

Rotated Explicit Curves

Thomas W. Sederberg, Chen Falai & Kryzysztof S. Klimaszewski

Abstract

This paper demonstrates that piecewise, rotated explicit curves are flexible enough for 2-D modeling applications such as font design. Their use simplifies many operations such as scan conversion, point classification, and curve intersection.

Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling.

General Terms: Algorithms

Additional Key Words and Phrases: Explicit curves, font outlines, scan conversion.

1 Introduction

Most high school students first encounter mathematically defined curves when studying the graphs of functions $y = f(x)$. These are often referred to as *explicit curves*. Some early books on computer-aided geometric design motivated the use of parametric curves ($x = f(t)$, $y = g(t)$) by pointing out that the more familiar explicit curves have limited practical value since most commonplace objects have geometries which are not single valued in y [2].

The early, well deserved excitement over the use of parametric forms such as Bézier and B-spline curves may have resulted in the premature dismissal of the explicit form as a practical modeling entity. This paper examines the use of explicit curves whose local coordinate systems are rotated, showing that such rotated explicit curves (or, RECs) have several attractive properties.

this note focuses on the ease with which they can be scan converted.

Section 2 proposes one way to define a REC. A simple algorithm for approximating a Bézier curve using piecewise RECs, is presented in section 4. Section 5 discusses application to scan conversion.

2 REC Definition

A polynomial Bézier curve

$$\mathbf{P}(t) = (x(t), y(t)) = \sum_{i=0}^n (x_i, y_i) B_i^n(t) \quad (1)$$

can represent an explicit polynomial curve $y = y(x)$ by evenly spacing the control points in the x direction: $x_i = \frac{i}{n}$. This results in

$$x(t) \equiv \sum_{i=0}^n x_i B_i^n(t) \equiv t[(1-t) + t]^{n-1} \equiv t \quad (2)$$

so $y(t) \equiv y(x)$. An explicit Bézier curve has been referred to as a non-parametric Bézier curve [1].

The main limitation in using explicit curves for modeling — that they can only express shapes that are single valued in y — is easily overcome by introducing a rotational degree of freedom. A REC is defined in its own lo-

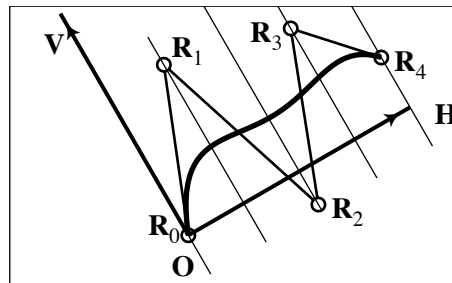


Figure 1: Rotated explicit curve

cal (H, V) coordinate system, with $H = t$ and $V = V(t)$. From Figure 1 we see that $n+4$ scalar values are needed to specify a degree n REC: the start point and local origin, \mathbf{O} ; the local horizontal axis, $\mathbf{H} = (H_x, H_y)$; and the local V coordinates of the control points. We define $\mathbf{V} = (-H_y, H_x)$.

3 Principal RECs

Of special interest are RECs that are aligned with the principal Cartesian coordinate axes. An XREC is a REC for which H is parallel to the x -axis, and in a YREC, H is parallel to the y -axis.

4 Approximation with RECs

It is straightforward to approximate a Bézier curve with one or more RECs. The rest of the paper will concentrate on polynomial cubic curves, though it is noteworthy that for degree two curves, choosing the (H, V) coordinate system so that $\mathbf{V} \parallel (\mathbf{P}_2 - 2\mathbf{P}_1 + \mathbf{P}_0)$ allows any polynomial degree two Bézier curve to be expressed exactly as a REC with no error.

