

Quadratic Approximation of Cubic Curves

NGHIA TRUONG, University of Utah
CEM YUKSEL, University of Utah
LARRY SEILER, Facebook Reality Labs



Fig. 1. Example hair models represented using (a,c) piecewise cubic curves and (b,d) their piecewise quadratic approximations, rendered using path tracing with 256 samples per pixel, producing virtually indistinguishable results. Depending on the renderer and settings, the quadratic versions can be rendered more than $3\times$ faster or with only a minor speed improvement.

We present a simple degree reduction technique for piecewise cubic polynomial splines, converting them into piecewise quadratic splines that maintain the parameterization and C^1 continuity. Our method forms identical tangent directions at the interpolated data points of the piecewise cubic spline by replacing each cubic piece with a pair of quadratic pieces. The resulting representation can lead to substantial performance improvements for rendering geometrically complex spline models like hair and fiber-level cloth. Such models are typically represented using cubic splines that are C^1 -continuous, a property that is preserved with our degree reduction. Therefore, our method can also be considered a new quadratic curve construction approach for high-performance rendering. We prove that it is possible to construct a pair of quadratic curves with C^1 continuity that passes through any desired point on the input cubic curve. Moreover, we prove that when the pair of quadratic pieces corresponding to a cubic piece have equal parametric lengths, they join exactly at the parametric center of the cubic piece, and the deviation in positions due to degree reduction is minimized.

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

Additional Key Words and Phrases: polynomial splines, cubic splines, quadratic splines, Bézier curves, degree reduction, hair rendering, ray tracing

ACM Reference Format:

Nghia Truong, Cem Yuksel, and Larry Seiler. 2020. Quadratic Approximation of Cubic Curves. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2, Article 16 (August 2020), 17 pages. <https://doi.org/10.1145/3406178>

Authors' addresses: Nghia Truong, University of Utah, nghiatruong.vn@gmail.com; Cem Yuksel, University of Utah, cem@cemyuksel.com; Larry Seiler, Facebook Reality Labs, larryseiler@fb.com.

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, <https://doi.org/10.1145/3406178>.

1 INTRODUCTION

Parametric cubic polynomial splines are by far the most popular curve representation in computer graphics. This can be attributed to the fact that they are the lowest degree polynomials that can form 3D curve segments, since quadratic curves are planar (i.e. can be defined by three points in space). Cubic curves are not only used for various modeling and animation tasks, but also for representing geometrically-complex models like hair, fur, grass, and cloth fibers.

For rendering, splines can be subdivided into many small segments that are approximated with linear pieces either for rasterization or ray tracing. More recently, it has been shown that it is possible to efficiently compute ray intersections with cubic splines with some thickness [Reshetov 2017]. Both of these approaches involve repeated evaluation of the underlying polynomial function, the cost of which can be substantial, particularly for ray tracing geometrically-complex models.

In this paper, we show that it is possible to use piecewise quadratic curves to closely approximate cubic ones, providing a degree reduction in the polynomial formulation. This can result in performance improvement when the curves are repeatedly evaluated. Our experiments with different ray tracing applications show improvements in render times up to 38% using our method, though the savings can vary significantly and are not always measurable (Figure 1).

Our degree reduction method is designed to maintain C^1 continuity. This is an important property, because most existing applications use C^1 -continuous cubic curves, particularly for representing geometrically-complex models. As such, our degree reduction method preserves the important theoretical properties of the original cubic representation. In that respect, our method can also be considered an alternative curve formulation, rather than an approximation technique. Indeed, existing popular cubic curve formulations, such as Catmull-Rom curves [Catmull and Rom 1974a; Yuksel et al. 2009b, 2011] and B-splines [Bartels et al. 1987], can be augmented with our degree reduction step to form new curve construction techniques that produce quadratic curves with similar properties as their cubic counterparts. Considering the performance advantages of quadratic polynomials over cubics, we would argue that there is no good reason for defining geometrically-complex objects like hair and fiber-level cloth models using existing piecewise cubic formulations. Instead, such models can be defined with piecewise quadratic curves using our approach.

The core of our degree reduction technique is representing each cubic curve segment with a pair of quadratic curves that maintain C^1 continuity. Even though both quadratic curves form perfectly planar segments, we show that the resulting curve that joins the two quadratic pieces closely approximate the 3D shape of the cubic segment.

One of our technical contributions is the discovery that enforcing C^1 continuity for approximating a cubic segment with a pair of quadratic curves guarantees that the resulting piecewise quadratic curve passes through a point on the cubic curve segment, in addition to the two end points. This interesting property is an important factor in achieving a close geometric approximation.

In addition, we show that the piecewise quadratic curve can be generated such that it passes through any desired point along the cubic segment. The approximation error is minimized when the piecewise quadratic curve passes through the parametric center of the cubic segment, in which case the two quadratic pieces join exactly at this parametric center point.

We also show that in 3D (and higher dimensions) the parametric C^1 continuity coincides with the geometric G^1 continuity, i.e. the alternative of enforcing only G^1 continuity results in identical quadratic curves. In the special case of 2D cubic curves, however, approximating with G^1 continuity offers infinitely many solutions, while C^1 continuity has one.

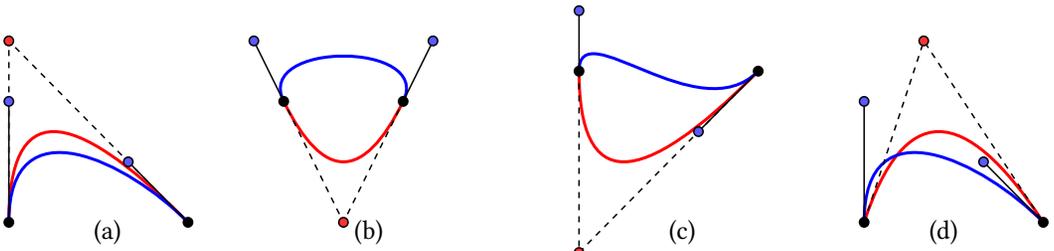


Fig. 2. The placement of the middle control point for quadratic Bézier curves (shown in red) that approximate cubic Bézier curves (shown in blue) in 2D: (a) using the intersection of the tangent lines at the end points is the only option for the middle control point that matches both tangents at the end points, but (b-c) it can also invert the derivative directions at the end points; (d) picking a different point fails to match the tangents at both end points.

2 RELATED WORK

Polynomial splines are commonplace in computer aided design and computer graphics. In particular, cubic Bézier curves [Bézier 1977] are included in most 2D and 3D applications that have curve editing properties. Most applications allow the user to directly specify the Bézier control points for automatically producing a G^1 -continuous curve. C^1 -continuity, however, is typically achieved by generating a piecewise cubic curve that interpolates a given set of data points, such as Catmull-Rom curves [Catmull and Rom 1974b]. The extension of Barry and Goldman [1988] allows arbitrary parameterizations [Foley et al. 1989; Lee 1989; Manocha and Canny 1992; Nielson and Foley 1989] and among them centripetal parameterization was shown to inherit unique properties, such as guaranteed cusp-free and self-intersection-free interpolation [Yuksel et al. 2009b, 2011].

2.1 Degree Reduction

In 2D, arguably the simplest way of approximating a piecewise cubic C^1 curve using a piecewise quadratic curve is converting each cubic piece to a quadratic piece: for each cubic piece with end points \mathbf{p}_i and \mathbf{p}_{i+1} , we generate a quadratic piece with the same end points (Figure 2). Placing the middle control point of the quadratic piece at the intersection of the two lines that are tangent to the cubic curve at \mathbf{p}_i and \mathbf{p}_{i+1} [Alfke 1994; Bily 2014; Groleau 2002], leads to a piecewise quadratic curve with G^1 continuity (Figure 2a). Yet, this simple solution can also invert some segments (Figure 2b-c), which breaks G^1 continuity. Other positions for the middle control points may provide a closer approximation to the cubic piece [Colomitchi 2006; Sutcliffe 2007], but lead to quadratic curves with only C^0 (or G^0) continuity (Figure 2d). The approximation error with any of these approaches can be reduced (and inversions can be eliminated) by first subdividing the cubic curve into smaller pieces. Alternatively, a similar approach can be used for approximating the whole cubic curve (instead of considering each cubic piece separately) by iteratively adding quadratic pieces as needed to satisfy a given error tolerance, forming a quadratic curve with G^1 continuity [Cox and Harris 1991]. Note that none of these approaches work in 3D or higher dimensions and they cannot preserve C^1 continuity.

