CONTROL OF CURVATURE EXTREMA IN CURVE MODELING

A Dissertation

by

ZHIPEI YAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

Chair of Committee,	Dr. Scott Schaefer
Committee Members,	Dr. John Keyser
	Dr. Ergun Akleman
	Dr. Shinjiro Sueda
Head of Department,	Dr. Scott Schaefer

August 2021

Major Subject: Computer Science and Engineering

ABSTRACT

We present a method for constructing almost-everywhere curvature-continuous curves that interpolate a list of control points and have local maxima of curvature only at the control points. Our premise is that salient features of the curve should occur only at control points to avoid the creation of features unintended by the artist. While many artists prefer to use interpolated control points, the creation of artifacts, such as loops and cusps, away from control points has limited the use of these types of curves. By enforcing the maximum curvature property, loops and cusps cannot be created unless the artist intends to create such features.

To create these curves, we analyze the curvature monotonicity of quadratic, rational quadratic and cubic curves and develop a framework to connect such curve primitives with curvature continuity. We formulate an energy to encode the desired properties in a boxed constrained optimization and provide a fast method of estimating the solution through a numerical optimization. The optimized curve can serve as a real-time curve modeling tool in art design applications.

DEDICATION

To those who love maths and computer science.

ACKNOWLEDGMENTS

I would like to thank the Aggie Graphics Group because I enjoyed my PhD program here. I extended my interests from mathematics to computer graphics and experienced lots of fascinating projects in graphics and digital arts. I learnt a lot from my advisor Dr. Scott Schaefer, not only doing research in computer graphics, but also other skills like debugging, presentation, and tricks in Mathematica. Dr. John Keyser, Dr. Ergun Akleman, Dr. Shinjiro Sueda and Dr. Jianer Chen provided a large variety of courses in graphics and visualization, which are really cool. Also I would like to thank Dr. Dean Baskin for his course in differential geometry which is the best course I've ever taken in A&M. Dr. Ligang Liu and Dr. Shizhe Zhou were my advisors when I was an undergraduate student in USTC. Whout their guidance and help, I wouldn't know the world of computer graphics.

When I came to Texas, everything was new to me. My lab mates Hang, Shenyao, Jason, Donghui and Songgang helped me through my first several months in a new environment. Later, we had even more people in AGG. In my spare time, I used to play games with my friends. I'm grateful to Valve and Gearbox, and my friends (in their nicknames): God Mountain, Teacher Xiao, JR, Teacher Nie, BB, BA. The gaming time made me relaxed and energetic for my study and research.

I did internships at Adobe and ByteDance. I'm thankful for the opportunities in the industry and my advisors and colleagues in the companies. I gained a lot of experience in connecting research and engineering applications.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors Dr. Scott Schaefer, Dr. John Keyser and Dr. Shinjiro Sueda of the Department of Computer Science and Engineering and Professor Dr. Ergun Akleman of the Department of Visualization.

The work of Adobe Illustrator's curvature tool in Chapter V and VI was in collaboration with Stephen Schiller, Gregg Wilensky and Nathan Carr from Adobe Research. The two pictures in figure 5.11 are from Adobe. Figure 5.10 is designed by Daichi Ito from Adobe Research.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by the Graduate Assistantship Research from Texas A&M University. No other outside source of funding was provided.

NOMENCLATURE

$\mathbb{R}^n$	Real space of dimension n
S^1	$\mathbb{R} \cup \{\infty\}$
C^n	Algebraic continuity of order n
G^n	Geometric continuity of order n
$f \circ g$	Function Composition
$\Delta(\cdot, \cdot, \cdot)$	Signed area of a triangle
f'	Derivative of function f
f''	Second derivative of function f
$v^\perp$	Perpendicular vector to vector v in counter-clock wise
M^T	Transpose of matrix M
L^1	Space of absolutely Lebesgue integrable functions, i.e. $\int \lvert f \rvert < \infty$
L^2	Space of square-Lebesgue-integrable functions, i.e. $\int \lvert f \rvert^2 < \infty$
CAD	Computer Aided Design
CAGD	Computer Aided Geometric Design
DDG	Discrete Differential Geometry
TOG	Transaction on Graphics
TAMU	Texas A&M University
CSE	Computer Science and Engineering
AGG	Aggie Graphics Group

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Curve modeling has a long history in computer graphics research and industrial applications, finding use in drawing, sketching, data fitting, interpolation, animation, digital arts, architecture, as well as vehicle engineering. This rich application space has led to decades of research for both representing and modifying curves. The goal of such curve representations is to provide the user with control over the shape of the curve while building a curve that has certain geometric properties. These properties may include smoothness, interpolation of various points, and locality.

To define and manipulate these curves, many researchers have relied on polynomial curves, which are a parametric form for shape that has been used extensively [5, 2, 1, 3, 6, 7, 8, 9]. Polynomials have a simple algebraic expression and are C^∞ smooth; so polynomial curves have the advantages of continuity and are easy to compute with. The choices of polynomial basis can lead to different geometric properties including interpolation, smoothness, control of tangent directions, etc [3]. Rational curves are extension to polynomial ones where the polynomial ring is extended to the quotient ring of polynomials.

The algebraic expression of polynomial curves can be expand to a linear combination of the control points and the coefficients are in terms of polynomials of some parameter. The control points can lie on the curve to interpolate or near the curve to approximate its shape. This property, interpolating versus approximating, defines two classes of representations. The left picture of figure 1.1 shows an example of Bézier curves, a typical approximating curve, where the the curve is a linear combination of the control points using the Bernstein basis. In each Bézier curve segment, two end points are on the curve and usually other middle control points are not on the created curve. The middle of figure 1.1 shows a uniform cubic B-Spline curve, which is a widely used approximating curve. Interpolatory curves provide direct control over specific locations and, in some cases, geometric features of a shape by directly manipulating points on the shape. Many research has been done on interpolatory curves. Catmull-Rom [2, 5], interpolatory B-Splines [3] and κ-Curves [4] are examples of interpolatory splines. The right picture in figure 1.1 shows an

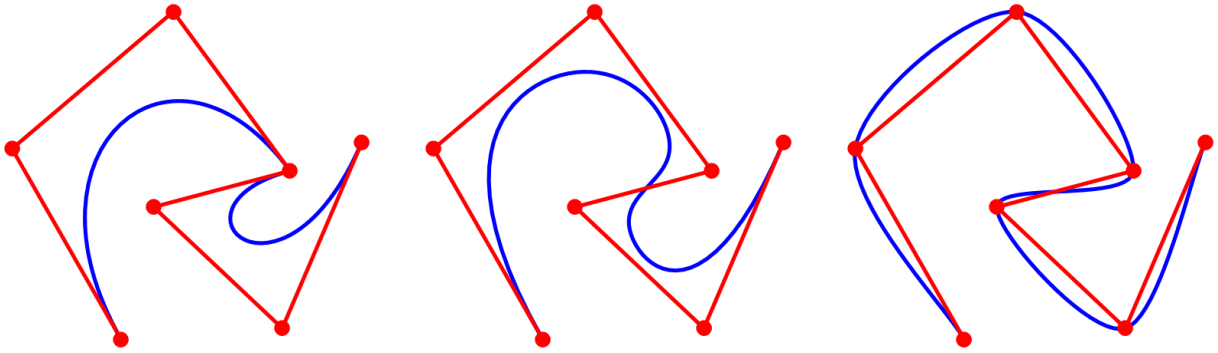

Figure 1.1: Different spline curve examples with the same control points. The control points are high lighted in red while the control polygons are high lighted in green. The blue curves from left to right are: cubic Bézier curves, uniform cubic B-Spline, uniform cubic Catmull-Rom spline.

example of a uniform Catmull-Rom spline where all control points are interpolated. Figure 5.6 shows an example of κ-Curves where local salient points only appear at control points.

In this dissertation we focus on interpolatory curves; that is, curves that pass through a set of control points specified by the user. While much research has concentrated on approximating curves, many users prefer direct control over salient geometric features of the curve such as the position of the curve. Yet interpolatory curves have a maligned past as they can often generate geometric features such as cusps and loops away from control points that the user has a hard time controlling (see Figure 1.2). Hence, it can be difficult for the user to control the placement of these features or even whether or not cusps and loops should exist within the curve.

Our premise is that salient geometric features should appear only at control points for interpolatory curves. Position is one such example of a feature that is automatically enforced in interpolatory curve constructions. However, the question is then: what other features should appear only at control points? Levien et. al. [10] argue that points of maximal curvature are also salient features and human beings are more sensitive to local salient geometric features, e.g. sharp corners rather than flatten points. In other words, humans are more sensitive to maximum curvature points along curves. Indeed, we can see that lack of control over points of maximal curvature has led to many of the historical problems with interpolatory curve constructions. For example, the propen-

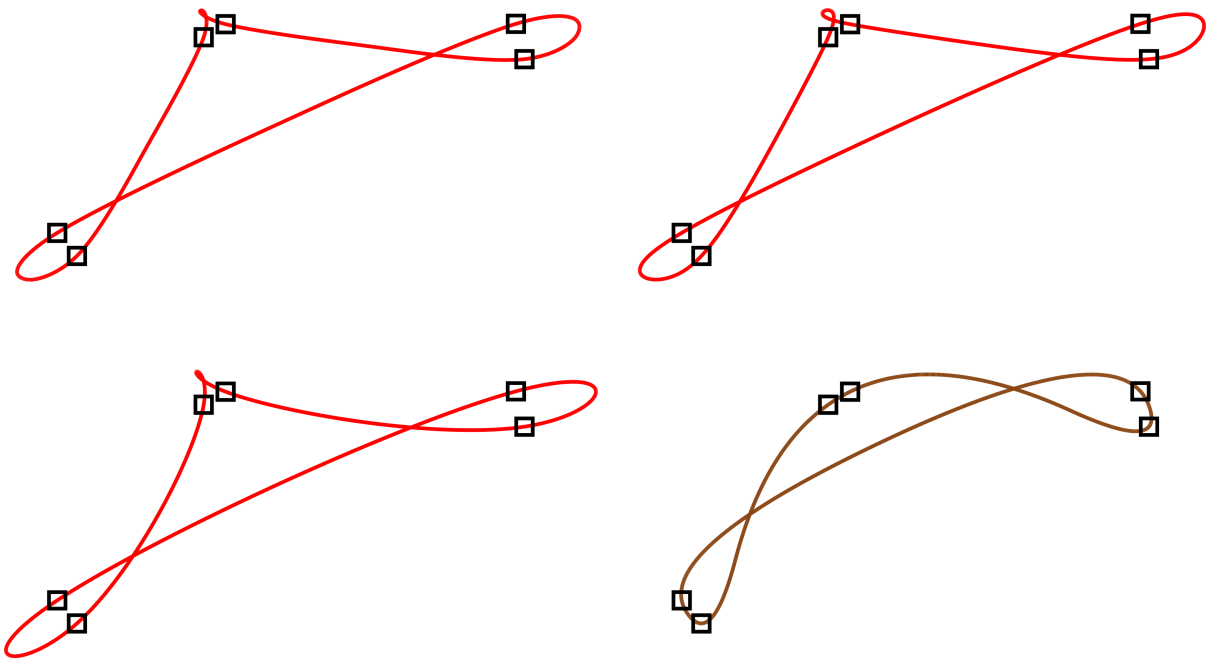

Figure 1.2: Comparison of different C^2 curves with the same control points. First row from left to right: 6-point interpolatory subdivision curve [1], C^2 Catmull-Rom spline [2]; second row from left to right: C^2 interpolating cubic B-spline [3], κ-Curve[4].

sity to produce cusps is due to a local maximum of curvature (in this case, infinite curvature) being produced away from the control points. Hence, it makes sense to identify local maxima of curvature as features of the curve and require that local maxima of curvature appear only at control points. While derivatives are easy to compute, differential properties such as curvature are nonlinear and independent of the parameterization of the curve. Controlling these intrinsic features [11] is difficult.

Beyond simply controlling features of the curve, the types of curves a modeling tool can produce are also important. While polynomial curves are prevalent due to their ease of use, circles are important shapes due to their common occurrence in every day life whether part of images or cylindrical or spherical 3D shapes. Therefore, a useful curve representation [12] would be able to represent shapes such as circles exactly. In particular, if the interpolated control points lie on a circle, even if they are not equally spaced, it would be reasonable to expect that a circle would be

reproduced. When the control points do not lie on a circle, the curve should be "fair" where "fair" means the curvature plot should be simple [13].

Also, because we envision these curves to be controlled by artists, the curve should change continuously under continuous motion of the control points. This last property is trivially satisfied by many interpolatory curve constructions, but not all. For example, Figure 3.1 shows an example of continuous movement of control points for a clothoid curve [9] that produces a discontinuous change of the resulting curve. The curve may suddenly change shape as their control points are dragged), introducing a loop where there was none before.

1.1 Organization and Contribution

We review the general definition of curve and curvature from differential geometry in Chapter 2 and related curve modeling work in Chapter 3. Our contribution starts from Chapter 4. We first analyze the geometric properties and monotonic curvature condition of polynomial quadratic, rational quadratic and polynomial cubic curves in chapter 4. Then we develop our own idea of utilizing the parameter range with monotonic curvature to create shape primitives with desired curvature properties. And in Chapter 5, 6, 7, we compute the curvature at end points of such shape primitives and develop two frameworks to compound a series of the shape primitives in a G^2 way except inflection points, where the continuity is G^1. We should point out that the primary application envisioned for these splines is more for artistic design, as opposed to CAD. Thus the occasional lack of G^2 continuity is not an issue. In one framework we compute a single curve segment that interpolates the input point somewhere in its domain. While in the other framework, we build a curve segment between each pair of two sequential input point in a Hermite interpolation way with predefined tangent and curvature constraints. In both frameworks we propose to create interpolatory curves where the local maximum of the absolute value of the curvature of the curve only appears at control points.

Besides the different shapes of the primitives, we try to involve more degrees of freedom of the curve to provide more shape feature control for users. We extend the quadratic curve in Chapter 5 to rational quadratic curves in Chapter 6 and polynomial cubic curves in Chapter 7. In those two

chapters, we let users to control local sharpness at the input points and partially let users choose the curve direction.

We use an iterative optimization method to rapidly estimate the result by holding some of the constraints as constants and solving for the rest. This fast solution can run in real time for engineering applications with small error tolerance. We build a box constrained optimization in addition to the fast solution to create a refined, converged result.

The contribution of this dissertation can be summarized as

- We create a series of shape primitives with monotonic curvature properties in Chapter 4.
- We build two schemes of connecting multiple curve segments in curvature continuity and interpolating the input points as local curvature maxima in Chapter 5, 6, 7.
- We develop a two-stage numerical method to solve the curve optimization problem with a fast approach and an accurate numerical optimization in Chapter 5, 6, 7.

2. BACKGROUND

2.1 Curves

Generally a curve r is a continuous map [14] from a simple one dimensional domain D to linear space $\mathbb{R}^n$ where $n \leq 1$:

$$r : D \longrightarrow \mathbb{R}^n \tag{2.1}$$

$$t \longmapsto r(t) \tag{2.2}$$

Here the domain can be any simple connected one dimensional space, like real numbers $\mathbb{R}$, intervals $[a, b]$ or a loop S^1 for a closed curve.

The image of the map can also reveal the properties of the curve. A constant map is always a single point. An injection is a curve without self-intersections. If there is only a finite number of points in the domain which are mapped to some image with multiple pre-images, we say the curve has self-intersections. If the images with multiple pre-images have a positive measure, the curve may overlap with part of its self. An example is a doubled ray: $r(t) = (t^2, 0)$. The map can be surjective. Extreme examples are Peano curves and Hilbert curves. They are continuous but non differential curves. In geometric modeling, we usually do not use such fractal curves as modeling tools. So, in this dissertation, we will focus on curves with at least continuous first derivatives except for at most a finite number of points in the domain. And we restrict that, for any point in the domain, there is a neighborhood that the curve is injection.

2.1.1 Parameterization

We can easily modify the domain to its homomorphic space by re-parameterizing the curve. For any continuous map f from $\mathbb{R}$ to $\mathbb{R}$, we can write the re-paremeterized curve $\tilde{r} = r \circ f$ as

$$\begin{aligned} \tilde{r} : \mathbb{R} &\longrightarrow \mathbb{R}^n \\ t &\longmapsto r(f(t)) \end{aligned}$$

Similar to curve segment, we can always map interval $[a, b]$ to $[0, 1]$ by different polynomial functions

$$\begin{aligned} g_n : [a, b] &\longrightarrow [0, 1] \\ t &\longmapsto (\frac{t-a}{b-a})^n \end{aligned}$$

where n can be any natural number. To avoid unnecessary singular points, we require the derivative of the re-parameterization and the derivative of the curve are non zero.

$$\begin{aligned} r'(t) &\neq 0 \\ f'(t) &\neq 0 \end{aligned}$$

The arc length can be computed by integrating the length of the derivative of the curve [14]. Given a curve $r(t)$ on $[a, b]$, the arc length from parameter a to some parameter $t \in [a, b]$ can be defined as

$$s(t) = \int_a^t |r'(u)|\, du, \quad t \in [a, b] \tag{2.3}$$

Notice the arc length $s(t)$ is always increasing about t, so we can write t as an inverse function about s. If we replace t with $t(s)$ in the definition of $r(t)$, we have a curve r in its arc length parameterization:

$$r(s) = r(t(s)), \quad s \in [0, \int_a^b |r'(u)|\, du].$$

The advantage is the derivative of the curve always has unit length with an arc length parameterization:

$$\left|\frac{dr}{ds}\right| = \left|\frac{dr}{dt}\frac{dt}{ds}\right| = \frac{ds}{dt}\frac{dt}{ds} = 1.$$

2.1.2 Continuity

Continuity defines the "smoothness" of a curve. A curve $r(t)$ with a simple parameterization is said to be C^n continuous if the n-th derivative of $r(t)$ is continuous. If curve $r(t)$ is C^n continuous for an arbitrary positive integer n, we say $r(t)$ is C^∞ infinity continuous. C^n continuity is called "parametric continuity". As indicated by the name, the parametric continuity depends on the parameterization. Different parameterizations may result in different parametric continuity. Another type of continuity is called geometric continuity. If we have a curve $r(s)$ in its arc-length parameterization, then the n-th continuous derivative of $r(s)$ indicates the n-th geometric continuity G^n. Similarly we can define G^∞ if the curve is G^n for any positive n.

We are interested in low order continuity: $C^0, C^1, C^2, G^0, G^1, G^2$. Higher order of continuity can provide smoother shapes but it is not necessary for visual fairness for humans [10] and will increase the complexity of curve modeling. C^0 is the same as G^0, which means the curve is simply connected. C^1 means the tangent vector is continuous along the curve and this depends on the parameterization. G^1 only require the tangent directions to be continuous, because the tangent vector is always normalized in arc-length parameterization. C^2 is the second order derivative of the curve. G^2 means the curvature continuity which will be discussed in the following section. Curvature is also an intrinsic property of the curve. In some papers, the curvature continuity is also called "Visual C^2" or VC^2.

2.2 Curvature

Curvature is the measure of how bend a curve is. For 2D and 3D curves, the second derivative of the curve is always parallel to the normal vectors: $\ddot{r}(s) \| n(s)$. We define the curvature $\kappa(s)$ of the curve at parameter s as the signed length of the second derivative of the curve according to the

normal direction:

$$\kappa(s) := \ddot{r}(s) \cdot n(s) = \begin{cases} |\ddot{r}(s)|, & \ddot{r}(s) \cdot n(s) \geq 0 \\ -|\ddot{r}(s)|, & \ddot{r}(s) \cdot n(s) < 0 \end{cases} \tag{2.4}$$

In CAGD, parametric curves usually are not in arc length parameterization. For 2D curves $r(t) = (x(t), y(t))$ with any valid parameterization, the curvature is

$$\kappa(t) = \frac{r''(t) \cdot r'(t)^{\perp}}{|r'(t)|^3} = \frac{x'(t)y''(t) - x''(t)y'(t)}{(x'(t)^2 + y'(t)^2)^{3/2}}. \tag{2.5}$$

The two equations above are the same under re-parameterization. In differential geometry, we have the theorem that:

Theorem 1. *Intrinsic properties like curve length, curvature will not change under re-parameterization or rigid transformation.*

2.2.1 Critical Points of Curvature

The curvature defines how a curve bends. A larger absolute value of curvature means the curve bends more, or visually the curve is more sharp at high curvature points. A low curvature mean the curve looks more flat. A straight line has a constant zero curvature. Since the sign of the curvature depends on whether the curve is "turning left" or "turning right", we are interested in the absolute value of the curvature.

In this dissertation, we call points on a curve "critical points" when the derivative of the curvature at those points are zero

$$\{r(t) | \kappa'(t) = 0\}.$$

So critical points means local maximum or minimum curvature points.

2.2.2 Curvature Control

Curvature distribution over a curve or points of min/max curvature is not something that can be easily controlled for most curve constructions. Levien [10] states that high geometric continuity G^n can provide fair shapes for human perception. However G^3 and G^4 continuity require many

constraints and increase the complexity of the construction. Yet G^2 continuity provides good fairness quality and creates curvature continuity across the object. Curvature continuity may be important for physical reasons, like reducing drag coefficients for aircraft, but also has aesthetic benefits. For example, reflections in car bodies depend on the normal of the shape. For G^1 shapes where the normal changes continuously over the object, reflected lines may have sharp corners at these discontinuities where none may exist within the environment. This lack of continuity is not desirable from an aesthetic perspective and curvature continuity is required to avoid such problem. In this proposal, we will focus on G^2 continuity or curvature continuity.

In addition to curvature continuity, the distribution of the curvature also includes the variation of the curvature. A small variation or a monotonic distribution of curvature usually can provide a more fair shape. We would list some typical types of spline curves or surfaces with curvature control. Levien [10] provides a more complete discussion about curvature based fairness.

2.3 Polynomial Curves

Polynomial curves are curves whose coordinates are in polynomial about the parameter according to some coordinate basis. The advantages of polynomial curves in CAGD are the simple form of the curve and ease of use. The same polynomial curve can be expressed in different ways for the ease of computing. Most polynomial curves share a similar form: a linear combination of control points using polynomial coefficients. We can express the curve in Lagrange basis [15], Bernstein basis, and many other choices. In this dissertation, we will use the Bézier form of polynomial curves in Bernstein basis for their simple algebraic properties.

Typically a Bézier curve with control points $\{P_0, P_1, \cdots, P_n\}$ is defined as

$$r(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i,\ t \in [0,1] \tag{2.6}$$

Bézier curves have lots of good mathematical properties. The curve is always inside the convex hull of the control polygon. Bézier curves can be constructed by interpolating pairs of control points in a hierarchical way which is called blossoming construction. And a Bézier curve in degree

n can always be elevated to degree n+1 by subdividing the control polygon and reassigning the control points. A more detailed discussion about Bézier curves can be found in Farin's textbook in CAGD [3]. We take advantage of degree elevation when constructing a restricted class of cubic curves with monotonic curvature later in this dissertation.

3. RELATED WORK

3.1 Interpolatory Curves

There are large numbers of interpolatory curve constructions that have been developed, and we cannot provide an exhaustive list but refer to Hoschek's book [16] for many such methods. Polynomial and rational functions are commonly used to represent interpolatory curves like Bézier curve, Hermite curves [3], etc. Catmull-Rom splines [2, 5] are one of the more common interpolatory curve representations and are combinations of Lagrange interpolation with B-spline basis functions. Subdivision curves [17, 1] can also be used to model interpolatory splines. Cubic splines, formed from approximating B-splines, interpolate points with C^2 continuity through the solution to a tri-diagonal system of equations [3]. Different data-dependant parameterizations, such as centripetal or chordal parameterizations, can be used to control the shape of these curves as well. In the case of C^1 Catmull-Rom splines, such a choice can guarantee that no cusps appear except at control points [18]. However, these results do not extend to C^2 Catmull-Rom splines. These constructions are easy to create and smooth, but enforce only parametric properties rather than geometric properties such as curvature in their construction. While all of these constructions build interpolatory curves, even with curvature continuity, none allow control over curvature. Figure 1.2 shows a comparison of many of these methods versus our construction. Note that all of these curves create cusps or have local maximum curvature points away from control points except for our curve.

3.2 Curves with Curvature Distribution Control

More related to our method are classes of curves that, not only interpolate control points, but control curvature in some way. Higashi et al. [19] restricted the locations of control points of Bézier curve to get monotonic curvature and generated a C^2 spline. The "Typical-Curve" [20] and "Class A" Bézier curves [6] have monotonic curvature, but few degrees of freedom and can be difficult to control when moving the points. Clothoids, also known as Euler spirals [21, 22, 9],

are perhaps the best known such curves. These curves have the property that the curvature of the curve changes linearly with respect to arc length. Hence, piecewise clothoid curves have local maximum curvature magnitude at the interpolated points. Such curves would be ideal for our purposes except that continuous motion of the control points does not always create a continuous deformation of the curve as is illustrated in Figure 3.1. Log-aesthetic curves [23, 24, 25, 26] are similar to clothoids (including clothoids are a special case) and have curvature plots that increase exponentially with respect to arc length. Levein et al. [10] also describe a two parameter spline family modulo conformal transformations.

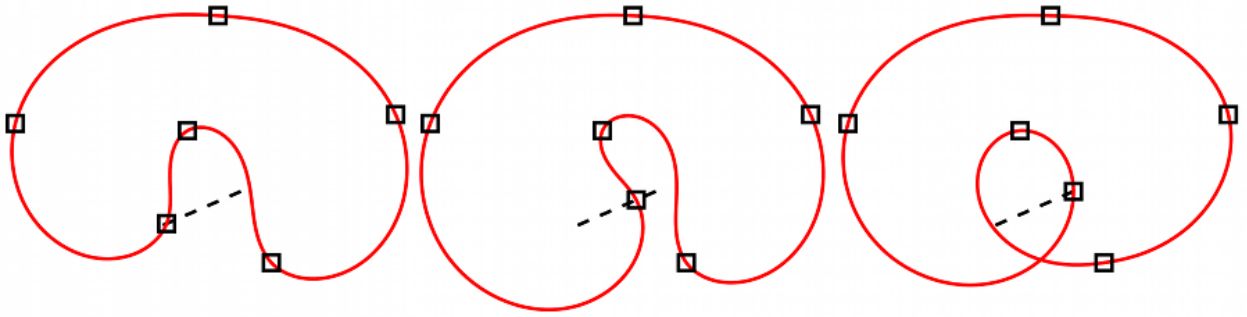

Figure 3.1: Moving one control point continuously for a piecewise clothoid curve can result in a discontinuous change in the curve as shown on the right where the curve suddenly flips over.

3.3 Quadratic Curves

Beyond parametric continuity, researchers have studied the conditions for geometric continuity as well. In addition to controlling curvature, our method starts from piecewise quadratic curves that meet with G^2 continuity. Most curve constructions require cubic curves to generate C^2 or G^2 curves, but G^2 quadratic curves have appeared in the past. Schaback [27] created a piecewise quadratic G^2 Bézier curve to interpolate a list of non-inflecting points, which means the sign of the curvature of the curve should be always positive or always negative. This approach creates a quadratic Bézier curve between interpolated points but tends to produce flat curves at the interpolated points. Feng et al. [7] modified this approach and build a G^2 quadratic curve to interpolate

a list of points with associated tangent directions where the end points of each quadratic appear between interpolated points. Each single curve segment interpolates a control point on the interior of the parameter range for the curve while remaining G^2 at the join points for the curve. Like Schaback, Feng's approach is restricted to non-inflecting points, though their numerical solver can be applied on an "S" shape. Gu et al. [8] uses quadratic Bézier curve to interpolate a list of points as G^1 joints with arbitrary tangent directions. Our approach to creating G^2 quadratic curves differs from all these approaches, and we develop an explicit solution of the join point between two quadratics to enforce G^2 continuity. In addition, we consider the added condition that control points are interpolated at maximal curvature magnitude locations.

We [4] utilize Feng's approach to interpolate a series of points at local maximum curvature points on a 2D plane as a curve modeling tool called "κ-Curves". The quadratic Bézier curve components are parabolas and all have a unique max curvature point. κ-Curves enforce a local maximum curvature at all input points. However, we relax the G^2 condition at join points (where two Bézier curves meet) that form inflection points. Instead we require that the absolute value of curvature is continuous, which means that κ-Curves can reproduce curves with inflection points. We enumerate the desired geometric criteria and use an iterative method to create a curve with those properties. Unfortunately, the iterative method does not guarantee convergence of the result, though the method appears to work well in practice. In addition, the polynomial nature of their representation means that common shapes such as circles cannot be reproduced.

