New Hermite cubic interpolator for two-dimensional curve generation

R. Szeliski, B.Eng., M.Sc., and M.R. Ito, Ph.D.

Indexing term: *Computer graphics*

Abstract: The problem of generating a smooth two-dimensional curve through a set of sample points is examined. Such curve generation techniques can be used in curve coding for transmission (e.g. 'Telewriting') and in curve design. The paper examines some parametric interpolators which will generate such curves, with emphasis on techniques based on the Hermite cubic (or cubic subspline) interpolator. A new method is presented which gives the interpolator a performance similar to that obtained with finite impulse response (FIR) filtering. Comparisons are made with existing techniques using both visual results and computational complexity measures. The new method is found to have better locality, smoothing characteristics and a lower computational cost than alternative approaches.

1 Introduction

The problem of generating a smooth curve through an ordered set of two-dimensional points is one that often arises in computer graphics. The curves generated can represent handwriting, sketching, contour lines or other graphical objects. Recent papers have presented such interpolating methods based on either low-pass filtering [1] or cubic interpolation [2, 3]. This paper presents a new method based on Hermite cubic (first-derivative continuous) interpolation that has the desirable features of both cubic interpolators and of low-pass filters while being computationally simpler than either of these approaches. The method is designed for non-uniformly spaced points ('non-uniform grids'); a simplified version, which can be used with uniform grids, is also presented. Several alternative interpolating techniques are reviewed and compared with the new method.

The requirements for a suitable curve generator include simplicity of specification, smoothness, ease of calculation and locality. Simplicity of specification is achieved by requiring that the curve must interpolate (pass through) the given data points. This requirement allows the method to be used to reconstruct time or space sub-sampled curves such as handwriting. The smoothness criterion requires that the generated curves be visually pleasing when displayed on the output device. The locality requirement means that only a few neighbouring points are needed to calculate a curve segment. Ease of calculation requires that the algorithm be simple and fast enough to be implemented on a microprocessor. The combination of these latter two requirements makes the method useful for the real-time coding of line graphics (such as sketches), since the reconstructed curve can be rapidly generated with a delay of only a few points. The locality criterion also ensures that the overall curve is not sensitive to the placement of any one sample point, and also facilitates local curve editing.

Previous work in curve generation has been in the field of curve design, with commonly used methods such as Bezier curves [4] requiring interactive adjustment of the curve. Dube [5] presented such a curve design method based on the same Hermite cubic basis which is used here. However, his interpolator uses the full cubic spline solution as a starting point, and is thus not sufficiently local. Shlien and Allard [1] have presented an interpolating method based on low-pass filtering which gives results visually similar to the ones presented here. Their method, however, requires large window sizes to eliminate certain artifacts. Renner and Pochop [2] have proposed a local cubic interpolator suitable for graphics which have a mixture of straight line and curved segments. Harada and Nakamae [3] have recently presented another four-point cubic interpolator. Both methods are similar to the new method presented here but do not generate curves that are as smooth. These and other interpolating methods are discussed later in the paper.

The primary application for the new interpolating method presented here is for the reconstruction of sub-sampled curves such as digitised handwriting. The method can thus be directly applied to the coding of sketches for instantaneous transmission ('Telewriting'), by sampling the writing at a fixed frequency and using the cubic interpolator to regenerate the curve. The interpolator is sufficiently flexible so that a controlled placement of the sample points would also allow it to be used for the coding of other line-drawing graphics such as contour lines or typeface fonts. The simplicity of the method and its strict locality also make it a suitable candidate for a new geometric primitive for Videotex (the PLP 'Spline' primitive which is yet to be standardised [7]). The emergence of specialised chips such as the Geometry Engine [8] which incorporate the generation of cubics should make this cubic interpolator attractive in general computer graphics applications.

The remainder of the paper is structured as follows. Section 2 presents the parametric formulation for the curve based on the Hermite cubic functions. Section 3 reviews a variety of methods for determining the curve derivatives ('slopes') at each sample point which are necessary to specify completely the interpolator. The new interpolator presented in this paper, which is based on a piecewise cubic approximation to the low-pass filter, is introduced. Section 4 compares the new method with true low-pass filtering and also gives a simplified version of the algorithm which can be used for uniform grids. Section 5 discusses some methods for implementing the interpolator using integer arithmetic on very simple systems (such as a microprocessor). Section 6 discusses the locality against smoothing trade off inherent in the new interpolator and presents some of the visual results obtained. The various interpolating methods examined in the paper are compared. The final Section summarises the advantages of the new interpolating method and some of its possible applications.

Paper 4860E (C1), received 12th March 1985

Mr Szeliski is with the Department of Computer Science, Carnegie-Mellon University, Schenley Park, Pittsburgh, Pennsylvania 15213, USA. Dr Ito is with the Department of Electrical Engineering, University of British Colombia, Vancouver, B.C., Canada V6T 1W5

2 Hermite cubic interpolation

The interpolator used to construct the curve generally has a parametric representation, i.e. the curve is a two-dimensional function of an underlying parameters s,

$$\boldsymbol{x} = \boldsymbol{f}(s) \tag{1}$$

which passes through the collection of sample points $\{\boldsymbol{x}_i\}$,

$$\boldsymbol{x}_i = \boldsymbol{f}(s_i) \tag{2}$$

This notation allows for easy extension to three dimensions [4]. The parametric notation also gives an interpolator that is *isotropic*, i.e. invariant with respect to orientation. The parameter s is usually related to the arc length along the curve. However, using the chord length

$$s_{i+1} = s_i + d_i$$

where

$$d_i = |\boldsymbol{d}_i| = |\boldsymbol{x}_{i+1} - \boldsymbol{x}_i| \tag{3}$$

with s varying linearly between control points, will give equally good results [9]. A further simplification, $s_i = i$, can be made if the point spacing is fairly uniform (see Section 4).

