Department: Address Reading
Subject Area: Electro-Optics and Character Recognition

A SYSTEM TO LOCATE AND RECOGNIZE ZIP CODES IN HANDWRITTEN ADDRESSES

Sargur N. Srihari,
Edward Cohen,
Johathan J. Hull, and
Leonard Kuan
Department of Computer Science
SUNY at Buffalo
226 Bell Hall
Buffalo, New York 1426O
Telephone (716)-636-3191
Telefax (716)-636-3464

Abstract

A system to automatically locate and recognize ZIP Codes in handwritten address is described. Given a grey-scale image of a handwritten address block, the system preprocesses the image by thresholding, border removal and underline removal. One or more candidate words for the ZIP Code are isolated. Each candidate is divided into 5 or 9 segments and recognition is attempted on each segment. Digit recognition is accomplished by means of an arbitration procedure that takes as input the decisions of three different classifiers: template matching using stored prototypes, a mixed approach that uses statistical and structural analysis of digit boundary, and a rule-based approach to analyze digit strokes. The result of ZIP Code recognition is verified using a postal directory. Performance of the system, still under refinement, is promising.

1. Introduction

We consider the problem of automatically determining the destination ZIP Code from handwritten postal addresses. The ZIP Code is a five- or nine-digit number which is the principal information used in United States post offices to sort mail. About 15 percent of First-Class mail handled by the United States Postal Service (USPS) is handwritten (either hand-printed or cursive), which translates into nearly ten billion mail pieces annually [6]. Present postal optical character recognition systems were designed to read machine printed mail. Only a small percentage of handwritten addresses are recognized thus leaving a large volume of mail that must be processed in costly semiautomatic or manual processes.

There are various characteristics that make the Handwritten ZIP Code Recognition (HZR) problem challenging. Some of these characteristics are: addresses where the ZIP Code is not in the last line, which makes the ZIP Code location problem non-trivial, underlines in the address which make ZIP Code location and digit segmentation difficult, and poorly formed addresses without a ZIP Code that may be incorrectly identified as containing a ZIP Code.

A complete system for unconstrained HZR needs many components. At the most basic image processing level, an input image must be thresholded and noise (e.g., irrelevant lines) must be removed. The text line locations in the address must then be determined, separated into words, and candidates for the ZIP Code, city name, and state name must be determined. The city and state names may have to be read either to verify the ZIP Code or because the ZIP Code is absent.

Section 2 gives the present control structure for a system under development. Section 3 describes the preprocessing steps of thresholding, border removal, and horizontal line removal. ZIP Code location is described in Section 4. Section 5 discusses ZIP Code recognition; it includes descriptions of three digit recognition algorithms and an arbitration procedure. Section 6 deals with performance evaluation, error normalization procedure, and the experimental results. Section 7 contains plans for further improving the system.

2. Control Structure

The current version of the system contains the following operational units:

1. Border removal;
2. Thresholding;
3. Horizontal line removal;
4. Text line segmentation;
5. Locating ZIP Code candidates;
6. ZIP Code segmentation;
7. Digit recognition;
8. Comparing recognized ZIP Code with legal ZIP Codes.

The flow of control in the current HZR system is shown in Figure 1. The system begins when a grey-level address block image is input. The image is thresholded using an adaptive thresholding technique. Underlining is removed from the image. Then, the image text is segmented into text lines. This process classifies each connected component as a member of one or more lines and thus establishes the relative locations of features in the image.

Although ZIP Code candidates can occur in any of the bottom six lines, roughly 90% of the ZIP Codes can be found in line 1, and another 5% of the ZIP Codes fall into line 2 [2]. Furthermore, in lines 3 and above, non-ZIP Code digits (such as a box number) that can be mistaken for a ZIP Code often occur. (Note: the lines are numbered from bottom to top.)

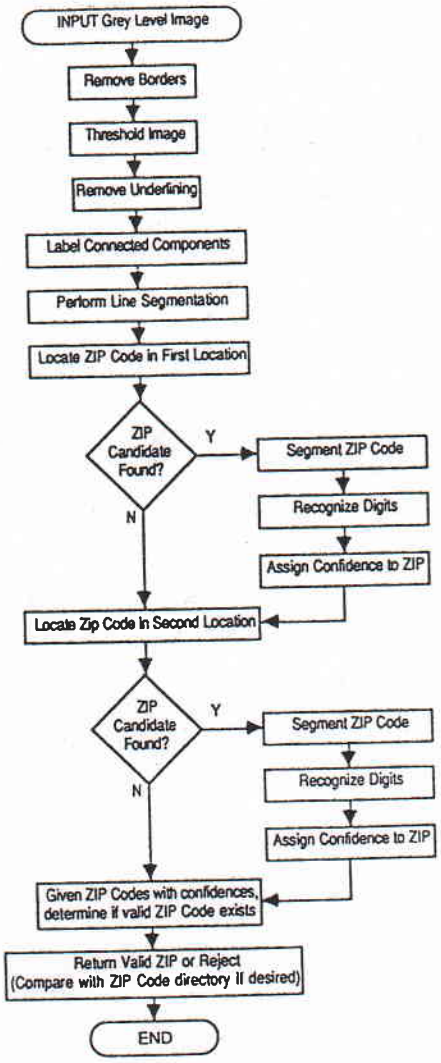


FIGURE 1. Flow of control in ZIP Code location and recognition.

