of κ-curves in Adobe Illustrator differs from the original formulation to address some of its shortcomings. Obviously, since these are different curve formulations, producing a similar curve shape would require a different set of control points. Nonetheless, all curve types within our class provide C^2 curves and local support without a global optimization step, while κ-curves are mostly G^2 (and G^1 at infliction points), have global support, and are generated through a costly numerical optimization process. The only arguable advantage κ-curves have over our curves with Bézier and hybrid interpolation functions is that the curvature maxima are exactly at the control points, instead of having them near control points. Yet, this strict policy also prevents κ-curves from forming linear curve segments. Also, κ-curves and their recent extension [Yan et al. 2019] are only shown to work in 2D and it is unclear if κ-curves can be easily extended to 3D (or higher dimensions). In comparison, our formulation naturally supports higher dimensions, even though interpolation functions are conveniently defined in 2D.

5.4 Three-dimensional Curves

Note that while most examples in this article are 2D curves, the properties of our formulation are maintained in higher dimensions as well. An example featuring 3D hair modeling is shown in Figure 14. The hair strands are generated from a hair mesh [Yuksel

Fig. 14. An example hair model generated using our curves with the hybrid interpolation function for both representing the hair strands and the edges of the hair mesh along the hair direction. The control points of the final hair curves are placed using procedural styling operations. The hair model consists of over 100,000 strands and 8.5 million curve segments. The character and hair mesh models by Lee Perry-Smith. Designed and rendered using Hair Farm, the ultimate hair plugin for Autodesk 3ds Max.

et al. 2009a]. All hair strands and the edges of the hair mesh along the hair direction are formed by our spline formulation with the hybrid interpolation function. Using an interpolating curve formulation (as opposed to an approximating curve) for representing the hair mesh allows the user to directly control the hair mesh surface. The C^2 continuity provided by our curves leads to continuous changes in specular reflections, and our hybrid interpolation function connects most control points with circular arcs, which can be desirable for this application. Most importantly, our formulation is computationally efficient enough for such applications that involve a vast amount of curves, since the curves are generated without global optimization.

5.5 Sharp Features

When sharp features are needed, our curves can provide them by placing control points in close proximity. Some examples including sharp features are shown in Figure 15. Still, the curves maintain C^2 continuity everywhere. When extremely sharp corners are needed, placing two consecutive control points exactly on top of each other (by collapsing the curve segment between them to a point) leads to a curve with only C^0 continuity at that point.

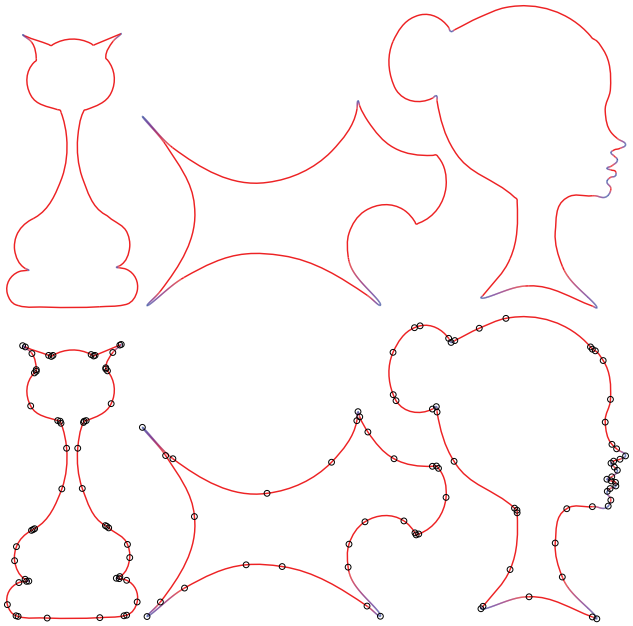

Fig. 15. Examples curves generated using the hybrid interpolation function, showing sharp features achieved by placing control points in close proximity. Artwork by Ozum Yuksel.