The interpolation requirement precludes the use of some functions such as quadratic B-splines [10] which are otherwise suitable curve generators. One simple class of functions that both interpolates and can be made sufficiently smooth are the cubic splines and sub-splines. These functions are piecewise cubic, i.e. on any interval (s_i, s_{i+1}) the function is a cubic

$$\boldsymbol{f}_i(s) = \boldsymbol{a}_{3i}s^3 + \boldsymbol{a}_{2i}s^2 + \boldsymbol{a}_{1i}s + \boldsymbol{a}_{0i} \tag{4}$$

For sufficient smoothness, we require that the curve be first derivative (C^1) continuous. Such curves are variously known as sub-splines or Hermite cubics [4]. They are completely defined by the control points $\{\boldsymbol{x}_i\}$ and the curve derivatives ('slopes') at these points $\{\dot{\boldsymbol{x}}_i\}$ (Fig. 1). Each

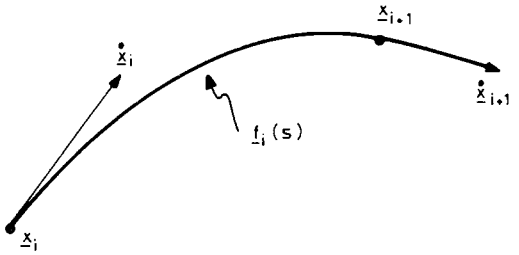

Fig. 1 *Cubic segment defined by its end points and end-point derivatives*

cubic segment can also be represented as a linear combination of four basis or 'blending' [5] functions weighted by the end points and the end-point derivatives

$$\boldsymbol{f}_i(s) = \boldsymbol{x}_i\phi_0(t) + \dot{\boldsymbol{x}}_i\phi_1(t) - \dot{\boldsymbol{x}}_{i+1}\phi_1(1-t) + \boldsymbol{x}_{i+1}$$

$$\phi_0(1-t), \qquad t = \frac{s - s_i}{d_i} \in [0, 1] \tag{5}$$

These basis functions, the Hermite Cubic basis functions, are

$$\begin{aligned} \phi_0(t) &= 2t^3 - 3t^2 + 1 \\ \phi_1(t) &= t(1-t)^2 \end{aligned} \tag{6}$$

and are shown in Fig. 2.

Since the sample points $\{\boldsymbol{x}_i\}$ are given, only the slopes $\{\dot{\boldsymbol{x}}_i\}$ need to be determined in order to specify completely the interpolator. The slope values $\dot{\boldsymbol{x}}_i$ are the true derivative values of the *interpolator* $\boldsymbol{f}(s)$, but can only be an estimate of the derivatives of the original curve. The overall Hermite cubic interpolant function $\boldsymbol{f}(s)$ is thus composed of piecewise cubic segments $\boldsymbol{f}_i(s)$ with first-derivative continuity across interval boundaries.

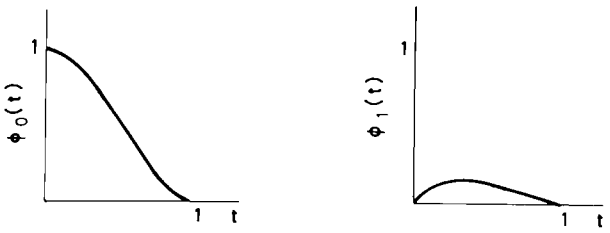

Fig. 2 *Hermite cubic basis (or 'blending') functions*

3 Slope determination

To generate the cubic interpolator, the slopes $\{\dot{\boldsymbol{x}}_i\}$ must be determined. One common way to specify the slopes is to require second derivative (C^2) continuity. The resulting system of tri-diagonal equations can then be solved using various iterative or direct methods [11, 12], with the resulting interpolator being known as the full cubic spline. However, this calculation requires all of the points on the curve to be known, and it is thus not sufficiently local to be used for real-time reconstruction.

The slopes can also be adjusted interactively. Methods such as Bezier curves [4] have been designed to do this naturally by specifying additional control points. This is not a satisfactory approach if the curve generation is to proceed automatically from a set of sample points without operator intervention. What is required instead is a method to estimate the individual slopes using only a few of the neighbouring points. Several such methods are discussed in this section, leading to a new method based on a piecewise-cubic approximation to finite impulse response (FIR) filtering.

