Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Graphics 29 (2005) 656–666
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Fast, precise flattening of cubic Bézier path and offset curves ☆

Thomas F. Hain[a,*], Athar L. Ahmad[b], Sri Venkat R. Racherla[c], David D. Langan[a]

[a]*School of CIS, University of South Alabama, Mobile, AL 36688, USA*
[b]*Konica Minolta Systems Laboratory, 5661 Airport Boulevard, Boulder, CO 80301, USA*
[c]*SL Technologies, 711 MLK Jr. Boulevard, Biloxi, MS 39530, USA*

Abstract

We present two related algorithms for flattening (generating polyline approximations for) curves associated with planar cubic Bézier segments. One flattens the path curve, and the other flattens the left and right offset curves. The algorithm for flattening path curves yields an average of 67% of the vertices generated by recursive subdivision, while maintaining flatness to within 4% of the specified value, and runs 37% faster. The algorithm for flattening offset curves generates 70% of the vertices as the methods based on recursive subdivision, such that 94% of all subsegments fall within 20% of the flatness criterion. This latter code runs as fast as recursive subdivision.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Bézier curve; Rendering; Flattening; Offset curve

1. Introduction

The output of a flattening algorithm on an input curve is a polyline approximation for that curve. In this paper, we present fast algorithms for flattening both the path, and left and right offset curves of planar cubic Bézier curve segments. The first algorithm [5] can be used, for example, for fast rendering of thin Bézier segments, by rendering the polyline approximation instead, using the midpoint line algorithm [3]. The second algorithm can be used for rendering thick Bézier segments, by using the polyline approximations of the offset curves to form a polygonal outline of the curve, which can then be filled. The goal of a polyline approximation is to produce a minimum set of discrete vertices on a curve, that partition the curve into disjoint connected subsegments such that the maximum transverse deviation of each subsegment curve from its chord—line segment joining its endpoints—is no greater than a distance, f, called the *flatness*. We will call the actual deviation for a subsegment in a specific approximation the *achieved flatness*. Thus, the maximum achieved flatness should be no greater than f. Clearly, to generate the minimum set of vertices in the approximating polyline, the achieved flatness should be close to f for all subsegments. The advantage of minimizing this set is to improve rendering speeds, and to reduce the space requirements for display lists containing curve approximations.

The standard technique for doing this for cubic Bézier segments is by recursive subdivision [1], wherein the curve is recursively divided into two subsegments unless the flatness criterion for the subsegment is met. The

☆ Based on "Fast, Precise Flattening of Cubic Bézier Path and Offset Curves", by Thomas F. Hain, Sri Venkat R Racherla, David D. Langan which appeared in Proceedings of SIBGRAPI/SIACG 2004 (ISBN 0-77 = 695-2227-0).© 2004 IEEE.

*Corresponding author. Tel.: +1 251 460 6390; fax: +1 251 460 7274.
E-mail address: thain@usouthal.edu (T.F. Hain).

0097-8493/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2005.08.002

advantage of recursive subdivision is that the number of segments generated is variable—depending on the nature of the curve—rather than being fixed, as in the case of forward differencing [3]. The problem with recursive subdivision is that, if the flatness criterion is exceeded by even a small amount, the division is performed one more time, with each of the resulting segments having an achieved flatness of as little as 25% of *f*. In the worst case, the number of segments in the resulting polyline is greater than necessary by a factor of two. A technique called adaptive forward differencing was used in [7] to adjust the step size in forward differencing by factors of two to move along the curve with more nearly constant velocity. In our algorithm, as we will see, we avoid such discrete decisions. This adaptive approach was also used to break the curve into roughly pixel-sized pieces rather than the linear segments generated by our algorithm.

A uniformly thick curve can be regarded as having a path (the curve itself), and two parallel boundary curves—the offset curves—at a distance called the half-thickness to the left[1] and right of the path. Generally, the offset curves are analytically very complex, and an exhaustive compilation of approximation techniques is found in [2]. The most common technique, however, is to flatten the path curve into a series of disjoint curve subsegments, and then calculating the points perpendicular to the curve at the subsegment endpoints, and at a distance equal to the half-thickness. Unfortunately, flattening the path curve to the flatness criterion, and calculating orthogonal offset points on either side, generates offset curve chords which underestimate the flatness on the inside offset curve sections, and perhaps do not meet the flatness criterion on the outside sections. Thus, having the same number of vertices in the polyline approximation for both offset curves generally does not provide the desired perceptual smoothness. This effect is exacerbated as the curve thickness is increased. Another problem is that, on average, recursive subdivision generates too many path subsegments because of discrete round-off.

The described algorithms generate the polyline approximations for the path and offset curves in much the same way. They iteratively reduce the front end of the curve by a segment which closely meets the flatness criterion, thereby minimizing the number of generated linear segments. Note that the approximating offset curve polylines for the left and right offset curves are calculated independently, and may contain a different number of vertices. A fundamental idea used here is to approximate small sections of the curve by circular arcs. A similar idea was independently used in [6] but this involved very complex computations.

Section 2 gives some mathematical preliminaries for developing the algorithms. The first task of both algorithms is to partition the curves near any inflection points that lie within, or near, the target segment. This partitioning is described in Section 3. Section 4 develops the mathematical basis of the algorithm to flatten the path curve, and gives experimental results for polyline segment reduction, and run-time performance. The mathematical basis for flattening of cubic Bézier offset curves is given in Section 5, which also gives experimental results for polyline segment reduction, and run-time performance. Section 6 summarizes and presents conclusions.

