# Precise Flattening of Cubic Bézier Segments

Thomas F. Hain* Athar L. Ahmad† David D. Langan‡


## Abstract

A method for flattening (generating polyline approximation for) cubic Bézier curve segments is given. It is shown to be more efficient than recursive subdivision by generating an average of only 2/3 as many linear segments, while maintaining the flatness criterion within 4%. The algorithm execution is 37% faster than recursive subdivision.


## 1 Introduction

A Bézier curve segment is generally rendered by subdividing it into a series of disjoint curve subsegments, and then approximating each subsegment by joining its endpoints by a line segment (chord). The maximum transverse deviation of each curve subsegment from the corresponding chord (the achieved flatness) should be no greater than a minimum error value, $f$, called the flatness. The standard technique for doing this is by a process called recursive subdivision [2], wherein the curve is recursively divided by two until the flatness criterion is met. The 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 [1]. 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 $f$. As a consequence, the number of segments in the resulting polyline is greater than necessary by as much as a factor of two. The described algorithm repeatedly reduces the front end of a curve by a segment whose flatness criterion is closely met, thus minimizing the number of generated segments in the approximating polyline.

## 2 Flattening by parabolic approximation

Figure 1 shows a Bézier curve defined on control points $\mathbf{P}_1(x_1, y_1), \cdots, \mathbf{P}_4(x_4, y_4)$. We wish to find the parametric value $t$ of a point $\mathbf{P}(x, y)$ on the curve such that the maximum transverse deviation of the curve from the line segment $\overline{\mathbf{P}_1\mathbf{P}}$ is equal to the flatness $f$. The curve is now subdivided[1] into two (generally unequal) curve segments. The first curve segment with parametric range $[0, t]$ can be replaced with sufficient accuracy by the line segment $\overline{\mathbf{P}_1\mathbf{P}}$. The remaining segment is the basis of a new calculation. This is continued until the solution for $t$ (with respect to the remaining curve) is greater than 1. At this point, the remaining curve can be replaced with a line segment $\overline{\mathbf{P}_1'\mathbf{P}_4'}$, where $\mathbf{P}_1'$ and $\mathbf{P}_4'$ are the curve endpoints.

While $t$ in the previous paragraph represents an optimal point, the analytical solution is computationally expensive, so we resort to an approximation. The method for estimating the value of $t$ where the curve should be subdivided relies on the fact that the beginning of the curve (for sufficiently small values of $t$) can be fitted to a parabola. This approximation works well for curves that have no inflection points, or for ranges of $t$ sufficiently removed from inflection points. We will examine this case first.

The parametric equation of the curve $\mathbf{Q}(t) = (x(t), y(t))$ is

$$\begin{cases} x(t) = (1-t)^3 x_1 + 3t(1-t)^2 x_2 + 3t^2(1-t)x_3 + t^3 x_4 \\ y(t) = (1-t)^3 y_1 + 3t(1-t)^2 y_2 + 3t^2(1-t)y_3 + t^3 y_4 \end{cases}$$

We now express the equations in terms of coordinates $r$ and $s$, with the origin being at $\mathbf{P}_1$, 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 $\mathbf{P}_2$), and the $s$-axis being right-handed orthogonal to the $r$-axis. That is,

$$\begin{aligned} \hat{\mathbf{r}} &= \frac{\mathbf{P}_2 - \mathbf{P}_1}{|\mathbf{P}_2 - \mathbf{P}_1|} \\ &= \left( \frac{x_2 - x_1}{\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}}, \frac{y_2 - y_1}{\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}} \right) \\ \hat{\mathbf{s}} &= \left( \frac{y_2 - y_1}{\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}}, \frac{-(x_2 - x_1)}{\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}} \right) \end{aligned}$$

Thus, a point $\mathbf{P}(x, y)$ has coordinates

$$\begin{aligned} r &= (\mathbf{P} - \mathbf{P}_1) \cdot \hat{\mathbf{r}} \\ &= \frac{(x-x_1)(x_2-x_1) + (y-y_1)(y_2-y_1)}{\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}} \\ s &= \frac{(x-x_1)(y_2-y_1) - (y-y_1)(x_2-x_1)}{\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}} \end{aligned}$$


*School of Computer & Information Sciences, University of South Alabama, thain@usouthal.edu

†Konica Minolta Systems Lab, Boulder, CO, athar.ahmad@bil.konicaminolta.us

‡School of Computer & Information Sciences, University of South Alabama, dlangan@usouthal.edu