3.4 Elliptical Shapes

Arc or circular splines are widely used to represent circles. Hoschek [28] inserts an arc of a circle between each pair of adjacent control points and connect all arcs with G^1 continuity at the control points. Meek [29], Yeung [30], and Kurnosenko [31] utilize biarcs, two circular arcs connected using G^1 point, between two control points and interpolate given tangent vectors at the control points where the curve is G^1. Meek [32] uses C-Shape curves of an arc and a conic to interpolate points, tangent vectors, and curvatures. Piegl [33] developed a method to estimate the tangent directions at control points and then interpolate data using biarcs. While these methods

can obviously reproduce circles, the curves lack curvature continuity and do not allow the user to control points of local maximum curvature.

Many non-polynomial methods have been developed to represent circles as well. Wenz [34], Sequin [35] and Sun [36] compute local circle arcs using three adjacent control points and then blend each pair of the adjacent circles to obtain a C^2 spline though the authors do not control the placement of curvature maxima. Schaefer [37] creates interpolatory curves through subdivision that can reproduce circles, though the method is sensitive to the parametric spacing of points along a circle.

Rational quadratic polynomial curves form conic sections and have the capability to represent circles exactly. Xu [38] and Canton [39] researched the geometric properties of conic sections in rational Bézier form. Similar to [27], [40] explored the existence and properties of interpolating a set of non-inflecting points using G^2 connected conic splines. Yang [41] compute a G^1 quadratic interpolatory splines and then adjust the tangent vectors at control points and the weights of the Bézier control points to achieve a G^2 curve. In this paper, we use G^2 almost everywhere connected conic splines to interpolate a series of 2D points with local max curvature points only occur at the interpolated points. Unlike many papers, our input points do not need to be non-inflecting. At inflection points, we maintain continuity in the magnitude of curvature, though the sign of curvature inverts.

4. CURVE PRIMITIVES

We'll start from planar curve primitives with their geometric properties. Some properties like arc length, curvature can be extended to 3D spatial curves or higher dimensional curve. However, we focus on planar curves and leave higher dimensional curves as further research.

Quadratic and cubic curves are widely used in geometric modeling, art design, architectures, and related areas due to its simple algebraic structures and curvature distributions. This chapter will mainly talk about the construction, curvature and other intrinsic properties of quadratic curves, and cubic curves.

4.1 Polynomial Quadratic Curves

A non degenerate polynomial quadratic curve is always a parabola or part of a parabola. We define a polynomial quadratic curve by three distinct planar points $\{Q_0, Q_1, Q_2\} \subset \mathbb{R}^2$ in Bernstein basis about parameter $t \in [0, 1]$:

$$r_2(t) = (1-t)^2 Q_0 + 2(1-t)tQ_1 + t^2 Q_2 \quad t \in [0, 1]. \tag{4.1}$$

Equation 4.1 shows the definition of our polynomial quadratic curve. When the three control points are co-linear, including three points are on a straight line, two points collapsed or all points collapsed, the curve will be a straight line in Euclidean space. We call the curve degenerate in these cases. Usually three distinct co-linear points will not affect the discussion about the algebraic and geometric properties in the following sections.

If two or all three points collapsed, the collapsed points may become singular points and the some algebraic values like curvatures may have a zero denominator at those points. To avoid these degeneration, we require all control points to be distinct and discuss collapsed cases separately.

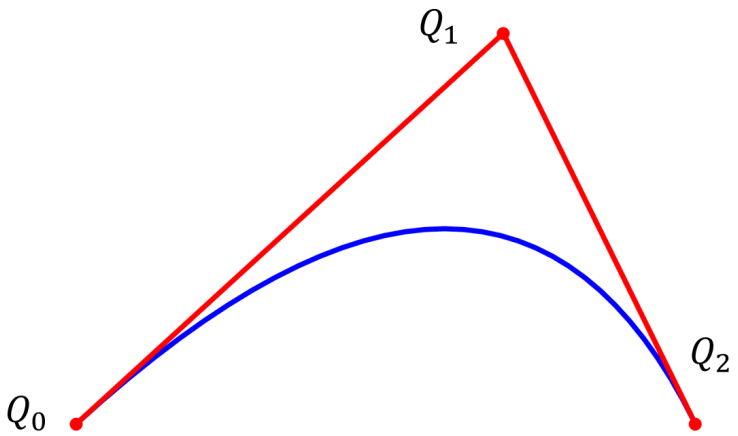

Figure 4.1: Notation of a polynomial quadratic curve.

4.1.1 Curvature Distribution

The curvature of a parabola in Bézier form has a simple expression. We substitute the definition (4.1) into the curvature formula (2.5) and get

$$\kappa_2(t) = \frac{\Delta(Q_0, Q_1, Q_2)}{||(1-t)(Q_1 - Q_0) + t(Q_2 - Q_1)||^3} \tag{4.2}$$

Directly we have the curvatures at two end control points Q_0 and Q_2 are

$$\kappa_2(0) = \frac{\Delta(Q_0, Q_1, Q_2)}{||Q_1 - Q_0||^3} \tag{4.3}$$

$$\kappa_2(1) = \frac{\Delta(Q_0, Q_1, Q_2)}{||Q_2 - Q_1||^3} \tag{4.4}$$

So when the triangle $\Delta(Q_0, Q_1, Q_2)$ is degenerate, the curvature is constant zero. For non degenerate cases, the signed area of the triangle $\Delta(Q_0, Q_1, Q_2)$ is fixed so the curvature of the curve is always positive or negative. The intuitive understanding is a parabola is always convex. If we extend the parameter's range to $(-\infty, +\infty)$, the absolute value of the curvature will increase from zero to some positive number then decrease to zero. We can compute the parameter value $\tilde{t}$ corresponding to the maximum absolute curvature by computing the root of the derivative of the

curvature: $\kappa_2'(t) = 0$ as

$$\tilde{t} = \frac{(Q_0 - Q_1) \cdot (Q_0 - 2Q_1 + Q_2)}{||Q_0 - 2Q_1 + Q_2||^2}. \tag{4.5}$$

4.1.2 Curvature Monotonicity and Max Curvature Points

The curvature of a parabola is monotonic on each side of the critical point at parameter $\tilde{t}$ in the last section. For quadratic curve restricted on the interval [0, 1], if the parameter $\tilde{t}$ is outside interval [0, 1], we obtain a quadratic segment with monotonic curvature. To get a more clear view when the curve $r_2(t), t \in [0, 1]$ has monotonic curvature, we compute a necessary and sufficient condition.

Without loss of generality, we can transform these control points to a canonical coordinate system via a rigid transformation to $\{(0, 0), (a, 0), (x, y)\}$ where $a \geq 0$. Ridid transformations do not affect curvature, so any curvature properties are maintained by this transformation. Hence, this quadratic has few degrees of freedom, which makes analysis simpler. If we compute $\kappa_2'(t) = 0$ and solve for the parameter $\tilde{t}$ corresponding to the critical point, we find that the parameter corresponding to the single critical point of curvature appears at

$$\tilde{t} = \frac{a(2a - x)}{(x - 2a)^2 + y^2}.$$

If we assume $y \geq 0$, the curvature of curve $r_2(t)$ will go up from zero to some positive number $\kappa_2(\tilde{t})$ on interval $t \in (-\infty, \tilde{t}]$ and down to zero on $t \in [\tilde{t}, +\infty)$. Restricted on $[0, 1]$, the derivative of curvature should be less than or equal to zero to create a monotonically decreasing curvature along this curve. When $x \geq 2a$, given that the denominator is always greater than or equal to zero, $\tilde{t} \leq 0$, the quadratic curve will have monotonically decreasing curvature. When $x < 2a$ and $y^2 > (2a - x)(x - a)$, we have $\tilde{t} \in (0, 1)$, i.e. the local maximum curvature point always appears on the interior of the curve. And when $y^2 \leq (2a - x)(x - a)$, $\tilde{t}$ is always greater than one so the curvature is increasing along the curve and the free end of the curve is the maximum curvature point.

We can summarize the algebraic result above as following:

Theorem 2. *In a set of quadratic control points* $\{Q_0, Q_1, Q_2\}$*, if the angle* $\angle Q_0Q_1Q_2$ *is an acute or right angle, the quadratic Bézier curve must have one local maximum curvature point on it interior with parameter* $t \in (0, 1)$*. When angle* $\angle Q_0Q_1Q_2$ *is obtuse angle, assuming the edge* Q_1Q_2 *is equal or longer than edge* Q_0Q_1*, we can project point* Q_2 *to line* Q_0Q_1 *at* $\tilde{Q}_2$*. If* $|Q_1\tilde{Q}_2| \geq |Q_0Q_1|$*, then the curvature is monotonically decreasing from* Q_0 *to* Q_2 *and* Q_0 *is the maximum curvature point on parameter range [0,1].*

Corollary 2.1. *If*

$$\min\left(\frac{|Q_0Q_1|}{|Q_1Q_2|}, \frac{|Q_1Q_2|}{|Q_0Q_1|}\right) \leq -\cos\angle Q_0Q_1Q_2,$$

the curvature on the parameter range [0, 1] is monotonic.

Figure 4.2 shows this case, geometrically.

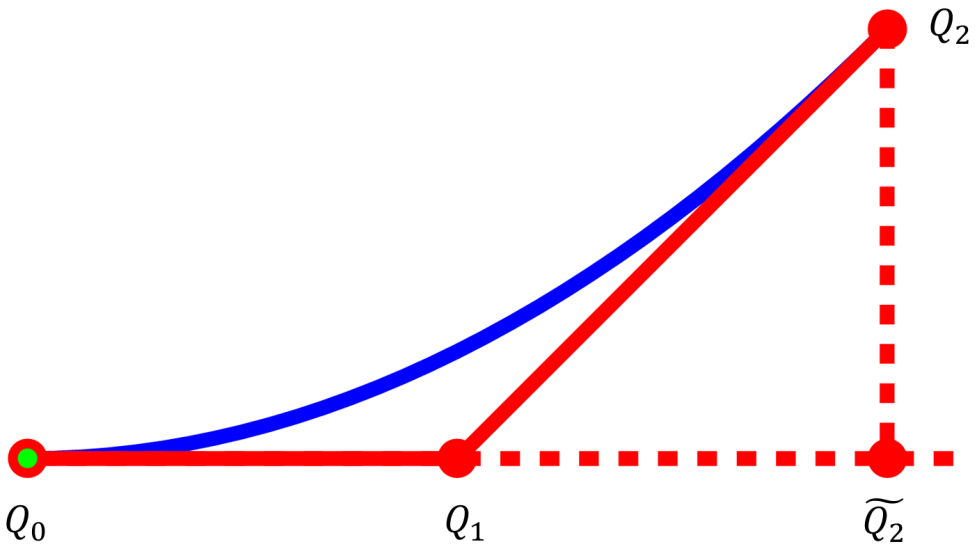

Figure 4.2: Geometric condition of a quadratic curve with monotonic curvature. $|Q_0Q_1| = |Q_1\tilde{Q}_2|$ and the green spot is the critical point of the parabola.

4.2 Rational Quadratic Curves

A general rational quadratic curve of control points $\{Q_0, Q_1, Q_2\}$ associating weights $\{w_0, w_1, w_2\}$ is defined as

$$r_{2r}(t) = \frac{(1-t)^2 w_0 Q_0 + 2(1-t)t w_1 Q_1 + t^2 w_2 Q_2}{(1-t)^2 w_0 + 2(1-t)t w_1 + t^2 w_2}$$

It can be proved that we can re-parameterize and re-weight the curve above such that $w_0 = w_2 = 1$ without changing the image of the curve [3]. So we can and should define our rational quadratic in such *standard form* [3] as

$$r_{2r}(t) = \frac{(1-t)^2 Q_0 + 2(1-t)t w Q_1 + t^2 Q_2}{(1-t)^2 + 2(1-t)t w + t^2}, \quad w > 0 \tag{4.6}$$

If we restrict the parameter t in [0,1], the curve is defined within the triangle $\Delta(Q_0, Q_1, Q_2)$. When the weight $w = 0$, the middle control point Q_1 will not affect the curve and the curve will be the line segment Q_0Q_2. When we increase the weight w to 1, curve (4.6) will be exactly the polynomial quadratic curve. When w goes to $+\infty$, the curve will approach the control polygon $\{Q_0, Q_1, Q_2\}$. Figure 6.1 shows different weights of curve (4.6).

4.2.1 Relationship to Conic Sections

A general planar quadratic or a conic section on a 2D plane can be written general Cartesian form as

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \tag{4.7}$$

The non degenerate condition of the quadratic function (4.7) is A, B and C are not all zeros at the same time. All the following discussion assumes the quadratic is not degenerate.

The discriminant of equation (4.7) is

$$B^2 - 4AC,$$

which determines the geometric shape of the curve. The geometric shape of equation (4.7) can

be ellipse, including circles, parabolas and hyperbolas. The exact shape depends on the discriminant. Table 4.1 shows the type of the shape according to the conditions on the coefficients in equation (4.7).

$B^2 - 4AC$	Shape
$=0$	parabola
<0	ellipse when A=C and B=0, circle
>0	hyperbola

Table 4.1: Quadratic shapes based on the discriminant.

It can be proved that the curve defined by equation (4.6) is always a conic section. Equation (4.6) is the parametric definition of a conic section while equation (4.7) is the implicit definition. We use the implicit form of a conic section to help restrict the shape of a curve in later sections. If we restrict the parameter t on interval [0, 1], the curve segment is always inside the polygon of the control points. And if we extend the parameter to $\mathbb{R}$, the curve define by equation (4.6) can be extended to the full curve of a hyperbola, parabola, or an ellipse. Notice when the curve is part of an ellipse, the domain $\mathbb{R}$ will only define an ellipse without a single point. Only when we add the infinity point to $\mathbb{R}$ so that the domain is

$$\bar{\mathbb{R}} = \mathbb{R} \cup \{\infty\},$$

the curve can be a closed ellipse.

On the other side, for any given conic section, pick two distinct points on the curve. If the curve is a hyperbola, the two points need to be on the same branch. If the tangent vectors at the two points are not parallel, we can always take the two points as $\{Q_0, Q_2\}$ and pick the intersection of the two tangent vectors as Q_1. There is a unique weight w such that the control points $\{Q_i\}$ together with weight w can reproduce the conic section by equation (4.6).

4.2.2 Curvature

The curvature of curve $r_{2r}(t)$ can be computed from equation (2.5) as

$$\kappa_{2r}(t) = \frac{r''_{2r}(t) \cdot r'_{2r}(t)^{\perp}}{|r'_{2r}(t)|^3}. \tag{4.8}$$

Since curve $r_{2r}(t)$ is defined by rational quadratic function about t, the full algebraic expression of (4.8) can be quite complicated. However we can evaluate the curvature distribution of the curve by the curvature of conic sections because the rational quadratic curves are always part of conic sections.

A non-circle ellipse always have two local maximum curvature points and two minimum curvature points. Between two sequential curvature extrema, the curvature is monotonic. If the ellipse is a circle, then the curvature is constant and is the reciprocal of the radius. If the curve is a parabola, the curvature has been discussed in section 4.1. And a pair of hyperbolas has one critical point on each branch.

Restricted on domain $[0, 1]$, a parabola or hyperbola segment can have a local maximum curvature point or have monotonic curvature. An elliptical curve segment can have monotonic curvature or have at most two critical points. Figure 4.3 shows all eight cases.

From algebraic standpoint, we can also compute the critical points, which will be used in the curve construction sections. We try to find all critical points with $\kappa'_{2r}(t) = 0$ from equation (4.8).

$\kappa'_{2r}(t)$ is a fraction of a polynomial and the square root of another polynomial in the variable t. The denominator of this expression depends solely on the length of the tangent $r'_{2r}(t)$, which is non-negative everywhere. Hence, we only need to consider the numerator of $\kappa'_{2r}(t)$ when analyzing its roots. The numerator of equation (4.8) is

$$\det(r'_{2r}(t), r''_{2r}(t))'(r'_{2r}(t) \cdot r'_{2r}(t)) - \frac{3}{2}\det(r'_{2r}(t), r''_{2r}(t))(r'_{2r}(t) \cdot r'_{2r}(t))' \tag{4.9}$$

This expression is a quartic polynomial in t. When $w = 1$, the leading coefficient of the numerator

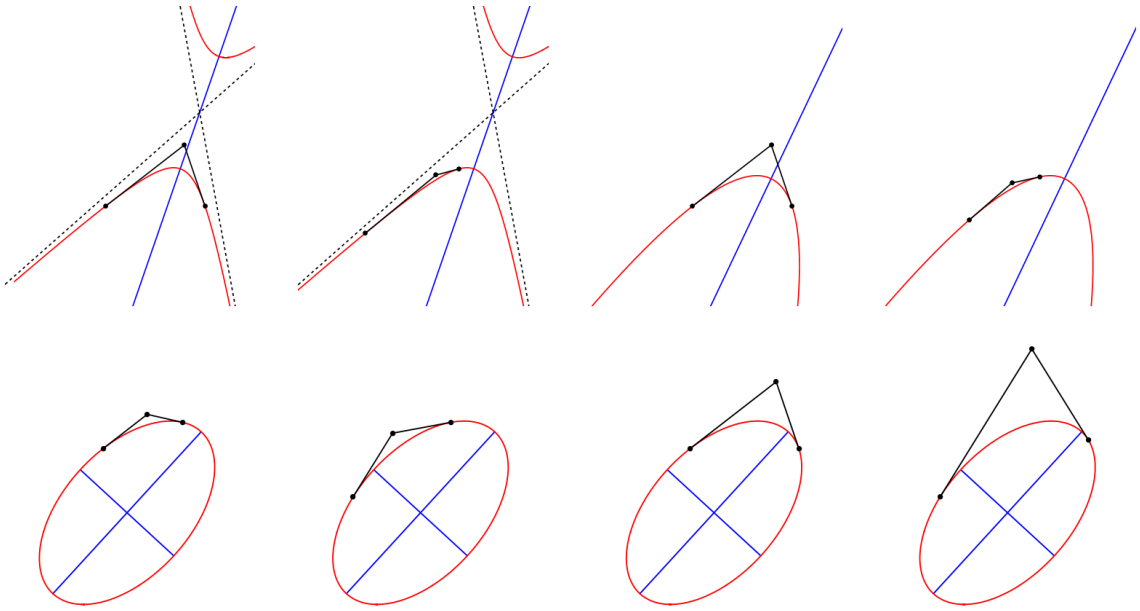

Figure 4.3: All eight cases of critical points a rational quadratic curve segment can have. Rational Bézier curves may have 0, 1 or 2 critical points of curvature within the parameter range (0,1). Solid black lines are Bézier control polygons, red lines are rational Bézier curves, blue lines are axes of symmetry for the conic sections. The first row shows a hyperbola (first two images) and a parabola (last two images), each of which can have one or zero max curvature points within a rational Bézier curve. The last row shows an ellipse, which may have 0, 1 min, 1 max, or 1 min and 1 max curvature points.

is zero and the expression degenerates to a cubic polynomial. Rewriting the numerator in the Bézier basis yields

$$\begin{bmatrix} (1-t)^4 \\ 4(1-t)^3 t \\ 6(1-t)^2 t^2 \\ 4(1-t)t^3 \\ t^4 \end{bmatrix}^T \cdot \begin{bmatrix} -2w^3|Q_0 - Q_1|^2 + w(Q_0 - Q_1)\cdot(Q_0 - Q_2) \\ -w^2|Q_0 - Q_1|^2 + \frac{1}{4}|Q_0 - Q_2|^2 \\ \frac{1}{2}w(Q_2 - Q_0)\cdot(Q_0 - 2Q_1 + Q_2) \\ -\frac{1}{4}|Q_0 - Q_2|^2 + w^2|Q_1 - Q_2|^2 \\ -w(Q_0 - Q_2)\cdot(Q_1 - Q_2) + 2w^3|Q_1 - Q_2|^2 \end{bmatrix}. \tag{4.10}$$

We care about the real roots of equation (4.10) in interval $[0, 1]$. A general quartic function has up to 4 real roots. Equation (4.10) may have 0, 1 or 2 roots in range [0,1] depending on the parameter w, which controls whether the curve is piece of a hyperbola, parabola, or ellipse. The

first two shapes can only have zero or one critical points of curvature in the interval $(0, 1)$. In the elliptical case, equation (4.10) can have up to four roots corresponding to the two maximal and two minimal points of curvature on an ellipse, but at most two of these roots, corresponding to one maximal and one minimal curvature point, can fall into the $(0, 1)$ interval.

Though the full expression of the curvature of a rational quadratic curve can be quite complex, the curvature at two end points are simple:

$$\kappa_2(0) = \frac{1}{w^2}\frac{\Delta(Q_0, Q_1, Q_2)}{||Q_1 - Q_0||^3} \tag{4.11}$$

$$\kappa_2(1) = \frac{1}{w^2}\frac{\Delta(Q_0, Q_1, Q_2)}{||Q_2 - Q_1||^3} \tag{4.12}$$

Compared to equation (4.4), the only difference is the weighted coefficient. When $w = 1$, the rational case meets the polynomial case.

4.3 Polynomial Cubic Curves

4.3.1 General Cubic Curves

Cubic Bézier curves can be much more complex. A general cubic curve in the Bézier form is defined by four points $\{C_0, C_1, C_2, C_3\}$ as

$$r_3(t) = (1-t)^3 C_0 + 3(1-t)^2 t C_1 + 3(1-t)t^2 C_2 + t^3 C_3.$$

Then the curvature of the cubic curve is

$$\kappa_3(t) = \frac{f}{g^{3/2}},$$

where f is a quadratic polynomial about t and g is a quartic polynomial about t. The full expression of f and g is easy to compute but long, so we omit the details. We can compute the derivative of the curvature as

$$\kappa_3'(t) = \frac{f'g - \frac{3}{2}fg'}{g^{5/2}}.$$

The denominator is non-negative and numerator is a polynomial of degree 5. So theoretically a cubic curve can have up to 5 curvature extrema if we expand the parameter t from [0, 1] to all real numbers $\mathbb{R}$. Figure 4.4 shows examples of one, two and three curvature critical points on cubic curves. This complexity of a cubic contrasts with the quadratic, which only had a single, linear

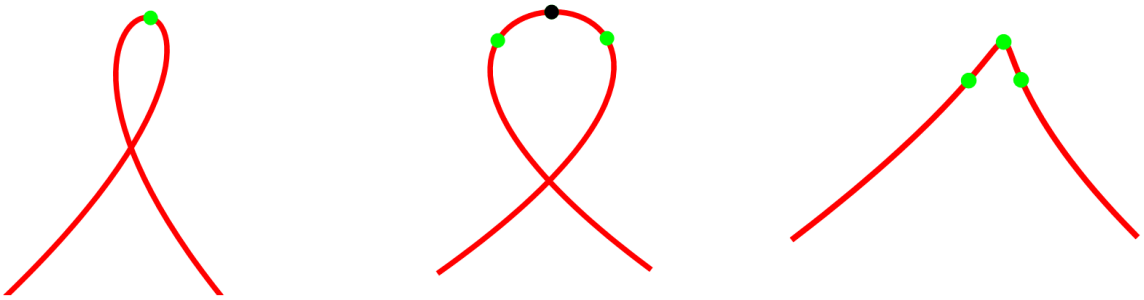

Figure 4.4: Examples of cubic curves with different critical points. The left and right picture have one and three local maximum curvature points in green. The middle one has two local maximum curvature points in green and one local minimum curvature point in black.

term in its numerator. Because of this complexity, it is hard to restrict some cubic curve to have only one local maximum curvature point. However it is much easier to define a cubic curve that does not contain any curvature extrema within its parameter range as we will show.

4.3.2 Restricted Cubic Curves

We note that the quadratic and cubic curves are related to one another through degree elevation. Therefore, there exists some choice of cubic control points that causes the degree of the numerator of the derivative of curvature to collapse. We exploit this idea and begin with the degree elevation formula for a quadratic curve.

To differ from the notations in quadratic curves, we use $\{C_0, C_1, C_2, C_3\}$ as the cubic control points in Bézier form. For arbitrary quadratic curve control by $\{Q_0, Q_1, Q_2\}$, we can always elevate

the curve to cubic with the control points

$$\begin{cases} C_0 &= Q_0 \\ C_1 &= \frac{1}{3}Q_0 + \frac{2}{3}Q_1 \\ C_2 &= \frac{2}{3}Q_1 + \frac{1}{3}Q_2 \\ C_3 &= Q_2 \end{cases} \tag{4.13}$$

without changing the curve, i.e. the two curves are identical after re-parameterization.

We intend to analyze the curvature monotonicity of the elevated curve (4.13). The critical case of the quadratic to have monotonic curvature is when Q_0 is the critical vertex of the parabola, i.e. the case shown in figure 4.2, where point Q_1 is the middle point of edge $Q_0\tilde{Q}_2$. In this case, if we apply the elevation (4.13) to the case in figure 4.2, we have another geometric relationship of the two middle points C_1 and C_2. Denote $\tilde{C}_2$ as the projection of C_2 to line Q_0Q_1, then C_1 is the middle point of line segment $C_0\tilde{C}_2$ as shown in figure 4.5. We will use this middle point location to substitute the definition of C_1 in equation (4.13).

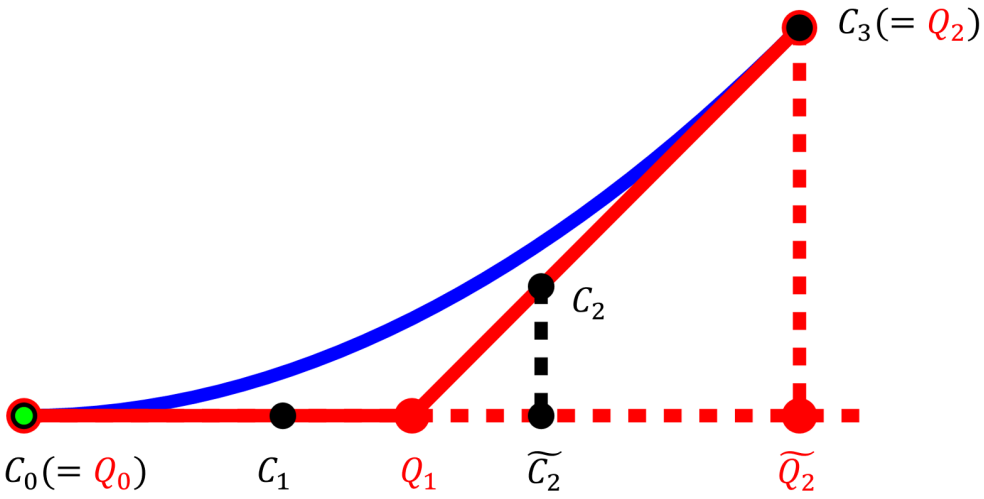

Figure 4.5: Elevation to cubic curve from a quadratic curve. In this picture, Q_1 is the middle point of line segment $Q_0\tilde{Q}_2$ so that Q_0 is the critical vertex of the parabolas. The red points and lines are the quadratic control polygon.The black points are cubic control points. The green spot is local maximum curvature of the blue curve.

We propose our restricted cubic curve as following: given any three distinct points $\{Q_0, Q_1, Q_2\}$, we can uniquely compute four cubic control points as

$$\begin{cases} C_0 &= Q_0 \\ C_1 &= \frac{C_0+\tilde{C_2}}{2} = \frac{1}{2}\frac{(\frac{2}{3}Q_1+\frac{1}{3}Q_2-Q_0)\cdot(Q_1-Q_0)}{(Q_1-Q_0)\cdot(Q_1-Q_0)}(Q_1-Q_0)+Q_0 \\ C_2 &= \frac{2}{3}Q_1+\frac{1}{3}Q_2 \\ C_3 &= Q_2 \end{cases}. \tag{4.14}$$

4.3.3 Curvature

The reason we manufactured the curve in the way above is that the curvature distribution can be extremely simple if we place C_1 at the location in equation (4.14). The intrinsic properties of polynomial curves will not change via translation and rotation. To make the formula simpler, we can always put Q_0 at (0, 0), Q_1 on the positive semi axis of X-axis at $(a, 0)$, here a is some positive number. And Q_2 can be any 2D point (x, y). Then the four cubic control points will be $\{(0,0), (\frac{a}{3}+\frac{x}{6}, 0), (\frac{2a+x}{3}, \frac{y}{3}), (x, y)\}$.

