PERGAMON

Pattern Recognition 34 (2001) 37–46

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Syntactic methodology of pruning large lexicons in cursive script recognition

S. Madhvanath[a], V. Krpasundar[b], V. Govindaraju[c,*]

[a]*IBM Almaden Research Center, San Jose, CA 95120, USA*
[b]*Viewlogic Systems Inc., West Marlboro, MA 01752, USA*
[c]*CEDAR, Department of Computer Science, SUNY at Buffalo, Amherst, NY 14228, USA*

Received 11 September 1997; received in revised form 2 June 1999; accepted 15 September 1999

Abstract

In this paper, we present a holistic technique for pruning of large lexicons for recognition of off-line cursive script words. The technique involves extraction and representation of downward pen-strokes from the off-line cursive word to obtain a descriptor which provides a coarse characterization of word shape. Elastic matching is used to match the image descriptor with "ideal" descriptors corresponding to lexicon entries organized as a trie of stroke classes. On a set of 23,335 real cursive word images the reduction is about 70% with accuracy above 75%. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Off-line handwritten word recognition; Lexicon reduction; Holistic matching; Shape descriptor; Trie organization; Elastic matching; Postprocessing; Cursive script

1. Introduction

Research in offline handwritten word recognition has traditionally concentrated on relatively small lexicons of ten to a thousand words. Several real-world applications, such as the recognition of English prose, involve large lexicons of 10,000–50,000 words. Recognition with such lexicons may be made efficient by initially eliminating lexicon entries that are unlikely to match the given image. This process is called lexicon reduction or lexicon pruning, and has the desirable side effect of improving recognition accuracy by reducing classifier confusion [1].

The approach to lexicon reduction described in this paper is inspired by the approach for *online* cursive words taken by Seni et al. [2]. A cursive word may be characterized as a sequence of alternating upstrokes and downstrokes. It has been suggested that downstrokes are more important than upstrokes for the reason that they are usually part of a letter while upstrokes are often ligatures used to connect letters [3]. A descriptor of word shape may be built from a coarse characterization of the shapes of downstrokes.

While the stroke sequence may be readily extracted from online cursive script, extracting the same from off-line script words is complex and computationally expensive. In this effort, we are concerned with coarse features of downstrokes, rather than the exact trace of downstrokes. We therefore adopt a heuristic strategy that detects downstrokes by identifying spatial configurations of local contour extrema.

Seni et al. classify the extracted strokes into a small set of "hard" stroke classes such as medium, ascender, descender, retrograde, and unknown strokes in order to compose a string descriptor, and match the descriptor with the lexicon entries using production rules. In this paper, an alternative "soft" representation of downstrokes is proposed. Each stroke is represented by its normalized extensions into the upper and lower zones of the word. The stroke sequence extracted from the image

* Corresponding author.
E-mail addresses: srig@almaden.ibm.com (S. Madhvanath), ksundar@viewlogic.com (V. Krpasundar), govind@cse.buffalo.edu (V. Govindaraju).

PII: S0031-3203(99)00201-0

is matched with the ideal strokes of lexicon words by an elastic matching scheme.

1.1. Measuring lexicon reduction performance

Given a set of n word images and corresponding lexicons, let us denote the lexicon corresponding to image x_i by L_i. A lexicon reduction algorithm takes as input x_i and L_i and determines a reduced lexicon $Q_i \subseteq L_i$. We denote the event that the truth t_i is contained in the reduced lexicon by a random variable A, defined as $A = 1$, if $t_i \varepsilon Q_i$; $A = 0$, otherwise. The extent of reduction is captured by random variable R, defined as $R = (|L_i| - |Q_i|)/|L_i|$.

Three measures of lexicon reduction performance are defined:

- *Accuracy of reduction:* $\alpha = E(A)$,
- *Degree of reduction:* $\rho = E(R)$, and
- *Reduction efficacy:* $\eta = \alpha^k.\rho$.

Note that $\alpha, \rho, \eta \in [0,1]$. The accuracy and degree of reduction are usually related inversely to each other. The accuracy α can often be made arbitrarily close to unity at the expense of ρ. The two measures are combined into one overall measure η. The emphasis placed on accuracy relative to the degree of reduction is expressed as a constant k, which in turn may be determined by the particular application.

2. Extraction of downstrokes from offline cursive script

The extraction of temporal information (stroke sequence or trace of stylus) from offline script is an interesting area for research and methods have been proposed for both binary and gray-scale images [4,5]. However, the analysis of the offline word necessary to reconstruct the stylus trace is complex and computationally expensive.

Given that we are only concerned with coarse features of downstrokes, rather than their exact trace, a computationally efficient heuristic approach may be adopted for the identification and ordering of downstrokes. The contour of the cursive word is extracted from the binary raster image and represented as chain code. After various preprocessing operations, downstrokes are extracted from the contour by heuristic grouping of local extrema. Each of these steps is briefly described below.

2.1. Preprocessing

The chain-code representation of the contour is generated by analyzing connected components in the binary raster image. Character slant normalization and smoothing of the contour are performed. Small components are discarded as noise. Floating components such as dots of '*i*'s are also discarded in this step. Since such components cannot possibly be part of the downward strokes — the feature of interest in our methodology. Hence, their removal does not affect recognition.