The simplest method for determining the slope locally is to use a parabola through a sample point and its left and right neighbours to determine the slope at the point (modifications can be made for the end points of the curve). The parabolic equation

$$\boldsymbol{g}(s) = \frac{\ddot{\boldsymbol{x}}_i}{2}(s - s_i)^2 + \dot{\boldsymbol{x}}(s - s_i) + \boldsymbol{x}_i$$

with

$$\boldsymbol{g}(s_{i-1}) = \boldsymbol{x}_{i-1} \text{ and } \boldsymbol{g}(s_{i+1}) = \boldsymbol{x}_{i+1} \tag{7}$$

can be solved to yield

$$\dot{\boldsymbol{x}} = \frac{d_{i-1}\boldsymbol{m}_{i,i+1} + d_i\boldsymbol{m}_{i-1,i}}{d_{i-1} + d_i} \tag{8}$$

The value $\boldsymbol{m}_{ij}$ is the *divided difference* of the curve (Fig. 3),

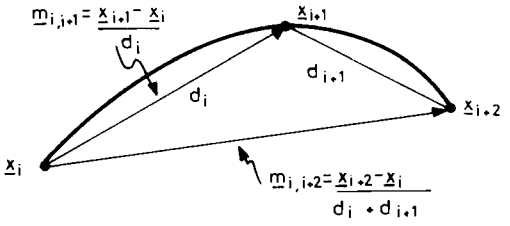

Fig. 3 *Divided differences of a curve used to calculate the slopes*

and is defined as the slope of a line connecting two sample points:

$$\boldsymbol{F}[s_i, s_j] \triangleq \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{s_i - s_j} \triangleq \boldsymbol{m}_{ij} \text{(for short)} \tag{9}$$

The interpolator resulting from this parabolic fit, known as the Bessel interpolator, is quite simple, but does not smooth as well as methods that use more of the neighbouring points. It is, in effect, a four-point interpolator, since the points x_{i-1}, x_i, x_{i+1} and x_{i+2} are needed to define the two end slopes $\dot{x}_i$ and $\dot{x}_{i+1}$ for the segment $f_i(s)$.

Higher-order polynomials (such as quartics) can be fitted to determine the slope, but the exact solution rapidly becomes complex. An assumption of near-uniform spacing can be used to obtain some simple results which take the form of a sum of divided differences weighted by fixed rational numbers (see 9.1 Appendix A). For example, the result for the quartic is

$$x_i = -\frac{1}{6} m_{i-2,i} + \frac{2}{3} m_{i-1,i} + \frac{2}{3} m_{i,i+1} - \frac{1}{6} m_{i,i+2} \quad (10)$$

It can be shown that the solutions to these polynomial fits are equivalent to the first step in the iterative solution of the full cubic spline [9]. The results using these simplified polynomial fits compare favorably with the Bessel interpolator (Fig. 4). Since more points are used to determine the slopes, better smoothing is achieved.

Fig. 4 *Cubic interpolator using parabolic and quartic fits*

a Parabolic
b Quartic
x marks the sample points
The quartic fit produces a smoother curve since it uses more points (6 instead of 4)

Other approaches to determining the slope can be based upon fitting a circular arc through three neighbouring points [6, 13]. However, the resulting curves are too rounded to be used without additional control specification (Fig. 5), and are thus not suitable for interpolation.

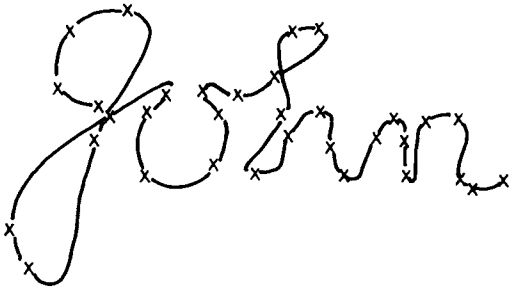

Fig. 5 *Cubic interpolator using arc circular arc fit. The result is too rounded*

The method presented by Rennor and Pochop [2] uses four neighbouring points to determine the slope. Straight line segments are preserved using an approach based on Akima's method [14]. The method used by Harada and Nakamae is very similar to the Bessel interpolant, since their derivative value is 6/7 times that obtained in eqn. 8 [3]. A comparison of the performance of these last two approaches with the new method is given in the Section 6.

The new method for slope determination developed in this paper approximates the response of the low-pass filter presented by Shlien and Allard [1]. Their FIR filter is an interpolator whose *impulse response* is a windowed sinc function (Fig. 6)

$$h(t) = w(t)\frac{\sin(\pi t)}{\pi t}, \quad \begin{array}{l} w(t) \geqslant 0 \text{ for } |t| \leqslant p \\ w(t) = 0 \text{ for } |t| > p \end{array} \quad (11)$$

The window function $w(t)$ is a positive symmetric function of *window size* $(2p + 1)$ which is used to limit the duration

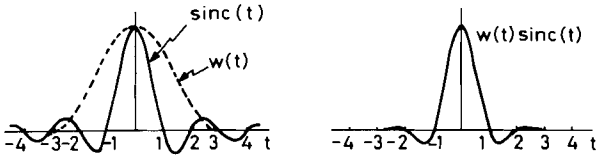

Fig. 6 *FIR low-pass filter response: windowed sinc function, $p = 3$*

of the impulse response and to improve the frequency response. A smaller window allows for faster calculation but yields poorer smoothing. The improper choice of window shape can lead to artifacts in the filtered output (Fig. 7). The interpolated curve is a sum of impulse