6 CONCLUSION AND FUTURE WORK

We have presented a class of splines that interpolate the control points, guarantee C^2 continuity everywhere, and provide local support with only four control points defining each curve segment. This formulation of splines allows defining novel curve types by simply specifying a custom interpolation function. Since our interpolation functions only consider three consecutive control points, unconventional formulations can be utilized. Furthermore, there is no need for a costly global optimization step with any spline formulation within this class, as the interpolation functions are defined locally.

We have also presented four example interpolation functions within this class, producing curves with different properties. Among them, Bézier and hybrid interpolation functions produce curves with guaranteed self-intersection-free segments, bounded distances to the control polygon, and local curvature maxima close to the control points. Furthermore, these curves also have local-support and C^2 continuity everywhere and they require no global optimization (as any other curve formulation within this class). To our knowledge, no prior spline formulation can provide all of these properties. Moreover, they can easily produce linear segments, and curves with hybrid interpolation function can also represent perfect circular arcs, a feature not supported by any polynomial interpolating spline.

Note that our formulation with the circular interpolation function example also appears in prior work [Séquin et al. 2005; Szilvási-Nagy and Vendel 2000]. This interpolation function leads to curves with undesirable properties and it is included mainly for aiding the descriptions of our elliptical and hybrid interpolation functions.

Nonetheless, the interpolation functions we describe in this article are merely examples within this class and other interpolation functions with different properties can be developed with future research. For example, cubic Bézier curves would be an interesting alternative to explore, as they provide more flexibility than the quadratic ones we use. Another one would be using a helix, which can be defined by three points in 3D [Goriely et al. 2012]. Yet, not all possible interpolation functions lead to curves with desirable properties, such as our circular interpolation function and the rational quadratic Bézier formulation of Sun and Zhao [2009].

Note that the compatibility of circular and elliptical interpolation functions, such that they form the same curve under certain conditions, makes it easy to combine them for defining our hybrid interpolation function, but a hybrid interpolation function can be defined using two (or more) incompatible interpolation functions as well. Instead of using a single threshold to switch between different interpolation types, a weighted average of different interpolations can be used to ensure continuous transitions between different interpolation types. Thus, the space of spline formulations within the class we present in this article also includes combinations of arbitrary interpolation functions.

Another interesting future research direction would be exploring interpolation functions that consider more than three control points for achieving a higher degree of continuity or other curve properties.

It has been argued that it would be a good idea to have local curvature maxima at the control points because of the sensitivity of human visual system to curvature maxima and minima [Levien and Séquin 2009]. In particular, for 2D artistic design applications, having control points strictly at the local curvature maxima might allow the user to "guess" the location of a control point just by looking at the curve, which might be considered a desirable property. However, for the purpose of making the curve formulation easier to use in practice, it is unclear how important it is to strictly enforce this property and whether having curvature maxima close to control points, like our curves, would provide a similar benefit. A user-study investigating this topic is out of the scope of this article, since we do not narrowly target this particular application, but it would be an interesting direction for future work.

APPENDIX

A BÉZIER INTERPOLATION FUNCTION PROPERTIES

Below we provide proofs for the properties of the Bézier interpolation function described in Section 4.1. Since Bézier curves are affinely invariant, without loss of generality, in the following we assume that $\mathbf{p}_i$ is at the origin and $\mathbf{p}_{i+1}$ is along the x-axis at $x = 1$.

Theorem A.1. *The distance of a curve segments with the Bézier interpolation function to the line that connects its end points is bounded by the distance between the interpolated control points.*

Proof. Let $\mathbf{b}_0$, $\mathbf{b}_1$, and $\mathbf{b}_2$ be the three control points of the quadratic Bézier curve that represents $\mathbf{F}_i$ between the interpolated control points. The Bézier curve parameter value t_{mc} for the point of maximal curvature can be written as [Yan et al. 2017]

$$t_{\text{mc}} = \frac{(\mathbf{b}_0 - \mathbf{b}_1) \cdot (\mathbf{b}_0 - 2\mathbf{b}_1 + \mathbf{b}_2)}{\|\mathbf{b}_0 - 2\mathbf{b}_1 + \mathbf{b}_2\|^2}. \tag{11}$$