Piecewise cubic curves can also be C^2 -continuous [Farin 2006; Higashi et al. 1988; Mineur et al. 1998]. However, achieving C^2 continuity with a piecewise quadratic curve can only happen in special cases. The most we can expect from a piecewise quadratic curve is G^2 continuity. Indeed, κ -curves [Yan et al. 2017] can produce a mostly G^2 -continuous piecewise quadratic curves that interpolate a given set of data points in 2D. Therefore, it is possible to use κ -curves for approximating any piecewise cubic curve by arbitrarily choosing data points on the piecewise cubic curve for the

κ -curve to interpolate. However, this approach only works in 2D and would not necessarily match the derivatives of the piecewise cubic curve at the chosen data points.

Piecewise cubic and quadratic curves can also be approximated using other formulations, such as circular pieces [Riškus 2006; Riškus and Liutkus 2013].

2.2 Curve Rendering

Hair and fur rendering is one of the major applications of curve rendering. In ray tracing, the ray-curve intersection test is a crucial component. Most applications approximate curves using small pieces of line segments or cylinders [Barringer et al. 2012; Han et al. 2019; Nakamaru and Ohno 2002; Qin et al. 2014; Woop et al. 2014]. Ray-curve intersection test is therefore simplified down to testing ray-line or ray-cylinder intersections. Recently, the support of cubic Bézier curve primitive has been introduced in ray tracing by Embree [Wald et al. 2014] and PBRT [Pharr et al. 2016]. For ray-curve intersection test, the ray is instead tested for intersection with a bounding object of the curve primitive. The bounding object (axis-aligned bounding box [Pharr et al. 2016] or bounding cylinder [Wald et al. 2014]) is computed to enclose the curve segment and iteratively refined until a desired accuracy is met. Such iterative refinement can include curve subdivision [Pharr et al. 2016], which is expensive. Phantom ray-curve intersector [Reshetov 2017; Reshetov and Luebke 2018] is an alternative method that finds the ray intersection with the surface of a thick curve (i.e. a tapered tube that is bent to take the shape of the curve). The intersection test is performed by a ray-swept volume intersection approach while curve subdivision is totally avoided.

In rasterization, there is a large body of work in hair, fur, and fiber-level cloth rendering [Andersen et al. 2016; Barringer et al. 2012; Wu and Yuksel 2017a,b]. Typically, rasterizing curve primitives produces better performance than ray tracing as it performs inexpensive curve evaluations during tessellation and does not require costly iterative intersection test.

3 DEGREE REDUCTION WITH C^1 CONTINUITY

Given a C^1 -continuous piecewise cubic curve $C(t)$ that interpolates a set of data points \mathbf{p}_i , such that $C(t_i) = \mathbf{p}_i$ for a set of parameter values t_i , our goal is to approximate it using a C^1 -continuous piecewise quadratic curve $Q(t)$, such that the new curve interpolates the same data points at exactly the same parametric positions (i.e. $Q(t_i) = \mathbf{p}_i$) and matches the derivatives of the cubic curve at these data points, such that $C'(t_i) = Q'(t_i)$, where $C'(t) = dC(t)/dt$ and $Q'(t) = dQ(t)/dt$ denote the derivatives.

Let $C_i(t)$ represent each piece i of the input cubic curve, such that $C_i(t)$ interpolates data points \mathbf{p}_i and \mathbf{p}_{i+1} for $t \in [t_i, t_{i+1}]$. Thus, $C_i(t_i) = \mathbf{p}_i$ and $C_i(t_{i+1}) = \mathbf{p}_{i+1} = C_{i+1}(t_{i+1})$. We use $C'_i(t) = dC_i(t)/dt$ to represent the derivative of the curve. Since the piecewise cubic curve has C^1 continuity, $C'_i(t_{i+1}) = C'_{i+1}(t_{i+1})$. We form $Q(t)$ by splitting each cubic piece $C_i(t)$ into two quadratic pieces $Q_{2i}(t)$ and $Q_{2i+1}(t)$, as shown in Figure 3. The first quadratic piece matches the beginning of the cubic piece and the second one matches the end of it, meaning

$$Q_{2i}(t_i) = C_i(t_i) \tag{1}$$

$$Q'_{2i}(t_i) = C'_i(t_i) \tag{2}$$

$$Q_{2i+1}(t_{i+1}) = C_i(t_{i+1}) \tag{3}$$

$$Q'_{2i+1}(t_{i+1}) = C'_i(t_{i+1}) . \tag{4}$$

The two quadratic pieces $Q_{2i}(t)$ and $Q_{2i+1}(t)$ join at $t_i^* \in (t_i, t_{i+1})$ with C^1 continuity, so we enforce

$$Q_{2i}(t_i^*) = Q_{2i+1}(t_i^*) \tag{5}$$

$$Q'_{2i}(t_i^*) = Q'_{2i+1}(t_i^*) . \tag{6}$$

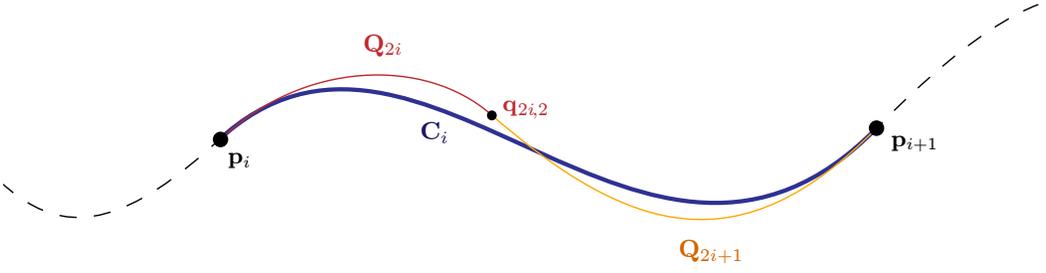


Fig. 3. C^1 -continuous approximation of a cubic piece C_i with two quadratic pieces Q_{2i} and Q_{2i+1} .

Thus, the piecewise quadratic curve is given by $Q_{2i}(t)$ for $t \in [t_i, t_i^*]$ and $Q_{2i+1}(t)$ for $t \in [t_i^*, t_{i+1}]$. We use $\gamma \in (0, 1)$ to define the parametric position of the splitting point, such that $t_i^* = (1 - \gamma)t_i + \gamma t_{i+1}$. Note that $\gamma = 0$ and $\gamma = 1$ are not permitted, since they would eliminate one of the curves and thereby break C^1 continuity in our formulation.

To formulate the corresponding quadratic representation of a cubic curve, let us represent the input cubic curve using Bézier control points $\mathbf{b}_{i,0}$, $\mathbf{b}_{i,1}$, $\mathbf{b}_{i,2}$, and $\mathbf{b}_{i,3}$, as

$$C_i(t) = (1 - s)^3 \mathbf{b}_{i,0} + 3(1 - s)^2 s \mathbf{b}_{i,1} + 3(1 - s) s^2 \mathbf{b}_{i,2} + s^3 \mathbf{b}_{i,3}, \quad (7)$$

where $s \in [0, 1]$ is the local parameter, such that $s = (t - t_i)/(t_{i+1} - t_i)$, $\mathbf{b}_{i,0} = \mathbf{p}_i$, and $\mathbf{b}_{i,3} = \mathbf{p}_{i+1}$. We can write the two quadratic curves that interpolate the same end points as

$$Q_{2i}(t) = (1 - s_0)^2 \mathbf{q}_{2i,0} + 2(1 - s_0)s_0 \mathbf{q}_{2i,1} + s_0^2 \mathbf{q}_{2i,2} \quad (8)$$

$$Q_{2i+1}(t) = (1 - s_1)^2 \mathbf{q}_{2i+1,0} + 2(1 - s_1)s_1 \mathbf{q}_{2i+1,1} + s_1^2 \mathbf{q}_{2i+1,2}, \quad (9)$$

where $s_0, s_1 \in [0, 1]$ are the local parameters, such that $s_0 = s/\gamma$ and $s_1 = (s - \gamma)/(1 - \gamma)$. Due to C^0 continuity, the first and the last control points are $\mathbf{q}_{2i,0} = \mathbf{b}_{i,0}$ and $\mathbf{q}_{2i+1,2} = \mathbf{b}_{i,3}$, and the two quadratic curves have a shared control point $\mathbf{q}_{2i,2} = \mathbf{q}_{2i+1,0}$. The positions of the interior control points are defined by C^1 continuity. The derivatives of the curves can be written as

