# Inter-Office Memorandum

| | | | |
|---|---|---|---|
| To | Spline Aficionados | Date | November 20, 1973 |
| From | Adrienne Payne | Location | Palo Alto |
| Subject | Spline Function Representation of Chinese Characters | Organization | PARC/CSL |

XEROX

Attached is a working draft of a paper by Bob Flegal; Bob told me to tell you all that he'd be around to talk to you individually about the paper. Or, you can stop by the Graphics Lab and chat with him.

Distribution

Patrick Baudelaire
Danny Bobrow
Alan Kay
Butler Lampson
Dick Shoup
Bob Sproull
Bob Taylor
Adele Goldberg
Diana Jones (archive files)
Dan Swinehart
David Liddle

# SPLINE FUNCTION REPRESENTATION OF CHINESE CHARACTERS

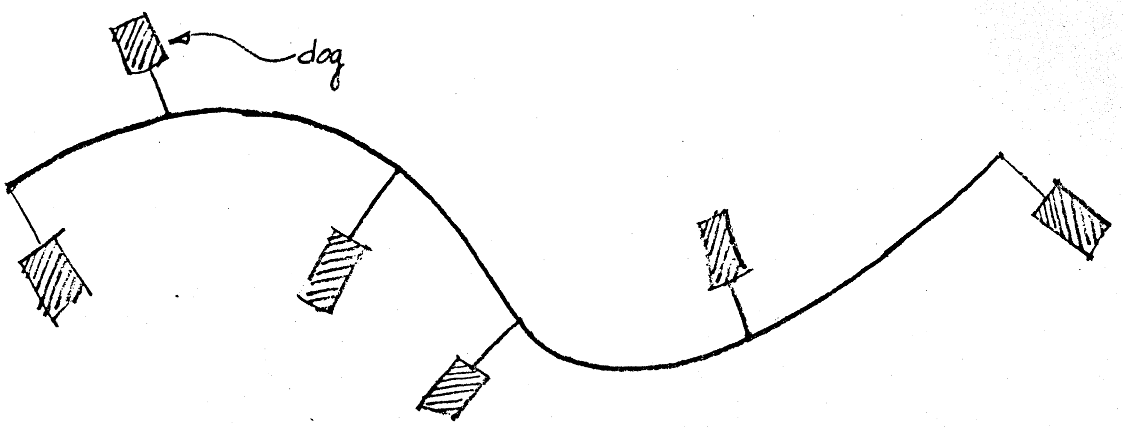
Working Draft / August 15, 1973

Robert Flegal

## INTRODUCTION

In using computers to represent and manipulate Chinese characters, much effort to date has quite properly been devoted to the primary problems of compactness of coding, storage and retrieval, and character recognition [References 1 through 5]. Aesthetic problems in matters of typography and calligraphy have necessarily played a lesser role. Yet looking ahead to the time when the aesthetic quality of computer generated output may be allowed to assume a greater significance, it is clear that much work remains to be done. Our purpose in this paper is to call attention to the possibilities inherent in the use of spline functions to represent arbitrary graphical figures in general, and Chinese characters in particular. Although the techniques have quite wide applicability, we have chosen to illustrate their use in the treatment of hand-written characters input to a small computer system in real time.

Cubic spline functions have properties that make them particularly suitable for approximating hand-drawn curves for which relatively few sample points are available. Most importantly they are the mathematical analog of splines used by draftsmen. A mechanical spline is a thin strip of plastic or metal that can be constrained to a particular shape with lead weights (called "dogs") that hold the strip in place.



As a consequence of behaving like mechanical splines, cubic spline functions are smooth, and furthermore, a change in the position of one of the dogs does not radically change the shape of the whole curve -- properties not shared, for example, by interpolatory polynomials.

## Outline of the method

Characters are written on a digitizing tablet which is interfaced to a small computer system.* [Footnote: The system consists of: a Data General Nova 800 minicomputer to which we have interfaced an Imlac PDS-1 display processor and a Scriptographics tablet.] As each stroke, or, in the case of cursive writing, each character, is written, the pen position is sampled periodically and the data temporarily stored.

