Sādhanā, Vol. 19, Part 2, April 1994, pp. 257–270. © Printed in India.

Script recognition

P V S RAO

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

MS received 11 May 1993; revised 25 February 1994

E-mail: rao@tifrvax.tifr.res.in

Abstract. This paper describes an approach for word-based on-line and off-line recognition of handwritten cursive script composed of English lower-case letters. The system uses simple and easily extractable features such as the direction of movement and curvature and the relative locations of regions where these suffer discontinuities.

Our approach was evolved based on our concept of 'shape vectors' introduced earlier. We visualise script characters as having shapes which are composed of comparatively straight segments alternating with regions of relatively high curvature. We derive the shape vectors from each script character essentially by identifying regions of least curvature and approximating these by straight lines. That these shape vectors carry adequate information about the identity of the character is established by showing that the original character can be faithfully reconstructed from the shape vectors.

We thus use slopes of the shape vectors and relative locations of points of maximum curvature (both highly quantised) as parameters for recognition. The system extracts parameters for individual characters from single specimens written in isolation and uses these to construct feature matrices for words in the vocabulary. These are used for matching with the feature matrices of test words during the recognition phase.

The advantage of the system is that it does not require elaborate training. Recognition scores are in the neighbourhood of 94% for vocabulary sizes of 200 words. The approach has been extended for off-line information as well and performs quite well even in this case.

Keywords. Character synthesis; cursive script; feature matrices; on-line and off-line recognition; overlapping segments; script recognition; shape vectors; tract segments; quantisation.

1. Introduction

The problem of script recognition by computers has been an active area of research over several decades. Cursive script recognition is, for obvious reasons, a much more complex problem than recognition of print or hand-printed characters.

Cursive script recognition involves deducing from the written script (which is a more or less continuous two-dimensional curve in an iconic form) the underlying sequence of discrete symbols or characters.

Cursive script information can be acquired in the on-line or off-line mode. In the former case, an electronic graphics tablet digitizes the pen coordinates continuously in real time, even as the characters are being written. Off-line systems use scanner digitizers to transfer a page image to the computer. Script recognition presents several difficult problems; we list the more important ones below.

(i) During cursive writing, strings of discrete characters are encoded into a more or less continuous curve. Decomposing this into character length segments is a non-trivial problem. Segmentation is eliminated if one does recognition at the word level directly, but then recognition will have to be accomplished at the word level; the number of classes increases very substantially. There is, in fact, a trade off between the complexities of segmentation and classification.
(ii) Proper choice of a small number of convenient parameters is an important step in script recognition. Individual script characters get considerably distorted in connected script, due to context effects caused by the continuous pen-down movement from one character to the next.
(iii) There are considerable differences in the handwritings of different writers. Even the same person writes individual characters and even words quite differently at different times. A recognition system, to be reasonably robust, has to be tolerant of such variations.
(iv) Word level recognition systems may use word level templates for direct comparison with the words in the test script. Alternatively, character templates can be used if it is possible to account for context effects; i.e. the distortions to the shapes of individual letters which occur in cursive writing.

2. Review of earlier work

Earlier script recognition systems have been based on statistical pattern recognition techniques. Mermelstein & Eden (1964a, 1964b) segment cursive script into a sequence of strokes. The strokes are characterised by velocity functions composed of pairs of quarter wave sinusoidal functions with different frequency and phase shift parameters and are classified into categories on the basis of topological similarities. There are rules constraining the adjacent occurrence of strokes and letters. Recognition accuracy is around 78%.

Riseman & Ehrich (1971) demonstrated the advantage of a structured approach: that contextural information helps to accomplish accurate word recognition even if character recognition performance is poor. Even with a character recogniser which divides characters only into five broad classes, word recognition scores could be as high as 98%, for a vocabulary of 300 seven-letter words. For longer words, performance would be even better.

Ehrich & Koeheler (1975) adopted such a strategy for actual cursive script recognition. Here, the first down stroke of each character is recognised as a precursor to segmentation. Unlikely candidates are weeded out by using topological information. Substitution sets are generated at this stage. Context constraints are applied to narrow down the lists, based on two-class Neyman–Pearson decision criteria. Error rates are low (less than 1%); however, the rejection rate for different writers could vary between 1% and 30% (for a dictionary size of 300 seven-letter words).