2. Mathematical preliminaries

A cubic Bézier curve is defined on four control points $\mathbf{P}_0(x_0, y_0), \ldots, \mathbf{P}_3(x_3, y_3)$. The parametric equation of the curve $\mathbf{Q}(t) = (x(t), y(t)), \quad 0 \leqslant t \leqslant 1$ is

$$x(t) = (1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t)\,x_2 + t^3 x_3,$$
$$y(t) = (1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t)\,y_2 + t^3 y_3.$$

We now express the equations in terms of coordinates *r* and *s*, with the origin being at $\mathbf{P}_0$, the start of the curve at $t=0$, the *r*-axis being oriented along the velocity vector of the curve at $t=0$ (i.e., toward $\mathbf{P}_1$), and the *s*-axis being right-handed orthogonal to the *r*-axis. That is,

$$\hat{\mathbf{r}} = \frac{\mathbf{P}_1 - \mathbf{P}_0}{|\mathbf{P}_1 - \mathbf{P}_0|} = \left(\frac{x_1 - x_0}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}}, \frac{y_1 - y_0}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}} \right),$$

$$\hat{\mathbf{s}} = \left(\frac{y_1 - y_0}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}}, \frac{-(x_1 - x_0)}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}} \right).$$

Thus, a point $\mathbf{P}(x,y)$ is transformed into coordinates $\mathbf{P}(r,s)$ as follows:

$$r = (\mathbf{P} - \mathbf{P}_0) \cdot \hat{\mathbf{r}} = \frac{\left((x-x_0)(x_1-x_0) + (y-y_0)(y_1-y_0)\right)}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}},$$

$$s = \frac{\left((x-x_0)(y_1-y_0) - (y-y_0)(x_1-x_0)\right)}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2}}.$$

In this coordinate system, the transformed control points are $\mathbf{P}_0(r_0, s_0), \ldots, \mathbf{P}_3(r_3, s_3)$, and the parametric equations of a cubic Bézier curve are

$$r(t) = (1-t)^3 r_0 + 3t(1-t)^2 r_1 + 3t^2(1-t)\,r_2 + t^3 r_3,$$
$$s(t) = (1-t)^3 s_0 + 3t(1-t)^2 s_1 + 3t^2(1-t)\,s_2 + t^3 s_3,$$

[1]The 'left' or 'right' of a parametric curve is defined while looking along the curve in the direction of increasing parametric value.

which may be restated as

$$\begin{aligned} r(t) &= r_0 + 3(r_1 - r_0)t + 3(r_2 - 2r_1 + r_0)t^2 \\ &\quad + (r_3 - 3r_2 + 3r_1 - r_0)t^3, \\ s(t) &= s_0 + 3(s_1 - s_0)t + 3(s_2 - 2s_1 + s_0)t^2 \\ &\quad + (s_3 - 3s_2 + 3s_1 - s_0)t^3. \end{aligned} \tag{1}$$

3. Inflection points

The curve segment, whether offset or path, is first partitioned into as many as five sections, each of which is rendered independently. A cubic parametric curve may have zero or two inflection points at parametric values t_1 and t_2 of the path curve, where $t_1 \leqslant t_2$. A section surrounding the first inflection point t_1 (if it exists), and specified by parametric ranges $[t_1^-, t_1^+]$, where $t_1^- < t_1 < t_1^+$ may be approximated to within the flatness criterion by a single linear segment, provided $t_1 - t_1^-$ and $t_1^+ - t_1$ are sufficiently small. A similar section will exist surrounding the second inflection point, defined by $[t_2^-, t_2^+]$. Three sections remain, defined by parametric ranges $[-\infty, t_1^-]$, $[t_1^+, t_2^-]$ and $[t_2^+, +\infty]$. These represent curved regions which would require a polyline approximation, and whose curvature is wholly positive or negative. The Bézier segment we wish to render lies in the parametric range $[0, 1]$ as defined by the control points, and the segment will be partitioned into the 1–5 curve sections which overlap this $[0, 1]$ range. It should again be noted that the above-defined sections of both path and offset curves are referred to as ranges in the parametric value of the path curve.

We now need to find approximations to the parametric values, t_1^-, t_1^+, t_2^- and t_2^+. We can write coordinates of the path curve as parametric functions

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y, \end{aligned}$$

where, using the Bézier basis matrix, the coefficients in terms of the control points are

$$\begin{aligned} a_x &= -x_0 + 3x_1 - 3x_2 + x_3, & a_y &= -y_0 + 3y_1 - 3y_2 + y_3, \\ b_x &= 3x_0 - 6x_1 + 3x_2, & b_y &= 3y_0 - 6y_1 + 3y_2, \\ c_x &= -3x_0 + 3x_1, & c_y &= -3y_0 + 3y_1, \\ d_x &= x_0, & d_y &= y_0. \end{aligned}$$

At inflection points, the component of the acceleration (second derivative of position) perpendicular to the velocity (first derivative of position) is zero; the cross product of the two vectors is zero. Thus,

$$\begin{aligned} \frac{dx}{dt}\cdot\frac{d^2y}{dt^2} - \frac{d^2x}{dt^2}\cdot\frac{dy}{dt} &= (3a_x t^2 + 2b_x t + c_x)(6a_y t + 2b_y) \\ &\quad - (6a_x t + 2b_x)(3a_y t^2 + 2b_y t + c_y) \\ &= 6(a_y b_x - a_x b_y)t^2 + 6(a_y c_x - a_x c_y)t \\ &\quad + 2(b_y c_x - b_x c_y) \\ &= 0. \end{aligned}$$

Solving this quadratic equation for t yields

$$\begin{aligned} t_1 &= t_{cusp} - \sqrt{t_{cusp}^2 - \frac{1}{3}\left(\frac{b_y c_x - b_x c_y}{a_y b_x - a_x b_y}\right)}, \\ t_2 &= t_{cusp} + \sqrt{t_{cusp}^2 - \frac{1}{3}\left(\frac{b_y c_x - b_x c_y}{a_y b_x - a_x b_y}\right)}, \end{aligned}$$

where

$$t_{cusp} = -\frac{1}{2}\left(\frac{a_y c_x - a_x c_y}{a_y b_x - a_x b_y}\right),$$

the parametric positions t_1 and t_2 of the inflection points, if they exist (i.e., have real solutions). If the two inflection points are coincident (or, in practice, very close), the common point is the cusp point, t_{cusp}.

We now describe the handling of regions surrounding inflection points. Here, we subdivide[2] the curve segment at inflection point, t_1. Consider the second subsegment in a local r–s coordinate system, whose origin is defined at $\mathbf{P}_{01}$ (the inflection point, and the first control point of the second segment), and whose r-axis is oriented along the velocity vector at t_1, as shown above. We now switch the meaning of the parametric variable t to be relative to the second segment (such that $0 \leqslant t \leqslant 1$).