It is desirable that the technique for stroke detection be tolerant of discontinuities along the trace of the script word introduced either by the author or as an artifact of binarization, in so far as each of the components created remains predominantly cursive. Consequently, the contour representation of the word may be expected to comprise of one or more exterior contours and any number of interior contours. The latter correspond to closed loops in the image. Exterior as well as interior contours are sorted from left to right, by column position of the centers of their bounding boxes. Each exterior contour is divided into upper and lower contours segments and global reference lines determined from the histogram of contour crossings [6].

Spurs are points of high curvature on exterior contours corresponding to ends of strokes. Spurs are traced starting from the tip until the two sides of the contour begin to diverge. The mean distance between opposite sides is taken as an estimate of stroke width. The final estimate is computed as the average over all spurs detected in the image, and forms the basis for all thresholds used in the matching of extrema.

The algorithm employed for detection of downstrokes is based on grouping local extrema on the upper contour with those on the lower contour using a small set of heuristic rules. The procedure may be divided into two steps or phases:

1. Detection of extrema and chords on upper and lower contours
2. Matching of upper contour maxima with corresponding lower contour minima

2.2. Detection of contour extrema and chords

1. *Determination of local extrema*: For each exterior contour, local maxima and minima are extracted separately from upper and lower contour segments. The ordering imposed on extrema by the left-to-right traversal is preserved. Strict alternation of local maxima and minima is ensured by the extraction algorithm, while spurious extrema caused by pixel noise on an otherwise smooth contour are eliminated. Each extremum is labeled as one of four: *lower minimum, lower maximum, upper minimum, upper maximum* (Fig. 1).
2. *Determination of peaks and valleys for interior contours:* Only the global maximum and minimum are determined for interior contours and are termed the *peak* and *valley* points of the interior contour respectively. Each interior contour therefore has exactly one peak point and one valley point.

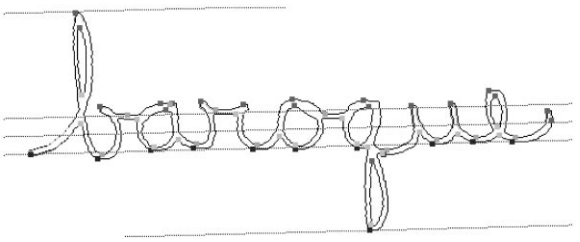

Fig. 1. Local extrema on exterior contours are of four types: upper maxima (red), upper minima (blue), lower maxima (pink), lower minima (black).

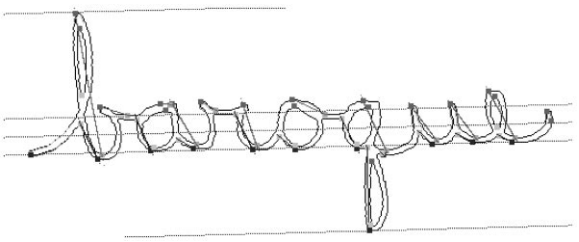

Fig. 2. Twin extrema on the exterior contour are paired to form upper and lower chords.

3. *Upper and lower chords:* Potential downstrokes are indicated by the transitions from local maximum to minimum on the upper or lower contours. A binary commutative relation *twin* is defined between contour extrema as follows: The *twin* of an upper maximum is defined to be the succeeding upper minimum in the order of left-to-right traversal. The twin of a lower minimum is defined to be the preceding lower maximum. Thus, every lower minimum has its immediately preceding lower maximum as its designated twin, except when there is no preceding lower maximum. Similarly, every upper maximum has its immediately succeeding upper minimum as its designated twin, except when there is no succeeding upper minimum.

   The pairs of twin-extrema on the upper and lower contours are termed as *upper chords* and *lower chords* respectively and are depicted as line segments in Fig. 2. A chord presents partial evidence for the existence of a downstroke.
4. *Auxiliary twins and auxiliary chords:* Since chords fail to identify downstrokes within loops, special procedures are required for the processing of interior contours. In general, interior contours are suspect because of (i) spurious interior contours caused by ligatures — for example, the loop of an 'o' may be divided into two interior contours by a "loopy" ligature, and (ii) the presence of open- and closed-loop versions of characters with loops. In an attempt to counteract (i), the relative sizes and positions of adjoining interiors are used to detect and reconstruct split loops.

   In terms of chord structure, open-loop versions of characters vary significantly from closed-loop ones. The downstrokes used in the writing of a character, while apparent in the open-loop versions, are obscured by the closing of the loop (Fig. 3). In particular, upper maxima in the open-loop versions are replaced by peaks of interior contours. When the valley of an interior contour is detected in the vicinity of a lower minimum, the peak of the interior contour is designated as an *auxiliary twin* of the lower minimum. Similarly, when the peak of an interior is found to lie in the vicinity of an upper maximum, the valley of the interior contour is designated as the auxiliary twin of the upper maximum. The auxiliary chords formed by pairing extrema with their auxiliary twins are spatially similar to the chords found on the open-loop versions of characters.