Farag (1979) used a stochastic model where each word is represented as a Markov chain of state transitions. He used eight types of basic strokes (straight line segments with slopes which are multiples of 45°) of standard length. Words are segmented into sequences of such strokes. Learning consists in arriving at a set of stochastic forward transition matrices which define the transition probabilities between states for each word. Recognition uses a maximum likelihood classifier to determine which specific word might have produced the particular sequence of strokes. 100% recognition was achieved on a set of ten cursively written key words.

Sayre (1973) used topological features for word recognition. Rather than use stroke sequence information, he utilised letter bigram and trigram probabilities to eliminate improbable sequences. Word recognition scores were in the neighbourhood of 80%.

Bozinovic & Srihari (1989) adopted a multi-tiered approach for off-line recognition of cursive script. The problem of script recognition is dealt with in a series of transformations between different levels of representation. As a first step, the raw image is smoothed and slant correction is performed. The horizontal reference lines (which define the vertical extent of different types of characters) are then located. Minima in the 'y' component of the lower contours of the inscription are used as the basis for segmentation. The topology of the image is derived by a contour tracing operation. A description of the test pattern is then obtained, as the next step, in terms of certain relevant features and their locations. This yields a number of competing letter string hypotheses which account for the features in the test word pattern. The correct word is picked up on the basis of a best fit between the word string hypotheses and entries in the word lexicon. This system performed with accuracies ranging between 50% and 75% correct scores, for different writers, as the training data and the size of the lexicon are changed.

Kundu *et al* (1989) used a Hidden Markov Model (HMM) scheme for word recognition. They use a set of forty features which are nonredundant, easy to extract and independent of rotation, translation and size (e.g. number of loops, T- and X-joints and zero crossings, ratio of horizontal to vertical size, existence of isolated dots, and existence of semicircles as part of the character shape). Vector quantisation is used to arrive at an optimum set of vectors from a corpus of 2500 letters. Characters with similar shapes (such as 'e' and 'l') get grouped together in this approach. These classes are the states in the Hidden Markov Model at the word level. Statistical studies for the English language help to determine letter to letter and letter pair to letter transition probabilities. Each unknown letter in the test word is converted into a representative 'symbol' by means of the code book. The test word becomes a sequence of such symbols. The HMM framework helps to determine which lexical word is most likely to have yielded the observation sequence. The error rate is around 7·5%.

1.1 *Comparison with other methods*

In general, script recognition systems in the literature require elaborate training. This is due to the fact that such systems employ parametric representations which retain local shape information (along with deformations). In contrast to this, we implement an approach which does not deal with such deformations but with a representative (based on maxima and minima) pertaining to their normalised counterparts. These retain their global or archetypal characteristics, after all the deformations pertaining to shape, orientation, positioning and size have been filtered out *a priori* (by heavy quantisation of slopes and coordinates). Since only canonical or archetypal shapes are used, training of the conventional type is dispensed with and robust recognition is possible.

3. Our approach

All the approaches described above use statistical methods of pattern recognition. They all need extensive training. Our approach, on the other hand, is knowledge-based, it seeks to exploit knowledge regarding the task for essentially dispensing with any extensive training.

3.1 *Synthesis of cursive script characters*

In our earlier papers (Ramasubramanian & Rao 1988; Rao & Ramasubramanian 1991) we addressed the problem of synthesizing connected handwritten script from individual characters written in isolation (using the weighted average and the Bezier splicing techniques).

Our approach is to treat connected writing as a process of writing individual characters continuously, in the proper sequence, with minimal effort. In cursive script, the transitional link line between adjacent characters takes the form of a gradual anticipatory movement into the next character while the earlier one is still being written. The shapes of the individual character would get altered to a certain extent in this process. However, short of grossly sacrificing legibility, some deterioration in shape is tolerated in favour of smoothness and continuity of shape and movement. We synthesize these transition regions by the concatenation of individual character shapes to generate connected script. We make sure that continuity of motion and shape are preserved in the transition. This results in economy of movement; it also ensures in a smooth and efficient (i.e. minimal effort, minimal time) pen-down motion, as in the case of cursive writing by humans.

To facilitate synthesis, we divide each character into three segments: a prefix, a core and a suffix. The centrally located core (or shape identification) segments of adjacent characters are linked to each other by transition segments which are influenced by the suffix of the earlier character as well as by the prefix of the later one. Our synthesis consists essentially in the generation of the transition segments. The transition segments move gradually from the prefix of one to the suffix of another. We tried out two alternative, essentially equivalent, approaches: a weighted average method and a shape-specific Bezier splicing technique. Both were successful in generating cursive script. We could even replicate the distortion that occurs during rapid natural writing.