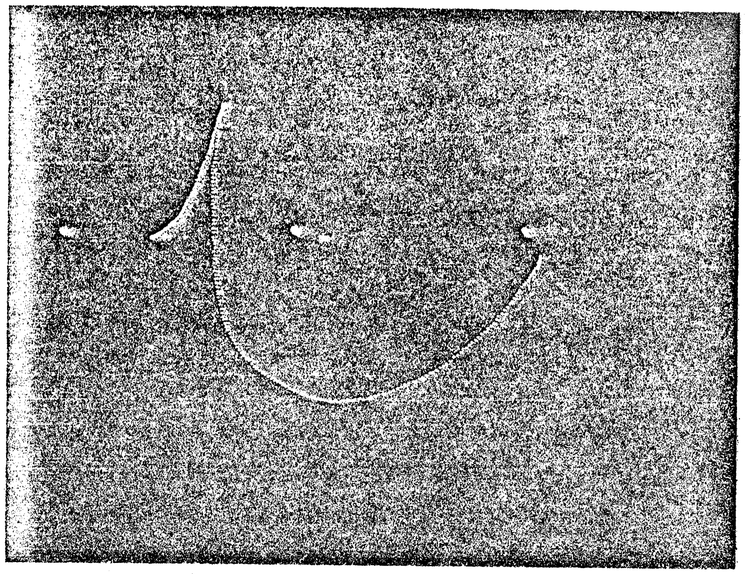


FIGURE 1

ORIGINAL TABLET COORDINATE DATA

This results in a relatively dense collection of coordinate pairs for each stroke. Two problems immediately arise: 1) for a sampling rate high enough to avoid loss of small detail (we used 5 milliseconds between samples), the number of points may run to several hundred per stroke -- far too many for convenient storage; 2) the sampling process may result in an unintended roughness, especially for slowly written strokes. Before proceeding with the analysis, therefore, the data are thinned to a relatively small number of representative points. This should be done in a manner which relates the number of points to the curvature. We have chosen an heuristic procedure for obtaining representative points which selects more of them when the curvature is high, less when the curvature is low, and yet enough of them that small detail is not lost when the stroke is reconstructed. Figure 2 shows a set of the representative points, as crosses, superimposed over the coordinate data in Figure 1.

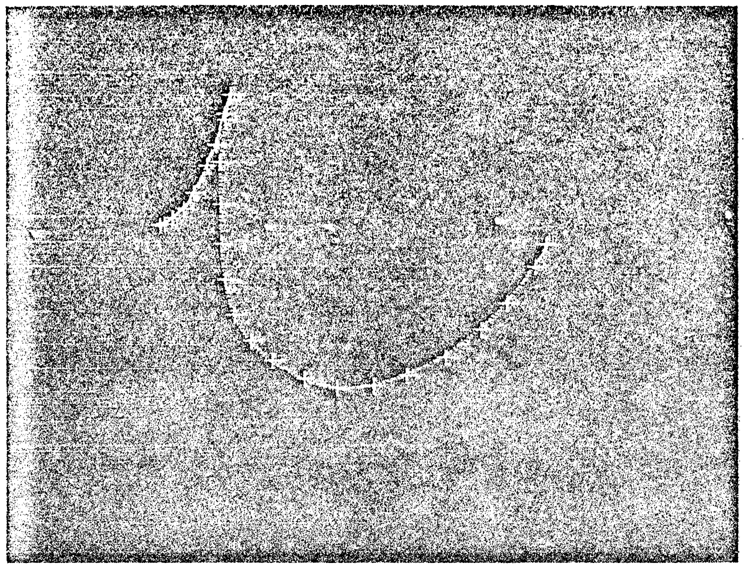


FIGURE 2

REPRESENTATIVE POINTS

All of the other data points are then discarded. To reconstruct the stroke, spline functions are used to interpolate a set of points between the representative points (we used 5 such points). These are then joined by straight lines and displayed, thus forming an approximation to the original stroke by linear segments.

