Sādhanā, Vol. 18, Part 1, March 1993, pp. 1–15. © Printed in India.

Shape vectors: An efficient parametric representation for the synthesis and recognition of hand script characters

P V S RAO

Computer Systems and Communications Group, Tata Institute of Fundamental Research, Homi Bhabha Road, Bombay 400 005, India

MS received 13 November 1991; revised 29 August 1992

Abstract. Earlier work by the author has established: (i) that cursive script can be synthesised out of individual characters by using polynomial merging functions which satisfy boundary conditions of continuity of the displacement functions $x(t)$ and $y(t)$ for each character and their first and second derivatives; and (ii) that the procedure lends itself to a Bezier curve based formulation. This was done since cursive writing avoids discontinuities (of shape) between individual characters as well as discontinuities in pen movement.

We show here that even individual characters can be synthesised out of more primitive elements by using the same merging functions. We choose directed straight lines which we call shape vectors as basic elements for this. Script characters generally have shapes which are essentially straight segments alternating with 'bends' or regions of relatively high curvature. For a character with n bends, we need only $n + 1$ shape vectors. Thus, each script character needs only three to seven shape vectors, depending on its complexity.

The "character generation" shape vectors are derived from the original character by means of a simple procedure that identifies comparatively straight regions in it. These are then approximated to straight lines by linear regression and positioned to be tangential to the original curve. The synthesised version of this character is obtained by 'merging' or concatenating these vectors. The close fit between the original and resynthesised characters demonstrates that the shape vectors adequately characterise their identities and shapes. Data reduction ratios in the range of 15 to 25 are thus possible. This method thus shows good promise as a possible basis for script character recognition, and a recognition scheme based on it has yielded an accuracy of 94% for a vocabulary size of 67 words.

Keywords. Shape vector; cursive script; character synthesis; script recognition.

1. Introduction

Computer recognition of script characters is a problem that has engaged the attention of research workers for several years, essentially as a sub-problem of the general problem of visual pattern recognition. Our interest in cursive script, on the other hand, has been due to its similarity to speech in that it is a signal generated by the human and is meant to convey complex information effectively and efficiently.

We treat connected writing as the process of writing individual characters in the proper sequence, with minimal effort. When characters are written in isolation, the pen starts from and finishes in a state of rest. To write a string of characters in the form of an unconnected sequence of isolated shapes would be a difficult and highly constrained activity; it would be much simpler and easier to make a continuous pen-down movement connecting each character to the next; this eliminates the stop, lift, reposition and start motions between the characters (within a word). In this mode of writing, the start and end points of individual characters no longer remain distinct. This is evident from the difficulty in segmenting cursive script. Between each pair of adjacent characters, there is a particular transition region which can be said to belong to both, in the sense that the shape of this transition depends on the shapes or identities of not one but both the characters. In this sense, the shapes of the individual characters would get altered to a certain extent in cursive writing. Short of grossly sacrificing legibility, some deterioration in shape is tolerated in favour of smoothness and continuity of shape and movement.

It can possibly be argued that each script character is not an entity but a class consisting of several variants and that to achieve smoothness in cursive writing, the writer chooses an appropriate member of the class each time, depending on the identity of adjacent characters. Even if such a formulation should turn out to be adequate, it would need to be discarded in preference to a (parsimonious) model which hypothesises essentially a single entity (rather than a class) for each character and provides a means for explaining (and replicating) the variation in shape due to context (i.e. identity of adjacent characters). This paper offers such a model. (For instance, the differences in the shape of the character 'o' when it follows 'v' and 'p' in the word 'avoirdupois' in figure 2 below have been achieved using only a single model for each character).

In cursive script, the pen-down transition stroke betwen two adjacent characters manifests itself as a gradual anticipatory movement into the next character while still writing the earlier one. This ensures economy of movement; it also results in a smooth and efficient (i.e. minimal effort, minimal time) pen-down motion. It is only the end portion of the first character and the beginning part of the second one that get altered to achieve continuity via the transition stroke. In general, the core or the central (identity bearing) portion of the character remains comparatively intact and unaffected. This ensures legibility.

Quite clearly, this model highlights the trade-off between speed and effort on the one hand and legibility on the other. Cursive script becomes more interesting and challenging to study in cases where legibility is compromised noticeably in the process of achieving speed and ease of writing; this happens in most cases of cursive writing. We therefore consider this aspect in particular.

Our approach consists in machine-synthesising the transition strokes which link individual character shapes to generate connected script. Particular emphasis is laid on recreating the context effect underlying the pen-down transition stroke and in preserving the continuity of motion and shape in the transition.

Using this strategy, we evolved an effective synthesis mechanism which was successful in generating realistic cursive script by concatenation of individual characters. Each character is divided into three segments: a prefix, a core and a suffix. The synthesiser links the centrally located core (or shape identification) segments of individual characters. The transition segments used to link the core segments (of pairs of adjacent characters) are generated under the combined influence of the suffix of the earlier character and the prefix of the later one. Thus, synthesis essentially consists in the generation of transition segments. Under this framework, we propose two approaches: one based on a weighted average method and the other on a shape-specific Bezier splining technique. These are very briefly described below.