At inflection points, only the derivative of the acceleration has a component perpendicular to the velocity vector, and consequently we have $r_0 = s_0 = s_1 = s_2 = 0$. The transverse portion of Eq. (1) becomes

$$s(t) = t^3 s_3.$$

If we set $s(t) = f$ and solve for t, we have

$$t_f = \sqrt[3]{\frac{f}{s_3}}.$$

[2]To subdivide a Bézier curve defined by control points $\mathbf{P}_0, \ldots, \mathbf{P}_3$ at t define

$$\begin{aligned} &\mathbf{P}'_0 = \mathbf{P}_0 + t \times (\mathbf{P}_1 - \mathbf{P}_0), \mathbf{P}'_1 = \mathbf{P}_1 + t \times (\mathbf{P}_2 - \mathbf{P}_1), \\ &\mathbf{P}'_2 = \mathbf{P}_2 + t \times (\mathbf{P}_3 - \mathbf{P}_2), \\ &\mathbf{P}''_0 = \mathbf{P}'_0 + t \times (\mathbf{P}'_1 - \mathbf{P}'_0), \\ &\mathbf{P}''_1 = \mathbf{P}'_1 + t \times (\mathbf{P}'_2 - \mathbf{P}'_1), \mathbf{P}'''_0 = \mathbf{P}''_0 + t \times (\mathbf{P}''_1 - \mathbf{P}''_0). \end{aligned}$$

The control points of the first segment are $\mathbf{P}_0, \mathbf{P}'_0, \mathbf{P}''_0, \mathbf{P}'''_0$, and of the second segment are $\mathbf{P}'''_0, \mathbf{P}''_1, \mathbf{P}'_2, \mathbf{P}_3$.

Table 1
Case analysis for inflection points

Case	Treatment
$[t_1^-,\ t_1^+] \subseteq [0,1] \wedge [t_2^-,\ t_2^+] \cap [0,1] =$	Use circular approximation to flatten path and offset sections $[0, t_1^-]$. Generate linear approximation to approximate the path and offset sections $[t_1^-, t_1^+]$. Use circular approximation to flatten path and offset sections $[t_1^+,\ 1]$.
$0 \in [t_1^-,\ t_1^+] \wedge [t_2^-,\ t_2^+] \cap [0,1] =$	Generate linear approximation to approximate the path and offset sections $[0,\ t_1^+]$. Use circular approximation to flatten path and offset sections $[t_1^+, 1]$.
$[t_1^-, t_1^+] \cap [t_2^-, t_2^+] \neq \wedge [t_1^-,\ t_2^+] \subseteq [0,1]$	Use circular approximation to flatten path and offset sections $[0, t_1^-]$. Generate linear approximation for the offset sections $[t_1^-, t_{cusp}]$ and $[t_{cusp}, t_1^+]$. Use circular approximation to flatten path and offset sections $[t_2^+,\ 1]$.
Other cases	Handled similarly.

The achieved flatness of the curve segments $[-t_f, 0]$ and $[0,\ t_f]$ will be less than the transverse displacement $s(t_f)$.[3] Since the maximum transverse displacement for these two segments are of opposite signs, we can merge these segments into a single segment having the parametric range $[-t_f,\ +t_f]$, and flatten it. Transforming this parametric range into the corresponding parametric range in the original curve yields $[t_1^-,\ t_1^+]$ where $t_1^- = t_1 - t_f(1 - t_1)$ and $t_1^+ = t_1 + t_f(1 - t_1)$. A similar parametric range $[t_2^-,\ t_2^+]$ is found surrounding the second inflection point t_2 (if it exists). Thus, any intersection of the calculated parametric ranges $[t_1^-,\ t_1^+]$ and $[t_2^-,\ t_2^+]$ with the range $[0, 1]$ allows replacement by a single linear segment for the path curve, and therefore also for both offset curves.

An arbitrary (infinite) cubic Bézier curve having inflection points is partitioned into five regions, by successively t_1^-, t_1^+, t_2^- and t_2^+. Each endpoint of the segment to be rendered can be in one of the five regions, yielding a total of 21 cases. Any portion of the curve overlapping regions $[t_1^-, t_1^+]$ or $[t_2^-, t_2^+]$ can be approximated by a linear segment . Actually, there is a set of ten additional cases, representing cusps, in which $t_2^- \leqslant t_1^+$. Here, the regions are instead delimited by successively t_1^-, t_{cusp} and t_2^+. In these cases, any portion of the curve segment overlapping regions $[t_1^-, t_{cusp}]$ or $[t_{cusp}, t_2^+]$ can be approximated by a linear segment. The treatment for a sample of cases, including the cusp case, is summarized in Table 1.

In Table 1 the $t_{1,2}^{\pm}$ values are the approximations calculated above. Note that in the third case, there is an overlap between intervals $[t_1^-,\ t_1^+]$ and $[t_2^-,\ t_2^+]$, representing a cusp, or near-cusp. In this case, t_{cusp} represents the point nearest the cusp, and straight line segments approximate the ranges $[t_1^-,\ t_{cusp}]$ and $[t_{cusp}, t_1^+]$. The sharp point of the cusp is thereby approximated.

[3]In the circular approximation used above it was smaller by a factor of 4, but here we make no such assertion, and use the conservative value.

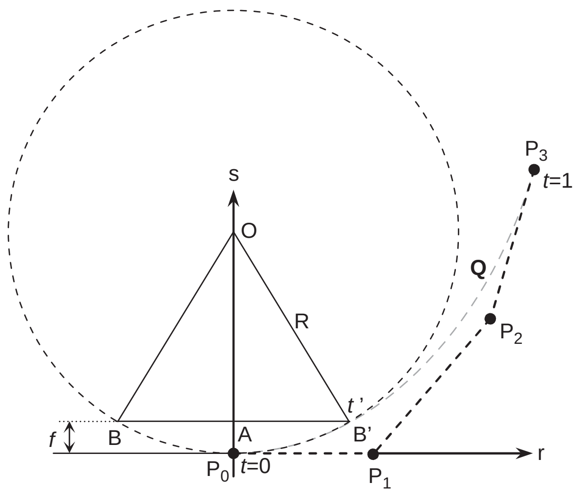

Fig. 1. Circular approximation to cubic Bézier path curve, **Q** (defined by control points $\mathbf{P}_{0,\ldots,}\mathbf{P}_3$) at $t = 0$.

