Machine Vision and Applications (1995) 8:31–40

Machine Vision and Applications

Handprinted word recognition on a NIST data set

Paul Gader[1], Michael Whalen[2], Margaret Ganzberger[3], Dan Hepp[3]

[1]Department of Electrical and Computer Engineering, University of Missouri - Columbia, Columbia, MO 65211, USA
e-mail: gader@sunpg.ece.missouri.edu
[2]Cybernet Systems Inc., 1919 Green Road, Suite B-101, Ann Arbor, MI 48105, USA
[3]Environmental Research Institute of Michigan, 3300 Plymouth Road, Ann Arbor, MI 48105, USA

Abstract. An approach to handprinted word recognition is described. The approach is based on the use of generating multiple possible segmentations of a word image into characters and matching these segmentations to a lexicon of candidate strings. The segmentation process uses a combination of connected component analysis and distance transform-based, connected character splitting. Neural networks are used to assign character confidence values to potential character within word images. Experimental results are provided for both character and word recognition modules on data extracted from the NIST handprinted character database.

Key words: Word recognition – Character recognition – Neural networks – Character segmentation – Document processing

1 Introduction

In this paper, we describe an algorithm for handprinted word recognition and provide training and testing results for words collected from the National Institute for Standards and Technology (NIST) handprinted character database (Wilson et al. 1990). A generic word-recognition algorithm has two inputs: a digital image, assumed to be an image of a word, and a list of strings called a lexicon, representing possible identities for the word image. The function of the algorithm is to assign a match score to each candidate in the lexicon. The match score assigned to a string in the lexicon represents the degree to which the image "looks like" the string.

Our approach is broken into two parts: (1) a distance transform-based computation of multiple segmentations of the word image and (2) Neural network based computation of match scores. The use of the distance transform for segmentation of a word image is a unique feature of our approach as are the features used as inputs to the neural network. In addition, the segmentation algorithm includes rules that are crucial to the performance of the system. Our system achieved high-recognition rates that were very similar on the training and testing sets. This indicates a robust algorithm.

Correspondence to: P. Gader

Significant advances have been made in the field of handwritten word recognition and handwriting recognition in general. These advances have come both in the online recognition area (Plamondon 1993; Tappert et al. 1990) and the offline recognition area (ATC 1992; IWFHR III 1993; IWFHR II 1991). Online recognition refers to recognition of handwriting obtained with an electronic input device. In this case, velocity and pressure measurements are available from the writing device. In offline recognition, a static two-dimensional image is all that is available. Our algorithm is an offline recognition algorithm.

Much of the research in offline recognition has been applied to data sets obtained from the United States Postal Service mail stream. A variety of approaches have been reported since 1990. Several researchers (Chen 1992; Chen and Kundu 1993; Gillies 1992; Park and Lee 1993) have used hidden Markov models for handwritten word recognition. This technique was applied to speech recognition problems prior to its application to handwritten word-recognition problems (Rabiner 1989). It has the advantages that it does not require segmentation of the word into characters and is quick to train. It has a disadvantage similar to that commonly found with neural networks: it is difficult to understand the relationships between the parameters of the model and the performance of the system. This make it difficult to improve performance.

Recognizing words without segmenting them into characters is an attractive feature of a word recognition system, since segmentation is ambiguous and prone to failure. These "holistic approaches" seem to provide auxiliary information to another word recognizer very well, but are not effective as stand-alone word recognizers (Favata and Srihari 1992; Hull et al. 1992; Madhvanath and Govindaraju 1992; Plessis et al. 1992; Senior and Fallside 1993).

Some researchers have tried to recover the dynamic information from static images (Boccignone et al. 1993; Govindaraju 1992; Govindaraju and Srihari 1991). These researchers try to recapture the path that the writing instrument followed while creating the handwritten word. This is also a very difficult problem, and great success has not been realized in solving it. If successfully solved, it would provide significant benefits

to word recognition, since online recognition techniques are more successful than offline techniques.

Some of the most successful results have come from segmentation based techniques that rely on dynamic programming (Gader et al. 1992; Kimura et al. 1992, 1993; Lecolinet 1991; Nohl et al. 1992). These approaches are often referred to as lexicon-driven approaches because an optimal segmentation is generated for each string in the lexicon. An input image is segmented into a sequence of primitive images. Each primitive is a subimage of the original image and ideally consists of a single character or a subimage of a single character. We define a segment to be either a primitive or a union of primitives and a segmentation to be a sequence of segments that uses all the primitives. Dynamic programming is used to find the segmentation that yields that best match to a given string. A match score is assigned to a segmentation and string by matching each segment to the corresponding character in the string with a character-recognition algorithm that returns confidence values for each character class.

An enormous amount of work has been done in handwritten character recognition. An interested reader can review the proceedings of three recent conferences to find a sample of the most recent work [ATC 1992; IWFHR II 1991; IWFHR III 1993). In particular, Casey and Takahashi (1992) have reported results on characters extracted from the NIST data set used here. We compare our results to those reported by them in the experimental results section.

The approach to word recognition described here has been designed specifically for handprinted words. It uses a novel segmentation technique based on the distance transform and novel feature sets for character recognition. The algorithm achieves very robust performance on the NIST data set, actually achieving a slightly higher-recognition rate on the test set than on the training set. The testing and training sets consist of more than 500 words and were randomly chosen. We first describe the character-confidence assignment techniques and then the segmentation algorithm. We include a discussion of the methodology used to construct the data sets. We then present experimental results.