[1] To subdivide a cubic Bézier curve defined by control points $\mathbf{P}_1, \cdots, \mathbf{P}_4$ at $t$ define

$$\begin{aligned} &\mathbf{P}_1' = \mathbf{P}_1 + t\times(\mathbf{P}_2 - \mathbf{P}_1),\ \mathbf{P}_2' = \mathbf{P}_2 + t\times(\mathbf{P}_3 - \mathbf{P}_2),\ \mathbf{P}_3' = \mathbf{P}_3 + t\times(\mathbf{P}_4 - \mathbf{P}_3) \\ &\mathbf{P}_1'' = \mathbf{P}_1' + t\times(\mathbf{P}_2' - \mathbf{P}_1'),\ \mathbf{P}_2'' = \mathbf{P}_2' + t\times(\mathbf{P}_3' - \mathbf{P}_2'),\ \mathbf{P}_1''' = \mathbf{P}_1'' + t\times(\mathbf{P}_2'' - \mathbf{P}_1'') \end{aligned}$$

The control points of the first segment are $\mathbf{P}_1, \mathbf{P}_1', \mathbf{P}_1'', \mathbf{P}_1'''$, and of the second segment are $\mathbf{P}_1''', \mathbf{P}_2'', \mathbf{P}_3', \mathbf{P}_4$.

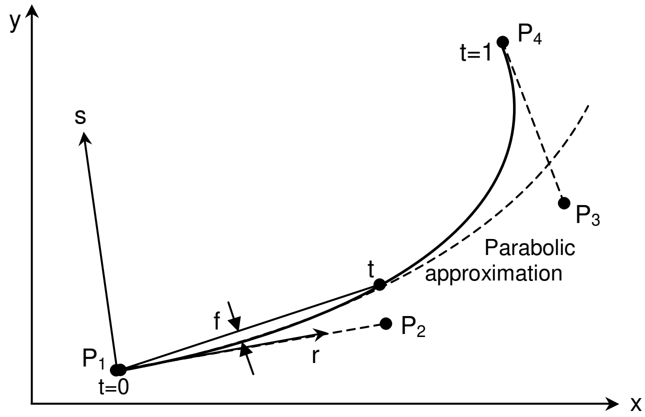


Figure 1: Approximating the start of a Bézier curve.

In this coordinate system, the control points are $\mathbf{P}_1(r_1, s_1), \cdots, \mathbf{P}_4(r_4, s_4)$, and considering only the movement of the curve in the $s$-direction

$$s(t) = (1-t)^3 s_1 + 3t(1-t)^2 s_2 + 3t^2(1-t)s_3 + t^3 s_4$$

Since $r_1 = s_1 = s_2 = 0$, we have,

$$\begin{aligned} s(t) &= 3t^2(1-t)s_3 + t^3 s_4 \\ &= 3s_3t^2 + (s_4 - 3s_3)t^3 \end{aligned} \qquad (1)$$

For small values of $t$, and assuming that $s_3$ is not close to zero (which would occur if the beginning of the curve were near an inflection point) the first term dominates. If we further assume the acceleration along the curve is reasonably small, i.e., the value of the $r$-coordinate varies reasonably linearly with $t$ (this assumption is met in all places except near a cusp point, which will be handled as a separate case,) the form of the curve can be approximated as parabolic as shown in Figure 2.

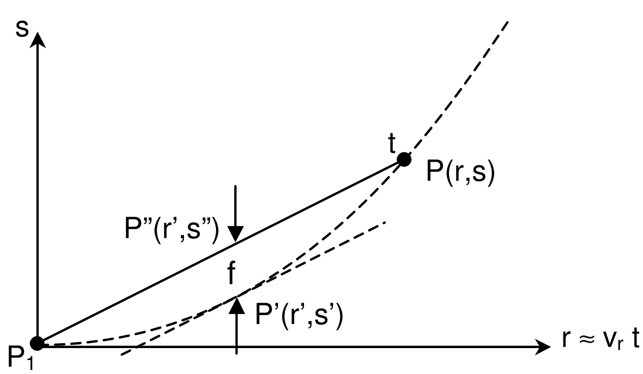


Figure 2: Parabolic approximation.

Thus, the form of the curve is $s = ar^2$, and

$$\frac{ds}{dr} = 2ar$$

Let $\mathbf{P}(r, s)$ be a point on the curve, and let $\mathbf{P}'(r', s')$ be the point on the curve, which has the maximum deviation from the line $\overline{\mathbf{P}_1\mathbf{P}}$. The slope of the curve at $\mathbf{P}'$ is equal to the slope of the line $\overline{\mathbf{P}_1\mathbf{P}}$. Thus,

$$2ar' = \frac{s}{r} = \frac{ar^2}{r}$$

I.e., $r' = \frac{1}{2}s$, and $s' = a(\frac{1}{2}r)^2 = \frac{1}{4}s$

Let $\mathbf{P}''(r', s'')$ be the point on $\overline{\mathbf{P}_1\mathbf{P}}$ at $r'$. By similar triangles,