(i) *Weighted averaging*: Here, the transition stroke is generated as the weighted average of the relevant prefix and suffix segments (Ramasubramanian & Rao 1988, pp. 163–76), as illustrated in figure 1. In this case, the handwritten characters 's' and 'e' are to be concatenated, to form the composite 'se'. The computer has available to it the x- and y- coordinates of the individual characters, sampled at equal intervals of time. This information is acquired from a graphics tablet connected to the computer, on which the subject writes these characters, individually. The right half of figure 1 illustrates how this method works. $w_1(u)$ and $w_2(u)$ are the weighting functions used for performing the merging operation and have the following properties.

(1) w_1 starts with an initial value of one and falls gradually to zero during the transition interval marked by the two vertical, broken lines.
(2) w_2 starts with an initial value of zero and gradually rises to one during the transition interval.
(3) $w_1 + w_2 = 1$ for all values of u.
(4) The functions w_1 and w_2 as well as their first and second derivatives are all continuous.

The characters 's' and 'e' are shown in the left half of figure 1. The x-coordinate of the character 's' is shown in the right half, below the weighting functions, such that the suffix portion (s_1) of 's' (the portion c to d of character 's') falls in the transition interval. The prefix and core portions fall to the left. The x-coordinate of character 'e' is aligned to have its prefix portion (p_2 of character 'e') in the transition interval.

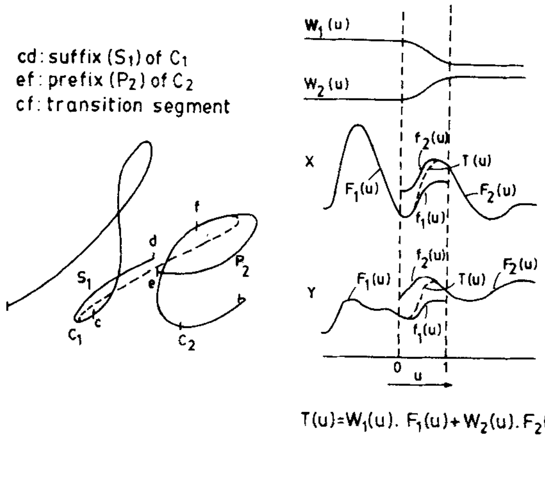

Figure 1. Character concatenation for script synthesis.

The rest of the character falls to the right. The suffix of the first character ('s') and the prefix of the second characters ('e') are therefore aligned.

The merged function $T(u)$ is obtained from the expression,

$$T(u) = w_1(u) \cdot F_1(u) + w_2(u) \cdot F_2(u).$$

It is easy to see that $T(u)$ will be equal to $F_1(u)$ to the left of the transition interval (because $w_2(u)$ will be zero) and equal to $F_2(u)$ to the right of the transition interval (because $w_1(u)$ will be zero). Within the transition interval, since the weightage due to $F_1(u)$ gradually decreases and that due to $F_2(u)$ gradually increases, $T(u)$ moves gradually away from $F_1(u)$ (towards $F_2(u)$) and merges with $F_2(u)$ at the end of the transition interval. This is indicated by broken lines.

Since all functions on the right side of the above equation as well as their first and second derivatives are continuous, $T(u)$ (and its first and second derivatives) are also continuous.

This process is repeated for the y-coordinates of 's' and 'e' in exactly the same manner. This is illustrated below the curves for the x-coordinates.

The resultant merged functions for the x- and y-coordinates are used to trace the curve in figure 1 (lift side), superimposed on the original 's' and 'e'. As is to be expected, it coincides exactly with the original characters 's' and 'e' during their core portions. During the transition region between 's' and 'e' (shown in broken lines), due to the influence of the weighing functions, it moves gradually away from 's' and into 'e'. The transition is smooth because of the continuity properties. The exact shape of the transition portion will depend on the shapes of the suffix of 's' and the prefix of 'e'.

(ii) *Bezier curve technique*: Here, the transition stroke is generated by a direct application of the Bezier curve formulation (Bezier 1972). The end segments of the characters are considered as the control points and the resulting Bezier curve forms the transition segment required in the concatenation of two characters. From the known properties of Bezier curves, it would seem that this formulation would be an appropriate one to use for the generation of the transition segment which replicates the observed characteristics of a transition stroke in natural connected writing. Since it would take too much space, this approach, which is discussed in detail in Rao & Ramasubramanian (1991), is not being dealt with here.

Both our synthesis schemes generated cursive script which was legible and looked very natural. When the prefix or suffix segments of individual characters were extended into the core, the resultant script replicated quite realistically the effects noticed in rapidly written cursive script. Figure 2 shows the word 'avoirdurpois' generated in this manner.

It would thus seem that in cursive writing, the writer takes care in reproducing more or less faithfully the 'cores' or central identity bearing segments of individual

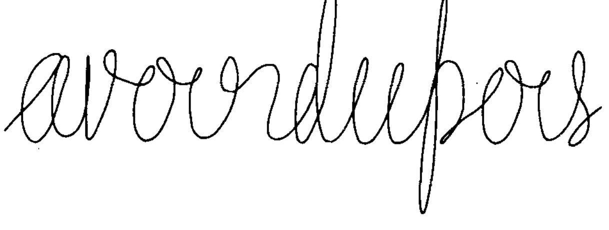

Figure 2. Connected script word 'avoirdupois' generated by the character concatenation procedure.

characters. He takes liberties with the prefix and suffix segments. He merges these to generate transition segments which link the cores in a smooth pen-down curve, thus ensuring continuity of shape and motion, economy of effort and maximum speed. This process causes some ambiguities in interpreting cursive script. For instance, the sequence 'vi' or 'oi' in cursive script may suggest the presence of an 'r': this happens even in the script generated using our method (see the sequence 'oi' in 'avoirdupois' in figure 2). Our synthesis approach thus provides an insight into the writing process, at the level of generating words from individual characters.