Let b_x be the x-component of $\mathbf{b}_1$, such that $\mathbf{b}_1 = [b_x \ \mathbf{b}_y]^T$, where $\mathbf{b}_y$ represents the remaining components. Since $t_{\text{mc}} = 0$ by the construction of $\mathbf{F}_i$, using $\mathbf{b}_0 = \mathbf{p}_i$ and $\mathbf{b}_2 = \mathbf{p}_{i+1}$, Equation (11) becomes

$$b_x = 2b_x^2 + 2(\mathbf{b}_y \cdot \mathbf{b}_y). \tag{12}$$

Thus, the distance of $\mathbf{b}_1$ to the x-axis is $\|\mathbf{b}_y\| = (b_x/2 - b_x^2)^{\frac{1}{2}}$. This distance is maximized when $b_x = 1/4$ and thereby the distance of $\mathbf{F}_i$ is bounded by $1/8$. The same bound also applies to $\mathbf{F}_{i+1}$ due to symmetry. Therefore, the ratio of the distance of $\mathbf{C}_i$ to the line that connects its end points h_i and the distance between the interpolated control points d_i is bounded by $h_i/d_i \leq 1/8$. □

Theorem A.2. *Curve segments with the Bézier interpolation function remain between the interpolated control points.*

Proof. If both $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ remain between the interpolated control points, then $\mathbf{C}_i$ must remain between them as well. It is sufficient to show this for $\mathbf{F}_i$, and $\mathbf{F}_{i+1}$ follows due to symmetry. Based on Equation (12), $b_x \geq 0$ and $b_x - 2b_x^2 \geq 0$, thereby $b_x \in [0, \frac{1}{2}]$. Due to the convex hull property of Bézier curves, $\mathbf{F}_i$ is guaranteed to remain between the interpolated control points. Therefore, $\mathbf{C}_i$ remains between the two lines (in 2D), two planes (in 3D), or two hyperplanes (in higher dimensions) defined by $x = 0$ and $x = 1$. □

Theorem A.3. *Curve segments with the Bézier interpolation function cannot contain cusps or self-intersections.*

Proof. The curve segment cannot contain cusps or self-intersections if its derivative along the x-axis is positive between the interpolated control points. Using a local parameter $t \in [0, 1]$, the derivative with respect to t along the x-axis can be written as

$$\begin{aligned} C'_{x,i}(t) = \cos^2\left(\frac{\pi}{2}t\right) F'_{x,i}(t) + \sin^2\left(\frac{\pi}{2}t\right) F'_{x,i+1}(t) \\ + \pi \cos\left(\frac{\pi}{2}t\right) \sin\left(\frac{\pi}{2}t\right) (F_{x,i+1}(t) - F_{x,i}(t)), \end{aligned} \tag{13}$$

where the x-components of the interpolation functions can be written in polynomial form using constants x_i and x_{i+1} as

$$F_{x,i}(t) = x_i\, t^2 + (1 - x_i)\, t, \tag{14}$$

$$F_{x,i+1}(t) = x_{i+1}\, t^2 + (1 - x_{i+1})\, t, \tag{15}$$

Since $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ are quadratic curves that remain between the interpolated control points, $F'_{x,i}(t) \geq 0$ and $F'_{x,i+1}(t) \geq 0$. Thus, $-1 \leq x_i \leq 1$ and $-1 \leq x_{i+1} \leq 1$. Using these bounds, we can write $F_{x,i+1}(t) - F_{x,i}(t) \geq -2t(1-t)$. Similarly, the first two terms of Equation (13) interpolate between possible $F_{x,i}$ and $F_{x,i+1}$ values and they are bounded by $2t$ and $2(1-t)$. Since the last term is bounded by

$$\pi \cos\left(\frac{\pi}{2}t\right) \sin\left(\frac{\pi}{2}t\right) (2t(1-t)) \leq 2t \qquad \text{and} \tag{16}$$

$$\pi \cos\left(\frac{\pi}{2}t\right) \sin\left(\frac{\pi}{2}t\right) (2t(1-t)) \leq 2(1-t), \tag{17}$$