To avoid selecting invalid ZIP Code candidates and still select most of the valid ZIP Code candidates, we currently consider only the bottom two lines.

Therefore, the system uses the ZIP Code location program to search for two ZIP Code candidates, one each from lines 1 and 2. Each candidate is assigned a confidence from -1 to $+1$, where -1 represents a very low confidence and $+1$ represents a very high confidence that the candidate is a ZIP Code. First, the control structure invokes the ZIP Code location program to locate the first ZIP Code candidate (from line 1). Either an image of the candidate or a response that no candidate was found is returned. If no first candidate was found, the first candidate is assigned a confidence value of "-1" and the control structure begins to process the ZIP Code from the second location. If a candidate is found in the first location, the candidate is analyzed to determine the number of components it contains. If the candidate contains less than 3 connected components, the candidate is assigned a confidence value of "-1" and the control structure begins to process the ZIP Code from the second location. If the candidate in the first location contains 3 or more connected components, the candidate is processed to determine the identification and confidence of its digits.

To process the candidate, the ZIP Code segmentation program divides it into either 5 or 9 digits. Then, the digit recognition programs are applied to determine the identity of each segmented digit. If all digits are identified with sufficient confidence, the ZIP Code candidate is assigned a confidence value based on the confidence of the individual digits. Furthermore, if the candidate has 9-digits, the confidence is increased. This is because any candidate with 9 recognized digits is very likely the corrrect ZIP Code. If any of the digits is not recognized, the ZIP Code candidate is assigned a confidence value of "-1". Also, if the candidate is not in a directory of valid ZIP Codes, it is assigned a confidence value of "-1".

Whether or not a ZIP Code was found in the first location, the second location is processed to find a ZIP Code. The processing of the second candidate is identical to the processing of the first candidate, except that no additional searches are made for other candidates.

Next, a check is made to see which (if either) of the candidates should be determined to be the ZIP Code. A candidate is considered valid if its confidence is above a threshold (currently set at 0.0). This simply means that all of the ZIP Code digits were recognized and the digits comprise a valid ZIP Code. If neither candidate is valid, the control structure returns a rejection message. If only the first candidate is valid, that candidate's ZIP Code value is returned as the solution. If only the second candidate is valid, that candidate's ZIP Code value is returned as the solution if its confidence is above 0.8. This higher confidence value is required because the candidates chosen from the second location have a higher likelihood of error.

If both candidates have valid 5-digit ZIP Codes, the control structure returns a rejection message. Both candidates are rejected to avoid confusion between a ZIP Code and some non-ZIP Code identification number (such as a box number). If one or more of the candidates is a valid 9-digit ZIP Code, the ZIP Code with the highest confidence is chosen.

3. Preprocessing

The HZR system receives grey-scale digital images as input. These images are digitized at 300 pixels per inch and are rectangular blocks that contain the entire address. These images must be thresholded and normalized to remove as much of the writer-dependent characteristics as possible. We have included routines for border removal, thresholding, and horizontal line removal in our preprocessing algorithms.

3.1. Border Removal

An artifact of the digitization process is that some images in the database have a black border in parts of their edges. This was caused by a black background on the digitizing table. If an address was written on a mailpiece so that it came very close to the edge, the black background might have come into the field of view. When digitized, the borders have very low grey levels. If not removed, the borders can interfere with the thresholding and connected component analysis.

The borders must be detected before they are removed. This

is done by scanning the edges for large, very dark areas. Pixels on the edge between the black background and the envelope are used to fit a line to the edge. A best fit line is used to compensate for local variations along the edge. After the fit, pixels between the line and the nearest edge are set to white, effectively removing the border.

3.2. Thresholding

The grey level images that are received by the HZR system contain only an address block. This is most often composed of the text and a plain background. The frequent lack of variation in the background and the relatively high degree of contrast between foreground and background simplifies the thresholding problem. After investigation of several alternative methods, including experimental implementations, a technique due to Otsu was adopted that performs quite well on handwritten address images [3].

This algorithm determines a global threshold which divides the grey level image into two classes (black and white pixels). The global threshold is chosen as the value that maximizes the between-class variance of the grey level image. Experimental results have demonstrated that this technique performs well for the HZR application.