Figure 2 shows a cubic Bézier curve $\mathbf{P}(t)$ (shown in dashed lines) approximated by a single cubic REC $\mathbf{R}(t)$. In order to assure G^1 continuity in a piecewise approximation, assign $\mathbf{R}_0 = \mathbf{P}_0$, $\mathbf{R}_3 = \mathbf{P}_3$, and choose \mathbf{R}_1 to lie on $\mathbf{P}_0\text{--}\mathbf{P}_1$ and \mathbf{R}_2 to lie on $\mathbf{P}_3\text{--}\mathbf{P}_2$. Of course, this

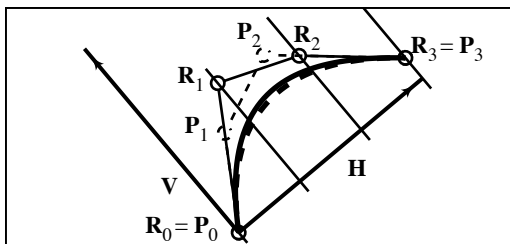


Figure 2: Approximating a Bézier curve with a REC

requires that $(\mathbf{P}_1 - \mathbf{P}_0) \cdot \mathbf{H} > 0$ and $(\mathbf{P}_3 - \mathbf{P}_2) \cdot \mathbf{H} > 0$.

For a given (H, V) coordinate system, we have

$$\mathbf{R}_1 = \mathbf{P}_0 + \frac{\mathbf{H} \cdot \mathbf{H}}{3(\mathbf{P}_1 - \mathbf{P}_0) \cdot \mathbf{H}}(\mathbf{P}_1 - \mathbf{P}_0),$$

$$\mathbf{R}_2 = \mathbf{P}_3 + \frac{\mathbf{H} \cdot \mathbf{H}}{3(\mathbf{P}_2 - \mathbf{P}_3) \cdot \mathbf{H}}(\mathbf{P}_2 - \mathbf{P}_3).$$

The approximation error is bounded by

$$\begin{aligned} & \max_{0 \leq t \leq 1} \|\mathbf{P}(t) - \mathbf{R}(t)\| \\ &= \max_{0 \leq t \leq 1} \|(\mathbf{P}_1 - \mathbf{R}_1)3t(1-t)^2 + (\mathbf{P}_2 - \mathbf{R}_2)3t^2(1-t)\| \\ &\leq \frac{3}{4} \max(\|\mathbf{R}_1 - \mathbf{P}_1\|, \|\mathbf{R}_2 - \mathbf{P}_2\|) = \epsilon_{max}. \end{aligned} \tag{3}$$

This means that for every point on $\mathbf{R}(t)$ there is a point on $\mathbf{P}(t)$ within a distance ϵ_{max} ; it is not a bound on the error in the \mathbf{V} direction. As illustrated in Figure 2, this bound can be quite conservative.

If the approximation error is too large, splitting $\mathbf{P}(t)$ in half using the de Casteljau algorithm, then computing REC approximations for each half will result in an error that is roughly 1/16 of the initial error, and in the limit it exhibits fourth order convergence for an arbitrarily chosen (H, V) coordinate system. It can be shown that if a new \mathbf{V} direction is chosen for each curve segment after each subdivision, with $\mathbf{V} \parallel -\mathbf{P}_0 + \mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_3$, fifth order convergence is possible.

Figure 3 shows a REC approximation of the outline for a character from the PostScript Times-Roman font.

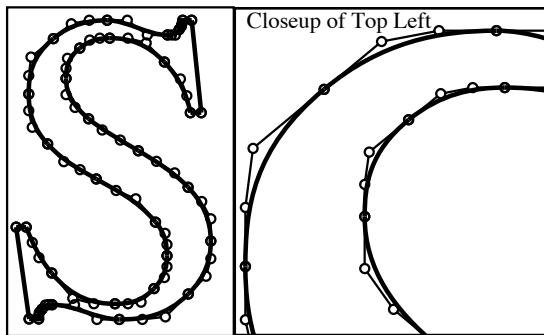


Figure 3: Times-Roman “S” Outline

The initial outline consists of 18 Bézier curves; this approximation contains 31 XRECs and YRECs, and agrees with the original to within the penwidth shown. Even fewer XRECs and YRECs would be needed if the font were designed with them initially.