We can compute the curvature of $c(t)$ from Equation 2.5 as

$$\kappa_3(t) = \frac{8y(2a+x+3(2a-x)t^2)}{\sqrt{(2a+x-3(2a-x)t^2)^2+(4yt)^2}^3},$$

and the derivative of the curvature is

$$\kappa_3'(t) = \frac{96yt(2(2a+x)(4a^2-x^2-2y^2)+(x-2a)(12a^2-3x^2+4y^2)t^2+9(x-2a)^3t^4)}{\sqrt{(2a+x-3(2a-x)t^2)^2+(4yt)^2}^5}$$

Since the curve can be flipped along the x-axis, we assume $y > 0$ and the sign of $k_3'(t)$ is decided by one term of its numerator

$$num(t; x, y) = 2(2a+x)(4a^2-x^2-2y^2)+(x-2a)(12a^2-3x^2+4y^2)t^2+9(x-2a)^3t^4$$

where $t \in [0,1]$. The sign of $num(t;x,y)$ can be classified into four cases according to the location of (x,y). $num(t;x,y)$ has only constant, quadratic and quartic terms of t. Therefore, this equation is exactly quadratic in t^2. The sign of the quartic coefficient, and the function value at 0 and 1 will help computing the sign of function f on the interval [0,1]. We have

$$num(0;x,y) = 2(x+2a)(4a^2 - x^2 - 2y^2) \quad (4.15)$$

$$num(1;x,y) = 4((x-a)(x-2a)(x-10a) - 4ay^2). \quad (4.16)$$

4.3.3.1 S Shape

When $x > 4a$, we have the x-coordinate of C_1 larger than Q_1. So the control polygon $\{C_i\}$ is no longer convex, and forms an "S" shape.

When $x > 10a$ and $|y| < \sqrt{\frac{(x-a)(x-2a)(x-10a)}{4a}}$, we have $num(0) < 0$ and $num(1) > 0$. Assuming $y > 0$, the curvature $\kappa(t)$ from 0 to 1 is going down from some positive number $\kappa_3(0)$ to some negative number, then going up to a negative number $\kappa_3(1)$. Hence there is a local maximum curvature point near C_3.

When $4a < x \leq 10a$ or ($x > 10a \,\&\&\, |y| > \sqrt{\frac{(x-a)(x-2a)(x-10a)}{4a}}$), we have $num(t) \leq 0$ on $t \in [0,1]$. So $\kappa(t)$ decreases from some positive number $\kappa(0)$ to some negative number $\kappa(1)$.

4.3.3.2 Convex Shape

When $-2a < x < 4a$, the shape of the control polygon $\{C_i\}$ is always convex.

When $2a < x < 4a$ or ($-2a < x < 2a \,\&\&\, \frac{x^2}{4a^2} + \frac{y^2}{2a^2} > 1$), $num(t)$ is negative on [0,1], curvature $\kappa(t)$ is positive and decreasing on [0,1]. So the curvature is monotonic and C_0 is the maximum curvature point on [0,1].

When $\frac{x^2}{4a^2} + \frac{y^2}{2a^2} < 1$, we have $num(0) > 0$. So C_0 is a local minimal curvature point. Inside the region of the ellipse $\frac{x^2}{4a^2} + \frac{y^2}{2a^2} = 1$, if $|y| < \sqrt{\frac{(x-a)(x-2a)(x-10a)}{4a}}$ we have $num(t)$ positive on [0,1]. So the curvature of the curve is monotonically increasing from C_0 to C_3. And if $|y| > \sqrt{\frac{(x-a)(x-2a)(x-10a)}{4a}}$, there is some $\tilde{t}$ s.t $num(t)$ is positive on $(0,\tilde{t})$ and negative on $(\tilde{t},1)$. So the curvature of the curve is increasing and then decreasing, i.e. there is a local maximum

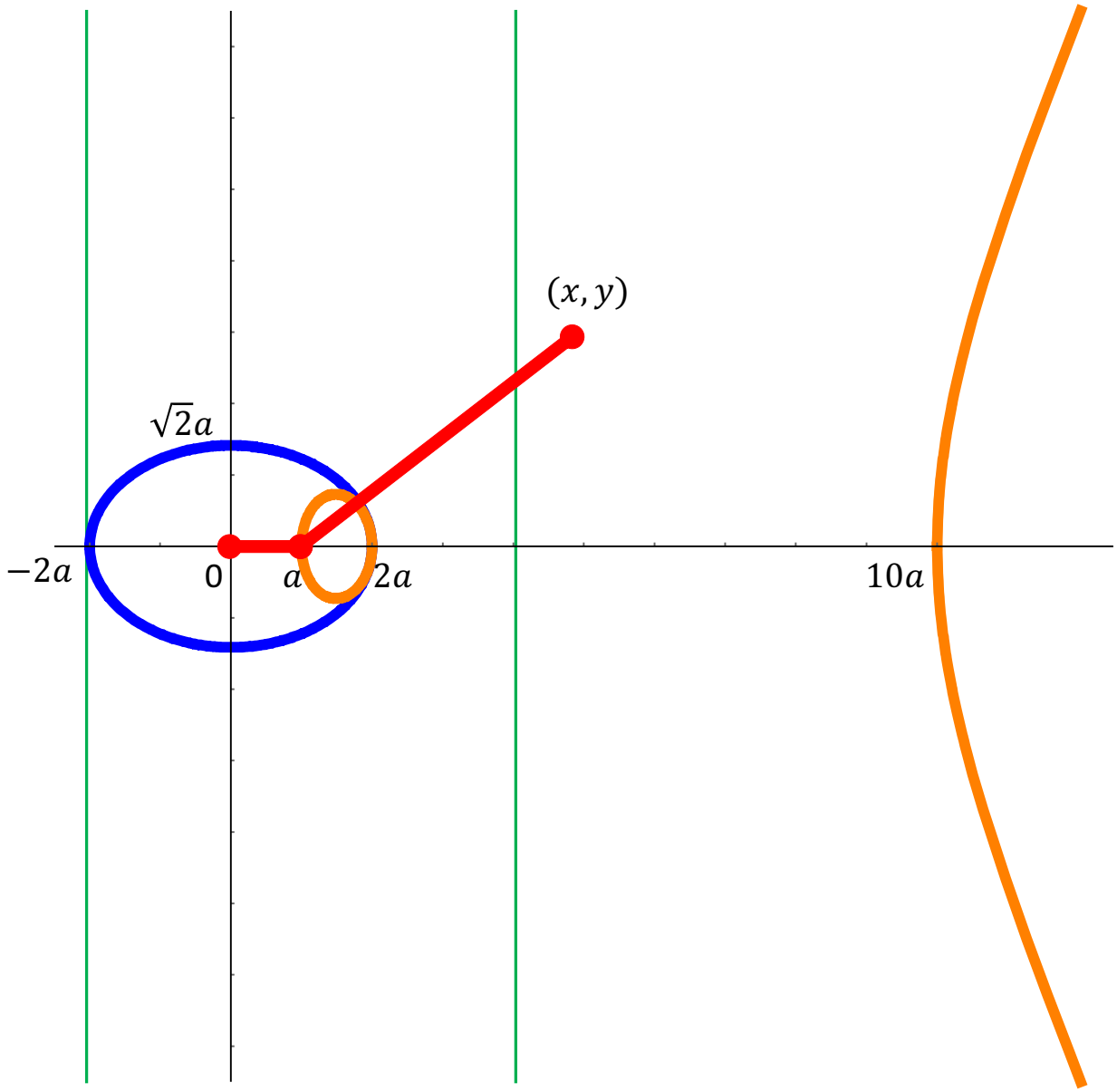

Figure 4.6: The regions of different monotonicity of our cubic curve.

curvature point on the interior of the curve segment.

4.3.3.3 Invalid Shape

When $x < -2a$, ray Q_0Q_1 is to the opposite direction of ray $P0P_1$, and the tangent constraint can't be satisfied. So this case should be discarded.

4.3.4 Edge-Angle Based Definition

We introduced the construction of our restricted cubic curve in the way of elevating and modifying a quadratic curve control by three points. But there are still other ways to construct the same curve using different views. Generally any cubic Bézier curve can be defined by its control points, or edge lengths and angles between edges, or treating the edges as vectors. In this section we will review our curve with other construction methods.

According to figure 4.5, denote θ as the angle from vector C_0C_1 to C_1C_2 and ϕ as the angle from vector C_1C_2 to vector C_2C_3. Since point C_1 is the middle point of line segment $C_0\tilde{C}_2$, we have

$$\cos\theta = \frac{|C_0C_1|}{|C_1C_2|}.$$

In triangle $\Delta C_1Q_1C_2$, we have

$$\frac{\sin\theta}{|C_2C_3|/2} = \frac{\sin(\theta+\phi)}{|C_1C_2|}$$

and

$$0 < \phi < \frac{\pi}{2} - \theta.$$

We can simplify the relationships between then lengths as:

$$\begin{aligned} |C_1C_2| &= \frac{|C_0C_1|}{\cos\theta} \\ |C_2C_3| &= 2|C_1C_2|\frac{\sin\theta}{\sin(\theta+\phi)} = \frac{2|C_0C_1|\sin\theta}{\cos\theta\sin(\theta+\phi)} \end{aligned} \tag{4.17}$$

So we can reform the construction of our curve as following: given any starting point C_0 and the initial direction of the first segment C_0C_1, length l of the first edge segment, two angles θ and

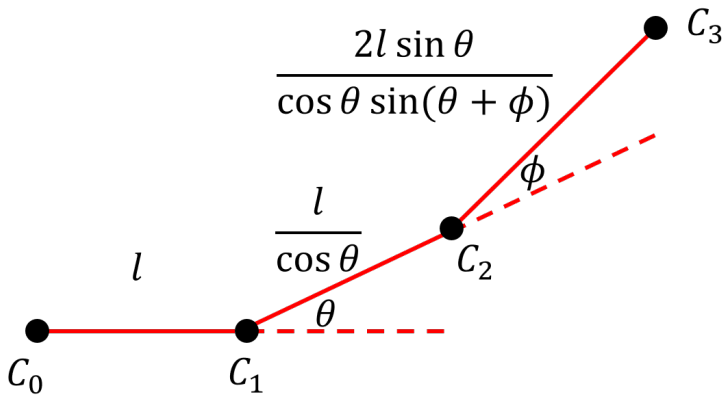

Figure 4.7: Our restricted cubic curve can be defined purely by the angles between three edges and the lengths of the edges according to the angles.

ϕ, we can compute the location of the rest control points. The direction and the length of the first edge can uniquely define the second control point C_1. We extend the ray C_0C_1 and turn that ray with angle θ, which is the direction of ray C_1C_2. On the ray C_1C_2, we find the location of C_2 so that

$$|C_1C_2| = \frac{l}{\cos\theta}.$$

We turn the ray C_1C_2 with angle ϕ as the direction of ray C_2C_3. We compute the location of C_3 so that

$$|C_2C_3| = \frac{2l\sin\theta}{\cos\theta\sin(\theta+\phi)}.$$

Figure 4.7 shows the quantities we need to define the curve.

4.4 Hybrid Curves

Both quadratic curves and cubic curves have their own shortcomings. Figure 4.5 shows the example of a quadratic and a cubic curve in the same control system. If we place $C_0 = Q_0$ at $(0, 0)$, and put Q_1 at $(a, 0)$. Assume the location of $C_3 = Q_2$ is (x, y), then when $x < 2a$, the quadratic curve no longer has monotonic curvature. If $x > 4a$, the cubic curve is non-convex or not curvature continuous. So we can combine the construction of the quadratic and cubic curves such that the monotonic curvature region is larger than both of the curves. When x is smaller than

$2a$, we use the cubic control polygon, and when x is larger than $2a$, we use the quadratic curve. When x equals to $2a$, the quadratic and cubic curves collapse and are identical. So if we move the point (x, y) along the whole 2D plane, the curve will change continuously.

We restate our hybrid curve as following: given three control points $\{Q_0, Q_1, Q_2\}$, if $(Q_2 - Q_1) \cdot (Q_1 - Q_0) \geq (Q_1 - Q_0)^2$, we use the quadratic curve generated by $\{Q_0, Q_1, Q_2\}$. Or, use equation (4.14) to compute the cubic curve. This hybrid curve will be used as curve primitives in a later curve modeling chapter.

5. QUADRATIC INTERPOLATION: κ-CURVES

Beginning this chapter, we talk about how to use the curve primitive in chapter 4 to construct a whole curve with multiple segments satisfying all the geometric criteria in chapter 1.

We proposed a curve modeling tool called κ-Curves [4] using quadratic curve segments to generate an almost G^2 everywhere interpolatory curve. More specifically, κ-Curves are piecewise-quadratic curves that interpolate a list of control points and have local maxima of curvature only at the control points. The advantage is that salient features of the curve will occur only at control points to avoid the creation of features unintended by the artist.

5.1 Problem Formulation

We start from the geometric requirement of the output curve. Given an ordered set of points $\{P_0, P_1 \ldots, P_n, P_{n+1}\}$ on a 2D plane, we would like to construct a curvature continuous curve (G^2 curve) $r(t)$, $t \in D$ such that the curve goes through the input points $\{P_i\}$ in order; that is, there exists some parameters $\{t_i\}$ such that $r(t_i) = P_i$.

The curve can be open or closed. If we want an open curve, the curve $r(t)$ should be defined on some closed interval and goes from P_0 to P_{n+1}. There are n segments between the input points. For a closed curve, to make the discussion consistent, we still want n curve segments. We take the input points $\{P_1, P_2 \ldots, P_n\}$ cyclic. So the curve should goes from P_1 to P_n and back to P_1. We add two virtual points that $P_0 = P_n$ and $P_{n+1} = P_1$. Figure 5.1 shows the subscripts of the points. Then no matter open or closed curves, the curve should have the same n segments and the same definition of the interpolated points. We allow P_0 and P_{n+1} to be at the same location for open curves, however, the curve at $P_0 = P_{n+1}$ can be considered as a G^0 point on a closed curve for this case, and $P_0 = P_{n+1}$ does not need to be a curvature continuous point.

Furthermore, we would like any local maximums of the curvature magnitude (the absolute value of curvature) to exist only at the P_i. This last criterion is based on the assumption that points at which the curve bends the most, at least locally, are salient features of the curve and should be

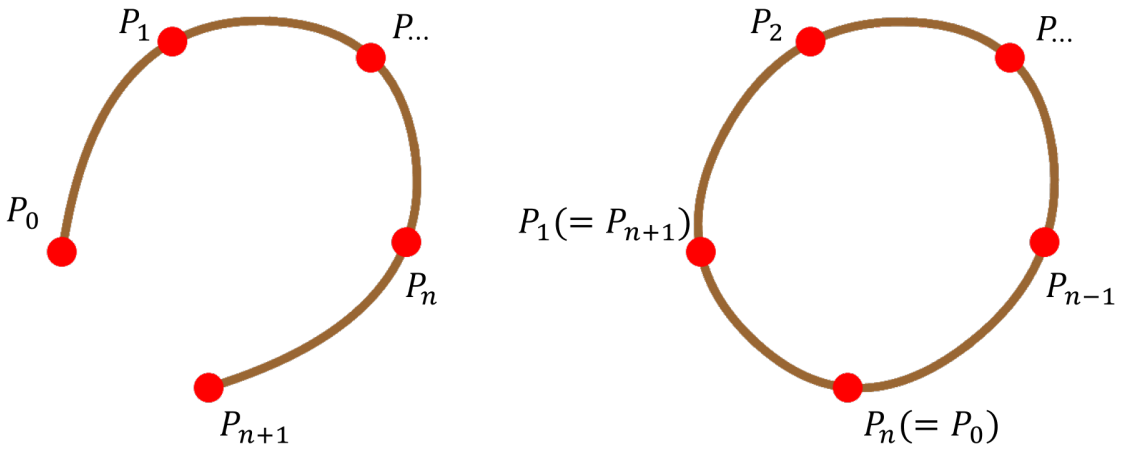

Figure 5.1: Notation and subscripts of the input points for open and closed curves.

under direct control of the user. Notice that this last property does not imply that the derivative of the curvature magnitude must be zero at every P_i. Hence, curvature may be locally increasing or decreasing at a interpolated point P_i. However, if a maximum of the curvature magnitude exists, it appears at an interpolated point.

5.2 Construction

The main idea of κ-Curves is to find a parabola segment including its critical point for each input. We align the critical point of the parabola at the input and rotate and adjust the shape of the parabolas so that all pieces of parabola can be connected G^1 and G^2 one by one.

To achieve those goals, we need to analysis the geometric properties of parabolas and prove the existence and uniqueness of such curve. The intrinsic properties of quadratic curves have been well discussed in chapter 4. We will focus on the interpolatory properties in this section.

For each input point P_i, denote the desired parabola segment is

$$c_i(t) = (1-t)^2 Q_{i,0} + 2(1-t)tQ_{i,1} + t^2 Q_{i,2}, \quad t \in [0,1]. \tag{5.1}$$

Here $\{Q_{i,0}, Q_{i,1}, Q_{i,2}\}$ are the control points we need to find.

5.2.1 Interpolation

First we need to the curve $\{c_i\}$ to interpolate the input points $\{P_i\}$, so we need some parameters $t_i \in [0, 1]$ s.t.

$$c_i(t_i) = (1 - t_i)^2 Q_{i,0} + 2(1 - t_i)t_i Q_{i,1} + t_i^2 Q_{i,2} = P_i. \tag{5.2}$$

5.2.2 Curvature Extrema

Then the input point should be the local maximum curvature point. So the points $\{P_i\}$ need to be at least critical points on the curve. Notice the only critical point of a parabola is always a local maximum curvature (in absolute value) point. So here we need and only need the derivative of the curvature at input points to be zero. Recall the curvature of c_i from equation (4.2)

$$\kappa_i(t) = \frac{\Delta(Q_{i,0}, Q_{i,1}, Q_{i,2})}{||(1 - t)(Q_{i,1} - Q_{i,0}) + t(Q_{i,2} - Q_{i,1})||^3}$$

We need $\kappa_i'(t_i) = 0$. From equation (4.5) we know the root of the derivative of the curvature

$$t_i = \frac{(Q_{i,0} - Q_{i,1}).(Q_{i,0} - 2Q_{i,1} + Q_{i,2})}{||Q_{i,0} - 2Q_{i,1} + Q_{i,2}||^2}. \tag{5.3}$$

5.2.3 Continuity

We aim to piece these curves together to form a curvature continuous (i.e; G^2 curve). Note that it is not possible to create a G^2 everywhere curve using piecewise quadratics if the sign of curvature changes along the curve. The reason is that quadratic curves cannot possess zero curvature unless the curve is trivially a straight line. Hence, unless our curves are strictly convex, we cannot hope to build piecewise quadratic curves that are strictly G^2 everywhere.

Our compromise is to build piecewise quadratic curves that are G^2 *almost* everywhere. The only place where our curves will lose continuity of curvature will be at points where the curve changes from convex to concave or vice versa. Hence, unlike standard spline constructions, the geometric smoothness of our piecewise construction changes dependent on the *geometry* of the

curve instead of how the curve is decomposed into polynomial pieces. Conditions for joining quadratic curves with G^2 smoothness have been discussed before [27]. However, given that it is not well known that building curvature continuous quadratic curves is even possible, we derive the conditions here and provide a closed-form solution, which will then be part of our optimization in section 5.4.

Our curves consist of multiple quadratic curves, each with control points $Q_{i,0}, Q_{i,1}, Q_{i,2}$, per interpolated point P_i. The G^0 continuity conditions between curves are trivial and simply require that

$$Q_{i,2} = Q_{i+1,0}, \quad i = \{1, \cdots, n\}.$$

G^1 continuity is simple as well and requires that

$$Q_{i,2} = Q_{i+1,0} = (1 - \lambda_i)Q_{i,1} + \lambda_i Q_{i+1,1}, \quad i = \{1, \cdots, n\} \tag{5.4}$$

for some $\lambda_i \in (0, 1)$, together with the G^0 continuity above.

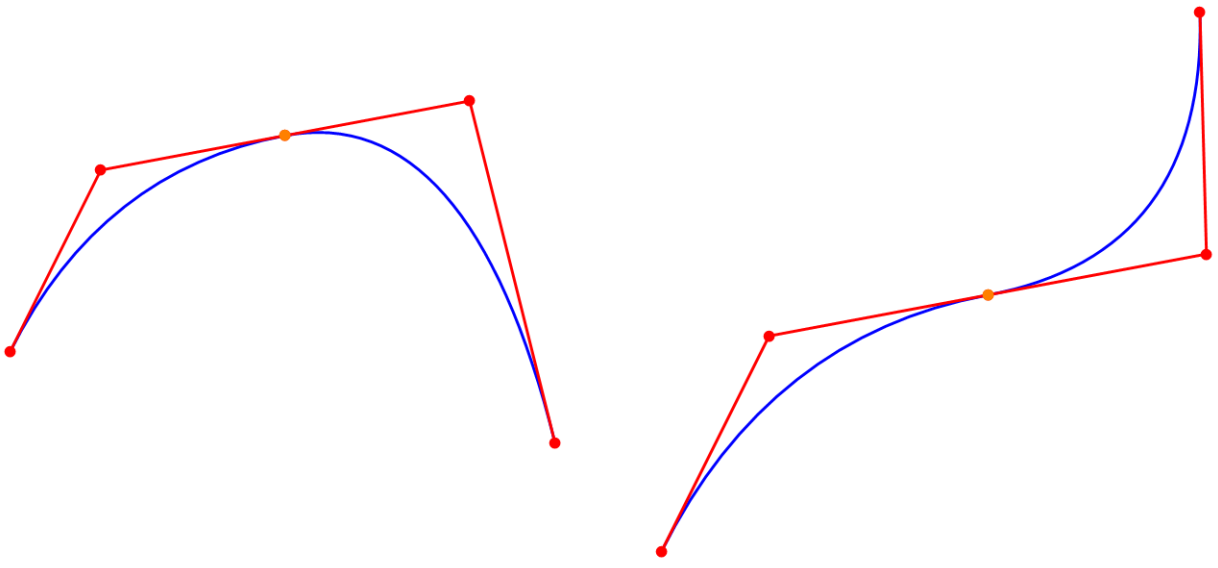

Figure 5.2: Two quadratic curve segments can be connected in a convex or a concave way satisfying G^1 condition.

For G^2 continuity, we need curvature on the sides of a connecting point to be the same. Figure 5.2 shows the two ways connecting two quadratic curve using a G^1 join point. On the left case, the two quadratic curves form a convex shape and it is possible for the join point to be curvature continuous. However if the two curve segments form a non-convex shape, it is impossible for the join to be curvature continuous, because the curvature of two segments have different signs but a quadratic curve can not have a zero curvature point unless the whole curve is a straight line. So in the right case, we weaken the G^2 condition so that the absolute value of curvature is the same at the join point. Under this assumption, the criteria will change to match the absolute value of the curvature. Using Equation 4.4 and we have the absolute G^2 condition at join points as

$$\frac{|\Delta(Q_{i,0}, Q_{i,1}, Q_{i,2})|}{|Q_{i,1} - Q_{i,2}|^3} = \frac{|\Delta(Q_{i+1,0}, Q_{i+1,1}, Q_{i+1,2})|}{|Q_{i+1,0} - Q_{i+1,1}|^3}. \tag{5.5}$$

5.3 Local Optimization

The problem is defined by equation (5.2), (5.3), (5.4) and (5.5) for $n = 1, \cdots, n$. There are in total $8n$ equation because equation (5.2) is an equation for x and y coordinates, and equation (5.4) is two equations about 2D coordinates. The variables are $\{Q_{i,0}, Q_{i,1}, Qi, 2\}$ for $6n$ degrees of freedom and t_i and λ_i for n degrees of freedom each. However this is a cubic system about the variables. Though the degrees of freedom of the variables match the number of the equation, it is hard to solve the system directly.

Here we substitute some of the variables to reduce the complexity of the system. Notice using equation (5.4), we can represent the end points $\{Q_{i,0}, Q_{i,2}\}$ as a function of $\{\lambda_i\}$ and off-curve control points $\{Q_{i,1}\}$, which means we can substitute equation 5.4 into equation (5.2), (5.3) and (5.5). Then we have $4n$ equations with $4n$ variables $\{Q_{i,1}, \lambda_i, t_i\}$, where each 2D point is considered as two variables.

Here we perform a local optimization to compute the variables to satisfy equations (5.2), (5.3) and (5.5) in an iterative way.

5.3.1 Maximum Curvature Condition

We rewrite equation (5.2) to represent the off-curve control point $Q_{i,1}$ as linear combination of the input point P_i and two end points $\{Q_{i,0}, Q_{i,2}\}$

$$Q_{i,1} = \frac{P_i - (1-t_i)^2 Q_{i,0} - t_i^2 Q_{i,2}}{2t_i(1-t_i)}. \tag{5.6}$$

Substituting the equation above into equation (5.3) results in a cubic function about parameter t_i:

$$||Q_{i,2}-Q_{i,0}||^2 t_i^3 + 3(Q_{i,2}-Q_{i,0})\cdot(Q_{i,0}-P_i)t_i^2 + (3Q_{i,0}-2P_i-Q_{i,2})\cdot(Q_{i,0}-P_i)t_i - ||Q_{i,0}-P_i||^2 = 0. \tag{5.7}$$

According to equation (5.4), the coefficients of equation (5.7) only depend on $\{Q_{i,1}\}$ and $\{\lambda_i\}$. So the algebraic meaning of equation (5.7) is that we fix $\{Q_{i,1}\}$ and $\{\lambda_i\}$ and solve for $\{t_i\}$ in the system of equations (5.2), (5.3) and (5.5).

While this equation could have three real roots, which would mean the solution t_i is not unique, luckily we have proved that this equation has exactly one real root in $[0, 1]$ for any choice of $Q_{i,0}, P_i, Q_{i,2}$.

Theorem 3. *For any given distinct points $Q_{i,0}, P_i, Q_{i,2}$, equation* (5.7) *has a unique root t_i in the closed interval* $[0, 1]$.

Proof. To show Equation 5.7 has exactly one real root in $[0, 1]$, we first write its coefficients in Bézier form, which leads to an extremely simple expression. Denote the vectors $v_0 = Q_{i,0} - P_i$ and $v_2 = Q_{i,2} - P_i$. Then the Bézier coefficients are

$$(-|v_0|^2, \frac{-v_0 \cdot v_2}{3}, \frac{v_0 \cdot v_2}{3}, |v_2|^2)$$

Given that we are interested in roots of this polynomial, we divide by the positive constant $|v_0||v_2|$

to obtain an even simpler expression for the coefficients

$$(-r, -\frac{\cos(\theta)}{3}, \frac{\cos(\theta)}{3}, \frac{1}{r}) \tag{5.8}$$

where $r = \frac{|v_0|}{|v_2|}$ and θ is the angle between the vectors v_0 and v_2. Note that polynomials in Bézier form follow Descartes' rule of signs for bounding the number of roots of a polynomial [42]. Since the first coefficient is negative while the last coefficient is positive, there exists at least one root on the interval $[0, 1]$. If $\cos(\theta) \geq 0$, then Descartes' rule of signs indicates exactly one root in $[0, 1]$. Therefore, we consider the case where $\cos(\theta) < 0$.

However, under this assumption, the polynomial with Bézier coefficients from Equation 5.8 is monotonically increasing over the interval $[0, 1]$. We verify this fact by computing the derivative of the polynomial, which yields the quadratic Bézier coefficients

$$(3r - \cos(\theta), 2\cos(\theta), \frac{3}{r} - \cos(\theta)).$$

Both the first and the last coefficient are positive since $\cos(\theta) < 0$ and $r > 0$. Furthermore, we can compute the minimum value of this quadratic, which is

$$\frac{3r - \cos(\theta)(1 + r^2 + r\cos(\theta))}{1 + r^2 - 2r\cos(\theta)}.$$

This value is also strictly positive because $1 + r^2 + r\cos(\theta) > 0$. Since the derivative of the quadratic at 0 is $6(\cos(\theta) - r)$, which is negative, the cubic function with Bézier coefficients in equation 5.8 is monotonically increasing and has exactly one real root in $[0, 1]$. □

So equation (5.7) can be written in an explicit function about other parameters.

$$t_i = t_i(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1}, \lambda_{i-1}, \lambda_i) \in [0, 1] \tag{5.9}$$

Given the existence of the unique root of this cubic, finding the root is simple, and there are

many ways to do so. A simple numerical way is to use bi-search on $[0, 1]$. A faster way is Newton's method using the derivative of the cubic. Here we use the explicit expression for roots of a cubic from Cardano's formula in our implementation (Appendix A). Even in the degenerate case where all three points form a straight line (i.e; $P_i = (1 - \alpha)Q_{i,0} + \alpha Q_{i,2}$), the cubic trivially has one root of $t_i = \alpha$. Once we have t_i, substituting this value into Equation 5.2 reveals the quadratic curve that interpolates p_i at the point of local maximum curvature magnitude.