We demonstrated (Rao 1993, pp. 1–15) that even individual characters can themselves be visualised as being composed of (or realised as combinations of) simpler (straight line vector) elements, in the same manner as cursive script can be visualised as being composed of individual characters.

3.2 *Decomposition of script characters*

We showed that we could segment characters using either of two criteria equivalently: minima in either the speed of movement or in the radius of curvature of the character shape. Based on this, we made a conjecture that script characters can be visualised as resulting from an effort to trace in rapid succession a sequence of straight strokes or vectors. We called these strokes 'shape vectors' for the character.

We used a simple geometric construction procedure for fixing the slopes and lengths of the shape vectors and for identifying the suffix and prefix segments therein.

The technique that we used to generate cursive script from individual characters was successful in generating individual characters from the shape vectors; each loop or curved segment is generated by the concatenation of two shape vectors.

We thus demonstrated that shape vectors adequately characterise the canonical shape and identity of the original character. We therefore argue that they should provide a basis for script character recognition. Here, we describe an approach for recognition of connected script using parameters which relate to the shape vectors.

3.3 *The recognition procedure*

For cursive script recognition, we use features concerned with points of minimum radius of curvature (maximum writing speed) and maximum radius of curvature (minimum writing speed). In figure 1, the minima bear even numbers and the maxima are the points with odd numbers.

The parameters we use for representation are the X and Y coordinates and the radius of curvature. We quantise Y to four possible values: b and m (base line and midline) for characters such as c and n and h and l (high line and low line) for characters with ascenders and descender such as h and g. We measure X as the distance from the nearest minimum to the left. Figure 1 and table 1 illustrate the scheme. X is n for minimum 6 because it is very near to minimum 2 at its left, and f for minimum 10 since it is far from its left neighbour. The X value is useful in distinguishing between a and u, g and y, o and v etc. We define X as an increment rather than by its absolute value because the coördinates remain unchanged even after concatening individual letters. We quantise curvature to two values: a for anti-clockwise and c for clockwise.

Maxima are represented by the slope angle, quantised as indicated below:

(1) $0 + 20°$: Right (R)
(2) $180 + 20°$: Left (L)
(3) $90 + 70°$: Up (U)
(4) $270 + 70°$: Down (D)

Fluctuations in slope due to varying slants in the angle of writing are eliminated by such coarse quantisation. Most maxima in script have values U or D. The last strokes of the letters v, w and b have value R.

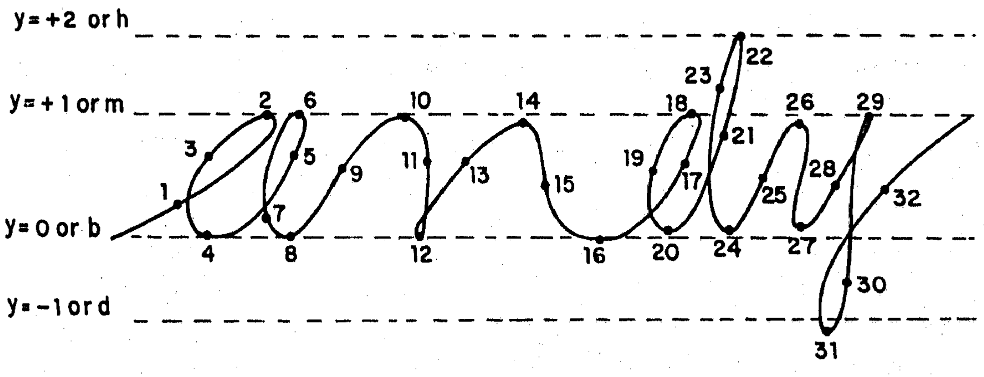

Figure 1. Maxima and minima for the script word 'andy'.

Table 1. Feature vectors for 'a' and 'n' of figure 1.

		character a									character n								
Maxima	Serial No.	1		3		5		7		9	9		11		13		15		17
	Slope	U		D		U		D		U	U		D		U		D		U
Minima	X		f		f		n		f			f		f		f		f	
	Y		m		b		m		b			m		b		m		b	
	Curvature		a		a		a		a			c		c		c		c	
	Serial No.		2		4		6		8			10		12		14		16	

We extract parameters from single samples of individual letters. These constitute the feature vectors for the character. The feature vectors of characters a and n in figure 1 are shown in table 1.