$C'_{x,i}(t)$ cannot be negative within $t \in [0, 1]$, and it can be zero only at $t = 0$ and $t = 1$. Therefore, the curve segment cannot have cusps or self-intersections. □

B ELLIPTICAL INTERPOLATION FUNCTION PROPERTIES

Below we provide proofs for the properties of the elliptical interpolation function described in Section 4.3. These properties also apply to the hybrid (circular-elliptical) interpolation function (Section 4.4). Using our affine invariant representation, without loss of generality, in the following we assume that $\mathbf{p}_i$ is at the origin and $\mathbf{p}_{i+1}$ is along the x-axis at $x = 1$. Note that transforming a given curve to this frame does not require non-uniform scale.

Theorem B.1. *The distance of a curve segment with the elliptical interpolation function to the line that connects its end points is bounded by the distance between the interpolated control points.*

Proof. Let $\theta \in [0, \frac{\pi}{2}]$ represent the local parameter for the curve segment $\mathbf{C}_i$. The distance of $\mathbf{C}_i$ to the line $\overline{\mathbf{p}_i\mathbf{p}_{i+1}}$ is bounded by the distance of $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ to this line for $\theta \in [0, \frac{\pi}{2}]$. It is sufficient to show the bound for $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$ follows due to symmetry. The distance is maximized when the elliptical arc $\mathbf{F}_i$ corresponds to the largest possible angle, which is $\frac{\pi}{2}$ by construction. Let $\mathbf{u}_i = [u_x\ u_y]^T$ and $\mathbf{v}_i = [v_x\ v_y]^T$ represent the primary axes of the ellipse, where $u_y = v_y$ and $u_x - v_x = 1$. Using Equation (10), we can write the maximum distance h_i to the line as

$$h_i = |u_y|(\cos\theta + \sin\theta - 1), \tag{18}$$

where h_i is maximized when $\mathbf{F}_i$ is a circular arc, such that $|u_y| = \frac{1}{2}$ and $\theta = \frac{\pi}{4}$. Thus, $\mathbf{C}_i$ is bounded by $h_i/d_i = (\sqrt{2} - 1)/2$, where d_i is the distance between the interpolated control points. □

Theorem B.2. *The tangent of a curve segment between control points with the elliptical interpolation function points toward the next control point, i.e., $\mathbf{C}_i'(\theta) \cdot (\mathbf{p}_{i+1} - \mathbf{p}_i) > 0$ for $\theta \in (0, \frac{\pi}{2})$.*

Proof. Consider the scalar portion of the curve derivative $\mathbf{C}_i'$ in Equation (3) along the x-axis. Only the first term of this equation can be negative and it is minimized when $\alpha_i = \alpha_{i+1} = 1$ and $\delta_i = \delta_{i+1} = 0$, such that both $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are on primary and secondary axes of both ellipses $\mathbf{F}_i$ and $\mathbf{F}_{i+1}$. The scalar equations for the two ellipses along the x-axis can be written as

$$F_i(\theta) = u_i \sin\theta + v_i \cos\theta + q_i, \tag{19}$$

$$F_{i+1}(\theta) = u_{i+1} \cos\theta + v_{i+1} \sin\theta + q_{i+1}, \tag{20}$$

where $u_i, v_i, q_i, u_{i+1}, v_{i+1}, q_{i+1} \in [0, 1]$ are the scalar components of the affine-invariant representation in Equation (10). Using properties $u_i - v_i = 1$, $-u_{i+1} + v_{i+1} = 1$, $u_i = 1 - q_i$, and $u_{i+1} = -q_{i+1}$, Equation (3) can be written as

$$C_i'(\theta) = c + (q_{i+1} - q_i)(2cs\,(c + s - 1)) + (c - s)(q_{i+1}s^2 - q_i c^2),$$

where $c = \cos\theta$ and $s = \sin\theta$. This equation is minimized when $q_i = 1$ and $q_{i+1} = 0$, and $C_i'(\theta)$ remains positive within the range $\theta \in (0, \frac{\pi}{2})$. □