At this point, the curve can be split at t_1^-, t_1^+, t_2^- and t_2^+ (where these points exist and fall within the range $[0, 1]$), with each section having its own set of control points. The purely curved sections generate approximating polylines as shown in Section 4 (for path curves) and 5 (for offset curves).

4. Bézier path curve flattening

A cubic Bézier segment, **Q**, having no inflection points, and defined by control points $\mathbf{P}_0, \ldots, \mathbf{P}_3$, can be approximated by a polyline using the following technique. First, we approximate the beginning of **Q** (around $\mathbf{P}_0$) by a circular arc having radius R, the same direction and curvature as **Q** at $t = 0$, and its center at O in a direction orthogonal to the curve. We define an r–s coordinate system aligned to the curve at $t = 0$, as shown in Fig. 1. We find the value t' such that

$R - \overline{OA} = f$, the flatness. For $f \ll R$, the curve from B to B', corresponding to the parametric range $[-t', t']$, can be approximated by the line segment $\overline{BB'}$which closely meets the flatness criterion, f.

Referring again to Eq. (1)

$$r(t) = r_0 + 3(r_1 - r_0)t + 3(r_2 - 2r_1 + r_0)t^2 + (r_3 - 3r_2 + 3r_1 - r_0)t^3,$$
$$s(t) = s_0 + 3(s_1 - s_0)t + 3(s_2 - 2s_1 + s_0)t^2 + (s_3 - 3s_2 + 3s_1 - s_0)t^3.$$

Because of the position and alignment of the r–s coordinate system, we see that $r_0 = s_0 = s_1 = 0$, yielding

$$r(t) \approx 3r_1 t + 3(r_2 - 2r_1)t^2 + (r_3 - 3r_2 + 3r_1)t^3,$$
$$s(t) \approx 3s_2 t^2 + (s_3 - 3s_2)t^3.$$

By the assumption of small t, the lower-order terms are dominant, and so

$$r(t) \approx 3r_1\, t,$$
$$s(t) \approx 3s_2\, t^2.$$

Thus, if we are trying to achieve a flatness f' on the path curve, we can calculate the value of t' where the curve deviates from its chord as follows:

$$s(t') = f' = 3s_2 t'^2,$$
$$\text{i.e. } t' = \sqrt{\frac{f'}{3|s_2|}}.$$

The modulus in the denominator takes care of positive or negative curvatures.

By simple extension, we can approximate the curve in the parametric range $[0, 2t']$ by a linear segment $\overline{\mathbf{P}_0\mathbf{Q}(2t')}$, also closely meeting the flatness criterion, f.

The segment is now subdivided at $2t'$ provided it is less than 1. The first subsegment is replaced by a linear subsegment from $\mathbf{Q}(0)$ to $\mathbf{Q}(\min(2t', 1))$, and the second subsegment is further reduced in the same way as just described. Thus, the curved section of the curve is approximated by a polyline such that the achieved flatness of each subsegment is very close to the required flatness. Only the last subsegment may have an achieved flatness between 0 and 1.

4.1. Segment reduction performance

The goal is to efficiently flatten a Bézier segment. We will compare the number of linear segments generated by our circular approximation algorithm (CA) with the number generated for the same curve by recursive subdivision (RS). The recursive subdivision algorithm we used uses the maximum deviation calculation method of Hain [4], which is more precise and no slower than conventional techniques for determining this value.

To generate a representative collection of 10,000 test curves, which attempts to cover a reasonable distribution of practical Bézier curves, we used a canonical representation [8], in which the first three control points are at (1,0), (0,0) and (0,1), and the fourth control point varies over a 100×100 grid from –3 to $+3$ in both x and y. The flatness criterion was fixed at 0.0005 (a typical relative resolution—however, the results were relatively insensitive to this value.)

For each of the test curves, the ratio of number of segments generated by both the RS and the CA algorithms was determined. The distribution of these ratios is given in Fig. 2. The ratios fall in the range from 1 to 2, with the mean being 1.496.

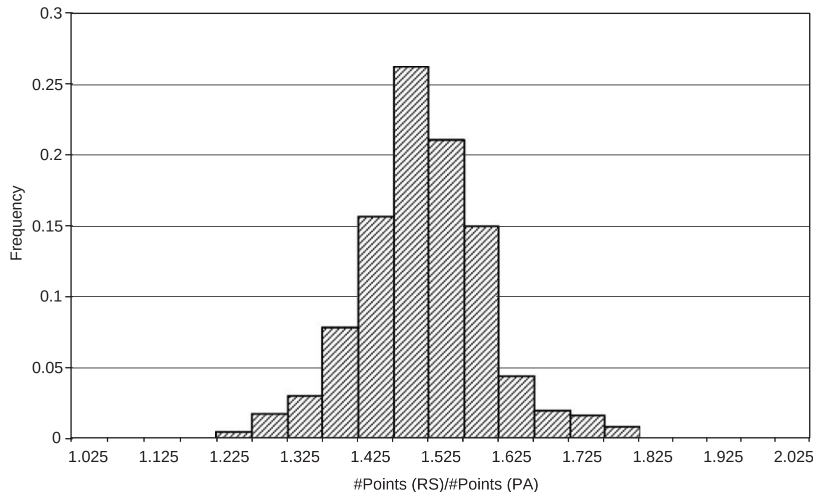

Fig. 2. Distribution of ratio of number of segments generated by recursive subdivision (RS) to circular approximation (CA).