The geometric meaning of the unique root in the theorem is

Corollary 3.1. *Given three ordered points on a 2D plane, there is a unique parabola going through the three points in the same order and the middle point is the critical point, or in the same meaning, the "vertex" of a parabola.*

This corollary is useful in geometric modeling. Not only does this corollary solves a local optimization problem, it gives us a method to interpolate any three points with strong curvature distribution property: the middle always the local maximum curvature and the curvature is decreasing along the two sides to the end points. This fact can be used in any curve modeling tool which requires "salient" points controlled. Indeed this idea has been used in a blending curve tool [43].

5.3.2 Curvature Condition at Join Points

The basic problem of the G^2 smoothness at join points is if it is always possible to connect two quadratic curve segments in a curvature continuous way. Here we explain the unique location of the join point in both of a geometric and an algebraic way. We only illustrate the convex connection way here in figure 5.3 since the non-convex case can be achieved by computing the absolute value of the curvature.

To satisfy the G^1 condition (5.4), the join point $Q_{i,2} = Q_{i+1,0}$ must be on the line segment $Q_{i,1}Q_{i+1,1}$. So when the ratio λ_i goes from 0 to 1, the join point will slide from $Q_{i,1}$ to $Q_{i+1,1}$. The curvature on the two sides of the join point will increase or decrease continuously as shown in figure 5.4. So there is a unique location of the join point that the curvature on the two sides meet.

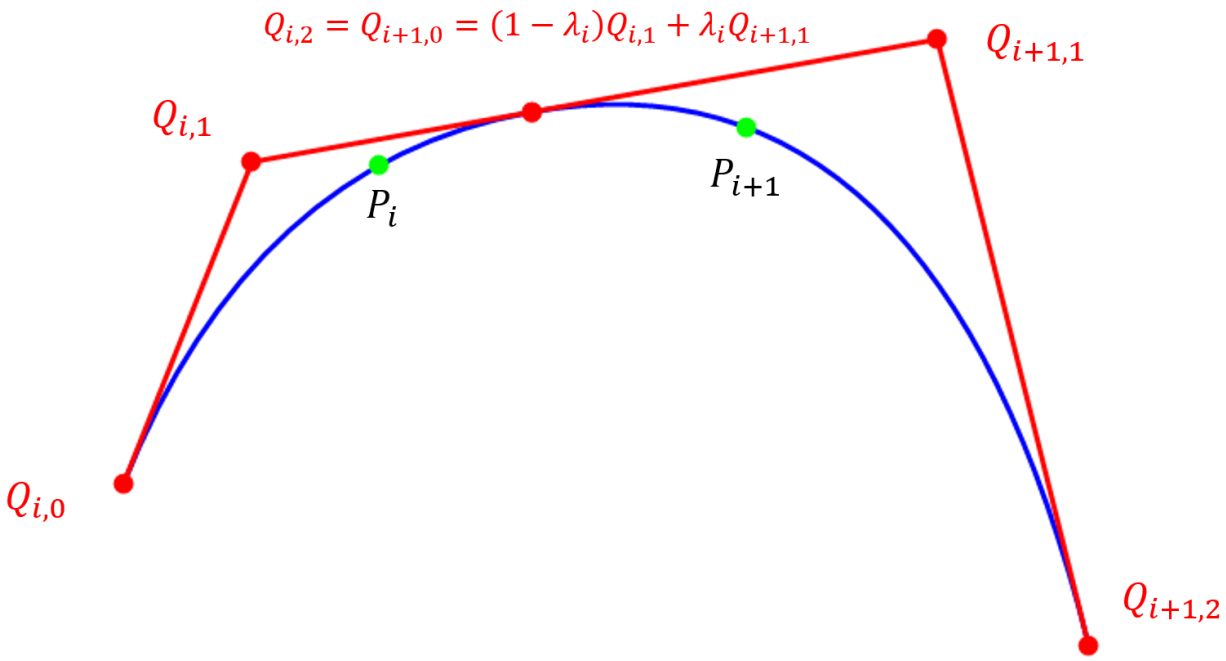

Figure 5.3: Connecting two polynomial quadratic curve segments.

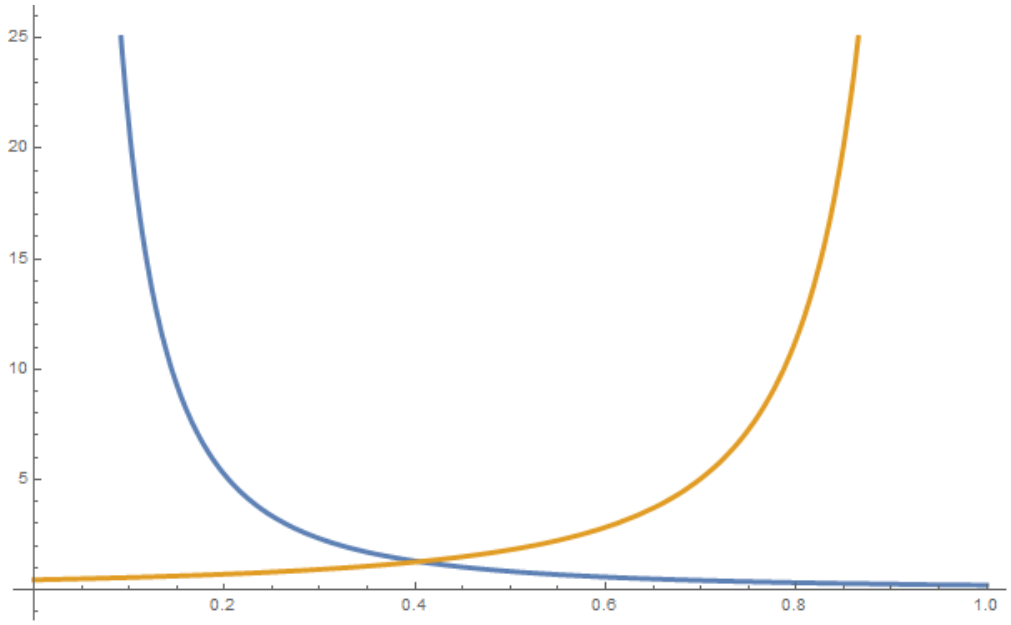

Figure 5.4: Plot of the curvature on the two sides of the join point when moving from one end to the other end on the line segment.

Using Equation 4.4 and writing the G^2 condition in terms of the off-curve points and λ_i yields

$$\kappa_i(1) = \frac{\Delta(Q_{i,0}, Q_{i,1}, Q_{i,2})}{|Q_{i,1} - Q_{i,2}|^3} = \frac{\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})(1-\lambda_{i-1})}{|Q_{i,1} - Q_{i+1,1}|^3 \lambda_i^2} \tag{5.10}$$

$$\kappa_{i+1}(0) = \frac{\Delta(Q_{i+1,0}, Q_{i+1,1}, Q_{i+1,2})}{|Q_{i+1,0} - Q_{i+1,1}|^3} = \frac{\Delta(Q_{i,1}, Q_{i+1,1}, Q_{i+2,1})\lambda_{i+1}}{|Q_{i,1} - Q_{i+1,1}|^3 (1-\lambda_i)^2} \tag{5.11}$$

The G^2 condition of the absolute value of the curvature

$$|\kappa_i(1)| = |\kappa_{i+1}(0)| \tag{5.12}$$

creates a quadratic equation in terms of λ_i that has exactly one root in $(0, 1)$, which is

$$\lambda_i = \frac{\sqrt{|\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})|(1-\lambda_{i-1})}}{\sqrt{|\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})|(1-\lambda_{i-1})} + \sqrt{|\Delta(Q_{i,1}, Q_{i+1,1}, Q_{i+2,1})|\lambda_{i+1}}}. \tag{5.13}$$

The above equations for for λ_i will be undefined if both triangle areas in the denominators are zero, which can happen when enough of the $\{Q_{i,1}\}$ are co-linear or coincident. This case can be robustly handled by adding a small constant, $\epsilon = 10^{-10}$, to the square roots of each such area.

The expression (5.13) of λ_i is about off-curve points $\{Q_{i,1}\}$ and two parameters λ_{i-1} and λ_{i+1}. So the meaning of equation (5.13) is we are fixing all $\{Q_{i,1}\}$, all $\{t_i\}$ to solve the parameter λ_i. Because we are also fixing λ_{i-1} and λ_{i+1}, this is not a standard numerical optimization to solve all λ_i. However in our experiments, this quasi optimization about the parameter λ_i works pretty well.

5.3.3 Input Point Interpolation

With given parameter λ_i and t_i, the only free variables are the off curve points $\{Q_{i,1}\}$. Any set of $\{Q_{i,1}\}$ will result in a piece-wise quadratic curve. We need each segment interpolate the input point P_i at Bézier parameter t_i, i.e. solving equation (5.2).

Equation (5.2) is in terms of the off-curve point and join points. So we substitute equation (5.4) into (5.2) so that we only have the variables of $\{Q_{i,1}\}$, $\{\lambda_i\}$, and $\{t_i\}$

$$P_i = (1-\lambda_{i-1})(1-t_i)^2 Q_{i-1,1} + (\lambda_{i-1}(1-t_i)^2 + 2(1-t_i)t_i + (1-\lambda_i)t_i^2)Q_{i,1} + \lambda_i t_i^2 Q_{i+1,1} \tag{5.14}$$

If we cluster all the n interpolation equations above, we get a linear system about off-curve points $\{Q_{i,1}\}$. And the coefficient matrix is a cyclic tri-diagonal matrix for a closed curve. For open curve, the two end points $Q_{1,0}$ and $Q_{n,2}$ should be fixed to the input points P_0 and P_{n+1}, so the coefficients should be exactly a tri-diagoal matrix. We explain the details of the linear system in the two followings sections:

Closed Curve. To make the discussion simpler, we just expand the equations in (5.14) to all $n = 1, 2, \cdots, n$:

$$\begin{cases} P_1 &= (1-\lambda_n)(1-t_1)^2 Q_{n,1} + (\lambda_n(1-t_1)^2 + 2(1-t_1)t_1 + (1-\lambda_1)t_1^2)Q_{1,1} + \lambda_1 t_1^2 Q_{2,1} \\ P_2 &= (1-\lambda_1)(1-t_2)^2 Q_{1,1} + (\lambda_1(1-t_2)^2 + 2(1-t_2)t_2 + (1-\lambda_2)t_2^2)Q_{2,1} + \lambda_2 t_2^2 Q_{3,1} \\ &\vdots \\ P_n &= (1-\lambda_{n-1})(1-t_n)^2 Q_{n-1,1} + (\lambda_{n-1}(1-t_n)^2 + 2(1-t_n)t_n + (1-\lambda_n)t_n^2)Q_{n,1} + \lambda_n t_n^2 Q_{1,1} \end{cases}$$

which is a linear system

$$\begin{pmatrix} * & * & \cdots & \cdots & * \\ * & * & * & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ * & \cdots & \cdots & * & * \end{pmatrix} \cdot \begin{pmatrix} Q_{1,1} \\ Q_{2,1} \\ \vdots \\ Q_{n,1} \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{pmatrix} \tag{5.15}$$

Open Curve. An open curve is slightly different from a closed curve. With input points $\{P_0, P_1, \cdots, P_{n+1}\}$. We are trying to construct n curve segments interpolating $\{P_1, \cdots, P_n\}$ as local maximum curvature points. The first and last segment should have control points $\{P_0, Q_{1,1}, Q_{1,2}\}$ and $\{Q_{n,0}, Q_{n,1}, P_{n+1}\}$. The according interpolation equations (5.14) turn to

$$\begin{aligned} P_1 &= (1-t_1)^2 P_0 + (2(1-t_1)t_1 + (1-\lambda_1)t_1^2)Q_{1,1} + \lambda_1 t_1^2 Q_{2,1} \\ P_n &= (1-\lambda_{n-1})(1-t_n)^2 Q_{n-1,1} + (\lambda_{n-1}(1-t_n)^2 + 2(1-t_n)t_n)Q_{n,1} + t_n^2 P_{n+1} \end{aligned}$$

Equations (5.14) remain the same. So the equations for an open curve turns to a tri-diagonal system

$$\begin{pmatrix} * & * & \cdots & \cdots & 0 \\ * & * & * & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & * & * \end{pmatrix} \cdot \begin{pmatrix} Q_{1,1} \\ Q_{2,1} \\ \vdots \\ Q_{n,1} \end{pmatrix} = \begin{pmatrix} P_1 - (1-t_1)^2 P_0 \\ P_2 \\ \vdots \\ P_n - t_n^2 P_{n+1} \end{pmatrix} \tag{5.16}$$

5.4 Global Optimization

We have established an equation system (5.2), (5.3) and (5.5) with variable $\{Q_{i,1}\}$, $\{t_i\}$ and λ_i. Based on the local solutions in section 5.3, we adopt a global approach to meet all the three classes of equations one by one. In each global iteration, we first estimate the time parameters $\{t_i\}$ by equation (5.9) to satisfy (5.3), then solve the join point parameters $\{\lambda_i\}$ by equation (5.13) to satisfy the G^2 condition (5.5), and then recompute the location of all corner points $\{Q_{i,1}\}$ by the linear systems in section 5.3.3. After one iteration of updating all variables, we update all the join points $\{Q_{i-1,2} = Q_{i,0}\}$ using the parameter $\{\lambda_i\}$. Notice satisfying one conditions of (5.2), (5.3) and (5.5) may break the other two. So we need to iterate this procedure again and again until converged. At the converged status, all geometric conditions are met and the variables remain at the same location.

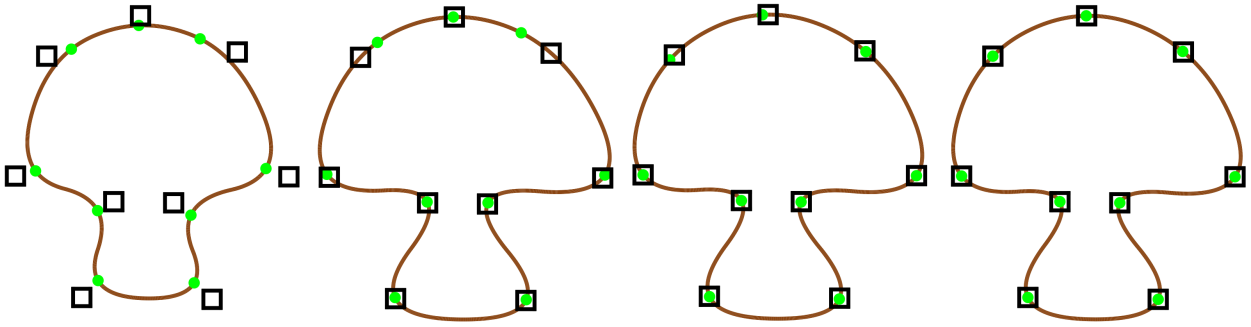

Figure 5.5: Iterations of our optimization showing convergence with control points (black boxes) and maximum curvature positions (green dots). From left to right: our initial guess, after 1 iteration, after 2 iterations, and final convergence after 30 iterations.

This optimization converges quickly and each iteration is fast to compute since we only solve a small, sparse linear system of equations. Figure 5.5 shows the progress of our optimization. After just one iteration, our result is very close to the final solution, but some maximum curvature points do not coincide with control points yet. After two iterations, the result is nearly indistinguishable from our final result. Even for large curves with many control points, our optimization yields results beyond interactive speeds due to its simplicity. Furthermore, it is possible to make the optimization even faster by starting with the results from the user's previous control point configuration as the user manipulates the curve, although we have found such improvements to be unnecessary.

Algorithm 1 κ-Curves

Input: $\{P_i\}$
$\{\lambda_i\} \leftarrow 0.5$
$\{t_i\} \leftarrow 0.5$
$\{Q_{i,1}\} \leftarrow P_i$
for $iterations \leftarrow 1$ to 60 **do**
    $\{t_i\} \leftarrow$ cubic root of equation (5.7)
    $\{\lambda_i\} \leftarrow$ solve from equation (5.13)
    $\{Q_{i,1}\} \leftarrow$ linearSolve equation(5.15) or (5.16)
$\{Q_{i,2} = Q_{i+1,1}\} \leftarrow \{(1-\lambda_i)Q_{i,1} + \lambda_i Q_{i+1,1}\}$
return Tuples $\{(Q_{i,0}, Q_{i,1}, Q_{i,2})\}$

Algorithm 1 shows the pseudo code of κ-Curves. This general pseudo code can be used for closed directly. For open curve, the only adjustment is on the two boundary points. Fixing $Q_{1,0} = P_0$ and $Q_{n,2} = P_{n+1}$, the rest will be the same as the closed curves.

5.5 Results

Our curves are designed to have maximal curvature magnitude at the control points P_i. Figure 5.6 shows the control points as hollow boxes and displays all local maxima of curvature magnitude as green points on the curve. Hence, green points should appear within each control point box

for our curve. These are the points where the curve, locally, bends the most. Despite its complex shape, no cusps or loops exist in the curve.

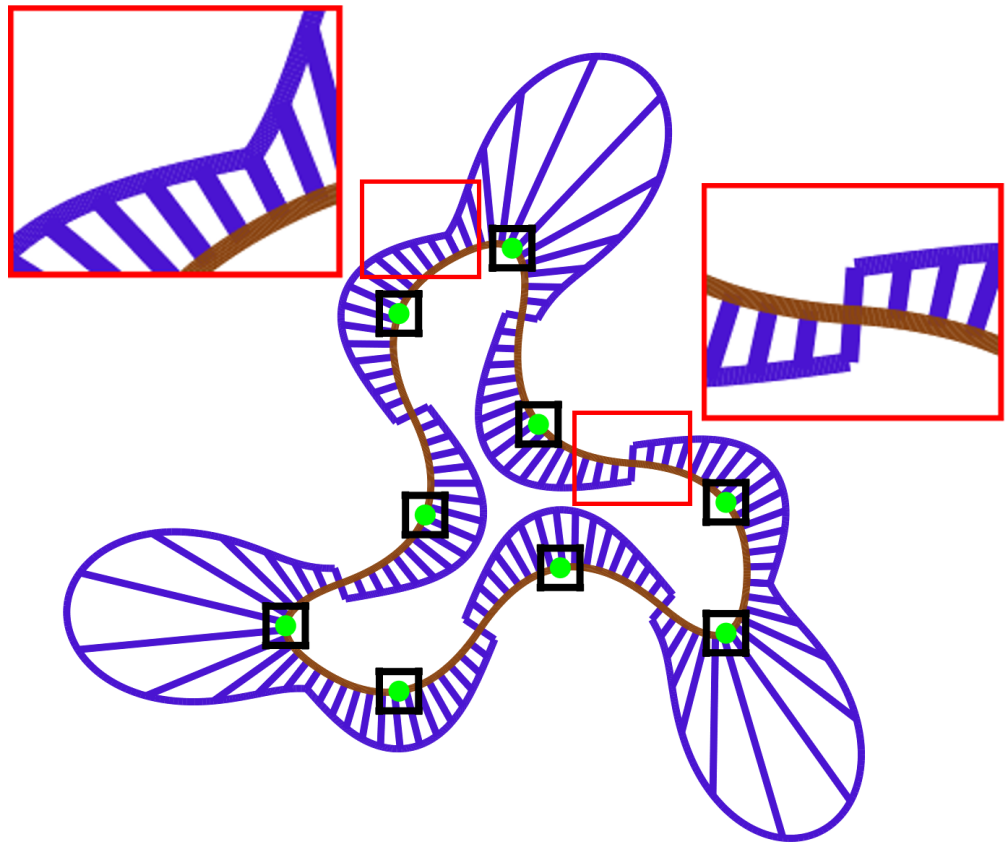

Figure 5.6: Our curve (brown) shown with control points as block boxes. Green points are positions of local maximal curvature magnitude. We also draw the curvature normal for the curve (purple). Our curve is G^2 everywhere as shown in the highlighted region except at inflection points where the curve is G^1 and the sign, but not magnitude, of the curvature changes.

Figure 5.6 also displays the continuity of the curve through the curvature normal (the normal whose length is proportional to curvature of the curve at that point). For a curvature continuous (G^2) curve, the magnitude of the curvature normal should change continuously over the curve. This is true for our curve everywhere except at the inflection points of the curve. At these points the curvature normal flips orientation but maintains the same magnitude.

Note that it is possible that a control point does not exist at a maximum curvature point as demonstrated in Figure 5.7. In this case, the curvature is decreasing at the highlighted point. However, all points of maximum curvature magnitude appear at control points.

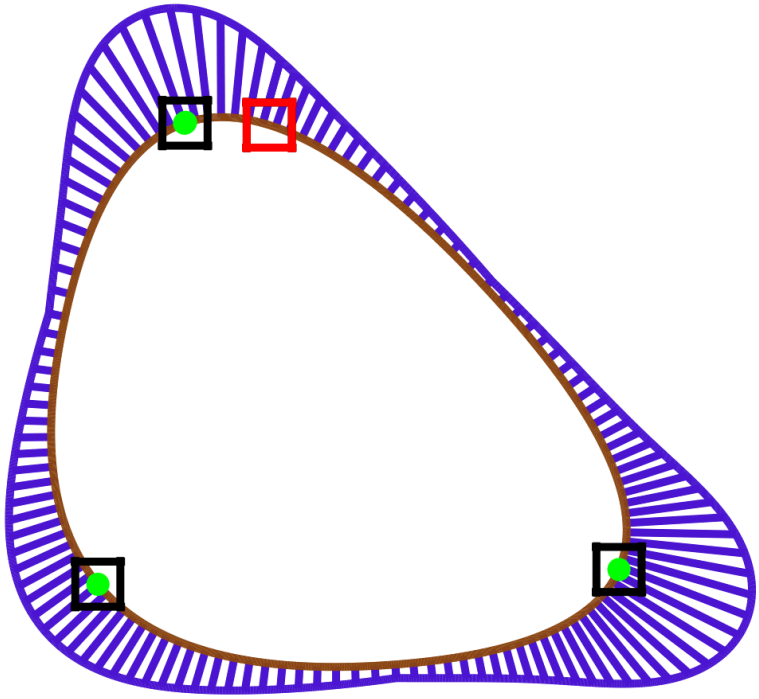

Figure 5.7: The red control point is not at a critical point of curvature, which is decreasing. However, all local maxima of curvature magnitude appear at control points.

Figure 5.8 depicts the creation of a cusp at a control point as the user manipulates the curve. Unlike curves such as clothoids, continuous motion of the control points results in continuous deformation of the curve. Clothoids use estimates of curvature that change continuously with motion of the control points. When moving from positive to negative curvature, the curve *must*

pass through a point of infinite positive curvature to a point of infinite negative curvature (a cusp) in order for the geometry of the curve to change continuously. Clothoids vary curvature piece wise linearly between control points and cannot possess such behavior. In contrast, our curves can generate infinite curvature, though only at control points. Having unbounded curvature is not a unwanted artifact but precisely the property that creates continuous motion of the curve. However, the cost of this continuous motion is a curvature profile that is not as "fair" as clothoids.

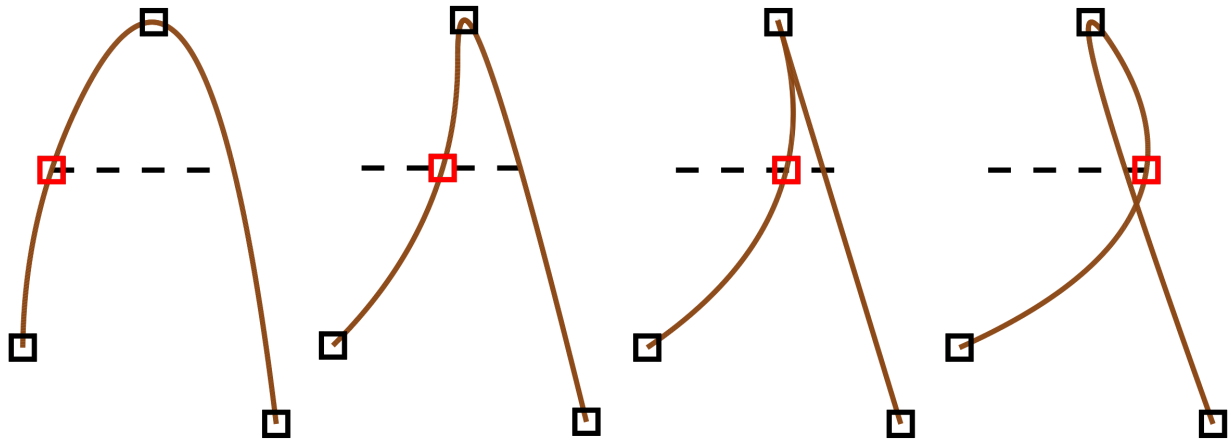

Figure 5.8: The creation of a cusp, which can only happen at control points with our method.

Since our optimization produces a global solution, the influence of one control point is technically global. That is, moving one point changes the shape of the entire curve. However, practically, the influence of one point is quite bounded. Figure 5.9 shows an example curve where we move a single control point and draw the original curve (blue) behind the new curve (brown). The figure illustrates that very little movement of the curve occurs outside of just a few control points away from the modified shape.

Figure 5.10 is an art piece designed by Daichi Ito in Adobe Illustrator. Figure 5.12, 5.13 and 5.14 are more examples in real art design with the input points.

Figure 5.12, 5.13 and 5.14 demonstrates shapes composed of multiple open and closed curves. And local maxima of curvature magnitude only appear at control points.

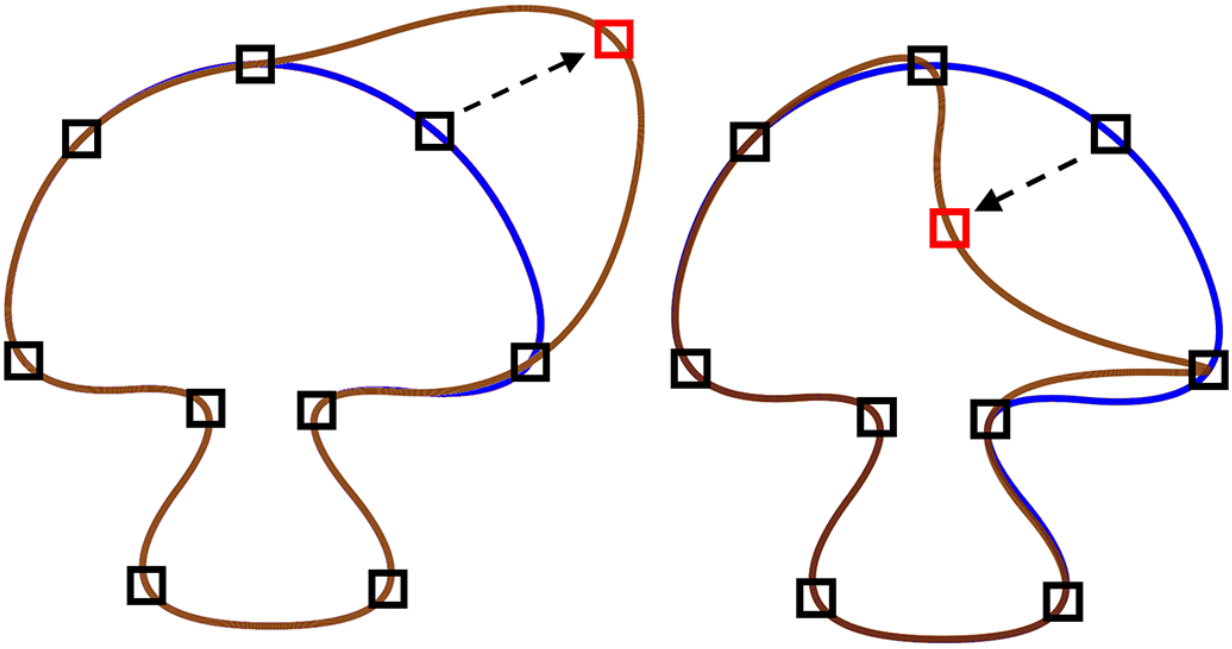

Figure 5.9: The original curve in blue with the new curve in brown drawn on top. While control points have a global influence on the curve, that influence drops dramatically with distance from the control point, which creates the effect of local influence.