$$C'_i(t) = \frac{ds}{dt} (3(1 - s)^2 (\mathbf{b}_{i,1} - \mathbf{b}_{i,0}) + 6(1 - s)s (\mathbf{b}_{i,2} - \mathbf{b}_{i,1}) + 3s^2 (\mathbf{b}_{i,3} - \mathbf{b}_{i,2})) \quad (10)$$

$$Q'_{2i}(t) = \frac{ds}{dt} \left(\frac{1}{\gamma} \right) (2(1 - s_0) (\mathbf{q}_{2i,1} - \mathbf{q}_{2i,0}) + 2s_0 (\mathbf{q}_{2i,2} - \mathbf{q}_{2i,1})) \quad (11)$$

$$Q'_{2i+1}(t) = \frac{ds}{dt} \left(\frac{1}{1 - \gamma} \right) (2(1 - s_1) (\mathbf{q}_{2i+1,1} - \mathbf{q}_{2i+1,0}) + 2s_1 (\mathbf{q}_{2i+1,2} - \mathbf{q}_{2i+1,1})), \quad (12)$$

where $ds/dt = 1/(t_{i+1} - t_i)$ is a constant term, accounting for the relative speeds of the two parameters s and t . Matching the derivatives $C'_i(t_i) = Q'_{2i}(t_i)$ at $s = 0$ and $C'_i(t_{i+1}) = Q'_{2i+1}(t_{i+1})$ at $s = 1$ gives

$$\mathbf{q}_{2i,1} = \mathbf{b}_{i,0} + \frac{3}{2}\gamma(\mathbf{b}_{i,1} - \mathbf{b}_{i,0}) \quad (13)$$

$$\mathbf{q}_{2i+1,1} = \mathbf{b}_{i,3} + \frac{3}{2}(1 - \gamma)(\mathbf{b}_{i,2} - \mathbf{b}_{i,3}). \quad (14)$$

The shared control point is determined by $Q'_{2i}(t_i^*) = Q'_{2i+1}(t_i^*)$ at $s_0 = 1$ and $s_1 = 0$ due to C^1 continuity of the quadratic curve, resulting

$$\mathbf{q}_{2i,2} = \mathbf{q}_{2i+1,0} = (1 - \gamma) \mathbf{q}_{2i,1} + \gamma \mathbf{q}_{2i+1,1}. \quad (15)$$

Thus, we can define a pair of quadratic curves for each cubic curve piece, interpolating the data points and matching the derivatives at the data points.

Since all control points of the quadratic curves are uniquely defined, for a given γ , there exists a *unique* piecewise quadratic curve that approximates each cubic piece of the given piecewise cubic curve using a pair of quadratic pieces with $\mathbf{Q}_{2i}(t_i) = \mathbf{C}_i(t_i)$ and $\mathbf{Q}_{2i+1}(t_{i+1}) = \mathbf{C}_i(t_{i+1})$, satisfying the C^1 continuity conditions $\mathbf{Q}'_{2i}(t_i) = \mathbf{C}'_i(t_i)$, $\mathbf{Q}'_{2i+1}(t_{i+1}) = \mathbf{C}'_i(t_{i+1})$, and $\mathbf{Q}'_{2i}(t_i^*) = \mathbf{Q}'_{2i+1}(t_i^*)$.

THEOREM 1. *For the special case of $\gamma = \frac{1}{2}$, the two quadratic curves approximating a cubic curve with C^1 continuity always join at the parametric center of the cubic curve. Thus, $\mathbf{C}_i(t_i^*) = \mathbf{Q}_{2i}(t_i^*) = \mathbf{Q}_{2i+1}(t_i^*)$.*

PROOF. Using Equations 13, 14, and 15, we can write the point where the two quadratic curves join as

$$\mathbf{Q}_{2i}(t_i^*) = \mathbf{Q}_{2i+1}(t_i^*) = (1 - \gamma) \left(1 - \frac{3}{2}\gamma\right) \mathbf{b}_{i,0} + \frac{3}{2}(1 - \gamma)\gamma (\mathbf{b}_{i,1} + \mathbf{b}_{i,2}) + \left(1 - \frac{3}{2}(1 - \gamma)\right) \gamma \mathbf{b}_{i,3}. \quad (16)$$

The position of the cubic curve at the splitting parameter $\mathbf{C}_i(t_i^*)$ corresponds to $s = \gamma$ in Equation 7. Therefore, to guarantee $\mathbf{C}_i(t_i^*) = \mathbf{Q}_{2i}(t_i^*)$, each term of Equation 7 must match the corresponding terms of Equation 16. Thus, γ must satisfy

$$(1 - \gamma)^3 = (1 - \gamma) \left(1 - \frac{3}{2}\gamma\right) \quad (17)$$

$$3(1 - \gamma)^2\gamma = \frac{3}{2}(1 - \gamma)\gamma \quad (18)$$

$$3(1 - \gamma)\gamma^2 = \frac{3}{2}(1 - \gamma)\gamma \quad (19)$$

$$\gamma^3 = \left(1 - \frac{3}{2}(1 - \gamma)\right) \gamma. \quad (20)$$

All of these equations are satisfied by only $\gamma = \frac{1}{2}$ for $\gamma \in (0, 1)$. Therefore, in this special case, the parametric center of the curve at $t = (t_i + t_{i+1})/2$ is $\mathbf{C}_i(t_i^*)$ and it always coincides with the point where the two quadratic curves join. \square

THEOREM 2. *When $\gamma \in (\frac{1}{3}, \frac{2}{3})$, each piece $\mathbf{C}_i(t)$ of a piecewise cubic curve is guaranteed to coincide with its C^1 -continuous quadratic approximation $\mathbf{Q}(t)$ at three parameter values: its two end points and one more point between them, such that $\exists \check{t}_i \in (t_i, t_{i+1})$ where $\mathbf{C}_i(\check{t}_i) = \mathbf{Q}(\check{t}_i)$.*

PROOF. The two intersections at the end points are guaranteed by construction. Let $\check{s} = (\check{t}_i - t_i)/(t_{i+1} - t_i)$ be the normalized parameter value where the cubic piece \mathbf{C}_i coincides with one of the quadratic pieces \mathbf{Q}_{2i} or \mathbf{Q}_{2i+1} , such that $\check{s} \in (0, 1)$. If $\check{t}_i \in (0, t_i^*]$ and thereby $\check{s} \in (0, \gamma]$, we can write $\mathbf{C}_i(\check{t}_i) = \mathbf{Q}_{2i}(\check{t}_i)$, which simplifies (using Equations 13, 14, and 15) to

$$\check{s} = \frac{3\gamma - 1}{2\gamma}. \quad (21)$$

Thus, \check{s} remains within range $(0, \gamma]$ for $\gamma \in (\frac{1}{3}, \frac{1}{2}]$. Similarly, if $\check{t}_i \in [t_i^*, t_{i+1})$ and $\check{s} \in [\gamma, 1)$, we can write $\mathbf{C}_i(\check{t}_i) = \mathbf{Q}_{2i+1}(\check{t}_i)$, which simplifies to

$$\check{s} = \frac{\gamma}{2 - 2\gamma}. \quad (22)$$

In this case, \check{s} remains within the valid range $[\gamma, 1)$ for $\gamma \in [\frac{1}{2}, \frac{2}{3})$. Therefore, the intersection between the end points has a valid solution if $\gamma \in (\frac{1}{3}, \frac{2}{3})$, regardless of the control point positions. \square

Note that for any value of $\gamma \notin (\frac{1}{3}, \frac{2}{3})$ there is no guarantee that the two quadratic curves would intersect with the cubic curve at any point other than the two end points of the cubic piece. In the special case where $\gamma = \frac{1}{3}$, we have $\mathbf{q}_{2i+1,1} = \mathbf{b}_{i,2}$ and with $\gamma = \frac{2}{3}$ we have $\mathbf{q}_{2i,1} = \mathbf{b}_{i,1}$.

In our formulation $\mathbf{Q}(t)$ always coincides with $\mathbf{C}(t)$ at the interpolated data points \mathbf{p}_i by construction. In addition, it might be desirable for $\mathbf{Q}(t)$ to coincide with $\mathbf{C}(t)$ at some other points along the curves as well, such that $\mathbf{C}(\check{t}) = \mathbf{Q}(\check{t})$ for some \check{t} parameter value. Our formulation allows this by simply adjusting the γ value accordingly.