2 Character-confidence assignment

Our approach to word recognition requires a module that can assign a match score between a segment, a subimage of a word image, and a character. We refer to this process as character-confidence assignment. It is distinguished from character recognition by the emphasis on reliably representing the possible class memberships for ambiguous characters as opposed to maximizing the recognition rate. Our system assigns character confidence using feature-based, multilayer, feedforward, neural networks trained with standard backpropagation. Two feature sets are used: the cavity and the direction value features. Both networks were originally developed for handwritten digit recognition and performed very well, with recognition rates in the mid-90% range being achieved with less than a 1% error rate (Gader et al. 1990a; Gader and Whalen 1990).

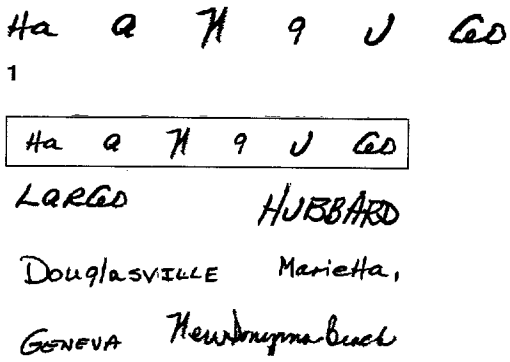


Fig. 1. The classification of these characters, viewed in isolation, is not clear

Fig. 2. The classifications of characters is clarified when they are viewed in words

Several differences exist between handprinted alphabetic character recognition and handwritten digit recognition. The most obvious is that there are more character classes than digit classes. Another is that there are a larger number of ambiguous alphabetic characters than numerals. Characters are usually used in the context of a word. Contextual information is routinely used by humans to read words containing illegible or ambiguous characters. For example, the classifications of the characters in Fig. 1 are not necessarily clear but when viewed in the context of the words written, shown in Fig. 2, they are not ambiguous. Character recognition is also complicated by other differences, such as upper and lower case, cursive representations of letters and the number of disconnected, and multistroke characters. The upper-case characters D, E, F, G, H, J, K, P, T, and Y have frequently occurring instances of multiple disconnected strokes. The digit 5 is the only example of this problem for digit recognition.

2.1 Neural network descriptions

In this section we discuss the feature inputs and architectures of the neural networks. The features are extracted from images that have been size normalized to 24 × 16 arrays.

2.1.1 Cavity features

The cavity features used in our networks are based on features developed by Gillies and Mitchell 1989 and Mitchell et al. 1990. They were adapted to neural networks by Hepp (1991) and Gader et al. (1990). A cavity is a region in the background of a binary image that is bounded by the stroke (the foreground) on at least three sides. There are six cavity-feature types: east, west, north, south, center, and hole. The cavities are named by the direction in which they open: that is, the side on which they are not bounded. Thus a west cavity is a region open to the west, but not open to the north, south, or east. A center cavity is a region that is surrounded on all four sides, but is not a hole. A hole is a region completely enclosed by the stroke.

We compute the cavities on an input image using mathematical morphology (Gader et al. 1991; Gillies and Mitchell

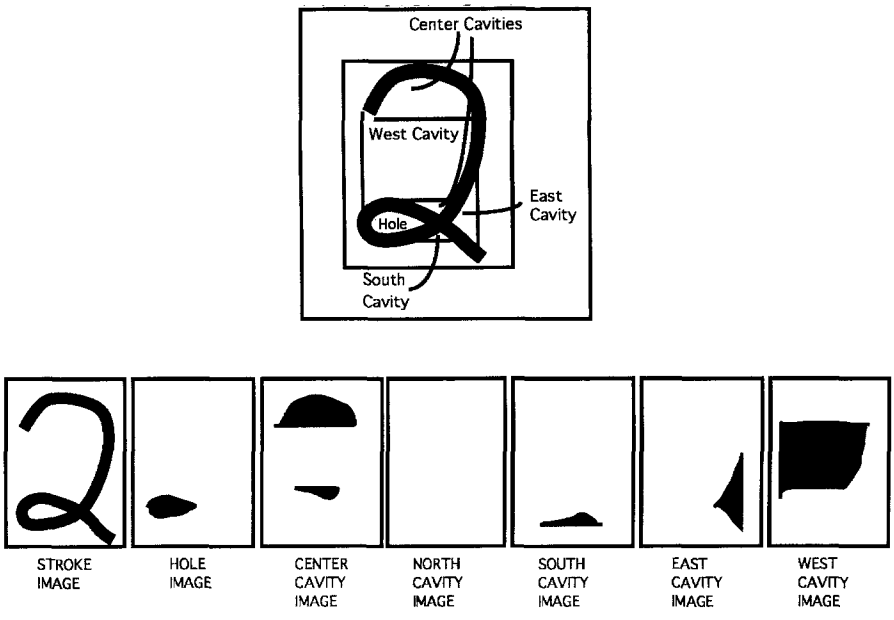


Fig. 3. Cavities and cavity-feature images

1989). A separate binary image, called a feature image, is created for each cavity type. These images are illustrated in Fig. 3. The feature images and the original input image are used to create numerical feature values using zoning. Zoning consists of counting the number of "on" pixels in a rectangular subregion of a binary image. The rectangular subregions are referred to as zones. The zones that we use are 8×8 subregions with upper left corners at the locations in the set { (row, col) | row $= 4 \times i$, col $= 4 \times j$; $i = 0, 1, ..., 4$; $j = 0, 1, 2$} (Recall that the images are all normalized to size 24×16). Thus, for each feature image and the input image we obtain 15 values, the count of the number of "on" pixels in each zone. These counts are linearly scaled between 0 and 1 and collected into a feature vector of dimensionality 105.

The cavity-feature neural networks are 4-layer networks. There is one network for upper case characters and another for lower-case characters. Each network has 105 inputs and 27 outputs. The extra output unit was used for a noncharacter class that was used to decrease the response of the network on input images that did not consisting of characters. The first hidden layers have 65 units and the second have 39 units. These architectures were not optimized.