3.3. Horizontal Line Removal

Frequently, the words in an address are written on horizontal machine-printed guidelines that are provided to assist the writer. Often the handwritten text of the address intersects the horizontal lines. The text and the lines are then fused during thresholding and result in an image that is not conducive to connected component analysis to locate the ZIP Code, and so on.

A technique for horizontal line removal has been developed to solve this problem. This algorithm makes two passes in scan line order through a thresholded image. In the first pass, the width of a pen stroke is estimated. In the second pass, runs are located that are longer than the stroke width plus a threshold. These runs are hypothesized to be horizontal lines and they are removed.

4. Zip Code Location

Before any recognition can be performed, the digit strings must be located in the image. A line segmentation algorithm separates the image into lines of text.

4.1. Line Segmentation

This bottom-up method is called *shading*. It takes a thresholded image as input, and divides it into vertical strips, each 100 pixels wide. In each strip, a horizontal profile (containing one position for each pixel row) is made. Each position in the profile (in each strip) is set to 1 if any image components pass though that strip's row. Otherwise, the position in the profile is set to 0. After a strip's profile is determined, the number of blocks in the strip is determined. A block is a group of adjacent profile positions which have a value of 1. These blocks can be considered portions of the strip which contain image components. All block positions are recorded, unless they are too small (vertically), in which case they are considered noise and are ignored.

After all the blocks from all the strips are created, the blocks are connected to form lines (of address block text). Preliminary testing has shown that blocks from adjacent strips which overlap vertically are usually from the same address line. So, blocks that are horizontally adjacent and vertically overlapping are connected. Using a set of heuristics, the blocks are connected, forming a topological graph, where blocks are represented as nodes and block connections are represented as node connections. However, sorting out the topological map into address block lines is not a straightforward task. Additional adjustments are needed in several situations.

Determining how to adjust the topological map to closely correspond to the address block lines is an area of ongoing research. Currently, we are using heuristics to separate (or join) blocks in each vertical strip.

4.2. ZIP Code Location

A ZIP Code candidate is determined by using information from the line segmentation program, digit recognition programs, and ZIP Code location characteristics. The program returns either a ZIP Code image or a message stating that no ZIP Code was found.

The ZIP Code location program first reads in the line segmentation results and examines only those connected components determined to be in the chosen line (the chosen line is the line in which the system chooses to look for a ZIP Code).

The ZIP Code location program then searches from right to left looking for a valid ZIP Code candidate. Each connected component is identified (with some uncertainty) as either a digit, a dash, or a comma. This is done by using size thresholds and by applying the digit recognition technique to each component. The program then uses a set of heuristics to select a set of connected components that may be interpreted as a ZIP Code. In straight-forward cases, the first five components found will be digits and a gap will be found to the left of the five components. However, digits are often touching and cannot be identified. In addition, it is not always easy to distinguish gaps between digits and gaps between words, since their widths can vary considerably.

Using heuristics, the program gathers up to five connected components to be a ZIP Code. If a dash is encountered, the system will check if a reasonable number of components are to the right of the dash. If not, the program will return a "no ZIP Code found" message. If a reasonable number of components are to the right of the dash, the program will assume that it is searching for a 9-digit ZIP Code and look for another 5 digits to the left of the dash.

The program continues gathering digits, until it has found the correct number (5 or 9) of digits. Then, the program will search to the left of the last component for another digit. This final step is to make sure the ZIP Code location program has not chosen the last 5 (or 9) digits of an identification number (such as a box number). The ZIP Code location program has been designed to return the largest set of recognized digits in a ZIP Code location. If the ZIP Code location program

returns a ZIP Code with too many digits, it is hoped that during the ZIP Code segmentation or digit recognition the system will determine that the current candidate has an invalid number of digits and the candidate will be rejected.

Another reason for looking to the left of the last component for another digit is that 9-digit ZIP Codes do not always contain an easily distinguished dash. By searching for the extra digit, the system will often find the entire 9-digit ZIP Code even when the dash is not identified.

One drawback to this approach is that characters may be recognized as digits and included in the ZIP Code. For instance the "O" in "CO" (abbreviation for Colorado) may be included in the ZIP Code. However, in most cases where extra characters are included as ZIP Code digits, the ZIP Code will be rejected since the ZIP Code contains an invalid number of digits. Therefore, our ZIP Code location system will tend to have more rejections and fewer errors, which is necessary to keep the error rate within the desired limits.

Further improvements to ZIP Code location can be achieved by using better digit recognition algorithms. Also, the heuristics can be made more sophisticated and more robust. For instance, it may be useful to tentatively identify the state name location so that none of the state name characters are included in the ZIP Code candidate.

5. ZIP Code Recognition

Our approach for ZIP Code recognition is based on the traditional segment and classify methodology. This approach is usually most successful for machine-printed text where the boundaries for digits are either well-determined or can be guessed with a reasonably high degree of accuracy. However, this is not nearly as true in unconstrained handwritten addresses where the text can be cursive or printed and where the letters or digits can be touching.