2. The writing process for cursive script

This opens up the question of the mechanisms involved in writing cursively. Three alternative views are possible for the visualisation of the writing hand system.

2.1 *Model for cursive script generation*

2.1a *Open loop model*: We can visualize an open loop source–system model where control is effected by a few simple driving signals delivered by the brain in the proper temporal sequence. The detailed shapes of the characters are fully determined at a lower level by the dynamic response (i.e. the impulse response or the transfer function) of the 'passive' (and linear) hand system. (These can be derived from the effective mass, friction, and stiffness of the system.) Thus, the brain does not play any part in controlling this level of the process.

Eden (1960) proposed a hand model of a pen driven by three groups of muscles acting essentially independently. Earlier, Van der Gon & Thuring (1962) designed and built a mechanical analog that could produce 'high speed' cursive script (also see Eden & Halle 1961, pp. 287–99). Yasuhara (1975, 1983) proposed a dynamic model which he used for simulating cursive script generation as well as for coding and data compression.

These models have the great virtues of simplicity and parsimony. Such efforts to simulate hand–pen systems using dynamic models have been reasonably successful in generating natural looking cursive script. However, they do not explain an important aspect: how hand writing remains distinctive and relatively invariant to changes in character size (e.g. between the normal characters written on paper and those written on the black board or a larger poster) even though different muscle groups and articulator masses (moving parts) are brought into play in each of these cases. The parameters for the model (mass, stiffness and friction), it is clear, would also have to change very substantially, as a consequence. It is unlikely that these substantial changes are such as to still leave the transfer function of the system invariant or very closely so.

2.1b *Closed loop model*: At the other extreme, we could visualise a closed loop control system; here, the behaviour of the hand system is under continuous control of the sensory motor system. It would be driven by an error signal which is the difference between the desired and actual positions of the pen at anytime; it is as if the hand is closely tracing an imaginary target character shape on the writing surface. This would be too complex a model; it appears unreasonable and is very unlikely to be valid, except in the case of the highly laboured and deliberate efforts of the neo-literate or non-fluent writer or that of a writer who is intentionally writing slowly and carefully.

2.1c *Target shape-driven model*: The third alternative would be a system which is target shape-driven only in a broad sense. It is neither an open loop system (responding passively to a fixed sequence of invariant control signals) nor one which is controlled entirely by feedback. In this model, conscious closed-loop feedback or active control would be restricted to ensuring conformity with only the very broad shape features. The finer details would be governed by the characteristics of the writing process (i.e. constraints regarding continuity of shape and movement) rather than by the dynamics of a specific passive hand–pen system. (This seems to be reasonable and could be achieved as a lower level brain function without involving the higher levels of the sensory motor system.)

In fact, our cursive script synthesis system does in effect implicitly assume precisely such a model at the inter-character level. That system is based on the premise that the writer consciously needs to generate only the character shapes and that connecting them to form cursive script is taken care of by the constraints regarding continuity of shape and movement.

2.2 *Cursive script at the character level*

This opens up the possibility that a similar process takes place at a lower (intra-character) level: i.e. that the characters can themselves be visualised as being composed of (or realised as combinations of) simpler, or more elemental or primitive shapes in the same sense that cursive script can be visualised as being constructed out of individual characters. This line of reasoning opens up two alternative (but mutually complimentary and in essence equivalent) approaches for investigation.

The first is the synthesis approach: can natural looking script characters be synthesised out of a small number of much simpler shapes? The second – the analysis or decomposition approach – consists in studying whether handwritten characters can be decomposed into a small number of simple elements.

2.3 *A study of script character shapes*

2.3a *The synthesis approach*: Our effort here will be to investigate whether simpler elements can be used to generate characters. A related and simpler question to answer is whether meaningfully complex shapes can be obtained by concatenating, say, the simplest of shapes to write: straight lines oriented in different directions. A particular direction of writing has of course to be specified for each line; the line has to be divided into prefix, core and suffix segments. Suffix–prefix concatenation between adjacent segments can be effected either by weighted averaging or by Bezier splining techniques.

In the former case, the transition curve is obtained as a weighted mean of the two overlapping segments at each time instant in the transition zone (Ramasubramanian & Rao 1988, pp. 163–76). The transition segment so generated effects a smooth and gradual shift from the position and direction specified by one directed line (or vector) to those of the other. This procedure has been briefly described in § 1, for character concatenation.

In the Bezier formulation, the transition is generated as a Bezier curve under the

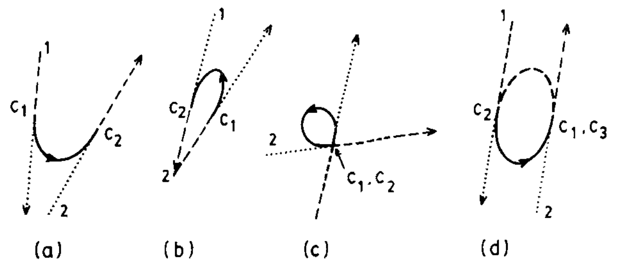

Figure 3. (a–d) Curves obtained by concatenation of four different combinations of line vectors.

influence of a control polygon whose vertices lie on the two vectors. In the limiting case, the control polygon reduces into a quadilateral or triangle, with the vertices being located at the extremes of the prefix and suffix segments being concatenated. (For details of this method, see Rao & Ramasubramanian 1991.)