5.6 Conclusion

Interpolatory curves typically suffer from shape artifacts and use high degree polynomials with large support for even low continuity curves. Our curves change this paradigm. We use low degree curves, yet achieve higher-order continuity. Despite their global nature, we have demonstrated that the influence of an input point is local for practical applications. Moreover, the input points coincide with features of the curve; namely, local maxima of curvature magnitude. Even though the user has no direct control over the tangent angle or curvature at the input points, the system automatically chooses natural values for these parameters. In fact one may see this work a way of automatically choosing natural and pleasing tangent angles and curvatures based only the input points. Finally, our curves change continuously under continuous motion of the control points. These two combined properties make them ideal for a number of creative tasks including vector design and motion path key framing. In summary, we believe that κ-Curves solve many of the problems that have plagued interpolatory curves, rendering them much more suitable for modern

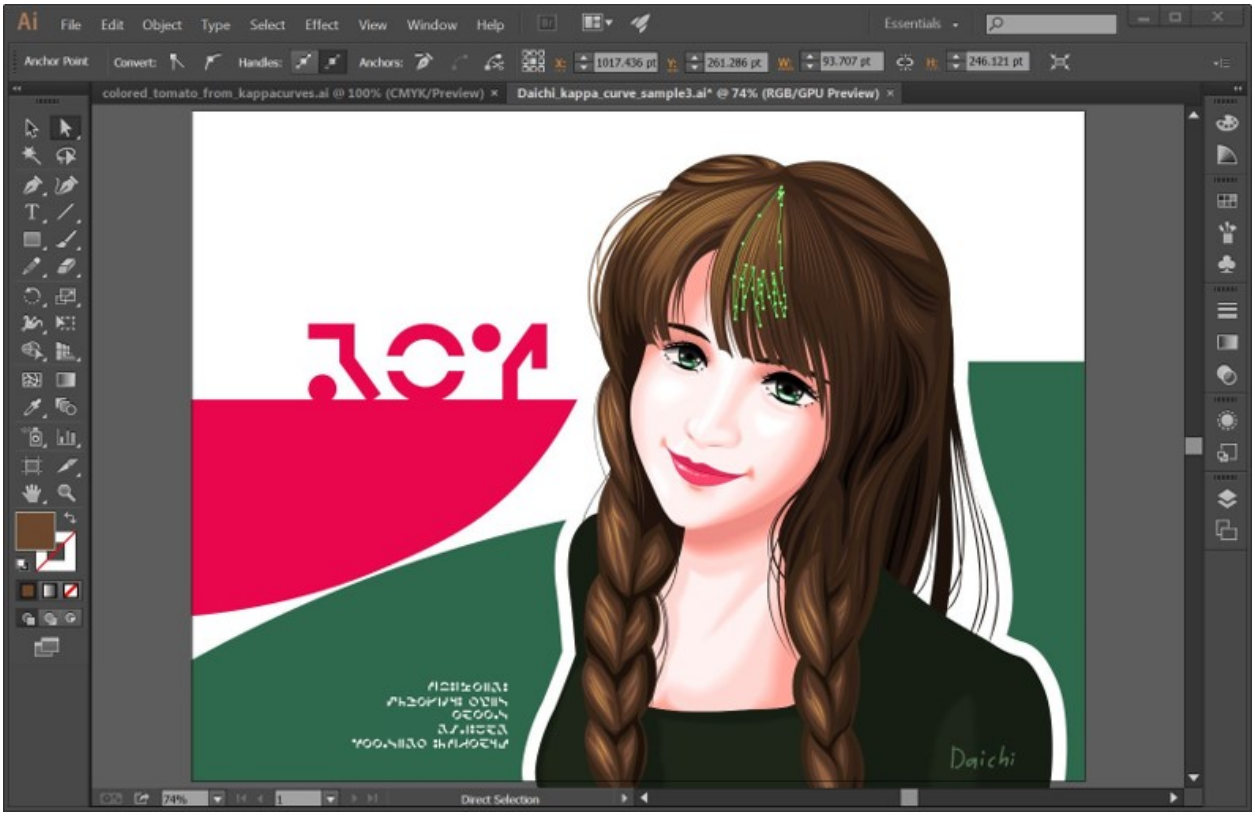

Figure 5.10: An art piece designed by Daichi Ito from Adobe Research using the "Curvature Tool" in Adobe Illustrator.

design applications.

The κ-curve system has been implemented as a tool in Adobe Illustrator and in Adobe Photoshop at a later time. The actual implementation in Adobe Illustrator differs slightly from what was presented in this paper due to compatibility and other issues, but the underlying technology is essentially the same. Adobe Illustrator is instrumented to report the amount of time users spend using each of the available tools. The new tool, called the curvature tool, is a direct competitor for the much older pen tool that uses cardinal splines. Even though there are experienced and exacting artists with many years invested in the the pen tool, six months after the release of the curvature tool 35% of the combined use of the two tools was with the newer curvature tool on desktop devices. On devices with touch screens, the curvature tool captured 66% of the combined use of the two tools. While more studies are needed to fully understand artists preferences, this usage data

strongly suggests that our model is a welcome addition to professional workflows. Figure 5.11 shows the usage of the curvature tool on both two types of devices. The original pictures can be found in the official guidance of "Curvature Tool" from Adobe's website.[1][2]

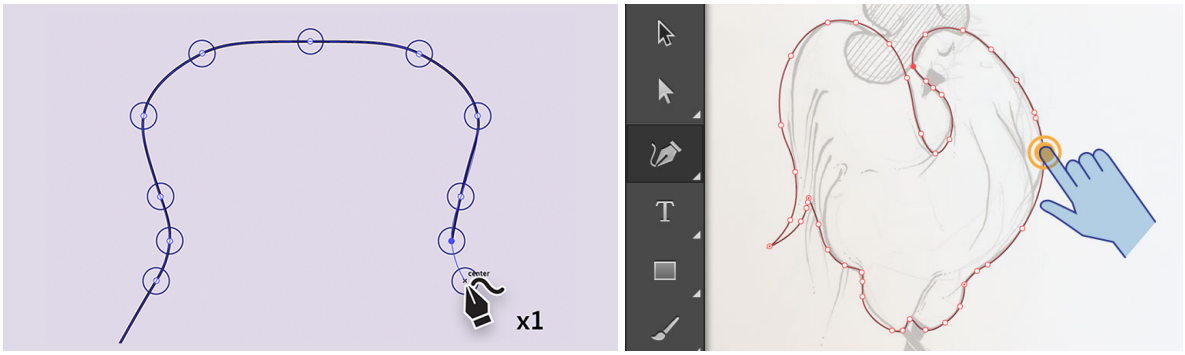

Figure 5.11: Adobe implemented a modified version of κ-Curves as the "Curvature Tool" in Adobe Illustrator. Theses figures show the way to use κ-Curves with a mouse and on a touch screen device.

[1] https://helpx.adobe.com/illustrator/how-to/draw-edit-curves.html
[2] https://helpx.adobe.com/illustrator/how-to/draw-touch-environment.html

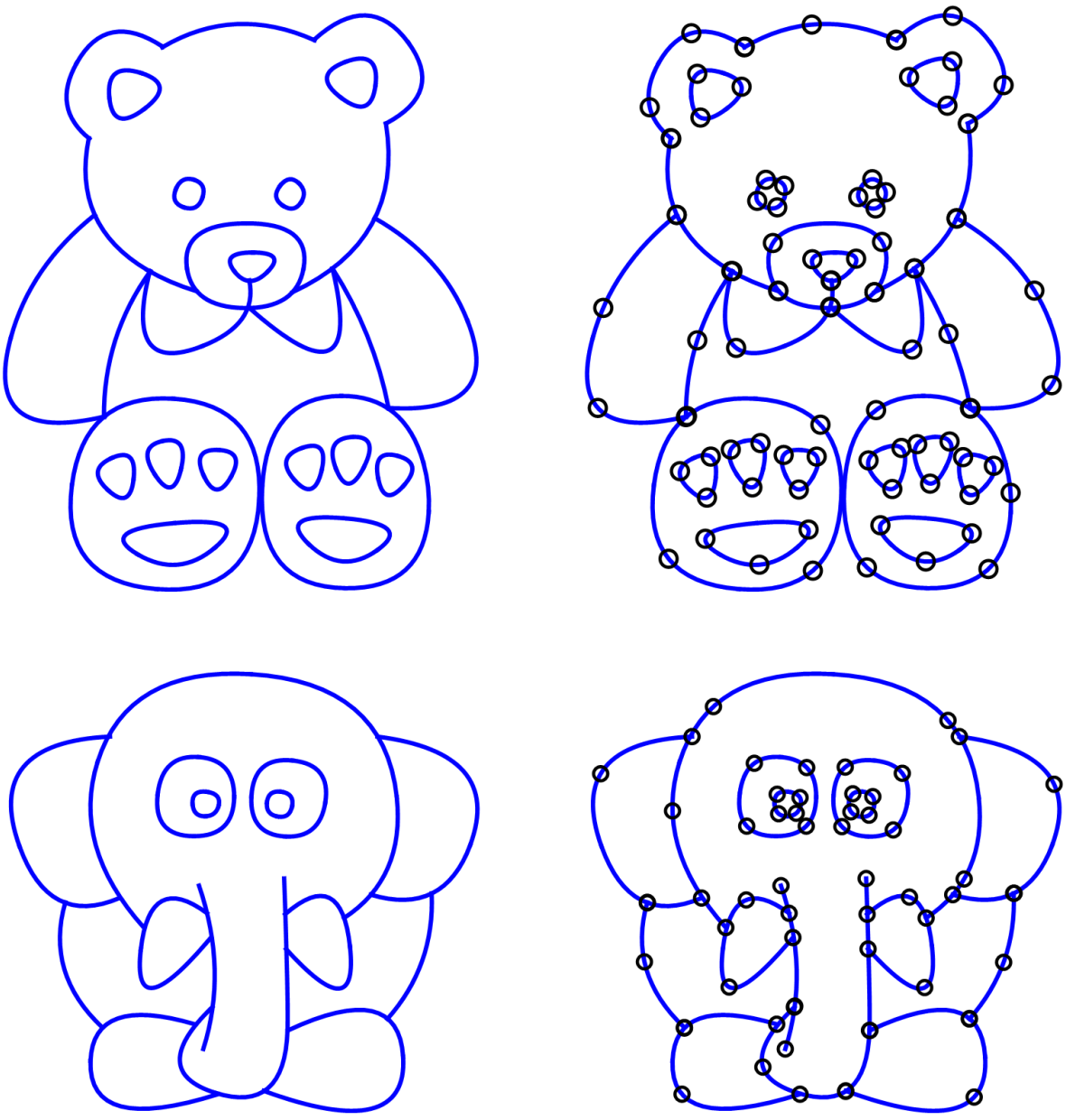

Figure 5.12: κ-Curve examples of a cartoon bear and elephant.

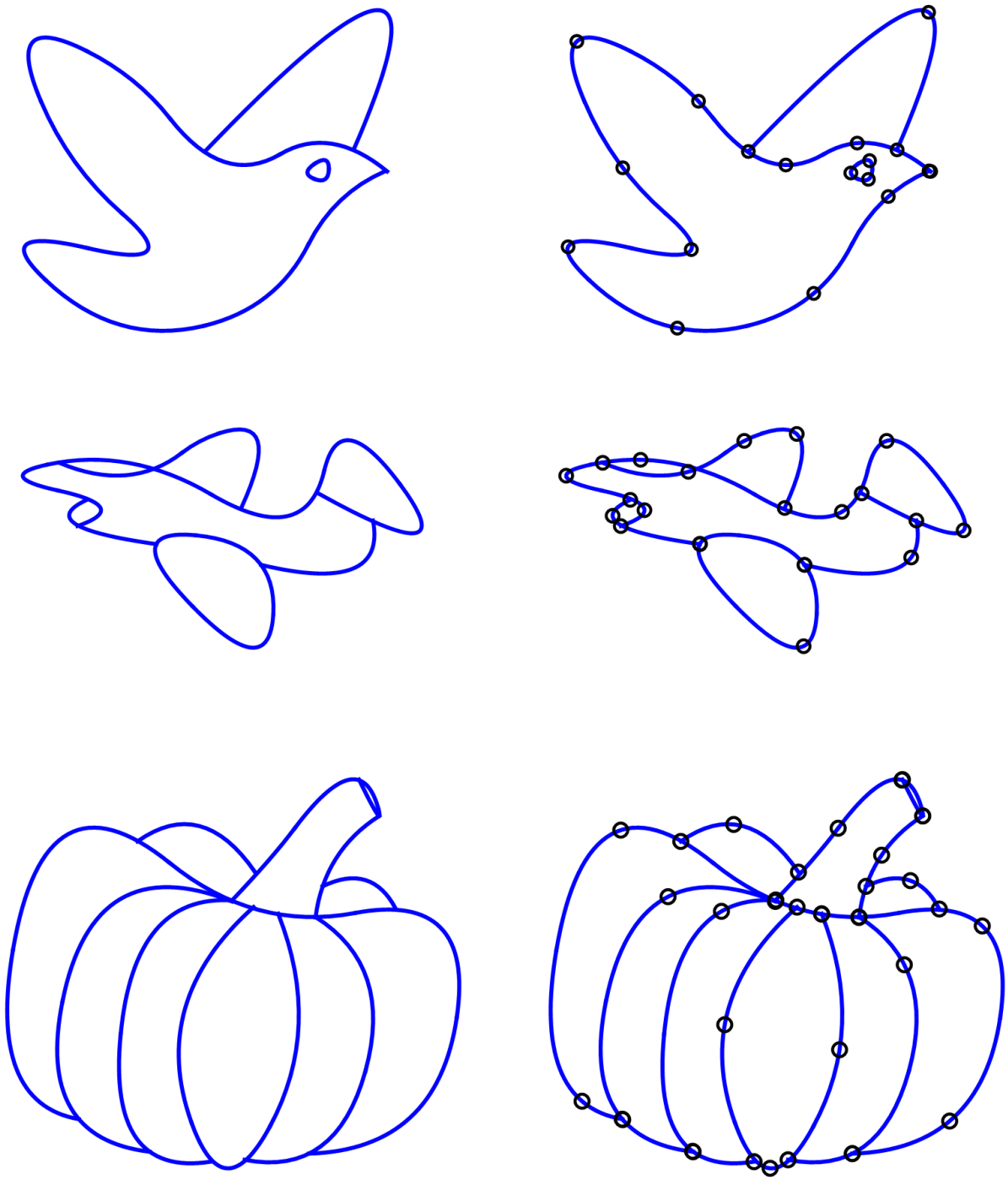

Figure 5.13: κ-Curve examples of a bird, a plane and pumpkin.

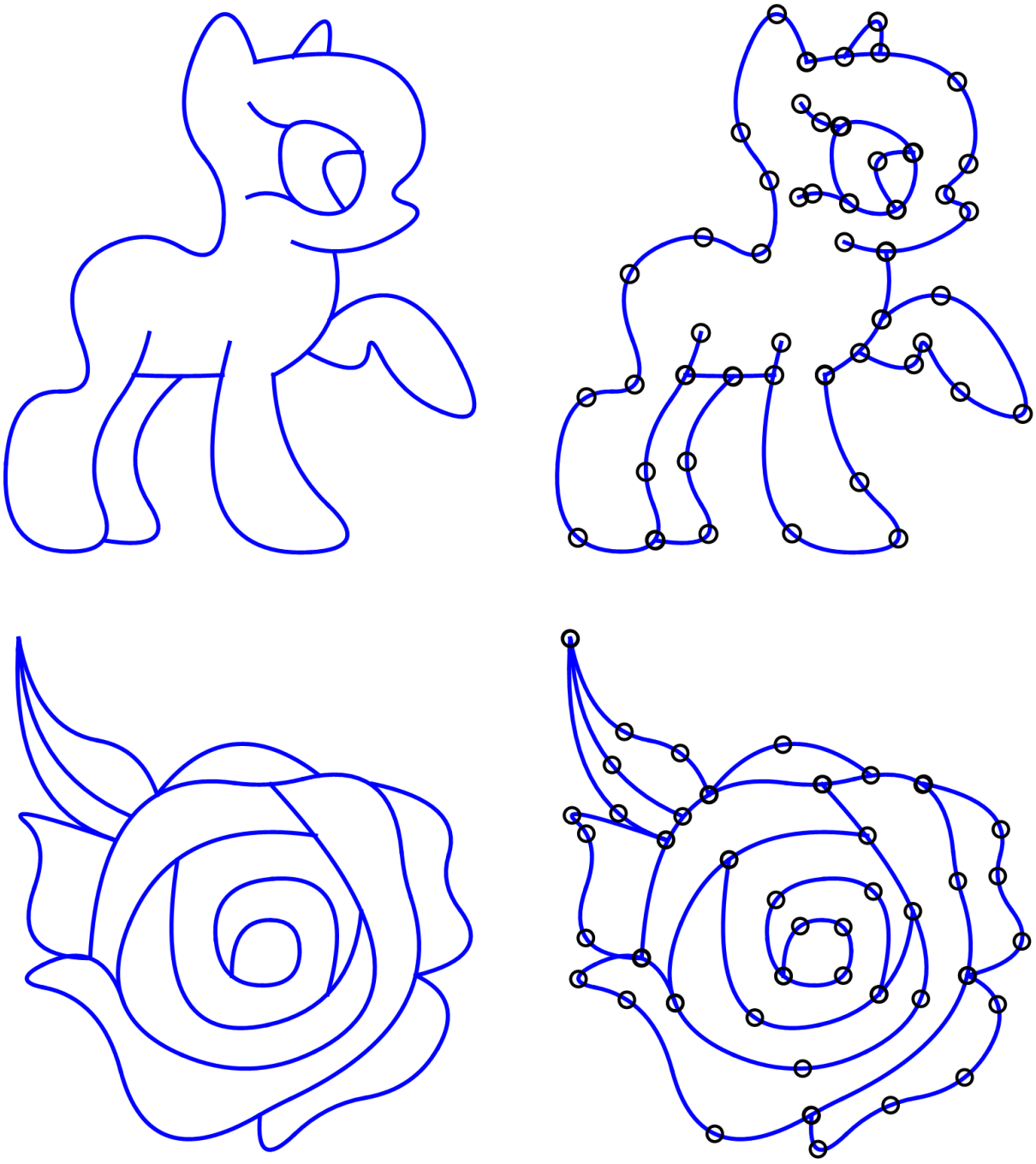

Figure 5.14: κ-Curve examples of a horse and a rose.

6. RATIONAL QUADRATIC INTERPOLATION

κ-Curves in chapter 5 showed that control points should be located at features of the curve. Moreover, such features should not occur in the curve outside of the control points.

Beyond simply controlling features of the curve, the types of curves a modeling tool can produce are also important. While polynomial curves are prevalent due to their ease of use, circles are important shapes due to their common occurrence in every day life whether part of images or cylindrical or spherical 3D shapes. Therefore, a useful curve representation would be able to represent shapes such as circles exactly. In particular, if the interpolated control points lie on a circle, even if they are not equally spaced, it would be reasonable to expect that a circle would be reproduced. When the control points do not lie on a circle, the curve should be "fair" where "fair" means the curvature plot should be simple [13].

In this chapter, we introduce a piece-wise rational curve that interpolates a series of input points on a 2D plane. All local maxima of curvature only occur at input points. For each input point, we compute a single rational quadratic curve that interpolates the input point somewhere in its domain. We use the remaining degrees of freedom in the curve to connect consecutive curves with curvature continuity everywhere except at inflections. At these inflection points, the magnitude of curvature will be continuous though the sign (positive/negative) is different. We build a box constrained optimization to solve for the final curve and use an iterative method to compute a close initial guess to accelerate the optimization. Finally, we show how to automatically compute the rational weights of the curve and incorporate this function into the optimization to ensure that circles are reproduced when the control points lie on a circle.

6.1 Problem formulation

We are trying to solve a similar problem from κ-Curves with more shape requirements. We would like to interpolate a series of input points such that the input points are local maximum curvature points. The geometric meaning is we are trying to control the "salient" features of a 2D

curve.

The general requirements of the interpolatory curves here is the same as κ-Curves. Given a series of input points on a 2D plane: n points $\{P_1, P_2, \cdots, P_n\}$ for closed curve, $n+2$ points $\{P_0, P_1, \cdots, P_n, P_{n+1}\}$ for open curves. We have explained the reason to use different numbers of the input points for open and closed curves in section 5.1. Then we would like to construct a curve interpolating all the input points as local maximum curvature points.

The additional feature of this chapter is that we require one more degree of freedom for each input point so that we let the user to adjust the local "sharpness" which we called "tension" control.

6.2 Construction

The main idea is to replace the polynomial quadratic curve segments in κ-Curves with rational quadratic curves. For each input point P_i, we try to compute some rational quadratic segment

$$c_i(t) = \frac{(1-t)^2 Q_{i,0} + 2(1-t)t w_i Q_{i,1} + t^2 Q_{i,2}}{(1-t)^2 + 2(1-t)t w_i + t^2}, \quad t \in [0,1],\ w_i > 0 \tag{6.1}$$

with three control points $\{Q_{i,0}, Q_{i,1}, Q_{i,2}\}$. And w_i is the weight for the "sharpness". When all $w_i = 1$, the curve is exactly the polynomial curve. However, if we adjust the value of w_i, the local shape will change and geometric criteria might be broken, So we slightly modify the numerical optimization of the curve.

6.2.1 Interpolation

For each curve segment $c_i(t)$, we need the input point P_i interpolated at some parameter $t_i \in [0,1]$

$$P_i = \frac{(1-t_i)^2 Q_{i,0} + 2(1-t_i)t_i w_i Q_{i,1} + t_i^2 Q_{i,2}}{(1-t_i)^2 + 2(1-t_i)t_i w_i + t_i^2}, \tag{6.2}$$

6.2.2 Curvature Extrema

The input point P_i should also be local maximum curvature point. So curve $c_i(t)$ should have zero derivative of curvature at parameter t_i from equation (4.10)

$$\begin{bmatrix} (1-t_i)^4 \\ 4(1-t_i)^3 t_i \\ 6(1-t_i)^2 t_i^2 \\ 4(1-t_i) t_i^3 \\ t_i^4 \end{bmatrix}^T \cdot \begin{bmatrix} -2w_i^3 |Q_{i,0} - Q_{i,1}|^2 + w_i (Q_{i,0} - Q_{i,1}) \cdot (Q_{i,0} - Q_{i,2}) \\ -w_i^2 |Q_{i,0} - Q_{i,1}|^2 + \frac{1}{4} |Q_{i,0} - Q_{i,2}|^2 \\ \frac{1}{2} w_i (Q_{i,2} - Q_{i,0}) \cdot (Q_{i,0} - 2Q_{i,1} + Q_{i,2}) \\ -\frac{1}{4} |Q_{i,0} - Q_{i,2}|^2 + w_i^2 |Q_{i,1} - Q_{i,2}|^2 \\ -w_i (Q_{i,0} - Q_{i,2}) \cdot (Q_{i,1} - Q_{i,2}) + 2w_i^3 |Q_{i,1} - Q_{i,2}|^2 \end{bmatrix}. \tag{6.3}$$

This is a quartic function about t_i, however we will prove there's a unique root of the function on interval [0, 1]. So we can rewrite the parameter t_i defined by the root of the function above as

$$t_i = t_i(Q_{i,0}, Q_{i,1}, Q_{i,2}, w_i; P_i) \tag{6.4}$$

where P_i is used as a constant.

6.2.3 Continuity

To create curvature continuity between consecutive curves $c_i(t)$ and $c_{i+1}(t)$, we must enforce G^0 and G^1 continuity in addition to curvature continuity. G^0 continuity is trivial and simply requires

$$Q_{i,2} = Q_{i+1,0}.$$

G^1 continuity requires that the tangents of both sides of a join point are in the same direction. With the G^0 condition we have the equation

$$Q_{i,2} = Q_{i+1,0} = (1 - \lambda_i) Q_{i,1} + \lambda_i Q_{i+1,1} \tag{6.5}$$

for some $\lambda_i \in (0,1)$. We utilize the G^0 and G^1 conditions as constraints and use $\{\lambda_i\}$ as new variables in the later optimization to eliminate the join point variables $\{Q_{i,0}, Q_{i,2}\}$.

G^2 condition is a little more complex but the basic idea is the same as κ-Curves. Curvature continuity requires the curvatures at the end-points of consecutive curves to be equal. Similar to chapter 5, requiring curvature continuity for rational quadratic curves at an inflection point is impossible since conic sections cannot possess zero curvature unless the entire curve degenerates to a line. Hence, we cannot require that curves are curvature continuous everywhere. Instead, like Chapter 5, we require that the absolute value of curvature is continuous $|k_i(1)| = |k_{i+1}(0)|$, meaning that the magnitude of curvature is continuous at inflection points but the sign of curvature will be discontinuous:

$$\frac{|\Delta(Q_{i,0}, Q_{i,1}, Q_{i,2})|}{w_i^2 |Q_{i,1} - Q_{i,2}|^3} = \frac{|\Delta(Q_{i+1,0}, Q_{i+1,1}, Q_{i+1,2})|}{w_{i+1}^2 |Q_{i+1,0} - Q_{i+1,1}|^3}. \tag{6.6}$$

6.3 Weights

The curve system has been defined by equation (6.2), (6.4), (6.5) and (6.6). The number of the variables meet the number of the equations. So the rational weight w_i can be considered as extra degrees of freedoms. w_i provides the users additional control over the shape of the curve and can be used to adjust the shape of the curve. However, for users who do not wish to manually adjust w_i, we can choose w_i automatically to optimize for certain geometric properties of the curve. Our original motivation is to choose w_i to produce a circle when the user places input points Q_i along a circle. Unlike other methods [37], we do not require that the vertices are evenly spaced in terms of arc length along the circle to reproduce this shape.

There are multiple choices of the weights and we consider 4 different possibilities below although other choices certainly exist.

6.3.1 Trivial Weights

The simplest weights are all one. Then the curve is exactly the same as κ-Curves and all curve segments are parabolas. Another trivial set of weights are all weights close to infinity, which causes

the curve to approach the control polygon. The weights can not be all zero since corner points will be ignored. If the weights are close to zero, the curve will have a curvature minima in the parameter range rather than a maxima.

6.3.2 Half Angle Weight

If the curve $c_i(t)$ is a circular arc, the control polygon must form an isosceles triangle i.e.

$$|Q_{i,0}Q_{i,1}| = |Q_{i,1}Q_{i,2}|.$$

And the weight of point $Q_{i,1}$ must be

$$w_i^{HA} = \frac{|Q_{i,0}Q_{i,2}|}{2|Q_{i,0}Q_{i,1}|} = \sin\left(\frac{\angle Q_{i,0}Q_{i,1}Q_{i,2}}{2}\right).$$

If we use this half-angle weight for all points, then when the two edges $Q_{i,0}Q_{i,1}$ and $Q_{i,1}Q_{i,2}$ are equal, the curve is always a circular arc. However this is also the disadvantage. When $Q_{i,0}$ and $Q_{i,2}$ are close enough and the off-curve point $Q_{i,1}$ are far, the desired shape is likely a cusp but the half angle weight will result in a short circular arc. Whats more, the half angle weight is always less or equal to 1.

6.3.3 Minimum Eccentricity Weights

All rational quadratic curves define a conic section as the solution to an implicit quadratic equation (4.7). We can easily transform our parametric quadratics into this implicit form using a resultant [44]. To create circles, we use the eccentricity of this conic, which measures the deviation of the shape from a circle [45]. For circles, the eccentricity is 0. Ellipses have an eccentricity between 0 and 1. Parabolas have eccentricity 1. And hyperbolas have eccentricity greater than 1. The eccentricity for conics of the form in equation 4.7 is [45]

$$\epsilon = \sqrt{\frac{2\sqrt{(A-C)^2+B^2}}{A+C+\sqrt{(A-C)^2+B^2}}}.$$

Our strategy is to choose w_i that minimizes the eccentricity squared ϵ^2. Substituting the coefficients of the implicit form of equation (6.1) into the definition of ϵ^2, taking its derivative with respect to w_i, and solving for the critical point with respect to w_i yields a very simple expression for the rational weight [38]

$$w_i^{ME} = \sqrt{\frac{|Q_{i,2} - Q_{i,0}|^2}{2(|Q_{i,0} - Q_{i,1}|^2 + |Q_{i,2} - Q_{i,1}|^2)}}. \tag{6.7}$$

We put the derivation of equation (6.7) in Appendix B. This expression is always less than 1 and, therefore, leads to elliptical arcs.