$$\frac{s''}{r'} = \frac{s''}{\frac{1}{2}r} = \frac{s}{r}$$

I.e., $s'' = \frac{1}{2}s$

For small values of $r$, where the slope of $\overline{\mathbf{P}_0\mathbf{P}}$ is small, we get

$$\begin{aligned} max\ deviation &\approx |\overline{\mathbf{P}'\mathbf{P}''}| \\ &= s'' - s' \\ &= \tfrac{1}{2}s - \tfrac{1}{4}s \\ &= \tfrac{1}{4}s \end{aligned}$$

Note that this is independent of the constant $a$. We can now substitute this into equation (1), with the assumption of small $t$ together with $|s_3| > 0$, yielding

$$f \approx \left|\tfrac{1}{4}s\right| \approx \left|\tfrac{1}{4} \times 3s_3t^2\right|$$

$$\text{or,} \qquad t \approx 2 \times \sqrt{\frac{f}{3\,|s_3|}}$$

where $t$ is the parametric value of the curve such that the maximum deviation of the point $\mathbf{P}(t)$ from the line $\overline{\mathbf{P}_1\mathbf{P}}$ is approximately the flatness $f$.

## 3 Inflection points

We can write coordinates of the 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,

$$\begin{aligned} &\frac{dx}{dt} \cdot \frac{d^2x}{dt^2} - \frac{d^2x}{dt^2} \cdot \frac{dy}{dt} = 0 \\ &= (3a_xt^2 + 2b_xt + c_x)(6a_yt + 2b_y) \\ &\quad -(6a_xt + 2b_x)(3a_yt^2 + 2b_yt + c_y) \\ &= 6(a_yb_x - a_xb_y)t^2 + 6(a_yc_x - a_xc_y)t + 2(b_yc_x - b_xc_y) \end{aligned}$$

Solving this quadratic equation for $t$ yields

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

the parametric positions $t_1$ and $t_2$ of the inflection points, if they exist (i.e., have real solutions).

## 4 Processing inflection points

At inflection points only the derivative of the acceleration has a component perpendicular to the velocity vector. Thus, if we subdivide the curve at an inflection point, 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

$$s(t') = (t')^3 s_4$$

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

$$t_f = \sqrt[3]{\frac{f}{s_4}}$$

The achieved flatness of the curve segments $[-t_f, 0]$ and $[0, t_f]$ will be less than the transverse displacement $s(t_f)$.[2] 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 (existing) inflection point $t_2$.

## 5 Handling segments around inflection points

The curve segment to be rendered $(0 \le t \le 1)$ may be partitioned into up to five sequential subsegments, depending on the values of $t_1^-, t_1^+, t_2^-, t_2^+$, each of which can be approximated by either a straight line, or by a polyline for a segment having a consistent curvature either to the left or to the right. The cases are summarized in Table 1.

[2]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

| Case | Treatment |
|---|---|
| $[t_1^-, t_1^+] \subseteq [0, 1]$ $\wedge [t_2^-, t_2^+] \cap [0, 1] = \emptyset$ | Use parabolic approximation to flatten segment $[0, t_1^-]$. Replace curve segment $[t_1^-, t_1^+]$ by line segment. Use parametric approx. to flatten segment $[t_1^+, 1]$. |
| $0 \in [t_1^-, t_1^+]$ $\wedge [t_2^-, t_2^+] \cap [0, 1] = \emptyset$ | Replace curve segment $[0, t_1^+]$ by line segment. Use parabolic approx. to flatten segment $[t_1^+, 1]$. |
| $[t_1^-, t_1^+] \cap [t_2^-, t_2^+] \neq \emptyset$ $\wedge [t_1^-, t_2^+] \subseteq [0, 1]$ | Use parabolic approximation to flatten segment $[0, t_1^-]$. Replace curve segments $[t_1^-, t_{cusp}]$ and $[t_{cusp}, t_1^+]$ by line segments. Use parabolic approximation to flatten segment $[t_2^+, 1]$. |
| Other cases | Handled similarly. |

Table 1: Case analysis for inflection points

## 6 Segment reduction performance

The goal is to efficiently flatten a Bézier segment. We will compare the number of linear segments generated by our parabolic approximation algorithm (PA) 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 [3], 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 [4], in which the first three control points are at (1,0), (0,0), and (0,1), and the fourth control point varies over a 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.)