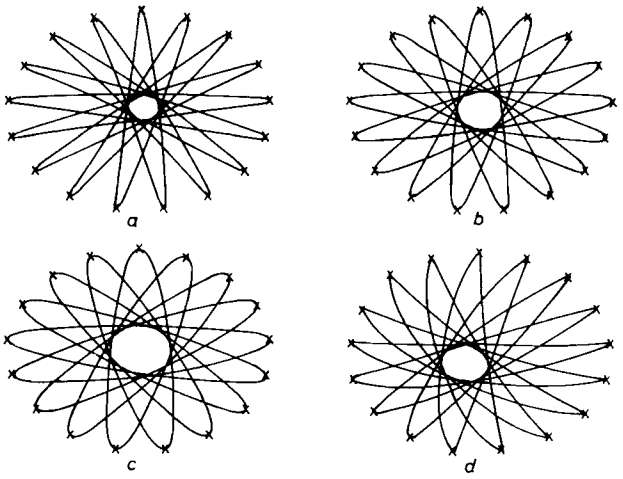

Fig. 7 *Effect of window size and shape on FIR filter*

a c Hanning window $p = 2, 3, 4$, respectively. The larger window sizes give more smoothing
d Bartlett (triangular) window $p = 3$. Artifacts are seen due to the nonlinear precision

response functions weighted by the sample points

$$f(s) = \sum_i x_i h(s - i) \quad (12)$$

where the parameter s is assumed to be on a uniform grid $(s_i = i)$. From this equation, the slope of the interpolator at a sample point is

$$\dot{x}_i = \sum_{\substack{j=-p \\ j\neq 0}}^{p} (-1)^{j-1} w(j) x_{i+j}/j \quad (13)$$

The FIR interpolator can be extended to non-uniform grids by using an asymptotic expansion for sinc (t) [9]. The corresponding slope value can be derived as

$$\dot{x}_i = \sum_{\substack{j=-p \\ j\neq 0}}^{p} (-1)^{j-1} w(j) m_{i,i+j} \quad (14)$$

where $m_{i,i+j}$ is the divided difference defined in eqn. 9. This new result produces an interpolator that works on non-uniform grids while preserving the filtering characteristics of the FIR filter. The choice of window shape and size will determine the smoothing (filtering) characteristics of this new ('sinc-type') interpolator. To ensure that the interpolator has linear precision (is straight-line preserving), it is required that

$$\sum_{\substack{j=-p \\ j\neq 0}}^{p} (-1^{j-1} w(j) = 1 \quad (15)$$

One such linearity preserving window that has been found by the authors to give good results is the Hanning

window, $w(j) = \cos^2(j\pi/2p)$. An example of the resulting slope estimator for $p = 3$ is

$$\dot{x}_i = -\frac{1}{4}\boldsymbol{m}_{i-2,i} + \frac{3}{4}\boldsymbol{m}_{i-1,i} + \frac{3}{4}\boldsymbol{m}_{i,i+1} - \frac{1}{4}\boldsymbol{m}_{i+1,i+2} \quad (16)$$

which is quite similar in form and performance to the simplified quartic fit (eqn. 10).

Rewriting the above equations to take advantage of the symmetry in the window function, we obtain the final formulation for the new sinc-type interpolator, which is a piecewise-cubic function (eqn. 5) with the slopes at the sample points defined by

$$\dot{x}_i = \sum_{j=1}^{p} (-1)^{j-1} w_j \{\boldsymbol{m}_{i,i-j} + \boldsymbol{m}_{i,i+j}\} \quad (17)$$

and the weighting coefficients $\{w_j\}$ satisfying

$$\sum_{j=1}^{p} (-1)^{j-1} w_j = \frac{1}{2} \quad \text{and} \quad w_j \geq 0, j = 1, p \quad (18)$$

4 Comparison with low-pass filtering

The results obtained visually with this new sinc-type cubic interpolator are similar to those obtained with true low-pass filtering. FIR filtering uses a sum of windowed impulse response functions to generate the interpolated curve (see eqn. 12). The windowing ensures that only a few points in the neighbourhood of the sample point are considered in the summation, ensuring the locality of the interpolant.

The greatest advantage of the cubic interpolator over the FIR filter is that it has linear precision (i.e. it preserves straight lines), even for non-uniformly spaced points. This is only approximately true of the FIR filter, even for large window sizes, and leads to artifacts for smaller windows (Fig. 7). The choice of a small window size ensures greater locality for the interpolator and allows the reconstruction of the curve to proceed with less delay. In practice, a window size of 7 ($p = 3$) has been found to give a good balance between sufficient smoothing and good locality (see eqn. 16).

