COMPUTER GRAPHICS AND IMAGE PROCESSING (1974) 3, (346–349)

SHORT NOTE

An Algorithm for High-Speed Curve Generation

GEORGE MERRILL CHAIKIN
Center for Interdisciplinary Programs, New York University, 26 Stuyvesant Street, New York, New York 10003

Communicated by H. Freeman

Received April 24, 1974

A fast algorithm for the generation of arbitrary curves is described. The algorithm is recursive, using only integer addition, one-bit right shifts, complementation and comparisons, and produces a sequential list of raster points which constitute the curve. The curve consists of concatenated segments, where each segment is smooth and open. The curve may be arbitrarily complex, that is, it may be smooth or discontinuous, and it may be open, closed, or self intersecting.

Implementation of the algorithm in a hardware microprocessor is considered as an extension to incremental plotting devices and other numerically controlled machines.

An extension of the algorithm to generate 3-D curves is described, along with techniques for their application to surface representation, nonlinear interpolation, and direction of numerically controlled milling machines and similar devices.

Consider a curve to be described by four points ($P1$, $P2$, $P3$, $P4$), where the points are represented by integer pairs specifying raster coordinates. The generated curve begins at $P1$ tangent to line $P1$, $P2$; intersects the midpoint of line $P2$, $P3$ tangent to $P2$, $P3$; and terminates at $P4$, tangent to line $P3$, $P4$.

A geometric view of the algorithm sees the quadrilateral described by the four points divided into two triangles at the midpoint of line $P2$, $P3$, each of which is divided in half to form a new quadrilateral. This process is continued until the two points of the triangle which lie on the curve occupy adjacent raster points. The curve is the set of all vectors joining such points. See Fig. 1. Curves of this type are known as quadratic B-splines, and are described by Riesenfeld in [1].

The algorithm is implemented as follows: Push points $P3$ and $P4$ onto a push-down stack, and compute a new $P4$ as the midpoint of line $P2$, $P3$. Compare this new $P4$ to $P1$: Is their distance less than three (raster) units in both the X and Y components? If not, compute:

$$P3 = (P2 + P4)/2,$$
$$P2 = (P2 + P1)/2.$$

We have now generated new points $P2$, $P3$, and $P4$, and apply the procedure again. The procedure is applied recursively until the above distance test succeeds. Then plot the line $P1$, $P4$, replace $P1$ by $P4$, and pop the push-

FIG. 1a. Generated curve.

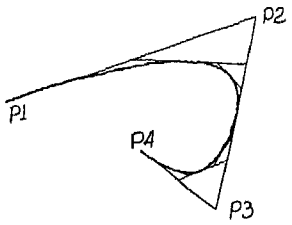

FIG. 1b. Curve with generating quadrilaterals.

down stack, calling the popped points $P2$ and $P4$. Now jump back to the distance test. See Fig. 2.

Note that we test for distances less than three units rather than less than two units in both the X and Y components, that is immediately adjacent raster points. This is done because integer division truncates, and carrying the procedure one level deeper yields degenerate terminal cases where $P1 = P4$ from which it is impossible to "climb up." In order to obtain the smoothest possible curve, then, we can add one bit of precision before invoking the routine, and remove it before plotting.