Theorem B.3. *Curve segments with the elliptical interpolation function remain between the interpolated control points.*

Proof. By construction, both ellipse pieces forming the curve remain between the interpolated control points. Therefore, the resulting curve that combines the two interpolation functions using Equation (2) must remain within this rage for $\theta \in [0, \frac{\pi}{2}]$. □

Theorem B.4. *Curve segments with the elliptical interpolation function cannot contain cusps or self-intersections.*

Proof. Based on Theorem B.2 the derivative of the curve segments are positive along the direction that connects the two end points of the curve segment within range $\theta \in (0, \frac{\pi}{2})$. Thus, the resulting curve segment cannot contain cusps or self-intersections within this rage. Cusps with $C_i' = 0$ can only appear at the control points (where $\theta = 0$ or $\theta = \frac{\pi}{2}$) in special cases. □

ACKNOWLEDGMENTS

We thank Scott Schaefer for his support and feedback, Zhipei Yan for the implementation of κ-curves, Ozum Yuksel for preparing the artwork in Figure 11 and Figure 15, Lee Perry-Smith for the character and hair mesh models in Figure 14, and anonymous reviewers for their helpful comments and pointing out some of the related prior work.

REFERENCES

Phillip J. Barry and Ronald N. Goldman. 1988. A recursive evaluation algorithm for a class of Catmull-Rom splines. *SIGGRAPH Comput. Graph.* 22, 4 (1988), 199–204.
Edwin Catmull and Raphael Rom. 1974. A class of local interpolating splines. *Comput. Aid. Geom. Des.* (1974), 317–326. DOI: https://doi.org/10.1016/B978-0-12-079050-0.50020-5
Gilles Deslauriers and Serge Dubuc. 1989. Symmetric iterative interpolation processes. *Constr. Approx.* 5, 1 (1989), 49–68.
Nira Dyn, David Levin, and John A. Gregory. 1987. A 4-point interpolatory subdivision scheme for curve design. *Comput. Aid. Geom. Des.* 4, 4 (1987), 257–268.
Gerald Farin. 2002. *Curves and Surfaces for CAGD: A Practical Guide* (5th ed.). Morgan Kaufmann, San Francisco, CA.
Gerald Farin. 2006. Class A Bézier curves. *Comput. Aid. Geom. Des.* 23, 7 (2006), 573–581.
Anton Gfrerrer and Otto Röschel. 2001. Blended hermite interpolants. *Comput. Aid. Geom. Des.* 18, 9 (2001), 865–873.
Alain Goriely, Sébastien Neukirch, and Andrew Hausrath. 2012. Helices through 3 or 4 points? *Note Mat.* 32, 1 (2012), 87–103. .
Sven Havemann, Johannes Edelsbrunner, Philipp Wagner, and Dieter Fellner. 2013. Curvature-controlled curve editing using piecewise clothoid curves. *Comput. Graph.* 37, 6 (2013), 764–773.
Masatake Higashi, Kohji Kaneko, and Mamoru Hosaka. 1988. Generation of high-quality curve and surface with smoothly varying curvature. In *Proceedings of the Annual Conference of the European Association for Computer Graphics (Eurographics'88)*. Eurographics Association.
Josef Hoschek and Dieter Lasser. 1993. *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Ltd.
Imre Juhász and Ágoston Róth. 2014. A scheme for interpolation with trigonometric spline curves. *J. Comput. Appl. Math.* 263, C (2014), 246–261.
Raph Levien and Carlo H. Séquin. 2009. Interpolating splines: Which is the fairest of them all? *Comput.-Aid. Des. Appl.* 6, 1 (2009), 97–102.
James McCrae and Karan Singh. 2009. Sketching piecewise clothoid curves. *Comput. Graph.* 33, 4 (2009), 452–461.
Yves Mineur, Tony Lichah, Jean Marie Castelain, and Henri Giaume. 1998. A shape controlled fitting method for Bézier curves. *Comput. Aid. Geom. Des.* 15, 9 (1998), 879–891.
Kenjiro T. Miura and Rudrusamy U. Gobithaasan. 2014. Aesthetic curves and surfaces in computer aided geometric design. *Int. J. Autom. Technol.* 8, 3 (2014), 304–316.
Kenjiro T. Miura, Dai Shibuya, Rudrusamy U. Gobithaasan, and Shin Usuki. 2013. Designing log-aesthetic splines with G2 continuity. *Comput.-Aid. Des. Appl.* 10, 6 (2013), 1021–1032.
Albert W. Overhauser. 1968. *Analytic Definition of Curves and Surfaces by Parabolic Blending*. Technical Report SL 68-40. Ford Motor Company.
Alexander P. Pobegailo. 1992. Geometric modeling of curves using weighted linear and circular segments. *Vis. Comput.* 8, 4 (1992), 241–245.
Alexander P. Pobegailo. 2013. Interpolating rational Bézier spline curves with local shape control. *Int. J. Comput. Graph. Anim.* 3, 4 (2013).
Otto Röschel. 1997. An interpolation subspline scheme related to B-spline techniques. In *Proceedings of the Computer Graphics International Conference (CGI'97)*. 131–136.
Malcolm A. Sabin and Neil A. Dodgson. 2005. A circle-preserving variant of the four-point subdivision scheme. In *Mathematical Methods for Curves and Surfaces*. Nashboro Press, 275–286.