As shown above, all the parameters are subject to heavy quantisation on a coarse grid. This quantisation filters out the fluctuations and shape deformations that occur in normal writing; only the maxima and minima are used to capture shape information. That our representation is adequate is clear from the high quality of resynthesis from the parametric representation. Variations in character sizes are taken care of by x and y quantisation while variations in slant angle are eliminated by slope quantisation.

We simply concatenate the feature vectors of individual letters to obtain the feature matrix of the word. In table 1, feature vectors for a and n are concatenated by merging the last column for a and the first column for n (both numbered 9).

We compile a lexicon consisting of all the words in the vocabulary from the feature vectors of individual characters. A similar feature matrix is obtained for the test word written by the subject, by analysing it in a similar manner. Recognition is accomplished on a best match basis, i.e. either the exact match or, failing this, the nearest match.

We use parameters relating to minima in our feature matrix. It might appear that this represents additional information not provided by the shape vectors. This, however, is not the case. The location, direction and magnitude of the shape vectors and the location of the junction point between the prefix and suffix segments are adequate to compute the location of the minima. Conversely, the lengths of the shape vectors are determined by means of a geometric construction that makes use of the location of the concerned minimum (Rao 1993, pp. 1–15). We use information regarding the minima (rather than equivalent information regarding shape vector size) purely for convenience; this is not a departure from our earlier stated approach.

4. Details of implementation

4.1 *Data acquisition and preliminary processing*

We acquire reference data and test data using a graphics tablet connected to a PC/XT compatible computer system. Reference data (not training data, in the commonly understood sense of the word) is acquired for all 26 English lower-case letters written by the subject. This is used to compute feature matrices of individual characters. Test data (pertaining to the word to be recognized) are similarly acquired.

To facilitate extraction of height information, it is necessary to ensure that the script is in a straight line and that the characters are of a reasonably standard size. To this end, the upper and lower reference lines are provided on the writing surface as in ruled copy books for children. The subject is required merely to write broadly within these lines. There is no need for anything like copy book precision.

As mentioned already, we can locate maxima and minima in the character curve on the basis of the radius of curvature or the speed of movement of the pen. Once these are located, the relevant parameters are extracted and quantised easily. We used three different methods (all dependent on the determination of the radius of curvature) for locating these.

Firstly, we can use the analytical expression for radius of curvature of the character in terms of the 'x' and 'y' components of the pen velocity and acceleration, using the expression,

$$(x'y''-y'x'')/[x'^2+y'^2]^{3/2}.$$

Secondly, the curvature 'ρ' is also the arc derivative of the slope angle, 'θ'. We can therefore compute 'ρ' as a mean over '$2n$' points using the relationship

$$\rho(j)=[\theta(j+n)-\theta(j-n)]/(2n.t.v).$$

Here, 't' is the sampling period, 'v' is the pen speed and '$\theta(n)$' the slope angle of the curve at the nth sampling point.

For computational simplicity, 'v' can be taken to be constant and be dropped from the denominator; 'n' and 't' being constants, the curvature can be taken as being proportional to

$$[\theta(j+n)-\theta(j-n)].$$

Pen speed 'v' of course, is minimum for maximum 'ρ' and maximum for minimum 'ρ'. If we assume this to be constant, our apparent minima will be higher and maxima lower. Despite this, we are able to locate the maxima and minima very effectively (see figure 2). The maxima in pen speed are minima in radius (*i.e.* our minima).

Thirdly, it is easy to locate minima (and maxima) using the fact that the stylus travels fast and almost in a straight line near the maxima but is slow and almost doubles back on itself as the curve turns nearly 180° around that minima. The chord distance (not the arc distance) between two sample points separated by constant time would therefore be minimum around minima and maximum around maxima.

Handwritten characters are usually not vertical in their orientation; they tilt to a greater or lesser extent, usually to the right. Even a left tilt, though less frequent, is not uncommon. This tilt or slant remains more or less invariant for each person and is one of the features that characterise an individual's handwriting. It would seem very desirable to eliminate this slant so that we can deal with more standard – essentially vertical – characters. We did this by means of a number of slant estimation and correction processes but found that, except for extreme slants, this is not really essential, since our method of parameter representation and quantisation for the script characters is robust enough to tolerate even major slants in the writing.