If the sample point spacing is fairly uniform, as is usually the case for curves which were time sampled at a fixed frequency, then a simplification can be made to the interpolator given in eqn. 17. Using the assumption that $s_i = i$, a simplified 'normalized' slope estimate $\hat{\dot{x}}_i$ can be calculated as

$$\hat{\dot{x}}_i = \sum_{j=1}^{p} w_j^* \cdot (x_{i+j} - x_{i-j}) \quad (19)$$

with

$$w_j^* = (-1)^{j-1} w(j)/j \quad (20)$$

The required linearity condition becomes

$$\sum_{j=1}^{p} j w_j^* = \tfrac{1}{2} \quad (21)$$

An example of the normalised derivative for a Hanning window ($p = 3$) is

$$\hat{\dot{x}}_i = \tfrac{3}{4}\{x_{i+1} - x_{i-1}\} - \tfrac{1}{8}\{x_{i+2} - x_{i-2}\} \quad (22)$$

The 'normalised' derivative must still be converted to the actual derivative by

$$\begin{aligned} \dot{x}_{i-} &= \hat{\dot{x}}_i \cdot d_i \\ \dot{x}_{i+} &= \hat{\dot{x}}_i \cdot d_{i+1} \end{aligned} \quad (23)$$

Here, the derivative of the curve becomes discontinuous at the control point if $d_i \neq d_{i+1}$ ($\dot{x}_{i-}$ is the derivative from the left $\dot{x}_{i+}$ from the right). However, since the derivative is *direction continuous*, sufficiently smooth curves are still produced if $d_i \simeq d_{i+1}$[4, 9].

For this simplified interpolator, the slope becomes a simple linear combination of neighbouring points. It now becomes possible to create an 'impulse response' for the interpolator by superimposing the contributions of several neighbouring points to one impulse value in the manner of eqn. 12. Fig. 8 shows the resultant response, which is

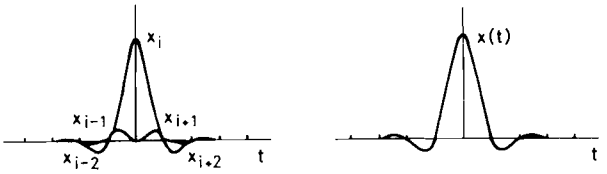

Fig. 8 *Impulse response of simplified cubic interpolator, calculated by superimposition*

The response is comparable to Fig. 6 and differs the most near the tail ends

similar to the true windowed sinc function (Fig. 6). The sinc-type cubic interpolator has similar smoothing characteristics to the FIR digital filter, and also has a better linearity. As well, the calculators of the cubic interpolator is faster, as is shown in the following Section.

5 Implementation

The calculation of the Hermite cubic interpolator for displaying a curve consists of two separate parts. The first is the calculation of the slopes at each control point. This can be done on-line, since only a small number of points have to be accumulated. If the divided differences are used to calculate the slopes (eqn. 17), then the distance values $\{d_i\}$ must also be calculated and retained. Because fixed rational weights are used, the slope calculation is easily implemented using integer arithmetic. The calculations are further simplified if simple differences are used (eqn. 19). For example, the slope for the simplified interpolator using a Hanning window given in eqn. 22 can be calculated using four add/subtracts and three arithmetic shifts.

One special case that arises in the slope calculation is the treatment of end points. Several approaches are possible [9]; one of the simplest is to repeat the end points before and after the curve, which in effect sets the divided differences to zero beyond the curve. Thus, if the curve has n control points, $\{x_i \mid i = 1, n\}$, then we define

$$\begin{aligned} x_i &= x_1 \quad \text{for} \quad i < 1 \\ x_i &= x_n \quad \text{for} \quad i > n \\ \boldsymbol{m}_{ij} &= 0 \quad \text{for} \quad j < 1 \text{ or } j > n \end{aligned} \quad (24)$$

This approach has been found to produce satisfactory results for the curve near its end points.

The other special case that arises is the occurrence of repeated points. Such double points do not arise often in sampled hand-drawn curves, except near corners or points of pen stoppage. One simple way of treating double points is to extend the divided difference definition (eqn. 7) to be zero at such points

$$\boldsymbol{F}[s_i, s_j] = \boldsymbol{m}_{ij} \triangleq 0 \text{ when } s_i = s_j \quad (25)$$

A more useful approach, however, is to treat the repeated point as an end point, and to calculate the left and right derivatives separately (ignoring contributions before or after the double point). This produces a discontinuity in

curvature, and can be used to place corners, sharp bends, or straight line segments, at selected locations along the curve (Fig. 9). This added flexibility enhances the interpolator for use in curve design.

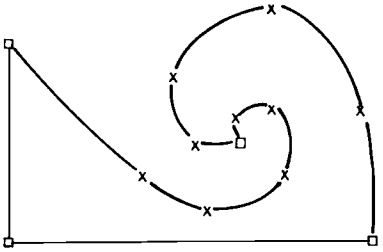

Fig. 9 *Corner, sharp bends and straight lines using double points*
Cubic interpolator $p = 3$
x: single point
□: double point

The second part of the interpolator calculation is the generation of each cubic segment once its end point derivatives are known. One simple approach is to use a look-up table for the basis functions ϕ_0 and ϕ_1 (eqn. 6). This is easily implemented using integer arithmetic (four multiplies per output point), and compares favourably with a look-up table implementation of FIR filtering ($2p$ multiplies).