In the matching step that follows, evidence from the upper and lower chords is combined to detect downstrokes.

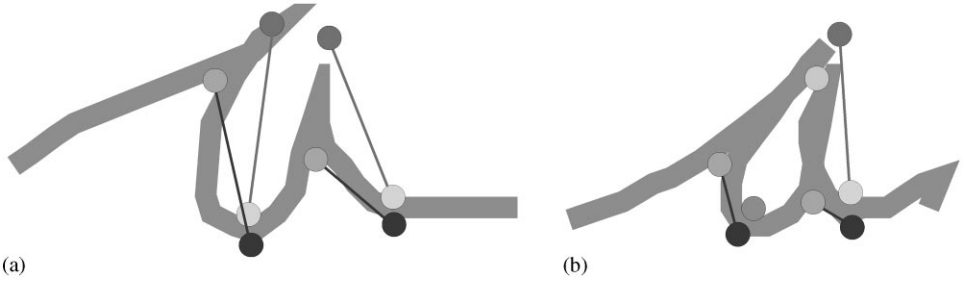

Fig. 3. Differences in chord structure between open- and closed-loop versions of the letter 'a'. Note that one upper maximum is replaced by a interior peak, and an upper minimum by an interior valley.

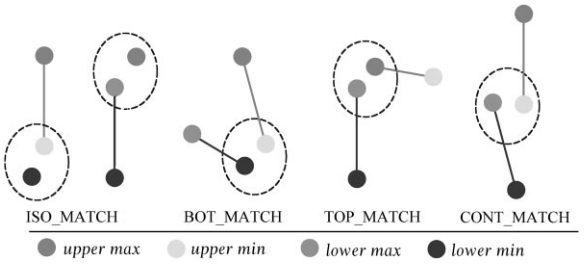

Fig. 4. Basic categories of chord configurations that define downstrokes

2.3. Spatial matching of extrema

A typical downward stroke is approximately bounded by an upper chord and a lower chord (Fig. 2). The actual positions of the chord extrema are dependent on stroke width and the presence of retraces. However, the spatial configurations of upper and lower chords on a downstroke are limited in number, and are captured by four basic configurations ISO_MATCH, BOT_MATCH, TOP_MATCH and CONT_MATCH (Fig. 4). A lower minimum is matched with an upper maximum by heuristic matching criteria which detect different variants of these chord configurations. The influence of pen thickness and image resolution is minimized by expressing all thresholds as functions of the stroke width estimated earlier.

Of these chord configurations, ISO_MATCH is typically found when either the upper maximum or lower minimum is without a twin by virtue of being the last extremum on the upper contour, or first on the lower, respectively. TOP_MATCH and BOT_MATCH are the most common. The detection of the CONT_MATCH configuration is relatively difficult and error-prone since the upper and lower chords may be widely separated in space. Chord configurations are searched for in the order presented in Fig. 4. Auxiliary chords when present are considered when a particular configuration cannot be detected based on the normal chords associated with a given lower minimum and upper maximum (Fig. 5).

A greedy strategy with limited backtracking is used to match lower minima with upper maxima, in the order of occurrence of lower minima on the lower contour. The "current" unmatched maximum, the preceding maximum, and k succeeding maxima are evaluated as prospective matches for a given lower minimum, where k is a small constant (a value of $k = 2$ was used in practice). The priority assigned to these maxima candidates corresponds to the stated order, and a maximum is allowed to match more than one (but at most two) lower minima.[1]

[1] The exception occurs with CONT_MATCH : A maximum involved in a CONT_MATCH with a lower minimum may match no others.

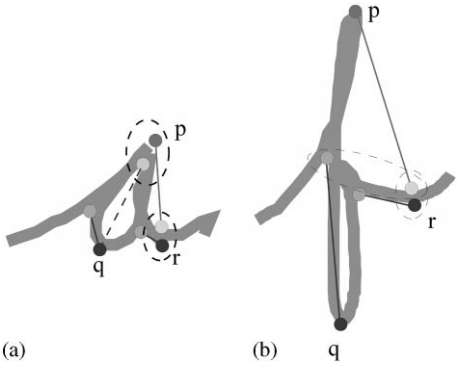

Fig. 5. Spatial chord configurations in two sample cursive characters. In (a), the upper maximum p may be matched with lower minimum q using the TOP_MATCH criterion, by invoking the auxiliary chord of q. p may be matched with lower minimum r using the BOT_MATCH criterion. In (b), CONT_MATCH may be used to match upper maximum p with lower minimum q. Following the CONT_MATCH, p is prevented from matching subsequent lower minima such as r.

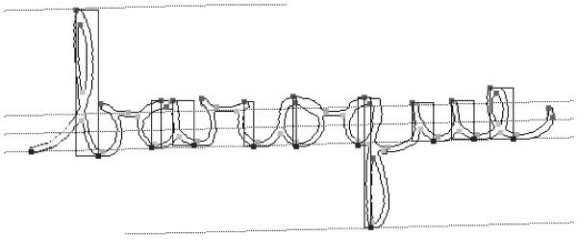

Fig. 6. Evidence from upper and lower chords is combined using knowledge of valid spatial configurations of chords to detect downstrokes, indicated by bounding rectangles.