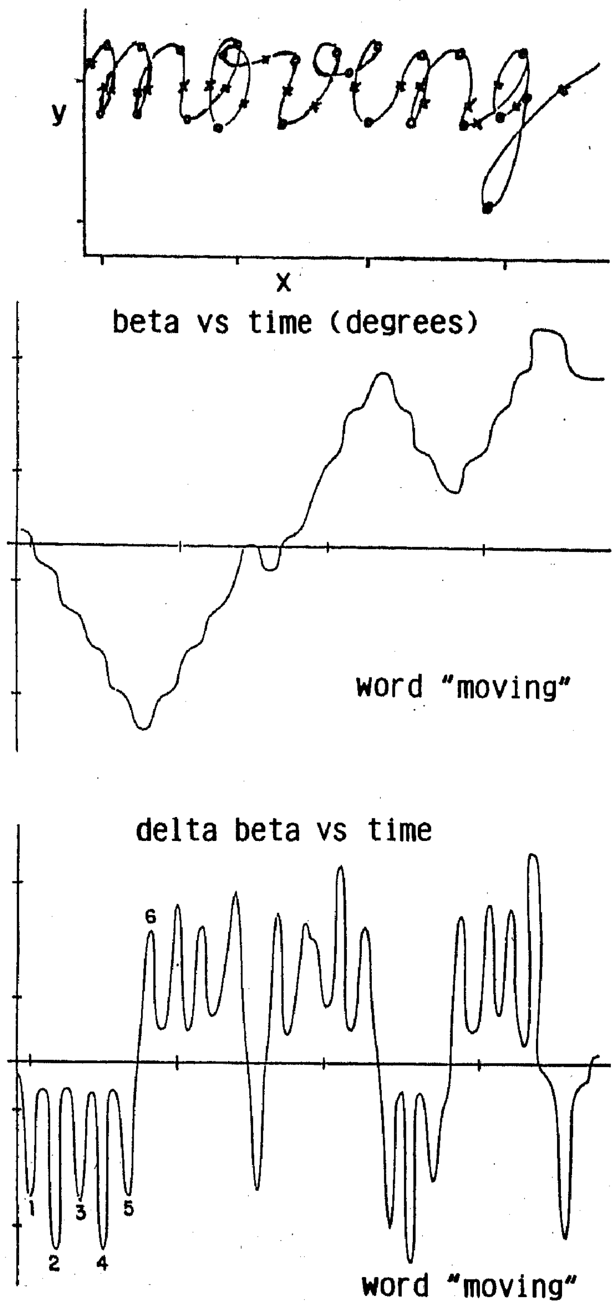

Figure 2. Radius of curvature against time for the word "moving" by the slope slant method.

4.2 *Word dictionary compilation*

We assemble each word in the dictionary by linking parameters of the constituent characters in a sequence. It is important to avoid duplicate or redundant maxima in this process. Each character has two maxima, one at each extremity. During concatenation, of two characters, the last maximum of the previous character and the first maximum of the next character are merged and one maximum (the second) is eliminated.

Some subjects tend to start with a minimum on the base line while writing isolated words or characters. We discard such spurious minima during concatenation. We also provide for certain letters being written in two or more different ways; we treat the different variants of a character as different characters. The dictionary also includes multiple versions of words using such letters.

4.3 *The recognition procedure*

It is important that the procedure for comparing the extracted features of the input word with the pre-stored features of the dictionary words be efficient. We narrow down the list of prospective candidates as far as possible on the basis of a rough match and perform detailed matching only within this short list. We use the number of minima in a word as the criterion for short listing. We reject all dictionary words where the mismatch in the number of minima exceeds three. A detailed matching of individual words is attempted only for words where the mismatch is smaller.

If an exact match is found within the short list, recognition is successful. Failing this, we select the best match, using the following procedure. We impose a penalty score for each non-matching minimum (-10 points) and each non-matching maximum (-2 points). We choose the word in the short-list with the minimum penalty score. Recognition fails if either the short list is empty or if, for all words in the short list, the mismatch during detailed matching turns out to be excessive.

We use a dynamic programming method of comparison to allow for the insertion of spurious maxima or deletion of genuine ones.