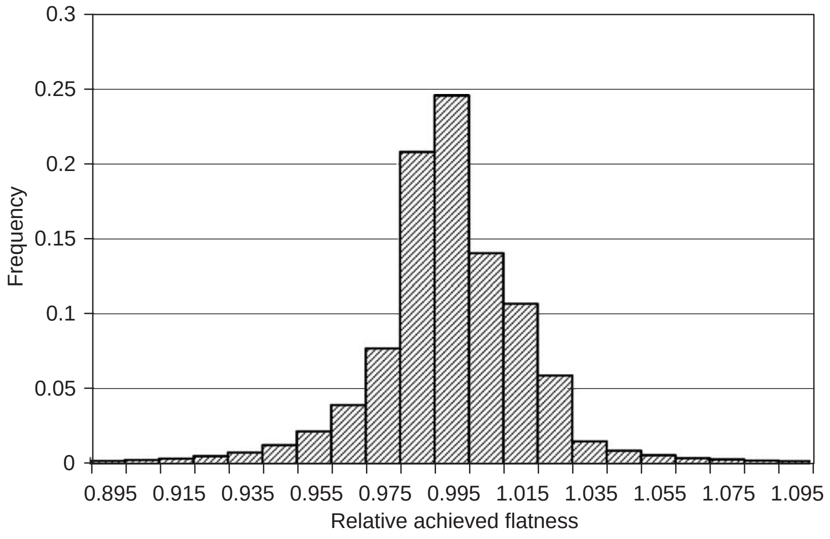

Fig. 3. Distribution of relative achieved flatness for CA algorithm.

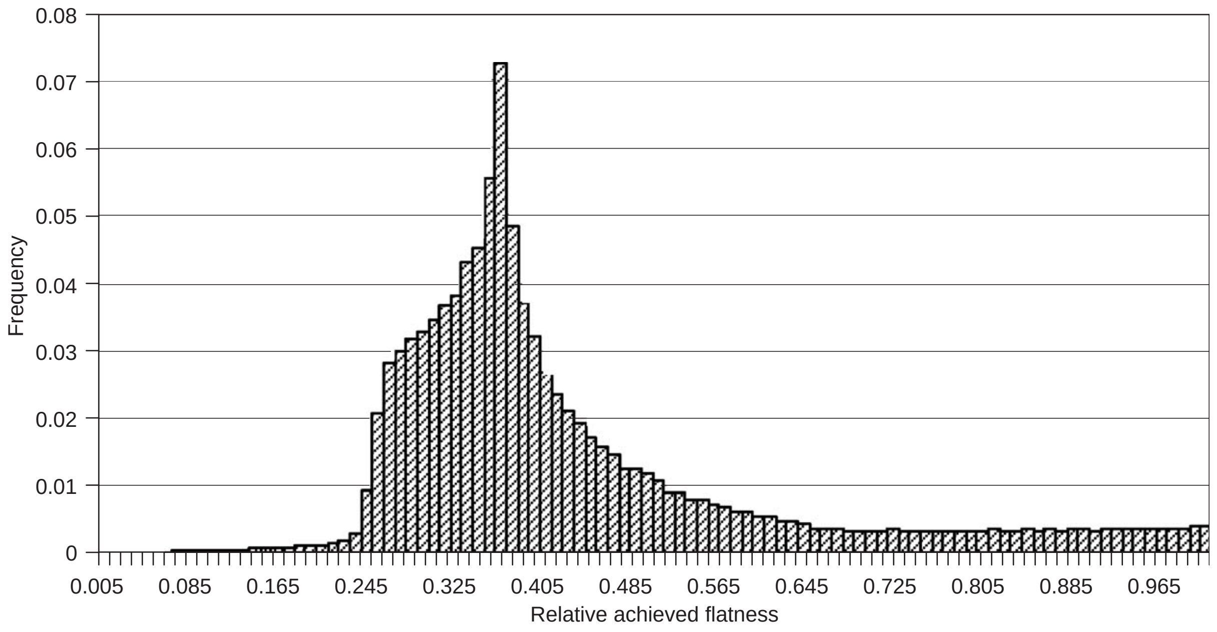

Fig. 4. Distribution of relative achieved flatness for RS algorithm.

The relative achieved flatness for all segments of all curves was determined for both the algorithms. As can be seen in Fig. 3, the distribution of the achieved flatness values in the CA algorithm, is very tight about the value 1. In fact, 95% of all segments fall within 3% of the specified value of *f*. This should be compared with the distribution of relative achieved flatness values for RS given in Fig. 4, which shows a large proportion of the segments have a value considerably below the optimal value of 1. Of course, by the nature of the RS algorithm, there are no segments whose achieved flatness exceeds *f*.

4.2. *Run-time performance*

The distribution of the ratio of RS run-time over CA run-time, collected over the 10,000 curves described above, is shown in Fig. 5. Codes were written in C++,

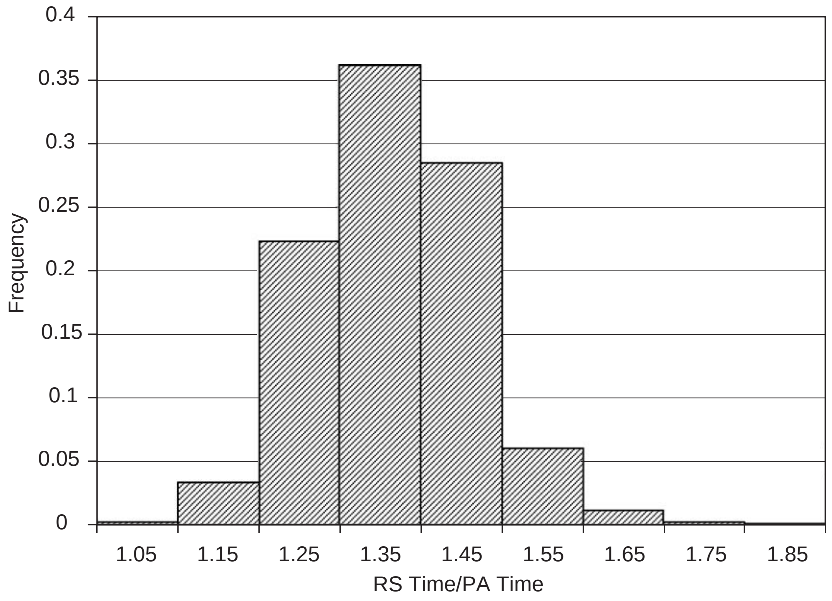

Fig. 5. Distribution of ratios of recursive subdivision (RS) to circular approximation (CA) run-times.

