UNDERSTANDING HANDWRITTEN TEXT IN A STRUCTURED ENVIRONMENT: DETERMINING ZIP CODES FROM ADDRESSES

EDWARD COHEN, JONATHAN J. HULL
and SARGUR N. SRIHARI
Center of Excellence for Document Analysis and Recognition
226 Bell Hall
State University of New York at Buffalo
Buffalo, NY 14260, USA

Understanding a block of handwritten text means mapping it into a semantic representation. We describe an approach to reading a block of handwritten text when there are certain loose constraints placed on the spatial layout and syntax of the text. Early recognition of primitives guides the location of syntactic components. A system to read handwritten postal addresses is described as an instance. The semantic representation in this case is a digit string (ZIP Code). Methods for segmenting a string of digits into components and for recognizing digits using a multiplicity of recognizers are given.

Keywords: Character recognition; Handwriting recognition; Classifier systems.

1. INTRODUCTION

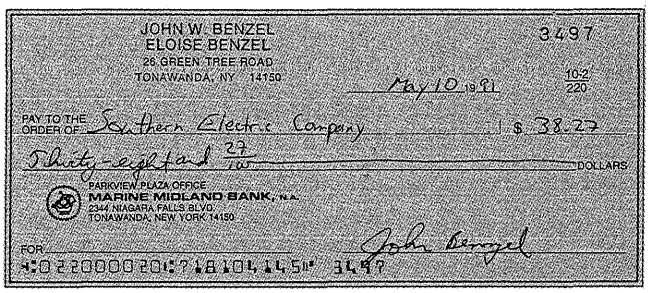
Understanding a text image involves both recognition and comprehension. The two processes can be disjoint for machine-printed text and highly constrained handwriting. In such cases, the recognition process does not need the comprehension process to help disambiguate uncertainty due to the small degree of variability in the input patterns. Constraints that greatly reduce handwriting variability can take various forms: preprinted boxes to limit the size and location of characters or digits, guidelines to specify location of words, suggestions for forming letters, suggestions for joining letters and ligatures in cursively written words, rigidly fixed syntax, no spelling errors, etc. Strengthening constraints makes the recognition process easier and reduces its dependence on comprehension.

On the other hand, comprehension must be used when the writing is unconstrained (or very loosely constrained). When handwritten text is unconstrained, no restrictions are placed on the writing style or implements used. A system that reads unconstrained writing must compensate for many factors that affect text appearance, e.g. variations in writing styles,[1,2] size of text,[3] writing implement used,[4] writing surface,[5] digitization,[6] and thresholding.[7] Several examples of handwritten text are shown in

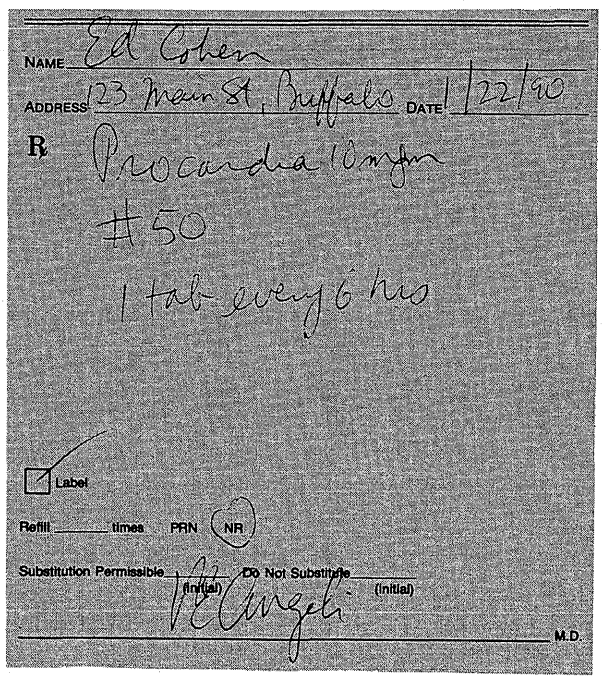
This work was supported by the United States Postal Service Office of Advanced Technology under Task Order 104230-86M-3990.

Figs. 1(a)–1(d). In each case the writing is unconstrained, but varying degrees of restriction are placed on their spatial layout.