Costs are computed for each of the maxima as a function of the priority assigned to the maximum, the type of matching chord configuration and the goodness of match. The lower minimum is matched to the upper maximum with the lowest cost. The complexity of the matching algorithm can be shown to be $O(n)$, where n is the number of lower minima.

Each pair of matched extrema delineates a downstroke, and the temporal sequencing of downstrokes follows from the natural order of traversal of lower minima. Downstrokes detected in our example image of 'baroque" are shown by bounding rectangles in Fig. 6.

3. Hard stroke classes and syntactic matching

In the lexicon reduction strategy for online cursive words developed by Seni et al., downward pen strokes

extracted from the online trace of the word are classified into a small number of "hard" stroke classes as ascender, descender, medium, retrograde and unknown strokes in order to compose shape descriptors. The lexicon is organized as a trie [7], and the descriptor extracted from the image is used to syntactically match lexicon entries using a set of production rules which encode valid transformations of ASCII characters into stroke sequences.

A similar approach may be taken for downstrokes extracted from offline cursive words by the procedure described above. A matched extrema pair is classified as one of five stroke types: *ascender, descender, f-stroke, medium* and *unknown*. The classification is based on the vertical separation between the extrema and their position relative to the reference lines determined earlier. Strokes which cannot be unambiguously classified as one of the first four types are classified as *unknown*. Stroke labels are concatenated to produce a shape descriptor. For example, the shape descriptor corresponding to the word 'baroque' is 'AMMMMMDMMM'.

However, there are several disadvantages to using production rules for matching ASCII strings from the lexicon to shape descriptors. Firstly, the production rules themselves have to be designed by hand from observation, and the system cannot easily be adapted for different genres of image data. Secondly, the accuracy and extent of reduction are "hardwired" into the production rules, and are consequently inflexible. There is no mechanism, for example, for decreasing the error rate at the expense of reduction achieved. Thirdly, the output of the reduction process is *abstract-level* — there is no ordering amongst the entries in the reduced lexicon. Such an ordering may be useful for *parallel* decision combination with other classifiers. Finally, the absence of confidence scores makes it impossible to have a thresholding strategy for rejecting poor quality images and poor and ambiguous matches with the lexicon.

In the rest of the paper, we describe an alternative approach to the representation and matching of strokes extracted from cursive script with ASCII strings. The approach addresses and overcomes some of the intrinsic shortcomings of using hard stroke labels and a production-rule driven matching scheme, and is equally applicable to strokes extracted from offline and online cursive script. The approach is characterized by the use of generalized strokes and elastic matching of image descriptors with ideal descriptors of lexicon entries organized in the form of a trie.

Notation. We use lowercase Roman letters $(u, l, m, n, k, s, \ldots)$ to denote numerical variables, and we use uppercase Roman letters $(X, Y, Z, I, P, Q, \ldots)$ to denote strings over the stroke alphabet. The four special stroke *constants* are M, A, D, and F, and correspond, respectively, to the *ideal* medium, ascender, descender and f-strokes.[2] When a stroke is known to take one of these values, we refer to it as a hard stroke. When this is not known, or is known not to be the case, we describe the stroke as a soft stroke.

The accuracy α and reduction ρ of a lexicon reduction system refer, respectively, to the probability of including the truth in the reduced lexicon, and the extent of reduction, as described earlier.

4. Generalized strokes

Given a cursive word (offline or online), let us assume that the reference lines have been detected and are of the form $y = mx + c$, where m is the baseline skew of the word. Each downstroke in the cursive word may be represented coarsely by the Cartesian coordinates of its endpoints. The endpoints of downstrokes are easily extracted from online data. In the offline case, where the stroke has nonzero thickness, the limiting contour extrema may be used to approximate the endpoints.

Let us assume that the coordinates of the endpoints are transformed by a shear or rotation transformation so that the reference lines are horizontal and of the form $y = c$. Let us denote the baseline and the halfline by the lines $y = y_0$ and $y_{1/2}$, respectively. The height h of the middle zone may be computed as

$$h = y_{1/2} - y_0.$$

Further, let us denote the coordinates of the start and endpoint of a downstroke after the skew normalization by (x_t, y_t) and (x_b, y_b), respectively (Fig. 7).

We compute the normalized upper and lower extensions of a stroke as

$$u = clip\left(-1.0, \frac{y_t - y_{1/2}}{h}, +1.0\right),$$

$$l = clip\left(-1.0, \frac{y_0 - y_b}{h}, +1.0\right),$$

where

$$clip(p, q, r) = \begin{cases} p & \text{if } q < p, \\ r & \text{if } q > r, \\ q & \text{otherwise.} \end{cases}$$

Each downstroke can now be represented by an ordered pair (u, l), rather than by a discrete code such as A, M, D, F, or U. A word is represented as a sequence of such (u, l) pairs.