5.1. *Preprocessing*

Several preprocessing steps are applied to a handwritten ZIP Code before its digits are recognized. The objective of these steps is to reduce the variability in ZIP Codes and to give the digit recognition algorithm images that are as free from writer-dependent effects as possible.

Segmentation

Given an image that contains a full string of ZIP Code digits, the algorithm segments the image into regions that each contain an isolated numerical. The resulting images are passed to the isolated numerical recognition subsystem. Because a legal ZIP Code consists of either five or nine digits, this algorithm applies this knowledge as one of its basic assumptions for the invocation of splitting or merging operations. The technique consists of the following major phases: connected component analysis, estimation of the number of digits in the input image, and grouping and dissection.

First, the physical attributes (X/Y coordinates, height, width, area size, etc.) of each connected component in the ZIP Code image are recorded. Other attributes based on the whole string of digits in the image [its upper and lower contour profiles, the topological relationship between blobs (e.g., left of, right of, etc.), the positions of ZIP Code string's upper line, central line, and lower line] are also computed and recorded. In addition to the creation of data structures, several operations are performed on the image. These include sorting components according to their heights, selecting the digit candidates from all the components, detecting and removing of underlining, removing noise, and removing long ligatures from digit candidates.

Based on the attributes described, the program makes an estimation of the number of digits in the image. In addition, spacing between connected components is used to group components into digit clusters, where each cluster is determined to contain one or more digits.

Then, the extraction process and the necessary grouping or dissecting operations are performed to generate isolated digit images from the clusters. To extract nontouching digits, separating line segments are drawn to enclose zones that contain isolated digits. Grouping operations merge separated blobs and their containing areas into a complete digit zone. Grouping is often required when a digit is fragmented after it is thresholded or when more than one blob constitutes a complete digit (digits "4" and "5" are the most frequently seen cases). The grouping function merges components in either neighboring clusters or a single cluster. Dissecting operations split connected or touching digits and try to maintain the original digit shapes for the purpose of recognition. Once the number of digits touching each other is determined, the splitting routine will select the proper positions and dissect the blob into that number of digits. The dissecting positions are determined by considering several factors: the slant angle of the whole ZIP Code string, the average width of digits in the image, the peaks and valleys detected from the lower and upper contour profile respectively, and the number of strokes near the place where touching is predicted. The dissector uses different techniques to separate digits depending on the factors mentioned above. For example, if both a peak and a valley are found near the predicted position, the dissector will first try to draw separating line segments in the gap between two digits by directly cutting through the touching portion from the peak to the valley. If only a peak or valley is found, then a similar operation called "hit and deflect" is performed. This technique is always guided by the slant angle. If no peak or valley is found, a "single stroke" area will be chosen to cut through.

After all the proper actions have been taken, each enclosed single digit is extracted and output along with the size of its minimum bounding rectangle.

Size Normalization

All the character recognition algorithms in the HZR system use the same size normalization algorithm. Size normalization is performed in two steps. The first step normalizes for height and the second for width. This is done so that inherently thin digits like "1" are not distorted with respect to thicker digits such as "8". An input digit of height h_i and width w_i pixels is mapped to height h and width w. An intermediate step is used that produces an image of size height h' and width w' where

$$w' = p \cdot w_i, \text{ and}$$
$$p = \frac{h}{h_i}.$$

The variable p holds the proportion used to scale the height. Height normalization is performed by scanning the original image and for every pixel with coordinates (x, y), the pixel at $(x \cdot p, y \cdot p)$

in the normalized image is set to black if and only if the pixel at (x, y) is black.

Width normalization is given the output of the height normalization and considers two cases. If $w' < w$, the height-normalized image is centered in an area of size $h \times w$. If $w' > w$, a shear transformation is performed. This is done by scanning the height-normalized image. For every pixel at (x, y), the pixel at $(x/w') \cdot w$, $y)$ is set to black if and only if the pixel at (x, y) is black.

5.2. Digit Recognition

Several previous algorithms for handwritten digit recognition have used a hierarchical approach in which more than one basic algorithm is applied either in sequence or in parallel. This is done to achieve (i) higher speed by applying the most efficient algorithms first and accepting their results only if confidence is high enough and (ii) greater accuracy by combining the results of more than one algorithm [1, 5]. Instead, we have followed a parallel approach in order to achieve a high recognition rate and a low error rate. Our method includes three algorithms: (i) a template matching algorithm to make a decision based on the overall holistic characteristics of the image, (ii) a mixed statistical and structural classifier used on features extracted from the contour of the character, and (iii) a structural classifier used on information about the size and placement of strokes in the image. These three algorithms were chosen because they utilize different types of information in the image and thus have a better chance of compensating for each other's weaknesses. Each of the algorithms is applied to the same digit image and their results are combined with a decision tree to achieve a classification decision.