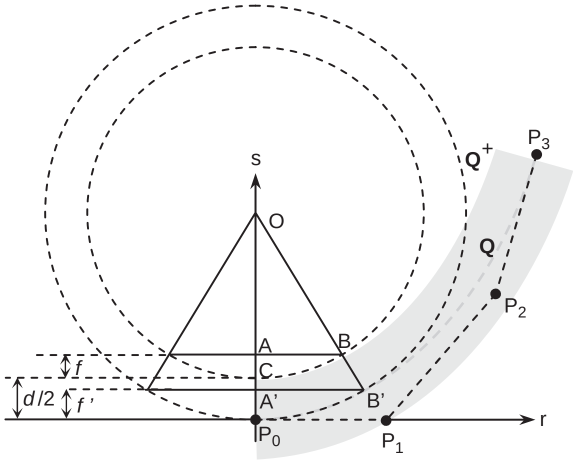

Fig. 6. Left offset curve.

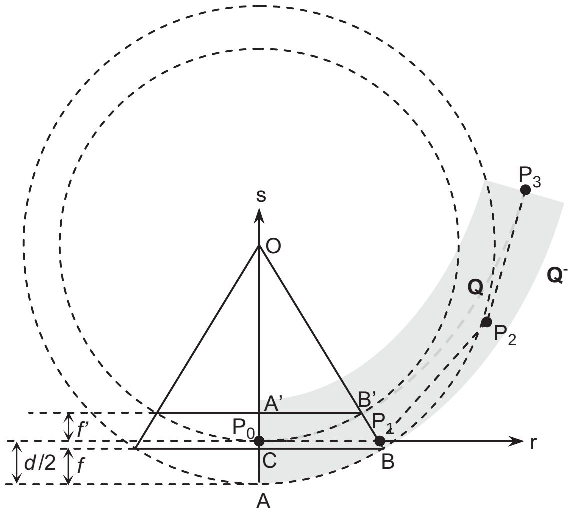

Fig. 7. Right offset curve.

and run on a 1.8 GHz Intel machine under MS Windows-XP. The mean speed-up is 1.37.

The reason for the CA speed-up is attributed to the facts that (1) fewer segments are generated, (2) no calculation of maximum deviation is required, and (3) the code is iterative rather than recursive.

5. Flattening cubic Bézier offset curves

Let $\mathbf{Q}^+$ and $\mathbf{Q}^-$ denote the left and right offset curves at a distance $d/2$ (half-thickness) from $\mathbf{Q}$, a cubic Bézier segment defined by control points $\mathbf{P}_0, \ldots, \mathbf{P}_3$. We will assume that either no inflection points exist in the path curve, or that they are sufficiently removed from the parametric range $[0, 1]$ (i.e., there is no overlap between either $[t_1^-, t_1^+]$ or $[t_2^-, t_2^+]$ and $[0, 1]$). This condition will be true for each of the "curved" sections of $\mathbf{Q}$ after it has been partitioned in a way described in Section 3. Furthermore, the curvature of this (sub)segment will be exclusively to the right or to the left. The algorithm processes $\mathbf{Q}$ twice, with the first pass generating a polyline approximating $\mathbf{Q}^+$ and the second pass generates the polyline approximating $\mathbf{Q}^-$.

Fig. 6 shows a thick Bézier curve $\mathbf{Q}$ defined on control points $\mathbf{P}_0(x_0, y_0), \ldots, \mathbf{P}_3(x_3, y_3)$, but drawn relative to an r–s coordinate system with the origin being at $\mathbf{P}_0$, the start of the curve at $t = 0$, the r-axis being oriented along the velocity vector of the curve at $t = 0$ (i.e., toward $\mathbf{P}_1$), and the s-axis being right-handed orthogonal to the r-axis. The control points relative to this coordinate system are $\mathbf{P}_0(r_0, s_0), \ldots, \mathbf{P}_3(r_3, s_3)$.

Over a sufficiently small range $[-t', +t']$, the path curve can be approximated by a circular arc of radius $\overline{\mathbf{OP_0}} = R$. The offset curve can therefore be approximated by a circular arc having the same center, O. We wish to find the parametric value t' of a point $\mathbf{B}'$ on the path curve such that the maximum transverse deviation of the offset curve from the line $\overline{\mathbf{AB}}$ is equal to the given flatness, i.e., $\overline{\mathbf{AC}} = f$.

Now consider point $\mathbf{B}'$ on the path, at coordinate $(r(t'), s(t'))$ calculated (see Eq. (1)) as

$$r(t') = r_0 + 3(r_1 - r_0)t' + 3(r_2 - 2r_1 + r_0)t'^2 + (r_3 - 3r_2 + 3r_1 - r_0)t'^3,$$
$$s(t') = s_0 + 3(s_1 - s_0)t' + 3(s_2 - 2s_1 + s_0)t'^2 + (s_3 - 3s_2 + 3s_1 - s_0)t'^3.$$

Since $\mathbf{P}_0$ is at the origin, and the r-axis is tangential to the path, we have $r_0 = s_0 = s_1 = 0$. Thus,

$$r(t') = 3r_1t' + 3(r_2 - 2r_1)t'^2 + (r_3 - 3r_2 + 3r_1)t'^3,$$
$$s(t') = 3s_2t'^2 + (s_3 - 3s_2)t'^3.$$

By the assumption of small t, the lower terms are dominant, and

$$r(t') \approx 3r_1t',$$
$$s(t') \approx 3s_2t'^2.$$

Thus, if we are trying to achieve a (positive) flatness f' on the path curve, we can calculate the value of t' such that the curve $[0, t']$ deviates from its chord as follows:

$$s(t') = f' = 3s_2t'^2,$$
$$\text{i.e. } t' = \sqrt{\tfrac{f'}{3|s_2|}}.$$

The maximum deviation of the left offset curve from its chord can be related to the maximum deviation of the path curve by noting that the triangles $\triangle OAB$ and $\triangle OA'B'$ are similar, as are $\triangle ABC$ and $\triangle A'B'P_0$. It can easily be seen that

$$\frac{f}{f'} = \frac{\overline{AC}}{\overline{A'P_0}} = \frac{\overline{AB}}{\overline{A'B'}} = \frac{\overline{OB}}{\overline{OB'}} = \frac{R - d/2}{R} = 1 - \frac{d}{2R},$$

where d is the thickness of the Bézier curve. Here, we need the radius of curvature, R. For small t', we may assert that

$$\overline{A'B'} \approx r(t') \approx 3r_1t'.$$

From Pythagoras we have

$$R^2 = r(t')^2 + (R - s(t'))^2,$$
$$\text{i.e. } R = \tfrac{s(t')^2 + r(t')^2}{2s(t')} \approx \tfrac{r(t')^2}{2s(t')} \approx \tfrac{3r_1^2}{2s_2}.$$

The sign of the radius depends on the sign of s_2, which determines whether the curvature is to the left (positive) or to the right (negative). Note also that s_2 will not be zero because of the assertion that we are sufficiently distant from inflection points (ranges immediately surrounding inflection points are handled separately in Section 3).