THEOREM 3. *Given a cubic curve piece $\mathbf{C}_i(t)$ and a point \mathbf{x}_i such that $\mathbf{x}_i = \mathbf{C}_i(\check{t}_i)$ with $\check{t}_i \in (t_i, t_{i+1})$, there exists a C^1 -continuous approximation $\mathbf{Q}(t)$ that matches the positions and derivatives at the ends points and passes through \mathbf{x}_i at the same parameter value, i.e. $\mathbf{x}_i = \mathbf{Q}(\check{t}_i)$.*

PROOF. Let $\check{s} = (\check{t}_i - t_i)/(t_{i+1} - t_i)$ be the normalized parameter value at \mathbf{x}_i . Combining Equations 21 and 22 using their valid ranges, we can solve for the γ value that would produce $\mathbf{x}_i = \mathbf{Q}(\check{t}_i)$, such that

$$\gamma = \begin{cases} 1/(3 - 2\check{s}) & \text{if } 0 < \check{s} \leq \frac{1}{2} \\ 2\check{s}/(1 + 2\check{s}) & \text{if } \frac{1}{2} < \check{s} < 1 \end{cases} . \quad (23)$$

For all valid parameter values $\check{s} \in (0, 1)$, γ remains within the valid range $(\frac{1}{3}, \frac{2}{3})$. Thus, $\mathbf{C}(\check{t}_i) = \mathbf{Q}(\check{t}_i)$ can be satisfied for any $\check{t}_i \in (t_i, t_{i+1})$. \square

4 ERROR ANALYSIS

We define the error function $\epsilon(t)$ for our degree reduction as the distance between two points on the two curves at the same parametric position t , such that

$$\epsilon(t) = \|\mathbf{C}(t) - \mathbf{Q}(t)\| . \quad (24)$$

Note that this error function bounds the distance between a point on $\mathbf{Q}(t)$ to the closest point on \mathbf{C} .

$$\min_{t'} \|\mathbf{C}(t') - \mathbf{Q}(t)\| \leq \epsilon(t) . \quad (25)$$

THEOREM 4. *When each piece of a piecewise cubic curve is converted to two quadratic pieces with C^1 continuity, the maximum error due to degree reduction is minimized using pairs of quadratic pieces with parametrically equal lengths.*

PROOF. Let $\epsilon_i(t)$ represent the error function within the parametric range $t \in [t_i, t_{i+1}]$, such that

$$\epsilon_i(t) = \begin{cases} \|\mathbf{C}_i(t) - \mathbf{Q}_{2i}(t)\| & \text{if } t_i \leq t < t_i^* \\ \|\mathbf{C}_i(t) - \mathbf{Q}_{2i+1}(t)\| & \text{if } t_i^* \leq t < t_{i+1} \end{cases} . \quad (26)$$

Using a polynomial representation of $\mathbf{C}_i(t)$ with a normalized parameter $s = (t - t_i)/(t_{i+1} - t_i)$,

$$\mathbf{C}_i(t) = s^3 \mathbf{a}_{i,3} + s^2 \mathbf{a}_{i,2} + s \mathbf{a}_{i,1} + \mathbf{a}_{i,0} , \quad (27)$$

the error function simplifies down to

$$\epsilon_i(t) = \begin{cases} \left\| \mathbf{a}_{i,3} s^2 \left(s + \frac{1}{2\gamma} - \frac{3}{2} \right) \right\| & \text{if } 0 \leq s < \gamma \\ \left\| \mathbf{a}_{i,3} (1-s)^2 \left(s - \frac{\gamma}{2(1-\gamma)} \right) \right\| & \text{if } \gamma \leq s < 1 \end{cases} . \quad (28)$$

Thus, the maximum error $\epsilon_i(t)$ within range $t \in [t_i, t_{i+1}]$ is minimized when $\gamma = \frac{1}{2}$, which forms two quadratic curves with equal parametric lengths. \square

The maximum error with $\gamma = \frac{1}{2}$ appears at $s = \frac{1}{3}$ and $s = \frac{2}{3}$, resulting

$$\max_{t \in [t_i, t_{i+1}], \gamma = \frac{1}{2}} \epsilon_i(t) = \frac{1}{54} \|\mathbf{a}_{i,3}\| . \quad (29)$$

Of course, it is always possible to reduce the magnitude of the error by simply pre-subdividing piecewise cubic curve into more cubic pieces prior to degree reduction. Note that $\mathbf{a}_{i,3}$ can be written in terms of the Bézier control points as

$$\mathbf{a}_{i,3} = -\mathbf{b}_{i,0} + 3\mathbf{b}_{i,1} - 3\mathbf{b}_{i,2} + \mathbf{b}_{i,3} . \quad (30)$$

5 DEGREE REDUCTION WITH G^1 CONTINUITY

In our formulations above we preserve the parametric continuity (i.e. C^1) of the given piecewise cubic curve. For some applications, C^1 continuity may not be necessary and geometric continuity (i.e. G^1) might be sufficient. In this section, we describe a similar degree reduction formulation that only enforces G^1 continuity and show that it leads to identical quadratic curves as the ones with C^1 continuity and $\gamma = \frac{1}{2}$ under some conditions (when the piecewise quadratic curve passes through the parametric center of each cubic piece and the cubic curve is not planar).

Again, we approximate each cubic curve piece with two quadratics. For enforcing G^1 continuity, the directions of the derivatives at the end points must be maintained, but not their magnitudes. Thus, we can write

$$\mathbf{Q}'_{2i}(t_i) = \beta_{2i} \mathbf{C}'_i(t_i) \quad (31)$$

$$\mathbf{Q}'_{2i+1}(t_{i+1}) = \beta_{2i+1} \mathbf{C}'_i(t_{i+1}) , \quad (32)$$

where β_{2i} and β_{2i+1} are some arbitrary positive scaling factors. This means that the internal control points $\mathbf{q}_{2i,1}$ and $\mathbf{q}_{2i+1,1}$ can be placed anywhere along the rays $\overrightarrow{\mathbf{b}_{i,0}\mathbf{b}_{i,1}}$ and $\overrightarrow{\mathbf{b}_{i,3}\mathbf{b}_{i,2}}$, respectively. More precisely, by arbitrarily assigning equal parametric lengths to both quadratic pieces, we can write

$$\mathbf{q}_{2i,1} = \mathbf{b}_{i,0} + \frac{3}{4}\beta_{2i}(\mathbf{b}_{i,1} - \mathbf{b}_{i,0}) \quad (33)$$

$$\mathbf{q}_{2i+1,1} = \mathbf{b}_{i,3} + \frac{3}{4}\beta_{2i+1}(\mathbf{b}_{i,2} - \mathbf{b}_{i,3}) . \quad (34)$$

We must also enforce G^1 continuity at the shared control point of the two quadratic pieces where they join. This means that the shared point $\mathbf{q}_{2i,2} = \mathbf{q}_{2i+1,0}$ must be on the line segment connecting $\mathbf{q}_{2i,1}$ and $\mathbf{q}_{2i+1,1}$. Thus, using an arbitrary value $\sigma_i \in (0, 1)$, we can write

$$\mathbf{q}_{2i,2} = \mathbf{q}_{2i+1,0} = (1 - \sigma_i) \mathbf{q}_{2i,1} + \sigma_i \mathbf{q}_{2i+1,1} . \quad (35)$$

The conditions above are sufficient for maintaining G^1 continuity, but they do not include any constraint for approximating the shape of the cubic piece. To form a better approximation, we add the constraint that the resulting piecewise quadratic curve passes through the parametric centers of the cubic curve pieces.

THEOREM 5. *If quadratic approximation of a cubic curve piece $\mathbf{C}_i(t)$ with G^1 continuity passes through its parametric center $\mathbf{C}_i((t_i+t_{i+1})/2)$ and $\mathbf{C}_i(t)$ is not planar, the resulting quadratic curves are identical to the ones with C^1 continuity and $\gamma = \frac{1}{2}$.*

PROOF. Either the first or the second quadratic piece will go through the center of the cubic curve. Let us assume that it is the first one. The other alternative follows due to symmetry.

Using [Equations 8 and 33-35](#), we can write the points on the first quadratic piece as a combination of the Bézier control points of the cubic curve, such that

$$\mathbf{Q}_{2i}(s_0) = a_0 \mathbf{b}_{i,0} + a_1 \mathbf{b}_{i,1} + a_2 \mathbf{b}_{i,2} + a_3 \mathbf{b}_{i,3} \quad (36)$$

where

$$a_{i,0} = 1 - \sigma_i s_0^2 - a_{i,1}, \quad a_{i,1} = \frac{3}{4} \beta_{2i} (2s_0 - s_0^2 - \sigma_i s_0^2), \quad a_{i,2} = \sigma_i s_0^2 \frac{3}{4} \beta_{2i+1}, \quad \text{and} \quad a_{i,3} = \sigma_i s_0^2 - a_{i,2}.$$

If $C_i(t)$ is not planar, none of the control points can be written as a linear combination of the other control points. Therefore, based on Equation 7, at the parametric center of the cubic curve at $s = \frac{1}{2}$, we must have $a_{i,0} = \frac{1}{8}$, $a_{i,1} = \frac{3}{8}$, $a_{i,2} = \frac{3}{8}$, and $a_{i,3} = \frac{1}{8}$. This is only achieved when $\beta_{2i} = \beta_{2i+1} = 1$, $\sigma_i = \frac{1}{2}$, and $s_0 = 1$. This means that the two quadratic pieces join at this point and it perfectly matches the quadratic curves formed by the C^1 -continuous solution with $\gamma = \frac{1}{2}$. \square

When $C_i(t)$ is planar, as in the case of 2D curves, there are infinitely many solutions satisfying the G^1 conditions with this constraint.