If using multiplications is computationally too expensive (as for some simple microprocessors or for direct hardware implementation), a finite algorithm can be used [9]. This approach, also known as the digital differential analyser (DDA), can be implemented using integer add-and-shift-only arithmetic (see 9.2 Appendix B for derivation). An implementation for a 16-point cubic segment would require an average of 7.5 adds and 4.5 shifts per output point, and could also be implemented directly in hardware. An existing hardware cubic generator, such as the one incorporated in the Geometry Engine [8], could also be used for displaying the cubic segments. If a raster device is being used for output, then an incremental algorithm could be designed to display the curve one pixel at a time.

The computational requirements for the sinc-type cubic interpolator are thus quite low. Each slope can be calculated as a weighted combination of neighbouring sample points, and the cubic segment can be calculated using either a look-up table, a finite difference algorithm, or direct hardware implementation.

6 Results

The new sinc-type cubic interpolator presented here can be looked at as a family of possible interpolators, since both the window shape and window size can be varied. The window shape determines the weighting coefficients used in eqns. 17 and 18, and has not been found to be a critical factor in the performance of the interpolator. Compare, for example, eqns. 10 and 16, which use different weighting factors, and whose responses (shown in Figs. 4 and 11) are nearly identical. In practice, the Hanning window, $w(j) = \cos^2(j\pi/2p)$, has been found to give good results (with linear precision guaranteed). The choice of window size gives a larger variation in performance. Better smoothing is achieved by using larger windows (Fig. 10), while a smaller window size gives better locality, and hence less delay in reconstruction.

The sinc-type interpolator can be compared visually (Fig. 11) to the FIR filter [1], the cubic interpolator due to Renner and Pochop [2], and Harada and Nakamae's cubic interpolator [3]. Compared to the new interpolator, the FIR filter exhibits some artifacts due to the limited window size, and both Renner and Pochop's and Harada and Nakamae's interpolators are not as smooth, since they are only four-point interpolants.

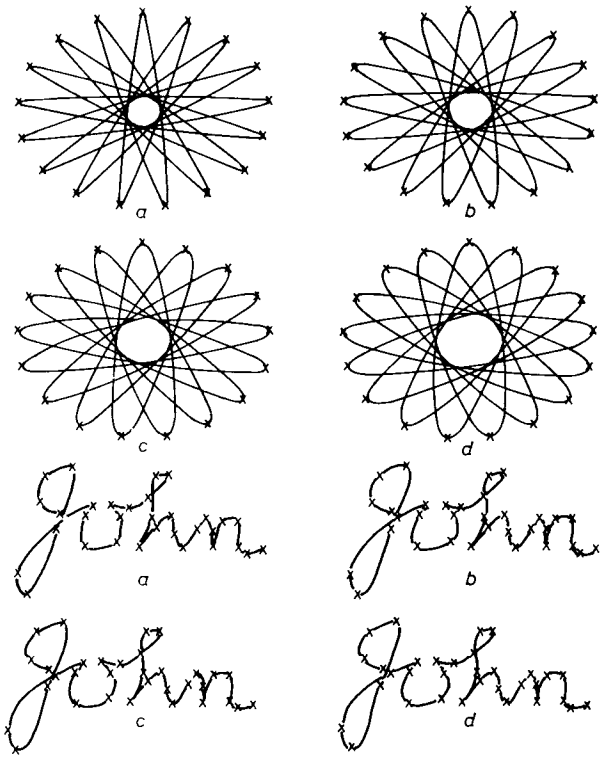

Fig. 10 *Cubic interpolator with Hanning window*
a $p = 2$
b $p = 3$
c $p = 4$
d $p = 5$
The smoothing is better for larger windows

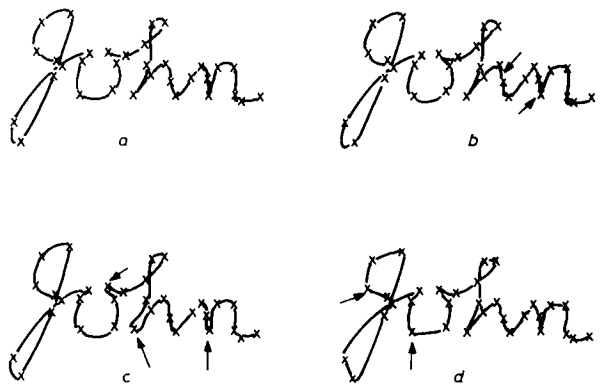

Fig. 11 *Comparison of four interpolators*
a New cubic interpolator, uniform grid ($p = 3$)
b FIR filter ($p = 3$)
c Renner and Pochop's interpolator
d Harada and Nakamae's interpolator
The arrows indicate some visual artifacts

The alternate curve generation methods can also be compared in terms of computational complexity. Once the slopes have been calculated, the displayed points in the cubic segment (for the cubic interpolators) can be calculated using four-table look-up/multiplies per output point, as compared to $2p$ such operations for the FIR filter. The three cubic-based methods can ultimately use the same cubic segment generator, so that they must be compared on the basis of parameter calculation. Compared to Renner and Pochop's tangent and parameter interval calculation [2], the slope determination for the new method is significantly simpler. Harada and Nakamae's method, a

simple parabolic fit, is very similar to the non-uniform grid cubic interpolator with $p = 2$. The slope calculation for the new interpolant is slightly simpler since it uses fixed weights to combine the divided differences (compare eqns. 17 and 8).