2.1.1 Direction-value features

The direction-value features provide orientation information of the strokes in zones using the boundary and skeletal pixels. The boundaries of a character are computed, and the character is then thinned to a one-pixel-wide stroke, creating two binary images, a boundary image, and a thinned image. Numerical features are generated by computing counts in zones in each of these images. The zones used in the cavity-feature calculation are also used to calculate the direction values. In each zone, four counts are generated, one for each of four directions: east, northeast, north, and northwest by counting the number of occurences of the associated patterns in Fig. 4. The counts are generated on the boundary and thinned image and the results are combined into a 60-dimensional feature vector. More precisely, let $N_{d,z,i}$ denote the number of occurences of the pattern associated with direction d in zone z in image i where $d = e, ne, n, nw$; $z = 1, 2, \ldots, 15$; and $i = b, t$ (background or thinned). The direction-value-feature vector values are given by

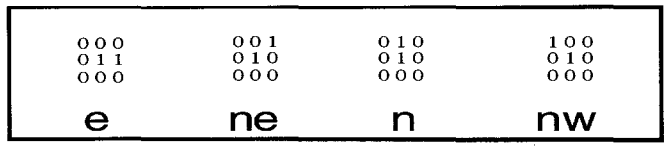


Fig. 4. The directional patterns counted in each zone of the boundary and thinned versions of the original character

$$\left(\frac{(N_{d,z,b} + N_{d,z,t})}{64}\right); \quad \text{for} \quad d = e, ne, n, nw$$

and $z = 1, 2, \ldots, 15$.

The direction-value network configurations are $60 \times 45 \times 35 \times 27$.

These networks are used to assign character confidence in the word-recognition algorithm as follows: let r denote a binary image and let s represent a character class. Let $C_u(r, s)$, $C_l(r, s)$, $D_u(r, s)$, and $D_l(r, s)$ denote the output activations of the output node associated with class s for the upper and lower-case cave and direction-value-feature networks. We define the match between image r and character class s to be the maximum of the averaged output activations:

$$\text{match\_char}(r, s)$$
$$= \max(C_u(r, s) + D_u(r, s), C_l(r, s) + D_l(r, s))$$

3 Segmentation and matching

A segmentation of a word image is a sequence of its subimages. The subimages in a segmentation are referred to as segments. Ideally, the segments of a word image would be images of all the characters in the word and nothing else. However, since handwritten characters are often ambiguous, as shown in Figs. 1 and 2, it is not possible to achieve this ideal unless recognition is somehow linked to segmentation.

One approach to solving the problem of ambiguous characters is to generate multiple segmentations of a word image. Given a string of length n from the lexicon, the word-recognition algorithm matches each segmentation of length n to the string by matching each segment to the corresponding character in the string, using the neural network of the modules for character-confidence assignment described in Sect. 2.

A flowchart of the word-recognition system is shown in Fig. 5. We first provide an overview of the system and then a detailed descriptions. A binary image of the word and a lexicon are input to the system. The connected components are found and used to detect punctuation, dots, and "bars" (such as the top of a "T" or the vertical bar in an "F"). Punctuation is removed, and horizontal bars are grouped with other connected components.

The result of connected component analysis and grouping is a single segmentation, called the initial segmentation. The initial segmentation is matched against all strings in the lexicon with the same length as the initial segmentation. If the match score for any string exceeds a threshold, then strings of different lengths are assigned a score of zero, and this ranking of the lexicon is returned. This initial matching allows easily segmentable words to be recognized with a relatively small amount of computation. Otherwise, multiple segmentations must be generated.

Multiple segmentations are generated by first splitting segments in the initial segmentation and then forming unions. Splitting a segment consists of dividing the segment into a sequence of subimages. Our system requires that each segment to be split into at most three pieces. Each segment is passed through the character-recognition modules before splitting. If the recognition confidence for a class exceeds a threshold, then the segment is assumed to be a character and is not sent to the splitting module. This step is included to reduce the computational overhead of the algorithm, but it can cause failures. For example, an occurrence of the character pair "tt" could be recognized as an "H" and not be split into segments. We refer to the segments obtained after splitting an initial segment as primitive segments. If an initial segment is not split, either because an appropriate splitting location could not be determined or because of high-recognition results, then the initial segment itself is refered to as an initial segment.

Once the initial segments are split, we form segmentations using unions. The relative positions of segments within the word are maintained while taking unions as shown in Fig. 6. Multiple segmentations are formed by taking different unions. We match to the lexicon by matching each string in the lexicon against each segmentation of the same length as the string and assigning the string the highest match score achieved. If there are no segmentations of the same length as the string, then the match score is 0.