Since the two approaches are essentially equivalent, the results are the same in both cases; the concatenated curve is a transition segment which effects a smooth and gradual shift from the position and direction specified by one vector to those of the other.

We illustrate this in four different cases in figure 3.

In each ease, line 2 is to be concatenated with line 1; i.e. the suffix of line 1 and the prefix of line 2 (shown in dotted lines) are to be merged. As mentioned in § 1 in the case of character concatenation, the transition segment (thick unbroken line) has to satisfy continuity constraints relating to the curve as well as its first and second derivatives: slope and radius of curvature. It therefore has to start off from point C1 on line 1 and join line 2 at point C2. The two lines have to be tangential to the curve at the points of contact. Even the curvature has to start at a value of zero (for the straight line) at C1, rise to a high value in between, and again fall to zero (straight line) at C2. This can be verified to be so in all four cases, as illustrated in figure 3. In figure 3d, the thick broken curve illustrates the transition that would result if the order is reversed; i.e. start with line 2 and end up in line 1.

The resulting shapes in figure 3 can easily be seen to be (very close in shape to) parts of script characters. In fact, simple characters like 'e' and 'l' can even be directly formed by concatenation of appropriately placed configurations of lines or vectors.

2.3b *The decomposition approaches*: A corollary to this question would be whether character shapes can be decomposed into simpler elements on the basis of reasonable (i.e. intuitively self-justifying) criteria: e.g. on the basis of discontinuities in shape and movement. Two good criteria suggest themselves in this context.

(i) *Discontinuities in shape*: These occur when there are rapid changes in the direction of movement of the pen; the curve that is traced will have high curvature (i.e. minimum radius of curvature) in such regions.

(ii) *Discontinuities in movement*: These occur when the pen pauses or slows down significantly during writing; the speed of pen movement will be minimum in such regions.

In this context, it would also be meaningful to study and investigate whether there is any correlation between these two.

It is, of course, reasonable to expect that the pen slows down in regions of shape discontinuities (i.e. where the curvature is high) and that the pen moves quite fast in

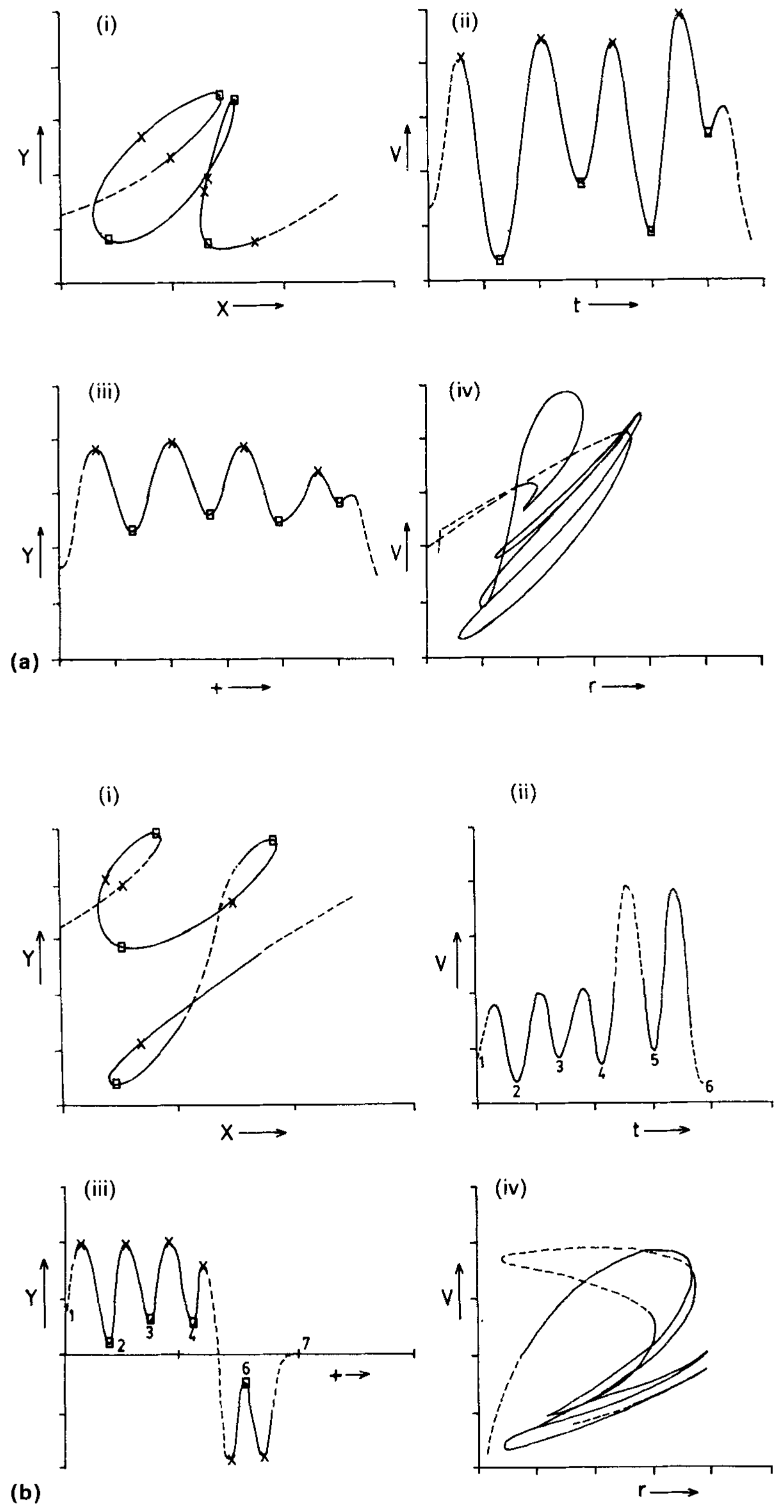

Figure 4.(a) and (b) Variation of pen speed and radius of curvature compared (as functions of time and against each other); (i) character shape; (ii) pen speed; (iii) radius of curvature (magnitude); (iv) radius of curvature versus pen speed.