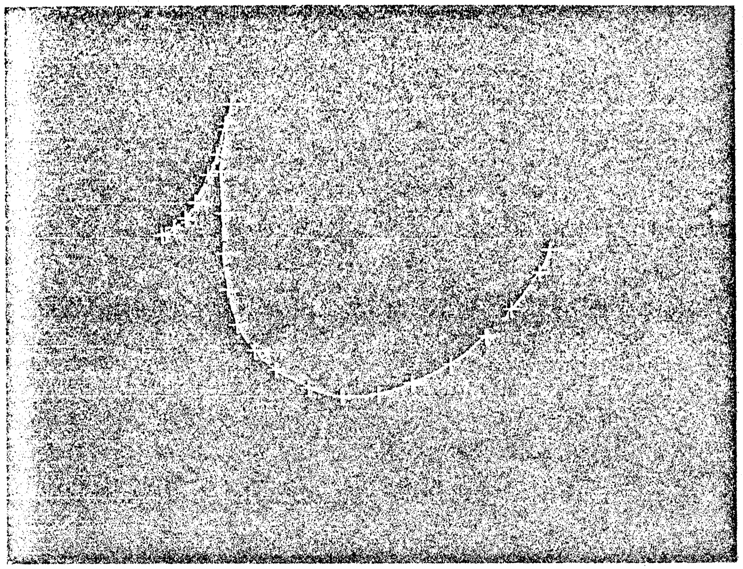


FIGURE 3

RECONSTRUCTED STROKE

Thus, a character or group of characters is formed by approximating and regenerating each stroke on the graphics display as it is written. Because the computations are rapid, this process is much like writing characters using a pen and paper.

Figures 4 and 5 are illustrative of characters generated using this method.

The remainder of this paper is devoted to the details outlined here with the hope that others may be able to use these techniques for their own applications.

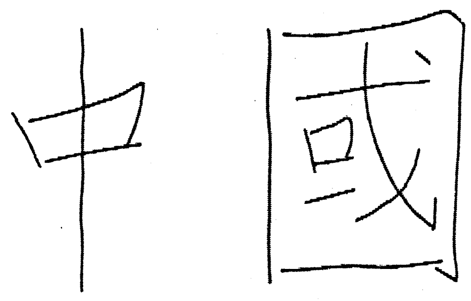


FIGURE 4

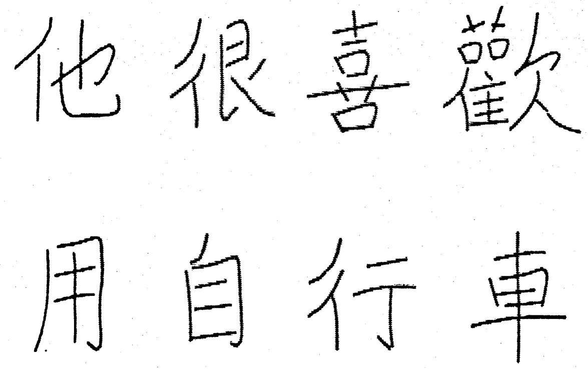


FIGURE 5

## SECTION 1

## SPLINE FUNCTIONS

Since the modern mathematical theory of spline approximation was introduced by I. J. Schoenberg in 1946 [Reference 6], many books and numerous articles have appeared in the literature dealing with their properties and characteristics as approximants [References 7, 8 and 9]. In this paper we will restrict our attention to cubic natural splines. For a more general treatment of spline functions, please refer to the aforementioned references.

Definition: Cubic Natural Spline.

Let

$$T = \{t_1, t_2, t_3, \ldots, t_n\}$$

be a set of real numbers (referred to as knots) such that

$$t_i < t_{i+1}.$$

A function $S(t)$ is called a cubic natural spline if it satisfies:

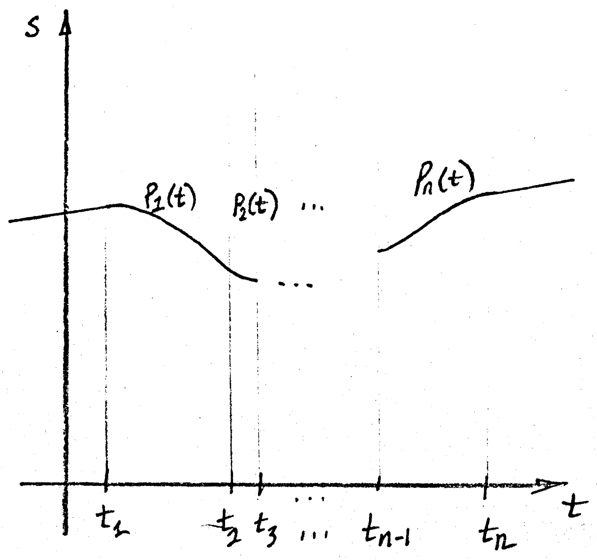
1) $S$ is a polynomial of degree 3 on each interval $(t_i, t_{i+1})$.