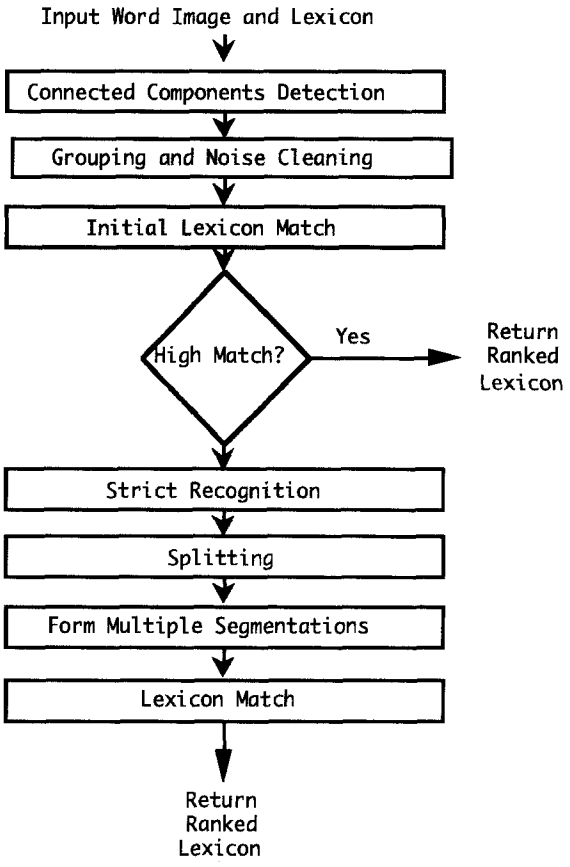


Fig. 5. Word recognition flowchart

Fig. 6. A handprinted word image and the union of two of the initial segments

We now describe the word-recognition modules in detail using pseudocode. We use a coordinate system with x as the horizontal axis and y as the vertical axis. The value of x increases as we move from left to right in an image and the value of y increases as we move from top to bottom in an image. There are also several rules in the algorithm description. We often phrase the rules using the structure "IF x if SMALL" as opposed to "IF x is below a threshold".

3.1 Handprinted word recognition algorithm

Inputs: W = binary image of a word
L = lexicon
Output: Ranking of L indicating strength of match between W and each string in L

3.1.1 Connected components detection

The connected components detection uses a combination of 8-connectivity and 4-connectivity. Let

C = list of connected components of W using 8-connectivity,
C_4 = list of connected components of W using 4-connectivity,
$Q = C - C_4$ where $-$ denotes set difference.

We detect regions that are 8-connected, but not 4-connected. If the 4-components have very little vertical overlap, we consider the region as composed of two regions.

IF $r \in Q$ THEN
  IF $r_1, r_2 \in C_4$ with $r = r_1 \cup r_2$ AND vertical_overlap (r_1, r_2) is SMALL
  THEN put r_1 and r_2 into C and remove r from C

Sort the regions in C from left to right using their positions in the word image W.

Remark: We often use the operation, group(r, s) for $r, s \in C$. It consists of the following steps:

1. Form the union $t = r \cup s$ as shown in Fig. 6,
2. Remove r and s from C
3. Put t into C while maintaining the left to right ordering of C.

3.1.2 Detect bars and group horizontal bars

We detect bars by computing the minimum eigenvalue of the moment matrix of each region $r \in C$. The moment matrix of r is defined as

$$M = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{xy} & m_{yy} \end{bmatrix}$$

where

$$m_{xx} = \sum_{(x,y)\in r} (x - x_c)^2 , \quad m_{yy} = \sum_{(x,y)\in r} (y - y_c)^2 ,$$
$$m_{xy} = \sum_{(x,y)\in r} (x - x_c)(y - y_c)$$

and (x_c, y_c) are the coordinates of the centroid of r. Let λ be the smallest eigenvalue of M.

IF λ is SMALL THEN
  label r as a bar
  IF the height to width ratio of r is BIG THEN
    label r as a vertical bar
  ELSE
    label r as a horizontal bar
  ENDIF

Horizontal bars are not considered likely candidates for characters and are grouped using vertical overlap and distance as criteria. Vertical overlap is appropriate for bars found in the characters "I", "J", "T", etc. Distance is appropriate for "E", "F", etc.

For each $r \in C$ let

  r_p = region immediately to the left of r
  r_n = region immediately to the right of r

IF there is no vertical overlap between r and r_p and between r and r_n THEN
  group r with the closest neighbor
  ELSE
    group r with the neighbor having the most vertical overlap with r
  ENDIF

3.1.3 Detect and group or remove dots

IF $r \in C$ AND r is SMALL and r is NOT CLOSE to the bottom of the word image W THEN
  Let $s \in C$ be the closest region to r.
  IF distance(s, r) is SMALL AND s is a vertical bar THEN
    group(s, r)
  ELSE
    remove r from C
  ENDIF

3.1.4 Detect and remove punctuation

For each region $r \in C$, let bottom(r) = max $\{y|(x, y) \in r\}$
Let baseline(W) = median $\{\text{bottom}(r)|r \in C\}$
IF $r \in C$ and r is SMALL and MOST of r is below the baseline(W) THEN remove r from C.

Remark: the adjectives SMALL and MOST depend on thresholds that change if r is the rightmost region, since punctuation is usually at the end of a word, although not always, as in "St. Louis".

Figure 7 shows a NIST word image that illustrates the dot grouping, punctuation removal, and combination of 4- and 8-connected component analysis requirements.

Remark: at this point, the list of regions C is the initial segmentation of the word image W. We now refer to the elements of C as segments.

3.1.5 Initial lexicon match

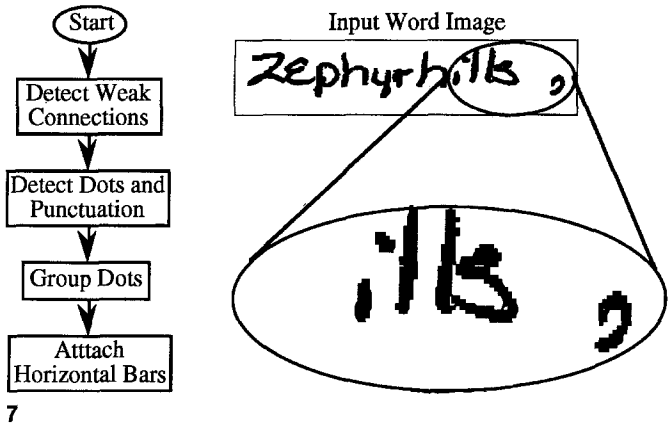
The initial segments are matched against the lexicon. Let $C = \{r_1, r_2, \ldots, r_n\}$ and L_n denote the set of all strings in L of length n. The lexicon L ranked by:

FOR EACH str $\in L$
  IF str $\in L_n$ THEN
    Let s_i = ith character in str, $i = 1, 2, \ldots, n$
    match_score(str) = $(1/n) \sum_{i=1}^{n}$ char_match(r_i, s_i)
  ELSE
    match_score(str) = 0
  ENDIF

If the maximum match_score exceeds a threshold, then this ranking is used as the output without further processing. Otherwise, the algorithm continues.