6 DIRECT CONSTRUCTION OF PIECEWISE QUADRATIC CURVES

C^1 Catmull-Rom curves [Catmull and Rom 1974b] that produce cubic splines are commonly used for representing hair, fur, and yarn-level cloth models [Kaldor et al. 2010; Wu and Yuksel 2017a,b; Yuksel et al. 2012, 2009a], as they form interpolating curves that go through a given set of data points. In particular, C^1 Catmull-Rom curves with centripetal parameterization were shown to inherit unique properties, such as guaranteed self-intersection-free curve segments [Yuksel et al. 2009b, 2011], that make them an ideal choice for such models. Since the resulting curves are piecewise cubic, we can convert them to piecewise quadratic curves, as explained above.

Alternatively, we can construct piecewise quadratic curves directly from the given data points, using the same procedure as above, but bypassing the intermediate piecewise cubic curve. Given four consecutive data points \mathbf{p}_0 , \mathbf{p}_1 , \mathbf{p}_2 , and \mathbf{p}_3 , the Bézier form of a cubic Catmull-Rom curve segment that interpolates \mathbf{p}_1 and \mathbf{p}_2 can be written using the Bézier control points

$$\mathbf{b}_0 = \mathbf{p}_1 \tag{37}$$

$$\mathbf{b}_1 = \mathbf{p}_1 + \frac{d_1^{2\alpha}(\mathbf{p}_2 - \mathbf{p}_1) + d_2^{2\alpha}(\mathbf{p}_1 - \mathbf{p}_0)}{3d_1^\alpha(d_1^\alpha + d_2^\alpha)} \tag{38}$$

$$\mathbf{b}_2 = \mathbf{p}_2 + \frac{d_3^{2\alpha}(\mathbf{p}_1 - \mathbf{p}_2) + d_2^{2\alpha}(\mathbf{p}_2 - \mathbf{p}_3)}{3d_3^\alpha(d_3^\alpha + d_2^\alpha)} \tag{39}$$

$$\mathbf{b}_3 = \mathbf{p}_2 \tag{40}$$

where $d_i = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$, and α controls the Catmull-Rom parameterization, such that $\alpha = \frac{1}{2}$ corresponds to centripetal parameterization. Using our degree reduction in Equations 13-15, we can write the control points of the corresponding pair of quadratic segments as

$$\mathbf{q}_{0,0} = \mathbf{p}_1 \tag{41}$$

$$\mathbf{q}_{0,1} = \mathbf{p}_1 + \gamma \frac{d_1^{2\alpha}(\mathbf{p}_2 - \mathbf{p}_1) + d_2^{2\alpha}(\mathbf{p}_1 - \mathbf{p}_0)}{2d_1^\alpha(d_1^\alpha + d_2^\alpha)} \tag{42}$$

$$\mathbf{q}_{1,0} = \mathbf{q}_{0,1} = (1 - \gamma)\mathbf{q}_{0,1} + \gamma\mathbf{q}_{1,1} \tag{43}$$

$$\mathbf{q}_{1,1} = \mathbf{p}_2 + (1 - \gamma) \frac{d_3^{2\alpha}(\mathbf{p}_1 - \mathbf{p}_2) + d_2^{2\alpha}(\mathbf{p}_2 - \mathbf{p}_3)}{2d_3^\alpha(d_3^\alpha + d_2^\alpha)} \tag{44}$$

$$\mathbf{q}_{1,2} = \mathbf{p}_2 \tag{45}$$

Using this formulation, piecewise quadratic curves can be directly generated from a given set of data points.

7 RESULTS

Figure 4 shows 2D example Bézier curve pieces approximated using our method with different γ values. The piecewise quadratic curve in all three examples intersect with the cubic curve at exactly three points: the two end points and one other point between them, determined by the chosen γ value. Note that the cubic and piecewise quadratic curves have the same parameter values at all three of their intersections. The derivatives of the curves, however, only match at the end points. Notice that the piecewise quadratic curve with $\gamma = \frac{1}{2}$ provides the closest approximation.

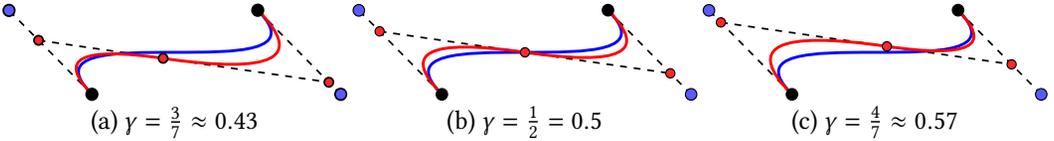


Fig. 4. An example cubic Bézier curve (blue) approximated with a pair of quadratic Bézier curves (red) using different values for γ , coinciding with the cubic curve at different parametric positions.

As expected, we get a closer approximation when the curvature of the cubic curve is not as exaggerated, such as the examples shown in Figure 5. Notice that the piecewise quadratic curves in these examples almost completely overlap with the cubic curves. Also, note that our approximation does not necessarily match the derivative of the cubic curve at the middle intersection point, since enforcing such a constraint would break the C^1 continuity of the quadratic curves.

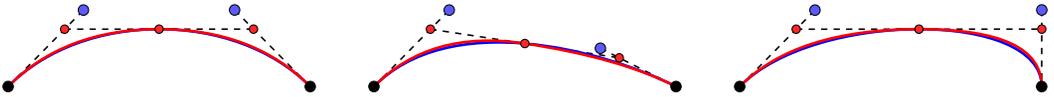


Fig. 5. (blue) Cubic Bézier curves approximated with (red) quadratic Bézier curves using $\gamma = \frac{1}{2}$.

7.1 Curves in 3D

Figure 6 shows an example 3D piecewise cubic C^1 Catmull-Rom curve [Catmull and Rom 1974b] along with its piecewise quadratic approximation. Notice that degree reduction in this case slightly changes the shape of the curve, even though each quadratic piece is planar. Note that the resulting piecewise quadratic curve is C^1 -continuous everywhere, just like the input piecewise cubic curve.

Our degree reduction approximates 3D curves with piecewise planar pieces. This poses a potential problem, since planar curves have zero torsion. We test the importance of this using a helix-shaped piecewise cubic curve, as a circular helix has constant non-zero torsion. Obviously, cubic polynomials cannot perfectly represent helices, but a cubic curve can approximate the shapes of arc lengths up to $\frac{\pi}{2}$ reasonably closely. The example in Figure 7 shows a helix-shaped piecewise cubic curve with each piece corresponding to an arc length of $\frac{\pi}{2}$. Notice that in Figure 7a, both cubic and quadratic curves have common data points (blue) while the quadratic curve has additional data points (red). Our degree reduction provides a close approximation, nonetheless, some minor variations in the shape of the curve and the positions/shapes of the specular reflections can be noticed as the helix rotates. Of course, using shorter arc lengths, we achieve a closer approximation, diminishing the impact of these subtle variations.