regions where the curve is comparatively straight; in other words, that regions of low radius of curvature are also likely to be regions of low pen speed.

Figures 4a and b illustrate the shape of the character and the pattern of the variation of the radius of curvature r and the pen speed v against time and against each other (r and v against time and r against v). It is quite easy to compute r and v (Mokhtarian & Mackworth 1986), given the x- and y-coordinates of the curve, sampled in time.

$$v = [(\mathrm{d}x/\mathrm{d}t)^2 + (\mathrm{d}y/\mathrm{d}t)^2]^{1/2},$$

$$r = [(\mathrm{d}x/\mathrm{d}t)^2 + (\mathrm{d}y/\mathrm{d}t)^2]^{3/2}/[\mathrm{d}x/\mathrm{d}t)(\mathrm{d}^2 y/\mathrm{d}t^2) - (\mathrm{d}y/\mathrm{d}t)(\mathrm{d}^2 x/\mathrm{d}t^2)].$$

For each of the characters studied, the r vs t and v vs t curves correspond very closely with each other in terms of the overall shape as well as the positions of the maxima and minima (except at the start and end where v is zero but r is not). (There is also an anomaly at the long downstroke of y, because the curvature becomes zero, i.e. radius becomes infinity and changes sign.) This correspondence is even more closely demonstrated in the r vs v curves which crowd around a straight line with positive slope, passing through the origin. This trend is consistent for all the characters studied.

This is a very significant result. It indicates to us that we could segment characters using either of the two criteria, equivalently: minima in either the speed of movement or in the radius of curvature of the character shape. The decomposition procedure therefore consists in the following steps: plotting r and v against t, identifying the minima, marking them in the character, fixing a threshold for v and r and, finally, deleting all points for which v (or, equivalently, r) is less than the specified minimum threshold.

This decomposes the character into disjoint segments. It is reassuring that the segments, which approximate to straight lines, are quite small in number; almost an order of magnitude fewer than the line segments used by earlier workers (Farag 1979). It would be reasonable to substitute these by straight lines and investigate whether these can, in some sense, be said to characterise the original shape of the character.

2.4 *The concept of shape vectors*

The results described in § 2.2 enable us to make the conjecture that script characters can be visualised as resulting from an effort to trace in rapid succession a sequence of straight strokes or vectors. In so far as the relative directions, orientations and sizes of these vectors determine the shape of the resulting character, we call them the 'shape vectors' for the character.

In other words, shape vectors are those straight lines from which the character can be generated by pair-wise concatenation in sequence, using the same technique that we used for generating cursive script by concatenation of individual characters. In this section, we study and illustrate the relationship between the shape of the character and the underlying shape vectors that can be said to have generated this shape. For this, we would need to decompose the shape vectors into prefix, core and suffix segments; we can then use the suffix–prefix pairs to generate transition strokes, either by using a suitable merging function or by means of the Bezier control polygon approach.

In both approaches, the prefix and suffix are discarded, after being used to generate the transition strokes; only the core segments retain their identity after concatenation.

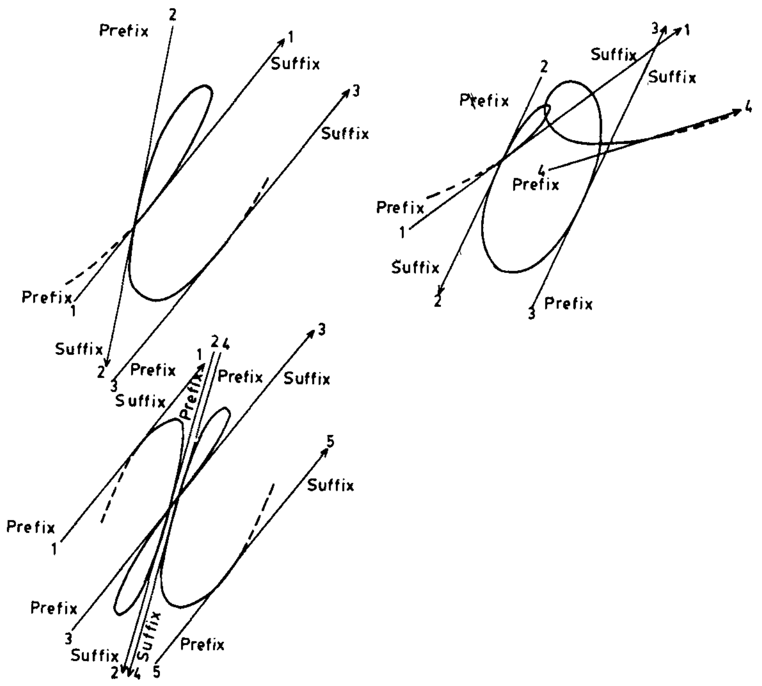

Figure 5. Synthesis of the characters ‘e’, ‘x’ and ‘o’ from shape vectors.