3.1.6 Strict recognition

For each initial segment $r \in C$, let char_conf(r) = max {char_match(r, c)} where the maximum is taken over all character classes. If char_conf(r) exceeds a threshold, then r is not split.



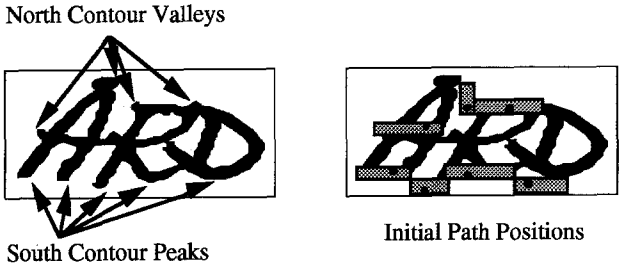


Fig. 7. Connected component analysis involves Identifying both 8- and 4-connected regions, grouping of dots and horizontal bars (not shown) and removal of punctuation

Fig. 8. Finding starting points for start generation

3.1.7 Splitting initial segments

Our approach to splitting a segment is to construct paths that extend from the bottom of the segment to the top of the segment. If the paths are removed from the segment, then the segment will be disconnected. The algorithm constructs paths that stay as far from the stroke as possible without curving too much. This is implemented by following ridges in the distance transform of the background of the segment image. The splitting module consists of three major submodules:

FOR EACH $r \in C$, IF char_conf(r) is SMALL and r is not a vertical bar THEN DO

1. Compute the distance transform of the background of r, $DT(r)$,
2. Generate candidate split locations,
3. Form subimages of r using the candidate split locations and pruning heuristics.

We describe each submodule in more detail:

1. The distance transform of the binary image r is computed on the background of r using 8-connected paths. Each point in the background is labeled with the distance to the stroke. That is,

IF $r(x, y) = 0$ THEN
  $D(x, y)$ = length of the shortest path from (x, y) to a point (x_0, y_0) with $r(x_0, y_0) = 1$
ELSE
  $D(x, y) = 0$
ENDIF

2. The splitting module generates paths that follow ridges in the distance transform of the background. This submodule performs the following functions in pursuit of this goal:
2.1 Find starting points for path generation.
2.2 Trace paths from starting point to bottom and top of segment.
2.3 Prune paths.

We describe each function in more detail:

2.1 The north and south contour of the segment r are computed. The valleys of the north contour and the peaks of the south contour are determined as shown in Fig. 8. Peaks and valleys are treated similary, so we describe the rest of the processing for peaks only. Close peaks are pruned. The remaining peaks are sorted from left to right. Adjacent peaks are used as corners for rectangular regions that are used as search regions. In each search region, relative maxima of the distance transform are located. If there is one or more relative maxima of the distance transform in a search region, then one is chosen as the starting point.

More precisely, define the south contour by

$$SC = \{(x, y) | r(x, y) = 1 \quad \text{and}$$
$$r(x, y_h) = 0 \quad \text{for every } y_h > y\}.$$

The south contour has exactly one point for each nonzero column of the segment r. A point (x_p, y_p) in the south contour is a peak if $y_p > y_{pp}$ and $y_p > y_{pn}$ where $(x_p - 1, y_{pp}), (x_p + 1, y_{pn}) \in SC$.

Let $SCPEAKS$ be the subset of SC consisting of all the peaks of SC and assume $SCPEAKS$ is sorted from left to right. If the distance between two adjacent peaks is below a threshold, then we remove the peak that is closest to it's other adjacent neighboring peak. We prune until all peaks are sufficiently far apart. That is, for each peak p, define

$d_l(p)$ = distance to first peak to the left of p,
$d_r(p)$ = distance to the first peak to the right of p,
$d(p) = d_l(p) + d_r(p)$.
DO WHILE more close peaks (min $\{d(p) | p \in SCPEAKS\}$ $<$ threshold)
  remove peak with smallest $d(p)$ from $SCPEAKS$

For each pair of peaks $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, define search_box(p_1, p_2) to be the rectangular subregion of $D = DT(r)$ with bounding box determined by the points p_1 and p_2. Let

$$M = \{(x, y) \in \text{search\_box}(p) | D(x, y) \text{ is a relative maxima } \}$$

IF $M \neq \phi$ the starting point, sp, associated with be search_box (p_1, p_2) is

$$sp = \text{argmax}\{y | (x, y) \in M\} .$$

The search boxes and corresponding starting points are illustrated in Fig. 8.

2.2 A splitting path is generated for each starting point by tracing from the starting point to the top and bottom of each image. The splitting paths are 8-connected and have either one or two pixels in each row of the initial segment. When tracing the path towards the top of the segment, a 1×5 search region, defined as N in the pseudocode below, is used to generate candidates for the next pixel or pixels on the path. More precisely,

FOR EACH starting point s
Let path(s) = ()
TRACE NORTH
curr_pt = s
DO UNTIL curr_pt is at top of segment
$x_c = x$ coordinate of curr_pt
$y_c = y$ coordinate of curr_pt
push curr_pt onto path(s)
$N = \{(x_c-2, y_c-1), (x_c-1, y_c-1), (x_c, y_c-1), (x_c+1, y_c-1), (x_c+2, y_c-1)\}$
curr_pt = argmax $\{D(x,y)|(x,y) \in N\}$
IF curr_pt = (x_c-2, y_c-1)
THEN push (x_c-1, y_c-1) onto path(s)
IF curr_pt = (x_c+2, y_c+1)
THEN push (x_c+1, y_c+1) onto path(s)
END UNTIL
TRACE SOUTH (* VERY SIMILAR TO
TRACE NORTH *)
END FOR

2.3 Paths are pruned with the same algorithm that prunes close peaks in the north contour with the additional constraint that there be no more than two paths left after pruning.