Figure 6.1 show an example of a single rational Bézier curve with three fixed control points. This figure shows the entire conic section instead of just the portion of the curve the corresponds to the Bézier curve with its parameter in $[0, 1]$. The only difference between the three curves shown is the rational weight w_i. We show two curves with fixed weights of 0.3 and 0.7 in blue and the curve with minimum eccentricity weight of 0.593 in red. Note that it is not possible to generate circle from this fixed set of control points by only manipulating w_i.

Unfortunately, these minimum eccentricity weights may still generate some undesired shapes in our experiments. Equation 6.7 approaches zero when $P_{i,0}$ approaches $P_{i,2}$. Though a zero weight is valid for rational Bézier curves, equation 6.5 becomes undefined, which leads to instability in the subsequent optimization for some shapes. Looking at this case from a geometric viewpoint, the control points of the Bézier curve are collapsing to form a line and potentially flipping orientation, which should change the sign of the curvature along the curve. This scenario corresponds to a cusp in the curve. Therefore, reproduction of a circle is undesirable as the curve at its point of maximum curvature should have a curvature value that approaches ∞.

6.3.4 Clamped Min Eccentricity Weights

Our solution to the zero weight issue in last section is to simply clamp the minimum value of a weight to 0.5 to avoid these instabilities. Doing so means that we cannot produce a circle if the arc length spanned by the end-points of a Bézier curve is more than $\frac{1}{3}$ of the circle's perimeter. For a set of input points consisting of three points, we will only be able to produce a circle from an

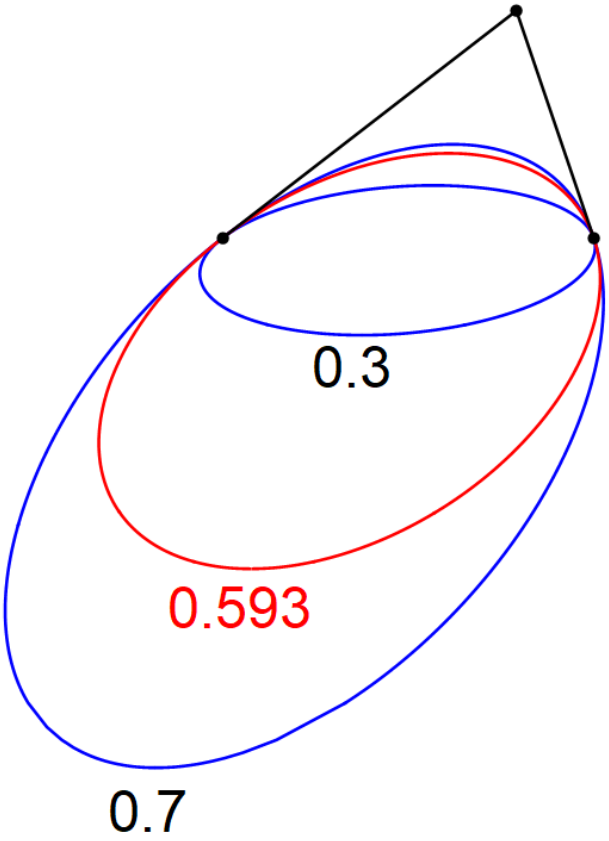

Figure 6.1: A rational Bézier curve with unit weights for the two end-points and different ellipses generated by modifying the weight of the central control point, which is set to 0.3 (blue), 0.7 (blue), and 0.593 (red), which is the minimal eccentricity weight for these control points.

equilateral triangle. However, for other curves consisting of more control points, most non-uniform distributions of input points on a circle can reproduce that circle. Figure 6.2 shows an example of four, non-uniformly spaced points on a circle and the curves produced using different weights.

With this modification, our weight function w_i becomes

$$w_i^{CME} = \max(\mu_i w_i^{ME}, \frac{1}{2}) \tag{6.8}$$

The coefficient μ_i in Equation (6.8) provides a form of tension control for the user to control the shape of the curve. The user can adjust this value to make the curve more round by reducing μ_i or more pointed by increasing μ_i. Or μ_i can be simply set to 1. Figure 6.6 shows an example of

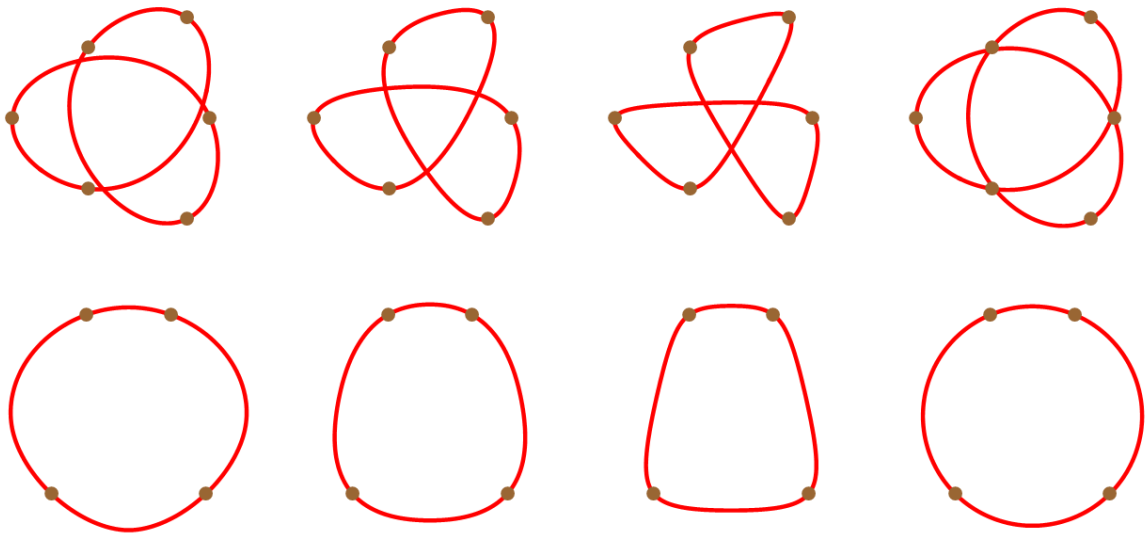

Figure 6.2: Each row contains the same input points $\{P_i\}$ but uses different weights for the input points in each column. The weights from left to right are: 0.6, 1, 2, and our clamped min eccentricity weight.

modifying μ_i to affect the shape of the curve.

6.4 Fast Approach

Our rational quadratic curve is defined by equations (6.2), (6.4), (6.5) and (6.6), with some standalone weight functions. Similar to κ-Curves, we use a local/global approach to solve the system. We substitute equation (6.5) into the rest three equations to eliminate all join points. Then the variables are $\{Q_{i,1}, t_i, \lambda_i\}$. In each iteration, we fix some variables to solve the equations one by one.

6.4.1 Maximum Curvature Condition

For each curve segment $c_i(t)$, we want the input point P_i to be located at a local maximum of curvature. Rewrite equation (6.2) in a Lagrange-type basis in terms of three on-curve points: $Q_{i,0}$, P_i, and $Q_{i,2}$. Since we know the parameters associated with the interpolated end-points (i.e.; $c_i(0) = Q_{i,0}$ and $c_i(1) = Q_{i,2}$), we only need to find the parameter t_i such that P_i is at the point of

maximum curvature satisfying equation (6.4). Solving for $Q_{i,1}$ in equation 6.2 yields

$$Q_{i,1} = \frac{((1-t_i)^2 + 2(1-t_i)t_i w_i + t_i^2)P_i - (1-t_i)^2 Q_{i,0} - t_i^2 Q_{i,2}}{2(1-t_i)t_i w_i}. \tag{6.9}$$

Substituting equation (6.9) into equation (6.3) gives a simple, polynomial expression for t_i in terms of the three on-curve points

$$-w_i A_i (1-t_i)^4 - (A_i + B_i)(1-t_i)^3 t_i + (B_i + C_i)(1-t_i)t_i^3 + w_i C_i t_i^4 = 0 \tag{6.10}$$

where

$$\begin{aligned} A_i &= (Q_{i,0} - P_i) \cdot (Q_{i,0} - P_i) \\ B_i &= (Q_{i,0} - P_i) \cdot (Q_{i,2} - P_i) \\ C_i &= (Q_{i,2} - P_i) \cdot (Q_{i,2} - P_i). \end{aligned}$$

Though a general quartic polynomial may have multiple or no roots in range $[0, 1]$, we prove that

Theorem 4. *Equation 6.10 always has a unique root in [0,1] for any given $Q_{i,0}$, P_i, $Q_{i,2}$ and a positive fixed weight w_i.*

Proof. The left-hand-side (LHS) of equation (6.10) is a polynomial of degree 4. If we use the notation that $v_0 = Q_{i,0} - P_i$ and $v_2 = Q_{i,2} - P_i$. Rewrite the polynomial in the Bézier basis and the coefficients are

$$(-w_i|v_0|^2, -\frac{1}{4}v_0 \cdot (v_0 + v_2), 0, \frac{1}{4}v_2 \cdot (v_0 + v_2), w_i|v_2|^2) \tag{6.11}$$

where $v_0 \neq 0, v_2 \neq 0$, and $w_i > 0$.

We want to show that there is a unique root t_i in $[0, 1]$. When $t = 0$, the value of the LHS of equation (6.10) is the first coefficient of equation (6.11), which is negative. When $t_i = 1$, the value

is the last coefficient of equation (6.11), which is positive. So the there must be at least one root t_i between zero and one because the function is continuous.

To prove the uniqueness, we use the fact that polynomials in Bézier form follow Descartes' rule of signs for bounding the number of real roots of a polynomial [46], which means the number of real roots is less than or equal to the number of sign changes in the sequence of the polynomial's coefficients. In our problem, we have five coefficients with the first being negative, the middle 0, and the last positive. Since weights w_i are always positive, we only need to know the sign of $v_0 \cdot (v_0 + v_2)$ and $v_2 \cdot (v_0 + v_2)$. If one of these coefficients is zero, then no matter what the other one is, the sign changes in the coefficients is one. Hence, there would be at most one root in $[0, 1]$. If both coefficients are non-zero, then the number of sign changes of the coefficients are always one unless both $v_0 \cdot (v_0 + v_2)$ and $v_2 \cdot (v_0 + v_2)$ are negative, which is not possible because the sum is non-negative:

$$v_0 \cdot (v_0 + v_2) + v_2 \cdot (v_0 + v_2) = |v_0 + v_2|^2 \geq 0 \tag{6.12}$$

Therefore, the number of times the coefficients change signs in equation (6.11) is always one, and there is exactly one root in $[0, 1]$ for equation (6.10). □

So equation (6.10) can be expressed as

$$t_i = t_i(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1}, \lambda_{i-1}, \lambda_i, w_i) \in [0, 1]$$

Notice here we only solved the root for $\kappa'(t_i)$=0, which means the interpolated point P_i is either a local minimum or a maximum point. If a small weight $w_i < 1$ is chosen for point $Q_{i,1}$, P_i may be a local minimum of $\kappa(t)$. However, in practice, our choice of weight function and optimization procedure converges to a local maximum point. When $w_i \geq 1$, all interpolated points will be local maximum points of $\kappa(t)$.

Solving the root of a quartic function can be achieved in multiple ways, including the analytic formula of quartic roots. Other methods include Newton's method, bi-search, and etc.

6.4.2 Curvature Condition

To make curves connect with curvature continuity, we rewrite the G^2 condition equation (6.6) in terms of $\{Q_{i,1}, t_i, \lambda_i\}$:

$$\frac{|\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})|(1-\lambda_{i-1})\lambda_i}{w_i^2|Q_{i,1}-Q_{i+1,1}|^3\lambda_i^3} = \frac{|\Delta(Q_{i,1}, Q_{i+1,1}, Q_{i+2,1})|(1-\lambda_i)\lambda_{i+1}}{w_{i+1}^2|Q_{i,1}-Q_{i+1,1}|^3(1-\lambda_i)^3}. \tag{6.13}$$

We can solve a unique parameter λ_i in [0, 1]:

$$\lambda_i = \frac{w_{i+1}\sqrt{|\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})|(1-\lambda_{i-1})}}{w_{i+1}\sqrt{|\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})|(1-\lambda_{i-1})} + w_i\sqrt{|\Delta(Q_{i,1}, Q_{i+1,1}, Q_{i+2,1})|\lambda_{i+1}}}. \tag{6.14}$$

The meaning of equation (6.14) is that fixing the points $Q_{i,0}, Q_{i,1}, Q_{i+1,1}, Q_{i+1,2}$, there is a unique join point on the line segment $Q_{i,1}Q_{i+1,1}$. We can also get the same statement from the geometric view: moving λ_i from 0 to 1, the two sides of the equation (6.13) will wary from ∞ to 0 and from 0 to ∞ monotonically. So the intersection is unique.

6.4.3 Input Point Interpolation

After satisfying the max curvature condition (6.4) and the continuity (6.6), we use the interpolation equation (6.2) to recompute the off-curve points $\{Q_{i,1}\}$. Substituting the G^1 condition (6.5) into (6.2) result in:

$$\begin{aligned} & (1-\lambda_{i-1})(1-t_i)^2 Q_{i-1,1} + (\lambda_{i-1}(1-t_i)^2 + 2(1-t_i)t_i w_i + (1-\lambda_i)t_i^2)Q_{i,1} + \lambda_i t_i^2 Q_{i+1,1} \\ = \;& ((1-t_i)^2 + 2(1-t_i)t_i w_i + t_i^2)P_i. \end{aligned} \tag{6.15}$$

Collect all the equations (6.15) for each input point P_i, we can have a linear system about off-curve points $\{Q_{i,1}\}$ with coefficients about λ_i and t_i. For closed curve, the linear matrix is a cyclic tri-diagonal matrix. For open curve $\{P_0, P_1, \cdots, P_n, P_{n+1}\}$, equation (6.15) works for

$P_2, \cdots, P_{n-1}$. The two end points are slightly different:

$$\begin{aligned} & (2(1-t_1)t_1w_1 + (1-\lambda_1)t_1^2)Q_{1,1} + \lambda_1 t_1^2 Q_{2,1} \\ = \; & ((1-t_1)^2 + 2(1-t_1)t_1w_i + t_1^2)P_1 - (1-t_1)^2 P_0, \\ & (1-\lambda_{n-1})(1-t_n)^2 Q_{n-1,1} + (\lambda_{n-1}(1-t_n)^2 + 2(1-t_n)t_n w_n)Q_{n,1} \\ = \; & ((1-t_n)^2 + 2(1-t_n)t_n w_n + t_n^2)P_n - t_n^2 P_{n+1}. \end{aligned}$$

So the linear matrix for open curves is a tri-diagonal matrix.

6.4.4 Fast Global Approximating Solution

We adopt a global alternating optimization to solve the three geometric conditions (6.4),(6.6) and (6.2) similar to the κ-Curves system.

We start from all weights $w_i = 1$, all join points are at middle points of the control polygon $\lambda_i = 0.5$, all time parameters t_i at 0.5, and use the input points as the initial location of the off-curve control points $Q_{i,1} = P_i$. In this global optimization, we first hold all variables constant except for w_i and pick one choice of the weight functions in section 6.3. Next we solve for t_i from equation (6.10) by holding the $\lambda_i, w_i, Q_{i,1}$. Our next step is to hold all variables constant except for the λ_i and solve for λ_i from Equation (6.14) to make join points curvature continuous. Finally we solve the linear system in equation (6.15) to update the $Q_{i,1}$. We iterate this entire process several times.

Algorithm 2 provides pseudo code for the entire optimization procedure. Note that, in an interactive system, we can take advantage of the temporal coherence of prior solutions to further accelerate the optimization. To do so, we use the results from the previous optimization as the initial guess for the procedure instead of the default values for λ_i, t_i, and w_i.

This procedure projects the current solution onto each constraint in turn along the direction of only one set of variables. Though there is no guarantee that the solution converges, we observe rapid convergence towards the minimizer for the early iterations with progress slowing afterwards. In practice, this approach is accurate enough for fast modeling and can run in real time for most

Algorithm 2 Fast Approach of Rational Quadratic Curves

Input: $\{P_i\}$
procedure FAST SOLUTION
    $\{\lambda_i\} \leftarrow 0.5$
    $\{t_i\} \leftarrow 0.5$
    $\{w_i\} \leftarrow 1$
    $\{Q_{i,1}\} \leftarrow P_i$
    for $j \leftarrow 1$ to 60 **do**
        $\{w_i\} \leftarrow w_i(Q_{i,1}, \lambda_i)$ for any choice of the weight function
        $\{t_i\} \leftarrow$ quartic root of equation (6.10)
        $\{\lambda_i\} \leftarrow$ equation (6.14)
        $\{P_{i,1}\} \leftarrow$ linearSolve equation (6.15)($\{\lambda_i, t_i, w_i\}$)
    return $\{\lambda_i, t_i, w_i, Q_{i,1}\}$

examples. However, there are some examples where the approach will damp between two status and can not converge to the real solution. Hence, we use this fast procedure to find a close initial guess for the non linear optimization in the following section, which then converges rapidly to the solution. In practice, this procedure quickly refines the initial guess to a curve close to the minimum of our energy, which is then refined by the optimization procedure.

6.5 Numerical Optimization

The fast approach in the last section to the solution of equation (6.4), (6.6) and (6.2) is not a standard numerical optimization. In this section, we develop a standard constrained least square numerical optimization embedding all the geometric criteria as polynomial energy terms.

6.5.1 Energy Definition

We require equation (6.2) to be hard constraints since we always need the curve to interpolate all input points.

Maximum Curvature. For each curve segment $c_i(t)$, input point P_i should be interpolated as local maximum curvature point. We utilize the L_2 norm of equation 6.10 and normalize the expression by dividing by the coefficient of the term of the highest degree of variable t_i. If $w_i \neq 1$, the polynomial is a quartic polynomial and the leading coefficient is $(w_i - 1)(C_i - A_i)$. And if

$w_i = 1$, then the quartic term is zero and the polynomial is a cubic function with leading coefficient $A_i - 2B_i + C_i$. This process yields the energy term for local maximal curvature

$$E_c = \sum_{i=1}^{n} (\frac{-w_i A_i (1-t_i)^4 - (A_i + B_i)(1-t_i)^3 t_i + (B_i + C_i)(1-t_i) t_i^3 + w_i C_i t_i^4}{L_i})^2 \tag{6.16}$$

where L_i is the leading coefficient of the numerator in equation (6.16).

$$L_i = \begin{cases} (w_i - 1)(C_i - A_i), & w_i \neq 1 \\ A_i - 2B_i + C_i, & w_i = 1 \end{cases}$$

If we substitute the G^1 condition (6.5) into equation (6.16), we can write E_c in terms of the variables $Q_{i,1}, \lambda_i, t_i$.

G^2 condition. We can easily encode this requirement as an error function of the squared curvature using equation 6.13.

$$(k_i(1)^2 - k_{i+1}(0)^2)^2 = \left(\frac{\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})^2 (1-\lambda_{i-1})^2}{w_i^4 |Q_{i,1} - Q_{i+1,1}|^6 \lambda_i^4} - \frac{\Delta(Q_{i,1}, Q_{i+1,1}, Q_{i+2,1})^2 \lambda_{i+1}^2}{w_{i+1}^4 |Q_{i,1} - Q_{i+1,1}|^6 (1-\lambda_i)^4} \right)^2$$

Given that $|Q_{i,1} - Q_{i+1,1}| > 0$, dropping this term does not affect the null space of this energy term. Remove this term and our final G^2 energy is:

$$E_{G^2} = \sum_{i=1}^{n} \left(\frac{\Delta(Q_{i-1,1}, Q_{i,1}, Q_{i+1,1})^2 (1-\lambda_{i-1})^2}{w_i^4 \lambda_i^4} - \frac{\Delta(Q_{i,1}, Q_{i+1,1}, Q_{i+2,1})^2 \lambda_{i+1}^2}{w_{i+1}^4 (1-\lambda_i)^4} \right)^2 \tag{6.17}$$

Weight energy. We can simply compute the L^2 norm of the weight to any of users' choice of the weight function $\tilde{w}_i$. Here the weight function $\tilde{w}_i$ can be either of the trivial weights, Minimum Eccentricity Weight w_i^{ME}, Clamped Min Eccentricity Weight w_i^{CME} or any other applicable weight functions. Then the general weight energy term can be defined as:

$$E_w = \sum_{i=1}^{n} (w_i - \tilde{w}_i)^2 \tag{6.18}$$

For any specific weight function, the form of the weight energy can be tweaked for the choice of the weight function. For example the Minimal Eccentricity Weight is a square root of a fraction of polynomial, then the energy term can be defined as

$$E_w = \sum_{i=1}^{n} (w_i^2 - (w_i^{ME})^2)^2 = \sum_{i=1}^{n} (w_i^2 - \frac{|Q_{i,2} - Q_{i,0}|^2}{2(|Q_{i,0} - Q_{i,1}|^2 + |Q_{i,2} - Q_{i,1}|^2)})^2$$

to avoid square root in the final energy expression.

Then the full geometric energy of the curve is

$$E(\{Q_{i,1}, \lambda_i, t_i, w_i\}) = E_c + E_{G^2} + E_w \tag{6.19}$$

6.5.2 Numerical Solution

Given the input points $\{P_i\}$, we use the energy terms from previous sections to form a numerical optimization problem for our curve as

$$\min_{Q_{i,1} \in \mathbb{R}^2, \lambda_i \in (0,1), t_i \in (0,1), w_i > 0} E_c(Q_{i,1}, \lambda_i, t_i, w_i) + E_{G^2}(Q_{i,1}, \lambda_i, w_i) + E_w(Q_{i,1}, \lambda_i, w_i) \tag{6.20}$$

with the constraints

$$\begin{cases} & \vdots \\ & (1-\lambda_{i-1})(1-t_i)^2 Q_{i-1,1} + (\lambda_{i-1}(1-t_i)^2 + 2(1-t_i)t_i w_i + (1-\lambda_i)t_i^2) Q_{i,1} + \lambda_i t_i^2 Q_{i+1,1} \\ = & ((1-t_i)^2 + 2(1-t_i)t_i w_i + t_i^2) P_i \\ & \vdots \end{cases}$$

Note that E_w can be omitted if we do not desire automatic reproduction of circles, which also removes w_i from the variables but still gives the user control over the w_i to affect the shape of the curve.

Since the constraints are linear in the points $\{Q_{i,1}\}$, we can solve for the $\{Q_{i,1}\}$ as a function

of $\{\lambda_i, t_i, w_i\}$ from the constraints. Then insert the expressions of $Q_{i,1}$ back to (6.20) and generate a new unconstrained optimization with reduced number of variable:

$$\min_{\lambda_i, t_i, w_i} E_c(\lambda_i, t_i, w_i) + E_{G^2}(\lambda_i, t_i, w_i) + E_w(\lambda_i, t_i, w_i). \tag{6.21}$$

The constraints on the variables are now simplified to $\lambda_i \in (0, 1), t_i \in (0, 1)$ and $w_i > 0$. So the optimization problem reduces to minimize a new energy only about $\{\lambda_i, t_i, w_i\}$:

$$E_{new}(\lambda_i, t_i, w_i) = E_c(\lambda_i, t_i, w_i) + E_{G^2}(\lambda_i, t_i, w_i) + E_w(\lambda_i, t_i, w_i) \tag{6.22}$$

We found numerical performance of this new energy is much better than the original definition (6.20).

To solve this simplified minimization problem, we build the functions to evaluate the energy value, the gradient of the energy and the Hessian of the energy. For the energy value, we first solve all off-curve points $\{Q_{i,1}\}$ from equation (6.15). Then use $\{Q_{i,1}\}$ and $\{\lambda_i\}$ to compute the location of all join points $\{Q_{i,0}, Q_{i,2}\}$. Then we use these 2D points together with the parameters $\{\lambda_i, t_i, w_i\}$ to compute the energy value E in equation (6.20) as the energy value E_{new}. The derivatives can be computed by the chain rule

$$\begin{aligned}
\frac{\partial E_{new}(\lambda_i, t_i, w_i)}{\partial \lambda_i} &= \frac{\partial E(Q_{i,1}, \lambda_i, t_i, w_i)}{\partial \lambda_i} + \sum_j \frac{\partial E(Q_{j,1}, \lambda_i, t_i, w_i)}{\partial Q_{j,1}} \cdot \frac{\partial Q_{j,1}}{\partial \lambda_i} \\
\frac{\partial E_{new}(\lambda_i, t_i, w_i)}{\partial t_i} &= \frac{\partial E(Q_{i,1}, \lambda_i, t_i, w_i)}{\partial t_i} + \sum_j \frac{\partial E(Q_{j,1}, \lambda_i, t_i, w_i)}{\partial Q_{j,1}} \cdot \frac{\partial Q_{j,1}}{\partial t_i} \\
\frac{\partial E_{new}(\lambda_i, t_i, w_i)}{\partial w_i} &= \frac{\partial E(Q_{i,1}, \lambda_i, t_i, w_i)}{\partial w_i} + \sum_j \frac{\partial E(Q_{j,1}, \lambda_i, t_i, w_i)}{\partial Q_{j,1}} \cdot \frac{\partial Q_{j,1}}{\partial w_i}
\end{aligned} \tag{6.23}$$

The derivative $\frac{\partial Q_{j,1}}{\partial \lambda_i}, \frac{\partial Q_{j,1}}{\partial t_i}, \frac{\partial Q_{j,1}}{\partial w_i}$ can be computed from the linear constraints (6.15) using implicit differential. Similarly we can compute the second derivative of the energy.

While any initial guess for the optimization could be chosen, we start with a very simple initial guess with $\lambda_i = 0.5$, $t_i = 0.5$, and $w_i = 1$. And solve a set of off-curve points $\{Q_{i,1}\}$ from the

linear constraints. This initial guess is an interpolatory C^1 curve. Yet this curve almost certainly does not satisfy the automatic weight conditions, curvature continuity, or maximum curvature conditions. In most cases, this simple initial guess can lead to slow convergence. To accelerate the optimization, we use the result of the fast solution in section 6.4 as the initial status. Because the fast solution is close enough to the final curve, our non linear optimization can converge quickly. Algorithm 3 shows the whole procedure of our rational quadratic curve.

Algorithm 3 Rational Quadratic Curve

Input: $\{Q_i\}$
procedure ENERGY(λ_i, t_i, w_i)
    $\{Q_{i,1}\} \leftarrow$ equation (6.15) of ($\{\lambda_i, t_i, w_i\}$)
    return $E_c(Q_{i,1}, \lambda_i, t_i, w_i) + E_{G^2}(Q_{i,1}, \lambda_i, w_i) + E_w(Q_{i,1}\lambda_i, w_i)$
procedure GRADIENT(λ_i, t_i, w_i)
    $\{Q_{i,1}\} \leftarrow$ equation (6.15) of ($\{\lambda_i, t_i, w_i\}$)
    return $(\frac{\partial E}{\partial \lambda_i}, \frac{\partial E}{\partial t_i}, \frac{\partial E}{\partial w_i})$ by equation (6.24)
procedure HESSIAN(λ_i, t_i, w_i)
    $\{Q_{i,1}\} \leftarrow$ equation (6.15) of ($\{\lambda_i, t_i, w_i\}$)
    return $((\frac{\partial^2 E}{\partial \lambda_i^2}, \frac{\partial^2 E}{\partial \lambda_i \partial t_i}, \frac{\partial^2 E}{\partial \lambda_i \partial w_i}), (\frac{\partial^2 E}{\partial t_i \partial \lambda_i}, \frac{\partial^2 E}{\partial t_i^2}, \frac{\partial^2 E}{\partial t_i \partial w_i}), (\frac{\partial^2 E}{\partial w_i \partial \lambda_i}, \frac{\partial^2 E}{\partial w_i \partial t_i}, \frac{\partial^2 E}{\partial w_i^2}))$
procedure MAIN
    $\{\lambda_i, t_i, w_i\} \leftarrow$ Fast_Solution($\{P_i\}$) from Algorithm 2
    $\{\lambda_i, t_i, w_i\} \leftarrow$ minimize Energy(λ_i, t_i, w_i) with $0 < \lambda_i, t_i < 1$, $w_i > 0$
    $\{Q_{i,1}\} \leftarrow$ linearSolve equation (6.15) of ($\{\lambda_i, t_i, w_i\}$)
    $\{Q_{i,0}, Q_{i,2}\} \leftarrow$ the G^1 condition equation (6.5)
    return $\{(Q_{i,0}, Q_{i,1}, Q_{i,2})\}$

We implemented the optimization in both Wolfram Mathematica and C++. In C++, we use the Cholesky decomposition from the Eigen[1] library to solve the linear system of $\{Q_{i,1}\}$ and use the boxed constrained numerical optimization of Alglib[2] library to compute the minimizer of E_{new}. According to our experiments, the combination of the fast solution in section 6.4 and the non linear optimization above can compute the curves in real time for curve modeling tools. The fast solution

[1]https://eigen.tuxfamily.org/
[2]https://www.alglib.net/

is fast when far from the energy minimizer and slow down when approaching the minimizer. The standard numerical optimization is only fast when close to the minimizer. So the combination works well in real practice.