4.4 *Ambiguities near the extremities*

A few extra points might be digitised when the stylus moves away from the paper after writing a word. This might look like a small curve in an entirely different direction. In normal script, on the other hand, there are no discontinuities close to the end of a character. We eliminate such discontinuities (and any points that follow such a discontinuity). We discard the first minimum of the test word if it lies on the base line (since no character and hence no word has its first minimum on this line). Also, we ignore the last maximum of the word, since this may not always be present in normal writing.

We use finite difference methods throughout to ensure immunity to noise. This means however that we might miss features very near the start and the end points, if the subject uses very short prefixes and suffixes while writing isolated characters during the reference data acquisition phase. This is true even in the case of start and end segments for test words. Also, in view of the quantisation procedure adopted for the positions of the minima and maxima, there could be problems if the subject writes the characters in a word too close to one another. We did not encounter such problems in our experiments.

4.5 *Missed and spurious minima*

Occasionally, we may miss a minimum: e.g. if the lower loop of the letter 'y' or 'j' is almost circular. We introduce a minimum between two existing maxima, if the slope changes over a range of say 220° without the occurrence of a minimum.

Spurious minima may occur due to wiggles or tremors in writing and we use a few simple techniques to eliminate them. Firstly, we fix thresholds and ignore all maxima and minima whose values are below these. Secondly, we expect minima to be physically well separated along the length of the curve. Hence, if two consecutive minima occur too close to each other (separated by less than a fixed number 'n' of sample points), then we discard the second. Alternatively, we can measure the difference in slope angles on either side and check this against a specific (minimum) threshold. Also, there are specific locations where only minima can occur. We define such regions in relation to the reference lines. We discard minima which do not satisfy these criteria as being spurious We found each of these techniques to be quite effective in eliminating spurious minima.

4.6 *Ambiguous segments*

Points of inflection, i.e., points where the curvature crosses zero, are often misleading. For example, we expect the downstroke for 'f' to be anticlockwise throughout. However, there is in some cases a momentary shift to clockwise movement midway and then back again. These, however, get eliminated by the criteria we use.

We give low weightage to minima and the corresponding maxima existing on the transition segments in the dictionary word, especially those pertaining to characters with a horizontal suffix such as 'b', 'o', 'v' and 'w'. This is because, in general, this transition shows wide variability.

5. System performance

We established the feasibility of the overall approach in a preliminary experiment which used a minimal set of parameters: the sign of the curvature at the minima and the direction of movement at the maxima (up, down, right and left). We could accomplish word recognition even with such drastic data abstraction. Understandably, there is confusion between similar letters such as 'e' and 'l', 'u' and 'a' etc.

We used the following in a second experiment:

(i) *for the maxima*: location (normalised and quantised) and direction of movement (clockwise or counter clockwise);
(ii) *for the minima*: direction of movement (quantised: up, down, left and right).

The results we obtained with this set of parameters are summarised in table 2.

6. Script recognition in the off-line mode

6.1 *Off-line to on-line conversion*

We assume that off-line information regarding the character of word to be processed is collected and made available in the raster scan mode. Our object is to organise

Table 2. Recognition scores for experiment 2.

	Run 1	Run 2	Run 3
Vocabulary size	24	67	67
Correct words recognised by exact match	20	39	57
Correct words recognised by nearest match	04	24	09
Number of words incorrectly recognised	nil	04	01
Recognition score	100%	94%	98%

the 'points' so provided in each scan line into individual tracks or sequences of adjacent or connected points. We then merge these into a single track i.e. a sequence of points which recreates the time order in which the curve has been traversed during writing.

6.1a *Formation of track segments*: At each point, a decision has to be made whether it is part of an earlier track. The criteria for this are:

(i) a distance threshold – that it be within a certain distance of one of the two ends of the track.
(ii) a slope threshold – that it be within a certain distance from the expected position of the point, as computed by linearly extrapolating the track.

We use only the distance threshold in case of single point tracks. We carry out extrapolation using the slope criterion based on the average slope computed using the previous three or four points in the track. We take a new point to be a part of an earlier track if these criteria are satisfied, and add it to that track; else, we form a new track.

This procedure yields track segments composed of between 10 to 20 points each. A few very small segments (of less than three points) also remain. We absorb each such segment into a large track if this can be done by inserting a point in the gap, but only provided that after this insertion, distance and slope matches are within specified limits.