3. The final submodule involved in splitting initial segments forms subimages using the paths. Given an initial segment r with paths p, we set all points lying on the paths to 0 in r. By construction, this process disconnects r into new connected components that are used as the primitive segments associated with r. More precisely,

FOR EACH $r \in C$, let paths(r) be the final set of paths returned from submodule 2.
FOR EACH $p \in$ paths(r) DO
FOR EACH $(x,y) \in p$ DO
Set $r(x,y) = 0$

Let primitives(r) = connected components(r). An example of splitting paths and the resulting segmentations of an initial segment is shown in Fig. 9.

3.1.8 Generation of multiple segmentations

Once each initial segment has been segmented into primitive segments, the final set of multiple segmentations can be generated. For each initial segment, r_i, we have a sequence of primitive segments, say $P_{i1}, P_{i2}, \ldots, P_{in_i}$, that have been generated by the splitting module. As mentioned previously, we require that $1 \leq n_i \leq 3$. A segmentation of r_i is a sequence

$$S_i = \{P_{ib_1e_1}, P_{ib_2e_2}, \ldots, P_{ib_ke_k}\}$$

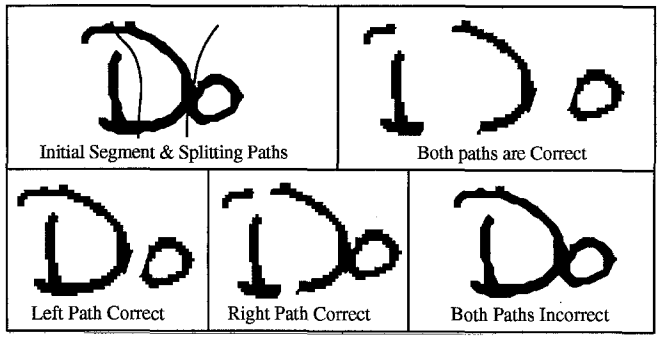


Fig. 9. Possible segmentations of an initial system

where

$$P_{ib_je_j} = \bigcup_{h=b_j}^{e_j} P_{ih}$$

and

$$e_j \geq b_j, b_1 = 1, \quad \text{and} \quad e_k = n_i \, .$$

These unions take into account all possible unions of subsegments of initial segments. One other important case is not taken care of: grouping of segments that are vertical bars. These segments can either be pieces of characters such as D, E, F, etc., or individual characters such as I or L.

A segmentation of the word image W with initial segmentation $\{r_1, r_2, \ldots, r_m\}$ is defined to be a sequence $d_1, d_2, \ldots d_k$ such that $\cup_{i=1}^{k} d_i = \cup_{i=1}^{m} r_i$ where each d_i is either a segmentation of an initial segment or the union of a subset of a segmentation of an initial segment and a vertical bar from a neighboring initial segment. We let $S(W)$ denote the set of all segmentations of W and note that $S(W) = \cup S_n(W)$ where $S_n(W)$ is the set of all segmentations of W of length n. Matching uses the set of all segmentations. Given $S(W)$ and a lexicon L we compute the match score for each string in L as follows:

For each string str $\in L$
Let n = length (str)
Let match_score =
MAX{match(s,str)} $|s \in S_n(W)\}$ if $S_n(W) \neq \phi$
0 else.

The value match(s, str) was defined in the discussion of the initial lexicon matching. The assignment of the match score is the final step in the word-recognition algorithm. The string with the highest match score can be taken as the recognition result. It is more common to use the top few choices in the context of a higher-level system.

4 Recognition results

In this section, we provide experimental results. We first describe the data sets and then provide recognition rates.

4.1 Data sets

The NIST data were collected for training and testing optical character-recognition algorithms and systems. The data consist of 2100 binary images of forms (Wilson et al. 1990). Each form has 33 boxed fields. Subimages of each boxed field are in the data set. There are two boxed fields of isolated alphabetic character data; one for uppercase and the other for lower case. The characters in the fields are written in various orders on various forms. Truth information identifying the sequence in which the characters are to be written in the box is provided. The location of the characters is not given, so the characters must be extracted. We extracted a character set with a computer program and manually screened the results for accuracy.

The subimages of the character fields were preprocessed by removal of the machine-printed text and the box containing the handprinted text. Many of the lower case characters with descenders, such as f, g, j, p, q, and y, extended through the box and had to be carefully recovered with morphological operations (Gader and Whalen 1990b). An automated segmentation and truthing algorithm was applied. Some of the characters were touching and others were broken. The fields were segmented into characters with a combination of simple segmentation techniques, including grouping of horizontal and vertical bars, filtering of small connected components, grouping of dots on i's and j's, connected components analysis, and vertical projection methods. The segmentation algorithm attempted to segment the field into 26 characters. If none of the answers were 26 characters long, then the field was rejected, and the characters from that field were never used. If a segmenter returned a segmentation that was 26 characters long, then the truth file was used to assign a truth to each of the segmented characters, and they were saved in the appropriate files. These files were then manually cleaned.

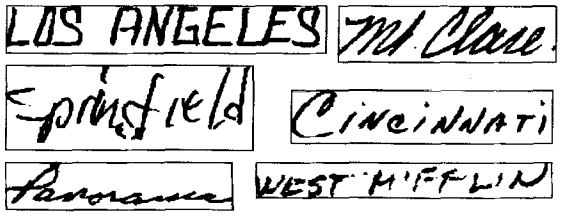
The extraction process led to the creation of a data set of 49 328 characters; 26 134 upper case characters and 23 194 lowercase. This data set contains approximately 45% of the total number of characters available (about 109 200 = 26 characters per box $\times 2$ boxes per form $\times 2100$ forms), but is significantly larger than the one used in (Casey 1992). Training and testing sets consisting of 250 characters from each class were constructed. Samples of the NIST data are shown in Fig. 10.