The new interpolator presented in this paper can also be used for curve design. While it does not have as much flexibility as a general design curve (such as a Bezier curve) where each individual slope can be adjusted, it offers greater simplicity in specification, since all control points lie *on* the curve. Moreover, extra points can be inserted or removed with only local effect, and double points can be used to place sharp bends in the curve (Fig. 12).

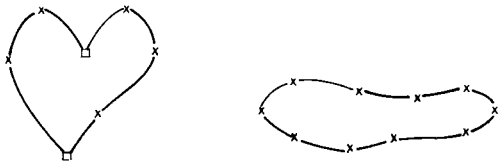

Fig. 12 *Curve design using the new cubic interpolator. Curvature control is effected by using more points*

The interpolator presented here was originally devised for the real-time coding of handwriting and sketches. The quality of the coding depends on the sampling rate, and using the cubic interpolator allows the use of lower rates than would be required with straight-line interpolation. Fig. 13 shows the same curve sampled at different *fixed* rates using the cubic interpolator for reconstruction.

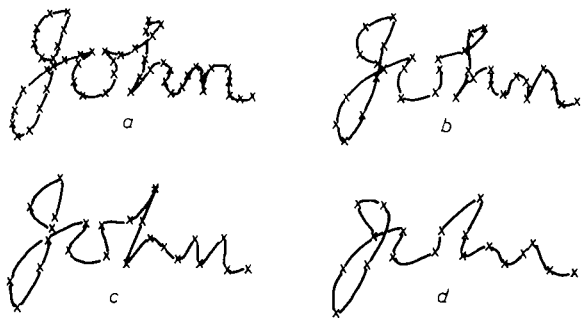

Fig. 13 *Cubic interpolation (Hanning window, $p = 3$) with different decimation (sub-sampling) rates*

a Decimation 3
b 5
c 7
d 9

The visual and computational results presented in this section show that the new sinc-type cubic interpolator has two major advantages over other methods. It is less computationally expensive (simpler), and it has desirable smoothing characteristics while maintaining linear precision.

7 Conclusions

This paper has examined the use of parametric cubic subsplines (Hermite cubics) for two-dimensional curve generation/reconstruction. The Hermite cubic approach encompasses a family of interpolators, with the different interpolants being defined by the choice of slope (derivative) estimator. In particular, a new local slope estimator has been presented which uses a linear combination of a few divided differences, and which has the same smoothing characteristics as the FIR filter. The six-point interpolant, which uses a combination of four divided differences (eqn. 16), has been found to give a good balance between calculation time and smoothing. A simplified algorithm which calculates the slope as a linear combination of a few neighbouring points can be used on uniform grids (where the point spacing is fairly regular). Once the end-point derivatives of a cubic curve segment are known, the points along the segment can be calculated for display, using either a look-up table or a finite difference (DDA) algorithm.

The new cubic interpolator has several advantages over other methods. It is strictly local (as opposed to the full cubic spline), and thus requires less storage, and can be calculated on-line with only a few points' delay. It has smoothing characteristics similar to the FIR filter (better smoothing than other local cubic methods such as Renner and Pochop's), and also has linear precision, which eliminates the artifacts associated with the FIR filtering approach. The simplicity of calculation, both in slope estimation and cubic segment generation, is another advantage. The interpolator is also isotropic (insensitive to orientation), and can be extended to higher-dimensional curves.

The curve generation method presented has several potential applications. It can be used with the coding of sketches and handwriting for transmission (Telewriting) or for storage. The coding of other curves such as contour lines or typeface fonts is also possible, and is enhanced by the simplicity of specification. The method can be used for general curve design, where control of the curvature is not critical. As such, it is a good candidate for selection as the PLP spline primitive used with Videotex coding. In summary, the simplicity of the method, its strict locality and good smoothing characteristics make it a good choice for two-dimensional curve generation.

8 References

1 SHLIEN, S., and ALLARD, P.: 'A FIR filtering approach for the generation of smooth curves on a graphics terminal', *Comput. Graphics & Image Processing*., 1981, **17**, pp. 269–280
2 RENNER, G., and POCHOP, V.: 'A new method for local smooth interpolation', EUROGRAPHICS' 81 Conference, (North-Holland, New York, 1981)
3 HARADA, K., and NAKAMAE, E.: 'An isotropic four-point interpolator based on cubic splines', *Comput. Graphics & Image Process.*, 1982, **20**, pp. 283–287
4 BOEHM, W.: 'On cubics: a survey', *ibid.*, 1982, **19**, pp. 201–226
5 DUBE, R.P.: 'Preliminary specification of spline curves', *IEEE Trans.*, 1979, **C–28**, pp. 286–290
6 KNUTH, D.E.: 'METAFONT, a system for alphabet design' (Digital Press, Bedford, Mass., 1979)
7 'Presentation Level Protocol, VIDEOTEX Standard', American Telephone and Telegraph Company, 1981
8 CLARK, J.H.: 'The geometry engine: A VLSI geometry system for graphics', *Comput. Graphics*, 1982, **16**, pp. 127–134
9 SZELISKI, R.: 'Real time coding of hand drawn curves', M.Sc. Thesis, University of British Columbia, Vancouver, Canada 1981
10 LAPALME, R.S.: 'An interactive data reduction technique for line drawings', M.Eng. Thesis, Royal Military College, Kingston, Canada 1977
11 LIOU, M-L.: 'Spline fit made easy', *IEEE Trans.*, 1976, **C–25**, pp. 522–527
12 MOSS, R., and LINGÅRD, A.: 'Parametric spline curves in integer arithmetic designed for use in microcomputer controlled plotters', *Comput. & Graphics*, 1979, **4**, pp. 51–61
13 MIDGELEY, J.E.: 'Isotropic four-point interpolation', *Comput. Graphics & Image Process.*, 1979, **9**, pp. 192–196
14 AKIMA, H.: 'A new method for interpolation and smooth curve fitting based on local procedures', *J. ACM*, 1970, **17**, pp. 589–602