2) $S$ and its first and second derivatives are continuous everywhere.

3) In the intervals $(-\infty, t_1)$, $(t_n, +\infty)$, S is linear.

(Commonly called the natural end conditions.)

Pictorially:



Here $P_i(t)$ denotes the $i^{th}$ polynomial. Note that if all the $P_i^s$ are equal, the function S(t) reduces to a single cubic polynomial in the interval $(t_1, t_n)$. It is for this reason that spline functions are often called a generalization of polynomial functions.

It follows easily from this definition that S(t) can be written in the form:

$$(*) \qquad S(t) = b_0 + b_1 t + \sum_{i=1}^{n} a_i (t - t_i)_+^3 \quad \text{where } (u)_+ = \begin{cases} u \text{ if } u > 0 \\ 0 \text{ if } u \leq 0. \end{cases}$$

It also follows simply from the definition that there is a unique solution to the interpolation problem. That is, if we constrain S to take on the values $y_i$ at the knots $t_i$ $i = 1,2,3,\ldots n$, there is a unique natural cubic spline function S, satisfying $S(t_i) = y_i$. Furthermore, S is the function in $AC^1$ (the class of all functions with absolutely continuous first derivatives) that minimizes

$$\int_{t_1}^{tn} [f''(t)^2]dt$$

and interpolates the $y_i$ s/. This integral is commonly called the strain energy and that cubic splines possess this property accounts for their smoothness. It should be noted that by simply applying the interpolation constraints and the natural end conditions to (*), the resultant (n+2)X(n+2) linear system can be solved for $b_i, b_2$, and $a_i$ $(i = 1,2,3,...n)$, thereby determining the spline. This procedure is to avoided, however, because the linear system is numerically unstable. There are much better ways to compute splines, one of which is detailed in section (3) of this paper.

## SECTION 2

## PARAMETRIC REPRESENTATION OF THE STROKES

To allow approximation and regeneration of non-functional strokes such as the curve in Figure 7, it is convenient to represent them parametrically.

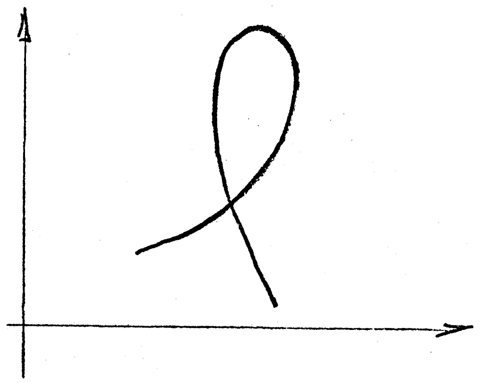


FIGURE 7

For example: X(t) and Y(t) shown in Figures 8 and 9 will generate the loop in Figure 7.

