Fast, Precise Flattening of Cubic Bézier Segment Offset Curves

THOMAS F. HAIN[1] SRI VENKAT R RACHERLA[2] DAVID D. LANGAN[1]

[1]School of CIS, University of South Alabama, Mobile, AL 36688, USA
[2]SL Technologies, 711 MLK Jr. Blvd, Biloxi, MS 39530
E-mail: thain@usouthal.edu, sri.racherla@sl-tech.net, dlangan@usouthal.edu

Abstract

A fast algorithm for generating a polyline approximation (flattening) for the offset curves of a cubic Bézier curve segment is described. It is shown to be more efficient than the standard recursive subdivision method by generating only 70% as many segments, but, just as importantly, 94% of all segments fall within 20% of the flatness criterion. The code runs as fast as recursive subdivision.

1. Introduction

Approximating offset curves by polylines has application in diverse fields such as computer graphics, and CAD/CAM. An exhaustive compilation of techniques is found in [2]. The current paper trades off the severe complexities of many of these techniques, by focussing on specifically cubic Bézier segments. A fundamental idea is to approximate small sections of the curve by a circular arc, a similar idea independently used in [7] but which involves very complicated computation.

The Douglass-Peucker method [2] appropriately reduces a set of precalculated (linear) subsegments in multiresolution applications, but is not applicable here since the goal is to originally calculate a minimal approximating set of subsegments.

A uniformly thick curve can be regarded as having a path (the curve itself), and two parallel boundary curves at a distance called the half-thickness to the left[1] and right of the path, called the offset curves. A thick cubic Bézier curve segment is generally rendered by filling a polygonal outline approximation consisting of flattened offset curves. That is, the polygon is composed of two polylines, which approximate the left and right offset curves. The offset curves are analytically very complex.

The vertices of this polygon are generally obtained by subdividing the path curve into a series of disjoint curve subsegments, and then calculating the positions perpendicular to the curve at the subsegment endpoints, and at a distance equal to the half-thickness. The maximum transverse deviation of each path subsegment from the corresponding chord (the achieved flatness of the path) is constrained to be no greater than an error value, *f*, called the flatness. A common technique for flattening the path curve is by a process called recursive subdivision [1], wherein the curve is recursively divided into two subsegments until the flatness criterion is met. The advantage of recursive subdivision is that the number of subsegments generated is variable—depending on the nature of the curve—rather than being fixed, as in the case of forward differencing [1]. An improved path flattening algorithm by Hain *et al* [6] generates a minimal number of subsegments each closely meeting the flatness criterion, and forms the basic idea for the current algorithm.

However, flattening the path curve to the flatness criterion, and calculating offset points, generates segments which underestimate the flatness on inside offset curve sections, and perhaps do not meet the flatness criterion on outside sections. 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 by increases in thickness.

Another problem is that, on average, recursive subdivision generates too many path subsegments because of discrete round off. As a consequence, the number of vertices in the approximating polygon is generally too large, by as much as a factor of two [6] . The described algorithm repeatedly reduces the front end of a path curve by a segment whose flatness criterion for the desired offset curve is closely met, thus minimizing the number of generated linear segments in the approximating offset curve polyline.

Section 2 gives an overview of a new algorithm. A circular approximation method of flattening offset curve regions in the absence of inflection points is described in Section 0. Location and handling of inflection points is given in Section 4. Experimental results are presented in Section 5, with conclusions made in Section 6.

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

2. Algorithm overview

Let Q be a Bézier curve and let Q^+ and Q^- denote the left and right offset curves at a distance $d/2$ (half-thickness) from Q. The algorithm processes Q twice, with the first pass generating a polyline approximating Q^+ and the second pass generates the polyline approximating Q^-.

We will use the parametric value *t* of the underlying path curve to represent points on the offset curves. Parametric ranges $[t_1^-, t_1^+]$ and $[t_2^-, t_2^+]$ (the calculation of these values is described below) surround respectively inflection points t_1 and t_2 (if they exist), and the corresponding offset curve sections can be approximated by linear segments. The value t_{cusp} is an approximation to a cusp point (also described below) if it exists.

Thus, in our approach, we first partition the Bézier segment $t = [0,1]$ into up to 5 regions, as outlined in Table 1. The offset curves corresponding to these regions are alternatively approximated by linear segments, or by polyline sublists.

Table 1 Case Analysis for Inflection Points

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

The curved regions (away from inflection points) have curvatures exclusively to the left or the right. By approximating a small section of the curve around $t = 0$ to a circular arc, we can find the value t such that the achieved flatness of the target offset curve has the desired value f. We then subdivide[2] the curve at that point, approximate the first curve by a linear segment, and repeat the process on the remaining curve until that curve itself can be approximated by a linear segment.