The "effective" flatness f' required for the path curve to ensure the required flatness f for the left offset curve is

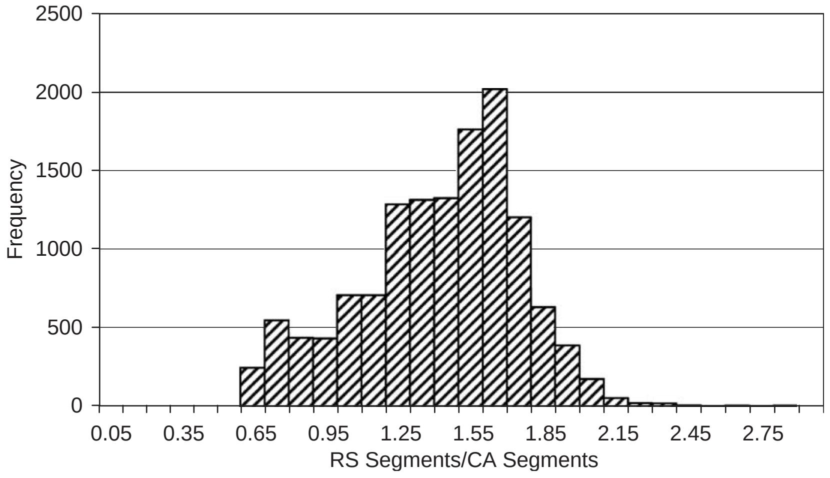

Fig. 8. Distribution of number of generated segments.

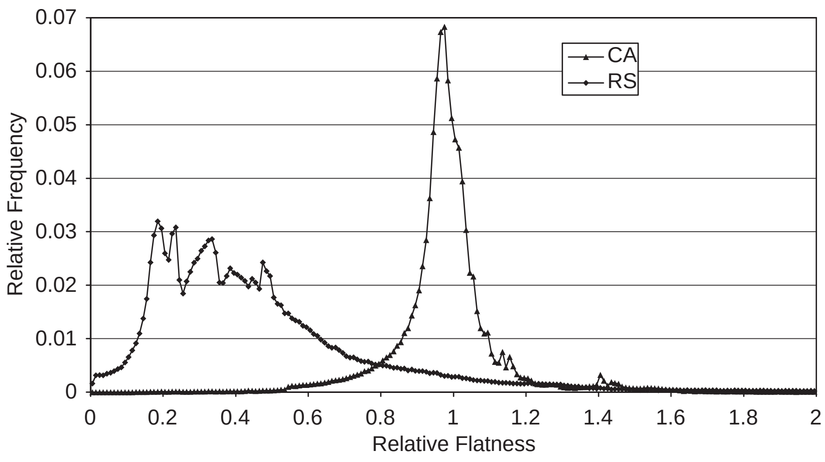

Fig. 9. Distributions of achieved flatness (relative to the given flatness criterion) for both COA and RSO algorithms.

thus

$$f' = \frac{f}{1 - ds_2/3r_1^2}.$$

We are actually interested in the magnitude of the maximum deviation. The required t' is calculated as

$$t' = \sqrt{\frac{f'}{3|s_2|}} = \sqrt{\frac{f}{3|s_2|\left(1 - ds_2/3r_1^2\right)}}$$

defining a point on the path curve. The corresponding point on the left offset curve (i.e., the polyline vertex) is calculated at a perpendicular distance of $d/2$ to the left of the path curve.

Under the assumption of small t', and a circular approximation, the maximum transverse deviation for the range $[-t', +t']$ is the same as $[0, 2\,t']$. Thus, we can subdivide the curve at

$$t = 2 \times \sqrt{\frac{f}{3|s_2|\left(1 - ds_2/3r_1^2\right)}} \tag{2}$$

such that the offset curve corresponding to the first path sub-curve has the required flatness f. The only other requirement is that the bracketed term in the denominator is positive. This will always be true if the curvature is to the right for the left offset curve (i.e., the offset curve is on the "outside" of the curve.) It will also be true if the radius of the path curve is greater than the half-thickness (i.e., either the thickness or the curvature is not too great.)

If the bracketed term is negative, the offset curve has retrograde motion, and is caustic. We handle this situation by replacing the term by its modulus to calculate the forward progress in t, but do not append a linear segment in the approximating polyline, thereby avoiding a self-intersecting polyline approximation.[4]

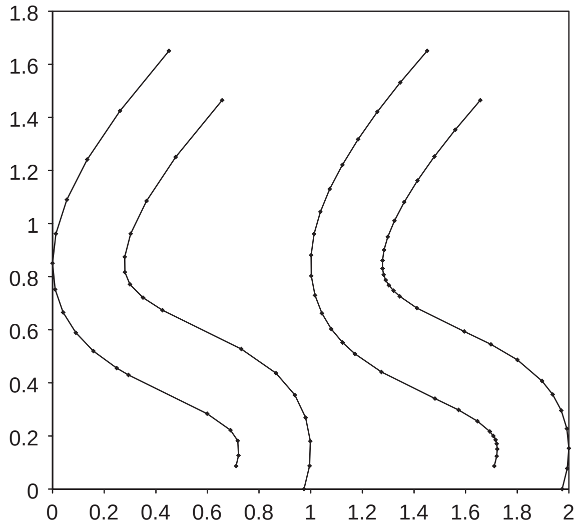

Fig. 10. COA (left) and RSO (right) flattened curves.