By placing some restrictions on spatial layout, the problem of using comprehension to assist in extracting the desired semantic information from the text becomes tractable. Comprehension or 'understanding' is a process that requires knowledge. In the pattern recognition literature, this knowledge is sometimes referred to as context. Context that has a bearing on reading is usually linguistic in nature, e.g. lexicons describe legal character strings as in a dictionary, syntax defines how words and punctuation combine to form phrases, and semantics describes how meaning is associated with phrases. While linguistic information is usually presented as a

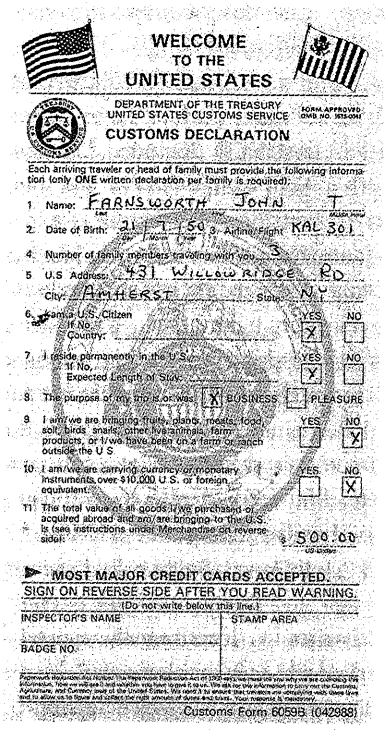


(a)



(b)

Fig. 1. Examples of spatially structured handwritten text. (a) Bank check, (b) drug prescription



WELCOME TO THE UNITED STATES

DEPARTMENT OF THE TREASURY
UNITED STATES CUSTOMS SERVICE

FORM APPROVED OMB NO. 1515-0041

CUSTOMS DECLARATION

Each arriving traveler or head of family must provide the following information (only ONE written declaration per family is required):

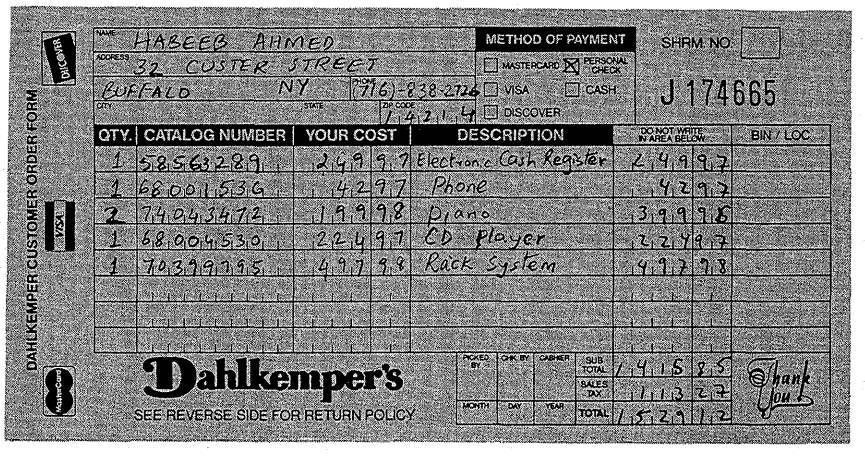
1. Name: FARNSWORTH (Last) JOHN (First) T (Middle Initial)
2. Date of Birth: 21 (Day) 7 (Month) 50 (Year) 3. Airline/Flight: KAL 301
4. Number of family members traveling with you: 3
5. U.S. Address: 431 WILLOW RIDGE RD
City: AMHERST State: NY
6. Are you a U.S. Citizen — YES [X] NO [ ]
If No, Country:
7. I reside permanently in the U.S. — YES [X] NO [ ]
If No, Expected Length of Stay:
8. The purpose of my trip is or was: [X] BUSINESS [ ] PLEASURE
9. I am/we are bringing fruits, plants, meats, food, soil, birds, snails, other live animals, farm products, or I/we have been on a farm or ranch outside the U.S. — YES [ ] NO [X]
10. I am/we are carrying currency or monetary instruments over $10,000 U.S. or foreign equivalent. — YES [ ] NO [X]
11. The total value of all goods I/we purchased or acquired abroad and am/are bringing to the U.S. is (see instructions under Merchandise on reverse side): $ 500.00 (US dollars)

MOST MAJOR CREDIT CARDS ACCEPTED.
SIGN ON REVERSE SIDE AFTER YOU READ WARNING.
(Do not write below this line.)

INSPECTOR'S NAME | STAMP AREA
BADGE NO.