[2] The stroke constant U (for the "unknown" or "unclassifiable" stroke) is of significance only when hard stroke classifications are demanded from the system.

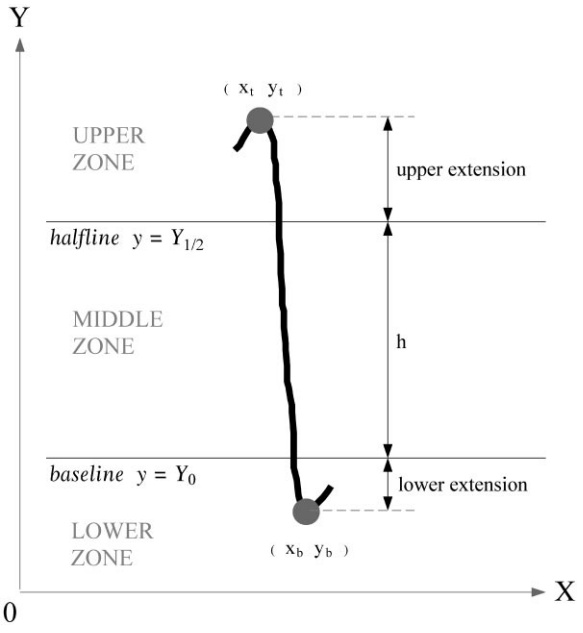

Fig. 7. Upper and lower extensions of a stroke: The coordinates of the limiting contour extrema may be used to compute the upper and lower extensions of the stroke in the offline case.

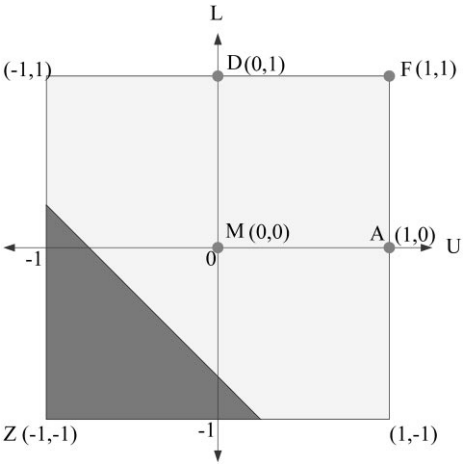

Fig. 8. The U–L representation space: The dark region corresponds to the strokes rendered invalid by the use of a threshold θ on the normalized extent (shown for $\theta = 0.25$).

By definition, u and l are in the range $[-1, +1]$. This delimits our region of interest in U–L space to the region shown in Fig. 8. All strokes, ideal or real, are points within this U–L region. Negative values of u and l indicate, respectively, that the top or the bottom of the downstroke lies within the middle zone. The normalized extent e of the stroke may be computed by adding the normalized extensions of the stroke into the upper and lower zones to the normalized height $h/h = 1.0$ of the middle zone: therefore, $e = u + l + 1.0$.

Since the top of the downstroke always lies above its bottom, the extent of the stroke is always positive, i.e., $e > 0$, which, in turn, implies that $u + l > -1.0$. The downstroke detection method uses a threshold $\theta > 0$ to discard spurious and insignificant downstrokes — strokes that have $e \leqslant \theta$. We thus have the stronger condition $u + l > \theta - 1.0$. In practice, therefore, all valid strokes (i.e., strokes that have $e > \theta$) lie within the light region in Fig. 8.

We note that the idealized hard stroke classes (*medium*, *ascender*, *descender*, and *f-stroke*) form the vertices of the unit square in the first quadrant of U–L space. They are thus represented by the points $M = (0,0)$, $A = (1,0)$, $D = (0,1)$ and $F = (1,1)$, respectively, as shown in Fig. 8.

5. Distance computation

Elastic matching is used to compute the distance between the descriptor extracted from the image and the "ideal" descriptor corresponding to a given ASCII string.

5.1. Ideal shape descriptors for lexicon entries

Our notion of an "ideally written" word refers to the following characteristic features: pure cursive writing style, with no baseline skew or character slant, and reference lines that are equidistant from one another, (i.e., all ascenders and descenders extend to the same height above or below the middle zone, and this extension is itself equal to the height of the middle zone). This notion of the ideal is motivated by the writing instruction imparted in elementary school, and the thesis that the writing style adopted by an individual in adulthood may be modeled as a distortion of the cursive ideal learned as a child. Real-world examples of a handwritten word may be viewed as distortions of this ideally written word.

An ideal descriptor is associated with each character in the lowercase alphabet, and is expressed as a sequence of hard stroke types M, A, D, and F. The ideal descriptor of an ASCII string is computed as the concatenation of the ideal descriptors of the constituent characters. Thus, the character 'a' written cursively is represented ideally by 'MM', the character 'b' written cursively by 'A', and so on, so that the ideal descriptor associated with the lexicon entry 'babble', for instance, is obtained as '$AMMAAAM$'.