The core segments, being part of the shape vectors, would themselves be straight lines. Thus, absolutely straight portions in the character (portions where curvature is zero, i.e dy/dx is constant) can be identified as the core segments retained from the shape vectors. However, script characters, in general, contain little or no straight portions, which could be considered to be the core segments of the shape vectors. Thus, if our conjecture is valid, we are dealing with a degenerate type of concatenation where the elements being concatenated have null or missing core segments. In other words, the shape vector has little or no core, almost the entire length being composed of only the prefix and suffix segments. Each shape vector is then split into a prefix and a suffix segment and is used up in the concatenation process. The character becomes essentially a series of transition segments linking notional or non-existent core segments: i.e. a series of transition segments linking each other. Figure 5 illustrates how the characters ‘e’, ‘x’ and ‘o’ are generated by concatenating shape vectors. It can be seen that the core segments in between the prefix and the suffix are of zero length.

2.5 *Extraction of shape vectors from the character*

How then does one extract the shape vectors from the character? We know that at least zeroth and first-order continuity is maintained between the core and the transition curve at the junction point. This means that each shape vector has to touch the character at the concatenation point; i.e. that each of the constituent shape vectors has essentially a single point tangential contact with the character. Also, since the notional core is a straight line segment, this point of contact has to be located in the comparatively straight (or low curvature) sections of the character and away from

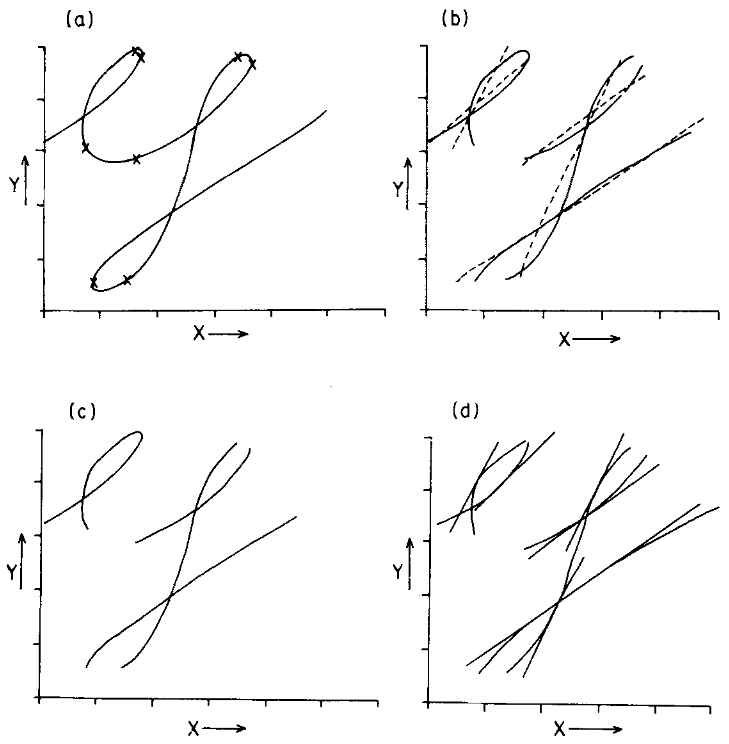

Figure 6. The various steps in the extraction of shape vectors from the character shape for 'y': (**a**) points where radius of curvature equals threshold are marked; (**b**) slopes are estimated by linear regression; (**c**) regions where radius of curvature are below-threshold are deleted; (**d**) slope lines are moved to touch the curves.

regions of large curvature. It follows therefore (if our conjecture is valid) that the shape vectors can be derived by locating the comparatively straight portions of the characters and drawing tangents to them at the points of minimum curvature.

Thus, a character synthesis system can be visualised where each loop or curved segment is generated by the concatenation of two shape vectors. Since each minimum (bend) is generated from two adjacent vectors and each vector contributes to generating two adjacent bends, it is easy to see that for a character with n bends, we need $n + 1$ vectors.

We can therefore construct the shape vectors by the following procedure (see figure 6).

(1) For each individual character, plot the pattern of variation for the radius of curvature against the arc length of the curve.

(2) Mark the minima: i.e. points of minimum radius of curvature (figure 6a).

(3) Discard regions in the neighbourhood of these points (i.e. regions where the radius of curvature is less than a maximum threshold) (figure 6b). This leaves a few unconnected regions which are comparatively straight.

(4) In each of these, mark the point of minimum curvature – i.e. maximum radius of curvature: the maximum point.

(5) At this point, draw a tangent to this curve. This can be done by determining the slope at this point. This, however, is not a very reliable method; because of sensitivity

to quantisation errors and noise, the supposed tangent might actually intersect the curve at an angle.

Therefore, alternatively: (a) Instead, fit a straight line to the curve segment using linear regression techniques (figure 6c); (b) Move this line parallel to itself, so that it touches the curve only at one point. This is taken to be the desired tangent (figure 6d).