Customs Form 6059B (042888)

(c)



DAHLKEMPER CUSTOMER ORDER FORM

NAME: HABEEB AHMED
ADDRESS: 32 CUSTER STREET
CITY: BUFFALO STATE: NY PHONE: (716)-838-2726 ZIP CODE: 14211-4

METHOD OF PAYMENT: [ ] MASTERCARD [X] PERSONAL CHECK [ ] VISA [ ] CASH [ ] DISCOVER

SHRM NO: J 174665

QTY.	CATALOG NUMBER	YOUR COST	DESCRIPTION	DO NOT WRITE IN AREA BELOW	BIN / LOC
1	58563289	249 97	Electronic Cash Register	249 97	
1	68001536	42 97	Phone	42 97	
2	74043472	199 98	Piano	399 98	
1	68904530	224 97	CD Player	224 97	
1	70399795	49 97	Rack System	49 97	

Dahlkemper's
SEE REVERSE SIDE FOR RETURN POLICY

PACKED BY | CHK BY | CASHIER | MONTH | DAY | YEAR

SUB TOTAL 1415 85
SALES TAX 113 27
TOTAL 1529 12

Thank You

(d)

Fig. 1. Cont'd. Examples of spatially structured handwritten text. (c) Customs declaration, (d) department store order form.

one-dimensional string, handwriting also has a spatial content. Non-linguistic context (such as layout information) can describe two-dimensional positional information between words and symbols.

The role of context in reading unconstrained handwriting is not well understood, or at least not easily formalized. Context plays a large role when the reader cannot rely on the text, e.g. if the text is physically degraded.[8] Unconstrained handwritten text is unreliable because it often has poor script quality and can contain spelling errors and malformed letters. We can explore the role of context experimentally, by attempting to build systems when the domain rather than the handwriting style is constrained. Examples of constrained text domains in which handwriting is often present are: a form, an address on an envelope, a bank check, a credit card slip, a drug prescription. In many of these domains the lexicon, syntax, and semantics are implicitly or explicitly available. Our research explores issues in comprehending unconstrained handwriting in constrained domains by considering the domain of handwritten postal addresses in the United States. We describe a general approach that characterizes how context can be used for reading text in our constrained domains. This approach is tested on a wide variety of handwritten postal addresses.

This paper is organized as follows. Section 2 discusses previous work. Section 3 describes a general approach to 'understanding' unconstrained handwritten text in certain limited domains. Section 4 describes the postal address domain. Section 5 details how the general approach was implemented in the domain of extracting ZIP Code information from handwritten addresses. Section 6 discusses how the system processes two handwritten address examples. Section 7 deals with performance evaluation, error normalization, and experimental results. Section 8 discusses possible refinements to the system. A description of the image database used in developing and testing the system is given in the Appendix.

2. PREVIOUS WORK

Computational reading of handwritten words and digits is an important but difficult task which has a large literature; a tutorial and key papers can be found in Ref. 9. However, most researchers do not use substantial amounts of the available context. Typically, researchers who use context are using only spelling information to compensate for errors in character recognition. While this type of context can offer substantial improvements (over not using it), much more context is often available. Baird and Thompson[10] use a variety of contexts while reading machine-printed chess games, but there are many issues unique to reading handwritten text that need not be addressed in reading machine-printed text. Two research projects that used contextual information (which included more than spelling information) for analyzing handwritten text are described in Refs. 11 and 12. They were both concerned with reading hand-printed Fortran coding sheets and are described below.

In Ref. 12, all the character recognition information is processed bottom-up. For each character in the image, the character recognition algorithm provides one or more choices with confidence values for each choice.[13] Using this character recognition

information and connected component location information, the authors apply contextual analysis to locate and correct recognition errors.

A two-pass approach processes the original recognition results. First, the expression type is identified, e.g. a line of FORTRAN may contain a DO-loop, a READ statement, an arithmetic operation, etc., and an identifier list (names of variables that are likely to be used more than once) is made. To prevent combinatorial explosion, all recognized characters are generically relabeled into digits, characters, and punctuation, e.g. *GOTO 3* becomes *CCCC #*, where *C* is a character and *#* is a number, and expressions are tentatively parsed and identified using these generic labels. The labels make matching words to syntax easier, while increasing ambiguity and the likelihood of multiple matches.

In the second pass, expressions are parsed using specific knowledge about their syntax, requiring separate control mechanisms for each type of FORTRAN expression. The list of identifiers is used to correct recognition errors.

The researchers avoid the combinatorial explosion that is possible with multiple recognition values by making assumptions about the data; e.g. that character recognition is fairly accurate (the correct character recognition result was always included in the list of possible character values) and that the number of characters is known from positional data. The authors state that more elaborate techniques are required for very poor data (such as unconstrained handwritten text).