The subsequent sections describe each digit recognition algorithm.

Template Matching

The template matching approach for digit recognition uses a large library of size-normalized templates as its training data. An input image is recognized by matching it to the prototypes in the library. The matching criterion is the sum of the exclusive-or, that is, the number of pixels that are different between the size-normalized input digit and the size-normalized prototypes. The smaller this value, the better the match.

A digit is classified by this method by determining the closest prototype in each class. The difference between the two minimum distances is computed and used as a measure of the goodness of the match.

Mixed Approach: Statistical and Structural Analysis of Boundary Approximation

This method first approximates the contour of a character with a piecewise linear fit. It then computes features from that fit and places them in a feature vector. This vector is matched to a stored set of labeled prototypes and two classes that match best are determined. The matching is done with both structural feature tests and a weighted Euclidean distance. The structural tests are performed on features such as the size and location of holes that have proven useful in partitioning a training set into known subsets. This approach is based on a method that has previously demonstrated high recognition rates [4].

The features that are extracted from these descriptions are shown in Table I. The classifier is a mixture of a structural, decision tree classifier that determines in which subset of classes a given vector belongs. A modified k-nearest neighbor classifier is then used on the feature vectors in those classes.

TABLE I. The features used for digit recognition by the mixed approach

feature	description
0	number of components in image (1 or 2).
1	number of holes in image.
2	hole description variable.
3	number of concave arcs on external boundary.
4	number of concave arcs on left side only.
5	distance from the top of the character to
6	the first, second and third largest concave
7	arcs respectively.
8	length of the outside polygonal boundary.
9	length of the outside polygonal boundary from the top to the first concave vertex down the left side.
10	top y-coordinate of the uppermost or only hole.
11	bottom y-coordinate of the uppermost or only hole.
12	top y-coordinate of the lowest hole, used only if feature #1 >= 2.
13	bottom y-coordinate of the lowest hole, used only if feature #1 >= 2.
14	horizontal thickness of component.
15,16,17	opening,perimeter and direction,respectively, of the concave arc with the largest cavity.
18,19,20	opening,perimeter and direction,respectively, of the concave arc with the second largest cavity.
21,22,23	opening,perimeter and direction,respectively, of the concave arc with the third largest cavity.
24	position of the arc with the largest cavity. (1-rightside,0-leftside)
25	number of significant concave arcs on the left side.
26	number of significant concave arcs on the right side. (significant : opening/perimeter < 0.9)

Stroke Analysis Approach

The structural method of numeral recognition decomposes a numeral into strokes that are used as features for classification. The feature extractor computes contour profiles, stroke run-length, holes and their estimated stroke width. Then the numerical is completely described as a group of nearly horizontal strokes (H-strokes) and nearly vertical strokes (V-strokes).

The numeral classifier uses the detected strokes, holes, and character profiles as features. The classifier consists of many rules, each specifies a certain type of numeral. These rules describe the structure and the topological criteria of various types of numerals. For example, a rule for a type of digit "0" is the following:

> There are two H-strokes, two V-strokes, and one hole; two V-strokes are connected to the upper H-stroke and the lower H-stroke, one at left and one at right; and the hole is positioned in the center of those strokes.

These rules were constructed by observing the results of the stroke decomposition process as applied to 1754 numeral samples of training data. We summarized the results as different prototypes, then built the corresponding rules. These rules enabled the classifier to recognize 88 percent of the 1754 digits.

Arbitration Procedure

The results of applying the three digit recognition algorithms discussed above are combined using the decision tree shown in Figure 2. The first branch of the decision tree is: if the best decision of the three techniques agrees, then output that decision. The confidence associated with that decision is 0.996, which reflects the proportion of correct decisions with a training set of 8142 digits. Subsequent tests check other combinations of decisions. Altogether there are nine such tests in the tree.

6. Performance Evaluation

An example of the operation of the complete system is shown in Figure 3. Some of the intermediate processing steps of an image are shown. A grey level image is shown in (a). The thresholded image is shown in (b). Next, underlining is removed (c) and two ZIP Code candidates are extracted from each of the bottom two lines (d and e). In this example, the word "ZIP CODE" is placed in the bottom most line by the line segmentation program. The ZIP Code segmentation segments each of the ZIP Code candidates (f and g). The segmented digits are passed to digit recognition algorithms and the results are seen in (h) and (i). For the recognition results, the first value is the value of the recognized "digit". The second value is the confidence. The third value is the branch number from the decision tree.

In our current system, all recognition results from branches 8 and 9 are rejected, so the ZIP Code in (h) is rejected. Digits from branches 8 and 9 are rejected to keep the error rate within reasonable limits (these two branches have the highest error rate). Furthermore, the ZIP Code in h "08000" is not a valid US ZIP Code and would be rejected anyway. So, our system correctly selects the second ZIP Code candidate as the proper ZIP Code and returns "10036" as the result.