7.2 Cubic Curves with C^2 Continuity

Cubic splines are not limited to C^1 continuity and it is possible to form C^2 -continuous cubic splines. A common example of this is the B-spline formulation. Unlike Catmull-Rom curves that interpolate

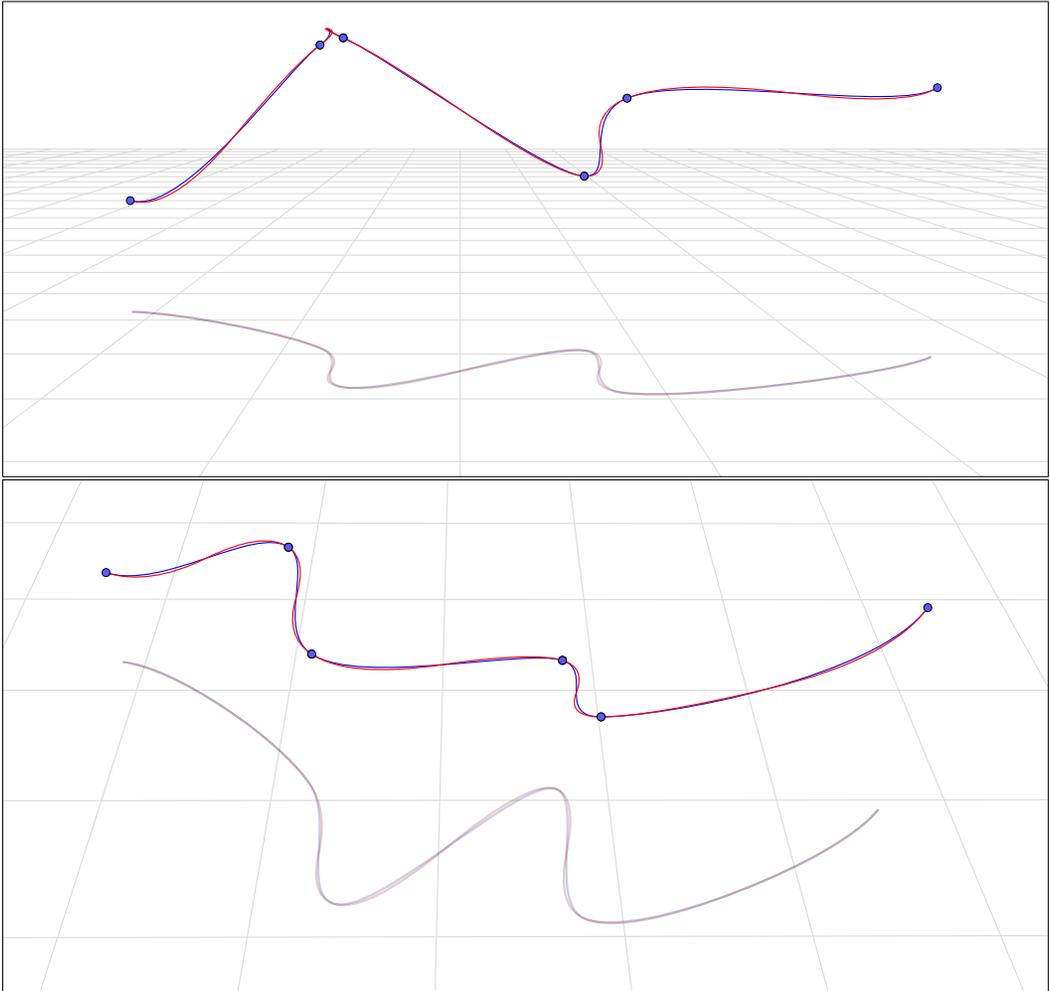


Fig. 6. Two different views of an example piecewise cubic C^1 Catmull-Rom curve (blue) in 3D and its piecewise quadratic C^1 approximation (red) using $\gamma = \frac{1}{2}$. The data points of the Catmull-Rom curve are shown.

a given set of data points, B-splines approximate the given control points and do not go through them. Yet, cubic B-splines have C^2 continuity.

Two example B-splines and their quadratic approximations are shown in Figures 8 and 9. Unfortunately, our degree reduction cannot maintain C^2 continuity. Therefore, the variation of the specular reflections visible on the thick curves is discontinuous for the piecewise quadratic curves. In these examples, our degree reduction produces a relatively close approximation of the original curve shapes, but some minor variations in the shapes of the specular reflections can be noticed, particularly in Figure 9.

7.3 Font Design

Quadratic curves are commonplace for font design. Figure 10a shows an example glyph with cubic curves converted to quadratic curves using our method with $\gamma = \frac{1}{2}$. This is a relatively simple

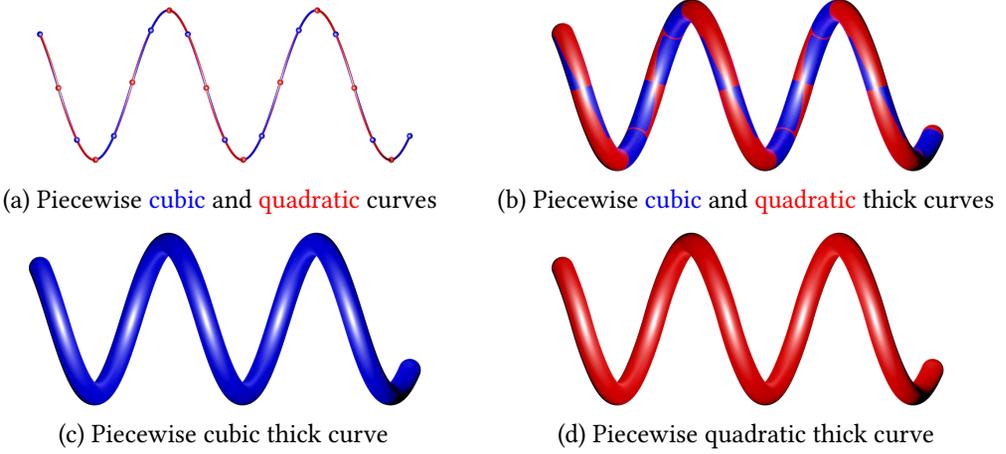


Fig. 7. A helix approximated by a piecewise cubic curve and its piecewise quadratic degree reduction using $\gamma = \frac{1}{2}$.

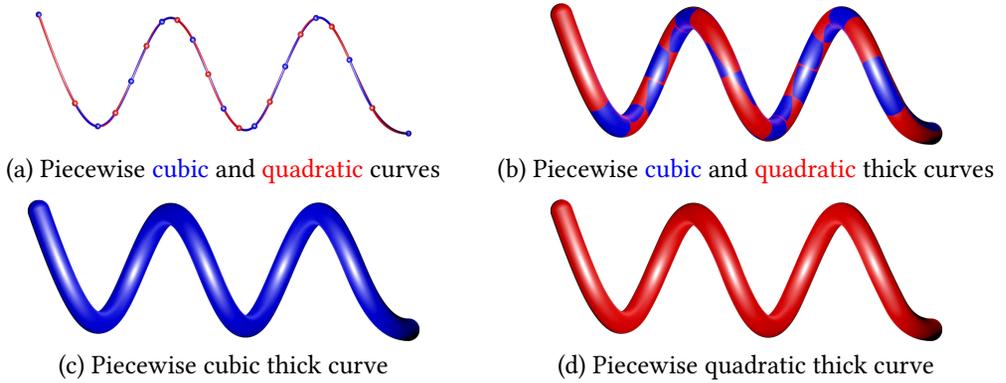


Fig. 8. A cubic C^2 clamped B-spline is approximated by C^1 piecewise quadratic curve using $\gamma = \frac{1}{2}$.

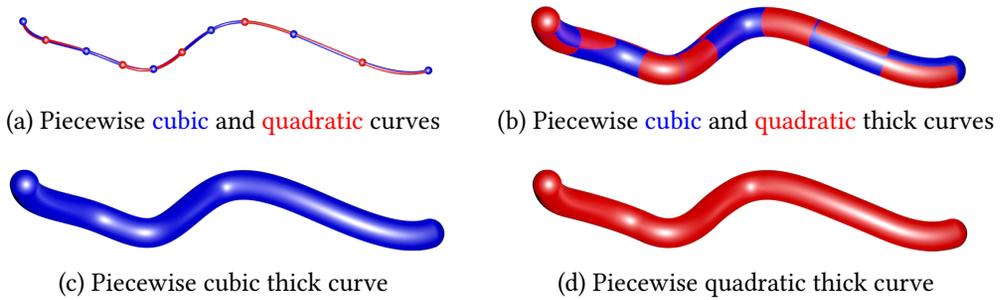


Fig. 9. A cubic C^2 clamped B-spline is approximated by C^1 piecewise quadratic curve using $\gamma = \frac{1}{2}$.

application for our method, since all curves are in 2D and most glyphs include relatively short cubic pieces. In fact, the example glyph in Figure 10a uses uncharacteristically long cubic pieces. Still, we can convert each cubic piece to a pair of quadratic pieces with minimal error, almost perfectly reproducing the glyph. Most other fonts we tested use more control points, resulting in extremely close approximations.