5.2. Elastic matching of descriptors

The distance between a given image descriptor I and a lexicon entry is computed using elastic matching as the minimum cost required to match the image descriptor

into the ideal descriptor J associated with the ASCII string corresponding to the lexicon entry. Matching involves modeling of, and associating costs with, spurious, missed, and distorted strokes which collectively cause the actual descriptor extracted from the image to differ from its ideal, represented by the lexicon. We note that these three stroke-error types correspond, respectively, to insertion, deletion, and substitution errors in string-matching parlance.

Spurious strokes detected by the stroke detection procedure must be ignored or "skipped over" in order to facilitate a match between the image descriptor I and the ideal descriptor I^* corresponding to the truth of the image. Stroke retraces and uncharacteristic styles of writing may cause valid downstrokes to be missed by the stroke detection procedure. This corresponds to skipping over a stroke from the ideal descriptor I^*, and the associated cost depends on the stroke type being skipped over. The infinite variability of human handwriting means that very few, if any, soft strokes identified in I will correspond exactly to any of the four hard strokes. Hence we have to associate costs with each substitution of soft stroke for hard stroke in the transformation from I to I^*.

5.2.1. Defining the edit costs

The descriptor I extracted from the image is matched with the idealized descriptors J corresponding to each of the lexicon entries in turn. We follow the convention that I is the *test string*, and J the *reference string*. The lexicon is then sorted by increasing cost of matching, and the top k entries of the ranked lexicon are considered the reduced lexicon. In effect, this procedure pretends that each of the J's is in turn the ideal descriptor I^* corresponding to the truth, and computes a score that quantifies the likelihood of this proposition.

Since we are interested in finding a (minimum-cost) sequence of operations for transforming I into J, the resulting edits will consist of some deletions of the soft strokes in I, some insertions of the hard strokes in J, and substitutions of the former by the latter.

As noted before, all strokes, ideal or real, are points in our U–L stroke representation space (Fig. 8). In U–L space, therefore, substitution is a translation operation. It is therefore natural to use Euclidean distances in this space as a measure of confusion, and hence as a cost-measure for substitution. The deletion and insertion operations may be modeled within this framework as translations to and from the "empty" stroke, if such can be defined. The point $Z = (-1, -1)$ at the lower left corner of the U–L square is chosen to serve this function for three principal reasons:

- Z is independent of any choice of θ.
- Since $\theta > 0$, Z is always in the dark region of the U–L representation space.
- Deletion and insertion costs are strictly greater than zero for all valid strokes, and directly related to their normalized extent e, satisfying the intuitive requirement that the more prominent a stroke, the greater the cost of inserting or deleting it.

In fact, it is clear that Z is the *only* point in the square that possesses these desirable properties.

We now define the deletion, insertion, and substitution costs C_D, C_I, and C_S as follows:

$$C_D(u_i, l_i) = d((u_i, l_i), (-1.0, -1.0)),$$

$$C_I(u_j, l_j) = d((-1.0, -1.0), (u_j, l_j)),$$

$$C_S((u_i, l_i), (u_j, l_j)) = d((u_i, l_i), (u_j, l_j)),$$

where (u_i, l_i) is the soft stroke from I, and (u_j, l_j) is the hard stroke from J.

6. Trie implementation

Elastic matching implemented naïvely is $O(mn)$ in complexity, where m and n are the lengths of the image and lexicon descriptors. The computational cost of matching the image descriptor with each of the lexicon descriptors sequentially is $O(mns)$, where s is the size of the lexicon, and n the mean length of a lexicon descriptor.

1. *Trie of stroke classes*: Let I be the descriptor extracted from the image. Let $P = X.Y$ and $Q = X.Z$ be two lexicon descriptors having the common prefix string X of length $|X| = l$. Initiated from the left, elastic matching involves filling in row-order, a cost matrix with the rows corresponding to a given lexicon descriptor and columns corresponding to the image descriptor. Then the first l rows of the cost matrix when matching I with Q are *identical* to the first l rows when I is matched with P, and so do not have to be recomputed.

   Motivated by this observation, we have organized the lexicon entries and their ideal descriptors as a trie of stroke classes (M, A, D, F). Every node of the trie corresponds to a unique valid prefix of lexicon descriptors. This prefix is common to all the lexicon descriptors found in the subtree rooted at the node, and to no descriptors outside the subtree. The unique prefix represented by a given trie node may be determined by concatenating the stroke classes associated with the trie nodes in the path from the root up to and including the given node. The structure of the trie node `Tnode` is as shown.

```
typedef struct Tnode
{ int id;
int EOW;
int level;
float *cost_row;
```

```
Name *name_head;
Name *name_last;
struct Tnode *mom;
int n_kids;
struct Tnode *kids[MAX_TYPES];
} Tnode;
```

The id of the node is the stroke class that it represents, and may be one of *M*, *A*, *D*, or *F*. EOW ("end-of-word") is a Boolean flag which is set when the prefix *X* determined by the given node is the ideal descriptor of at least one lexicon entry. The level field denotes the depth of the node within the trie.