In the design phase, most algorithms for the HZR system were designed using a "training" data set of 1000 handwritten address images. The performance of the system was tested using 500 other images. All the images were gathered from live mail at the USPS sectional center facility in Buffalo, New York. The results shown in Table II describe the system performance as of March 1, 1989 (see Section 7 for further information). As we have described, the HZR system research is ongoing and system performance is being improved on a (nearly) day-to-day basis.

For each test image, the HZR system returned one of two responses: a ZIP Code (5- or 9-digit) or a rejection message. A rejection message indicated that the system did not find sufficient information to determine a ZIP Code value.

The contents of each image had been manually examined and entered into an index. The desired response for each image was determined by examining the index entries for the proper images. A correct response would be the proper ZIP Code (if present) or a rejection (if no ZIP Code was present). The per-

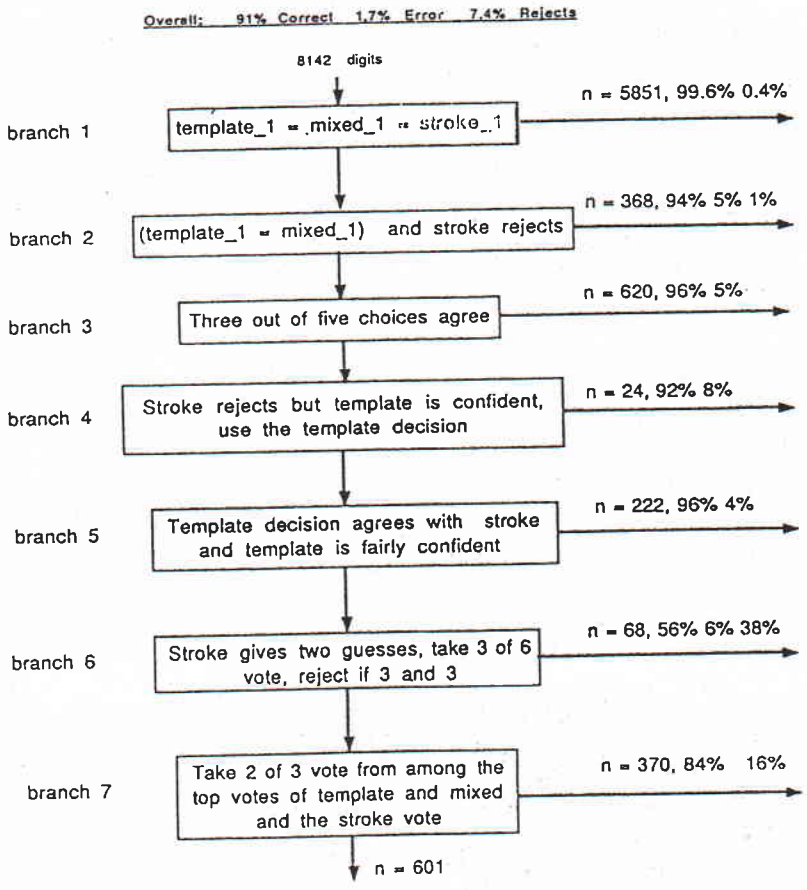
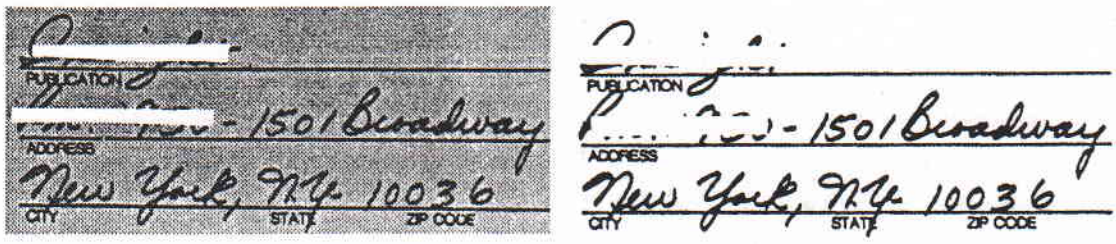
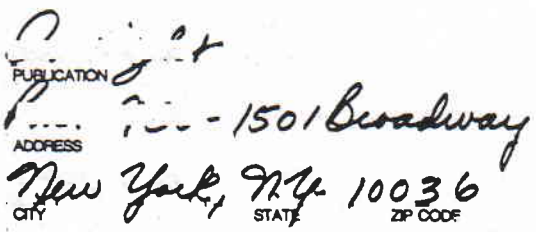