9 Appendix

9.1 Appendix A

This appendix derives the slope estimate obtained by using a quartic polynomial fit. An assumption of quasi-uniform spacing is made in order to simplify the solution. The

quartic polynomial

$$g_i(s) = a(s - s_i)^4 + b(s - s_i)^3 + c(s - s_i)^2 + \dot{x}_i(s - s_i) + x_i$$

is made to pass through the four neighbouring points

$$g_i(s_j) = x_j, \quad j = i - 2, i - 1, i + 1, i + 2$$

In matrix notation, we have

$$\begin{bmatrix} (d_{i-2} + d_{i-1})^4 & -(d_{i-2} + d_{i-1})^3 & (d_{i-2} + d_{i-1})^2 & -(d_{i-2} + d_{i-1}) \\ d_{i-1}^4 & -d_{i-1}^3 & d_{i-1}^2 & -d_{i-1} \\ d_i^4 & d_i^3 & d_i^2 & d_i \\ (d_i + d_{i+1})^4 & (d_i + d_{i+1})^3 & (d_i + d_{i+1})^2 & (d_i + d_{i+1}) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} x_{i-2} - x_i \\ x_{i-1} - x_i \\ x_{i+1} - x_i \\ x_{i+2} - x_i \end{bmatrix}$$

or

$$\begin{bmatrix} -(d_{i-2} + d_{i-1})^3 & (d_{i-2} + d_{i-1})^2 & -(d_{i-2} + d_{i-1}) & 1 \\ -d_{i-1}^3 & d_{i-1}^2 & -d_{i-1} & 1 \\ d_i^3 & d_i^2 & d_i & 1 \\ (d_i + d_{i+1})^3 & (d_i + d_{i+1})^2 & (d_i + d_{i+1}) & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} m_{i-2,i} \\ m_{i-1,i} \\ m_{i,i+1} \\ m_{i,i+2} \end{bmatrix}$$

This general matrix inversion must be performed for each point if a non-uniform grid is assumed. However, if a uniform grid $s_i = i$ (and hence $d_i = 1$) is assumed for the left-hand side, a simplified matrix results

$$\begin{bmatrix} -8 & 4 & -2 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} m_{i-2,i} \\ m_{i-1,i} \\ m_{i,i+1} \\ m_{i,i+2} \end{bmatrix}$$

This can be solved for

$$x_i = \tfrac{2}{3}\{m_{i-1,i} + m_{i,i+1}\} - \tfrac{1}{6}\{m_{i-2,i} + m_{i,i+2}\}$$

This approach is referred to as 'quasi-uniform' grid spacing, since some of the non-uniformity in spacing is reflected in the divided differences $m_{i,j}$. Since the slope is a linear combination of these divided differences, the resulting interpolator has *linear precision*, i.e. it reproduces straight lines.

The approach given here can be extended to higher-order polynomials, but is of limited usefulness. The results are in all cases similar in form to the sinc-type cubic interpolator

$$\dot{x}_i = \sum_{j=1}^{m} (-1)^{j-1} w_j \{m_{i,i-j} + m_{i,i+j}\}$$

but with a resulting performance which is poorer than that obtained by using a Hanning window, $w(j) = \cos^2(j\pi/2m)$.

9.2 Appendix B

The calculation of the cubic segment once its end-point derivatives are known can be implemented using finite difference equations (also known as the digital differential analyser). To generate a cubic on the interval [0, 1] knowing the end-point partials $g(0)$, $g(1)$, $g'(0)$ and $g'(1)$, the following set of difference equations can be used:

$$g''(t + \Delta t) = g''(t) + \Delta t \cdot g'''(0)$$

$$g'(t + \Delta t) = g'(t) + \Delta t \cdot [g''(t) + g''(t + \Delta t)]/2$$

$$g(t + \Delta t) = g(t) + \Delta t \cdot [g'(t) + g'(t + \Delta t)]/2 - \Delta t^3 \cdot g'''(0)/12$$

The original second and third derivatives can be obtained from

$$g''(0) = 6g(1) - 6g(0) - 4g'(0) - 2g'(1)$$

$$g'''(0) = 12\{g(0) - g(1)\} + 6\{g'(0) + g'(1)\}$$

If the number of points to be generated on the cubic segment is n, then $\Delta t = 1/n$. Divisions by n can be replaced by arithmetic shifts if n is chosen to be a power of two. This yields an algorithm that uses only additions, subtractions, and arithmetic shifts (see Reference 9 for a complete example including appropriate scaling).