When EOW is set, the list of lexicon entries with *X* as the ideal descriptor[3] are represented as a linked list of Name structures (shown below), and name_head and name_last point to the head and tail of this list.

```
typedef struct Name
{ char name[BUFSIZ];
   struct Name *nxt;
} Name;
```

The trie node points to the Tnode structures of its parent node (mom) as well as child nodes (kids). Now, consider a trie node t (with the stroke prefix *X* associated with it). Since t is part of elastic matching comparisons with all lexicon descriptors having *X* as prefix, one row of the cost matrix (cost_row) is stored as part of t's Tnode structure. Given the prefix *X*, the entire cost matrix of matching *X* with an arbitrary descriptor *I* may be composed by vertically stacking all the cost rows from the root node down to and including t.

2. *Trie creation*: The trie of lexicon descriptors is created by repeated calls to the function insert_trie() with the ideal descriptors corresponding to lexicon entries. The insert_trie() procedure traverses down the (partially constructed) trie by calling itself recursively to insert shorter and shorter suffixes of the original descriptor, and creating nodes and links when required.
3. *Trie matching*: Given the image descriptor *I*, the task is to match *I* with each of the ideal descriptors represented in the trie. This is accomplished by a depth-first traversal of the trie and is implemented by a call to the procedure trie_match() with the image descriptor. The trie_match() procedure calls itself recursively to compute each succeeding row of the cost matrix and saves the results in the appropriate node structure. When the node also has the EOW flag set, indicating the end of a descriptor, the distance so far (represented by the rightmost entry in the cost row) is read off and associated with all of the lexicon entries ending at the node.
4. *Increasing efficiency*: The efficiency of the matching process may be improved in several ways, some of which are described below.

Lexicon words that are too long or too short compared to the image are unlikely matches. The intuition here is that the system is unlikely to claim a large number of spurious strokes, and is equally unlikely to overlook a large number of real strokes. This observation is encoded as a constraint on the length of lexicon descriptors that are considered by trie_match(). If the length of the image descriptor is *m*, trie_match() does not proceed beyond a certain level $(m + \delta)$ of the trie, where δ is chosen to reflect the above intuition. Similarly, trie_match() computes the cost rows of descriptors ending within the first $(m - \delta)$ levels of the trie (since they are needed by subsequent rows), but does not consider these descriptors as matches.

Since the number of stroke classes is small and finite, the costs associated with missed strokes, spurious strokes and matched strokes can be pre-computed for each of the image strokes and stroke classes. During the actual matching process, these costs can then be looked up from a table rather than computed on-the-fly.

It is also intuitively appealing to limit the *sum* of the deletions (*d*) and insertions (*i*) that can be performed when comparing two descriptors. This is a stricter constraint than imposed by a limit on either of these operations separately. If, in practice, we can determine that any comparison that involves more than δ' deletions and insertions is not viable, then we can reduce the total cost computation involved. Specifically, this allows us to prune the number of alternate paths in the dynamic programming cost computation matrix, when a single lexicon entry (that is *not* too long or short) has used up $(d + i) = = \delta'$ deletions and insertions at a certain point in the cost computation.

7. Pruning strategy

The trie_match() procedure computes for each lexicon entry a distance score. The pruning strategy we have used for evaluation of reduction performance is essentially the same as that described in the context of reduction using perceptual features, and is as follows:

> Given a scored and ranked lexicon, and a rank threshold T_r, the bounds on trie levels searched are employed implicitly and scores are computed for the lexicon words in the permissible trie-levels. These words are ordered by score, and all entries of

[3] The mapping from descriptors to lexicon entries is one-many. Thus, '*FMMMMMD*' maps to 'fairy' as well as 'furry' in the lexicon.

rank greater than T_r are discarded. Finally, all entries in the same score-equivalence class as the $(T_r + 1)$th entry are also *discarded*, to obtain a reduced lexicon whose size is never greater than the threshold T_r.

It is clear that the rank threshold T_r may be used to control the reduction ρ and accuracy α achieved by the system.

8. Empirical evaluation

Tests were conducted using a lexicon of 23,665 words on a set of 760 real cursive word images (Fig. 9). Images were cursive but could have upto one break. The α obtained was 75.504% at 10,000. Table 1 shows the results at various values of α.

The image descriptor is matched with the ideal descriptors of a lexicon using the `trie_match()` procedure described earlier. The search is restricted to two trie levels above and below the extracted length of the image. Thus scores are computed for only a subset of the original lexicon. Cases for which either the stroke extraction failed, or there was no match in the lexicon, are treated as rejects.

The accuracy of reduction α is defined as the probability of including the true street name in the reduced lexicon computed over all accepted test cases. The value of α for different values of T_r in the range [0, 10,000] is tabulated in Table 1.

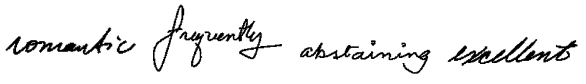

Fig. 9. Sample images in the test set.

Reduction accuracy is seen to saturate at a value of 75% above $T_r = 10{,}000$. The remaining 25% represents cases lost because of the implicit restrictions on trie-levels searched. Such lexicon reduction represents significant savings in computational cost of matching. At $T_r = 10{,}000$, the system delivers reduction of at least 95% in the size of the original lexicon with an accuracy exceeding 75%.