Bornat, Brady, and Wielinga[11,14] discuss control issues for computer interpretation of handwritten FORTRAN coding sheets and describe techniques for combining character recognition information with high-level FORTRAN information. Their input is a FORTRAN coding sheet image with certain restrictions on size, neatness, and programming correctness.

They use frames to store character recognition information, including several recognition hypotheses for each character. The frames keep track of information obtained, information desired, and whether or not a conclusion is reached. A conclusion is reached if most character feature requirements are satisfied or if syntax or domain knowledge can identify the character.

The authors describe several techniques for combining diverse knowledge sources to improve character recognition. Much of their work involves improving character recognition by re-examining previous recognition results, e.g. if a *T* was not recognized because its strokes did not touch and additional evidence indicated that the character was a *T*, the authors would extend the strokes, trying to form a recognizable *T*. They also use confidence information.

Our research is not geared towards actively trying to re-recognize characters. Instead, efforts will focus on using available character recognition results and context to identify words; e.g. if context rules out all but one choice for a word, then word recognition has succeeded even if most of its characters have not been identified. Both FORTRAN coding sheet research projects constrain the writing style, which reduces the need for context. Our approach is to let the writing style be unconstrained, but to use contextual information from the domain to assist in extracting the desired semantic information from the text.

We describe a system that can use substantial amounts of context. Even though the system has not been fully implemented, we have still demonstrated substantial improvements over previous approaches. We believe the methods described in this paper will work for a variety of domains where handwriting style is unconstrained.

3. GENERAL APPROACH

Experiments with unconstrained handwriting in handwritten address blocks have confirmed that straightforward bottom-up approaches are insufficient for solving the reading problem because low-level features (such as word boundaries and character recognition results) cannot be reliably extracted from the image. In strictly bottom-up systems, the unreliable low-level information propagates upwards and makes high-level information unreliable as well. These bottom-up approaches have low correct performance and high error rates. A high-performance system must compensate for the following difficulties:

(1) unreliable character recognition results;
(2) non-text information that may appear in the image (e.g. underlining);
(3) uneven spacing between text lines and overlapping or touching components from adjacent lines;
(4) word isolation uncertainties (i.e. selection of image components so that the component set contains one complete word with no extraneous components is not always accurate);
(5) syntax variations hinder correct parsing of text lines; some writers use suggested syntax only as a rough guideline (e.g. punctuation is often omitted or confused with image noise);
(6) semantic uncertainties can make the choice of proper meanings difficult (e.g. 14215 may be a ZIP Code, box number, street number, etc.).

Our approach works by first developing a global description of the text and then verifying the hypotheses created from the description. This results in a bottom-up and top-down approach (see Fig. 2). Bottom-up information is used to create a global description and top-down information focuses attention on areas of the image that need further investigation.

Our technique of developing the global description from bottom-up information can be divided into two stages. In the first stage, signals from the image (pixel information) are translated into symbolic information. Examples of this stage are: connected components being recognized as characters, connected components being identified (but not recognized) as text words, and determining that a group of pixels are on a single text line. The translation from pixel information to symbolic information is done using knowledge of how text is written, e.g. knowing the shape of characters, digits, words, and punctuation.

The second stage converts this symbolic information into a conceptual structure of the text. This stage uses lexical information, syntax, and semantics to determine the likely locations of phrases in the text (e.g. the topmost text line in an address usually contains a person's first and last names). For our purposes, the conceptual structure

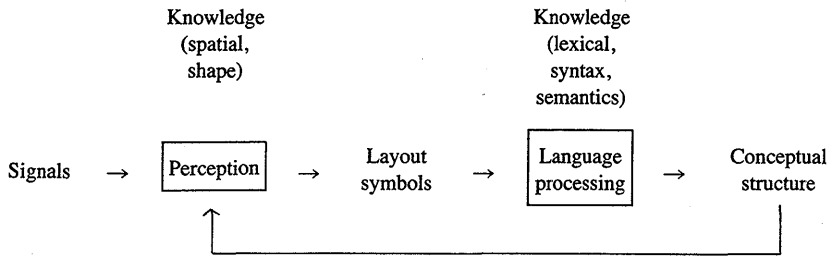


Fig. 2. Conceptual control flow of the reading process.

will be domain-specific. That is, the type of structure used to describe the global information will be tailored to the domain from which the image text is taken. We assume that the image domain is known *a priori*.