6.6 Results and Conclusion

Our curves extend the geometric properties of κ-Curves [4] by introducing rational weight parameters and improve the optimization procedure for these curves. Figure 6.2 shows the effect of choosing different rational weights on the shape of the curve. In each case the weights for all control points are uniform though the user could set weights on a control point by control point basis. As the weights become higher, the curves tend to become sharper, though still smooth. When all weights are one, we produce the piece-wise polynomial κ-curves. The right portion of figure 6.2 shows the result of our minimum eccentricity weights with the bottom right figure reproducing a circle as the four points are co-circular even though they are non-uniformly spaced on a circle.

Our minimum eccentricity weights do not require that the user set any weights individually on the curve. Instead, our method chooses the weights automatically through our optimization. At the same time, the minimum eccentricity weights become unstable as weights approach zero, which leads to our clamped solution. Figure 5.6 shows the results of clamping our minimum eccentricity weights compared to κ-curves [4]. In both cases, our minimum eccentricity weights provide a "fairer" curve as measured by variation of curvature. At the same time, all curves have the property that the local maximum of curvature is reached at the control points even though the (absolute value of) curvature is near constant for the far left image. Each curve is also curvature continuous everywhere except at inflection points.

In figure 6.3 (middle), our clamped minimum eccentricity weights produce a solution similar to κ-curves though a bit more round. In regions where curves have concave corners, our clamped minimum eccentricity weights often produce shapes that locally resemble κ-curves.

However, figure 6.4 shows an example with mostly convex (though overlapping) curves. In these cases, our clamped minimum eccentricity weights appears far closer to the unclamped version

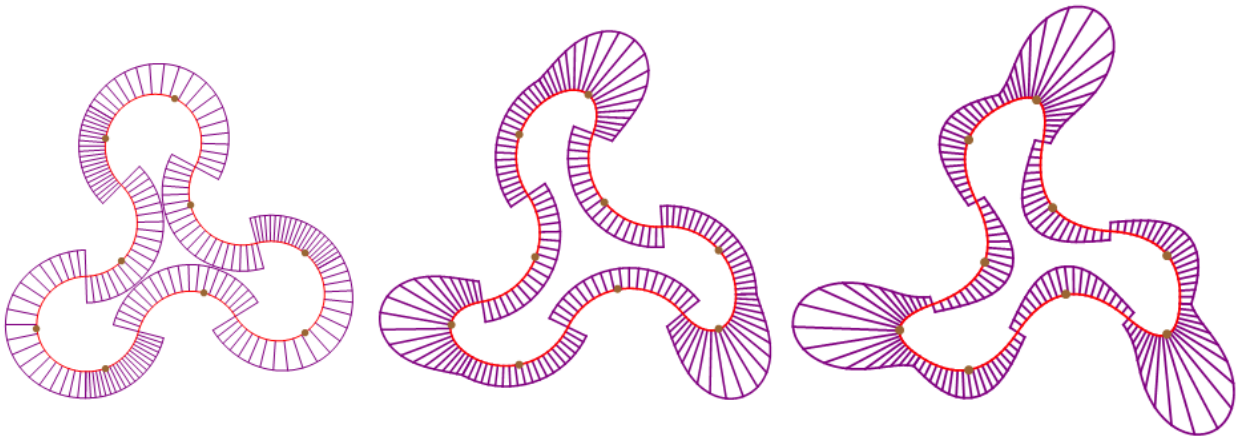

Figure 6.3: The purple lines show curvature normals. Each picture uses the same input points with different weight functions. From left to right: min-eccentricity weights, clamped min-eccentricity weights, and the original κ-curve.

of the curves than κ-curves. Figure 6.5 shows the results of our method on a non-symmetric shape with non-uniform control point spacing from [17]. Figure 6.6 demonstrates our minimum eccentricity weights where the user modifies the parameter μ_i at a single control point to make the curve appear sharper. Figure 6.7 shows our method creating a cusp as points move closer to one another. Cusps represent local maxima of curvature and, hence, can only appear at input points.

Like κ-curves, our solution is a global solution, which means the whole curve will change after a small movement of a control point. However, in practice, the influence of a control point drops dramatically away from that point. Figure 6.8 shows an example of this effect using our minimum eccentricity weights. In both cases the original curve is shown in blue with the newly modified red curve drawn on top. The resulting curve only changes within a small region away from the initial control point in any significant fashion.

The rational quadratic curve also improves upon the original κ-curve optimization by constructing a formal energy that can be minimized in addition to extending the method to rational curves. Figure 6.9 shows the results of this optimization with the top row having a uniform weight of one and the bottom using our minimum eccentricity weights. The starting curve appears on the left and its error is listed below. After performing 60 iterations of our initial guess, we obtain the

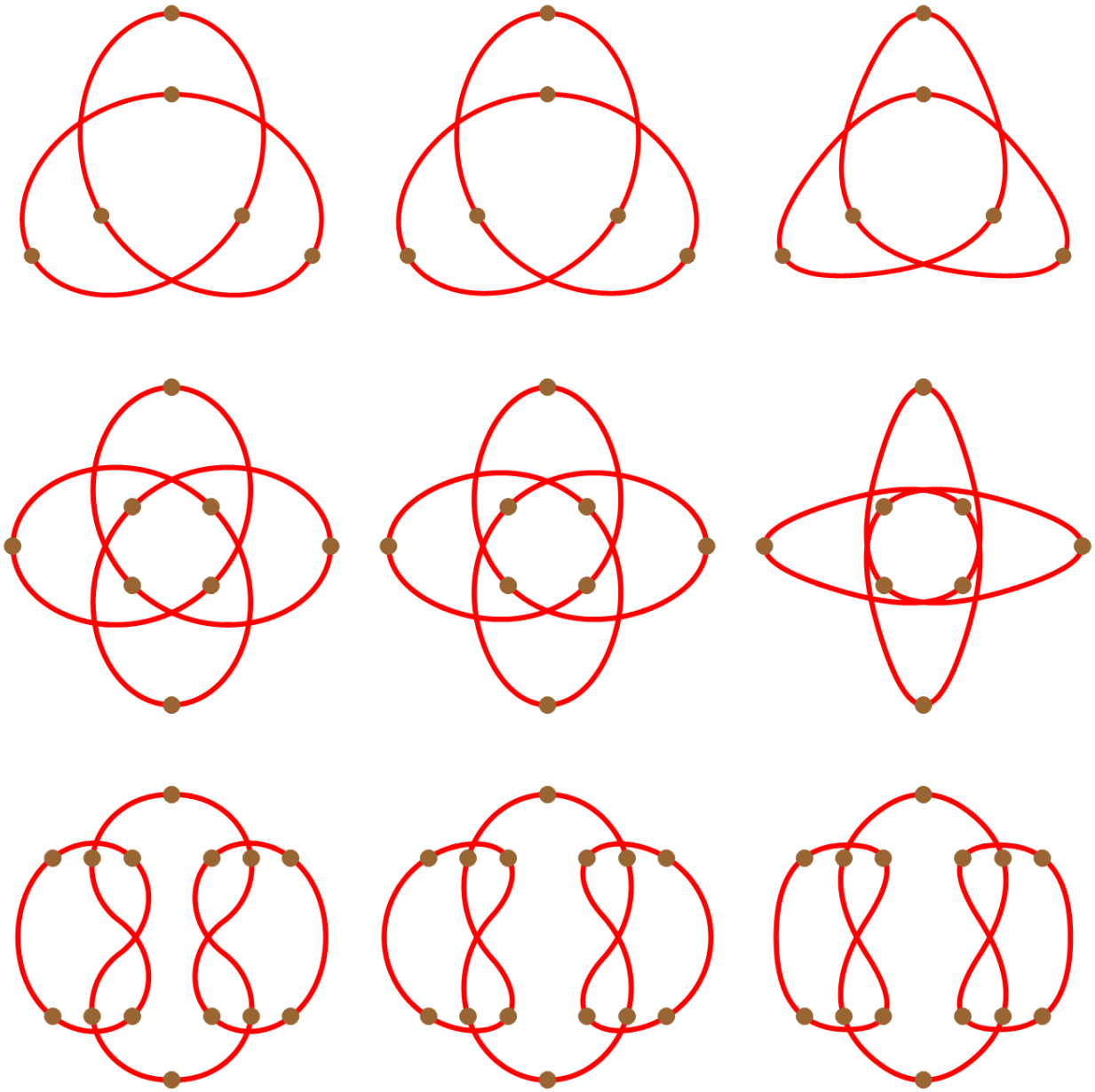

Figure 6.4: Comparisons between different weight choices. Left to right: min eccentricity weights, clamped min eccentricity weights, original κ-curve.

curve in the middle. For many applications, this curve may be a sufficiently good approximation. However, optimizing our energy function produces a significant reduction in error as shown on the right, which does affect the shape of the curve and is particularly visible on the bottom row.

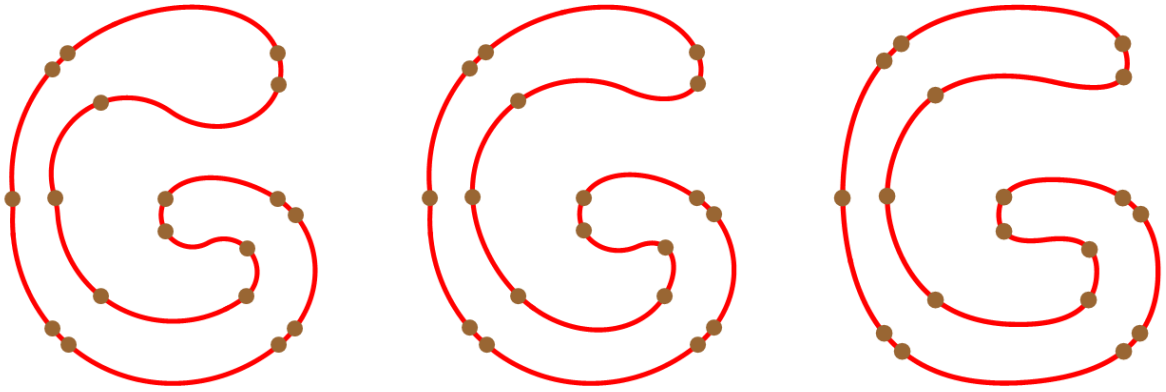

Figure 6.5: An asymmetric example with non-uniformly spaced control points. From left to right: min eccentricity weights, clamped min eccentricity weights, original κ-curve.

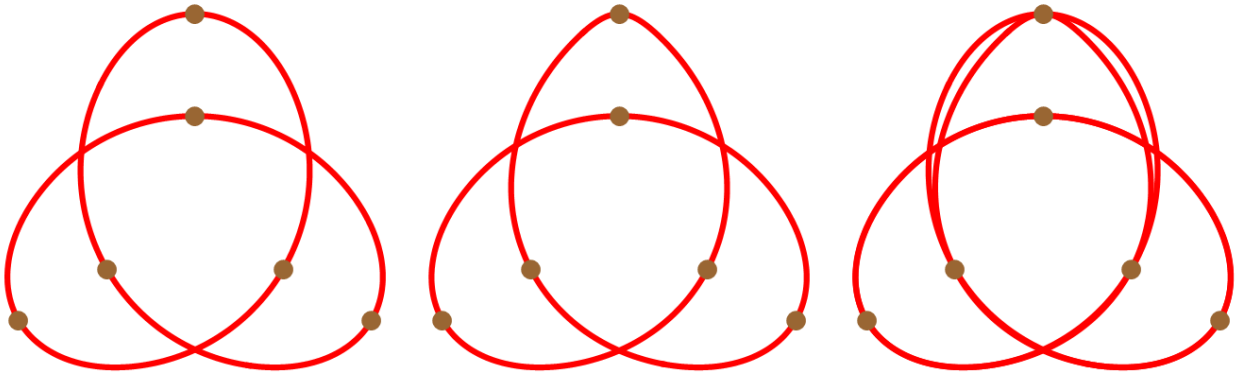

Figure 6.6: Clamped minimum eccentricity weight curves with tension values μ_i set to 1 (left) and with the weight of the top input point set to 2 (middle). The right picture shows the overlapping curves for comparison.

Figure 6.7: The creation of a cusp using clamped min eccentricity weights. Moving the 2^{nd} and 4^{th} points closer creates a cusp at an input point.

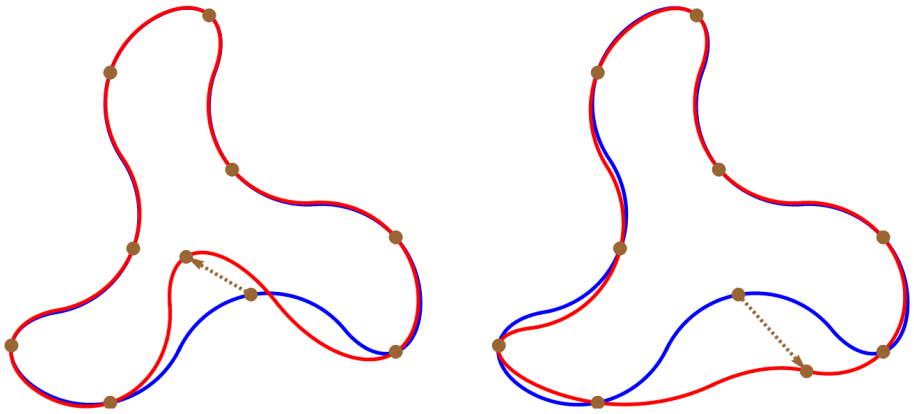

Figure 6.8: The blue/red curves show our curves before/after moving a single input point. Despite the global nature of the optimization, the change in the curve tends to decrease with distance from the input point.

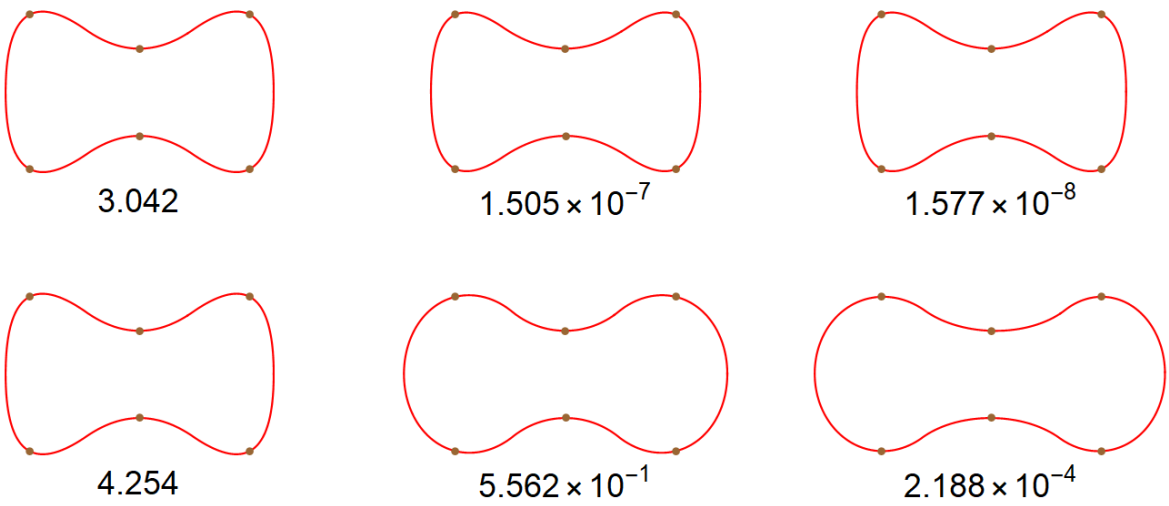

Figure 6.9: Curve optimization. Each image has the same input points with the error shown below. Each row shows the initial guess (left), after 60 iterations of refining the initial guess (middle), and the result of our optimization (right). The first row uses weight 1 while the second row are our clamped min eccentricity weights. The top middle picture corresponds to the κ-curve result in [4].

6.7 Limitations

In our construction we require that the rational weights w_i for all off-curve Bézier control points $Q_{i,1}$ be positive. The implication of this requirements is that all pieces of the rational quadratic

curve are minor arcs. Hence, in the elliptical case, the arc cannot span more than half of an ellipse. This restriction implies that when all input points are on a circle, the curve produced by our minimum eccentricity method may not form a circle in some scenarios. In particular, when the end-points of each Bézier curve $Q_{i,0}, Q_{i,2}$ span more than half a circle, we cannot produce an exact circle.

The connection between the end-points of each Bézier curve and the control points the user specifies is, unfortunately, governed by a non-linear optimization. Let θ be the angle between consecutive input points formed from the center of the circumscribed circle as shown in figure 6.10. If the angle spanned by two consecutive segments is less than π, for all segments, then the end-points of each Bézier curve cannot span more than half a circle since $Q_{i,0}$ lies in between P_{i-1} and P_i. Yet this property is too restrictive in practice. We tested our technique on many different point distributions of input points P_i on a circle. In our tests we found that if the angle spanned by consecutive input points was less than π, our method reproduced the circumscribed circle. However, our method would often fail to produce a circle if the angle spanned by consecutive input points exceeded π. Figure 6.10 shows several examples where non-uniform point distributions produce circles and two failure cases where our method was unable to reproduce a circle.

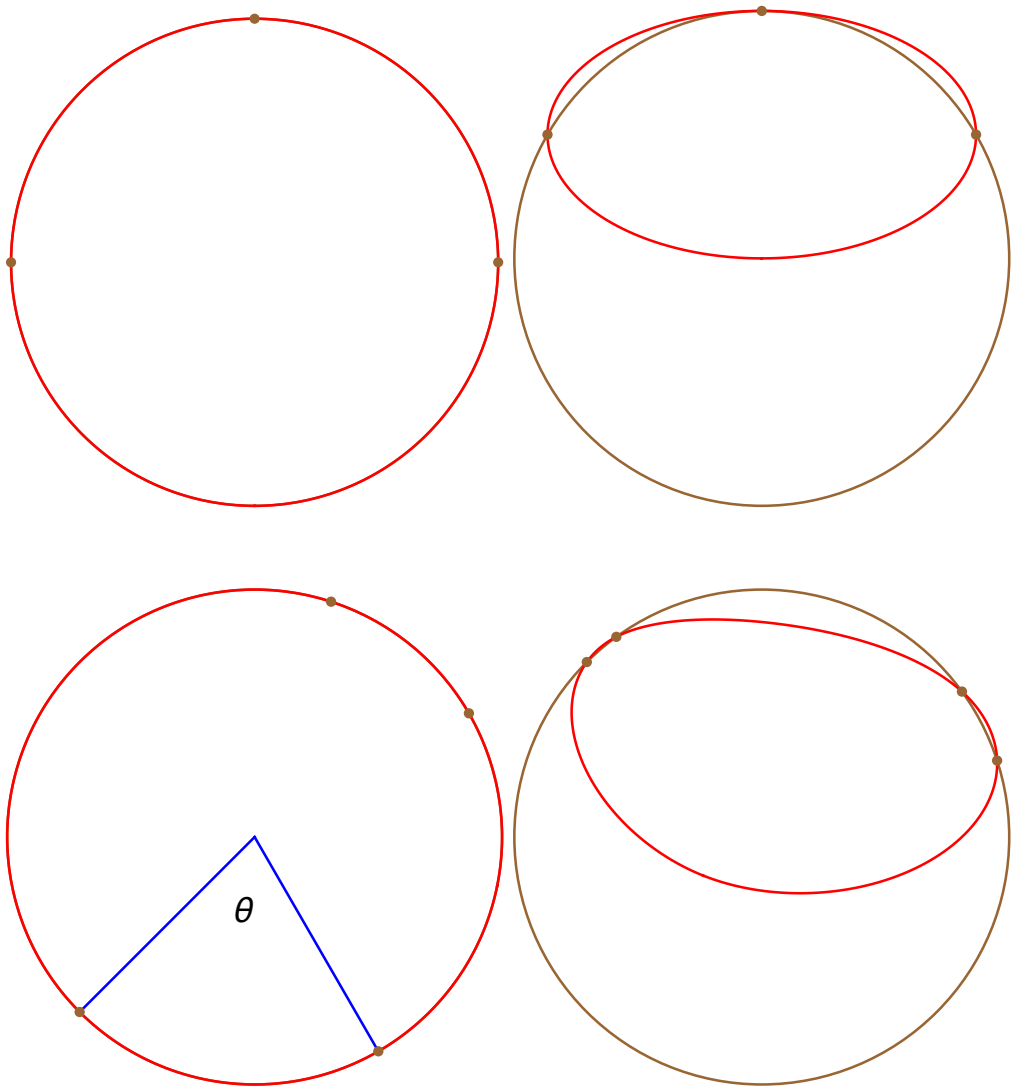

Figure 6.10: Circle reproduction with three (top) or four points (bottom). The left column has input points with central angles less than π. The right has points with one central angle over π, which leads to a non-circular curve.

7. CUBIC INTERPOLATION WITH CURVATURE CONTROL

Both quadratic κ-Curves and the rational quadratic curves solved the problem of 2D curves interpolating control points as local maximum curvature points and the curvature should be decreasing first then increasing from one control point to the next control point. And both curves have G^2 continuity except at inflection points. However, κ-Curves do not have extra degrees of freedom, i.e. the result is unique to each set of input points, which means the users can not customize the result curve. The rational quadratic curve has one degree of freedom for each input point. The point weight can directly affect the sharpness at the input point, and indirectly affect the tangent and curvature at the input point.

Our original motivation of the proposed method is to provide a curve that interpolates the control points at local maximum curvature points with local tangential directions and tension control. To provide a complete replacement for the existing pen tool, users should be able to assign the tangent directions and potentially curvature at the input points. Doing so would decouple the global constraints from κ-Curves and its extension to produce true locality. To achieve this new requirement, we adapt the cubic curve and hybrid curve in section 4.3 and develop a new framework using Hermite interpolation.

7.1 Problem Formulation

Given a set of 2D points $\{P_i\}$, we want to interpolate all the points and at each point, the tangent directions $\{T_i\}$ and curvature k_i. Because the location, tangent and curvature information at each point has been fixed, we can split the curves into several segments with each segment connecting two input points. We only need to think about single piece of the cubic or hybrid curves since global G^2 condition is achieved by the geometric information at the control points.

We focus on the curve segment between points P_i and P_{i+1}. Other segments are the same. We

would like to compute some curve piece $C_i(t)$ satisfying

$$\begin{aligned} c_i(0) &= P_i \\ c_i(1) &= P_{i+1} \\ \frac{c_i'(0)}{|c_i'(0)|} &= T_i \\ \frac{c_i'(1)}{|c_i'(1)|} &= T_{i+1} \\ \kappa_i(0) &= k_0 \\ \kappa_i(1) &= k_1 \end{aligned}$$

And there should be some parameter $\tilde{t} \in (0,1)$ s.t. the curvature is decreasing from k_0 to some positive number on $[0,\tilde{t}]$ and increasing to k_1 on $[\tilde{t},1]$.

7.2 Construction

Since we are computing the curve segment with boundary constraints, we eliminate the monotonic curvature requirement by using two curve segments from section 4.3, each with monotonically varying curvature that interpolate the constraints at the end points and meet with curvature continuity. Figure 7.1 shows our idea.

Denote the two input points P_i, P_{i+1} as Q_0 and Q_4 for this segment. First we put tangent vectors T_i and negative T_{i+1} at Q_0 and Q_4. We extend the two tangent vectors to some points Q_1 and Q_3 with distances a and b to the input points: $Q_1 = Q_0 + aT_i, Q_3 = Q_4 + b(-T_{i+1})$. On line segment between Q_1 and Q_3, we choose some splitting point $Q_2 = (1-\lambda)Q_1 + \lambda Q_3$ where λ is between 0 and 1. Then we use triple (Q_0, Q_1, Q_2) and (Q_2, Q_3, Q_4) as the control triangles. We can use the three control points as a quadratic control polygon directly, or elevate to the restricted cubic curve in Section 4.3, or we can use a hybrid curve of both quadratic and cubic curves.

We first introduce our three-point cubic curve. We can construct the cubic Bézier control points

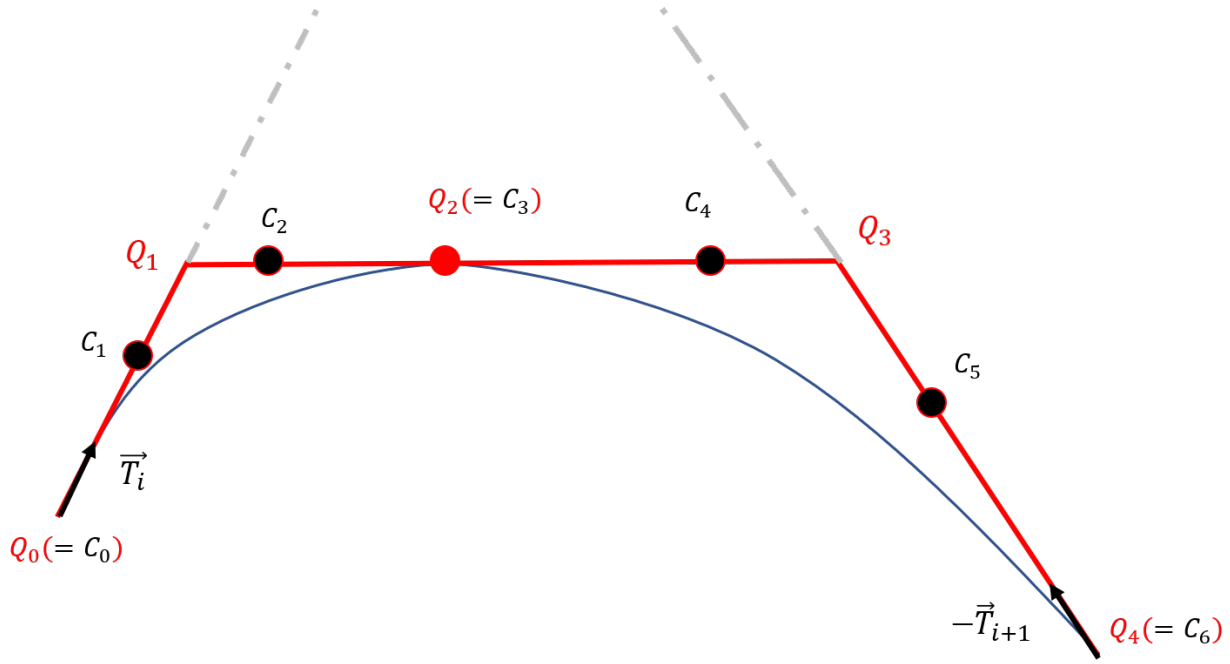

Figure 7.1: Connect two single quadratic or cubic pieces as a whole curve segments. Quadratic control points are rendered in red $\{Q_i\}$, while the cubic control points $\{C_i\}$ are in black.