The lengths of the suffix and prefix segments of the shape vectors are important because, for our method of concatenation, the shape of the transition curve changes with the lengths of the prefix and suffix segments being used. These can be determined by the following geometrical construction (figure 7a): Bisect the angle formed by the two shape vectors (dotted line) and mark the point X where the line intersects the portion of the character sought to be synthesised by the two vectors. Draw a normal to the bisector at this point, to meet the shape vectors at point X_1 and X_2. Point X, which is midway between X_1 and X_2 represents the midpoint of the transition curve and results from the controlling influence of the points X_1 and X_2 in the prefix and suffix segments of the vectors to be concatenated. Thus, X_1 and X_2 themselves have to be midpoints of the respective prefix and suffix segments. The segments are therefore formed by extending them to twice the lengths of XX_1 and XX_2. The premise is that the sample points on the shape vectors are uniformly spaced: i.e. that the shape vector is traced with uniform velocity. In the case of Bezier curve concatenation, the ratios P_1X_1/P_1Y_1 and P_2X_2/P_2Y_2 are 3/8 instead of 1/2. These ratios are derived from the fact that the Bernstein polynomials $3u(1-u)^2$ and $3u^2(1-u)$ have a value of 3/8 for $u = 1/2$ (vide Rao & Ramasubramanian 1991).

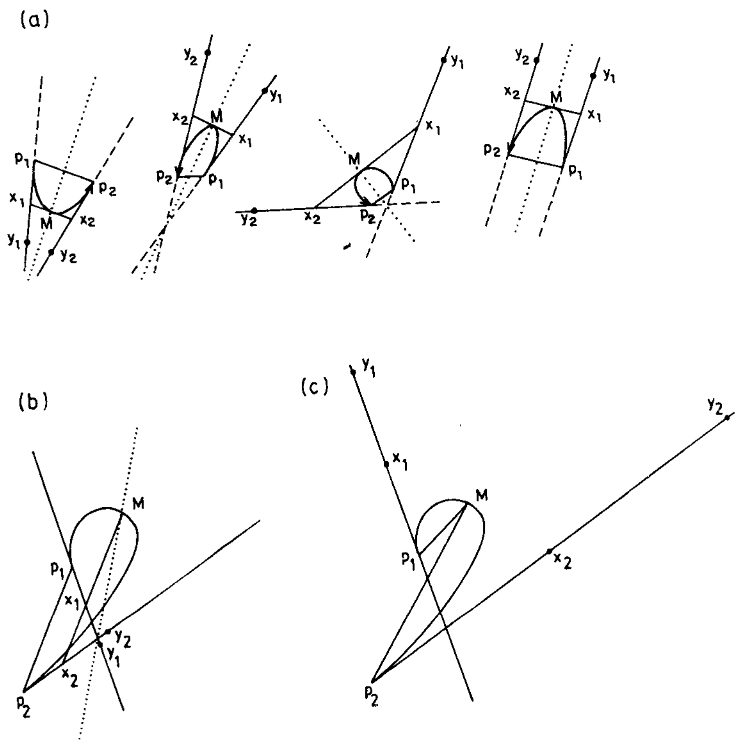

Figure 7. (a–c) Determination of lengths of prefix and suffix segments of shape vectors.

This procedure is followed for reconstructing the suffix and prefix segments s_{i-1} and p_i responsible for each transition segment (i.e. the portion of the curve that lies between adjacent maxima M_{i-1} and M_i). Needless to say, the prefix and suffix segments p_i and s_i forming the shape vector are collinear, since they touch the character at the maximum M_i.

In a few cases, where the curve is grossly asymmetric, this method may not work, because the tangent X_1MX_2 may meet one of the two shape vectors on the wrong side of P_1 or P_2; i.e. on the side pointing away from M (figure 7b). In such a case, we follow a different method (figure 7c); X_1 and X_2 are located such that the arc distance $P_1M = P_1X_1$ and similarly $P_2M = P_2X_2$.

2.6 *Procedure for concatenation of shape vectors*

Resynthesising the original character from the shape vectors is a very straightforward procedure. Each shape vector is again broken up into three segments: prefix, (non-existent) core and suffix. This contact point at which the shape vector touches the character represents the prefix-(core-)suffix junction. Each shape vector can then be split into prefix and suffix segments p_i and s_i. s_i and p_{i+1} can be concatenated to form the transition curve. Suffix and prefix sections are overlapped in time and merged to form the transition segments.

The overlapping operation causes the prefix and suffix segments to merge, losing their identity in forming the transition segments. Only the core segments remain intact. The original shape vectors, since they have no core segments, get almost completely consumed in the merging process.

3. Results

The synthesised character is legible and looks quite natural. Even the agreement between the original character and its synthesised version is quite close in each case (figure 8). It is possible to achieve a near-perfect fit by slightly varying the lengths of the shape vectors; i.e. by interactively shifting Y_1 and Y_2 along the lines P_1X_1 and P_2X_2. In fact, the fit is then so good that the original and the resynthesised versions are indistinguishable from each other (figure 9). This is true for all the 26 script characters tried.

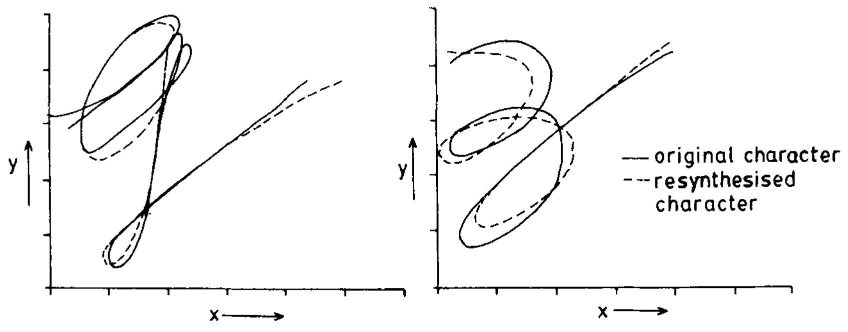

Figure 8. Characters automatically resynthesised from shape vectors compared with original characters.