The conceptual structure contains all the high-level information that is currently 'known' about the image. Using the global structure, we can predict where words (which may not have been recognized) will be found in the image. Suppose we are examining a credit card slip. The global structure may predict that a certain word group contains a description of an item and that another group of words contains the price of that item. When we have identified the type of words expected and their location, we can use that knowledge to return to the perception stage (where image signals are available) to recognize those words.

This secondary recognition is driven top-down. We make hypotheses of the word types and then use signals from the image to confirm or deny the hypotheses. In some cases, verification may be needed to identify the words, while in other cases it may be sufficient to determine that the words have the proper size and shape characteristics. A more specific implementation of our approach is described below:

(1) perform preprocessing to account for any artifacts unique to the current domain (e.g. underlining, common noise patterns);
(2) use positional information to make a preliminary line segmentation;
(3) classify connected components without using positional information;
(4) use classification and positional information to further refine text line segmentation and component classification;
(5) generate several different hypothetical segmentations for each line, dividing lines into words;
(6) coarsely categorize each word using the available character recognition information and an estimate of the number of characters;
(7) grade the degree of the match between word categories and syntax;
(8) recognize words using character recognition information and syntactic positions for highly rated words from Step 7;
(9) constrain word recognition using lexical information and semantics;
(10) if a reliable semantic evaluation is found, derive the proper semantic interpreta-

tion; otherwise reject the current syntax–category match and return to Step 8 if other highly rated syntax–category matches exist.

This approach divides the task into four manageable stages: preprocessing, line segmentation, word segmentation, and word identification. For each stage, low-level information is gathered and all available context is used to complete the processing. The motivation for these steps is described in the rest of this section.

In its simplest form, a text image contains only two types of pixels, text pixels and background pixels. For this reason, our preprocessing step produces a binary image containing these two types of pixels. Although some information may be lost by discarding gray-scale or color information, our approach isolates a fundamental aspect of reading, i.e. interpreting what is considered to be text. Two additional motivations for using binary images are that some images only come in binary form, and the processing required for binary images is usually much less than for gray-scale or color images.

After preprocessing, we begin line segmentation. Even when an image contains only 100 characters, trying all combinations of connected components to form word and line groupings results in combinatorial explosion. The situation worsens if segmentation and merging of connected components are required to form characters. Since text lines are usually the most prominent text image features, our approach takes advantage of their prominence and segments text lines immediately after preprocessing.

To perform text line segmentation, we first extract the 'easy' information, such as connected component positional information, and perform preliminary line segmentation. However, some text cannot be placed into lines based only on positional information, e.g. commas are often closest to the line beneath their actual line. For these cases, we classify components and combine the classification and positional constraints to refine line segmentation. This final step moves improperly placed components into their proper text line and segments connected components if those connected components span more than one line (due to text from separate lines in the image which touch).

Word isolation is difficult because grouping word components based only on positional information is often impossible. We cannot be sure of proper grouping until the word is identified. Our approach divides word isolation into two parts, vertical isolation (inter-line segmentation) and horizontal isolation (intra-line segmentation). Once we have completed vertical isolation (Steps 2–4), we can restrict our word isolation activities to horizontal isolation (Step 5). This fifth step creates all reasonable word groupings and postpones the final word grouping decision until syntax is applied. General word categories are created in the sixth step. The seventh step applies syntax by looking for matches between word groupings, word categories, and domain syntax.

Word recognition is performed in the eighth step on words with the best matches between syntax and classification. Lexical information and word recognition could have been applied prior to using syntax, but applying lexical information with syntax allows the dictionaries to be further constrained. Semantics is applied in the last step after syntax and preliminary word recognition are known. At this point, all contextual

knowledge has been brought to bear on the problem, and we can determine if we have achieved an acceptable solution (which included proper word grouping, word identification, and semantics). If we have an acceptable solution, the desired semantic information is extracted and the system is complete. Otherwise, we return to Step 8 and repeat Steps 9 and 10 with another highly graded match between syntax and classification.