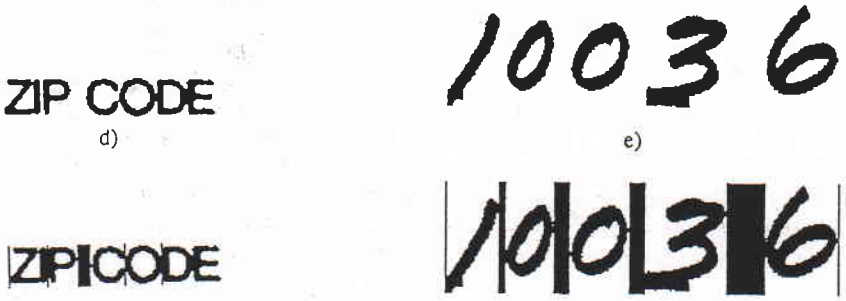
FIGURE 2. Decision tree to combine results of three digit recognition algorithms.



a) b)



c)



d) e)

f) g)

0 0.996 1
8 0.938 2
0 0.938 2
0 0.996 1
0 0.585 9

h)

1 0.996 1
0 0.996 1
0 0.996 1
3 0.996 1
6 0.996 1

i)

FIGURE 3. Steps in processing a handwritten address.

TABLE II. System performance

Group	Code	Condition	Count	%
1	1.1	ZIP correctly read	215	43.0
(Correct)	1.2	No ZIP Code present	44	8.8
		Subtotal	259	51.8
2	2	Rejection Failure	229	45.8
(Reject)				
3	3.1	Location failure (No ZIP)	1	0.2
(Error)	3.2	Location or recognition failure (w. ZIP)	11	2.0
		Subtotal	12	2.4
		TOTAL	500	100.0

1.1 ZIP Code present and correctly read.
1.2 No ZIP Code present and image rejected.
2 ZIP Code present, but no ZIP Code candidates found.
3.1 No ZIP Code present, but something else was identified as a ZIP Code and recognized.
3.2 ZIP Code present, but other candidates were found and/or their digits were recognized.

formance of the recognition system was measured by comparing the system's response for each image to the appropriate index entry.

For each image, one of three outcomes was possible. A CORRECT response meant that the recognition system response matched the index entry ZIP Code or that the index entry indicated no ZIP Code was present and the recognition system returned a rejection message. A REJECT response meant that the recognition system returned a rejection message and the index entry indicated that a ZIP Code exists for that image. An ERROR response occurs when the system returns a ZIP Code that is different from the value listed in the index entry (including cases where the index entry indicates no ZIP Code exists). Using these outcomes, we were able to determine the overall system response.

Further analysis was done to determine the exact causes of failure (both rejects and errors). By analyzing the failure rate of the HZR system we found that the primary causes of failure and the percentages they account for are ZIP Code location (48%), digit recognition (18%), preprocessing [primarily underline removal (14%)], and line segmentation failures (5%).

7. Discussion

The system described in this paper is able to correctly recognize 51.8% of the ZIP Codes with an error rate of 2.4%. Although the basic algorithms are the same, a number of improvements have been made since this paper was originally submitted. Our *current improved* system correctly recognizes 72.2% of the ZIP Codes with an error rate of 2.4%. Although many refinements contributed to the higher success rate, the major advance was in ZIP Code location. The improved location system creates a feature map of the address block. The feature map includes likely location of the city name, state name, and ZIP Code. The feature map allows us to examine a variety of information which is helpful in identifying the ZIP Code location and the location of other words in the address (i.e., if a line contains a city name followed by the state name, the line below is likely to contain the ZIP Code). The next two sections describe our future directions.

ZIP Code Location

The ZIP Code location algorithm described in this paper identifies digits, dashes, and commas and then tries to locate the ZIP Code based on the spacing and size of these components. This approach has difficulties since not all ZIP Codes will look like digits (prior to segmentation) and some non-ZIP Code digits (such as box numbers) will look identical to ZIP Codes. Global information about the address block is needed to isolate ZIP Codes.

For instance, knowing the location of the street address can help direct our search for the ZIP Code, since the ZIP Code will usually be below the street address. If we knew the approximate location of the state name, this may help prevent us from including characters from the state name in the ZIP Code candidate.

Our improved system develops a map of the address block using the location of features such as the state and city names. The map is used as an organizational device to maintain information about the image and helps direct the search for a particular address block.

Another advantage of creating an address map is that the knowledge is available to the control structure. The heuristics used by the ZIP Code location program were originally buried inside the program. Moving the address block information out of the individual programs and into a globally accessible database gives us greater flexibility in how the knowledge is used. In addition, the more accessible the information is, the more likely that the proper information will be available when it is needed.

One example of useful information that may become available is contextual information. Identifying the state name (or the first letters of the state name) may provide additional clues as to where the ZIP Code digits start. So, work on ZIP Code location is focused on the development of a top-down approach that uses a wide variety of information.

Digit Recognition

There are many areas that can be pursued to improve the accuracy of digit recognition. Because our system is based on several algorithms applied in parallel, our first effort will be to add another recognition algorithm. We have had a pure rule-based method that analyzes the contour of a digit under development for several months. Recently, we have begun to test this technique. Preliminary results are similar to those we achieved with the other methods. Therefore, it can be expected that a similar effect can be achieved if we add this classifier to our parallel scheme, namely, an increased rate of correct recognition and a lower error rate.