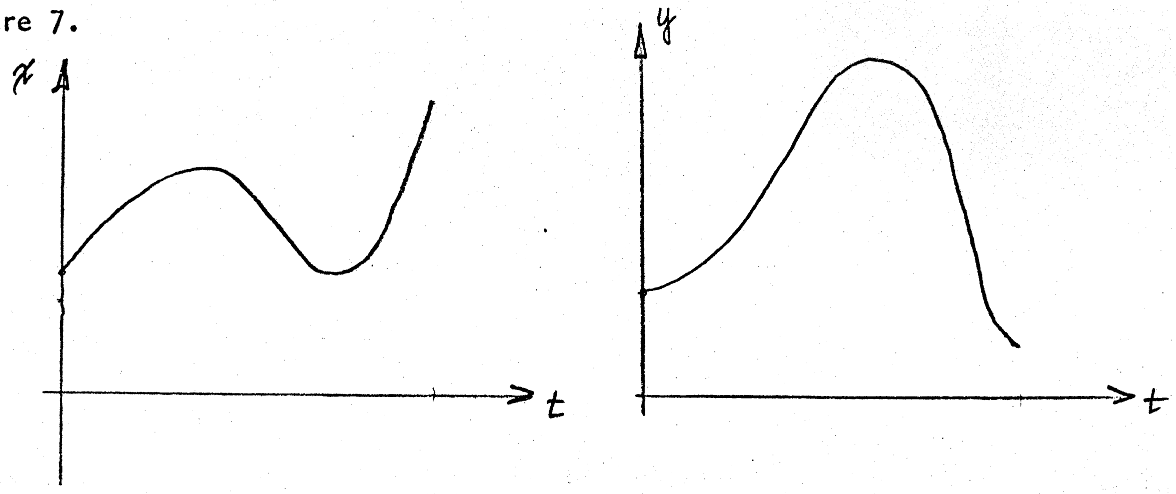


FIGURE 8

We represented each stroke by obtaining two functions X(t) and Y(t) which are the natural cubic spline functions that interpolate the representative point data $x_i$ and $y_i$ at the parametric values i. Of course, the function F = (X(t),Y(t)) interpolates the $x_i, y_i$ where $i = 1,2,3,\ldots n$, but as a consequence of this particular representation F is not a cubic spline itself. F is rather a function with first derivative and curvature continuity. This scheme works well visually and the loss of the second derivative continuity does not noticeably affect the aesthetic appearance of the curve. We chose the rather simple scheme of parameterizing by knot number for several reasons. First, at least visually we could not see the difference between this choice and other computationally more complicated schemes such as parameterizing by length between knots. Secondly, this scheme simplifies the computation considerably when solving the linear systems for the functions X(t) and Y(t). Finally, when one plots a constant number of short line segments between knots in the parameter space t, the corresponding line segments in the (x,y) space will be short when the curvature is high and longer when the curvature is low. Aesthetically this is a very desirable property, and curves whose plots have this property are called "fair" in the sense of Forrest [Reference 12]. "Fairness" is a consequence of the knot placement heuristic which selects knots (representative points) more densely when the curvature is high.

## SECTION 3

## COMPUTATION OF THE SPLINES X(t) AND Y(t)

In this section a method is presented to solve the following problem:

> Given n representative points $y_i$ where $i = 1,2,3,\ldots,n$ and the n values $T = \{1,2,3,\ldots,n\}$, compute the cubic natural spline function that interpolates the $y_i^s$ with knots T.

The general formulation of this method for computing interpolatory spline functions can be found in Reference 10 (see References at the end of this memo). Because the knots are at integral values, the method becomes computationally particularly simple and rapid. For simplicity, method is presented in terms of Y(t). Computation of X(t) is the same.

Because a cubic spline function is a cubic polynomial in each interval (i,i+1), letting $P_i(t)$ denote the polynomial in the interval (i,i+1), $P_i(t)$ can be represented by its Taylor series:

$$P_i(t) = y_i + P_i'(i)(t-i) + P_i''(i)(t-i)^2/2! + P_i'''(i)(t-i)^3/3!$$

Hence, if we can determine $P_i'(i), P_i''(i)$ and $P_i'''(i)$ for $i = 1,2,3,\ldots n-1$, the spline function will be completely specified. Using the continuity constraints associated with cubic splines, it is easy to show that the following relationships hold:

1) $P_i'(i) = \Delta y_i - \big(2P_i''(i) + P_{i+1}''(i+1)\big)/6$

2) $P'''(i) = P''_{i+1}(i+1) - P''_i(i)$

3) $P''_{i-1}(i-1) + 4P''_i(i) + P''_{i+1}(i+1) = 6\Delta^2 y_{i-1}$

where $\Delta$ is the difference operator $\Delta y_i = y_{i+1} - y_i$.

The basic idea behind the method can now be stated:

Using recursion relationship (3) and the fact that $P''(1) = P''(n) = 0$ (since the spline is natural) obtain a linear system of equations with the $P''^s_i$ as unknowns. Then solve this system. Since relationships (1) and (2) only involve the $y^s_i$ and the second derivatives of the polynomials in adjacent intervals, the first and third derivatives for each polynomial can be found. Thus, the spline is completely specified over the whole interval (1,n) by obtaining the Taylor series for each of the n-1 cubic polynomials that make up the spline.