The ratio of number of segments generated by the RS to PA algorithms is given in Figure 3. The ratios fall in the range from 1 to 2, with the mean being 1.496. As can be seen in Figure 4, the distribution of the relative achieved flatness values in the PA algorithm (over all segments of all curves), are very tight about the value 1. In fact, 95% of all segments fall within 3% of the specified value of $f$. This can be compared to the distribution of relative achieved flatness values for RS given in Figure 5, which shows a large number of segments with values considerably below the optimal value of 1.

## 7 Run-time performance

The distribution of the ratio of RS over PA run-time, collected over 10,000 curves described above, is shown in Fig-

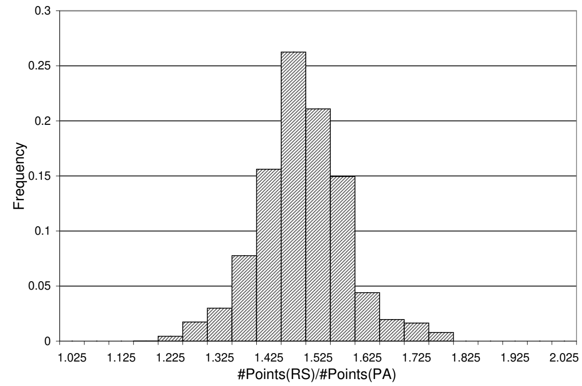


Figure 3: Distribution of ratio of number of segments generated by RS to that of PA.

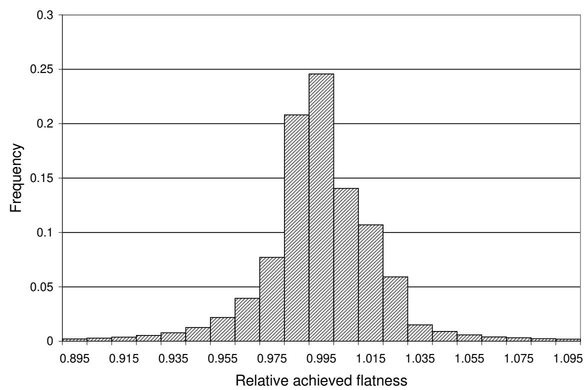


Figure 4: Distr. of relative achieved flatness for PA.

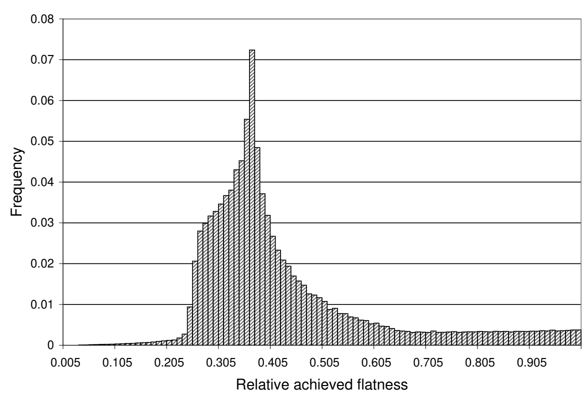


Figure 5: Distr. of relative achieved flatness for RS.

ure 6. Codes were written in C++, and run on a 1.8 GHz Intel machine under MS-XP. The mean speedup is 1.37. The reason for the PA speedup 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.

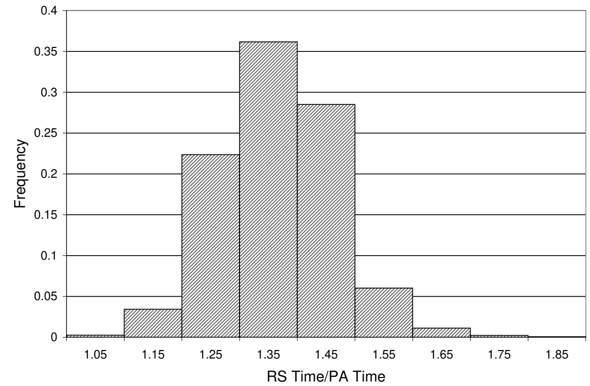


Figure 6: Distribution of ratios of recursive subdivision (RS) to parabolic approximation (PA) run-times.

## 8 Conclusion

An algorithm for the flattening of cubic Bézier curve segments has been described. It 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[3] runs 37% faster than recursive subdivision.

## References

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

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

[3] T. F. Hain. "Rapid Termination Evaluation for Recursive Subdivision of Bézier Curves," *Proc. of the Intern. Conf. on Imaging Science, Systems, and Technology*, Las Vegas, Nevada, June 24–27, 2002, pp. 323–328.

[4] M. C. Stone and A. D. DeRose. "A Geometric Characterization of Parametric Cubic Curves," *ACM Transactions on Graphics*, Vol.9, No.3, July 1989, pp. 147–163.

[3] A testing platform and code may be obtained from `thain@usouthal.edu`