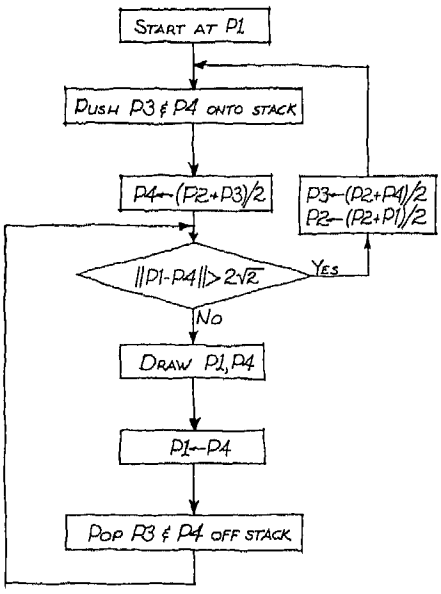

FIGURE 2

More complex curves than those described by four points can be achieved in two ways. First, since upon completion of the algorithm the last point is called $P1$, we need supply only three points ($P2$, $P3$, $P4$) in order to generate each subsequent segment by reinvoking the algorithm. Another (not so obvious) technique involves some loss of generality in return for increased speed. In this case, two points are needed to describe each subsequent segment. These points are "prepushed" onto the push-down stack (in reverse order) before the routine is called, so that when the first segment is finished, the algorithm finds more points in its stack, and continues. In either case it should be noted that concatenated segments will join smoothly if the first point describing the concatenated segment lies on the same line as points $P3$ and $P4$ of the previous segment. See Fig. 3.

Hardware implementation should be straightforward, requiring only integer addition, division by two (a one-bit right shift), complementation and comparison functions, eight registers for the four points, an index register for the stack memory, and the stack memory itself. A 1024×1024 raster requires four ten-word stacks, with one additional word in each stack for each additional curve segment if "prepushing" is to be used.

Concerning speed, n iterations of the algorithm are required to generate $n + 2$ points (the start point, n intermediate points, and the end point). With a hardware implementation which would take full advantage of parallelism, this would mean about 15 instruction cycles per point.

The algorithm described thus far was solely two-dimensional, although it

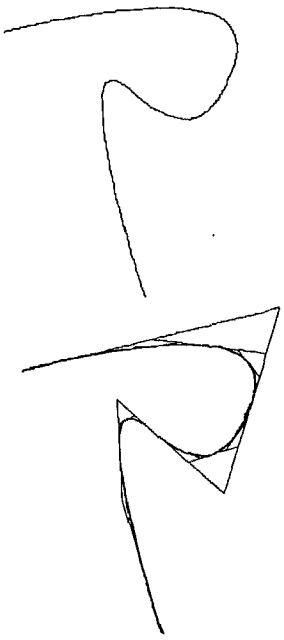

FIGURE 3

clearly can be applied to the two-dimensional projections of curves whose points $P1$, $P2$, $P3$, $P4$ are arbitrary points in three-space, that is, they need not be coplanar. A more challenging problem, however, is to consider the application of directing the movement of three-dimensional incremental devices such as numerically controlled milling machines, for example. A straightforward extension of the algorithm so that it handles coordinate triples instead of coordinate pairs will yield a smooth curve in three-space, but the curve will lie in two distinct planes, $P1$, $P2$, $P3$ and $P2$, $P3$, $P4$, with the curve passing (smoothly) from one plane to the other at the midpoint of line $P2$, $P3$.

Surface representation may be accomplished by iterative application of the curve generator, where successive parameters describing the curve at each iteration are obtained either by linear interpolation or by using the generator to provide points to a more deeply nested instance of itself. For example, four curve generators running in step could be used to provide successive values of $P1$, $P2$, $P3$ and $P4$ to a fifth generator which is actually drawing curves on the surface. Alternatively, a generated curve could be used to define the edge of a surface while two others are used to supply values of $P2$ and $P3$. These and other applications are currently under study.

ACKNOWLEDGMENTS

The research described in this report was supported by the Information Systems Branch, Office of Naval Research, under contract N00014-67-A-0467-0010, Professor Herbert Freeman, Principal Investigator. The original idea for the algorithm was proposed by Mr. Ronald Bianchini. It was subsequently developed by Mr. Bianchini, Mr. V. Prestianni, Prof. J. T. Schwartz, and the author at the AEC Computing Center at New York University.

REFERENCES

1. RIESENFELD, R. Applications of B-spline Approximation to Geometric Problems of Computer-Aided Design, Computer Science Dept., University of Utah, Salt Lake City, Utah UTEC-CSc-73-126, 1973.