The linear system arising from the recursion relationship (3) and the natural end conditions, is:

$$(*)\quad \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ & & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} P''_2(2) \\ P''_3(3) \\ P''_4(4) \\ \vdots \\ P''_{n-3}(n-3) \\ P''_{n-2}(n-2) \\ P''_{n-1}(n-1) \end{pmatrix} = \begin{pmatrix} 6\Delta^2 y_1 \\ 6\Delta^2 y_2 \\ 6\Delta^2 y_3 \\ \vdots \\ 6\Delta^2 y_{n-4} \\ 6\Delta^2 y_{n-3} \\ 6\Delta^2 y_{n-2} \end{pmatrix}$$

Because this system of equations is diagonally dominant, it is numerically stable and can be solved by any of the standard techniques. However, due to the particularly simple form of the coefficient matrix and our desire to make the computation rapid, we choose to use Gaussian elimination and back substitution to obtain the second derivatives.

The details of the Gaussian elimination are as follows:

(Consider the left-hand side of (*) first.)

Multiply the first row of (*) by 1/4 and subtract row 1 from row 2. This eliminates the 1 in column 1 of row 2. The new diagonal element in row 2 is $v_2 = 4 - 1/4$. Next, row 2 is multiplied by $1/v_2$ and subtracted from row 3. This eliminates the 1 in column 2 of row 3 and changes the diagonal element in row 3 to be $v_3 = 4 - 1/v_2 = 4 - \dfrac{1}{4 - 1/4}$.

Continuing this process, it is clear that the following recursion relationship holds for the diagonal elements:

$$v_1 = 4, \; v_{i+1} = 4 - 1/v_i,$$

and all of the ones in the lower triangular part of the coefficient matrix will be eliminated. That is, the coefficient matrix will have the form:

$$\begin{pmatrix} v_1 & 1 & 0 & \cdots\cdots & 0 & 0 & 0 \\ 0 & v_2 & 1 & \cdots\cdots & 0 & 0 & 0 \\ 0 & 0 & v_3 & \cdots\cdots & 0 & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & \cdots\cdots & 0 & v_{n-3} & 1 \\ 0 & 0 & 0 & \cdots\cdots & 0 & 0 & v_{n-2} \end{pmatrix}$$

If one knows the maximum size of the linear system to be solved (as we did in this work) all of the $v_i^s$ can be calculated in advance and stored in a table.

The row operations described above also affect the right hand side of (*).

Letting $b_i$ denote the $i^{th}$ element of the right hand side of (*), and after the row operations have been performed, we have:

$$b_{i+1} = 6\Delta^2 y_{i+1} - b_i/v_i \qquad \text{with } b_1 = 6\Delta^2 y_1.$$

Having computed the right hand side of (*), the back substitution proceeds:

$$P''_{n-1} = b_{n-2}/v_{n-2}$$

and

$$P''_i(i) = (b_{i-1} - P''_{i+1}(i+1))/v_{i-1}$$

and we are done with the second derivative calculations.

Estimating the computation time required for the calculation of the n second derivatives (assuming addition time = subtraction time and multiplication time = division time, and neglecting single operations) we have:

to compute the $\Delta^2 y_i$s/ requires 3n additions;

to compute the $b_i$s/ requires n additions and 2n multiplications;

the back substitution requires n additions and n multiplications.

Summing these, we conclude that the second derivative computation requires 5n additions and 3n multiplications.

From relationships (1) and (2) in this section, we conclude that calculation of the third derivatives of the spline requires n additions, and the calculation of the first derivatives requires 2n additions and 2n multiplications. Hence the total spline computation uses 8n additions and 5n multiplications. For example, using our Nova 800 computer, the computation of X(t) and Y(t) for a typical stroke (10 representative points) requires approximately 50 milliseconds.

## CONCLUSION

The calculating power of computers of modest size is sufficient to permit the representation of hand-written characters by spline function approximations in real time. Further application might involve tracing or other methods of coordination. The method requires the retention of only relatively few coordinate pairs representative of the original data, yet make possible reconstructions of arbitrary scale in which the visual quality of the reconstructed curve is limited only by the number of interpolated points. The required calculations are tractable and fast.