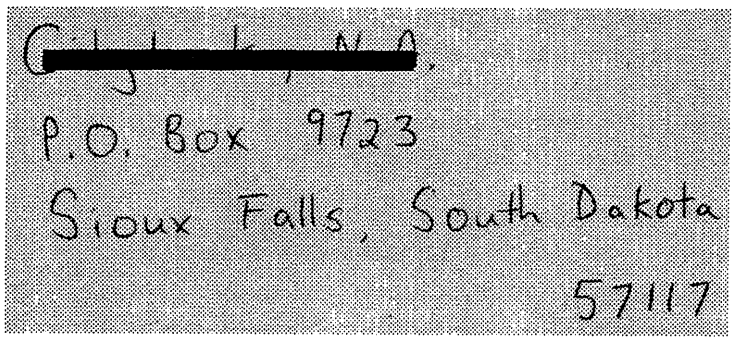
4. GENERAL APPROACH APPLIED TO HANDWRITTEN ADDRESSES — BACKGROUND

Using this approach, we have developed a system that determines the ZIP Codes of handwritten addresses. This system is able to locate and recognize ZIP Codes with a correct sort rate of 76.4% and an error rate of 1.2%. The error rate is computed as the ratio of the number of ZIP Codes in error (one or more digits wrong) to the number of images examined. The present system does not utilize recognition information beyond the ZIP Code and the state name for determining the first five ZIP Code digits of the address. (However, we do use punctuation, word position, and rough estimates of word shape to determine the syntax of the address image text.) We are in the process of developing methods to recognize city name, post office box number, and street address number. Preliminary results indicate that the performance will improve when this additional recognition information is used.

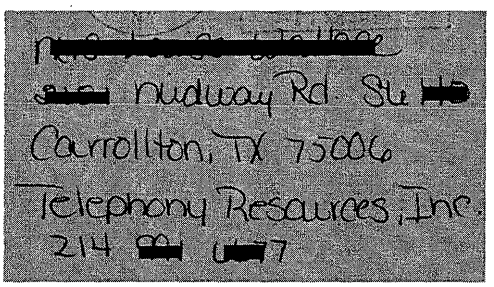
Handwritten ZIP Code recognition (HZR) is difficult due to wide variations in characters and digits as well as in address structure. Some characteristics are illustrated by the examples of Fig. 3, which are address blocks taken from actual mail.[a] Figure 3(a) is a very neat hand-printed address block in which the ZIP Code digits are prototypical. However, the address contains a P.O. Box number which is also a string of digits. Figure 3(b) is a hand-printed address block where the ZIP Code is not in the last line, which makes the ZIP Code location problem non-trivial. In Fig. 3(c), underlines are present in the address, which make ZIP Code location and digit segmentation difficult. Figure 3(d) is an example of a poorly formed cursive address that does not contain a ZIP Code.

An address has several natural constraints that can be exploited in locating and recognizing the digits of the ZIP Code. If the state name is known, we can determine the first digit of the ZIP Code and restrict the second digit to at most five choices. If the city name is also known, we can specifically determine the first three digits, and restrict the remaining two digits. If the street name is also known, we can precisely determine the complete nine-digit ZIP Code for that address. Conversely, if we know the first digit, we can restrict the state name to six possible choices. Aside from the redundancy, an address has a spatial syntax that can also be exploited in determining the location of function words (such as street, city, state).[15] The following section gives the details of our current HZR system (HZRS).

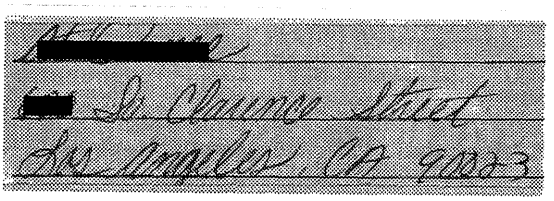
[a] The Appendix describes the image database from which these examples were taken.



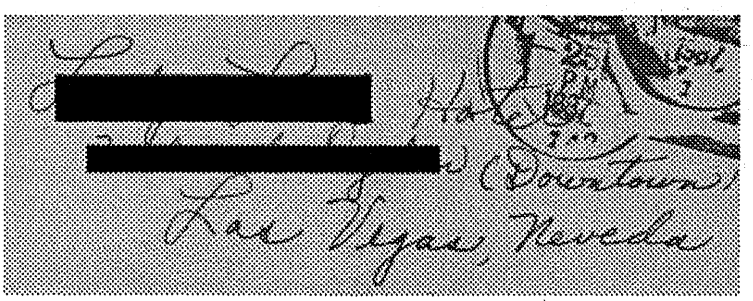
(a)



(b)



(c)



(d)

Fig. 3. Examples of handwritten address blocks extracted from actual mail. Addressee names are intentionally blacked out.