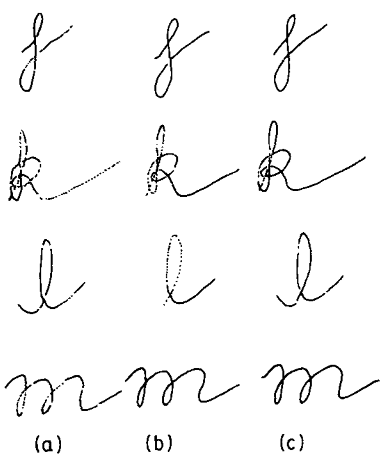

Figure 9. Characters interactively resynthesised from shape vectors compared with original characters: (**a**) original, (**b**) synthesised, (**c**) the two superimposed.

4. Data reduction efficiency

Apart from being able to capture and faithfully reproduce the original shape, any parametric representation system has to satisfy an equally important criterion: that it should achieve a significant amount of data reduction. Our method requires $(n + 1)$ shape vectors for a character with n loops. It is necessary to specify three points (the

Table 1. The extent of data reduction achieved by means of shape vectors.

Character	No. of sample points	No. of loops	No. of shape vectors	No. of points required	Data reduction ratio
a	238	4	5	13	18·38
b	209	4	5	13	16·08
c	200	2	3	7	28·22
d	304	4	5	13	23·38
e	146	2	3	7	20·86
f	294	3	4	10	29·4
g	280	4	5	13	21·54
h	245	4	5	13	18·85
i	157	2	3	7	22·4
j	173	2	3	7	24·7
k	369	5	6	16	23·06
l	221	2	3	7	31·55
m	295	6	7	19	15·33
n	238	4	5	13	18·30
o	217	3	4	10	21·7
p	298	4	5	13	22·97
q	236	4	5	13	18·15
r	212	4	5	13	16·37
s	150	2	3	7	21·43
t	192	2	3	7	27·43
u	210	4	5	13	16·15
v	198	4	5	13	15·23
w	309	5	6	17	18·17
x	206	4	5	13	15·84
y	297	4	5	13	22·85
z	252	4	5	13	19·38

start of the prefix, the prefix–suffix juncture and the end of the suffix) per shape vector. The prefix and suffix segments at the beginning and end of the character, on the other hand, require only two points each. We thus need only $2 + (n-1)\cdot 3 + 2$ or $3n + 1$ points to completely specify a character with n loops.

Table 1 shows the extent of data reduction achieved; this ranges between 14 to 25.

5. Discussion and conclusions

The foregoing analysis and results demonstrate that the hand–pen system can indeed be modelled as being shape-driven (or actively feedback controlled) only with respect to the broad shape features (as represented by the shape vectors) and that the detailed shape is traced as a consequence of constraints relating to continuity of shape (continuity of the curve and its derivative) and movement. More importantly from a practical point of view, they also demonstrate that the shape vectors adequately characterise the identity and canonical shape of the original character. It is therefore obvious that they should provide a basis for script character recognition. We have in fact been able to use shape vector-related parameters for recognition of cursive script (Rao 1990a, pp. 441–4, 1990b, pp. 1237–41). This method does not require elaborate learning; it needs only one sample each of the 26 script characters and a list of words in the vocabulary. Recognition accuracies around 94% have been possible with a vocabulary size of 67 words.

References

Bezier P E 1972 *Numerical control – mathematics and applications* (transl.) A R Forrest (London: John Wiley and Sons)

Eden M 1960 Handwriting and pattern recognition. *IRE Trans.* IT-8: 160–166

Eden M, Halle M 1961 The characterization of cursive writing. *4th London Symposium on Information Theory* (ed.) C Cheng (London: Butterworth)

Farag R F H 1979 Word-level recognition of cursive script. *IEEE Trans. Comput.* C-21: 172–175

Mokhtarian G, Mackworth A 1986 Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-8: 34–43

Ramasubramanian V, Rao P V S 1988 Connected script synthesis by character concatenation – An overlap and weighted average formulation. *Modern trends in information technology* (Proc. SEARCC '88) (eds) P V S Rao, P Sadanandan (New Delhi: Tata McGraw-Hill)

Rao P V S 1990a Word based recognition of cursive script. *Proceedings of IAPR Workshop on Machine Vision Applications* (Nov. 28–30) Tokyo

Rao P V S 1990b Cursive script recognition using neural nets. *Proceedings of International Conference on Automation, Robotics and Computer Vision* (ICARCV '90) (Singapore: The Institution of Engineers)

Rao P V S, Ramasubramanian V 1991 Connected script synthesis by character concatenation – A Bezier curve formulation. *Inst. Electron. Telecommun. Eng.* 37: 485–493

Van der Gon D, Thuring J 1962 A handwriting simulator. *Physiol. Med. Biol.* 6: 407–414

Yasuhara M 1975 Experimental studies of handwriting process, Rep. Univ. Electron. Commun. 25-2, Science & Technology Section, pp. 233–254

Yasuhara M 1983 Identification and decomposition of fast handwriting system. *IEEE Trans. Circuits Syst.* CAS-30: 828–832