6.1b *Formation of tracks*: We need to combine the track segments so formed into one track. This is done as follows. We compute the degree of mismatch between two tracks pair-wise, as a linear function of the position mismatch and slope mismatch at the joining point. We also join track segments, pair-wise, best match first; if the procedure is successful, we will eventually end up with a single track.

This method works quite well even when the track crosses itself. There is a problem, however, when two segments of the track overlap, merging into one. Track consolidation cannot be completed in such a case because, two segments having merged, we would be one segment short. Also, there would be two different ways of connecting the segments (see figure 3); we resolve this ambiguity by using the property that points of inflection are quite rare in cursive script; and hence that track curvature

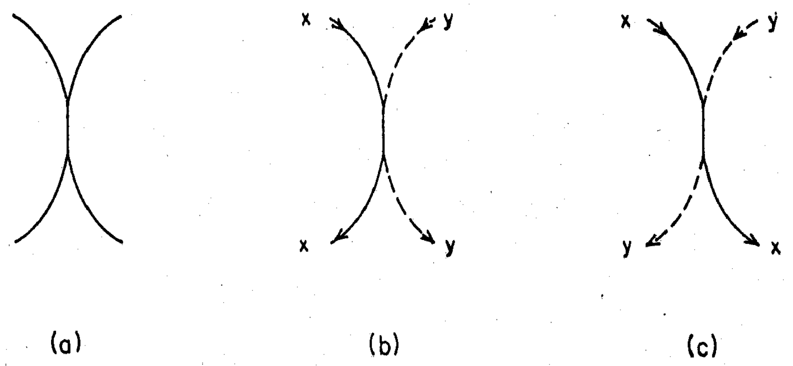

Figure 3. (**a**) Overlapping segment. (**b**) & (**c**) Two different ways of joining the segments. Case (b) (where the curves remain clockwise on either side of the common segment) is chosen; (c) is very unlikely.

Table 3. Recognition with off-line and on-line data: total number of words 47.

	Correctly recognised		Wrongly recognised	Connection failure
	Exact match	Nearest fit		
Raw on-line data	39	7	1	Not applicable
Reconverted data (on-line to off-line to on-line)	20	15	5	7

does not change sign (from clockwise to anti-clockwise or vice versa) on the two sides of the common segment. Thus, we link the two clockwise segments together, as also the two anti-clockwise segments.

6.2 *Comparison of off-line and on-line performance*

We use the following procedure to evaluate the performance of the off-line to on-line conversion procedure. We collect word data on-line and convert it into off-line raster scan mode by a fairly simple procedure. This procedure introduces errors due to quantisation. We provide the raster scan information so generated as input to the off-line to on-line conversion procedure.

This original and reconverted on-line data sets are independently used for recognition. Table 2 shows the results.

These results demonstrate the effectiveness of the off-line to on-line conversion procedure. It may be noted that while off-line recognition scores do drop, this is due more to connection failures rather than due to misrecognition. In other words, the system tends to err on the safer side of not connecting tracks rather than connecting them wrongly.

6.3 *Problems in off-line to on-line conversion*

An analysis of the failures of track connection reveals the following problems.

(i) Due to slow pen movement, the density of points is very high near the beginning and end of each word. This makes track creation difficult. A solution is to eliminate short segments at the beginning and end of each word. This seems to work in most cases. This problem, it may be noticed, would exist only when off-line data is created from data acquired on-line and will not occur when information is acquired directly from a scanner.
(ii) Where tracks turn sharply, track segments might be assembled in the wrong order.
(iii) We have already discussed the problem of overlapping segments.

7. Discussion and conclusions

7.1 *Knowledge based system*

The statement that our system is knowledge-based needs justification. This knowledge in our system is implicitly contained in the model we propose for synthesis of cursive script (from individual characters) and of the characters themselves (from shape vectors) and in the parameters chosen for representing the signal.

Our system is able to operate without training because it incorporates task domain knowledge in its structure. In a manner of speaking, all recognition systems require knowledge regarding the task domain in order to perform recognition successfully. Statistical systems start without any a priori knowledge. Such classifiers acquire this knowledge in a process of elaborate training. Hidden Markov Model and neural nets incorporate some knowledge about the task domain in terms of the exact topology chosen for them. Since, however, this knowledge is minimal, such systems still need extensive training. A knowledge-based system, on the other hand, incorporates a significant amount of task specific knowledge and, as a consequence, is able to operate with minimal training (or even without any training at all). Our system is knowledge based in this sense.