5. HZRS DESCRIPTION

A complete system for unconstrained HZR needs many components. At the image processing level, the written matter from the handwritten address block image must be distinguished from the background, and irrelevant lines such as streaks must be removed, etc. The text line locations in the address must be determined, the text lines 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.

The control flow in the current HZRS is based upon our general theory of reading unconstrained handwritten text in limited domains. There are ten operational modules in the system:

1. preprocessing;
2. line segmentation based on positional information;
3. feature identification;
4. text line segmentation (combining 2 and 3);
5. word segmentation;
6. word classification;
7. grade match of word classification and syntax;
8. recognition of words that have high syntax–classification matching;
9. use of semantics to constrain word recognition;
10. if a reliable semantic evaluation is found, derive an interpretation; otherwise, return to Step 8 until no reliable syntax–classification matches are found;

Subsections 5.1–5.10 give details of each of the ten modules, but we will first present a brief overview of the entire system.

The HZRS takes a gray-scale address block image as input. The image is binarized using a thresholding technique. Borders and underlines are removed from the image.

An initial line segmentation is performed. Connected components are placed into text lines based only on positional information. If connected components cannot be placed into single lines due to insufficient information and/or due to components from adjacent lines touching, they are assigned to a set of possible text lines.

Next, each connected component is identified (based on its size and shape) as a dash, a comma, a disconnected 5-hat,[b] or none of these. In addition, each connected component is identified as a digit or a non-digit. Note that there may be multiple identifications, e.g. a comma and the digit 1 may have identical shapes, but these multiple identifications will be reduced after positional information is considered.

When positional information and identification are combined, a more accurate interpretation of the text-line structure in the address is achieved. At this time, connected components can be moved to different lines, e.g. a comma may be moved from the line to which it is spatially closest to the line above. Connected components

[b] Handwritten 5's are often written in two strokes. The vertical bar and the bottom curve are written in one stroke and the horizontal bar is written as a separate stroke. When the horizontal stroke is not connected to the other stroke, it is considered to be a disconnected 5-hat.

that span more than one text line may be placed into a single text line or they may be split into two or more components and placed in separate text lines. At this point, we will assume that all components have been placed in their proper lines; however, there is still some flexibility in the syntax (described later) that allows us to compensate for improper line placement.

Positional information and identification are combined to determine multiple word segmentation hypotheses. Each text line can be divided into one to five words, and multiple hypotheses can be created for each number of words, i.e. a text line can be divided into two words in several different ways, depending on the location of the hypothesized word break in the line.

Each hypothesized word is tentatively classified. Words can presently be classified as *city*, *text*, *state-abbreviation*, *digits*, *9-digit ZIP + 4 Codes*, *digits-dash* (digits followed by a dash), *P.O.*, the word '*box*' (as in P.O. Box number), *noise*, and *barcodes*. Words can have multiple classifications, e.g. a word can be identified as *state-abbreviation*, *text* and *3-digits*.

All hypothesized words and word classifications are examined to find consistent classifications that match the expected syntax. For instance, if a line contained three words identified as *city*, *text*, and *5-digits* respectively, the words in the line would match the *city + state + ZIP Code* syntax.

Based on statistical analysis of information location in mail pieces,[16] we have determined the likely syntax constructions to be found in handwritten addresses. Although the most likely syntax of the address has the bottom line containing the city, state, and ZIP Code, many other address syntaxes are possible. For instance, line 1 (our line numbering begins at the bottom of the address) may contain a ZIP Code and line 2 may contain the city and state name. Similarly, line 1 may contain an attention line (e.g. 'Attn: Joe Smith'), line 2 may contain state and ZIP Code, and line 3 may contain the city. Matches between syntax, word groupings, and word classifications are determined and the most consistent matches are examined in order to find likely ZIP Code and state name candidates.

Words that match the syntax of a ZIP Code are selected as ZIP Code candidates. Depending on the number of consistent word classifications, zero or more ZIP Code candidates are selected. The system processes up to three candidates to determine the ZIP Code of the address.

To recognize a candidate, the ZIP Code segmentation program divides the candidate into either five or nine digits. The program determines the proper number of digits into which the candidate is divided. Then, four digit recognition algorithms are applied and their results are combined to determine the identity of each segmented digit. (Due to segmentation, these identities may be different from the original connected component identities.) After the ZIP Code candidate is processed, the state name candidate is processed. Our current character recognition algorithms only recognize hand-printed state names and state abbreviations. State name recognition information is used to compensate for ZIP Code recognition uncertainties. In addition to ZIP Code and state name segmentation and recognition, ongoing work involves segmenting and recogniz-

ing city names, and P.O. Box numbers. We plan on combining the recognition results of these words with the ZIP Code recognition information to allow more semantic processing.