Another way to improve performance is to read ZIP Codes without segmenting them. We have developed a preliminary plan of how to do this. Our method will be based on the template matching digit recognizer. A ZIP Code will first be normalized for height. The prototypes will be matched continuously across the image. A ZIP decision will be based on a combination of these results.

We also plan to investigate the application of digit recognition during segmentation. As the segmentation program operates, it would be useful to know if a certain cut was performed whether the digits would be recognized. If they are, then it is a good bet that this is a proper cut to make. The ability of such a scheme to improve digit recognition will be determined.

A thorough method of contextual post-processing (CPP) also should be developed. CPP uses external information to compensate for and detect errors or correct rejected digits. Right now we are using a dictionary of allowable ZIP Codes to check the validity of a ZIP Code. We would like to develop a more complete method that uses the allowable digits provided by the dictionary to perform digit recognition again. The difference would be that this time the classes could be constrained ahead of time. This could potentially be very helpful information in that it could eliminate the consideration of illegal classes. This would be useful in the template matching and mixed approaches that use sets of prototypes where such top-down knowledge could be useful.

Another way to improve digit recognition is to use information about the identify of city and state names. We have several techniques under development for character recognition that could

prove useful. They will be further developed and their utility will be determined.

Acknowledgment

This work was supported by the United States Postal Service Office of Advanced Technology under Task Order 104230-86M-3990. We are indebted to Dr. Timothy Barnum of USPS OAT and Dr. John Tan of Arthur D. Little, Inc. who contributed in several ways to the development of the system.

References

1. **B. DUERR, W. HAETTICH, H. TROPF AND G. WINKLER. "A combination of statistical and syntactical pattern recognition applied to classification of unconstrained handwritten numerals",** ***Pattern Recognition 12,*** **3 (1980), 189-199.**

2. **J. J. HULL, D. S. LEE AND S. N. SRIHARI. "Characteristics of handwritten mail addresses: A statistical study for developing an automatic ZIP Code recognition system", Technical Report 88-06, Department of Computer Science, State University of New York at Buffalo, March, 1988.**

3. **B. OTSU. "A threshold selection method from gray-level histograms",** ***IEEE Transactions on Systems, Man, and Cybernetics SMC-9,*** **1 (January, 1979), 63-66.**

4. **T. PAVLIDIS AND F. ALI. "A hierarchical shape analyzer",** ***IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1,*** **1 (January, 1979), 2-9.**

5. **A. PEREZ AND C. Y. SUEN. "Computer recognition of totally unconstrained handwritten ZIP Codes",** ***International Journal of Pattern Recognition and Artificial Intelligence 1,*** **1 (March, 1987), 1-15.**

6. **USPS, Annual Report of the Postmaster General, 1983.**

Edward Cohen

Edward Cohen is a Graduate Research Assistant in the Department of Computer Science at the State University of New York at Buffalo. He received a B.S. degree in Electrical Engineering from Cornel University in 1982 and an M.S. in Computer Science from the State University of New York at Buffalo in 1988. He has also worked at RCA in Pennsylvania. He is currently working on his Ph.D., with a dissertation on the topic of a "Computational theory for determining ZIP Codes from handwritten address".

Leonard Chin-Chau Kuan

Leonard Chin-Chau Kuan is a Research Associate in the Department of Computer Science at the State University of New York at Buffalo. He received a B.S. degree in Electrical Engineering from National Tsing-Hua University (Taiwan) in 1982 and an M.S. in Computer Science from the State University of New York at Buffalo in 1988. He is the principal developer of several runlength based algorithms for processing images of handwriting.

Jonathan J. Hull

Jonathan J. Hull is a Research Assistant Professor in the Department of Computer Science at the State University of New York at Buffalo. He received a B.A. in Computer Science and Statistics in 1980, an M.S. in Computer Science in 1983 and a Ph.D. in Computer Science in 1988, all from the State University of New York at Buffalo. He has been working on USPS related projects since 1984 and at present has principal responsibility for the "Handwritten ZIP Code Recognition" and "Contextual Analysis of Machine Printed Addresses" tasks.

Sargur N. Srihari

Sargur N. Srihari is a Professor in the Department of Computer Science at the State University of New York at Buffalo. He received a B.S. in Physics and Mathematics from Bangalore University (India) in 1967, a B.E. in Electrical Communication Engineering from the Indian Institute of Science in 1970 and an M.S. and Ph.D. in Computer and Information Science from the Ohio State University (Columbus) in 1971 and 1976 respectively. He has been working on pattern recognition and artificial intelligence problems for fifteen years and his present research interests are mainly in Document Image Recognition and Understanding.