9. Summary

Most of the information in cursive script words is widely regarded to be in the downstrokes, with the upstrokes being used mainly to connect characters. Extraction of downstrokes from offline cursive script is difficult without the use of complex analysis. However, since our interest is in coarse features of the strokes, the system described in this paper uses an efficient heuristic procedure to detect downstrokes that is based upon detection of spatial configurations of local contour extrema.

One approach to lexicon reduction is that of classification of the extracted strokes into a small set of hard stroke types to form a string descriptor, which is then matched with a large static lexicon organized as a trie, using a set of production rules. The reduction achieved by this approach is at the expense of accuracy. Moreover, the syntactic scheme used for matching does not permit thresholding or ranking of the results.

To address these shortcomings, we proposed an alternative representation for downstrokes based on the normalized extensions into the upper and lower zones of the word. An elastic matching scheme for matching of generalized stroke sequences with ideal strokes of lexicon

Table 1
Accuracy α (expressed in percent) for different choices of T_r

T_r	T_r *percent*	α	T_r	T_r *percent*	α	T_r	T_r *percent*	α
1	0.004	1.07	600	2.54	43.74	14,000	59.16	75.50
10	0.04	5.65	700	2.69	45.09	16,000	67.64	75.50
20	0.085	8.48	800	3.38	47.77			
30	0.13	9.83	900	3.81	49.52			
40	0.17	11.44	1000	4.23	51.00			
50	0.21	13.06	2000	8.46	62.44			
60	0.25	14.0	3000	12.68	69.04			
70	0.30	15.48	4000	16.91	72.14			
80	0.34	16.15	5000	21.13	72.95			
90	0.38	16.82	6000	25.36	73.89			
100	0.42	18.97	7000	29.60	79.97			
200	0.85	27.72	8000	33.82	75.50			
300	1.27	31.36	9000	38.05	75.50			
400	1.69	36.87	10,000	42.27	75.50			
500	2.11	41.45	12,000	50.71	75.50			

words was described. The matching scheme models missed, spurious and distorted strokes as Euclidean distances within the stroke representation space. The matching scheme was implemented using a trie-representation of the lexicon for greater computational efficiency. On a set of offline cursive word images the approach was used to reduce a static lexicon of more than 23,000 words by 70% with greater than 75% accuracy.

References

[1] S. Madhvanath, V. Govindaraju, Serial classifier combination for handwritten word recognition, Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR), Montreal, Canada, August 14–16, 1995, pp. 911–914.

[2] G. Seni, N. Nasrabadi, R. Srihari, An online cursive word recognition system, Proceedings of the IEEE CVPR-94, Seattle, WA, June 17–23, 1994.

[3] E.R. Brocklehurst, P.D. Kenward, Preprocessing for cursive script recognition, NPL Report, November 1988.

[4] V. Govindaraju, D. Wang, S.N. Srihari, Using temporal information in off-line word recognition, Proceedings of the 5th USPS Advanced Technology Conference, Washington, DC, November 30–December 2, 1992, pp. 529–544.

[5] D.S. Doermann, A. Rosenfeld, The interpretation and reconstruction of interfering strokes, Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition (IWFHR-III), Buffalo, NY, May 25–27, 1993, pp. 41–50.

[6] S. Madhvanath, V. Govindaraju, Contour-based image processing for holistic handwritten word recognition, Proceedings of the Fourth International Conference on Document Analysis and Recognition (ICDAR-97), Ulm, Germany, August 18–20, 1997.

[7] E. Horowitz, S. Sahni, Fundamentals of Data Structures, Computer Science Press, Inc, 1983.

About the Author—SRIGANESH MADHVANATH received his B.Tech. in Computer Science from the Indian Institute of Technology, Bombay, India in 1990. He obtained his M.S. in Computer Science in 1993 and Ph.D. in 1997 from the State University of New York at Buffalo. He worked as a Research Assistant at the Center of Excellence for Document Analysis and Recognition (CEDAR) from 1991 to 1996, and is presently with the Document Analysis and Recognition group at IBM Almaden Research Center, San Jose, USA. His research interests include Document Analysis and Understanding, Handwritten Word Recognition, and the specification and effective use of context for document understanding systems.

About the Author—V. KRPASUNDAR received his B.Tech. in Computer Science from the University of Kerala (Thiruvananthapuram, India) in 1988. He was awarded a Woodburn fellowship in 1990 to attend the State University of New York at Buffalo. He obtained his M.S. in Computer Science in 1993 and Ph.D. in 1998. His dissertation research is on the use of linguistic information to resolve ambiguities in document understanding applications. He is currently employed with Viewlogic Systems, Inc.

About the Author—VENU GOVINDARAJU received his Ph.D. in Computer Science from the State University of New York at Buffalo in 1992. He has coauthored more than 90 technical papers in various International journals and conferences and has one US patent. He is currently the associate director of CEDAR and concurrently holds the research associate professorship in the department of Computer Science and Engineering, State University of New York at Buffalo. Dr. Govindaraju has been a co-principal investigator on several federally sponsored and industry sponsored projects. He is a senior member of the IEEE.