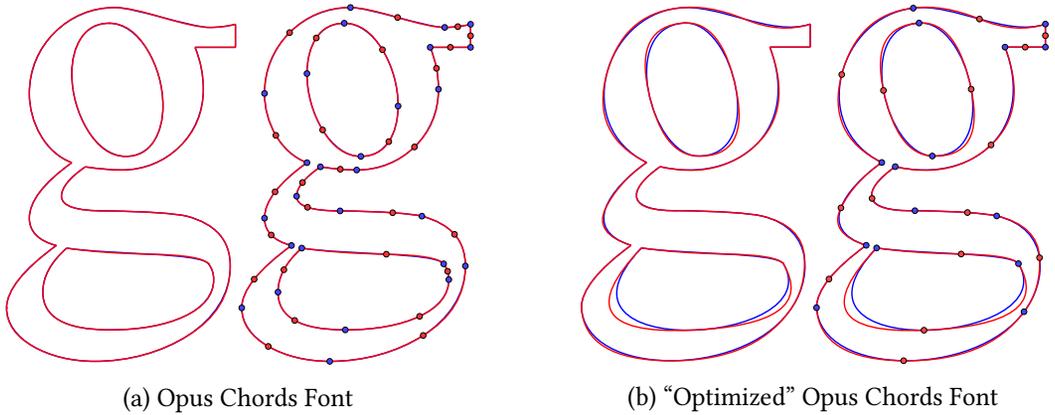


Fig. 10. Conversion of fonts with cubic curves (blue) to quadratic curves (red) using $\gamma = \frac{1}{2}$. The cubic and quadratic curves share common data points (blue). The quadratic curves also have additional data points (red). The curves coincide at all data points of the quadratic curves. (a) an example glyph from the OpenType Opus Chords font, showing that the quadratic curves closely approximate the cubic ones, and (b) an “optimized” version of the same glyph we generated to minimize the number of cubic pieces for exaggerating the differences between the cubic and quadratic curves. Of course, any visible differences can be reduced by subdividing cubic pieces that have approximation errors beyond a given threshold.

For generating a slightly more challenging case, we have manually modified the example glyph by minimizing the number of cubic pieces used for representing it, as shown in Figure 10b. The resulting cubic curves do not exactly match the glyph, but closely approximate it. In this case, we can see some differences between the cubic curves and the quadratic version, particularly for long segments with higher curvature. Obviously, subdividing such cubic pieces prior to generating the quadratic curves would result in a much closer quadratic approximation with more quadratic pieces. Such a subdivision can be applied adaptively by computing the maximum error using Equation 29 and comparing it to a given threshold.

7.4 Hair Rendering

Two hair models, providing examples of complex 3D geometry, are shown in Figure 1. Notice that the hair models represented using cubic and quadratic curves produce virtually indistinguishable results. Though there are minor geometric differences between the cubic and quadratic models, the resulting images are qualitatively equivalent, and it is difficult to identify which one is which, if possible at all.

The difference between cubic and quadratic curves, however, can be more distinguishable in terms of rendering performance. Since evaluating quadratic curves is computationally cheaper, they can provide performance advantages of varying degrees. We provide comparisons of ray tracing performance implemented using three different ray tracing applications: path tracing using

Table 1. Render time comparisons of hair rendering using cubic and quadratic curves within different ray tracing applications.

						
	216K curves	254K curves	800K curves	848K curves	50K curves	400K curves
PBRT, path tracing, 256spp						
Cubic	1130 s 241%	1372 s 220%	1830 s 233%	3821 s 293%	1583 s 228%	1729 s 230%
Cubic ×2	595 s 127%	655 s 105%	845 s 108%	1382 s 107%	703 s 101%	818 s 109%
Quadratic	469 s 100%	623 s 100%	785 s 100%	1303 s 100%	694 s 100%	752 s 100%
Embree, path tracing, 512spp						
Cubic	200 s 132%	183 s 115%	250 s 132%	323 s 141%	155 s 150%	163 s 129%
Cubic ×2	158 s 105%	166 s 104%	214 s 113%	274 s 120%	104 s 101%	134 s 106%
Quadratic	151 s 100%	159 s 100%	190 s 100%	229 s 100%	103 s 100%	126 s 100%
DXR, ray casting, 1spp						
Cubic	9.6 ms 240%	15.3 ms 243%	14.0 ms 259%	32.0 ms 344%	15.0 ms 250%	13.1 ms 262%
Cubic ×2	5.5 ms 138%	8.3 ms 130%	7.1 ms 131%	12.4 ms 133%	7.7 ms 128%	6.6 ms 132%
Quadratic	4.0 ms 100%	6.3 ms 100%	5.4 ms 100%	9.3 ms 100%	6.0 ms 100%	5.0 ms 100%
DXR, ray casting, phantom ray-hair intersector, 1spp						
Cubic	6.0 ms 214%	11.7 ms 266%	9.4 ms 254%	24.6 ms 351%	16.0 ms 327%	13.0 ms 361%
Cubic ×2	2.8 ms 100%	4.4 ms 100%	3.9 ms 105%	7.1 ms 101%	4.1 ms 104%	4.0 ms 111%
Quadratic	2.8 ms 100%	4.4 ms 100%	3.7 ms 100%	7.0 ms 100%	4.9 ms 100%	3.6 ms 100%

PBRT [Pharr et al. 2016] and Intel’s Embree [Wald et al. 2014] running on an Intel Core i9-9980XE CPU, and ray casting with ray traced shadows using the DXR running on an NVIDIA Quadro RTX 8000. All three implementations are tested using ray intersections with cubic and quadratic curve primitives computed with adaptive subdivision. We also include tests using DXR with phantom ray-hair intersector [Reshetov and Luebke 2018]. The quadratic primitives are generated from the cubic ones using our method with $\gamma = \frac{1}{2}$.

In all our tests we have observed substantially better performance when comparing piecewise quadratic models to the original piecewise cubic models that have half the number of curve primitives. Yet, most of this performance improvement with the quadratic representation is tied to the fact that splitting each cubic primitive into two primitives leads to better BVH performance, reducing the number of ray-curve intersection tests. This is because by replacing each cubic curve piece with two quadratic pieces, we get shorter curves with smaller bounding boxes.

To achieve a more fair comparison, we have also tested our quadratic curves to subdivided cubic curves, resulting in the same number of curve primitives. In this case, all ray tracing implementations produce similar BVHs with similar performances. Though the quadratic curves slightly different bounding boxes, as compared to their cubic counterparts, the differences are minor. In this case, our quadratic curves result in a relatively minor performance improvement in most cases. This is because the ray-curve intersection computation is a small percentage of the rendering operations, the rest of which are practically identical for all methods.

The results are summarized in Table 1, comparing the performance using the original cubic curves, cubic curves that are subdivided once (Cubic ×2) to form the same number of primitives as our quadratic curves, and our quadratic curves. Notice that, using adaptive subdivision for

intersections, the performance improvement of our quadratic approximation over the original cubic curves can be up to $3\times$ with PBRT, 50% with Embree, and more than $3\times$ with DXR. Subdivided cubic curves bring the performance closer to quadratics, but we can still observe improvements up to 27% with PBRT, 20% with Embree, and 38% with DXR. Our tests with phantom ray intersector using DXR resulted in closer rendering performances that are not always measurable, but can be up to 11%.

8 DISCUSSION

Our degree reduction approach produces curves with similar shapes and preserves C^1 continuity, since the resulting quadratic pieces pass through the end points of the cubic pieces and match the derivatives at the end points. C^2 or G^2 continuity, however, is not an attainable goal when using quadratic curves in 3D (or higher dimensions). Quadratic curves can have C^2 or G^2 continuity only in 2D and in special cases, when the curvature does not change sign. Therefore, even though mostly- G^2 curves can be formed using quadratic polynomials in 2D [Yan et al. 2017], the same properties cannot be obtained in 3D and higher dimensions. Therefore, even when the input piecewise cubic curve has C^2 continuity, the resulting piecewise quadratic curve has to be limited to C^1 in 3D.

Note that our degree reduction is *lossless*, in the sense that the original cubic curve can be perfectly reconstructed from the quadratic curves we generate, though the cubic and quadratic curves have slightly different shapes. Therefore, if the input piecewise cubic curve has C^2 continuity, the reconstructed cubic curve would have C^2 continuity as well.

In that respect, our method can also be used as a form of degree elevation: given a C^1 continuous piecewise quadratic curve (with even number of pieces), we can construct a C^1 piecewise cubic curve that passes through all data points (i.e. control points that are interpolated) of the quadratic curve. Obviously, unlike typical degree elevation that would convert each quadratic piece into a cubic piece [Farin 1997] and perfectly preserve the shape of the quadratic curve, degree elevation using our method would approximate the piecewise quadratic curve by replacing each pair or consecutive quadratic pieces with a cubic piece.