$\{C_i\}_{i=0}^{6}$ as following:

$$
\begin{aligned}
C_0 &= Q_0 = P_i \\
C_1 &= Q_0 + \frac{1}{2}\frac{(C_2 - Q_0)\cdot(Q_1 - Q_0)}{(Q_1 - Q_0)\cdot(Q_1 - Q_0)}T_i = Q_0 + (\frac{3-\lambda}{6}a + \frac{\lambda}{6}(Q_4 + b(-T_{i+1}) - Q_0)\cdot T_i)T_i \\
C_2 &= \frac{2}{3}Q_1 + \frac{1}{3}Q_2 = (1 - \frac{\lambda}{3})(Q_0 + aT_i) + \frac{\lambda}{3}(Q_4 + b(-T_{i+1})) \\
C_3 &= (1-\lambda)Q_1 + \lambda Q_3 = (1-\lambda)(Q_0 + aT_i) + \lambda(Q_4 + b(-T_{i+1})) \\
C_4 &= \frac{1}{3}Q_2 + \frac{2}{3}Q_3 = \frac{1-\lambda}{3}(Q_0 + aT_i) + \frac{2+\lambda}{3}(Q_4 + b(-T_{i+1})) \\
C_5 &= Q_4 + \frac{1}{2}\frac{(C_4 - Q_4)\cdot(Q_3 - Q_4)}{(Q_3 - Q_4)\cdot(Q_3 - Q_4)}(-T_{i+1}) \\
&= Q_4 + (\frac{2+\lambda}{6}b + \frac{1-\lambda}{6}(Q_0 + aT_i - Q_4)\cdot(-T_{i+1}))(-T_{i+1}) \\
C_6 &= Q_4
\end{aligned}
\tag{7.1}
$$

To keep the join point Q_2 as a G^2 connection point, add the constraint that that curvature of the

left side of point Q_2 equals to the right side of Q_2:

$$\begin{aligned} &\frac{(1-\frac{\lambda}{3}((P_{i+1}+b(-T_{i+1})-P_i)\cdot T_i/a-1))|\Delta(P_i,Q_1,Q_3)|}{\lambda^2} \\ =&\frac{(1-\frac{1-\lambda}{3}((P_i+aT_i-P_{i+1})\cdot(-T_{i+1})/b-1))|\Delta(Q_1,Q_3,P_{i+1})|}{(1-\lambda)^2} \end{aligned} \tag{7.2}$$

where $\Delta(\cdot,\cdot,\cdot)$ is the signed area of a triangle. The G^2 constraint of the join point Q_2 is a cubic equation of parameter λ. When λ goes to 0^+, the left hand side of equation 7.2 is $+\infty$ while the right hand side is some limited number. And when λ goes to 1^-, the left hand side is limited and the right hand side is $+\infty$. This simple observation indicated equation 7.2 always has at least a root $\lambda \in (0,1)$. However, the root does not necessarily need to be unique.

The tangent control at two end points has been satisfied by construction. The curvature conditions at c_0 and c_1 can be achieved by the constraints:

$$\begin{aligned} k_i &= \frac{8}{9}\frac{\lambda(\det(T_i,P_{i+1}-P_i)+b\det(T_i,-T_{i+1}))}{((1-\frac{\lambda}{3})a+\frac{\lambda}{3}(P_{i+1}+b(-T_{i+1})-P_i)\cdot T_i)^2|T_i|^3} \\ k_{i+1} &= \frac{8}{9}\frac{(1-\lambda)(\det(-T_{i+1},P_i-P_{i+1})+a\det(-T_{i+1},T_i)))}{(\frac{2+\lambda}{3}b+\frac{1-\lambda}{3}(P_i+aT_i-P_{i+1})\cdot(-T_{i+1}))^2|T_{i+1}|^3} \end{aligned} \tag{7.3}$$

7.3 Optimization

In the construction of each curve segment between P_i and P_{i+1}, points P_i, P_{i+1}, vector T_i, T_{i+1} and curvature k_i, k_{i+1} are input constant. We have three variables $a>0, b>0, 0<\lambda<1$ and three constraints, one for the G^2 condition and two for the curvature fitting at end points. Here we provide a fast approach of the root. Any standard numerical optimization can be adopted to refine the result. In our experiments, the error of the fast approach is small enough and speed fast enough that real time curve modeling is possible.

We start from $a=1, b=1, \lambda=\frac{1}{2}$. In each iteration, fix a, b and solve λ from equation 7.2. We can always use Cardano's formula to solve roots. Here we use an adaptive method to update the root in an iterative way. Between iterations, the value change of λ can not be large. So each time we start from the value in last iteration and compute the derivative of λ and use Newton's method

to solve the root of equation (7.2). Then fix λ and solve a, b from equation 7.3. Here a, b is the intersection of two quadratics in the first quadrant. We can also use the iterative strategy that uses the value from last iteration as the initial status in current iteration. In our experiments, usually 30 iterations of computing λ and (a, b) are enough to converge. Algorithm 4 shows the pseudo code of our optimization.

Algorithm 4 Cubic Cuves

Input: $\{P_i, T_i, k_i\}$
procedure LOCAL CURVE($P_i, P_{i+1}, T_i, T_{i+1}, k_i, k_{i+1}$)
    $\lambda \leftarrow 0.5$
    $a \leftarrow 1$
    $b \leftarrow 1$
    $Q_1 \leftarrow P_i + aT_i$
    $Q_3 \leftarrow P_{i+1} - bT_{i+1}$
    $C_3 \leftarrow (1-\lambda)Q_1 + \lambda Q_3$
    $\{C_1, C_2, C_4, C_5\} \leftarrow$ equation (7.2)
    for $j \leftarrow 1$ to 30 **do**
        $\lambda \leftarrow$ Solve the cubic root of equation (7.2)
        $a, b \leftarrow$ Solve from equation (7.3)
        $Q_1 \leftarrow P_i + aT_i$
        $Q_3 \leftarrow P_{i+1} - bT_{i+1}$
        $C_3 \leftarrow (1-\lambda)Q_1 + \lambda Q_3$
        $\{C_1, C_2, C_4, C_5\} \leftarrow$ equation (7.2)
    return (P_i, C_1, C_2, C_3), (C_3, C_4, C_5, P_{i+1})
procedure GLOBAL CUBIC CURVE($\{P_i, T_i, k_i\}$)
    for $i \leftarrow 1$ to n **do**
        $c_i \leftarrow$ LocalCurve($P_i, P_{i+1}, T_i, T_{i+1}, k_i, k_{i+1}$)
    return $\{c_i\}$

Notice after convergence, the location of the join point may not be in the valid zone computed from section 4.3.3. So the input values of the tangents and curvatures can not be arbitrary. This is also a limitation of this method.

7.4 Relationship to Quadratic Curves

Our restricted cubic curve have the same degrees of freedom with a quadratic curve. If we fix the two end points, the two middle cubic control points are defined uniquely by the off-curve quadratic control point. Since the cubic curve is controlled by the quadratic polygon in figure 4.5, we use the quadratic polygon $\{Q_0, Q_1, Q_2\}$ for simpler discussion in the following. According to figure 4.6, if we place Q_0 at (0, 0), Q_1 at $(a, 0)$, then the region of monotonic curvature of the quadratic and cubic curve will only rely on the x-coordinate of the location of Q_2. The region of monotonic curvature for quadratic curves is the the semi plane where $x \geq 2a$. And the monotonic region for our cubic one is $-2a \leq x \leq 4a$. Notice $x = 2a$ means the angle $\angle Q_0Q_1Q_2$ is a right angle. So our cubic curve can create curves with monotonic curvature when $\angle Q_0Q_1Q_2$ is less than 90°. In contrast, quadratic curves must have an angle for $\angle Q_0Q_1Q_2$ of 90° or more to have monotonic curvature. Our hybrid curve combines both curvature regions and leads to a more flexible modeling paradigm.

Another aspect compared to κ-Curves is range of the curves. The restricted cubic curves are defined by the input points, and curvature and tangents at input points. If we use curvature and tangent information from the κ-Curve computed from the input points, we can compute a unique curve from our cubic method since there is no extra degree of freedom. On the other hand, the κ-Curve satisfies all the geometric constraints of our cubic curve. So we are reproducing the κ-Curves if choose the specific tangents and curvature at the input points. This result is important because using the new restricted cubic curve, we are not losing any intrinsic geometric properties of κ-Curves. If we split each segment of a quadratic κ-Curve at the input points, we get two half quadratic segments. If we connect the two half segments between two input points as a single piece of G^2 curve, we are forming a similar Hermite interpolating curve with our restricted cubic curves. Indeed, this cubic curve is an extension to the original quadratic κ-Curve.

7.5 Results

The main geometric feature of our cubic curve is that we allow the tangent vectors and curvatures to be defined at input points. We show several examples of this feature. Figure 7.2 shows an example of four input points. We fix the curvature at the input points and modify the tangent direction at one of the input points. Figure 7.3 shows an example of four input points where the tangent vectors are fixed and one input point is assigned different values of curvature.

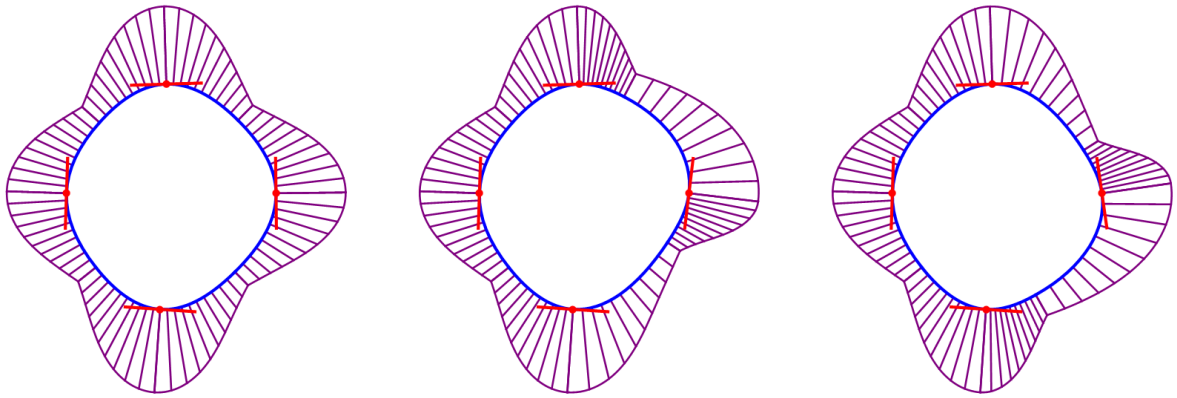

Figure 7.2: All the three pictures have the same input points and the same curvature assigned for each input point. Compare to the left picture, the tangent direcion of the right point of the middle and the right pictures are rotated 7.5° in the clockwise and counter-clockwise directions.

7.6 Conclusion and Limitation

In this chapter, we developed a framework of compounding quadratic and restricted cubic curve segments with boundary information. We analyzed the tangent and curvature of the input points in the construction of original κ-Curves. We reformulated the construction of κ-Curves in a different splitting way and extend to a larger class of cubic curves. The new type of curve has all the κ-Curves shape properties like G^2 continuity, all local maximum curvature points only appear at input points, and the locality. Besides, the new kind of curve provide more direction control over the input points with local direction and sharpness information. The optimization can also

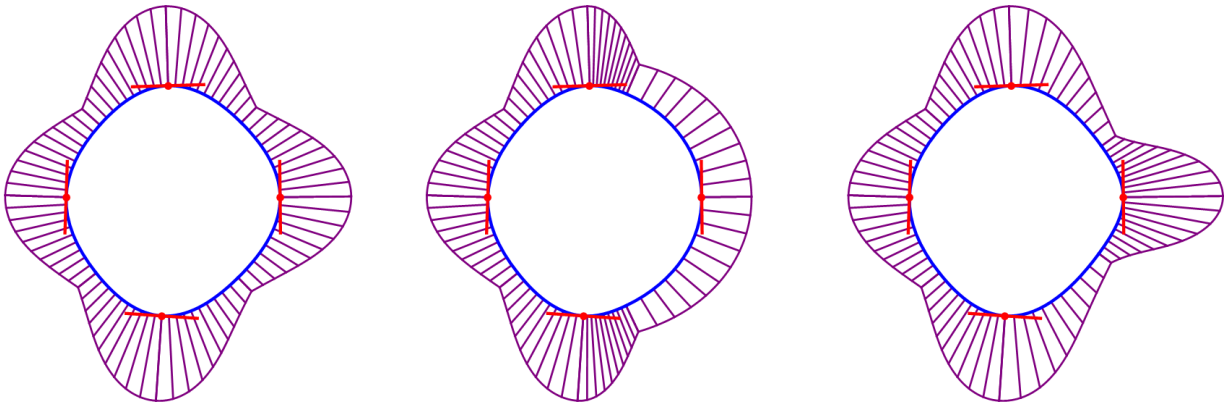

Figure 7.3: The three pictures have the same input points and all input points are assigned the same tangent directions. Compared to the left picture, the curvature of the right point in the middle and the right pictures are multiplied by 0.7 and 1.4.

converge in real time for curve modeling applications.

The main limitation is that tangent and curvature at the input point can not be arbitrary. We have a set of valid input but we don't know the range of the valid user input as the constraints. For any set of tangent and curvature, our method can compute the proposed curve to check if the curve has monotonic curvature but we can not predict if the input is valid without evaluating the optimization problem.

8. SUMMARY AND CONCLUSIONS

In this dissertation, we reviewed the state of the art of current curve modeling tools. We develop the features a good curve should have for geometric modeling and art design, including the geometric features of the whole curve and the constraints of the local shape. The main requirements are: interpolating input points as local curvature maxima, G^2 continuity and continuously moving along with the control points. Controlling the curvature maxima has rarely been researched though the salient feature can benefit designers. Our work enhanced the usage of intrinsic geometric feature of spline curves and filled the gap between spline curves and direct control of sharp features of curves in real applications.

We analyzed the monotonicity condition of polynomial quadratic, rational quadratic and polynomial cubic curves. And we developed a type of restricted cubic curves with the same degree of freedom of quadratic curve but a larger region of curvature monotonicity. We developed two frameworks to combine curves primitives in two different ways: one is to build a curve segment for each input point and connect the segments between the input points, another one is to construct a curve segment between each pair of two adjacent input points. The different frameworks provide more freedom and flexity of the curve design. Also we build up a fast solution system of our curve modeling that can serve as a real-time curve modeling tool in art design applications. Another advantage of our system is that we do not require any specific type of curve primitives. Any shape primitives with monotonic curvature distribution and the boundary information can be adopted in our system.

One limitation of our curve modeling framework is that we never dealt with zero curvature. The absolute value of the curvature at inflection points is continuous but the normal direction is opposite. Though this setting won't affect the visual continuity a lot, mathematically our curve can not cover arbitrary curvature in $\mathbb{R}$.

In this dissertation we combine the same types of curve primitives as a whole curve. Indeed, the G^2 continuity only require tangent and curvature information at end points. It is possible to

connect different types of curve primitives as a new curve modeling tool. For example, we can build a curve using a mixture of rational quadratic curves and cubic curves. We would leave these challenges as future work.

REFERENCES

[1] G. Deslauriers and S. Dubuc, "Symmetric iterative interpolation processes," *Constructive Approximation*, vol. 5, no. 1, pp. 49–68, 1989.

[2] E. Catmull and R. Rom, "A class of local interpolating splines," *Computer aided geometric design*, pp. 317–326, 1974.

[3] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann Publishers Inc., 5th ed., 2002.

[4] Z. Yan, S. Schiller, G. Wilensky, N. Carr, and S. Schaefer, "k-curves: interpolation at local maximum curvature," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 129, 2017.

[5] P. J. Barry and R. N. Goldman, "A recursive evaluation algorithm for a class of catmull-rom splines," *SIGGRAPH Comput. Graph.*, vol. 22, p. 199204, June 1988.

[6] G. Farin, "Class A Bézier curves," *Computer Aided Geometric Design*, vol. 23, no. 7, pp. 573–581, 2006.

[7] Y. Y. Feng and J. Kozak, "On G^2 continuous interpolatory composite quadratic Bézier curves," *Journal of Computational and Applied Mathematics*, vol. 72, no. 1, pp. 141–159, 1996.

[8] H.-J. Gu, J.-H. Yong, J.-C. Paul, and F. F. Cheng, "Constructing G^1 quadratic Bézier curves with arbitrary endpoint tangent vectors," *International Journal of CAD/CAM*, vol. 9, no. 1, 2009.

[9] S. Havemann, J. Edelsbrunner, P. Wagner, and D. Fellner, "Curvature-controlled curve editing using piecewise clothoid curves," *Computers & Graphics*, vol. 37, no. 6, pp. 764–773, 2013.

[10] R. Levien and C. H. Séquin, "Interpolating splines: Which is the fairest of them all?," *Computer-Aided Design and Applications*, vol. 6, no. 1, pp. 91–102, 2009.

[11] E. Corman, J. Solomon, M. Ben-Chen, L. Guibas, and M. Ovsjanikov, "Functional characterization of intrinsic and extrinsic geometry," *ACM Trans. Graph.*, vol. 36, Mar. 2017.

[12] Z. Yan, S. Schiller, and S. Schaefer, "Circle reproduction with interpolatory curves at local maximal curvature points," *Computer Aided Geometric Design*, vol. 72, pp. 98–110, 2019.

[13] G. Farin and N. Sapidis, "Curvature and the fairness of curves and surfaces," *IEEE Computer Graphics and Applications*, vol. 9, no. 2, pp. 52–57, 1989.

[14] M. P. Do Carmo, *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

[15] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, vol. 55. US Government printing office, 1948.

[16] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Ltd., 1993.

[17] N. Dyn, D. Levin, and J. A. Gregory, "A 4-point interpolatory subdivision scheme for curve design," *Computer aided geometric design*, vol. 4, no. 4, pp. 257–268, 1987.

[18] C. Yuksel, S. Schaefer, and J. Keyser, "Parameterization and applications of Catmull–Rom curves," *Computer-Aided Design*, vol. 43, no. 7, pp. 747–755, 2011.

[19] M. Higashi, K. Kaneko, and M. Hosaka, "Generation of high-quality curve and surface with smoothly varying curvature," in *EG Technical Papers*, Eurographics Association, 1988.

[20] Y. Mineur, T. Lichah, J. M. Castelain, and H. Giaume, "A shape controled fitting method for Bézier curves," *Computer Aided Geometric Design*, vol. 15, no. 9, pp. 879–891, 1998.

[21] R. Schneider and L. Kobbelt, "Discrete fairing of curves and surfaces based on linear curvature distribution," tech. rep., DTIC Document, 2000.

[22] J. McCrae and K. Singh, "Sketching piecewise clothoid curves," *Computers & Graphics*, vol. 33, no. 4, pp. 452–461, 2009.

[23] N. Yoshida, R. Fukuda, and T. Saito, "Log-aesthetic space curve segments," in *SIAM/ACM Conference on Geometric and Physical Modeling*, pp. 35–46, 2009.

[24] K. T. Miura, D. Shibuya, R. U. Gobithaasan, and S. Usuki, "Designing log-aesthetic splines with G^2 continuity," *Computer-Aided Design and Applications*, vol. 10, no. 6, pp. 1021–1032, 2013.

[25] K. T. Miura and R. Gobithaasan, "Aesthetic curves and surfaces in computer aided geometric design," *International Journal of Automation Technology*, vol. 8, no. 3, pp. 304–316, 2014.

[26] N. Yoshida and T. Saito, "Quadratic log-aesthetic curves," *Computer-Aided Design and Applications*, vol. 14, no. 2, pp. 219–226, 2017.

[27] R. Schaback, "Interpolation with piecewise quadratic visually C^2 Bézier polynomials," *Computer Aided Geometric Design*, vol. 6, no. 3, pp. 219–233, 1989.

[28] J. Hoschek, "Circular splines," *Computer-Aided Design*, vol. 24, no. 11, pp. 611–618, 1992.

[29] D. S. Meek and D. J. Walton, "Approximation of discrete data by g1 arc splines," *Computer-Aided Design*, vol. 24, no. 6, pp. 301–306, 1992.

[30] M. K. Yeung and D. J. Walton, "Curve fitting with arc splines for nc toolpath generation," *Computer-Aided Design*, vol. 26, no. 11, pp. 845–849, 1994.

[31] A. Kurnosenko, "Biarcs and bilens," *Computer Aided Geometric Design*, vol. 30, no. 3, pp. 310–330, 2013.

[32] D. Meek and D. Walton, "Planar G^2 Hermite interpolation with some fair, c-shaped curves," *Journal of Computational and Applied Mathematics*, vol. 139, no. 1, pp. 141–161, 2002.

[33] L. A. Piegl and W. Tiller, "Data approximation using biarcs," *Engineering with computers*, vol. 18, no. 1, pp. 59–65, 2002.

[34] H.-J. Wenz, "Interpolation of curve data by blended generalized circles," *Computer Aided Geometric Design*, vol. 13, no. 8, pp. 673–680, 1996.

[35] C. H. Séquin, K. Lee, and J. Yen, "Fair, G^2- and C^2- continuous circle splines for the interpolation of sparse data points," *Computer-Aided Design*, vol. 37, no. 2, pp. 201–211, 2005.

[36] C. Sun and H. Zhao, "Generating fair, C^2 continuous splines by blending conics," *Computers & Graphics*, vol. 33, no. 2, pp. 173–180, 2009.

[37] S. Schaefer, "A factored interpolatory subdivision scheme for surfaces of revolution," Master's thesis, Rice University, 1993.

[38] C. Xu, T.-w. Kim, and G. Farin, "The eccentricity of conic sections formulated as rational Bézier quadratics," *Computer Aided Geometric Design*, vol. 27, no. 6, pp. 458–460, 2010.

[39] A. Cantón, L. Fernández-Jambrina, and E. R. María, "Geometric characteristics of conics in Bézier form," *Computer-Aided Design*, vol. 43, no. 11, pp. 1413–1421, 2011.

[40] R. Schaback, "Planar curve interpolation by piecewise conics of arbitrary type," *Constructive Approximation*, vol. 9, no. 4, pp. 373–389, 1993.

[41] X. Yang, "Curve fitting and fairing using conic splines," *Computer-Aided Design*, vol. 36, no. 5, pp. 461–472, 2004.

[42] J. M. Lane and R. F. Riesenfeld, "Bounds on a polynomial," *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 112–117, 1981.

[43] C. Yuksel, "A class of c 2 interpolating splines," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 5, pp. 1–14, 2020.

[44] A. G. Akritas, "Sylvesters form of the resultant and the matrix-triangularization subresultant prs method," in *Computer Aided Proofs in Analysis*, pp. 5–11, Springer, 1991.

[45] A. B. Ayoub, "The eccentricity of a conic section," *The College Mathematics Journal*, vol. 34, no. 2, p. 116, 2003.

[46] J. M. Lane and R. F. Riesenfeld, "Bounds on a polynomial," *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 112–117, 1981.

[47] R. W. Nickalls, “A new approach to solving the cubic: Cardan’s solution revealed,” *The Mathematical Gazette*, vol. 77, no. 480, pp. 354–359, 1993.

APPENDIX A

ROOTS OF CUBIC EQUATIONS

A.1 Cardano's Formula for Single Root

Cardano's formula is the root of cubic equation with single real root without the quadratic term

$$x^3 + px + q = 0. \tag{A.1}$$

Let u and v be some real numbers satisfying

$$\begin{cases} p = -3uv \\ q = -u^3 - v^3 \end{cases}. \tag{A.2}$$

Then $x = u + v$ is a root of equation (A.1) because of the identity

$$(u+v)^3 - 3uv(v-u) - u^3 - v^3 \equiv 0.$$

u^3 and v^3 can be solved from a quadratic equation induced from equation (A.2). A set of solution is

$$\begin{cases} u^3 = -\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}} \\ v^3 = -\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}} \end{cases}.$$

The other solution is just a swapped order of u and v, which won't affect $u + v$. So if $\frac{q^2}{4} + \frac{p^3}{27} > 0$, equation (A.1) has a single real root

$$x = \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

or (A.1) has three real roots.

A.2 Three Real Roots

When $\frac{q^2}{4} + \frac{p^3}{27} \leq 0$, equation (A.1) has three real roots. In this case p is negative and we use the results from Nickalls [47]. All the three real roots are

$$x_i = 2\sqrt{\frac{-p}{3}} \cos\left(\frac{1}{3}\arccos\left(-\frac{3q}{2}\sqrt{\frac{3}{-p^3}}\right) + \frac{2\pi}{3}i\right),\ i = 1, 2, 3.$$

A.3 General Cubic Equations

A general cubic equation with non-zero leading coefficient

$$ax^3 + bx^2 + cx + d = 0,\ a \neq 0$$

can always been transformed to equation (A.1) by substitution

$$x = \hat{x} - \frac{b}{3a}.$$

Then we have the new equation about $\hat{x}$

$$\hat{x}^3 + (\frac{c}{a} - \frac{b^2}{3a^2})\hat{x} + (\frac{d}{a} - \frac{bc}{3a^2} + \frac{2b^3}{27a^3}) = 0.$$

Solve $\hat{x}$ by last two sections and then insert back to the expression of x.

APPENDIX B

DERIVATION OF MIN ECCENTRICITY WEIGHT

To make the notation simpler, denote $\{Q_0, Q_1, Q_2\}$ as the quadratic control polygon and w is the weight of the middle point Q_1. Define $Q_i = (x_i, y_i)$ on a 2d plane, then the rational quadratic curve is

$$r(t) = \left(\frac{(1-t)^2 x_0 + 2(1-t)twx_1 + t^2 x_2}{(1-t)^2 + 2(1-t)tw + t^2}, \frac{(1-t)^2 y_0 + 2(1-t)twy_1 + t^2 y_2}{(1-t)^2 + 2(1-t)tw + t^2} \right).$$

Denote $X(t)$ and $Y(t)$ as the two coordinates of $r(t)$, we have

$$\begin{cases} ((1-t)^2 + 2(1-t)tw + t^2)X - ((1-t)^2 x_0 + 2(1-t)twx_1 + t^2 x_2) = 0 \\ ((1-t)^2 + 2(1-t)tw + t^2)Y - ((1-t)^2 y_0 + 2(1-t)twy_1 + t^2 y_2) = 0 \end{cases}. \tag{B.1}$$

We use the Sylvester resultant [44] of the two polynomials in equation (B.1) to eliminate variable t and the result is

$$AX^2 + BXY + CY^2 + DX + EY + F = 0,$$

where

$$\begin{cases} A = (y_0 - y_2)^2 + 4w^2(y_0 - y_1)(y_2 - y_1) \\ B = -2(x_0 - x_2)(y_0 - y_2) - 4w^2((x_0 - x_1)(y_2 - y_1) + (x_2 - x_1)(y_0 - y_1)) \\ C = (x_0 - x_2)^2 + 4w^2(x_0 - x_1)(x_2 - x_1) \\ D = 2(y_0 - y_2)(x_0 y_2 - x_2 y_0) + 4w^2((x_0 y_1 - x_1 y_0)(y_2 - y_1) + (x_2 y_1 - x_1 y_2)(y_0 - y_1)) \\ E = 2(x_0 - x_2)(y_0 x_2 - y_2 x_0) + 4w^2((y_0 x_1 - y_1 x_0)(x_2 - x_1) + (y_2 x_1 - y_1 x_2)(x_0 - x_1)) \\ F = (x_2 y_0 - x_0 y_2)^2 + 4w^2(x_1 y_0 - x_0 y_1)(x_1 y_2 - x_2 y_1) \end{cases}.$$

Then we try to find the minimizer of the square of eccentricity according to weight w

$$\epsilon^2 = \frac{2\sqrt{(A-C)^2+B^2}}{A+C+\sqrt{(A-C)^2+B^2}}.$$

If we compute the derivative of ϵ^2 to w, then the numerator of

$$\frac{\partial \epsilon^2}{\partial w}$$

can be factored as multiple terms. The term related to the variable w is

$$2w^2((x_0-x_1)^2+(x_2-x_1)^2+(y_0-y_1)^2+(y_2-y_1)^2)-(x_0-x_2)^2-(y_0-y_2)^2.$$

So the weight can minimize the eccentricity of the quadratic curve is the root of the equation above

$$w = \frac{\sqrt{(x_0-x_2)^2+(y_0-y_2)^2}}{\sqrt{2}\sqrt{(x_0-x_1)^2+(x_2-x_1)^2+(y_0-y_1)^2+(y_2-y_1)^2}} = \frac{|Q_0-Q_2|}{\sqrt{2}\sqrt{|Q_0-Q_1|^2+|Q_2-Q_1|^2}}.$$

Notice this weight is always smaller than one so the minimized eccentricity shape is always a circle or ellipse.