3. Flattening by circular approximation

We now describe the flattening of the left offset curve, $\mathbf{Q}^+$, under the assumption that either no inflection points exist in the path curve, or 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 $[0,1]$). Because of the previous assertion, the curvature of this [sub]segment will be exclusively to the right or to the left. Figure 1 shows a thick Bézier curve **Q** defined on control points $\mathbf{P}_0(x_0, y_0), \cdots, \mathbf{P}_3(x_3, y_3)$, 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. That is,

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

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

The control points relative to this coordinate system are $\mathbf{P}_0(r_0, s_0), \cdots, \mathbf{P}_3(r_3, s_3)$

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., $\overline{\mathbf{AC}} = f$.

---

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

$$\mathbf{P}_0' = \mathbf{P}_0 + t \times (\mathbf{P}_1 - \mathbf{P}_0), \mathbf{P}_1' = \mathbf{P}_1 + t \times (\mathbf{P}_2 - \mathbf{P}_1), \mathbf{P}_2' = \mathbf{P}_2 + t \times (\mathbf{P}_3 - \mathbf{P}_2)$$
$$\mathbf{P}_0'' = \mathbf{P}_0' + t \times \left(\mathbf{P}_1' - \mathbf{P}_0'\right), \mathbf{P}_1'' = \mathbf{P}_1' + t \times \left(\mathbf{P}_2' - \mathbf{P}_1'\right), \mathbf{P}_1''' = \mathbf{P}_1'' + t \times \left(\mathbf{P}_1'' - \mathbf{P}_0''\right)$$

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

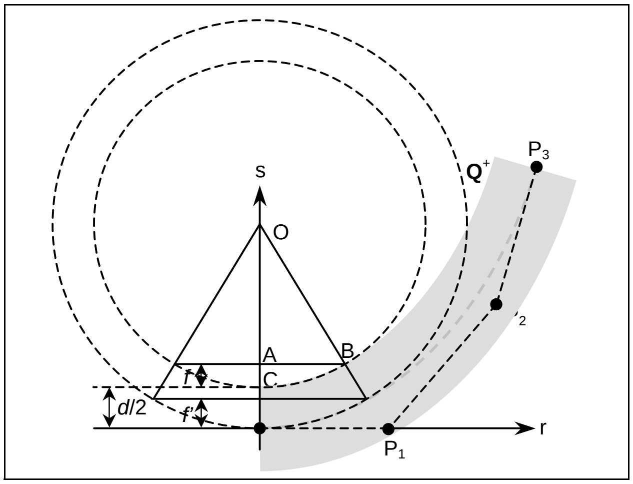

Figure 1 Left Offset Curve.

Now consider point $\mathbf{B}'$ on the path, at coordinate $\left(r(t'), s(t')\right)$ calculated (see [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 that $r_0 = s_0 = s_1 = 0$. Thus,

$$\begin{aligned} r(t') &= 3r_1t' + 3(r_2 - 2r_1)t'^2 + (r_3 - 3r_2 + 3r_1)t'^3 \\ s(t') &= 3s_2t'^2 + (s_3 - 3s_2)t'^3 \end{aligned} \qquad (1)$$

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:

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

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

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

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

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

The sign of the radius depends on the sign of s_2, which determines whether the curvature is to the left (positive) or 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 4).

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

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

We are actually interested in the magnitude of the maximum deviation. The required t' is calculated from (2)

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

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 $[0, 2t']$. Thus we can subdivide the curve at

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

such that the offset curve corresponding to the first path subcurve 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.) In this case, the offset curve has retrograde motion, and the outline polygon is self-intersecting. While this situation can be easily handled by separate means, for the sake of this paper's brevity, we choose to avoid this problem.

Figure 2 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

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

defining a point on the path curve. The corresponding polyline vertex on the right offset curve is calculated at a perpendicular distance of d/2 to the right of the path curve.

4. Location and Processing of Inflection Points

We can write coordinates of the path curve as parametric functions

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

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

$$\begin{array}{ll} a_x = -x_1 + 3x_2 - 3x_3 + x_4 & a_y = -y_1 + 3y_2 - 3y_3 + y_4 \\ b_x = 3x_1 - 6x_2 + 3x_3 & b_y = 3y_1 - 6y_2 + 3y_3 \\ c_x = -3x_1 + 3x_2 & c_y = -3y_1 + 3y_2 \\ d_x = x_1 & d_y = y_1 \end{array}$$

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,

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

Solving this quadratic equation for t yields

$$t_1 = t_{cusp} - \sqrt{t_{cusp}^2 - \frac{1}{3}\left(\frac{b_y c_x - b_x c_y}{a_y b_x - a_x b_y}\right)}$$