Task domain knowledge is known to be useful for signal estimation in the presence of noise: e.g. cleaning a scanned picture of a script word into an $x-y$ displacement representation of pen tip movement, time domain representation of the speech signal into spectrographic, formant trajectory or vocal tract area function type of representations. The following paragraph illustrates how this knowledges is useful even for our own recognition task.

Script recognition in real life gets complicated due to context effects, sloppy writing, inter-subject variability, etc. Most recognition systems (e.g. HMM systems; Kohonen nets) seek to model these distortions, delineate class distributions (e.g. by modelling the probability densities) and select class prototypes. We take into account the fact that there exists a notional ideal pattern (for each script character or word), which the writer is attempting to generate while he writes. The distortions that occur in actual writing can then be treated as added noise. Our aim, in a sense, is to map the noise-added version into the copy book counter part and this is accomplished in our system. We do this in two stages. To start with, we use a parametric representation that ignores the actual and detailed shapes of the character. We, thus, are able to filter out the deformations and retain only the broad shape features. In addition, we quantise the values of the parameters in a coarse grid, which filters out fluctuations due to character postioning, slant orientation, size and spacing.

On the other hand, conventional pattern classifiers would have operated on the parameter matrix and estimated the mean, variance and spread by means of an extended training phase, accomplishing recognition by a distance criterion.

A major advantage of our recognition approach is that it does not require a training phase. The dictionary words need not even be fed in the script form. We need merely to 'initialise' the system by feeding it one sample each of the 26 lower case script letters of the alphabet. The robustness of the method is amply demonstrated by the fact that performance does not degrade significantly even when the initialisation (training) and test samples are written by different subjects.

In its present form, this method is very suitable for vocabularies in the range of upto a few hundreds of words. Currently we make a straight comparison between the feature vectors of the test word and the dictionary words; we thus pay a heavy penalty for spurious (or missed) maxima and minima because these will cause the remaining part of the word to be misaligned. With more elaborate dictionary match methods (e.g. stack-search algorithms), the system can be used even with significantly larger vocabularies. The features extracted by the present system can be used for neural network and HMM techniques. Preliminary investigations in this direction are very promising. The recognition scheme works very well for on-line inputs. It imposes only a few acceptable constraints.

While the principles underlying the present approach are general enough to be valid for all cursive scripts, they are particularly well shifted for the Roman scripts. With some changes, it should be possible to apply them for machine recognition of cursive scripts in, say, Indian langauges.

The scheme has been tried so far mainly with data acquired in the on-line mode. It has been shown that it is capable of being extended even for the recognition of page scan data, by converting this information into the on-line mode. The results obtained demonstrate the feasibility of the approach, and the potential for improvement in performance with further finetuning of the techniques.

References

Bozinovic R M, Srihari S N 1989 Off-line cursive script word recognition. *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-2: 68–83

Ehrich W, Koeheler K J 1975 Experiments in the contextual recognition of cursive script. *IEEE Trans. Comput.* EC-24: 182–195

Farag R F H 1979 Word-level recognition of cursive script. *IEEE Trans. Comput.* EC-28: 172–175

Kundu A, He Y, Bahl P 1989 Recognition of hand-written word: first and second order hidden Markov model based approach. *Pattern Recogn.* 22: 283–297

Mantas J 1986 An overview of character recognition methodologies. *Pattern Recogn.* 19: 425–430

Mermelstein P, Eden M 1964a A system for automatic recognition of handwritten words. *Fall Joint Comput. Conf., AFIPS Conf. Proc.* 25

Mermelstein P, Eden M 1964b Experiments on computer recognition of connected handwritten words. *Inf. Control.* 7: 255–270

Ramasubramanian V, Rao P V S 1988 Connected script synthesis by character concatenation – An overlap and weighted average formulation. *Comput. Sci. Inf.* 19: 1–10

Rao P V S 1993 Shape vectors: an efficient parametric representation for the synthesis and recognition of hand script characters. *Sādhanā* 18: 1–15

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

Riseman E M, Ehrich R W 1971 Contextual word recognition using binary diagrams. *IEEE Trans. Comput.* C-20: 397–403

Sayre K M 1973 Machine recognition of handwritten words: A project report. *Pattern Recogn.* 5: 213–228