Note that degree elevation using our method reduces the number of control points (as opposed to adding more control points). Thus, the resulting cubic curve has lower degrees of freedom.

9 CONCLUSION

We have presented a simple approach for approximating a piecewise cubic polynomial curve with a C^1 -continuous piecewise quadratic curve that maintains the positions and the derivatives at the data points that the piecewise cubic curve interpolates. We achieve this by approximating each cubic piece with a pair of quadratic pieces. We have also shown that when the pair of quadratic curve pieces corresponding to a cubic piece have equal parameter lengths, the error due to degree reduction is minimized and the two quadratic pieces join at the parametric center of the cubic piece. Unlike typical degree reduction methods for cubic curves, our approach produces C^1 -continuous curves and it works in 3D and higher dimensions.

Our experiments with different hair rendering applications using ray tracing show that quadratic curves can lead to significant performance improvements in render time, though the savings vary significantly. While the performance of some implementations and hair models are greatly improved using quadratic curves, the improvements are not consistent and they can be relatively minor.

ACKNOWLEDGMENTS

This project was supported in part by a grant from Facebook Reality Labs.

REFERENCES

- Jens Alfke. 1994. *Converting Bézier Curves to Quadratic Splines*. <http://steve.hollasch.net/cgindex/curves/cbez-quadspline.html>
- Tobias Gröbeck Andersen, Viggo Falster, Jeppe Revall Frisvad, and Niels Jørgen Christensen. 2016. Hybrid Fur Rendering: Combining Volumetric Fur with Explicit Hair Strands. *Vis. Comput.* 32, 6–8 (June 2016), 739–749.
- Rasmus Barringer, Carl Johan Gribel, and Tomas Akenine-Möller. 2012. High-Quality Curve Rendering Using Line Sampled Visibility. *ACM Trans. Graph.* 31, 6, Article 162 (Nov. 2012), 10 pages.
- Phillip J. Barry and Ronald N. Goldman. 1988. A Recursive Evaluation Algorithm for a Class of Catmull-Rom Splines. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 199–204.
- Richard H. Bartels, John C. Beatty, and Brian A. Barsky. 1987. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- P. Bézier. 1977. *Essai de définition numérique des courbes et des surfaces expérimentales: contribution à l'étude des propriétés des courbes et des surfaces paramétriques polynomiales à coefficients vectoriels*. Number v. 1. Université Pierre et Marie Curie (Paris VI).
- Barbara Bily. 2014. On the method of rapid approximation of a cubic Bézier curve by quadratic Bézier curves. *Silesian Journal of Pure and Applied Mathematics* (2014).
- Edwin Catmull and Raphael Rom. 1974a. A Class of Local Interpolating Splines. In *Computer Aided Geometric Design*, Robert E. Barnhill and Richard F. Riesenfeld (Eds.). Academic Press, 317 – 326.
- Edwin E. Catmull and Raphael Rom. 1974b. A class of local interpolating splines. *Computer Aided Geometric Design* (1974), 317–326.
- Adrian Colomitchi. 2006. *Approximating cubic Bézier curves by quadratic ones*. <http://www.caffeineowl.com/graphics/2d/vectorial/cubic2quad01.html>
- M. G. Cox and P. M. Harris. 1991. The Approximation of a Composite Bézier Cubic Curve by a Composite Bézier Quadratic Curve. *IMA J. Numer. Anal.* 11, 2 (04 1991), 159–180.
- Gerald Farin. 1997. *Curves and Surfaces for Computer Aided Geometric Design (4th Ed.): A Practical Guide*. Academic Press Professional, Inc., USA.
- Gerald Farin. 2006. Class A Bézier Curves. *Computer Aided Geometric Design* 23, 7 (2006), 573–581.
- Thomas A. Foley, Nielson, and Gregory M. 1989. *Knot Selection for Parametric Spline Interpolation – Mathematical Methods in Computer Aided Geometric Design*. Academic Press Professional, Inc., San Diego, CA, USA. 261–272 pages.
- Timothée Groleau. 2002. *Approximating Cubic Bézier Curves in Flash MX*. http://www.timotheegroleau.com/Flash/articles/cubic_bezier_in_flash.htm
- Mengjiao Han, Ingo Wald, Will Usher, Qi Wu, Feng Wang, Valerio Pascucci, Charles D. Hansen, and Chris R. Johnson. 2019. Ray Tracing Generalized Tube Primitives: Method and Applications. *Computer Graphics Forum* (2019).
- Masatake Higashi, Kohji Kaneko, and Mamoru Hosaka. 1988. Generation of High-Quality Curve and Surface with Smoothly Varying Curvature. In *Eurographics 1988*. Eurographics Association.
- Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2010. Efficient Yarn-Based Cloth with Adaptive Contact Linearization. *ACM Trans. Graph.* 29, 4, Article 105 (July 2010), 10 pages.
- E. T. Y. Lee. 1989. Choosing nodes in parametric curve interpolation. *Computer Aided Design* 21, 6 (1989), 363–370.
- Dinesh Manocha and John F. Canny. 1992. Detecting cusps and inflection points in curves. *Computer Aided Geometric Design* 9, 1 (1992), 1–24.
- Yves Mineur, Tony Lichah, Jean Marie Castelain, and Henri Giaume. 1998. A Shape Controlled Fitting Method for Bézier Curves. *Computer Aided Geometric Design* 15, 9 (1998), 879–891.
- Koji Nakamaru and Yoshio Ohno. 2002. Ray Tracing for Curves Primitive. *WSCG* (2002), 311–316.
- G. Nielson and T. Foley. 1989. A Survey of Applications of an Affine Invariant Metric – Mathematical methods in computer aided geometric design. (1989), 445–468.
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- H. Qin, M. Chai, Q. Hou, Z. Ren, and K. Zhou. 2014. Cone Tracing for Furry Object Rendering. *IEEE Transactions on Visualization and Computer Graphics* 20, 8 (2014), 1178–1188.
- Alexander Reshetov. 2017. Exploiting Budan-Fourier and Vincent's Theorems for Ray Tracing 3D Bézier Curves. In *Proceedings of High Performance Graphics (Los Angeles, California) (HPG '17)*. Association for Computing Machinery, New York, NY, USA, Article 5, 11 pages.
- Alexander Reshetov and David Luebke. 2018. Phantom Ray-Hair Intersector. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 34 (Aug. 2018), 22 pages.
- Aleksas Riškus. 2006. Approximation Of A Cubic Bézier Curve By Circular Arcs And Vice Versa. *Information Technology and Control* 35, 4 (2006).

- Aleksas Riškus and Giedrius Liutkus. 2013. An Improved Algorithm for the Approximation of a Cubic Bézier Curve and its Application for Approximating Quadratic Bézier Curve. *Information Technology and Control* 42, 4 (2013).
- Mat Sutcliffe. 2007. *Approximating cubic Bézier curves*. https://academia.fandom.com/wiki/Approximating_cubic_Bezier_curves
- Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM Trans. Graph.* 33, 4, Article 143 (July 2014), 8 pages.
- Sven Woop, Carsten Benthin, Ingo Wald, Gregory S. Johnson, and Eric Tabellion. 2014. Exploiting Local Orientation Similarity for Efficient Ray Traversal of Hair and Fur. In *Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics*, Ingo Wald and Jonathan Ragan-Kelley (Eds.). The Eurographics Association.
- Kui Wu and Cem Yuksel. 2017a. Real-time Cloth Rendering with Fiber-level Detail. *IEEE Transactions on Visualization and Computer Graphics* PP, 99 (2017), 12.
- Kui Wu and Cem Yuksel. 2017b. Real-time Fiber-level Cloth Rendering. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D 2017)* (San Francisco, CA). ACM, New York, NY, USA, 8.
- Zhipei Yan, Stephen Schiller, Gregg Wilensky, Nathan Carr, and Scott Schaefer. 2017. κ -Curves: Interpolation at Local Maximum Curvature. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4, Article 129 (2017), 7 pages.
- Cem Yuksel, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2012. Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)* 31, 3, Article 37 (2012), 12 pages.
- Cem Yuksel, Scott Schaefer, and John Keyser. 2009a. Hair Meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)* 28, 5, Article 166 (2009), 7 pages.
- Cem Yuksel, Scott Schaefer, and John Keyser. 2009b. On the Parameterization of Catmull-Rom Curves. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (San Francisco, California). ACM, New York, NY, USA, 47–53.
- Cem Yuksel, Scott Schaefer, and John Keyser. 2011. Parameterization and Applications of Catmull-Rom Curves. *Computer Aided Design* 43, 7 (2011), 747–755.