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

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

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}

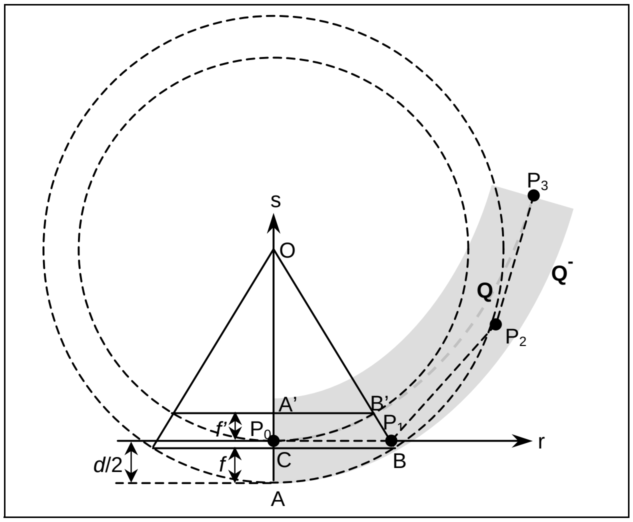

Figure 2 Right Offset Curve.

We now describe the handling of regions surrounding offset points. Consider again equation (1),

$$s(t) = 3s_2 t^2 + (s_3 - 3s_2)t^3$$

At inflection points, only the derivative of the acceleration has a component perpendicular to the velocity vector. Thus, if we subdivide the curve at one of the two inflection points (if any exist,) say t_1, and consider the second segment, again using an *r*-*s* coordinate system with the *r*-axis aligned with the velocity at the inflection point, and the origin at the inflection point, we have $r_1 = s_1 = s_2 = s_3 = 0$, and equation (1) becomes

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

where t is the parametric value relative to this segment (in which $t \in [0,1]$).

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

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

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 $[0,1]$ allow replacement by a single linear segment for the path curve, and therefore for both offset curves. The various arrangements are summarized in Table 1.

5. Results

The goal is to efficiently flatten Bézier offset curve segments. We will compare the number of polyline segments generated by our circular approximation algorithm (CA) with the number generated for the same curve by recursive subdivision (RS). We will also compare the maximum deviation of the offset curve from each polyline segment (achieved flatness) for both algorithms. The recursive subdivision algorithm we used uses the maximum (path) deviation calculation method of Hain [5], 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 [6], 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 this value.). All curves having a section where the path curve radius was less than 125% of the curve half-thickness were discarded as pathological (as explained in Section 0.) The curve thickness is 0.5, representing a reasonably thick curve, given the positions of the first three control points.

Figure 3 shows the distribution (frequency) of curves as a function of the ratio of the number of segments generated by the RS and the CA algorithm. Overall, RS produces 42% more segments than CA.

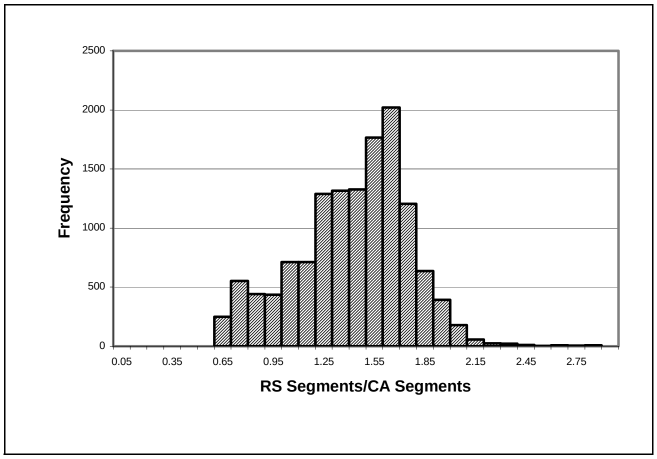

Figure 3 Distribution of number of generated segments.

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 CA algorithm, as is shown in Figure 4. It should be noted that achieved flatness values 20% over the specified flatness do not significantly affect the perceptual smoothness of the curve. The RS algorithm generates many more than the required number of segments on inside offset curve sections. The only reason that RS does not frequently generate an insufficient number of segments on outside offset curve sections is that RS tends to be overly conservative in meeting the flatness criterion (for the path curve). This effect can be seen in the sample CA and RS flattened offset curves in Figure 5.

Figure 6 gives the relative RS to CA runtime 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 CA algorithm. The reason for the lower CA runtime is a combination of (1) fewer segments are generated, (2) no calculation of maximum deviation is required, and (3) the code is iterative rather than recursive.