Fig. 7 shows the right offset curve, $\mathbf{Q}^-$. The mathematics is similar to the $\mathbf{Q}^+$case, with the result that here the path curve is subdivided at

$$t = 2 \times \sqrt{\frac{f}{3|s_2|\left(1 + ds_2/3r_1^2\right)}} \tag{3}$$

defining a point on the path curve. The corresponding polyline vertex on the right offset curve is calculated at a

[4]Of course, a thick-looped Bézier curve will necessarily produce a self-intersecting outline.

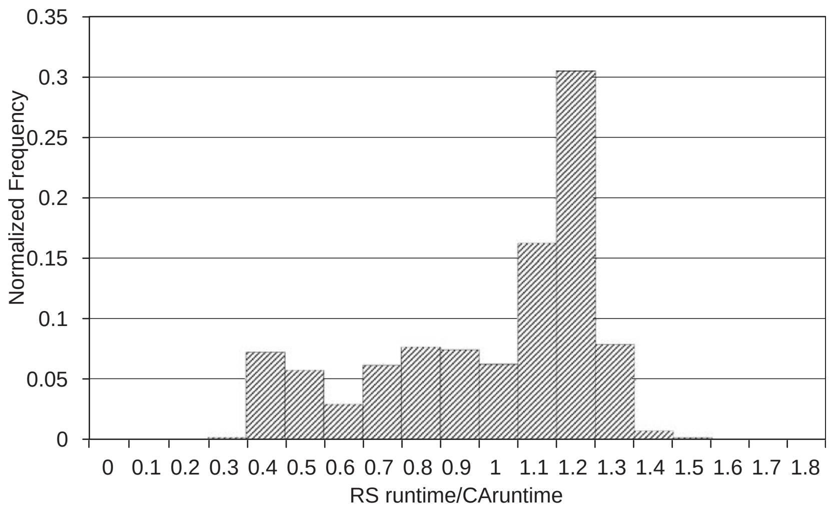

Fig. 11. Distribution of relative run-times.

perpendicular distance of $d/2$ to the right of the path curve.

5.1. *Segment reduction performance*

The goal is to efficiently flatten Bézier offset curve segments. We will compare the number of polyline segments generated by our circular offset approximation algorithm (COA) with the number generated for the same curve by recursive subdivision with offset point calculation (RSO) described in Section 1. We also compare the maximum deviation of the offset curve from each polyline segment (achieved flatness) for both algorithms.

The same 10,000 curves discussed in Section 4.1 were used here for test purposes The curve thickness was set to 0.5, representing a reasonably thick curve, given the positions of the first three control points. All curves having a section where the path curve radius was less than 125% of the curve half-thickness were discarded as pathological.

Fig. 8 shows the distribution (frequency) of curves as a function of the ratio of the number of segments generated by the RSO and the COA algorithms. Overall, RSO produces 42% more segments than COA.

Also importantly, the distribution of the segment vertices is such that the achieved flatness (on a scale relative to the specified flatness) is much more consistently around the desired value of 1 for the COA algorithm, as is shown in Fig. 9. It should be noted that achieved flatness values 20% over the specified flatness do not significantly affect the perceptual smoothness of the curve. The RSO algorithm generates many more than the required number of segments on the inside offset curve sections. The only reason that RSO does not frequently generate an insufficient number of segments on the outside offset curve sections is that RSO tends to be overly conservative in meeting the flatness criterion (for the path curve). This effect can be seen in the sample COA and RSO flattened offset curves in Fig. 10.

5.2. *Run-time performance*

Fig. 11 gives the relative RSO to COA run-time ratio distribution. The average ratio is 1.04. However, more common instances (well-behaved curves that are more likely to occur in practice) run almost 20% faster using the COA algorithm. The reasons for this observed speed-up in the COA were the same as those for the CA algorithm.

6. Conclusion

Algorithms for generating a polyline approximation (flattening) for both the path and offset curves of a planar cubic Bézier curve segment have been described. The path flattening algorithm (CA) is shown to be more efficient than recursive subdivision by generating only 2/3 as many segments, while 97% of all segments fall within 4% of the flatness criterion. The code[5] runs 37% faster than recursive subdivision. The offset curve flattening algorithm (COA) is shown to be more efficient than recursive subdivision by generating only 70% as many segments, but, just as importantly, 94% of all segments fall within 20% of the flatness criterion, although these numbers are somewhat dependent on the half-thickness. The COA code runs as fast as recursive subdivision.

[5]The codes for both algorithms, and an interactive testing platform, may be obtained from thain@usouthal.edu

The major contribution of this research was to provide fast algorithms to generate polyline approximations to cubic Bézier segment path curves, and their offset curves, using the least number of vertices, but while maintaining the flatness criterion. Both thick and thin cubic Bézier curves can be quickly rendered with perceptual smoothness using the algorithms presented.

References

[1] Catmul E. A subdivision algorithm for computer display of curved surfaces. UTEC-CSe-74-133, University of Utah, July 1974.

[2] Elber G, In-Kwon Lee, Myung-Soo Kim. Comparing offset curve approximation methods. IEEE Computer Graphics and Applications 1997;17(3):62–71.

[3] Foley JD, van Dam A, Feiner SK, Hughes JF. Computer graphics: principles and practice in C. Reading, MA: Addison-Wesley; 1996.

[4] Hain TF. Rapid termination evaluation for recursive subdivision of Bézier curves. In: Proceedings of the international conference on imaging science, systems, and technology, June 2002, Las Vegas, NV. p. 323.

[5] Hain TF, Ahmad A, Langan DD. Precise flattening of cubic Bézier segments. In: Proceedings of Canadian conference on computational geometry, August, 2004, Montreal, Canada.

[6] In-Kwon Lee, Myung-Soo Kim, Elber G. Planar curve offset based on circle approximation. Computer-aided Design 1996;28(8):617–30.

[7] Lien SL, Shantz M, Pratt V. Adaptive forward differencing for rendering curves and surfaces. Computer graphics, SIGGRAPH 1987 proceedings, vol. 21, no. 4. p. 111.

[8] Stone MC, DeRose TD. A geometric characterization of parametric cubic curves. ACM Transactions on Graphics 1989;9(3):147–63 July.