One boxed field in the NIST database consists of city, state, and ZIP-code words. The city words were automatically extracted from this boxed field with a vertical projection-based segmentation algorithm. They were later screened manually for correctness. The result was a set of 1100 city-word images. This set was randomly divided into two sets of equal size, one for training and one for testing. Samples of word images extracted from the NIST database are shown in Fig. 11. Note that some of the words are written in cursive style.

4.2 Experimental results

The neural networks for character-confidence assignment were trained using the standard backpropagation algorithm. The training rates are given in Table 1. The percentages are the percentage of characters for which the highest output-activation value corresponded to the correct class. The combined feature type refers to the output of the match_char function described previously, which averages the outputs of the cave and direction-value networks.

10



11

Fig. 10. Samples of characters extracted from a NIST database
Fig. 11. Samples of word images extracted from a NIST database

Table 1. Recognition rates for character-confidence assignment modules

Feature type	Case	Training rate (percentage)	Testing rate (percentage)
Cave	Upper	96	91
Cave	Lower	93	77
Direction values	Upper	92	88
Direction values	Lower	88	75
Combined	Upper	97	94
Combined	Lower	93	82

The upper-case characters are easier for both networks than the lower-case characters. This is probably due to the increased ambiguity in lower-case characters. Casey and Takahashi (1992) reported a classification rate of 95% on a testing set of upper-case characters extracted from the NIST database, compared to our combined rate of 94%. Several differences exist our system and theirs. The set their group used was smaller (250 per class in training and 125 per class in testing.) They used one network with only hidden layer. In particular, they combined two feature sets as inputs to the network, whereas our approach was to train networks for each feature set separately and then combine the outputs. The total number of

features used in our networks is 165, whereas they used 184. Finally, our networks were trained with samples of noncharacters to reduce the response to segments of words that are not characters. Despite all these differences, the two approaches yield very similar results.

A lexicon of 746 distinct strings was used for evaluating the performance of the word-recognition algorithm. The same lexicon was used for each word in both the training and the testing sets. The correct string for each word is included in the lexicon. The recognition results, shown in Table 2, were very similar on the training and testing sets. The table shows the percentage of words for which the correct string was ranked among the top n for $n = 0, 1, 2, 3$.

Table 2. Word recognition results with a lexicon of 746 strings

Rank	NIST Training Set (percentage)	NIST test set (percentage)
0	80.15	80.41
1	83.18	82.81
2	84.31	83.36
3	84.50	83.92

A remark on the nature of the training set is in order. The characters used to train the neural networks were taken from boxes of isolated alphabetic characters in the NIST data set. Thus, none of the characters in the character training set appear in the word-training set, (although some of the words and characters may have been written by the same authors). There was no formal training on the word-training set. The training consisted of interactive development of the algorithms and rules used to segment and match.

5 Conclusion

A system for recognizing handprinted words has been presented. The system relies on the generation of multiple segmentation hypotheses to handle ambiguous characters. Despite a large number of rules, the system achieves robust performance on a NIST data set. This work indicates that a multiple-segmentation based system can achieve good performance. Additional research can still be done. Minimizing the number of segmentation candidates is important, as is reducing the amount of time required to generate the segmentations. Furthermore, character-confidence assignment could be greatly improved. Different neural network architectures may improve recognition performance, speed of training, and speed of implementation. We feel that a particularly important area of research in the area of segmentation-based word recognition is the development of character-confidence assignment modules designed to represent the ambiguity of characters well, rather than to maximize the recognition rates.

Acknowledgements. Research at the Environmental Research Institute of Michigan was supported by the United States Postal Service Office of Advanced Technology.

References

ATC (1992) Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC

Boccignone G, Chianese A, Cordella L, Marcelli A (1993) Recovering dynamic information from static handwriting. J Patt Recogn (Special issue on handwriting processing and recognition) 26:409–418

Casey R, Takahashi H (1992) Experience in segmenting and classifying the NIST data base. Proceedings of the 2nd International Workshop on Frontiers in Handwriting Recognition. Chateau de Bonas, France, pp 149–159

Chen M (1992) Off-line handwritten word recognition using hidden Markov models. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 563–579

Chen M, Kundu A (1993) An alternative to variable duration HMM in handwritten word recognition. Proceedings of the 3rd International Workshop of Frontiers in Handwriting Recognition, Buffalo NY, pp 82–92

Favata J, Srihari S (1992) Recognition of general handwritten words using a hypothesis generation and reduction methodology. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 237–253

Gader PD, Whalen MP (1990) Advanced research in handwritten ZIP-code recognition. Technical Report to the United States Postal Service Office of Advanced Technology

Gader PD, Hepp D, Forester B, Peurach T, Mitchell B (1990) Pipelined systems for recognition of handwritten digits in USPS ZIP codes. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 539–559

Gader PD, Forester B, Ganzberger M, Gillies A, Mitchell B, Whalen M, Yocum T (1991) Recognition of handwritten digits using template and model matching. J Patt Recogn 24:421–433

Gader P, Mohamed M, Chiang J-H (1992) Segmentation-based handwritten word recognition. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 215–226

Gillies A (1992) Cursive word recognition using hidden Markov models. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 557–563

Gillies AG, Michell BT (1989) A model-based approach to handwritten digit recognition. Machine Vision Appl 2: 231–343

Govindaraju V (1992) Using temporal information in off-line word recognition. Proceedings of the United Stated Postal Service Advanced Technology Conference, Washington DC, pp 529–545

Govindaraju V (1991) Separating handwritten text from overlapping non-textual contours. Proceedings of the 2nd International Workshop on Frontiers in Handwriting Recognition, Chateau de Bonas, France, pp 111–120

Hepp D (1991) An application of backpropagation to the recognition of handwritten digits using morphologically derived features. Proceedings of the 1991 SPIE/SPSE Symposium on Electronic Imaging Science and Technology, Nonlinear Image Processing II Conference, San Jose, Calif.