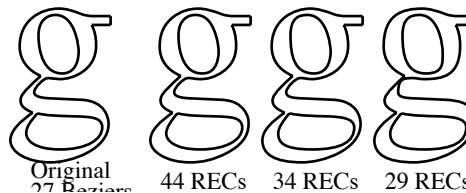


Figure 4: Times-Roman “g” Outline

Figure 4 shows a Times-Roman “g” approximated with different numbers of XRECs and YRECs. This illustrates that a given Bézier outline can usually be approximated with roughly 1.1–1.6 more RECs than Bézier segments.

5 Scan Conversion

We impose a constraint on the slope s of an XREC (or YREC) that $|s| \leq 1$ (or $|s| \geq 1$) to facilitate scan conversion. This requires that the initial Bézier curves be split at points where $|s| = 1$. With that constraint, scan conversion amounts to merely evaluating the YREC (or XREC) at parameter values corresponding to the y (or x) coordinates of the centers of the pixels, as illustrated in Figure 5. The slope constraint assures that the string of pixels is contiguous.

Forward differencing makes this operation very cheap. For a cubic XREC, the x coordinate of the current pixel is found by an integer increment, and the y coordinate can be computed using three additions. This, of course, requires a difference table initialization which consumes an additional eight multiplies for three of the first four pixels, and also two operations to initialize the parameter step size.

This compares quite favorably with the expense of

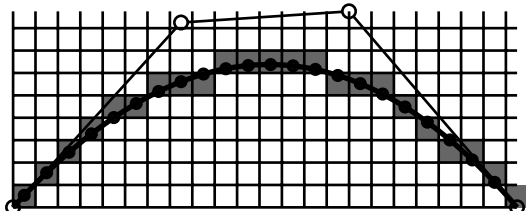


Figure 5: Scan converting an XREC

scan converting fonts whose outlines are defined using standard cubic Bézier curves. Font outlines are most commonly defined using piecewise cubic Bézier curves, Scan conversion for a Bézier curve can be performed by sampling it densely enough so that no two adjacent sample points are more than one pixel apart. This inevitably leads to wasted sampling. And since two polynomial evaluations are required per sample, scan converting a Bézier curve is, on average, at least twice as costly as a REC.

Reference [4] states that the fastest possible scan conversion for fonts is accomplished by approximating the font outline with circular arc segments and then scan converting the circular arcs. The circular arcs require five arithmetic operations per pixel, plus startup cost of three operations per arc [3]. Also, since arcs can only achieve third order approximation convergence, many more of them are needed to approximate a font to a given accuracy than are needed using REC approximation.

Polygonal approximation is also a standard approach to font scan conversion [7]. For a single line segment, the midpoint line algorithm [3] uses two operations per pixel, plus startup cost of seven operations per line segment. Since polygon approximation provides only second order convergence, For the outline in Figure 5, to achieve the approximation accuracy obtained using 31 RECs, one would need about 400 line segments. The polygonalization is also a significant expense, but it only needs to be performed once per font scale while no similar initialization is needed to scale a REC outline.

Another attractive alternative to Bézier curves for font outlines, conic splines [6], can be scan converted using Pitteway's algorithm [5] at an expense of seven arithmetic operations per pixel, plus startup costs.

In summary, RECs appear to be the best way to scan convert fonts.

References

[1] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Boston, 1988.

[2] Iver D. Faux and Michael J. Pratt. *Computational Geometry for Design and Manufacture*. Ellis Horwood, Chichester, 1979.

[3] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 2nd edition, 1990.

[4] Roger D. Hersch. Character generation using grid constraints. *Computer Graphics*, 21(4):243–252, 1987.

[5] M. L. V. Pitteway. Algorithm for drawing ellipses or parabolae with a digital plotter. *Computer Journal*, B10P:282–289, 1967.

[6] Vaughan Pratt. Techniques for conic splines. *Computer Graphics*, 19(3):151–159, 1985.

[7] Bob Wallis. Fast scan conversion of arbitrary polygons. In Andrew Glassner, editor, *Graphics Gems*, pages 92–97. Academic Press, New York, 1990.