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

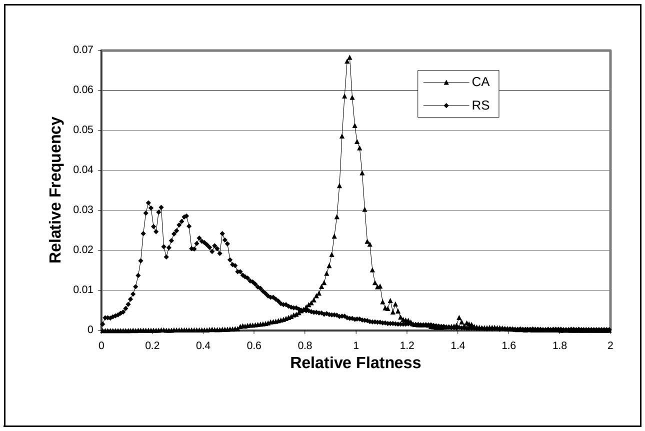

Figure 4 Distributions of achieved flatness (relative to the given flatness criterion) for both CA and RS algorithms.

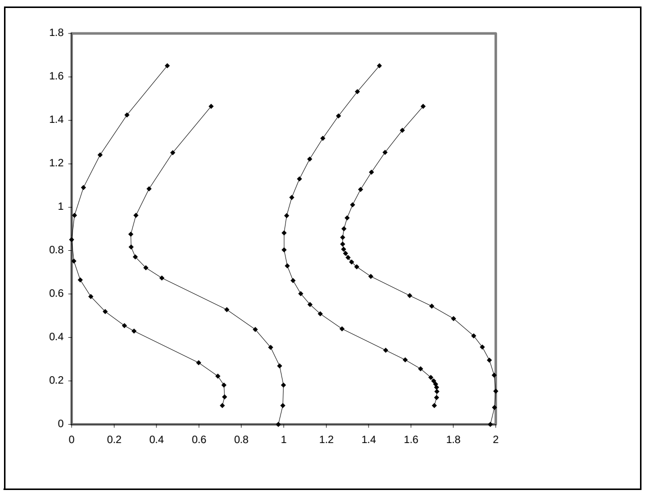

Figure 5 CA (left) and RS (right) flattened curves

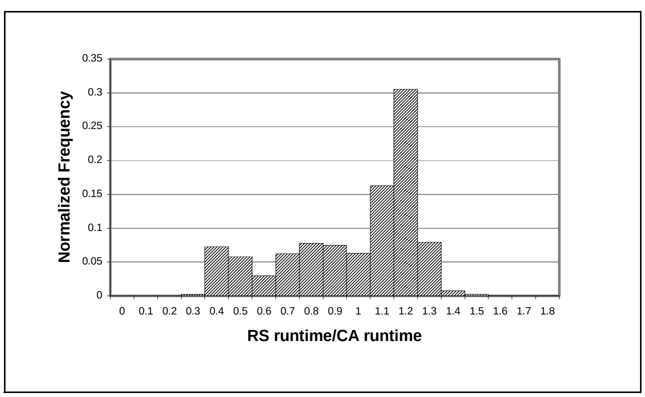

Figure 6 Distribution of relative runtimes

6. Conclusion

An algorithm for generating a polyline approximation (flattening) for the offset curves of a cubic Bézier curve segment has been described. It 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, athough these numbers are somewhat dependent on the half-thickness. The code[4] runs as fast as recursive subdivision.

References

[1] E. Catmul, "A Subdivision Algorithm for Computer Display of Curved Surfaces," *Ph. D. Thesis in Computer Science*, University of Utah, July 1974.

[2] D. Douglas, and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", *The Canadian Cartographer* 10(2), 1973, pp. 112–122

[3] G. Elber, In-Kwon Lee, and Myung-Soo Kim, “Comparing Offset Curve Approximation Methods,” *IEEE Computer graphics & Applications*, 17(3), 1997, pp. 62–71

[4] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice in C*, Addison Wesley, 1996.

[5] T.F. Hain, "Rapid Termination Evaluation for Recursive Subdivision of Bézier Curves," *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, Las Vegas, Nevada, June 2002, pp. 323–328.

[6] T.F. Hain, A. Ahmad, and D.D. Langan, “Precise Flattening of Cubic Bézier Segments,” *Proc. of Canadian Conference on Computational Geometry*, Montreal, Canada, Aug 2004.

[7] In-Kwon Lee, Myung-Soo Kim, and G. Elber, "Planar curve offset based on circle approximation", *Computer-aided Design*, 28(8), 1996, pp. 617–630

[8] M.C. Stone, and T.D. DeRose, “A Geometric Characterization of Parametric Cubic Curves,” *ACM Transactions on Graphics*, 9(3), July 1989, pp. 147-163.

---

[4] The code and an interactive testing platform may be obtained from thain@usouthal.edu.