One must look beyond the poorly written characters we have represented here in illustration of the basic techniques. Further refinements may lead to a set of practical tools which, under the guidance of accomplished calligraphers and typographers, will help to enhance the quality of computer generated spline characters or pictures in future systems.

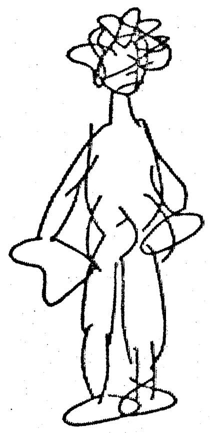


FIGURE 9 - A Spline Generated Man

## REFERENCES

[6] Schoenberg, I.J. "On Spline Functions" with Supplement by T.N.E. Greville, *Inequalities* (O. Shisha, editor), Academic Press (1967), pp. 255-291.

[7] Greville, T.N.E. (editor). *Theory and Applications of Spline Functions*. Academic Press (1969).

[8] Schoenberg, I.J. (editor). *Approximations with Special Emphasis on Spline Functions*. Academic Press (1969).

[9] Schultz, Martin. *Spline Analysis*. Prentice-Hall (1973).

[10] Ahlberg, J.H., E.N. Nilson, and J.L. Walsh. *The Theory of Splines and their Applications*. Academic Press (1967).

[11] Faddeeva, V.N. *Computational Methods of Linear Algebra*. Dover (1969).

[12] Forrest, A.R. "Curves and Surfaces for Computer-Aided Design," Joint Computer-Aided Design Group Mathematical Laboratory and Engineering Department, University of Cambridge (1968).

Appendix to Spine Function representation of Chinese Characters

R. Flegal

# A technique for rapid selection of knots for subsequent regeneration of curves and areas with spline functions.

The problem of finding the best (say in the least squares sense) approximation to an arbitrary function by an interpolatory cubic spline function with a fixed number of knots is non-linear and hence requires much computation. In an interactive enviornment where speed is often everything and the number of knots is large ( as it is with hand-sketched curves of arbitrary complexity ), standard techniques are insufficient. By relaxing the constraints that the approximation be mathematically "best" and that the number of knots be fixed I have devised a technique for selecting the placement of the knots that is fast (approximatly one millisecond per data point ) and whose resultant spline approximations fit ( at least within visual accuracy ) a wide variety of hand sketched data. The technique also selects "few" knots relative to the amount of original data... thus achieving significant data compression and speed in computation when regenerating the curves. I typically achieve compressions of one hundered to one. The technique will not approximate all curves equally well ( tends by itself to miss small corners ), but as the knot selection process eventually leads to a closed form mathematical representation of the curve , it is possible to precisly edit those curves whose approximation "misses".

## The technique

Let $(x_i, y_i)$ i=0,1,2,...,N be the original unthinned tablet data then:

1.place the first knot at $(x_0,y_0)$ and let $(x_L,y_L)=(x_0,y_0)$.

2. at each subsequent data $(x_i,y_i)$ $i=L,L+1$ ... calculate the area of the triangle formed by the last knot $(x_L,y_L)$ and the points $(x_i,y_i)$ ,$(x_{i+1},y_{i+1})$ and call this area $E_i$. Also form the approximation $A=\sum_{j=L}^{i} E_j$ . At the point $(x_i,y_i)$, A approximates $\int_{x_L}^{x_i}(f(x)-s(x))\,dx$ where s(x) is the straight line passing through $(x_L,y_L)$ and $(x_i,y_i)$.

3. Choose a knot at $(x_i,y_i)$ if either:

$$(A\geq C) \text{ or } (E_i\geq\alpha(E_{i-1}+E_{i-2}) \text{ and } A\geq A_{min})$$

The constants C, $\alpha$ and $A_{min}$ are somewhat application and user dependant, however I have found that =1,A=1/4000th of the total area of the tablet measured in square resovable units,$A_{min}$= 1/20000th of the total area of the tablet measured in the same units are good choices of these constants for most free-hand sketching systems including the Chinese character creation system.

In summary, in this technique:

A is a measure of the total "curvyness" between knots.

The condition $E_i\geq E_{i-1}+E_{i-2}$ picks up abrupt changes in curvature of the tablet data.

$A_{min}$ is simply included so that noise in the tablet data will not affect the knot selection process when near a knot.