Hull JJ, Commike A, HO T-K (1990) Multiple algorithms for handwritten character recognition. In: Suen CY (ed) Proceedings of International Workshop on Frontiers in Handwriting Recognition, pp 117–131, Montreal

Hull J, Ho T, Favata J, Govindaraju V, Srihari S (1992) Combination of segmentation-based and holistic handwritten word recogniton algorithms. Proceedings of the 2nd International Workshop on

Frontiers in Handwriting Recognition, Chateau de Bonas, France, pp 229–240

IWFHR II (1991) Proceedings of the 2nd International Workshop on Frontiers in Handwriting Recognition, Chateau de Bonas, France

IWFHR III (1993) Proceedings of the 3rd International Workshop on Frontiers in Handwriting Recognition, Buffalo NY

Kumura F, Shridhar M, Tsuruoka S, Chen Z (1992) context-directed handwritten word recognition for postal service applications. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 199–214

Kumura F, Shridhar M, Narasimhamurthi N (1993) Lexicon-directed segmentation – recogniton procedure for unconstrained handwritten words. Prodeedings of the 3rd International Workshop on Frontiers in Handwriting Recognition, Buffalo NY, pp 122-132

Lecolinet E, Crettez J (1991) A grapheme-based segmentation technique for cursive script recognition. Proceedings of the 1st International Conference on Document Analysis and Recognition, Saint Malo, France, pp 740–748

Madhvanath S, Govindaraju V (1992) Using holistic features in handwritten word recognition. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 183–199

Mitchell BT, Gillies AG, Whalen MP, Gader PD (1990) A model-based computer vision system for the recognition of handwritten address ZIP codes. Technical Report to United States Postal Service Office of Advanced Technology

Nohl C, Burges C, Ben J (1992) Character-based handwritten address word recognition with lexicon. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 167–180

Park H, Lee S (1993) Off-line recognition of large-set handwritten Hangul with hidden Markov models. Proceedings of the 3rd International Workshop on Frontiers in Handwriting Recognition, Buffalo NY, pp 51–62

Plamondon R (1993) Handwriting Processing and recognition. J Patt Recogn (Special Issue on Handwriting Processing and Recognition) 26:379–381

Plessis B (1992) Isolated handwritten word recognition for contextual analysis reading. Proceedings of the United States Postal Service Advanced Technology Conference, Washington DC, pp 579–593

Rabiner L (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings IEEE 77:257–286

Senior AW, Fallside F (1993) An off-line cursive script recognition system using recurrent error propagation networks. Proceedings of the 3rd Workshop on Frontiers in Handwriting Recognition, Buffalo NY, pp 132–142

Suen C, Nadal C, Mai T, Legault R, Lam L (1990) Recognition of totally unconstrained handwritten numerals based on the concept of multiple experts. In: Suen CY (ed) Proceedings of International Workshop on Frontiers in Handwriting Recognition, pp 131–145

Tappert C, Suen C, Wakahara T (1990) The state-of-the art in on-line handwriting recognition. IEEE Trans Patt Anal Machine Intell 12:787–808

Wilson CL, Garris MD (1990) Handprinted character database. National Institute of Standards Advanced Systems Division Report

Paul Gader is an assistant professor in the department of Electrical and Computer Engineering at the University of Missouri – Columbia. He has been involved in handwriting recognition research since 1989, initially as a research engineer at the Environmental Research Institute of Michigan and later at the University of Missouri. Dr. Gader is also involved in medical image processing and automatic target-recognition research. His research interests include image processing and computer vision, image algebra and mathematical morphology, and neural networks. Dr. Gader received his Ph.D. in applied mathematics from the University of Florida in 1986.

Michael Whalen is a research engineer at Cybernetic Systems in Ann Arbor, Michigan. Mr. Whalen has played major roles in the development of research systems for reading machine printed and handwritten addresses for the United States Postal Service. In particular, he has worked on segmentation, ZIP code location and recognition, model-based address block parsing, word recognition, and decision control modules. He has also worked on a variety of other imaging projects including SAR image formation using parallel processing and detection and removal of quadratic phase errors in SAR imagery. Mr. Whalen received a B.S. in electrical engineering and computer engineering from Wyne State University in 1986 and a M.S. in electrical engineering from the University of Michigan in 1990.

Margaret Ganzberger is a research engineer at Cybernet Systems in Ann Arbor, Michigan. She has played major roles in developing address recognition systems for both machine printed and handwritten applications. While at the Environmental Research Institute of Michigan, she contributed to the development of a state-of-the-art OCR for automated postal address interpretation under United States Postal Service sponsorship. Ms. Ganzberger headed a task to interpret handwritten addresses and assign a 9-digit ZIP code to the mail piece using a national ZIP code database. This research has included developing segmentation and recognition subsystems for handwirtten and machine printed characters, image processing components for segmenting lines and words, and high-level address-interpretation modules. Ms. Ganzberger received a B.S. in computer science from the University of Michigan – Dearborn in 1991.

Daniel Hepp is a research engineer with the Environmental Research Institute of Michigan, Ann Arbor, Michigan. He is currently pursuing his doctorate at the University of Michigan, where he obtained his Bachelor's and Master's degrees in computer engineering. Mr. Hepp has been involved in document-processing research since 1989. His research interests include computer vision, neural networks, machine learning, and statistical pattern recognition.