Robert Schneider and Leif Kobbelt. 2000. Discrete fairing of curves and surfaces based on linear curvature distribution. In *Curve and Surface Design*. University Press, 371–380.

Carlo H. Séquin, Kiha Lee, and Jane Yen. 2005. Fair, G^2- and C^2-continuous circle splines for the interpolation of sparse data points. *Comput.-Aid. Des.* 37, 2 (2005), 201–211.

Chuan Sun and Huanxi Zhao. 2009. Generating fair, C^2 continuous splines by blending conics. *Comput. Graph.* 33, 2 (2009), 173–180.

Márta Szilvási-Nagy and Teréz P. Vendel. 2000. Generating curves and swept surfaces by blended circles. *Comput. Aid. Geom. Des.* 17, 2 (2000), 197–206.

Hans-Jörg Wenz. 1996. Interpolation of curve data by blended generalized circles. *Comput. Aid. Geom. Des.* 13, 8 (1996), 673–680.

Albert Wiltsche. 2005. Blending curves. *J. Geom. Graph.* 9, 1 (2005), 67–75.

Zhipei Yan, Stephen Schiller, and Scott Schaefer. 2019. Circle reproduction with interpolatory curves at local maximal curvature points. *Comput. Aid. Geom. Des.* 72, 1 (2019), 98–110. DOI: https://doi.org/10.1016/j.cagd.2019.06.002

Zhipei Yan, Stephen Schiller, Gregg Wilensky, Nathan Carr, and Scott Schaefer. 2017. K-curves: Interpolation at local maximum curvature. *ACM Trans. Graph.* 36, 4, Article 129 (2017), 7 pages.

Norimasa Yoshida, Ryo Fukuda, and Takafumi Saito. 2009. Log-aesthetic space curve segments. In *Proceedings of the 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling (SPM'09)*. ACM, 35–46.

Norimasa Yoshida and Takafumi Saito. 2017. Quadratic log-aesthetic curves. *Comput.-Aid. Des. Appl.* 14, 2 (2017), 219–226.

Cem Yuksel, Scott Schaefer, and John Keyser. 2009a. Hair meshes. *ACM Trans. Graph.* 28, 5, Article 166 (2009), 7 pages.

Cem Yuksel, Scott Schaefer, and John Keyser. 2009b. On the parameterization of catmull-rom curves. In *Proceedings of the 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*. ACM, New York, NY, 47–53.

Cem Yuksel, Scott Schaefer, and John Keyser. 2011. Parameterization and applications of catmull-rom curves. *Comput. Aid. Des.* 43, 7 (2011), 747–755.

Received September 2019; revised March 2020; accepted May 2020