Once the ZIP Code is recognized, we apply semantic information to determine the recognition reliability. If all digits are identified with sufficient confidence, the ZIP Code candidate is assigned a confidence value based on the recognition confidence of the individual digits and the amount of segmentation required. Furthermore, if the candidate has nine digits, the confidence is increased, because any candidate with nine recognized digits is very likely the correct ZIP Code. If the state name is recognized and the first two digits of the ZIP Code are not recognized with high confidence, a ZIP Code directory (containing state name listings) is used to assign high-confidence digit recognition (consistent with the state name) to replace the low-confidence ZIP Code digit recognition results. If any of the digits is still not recognized, the ZIP Code candidate is assigned a confidence value of -1. The confidence of the syntax match is also used to compute an overall confidence in the ZIP Code candidate. In addition, if recognized digits are not in a directory of valid US ZIP Codes, the ZIP Code candidate is assigned a confidence value of -1.

Up to three ZIP Code candidates are examined. The first candidate found whose confidence is greater than a threshold is selected as the ZIP Code for the address block.

5.1. Preprocessing

Three preprocessing steps are performed on address images in an effort to produce a binary image where text pixels are black and background pixels are white. These three processes (thresholding, border removal, and underline removal) are described below.

5.1.1. Thresholding

The gray-level images received by the HZRS contain only an address block. This is most often composed of 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 investigating several methods, we adopted a technique due to Otsu that performs quite well on handwritten address images.[17] This algorithm first computes the zeroth- and first-order cumulative moments of an image with L gray levels:

$$\omega(k) = \sum_{i=1}^{k} p_i \,, \qquad \mu(k) = \sum_{i=1}^{k} i p_i$$

where

$$p_i = \frac{\text{total number of pixels at gray level } i}{\text{total number of pixels in the image}} \,.$$

The optimal threshold is selected as the value of k, *viz.* k^*, that maximizes the

between-class variance, defined as:

$$\sigma_B^2(k) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$

$$= \omega_0\omega_1(\mu_0 - \mu_1)^2$$

where

$$\omega_0 = \omega(k)\ , \qquad \omega_1 = 1 - \omega(k)\ ,$$

and

$$\mu_0 = \frac{\mu(k)}{\omega(k)}\ , \qquad \mu_1 = \frac{\mu_T - \mu(k)}{1 - \mu(k)}\ , \qquad \mu_T = \mu(L) = \text{total mean level.}$$

Once the optimal threshold is determined, a pixel of the gray-scale image is converted to white if it is greater than k^*; otherwise it is converted to black. Experimental results have demonstrated that this technique performs well for the HZR application.

5.1.2. Border removal

An artifact of the digitization process is that some images have a very dark border near the image edge.[c] The borders are detected by scanning the image edges for large black areas (the borders are black after thresholding). Pixels which border both the large black areas and the inner image are used to determine a best-fit line along the border. A best-fit line is used to compensate for local variations along the edge. After the fit, pixels between the best-fit line and the nearest image edge are set to white, effectively removing the border.

5.1.3. Horizontal line removal

Approximately 5% of the time, the words in a handwritten address are written on machine-printed guide lines that are provided to assist the writer. Often the handwritten text intersects these horizontal lines. The text and the lines are then fused during thresholding and the result is an image that is not conducive to connected component analysis for locating the ZIP Code. Artifacts introduced by the lines could also distort the shapes of characters or digits and prevent them from being correctly recognized.

We develop a technique for horizontal line removal to solve this problem. Based on the run-length encoding method, this algorithm makes two passes in scanning through a thresholded image. In the first pass, the widths of horizontal lines are estimated. In the second pass, line segments which have widths within the estimated range are

[c] This was caused by the presence of black construction paper on the digitizing tablet. If an address was written near the mail-piece edge, the black construction paper might have come into the field of view. When digitized, the construction paper becomes a very noticeable black, i.e. the gray levels are very low.

located. These line segments are hypothesized to be parts of horizontal lines and are removed.

5.2. Initial Text Line Segmentation

Before a proper semantic interpretation can be achieved, the words or digit strings in the image must be located. A line segmentation algorithm separates the image into text lines. The line segmentation is divided into two steps. In the first step, positional information is used to determine line segmentation for 'easy-to-place' components. These initials results are refined in Step 4 when feature identification information is available.

The initial line segmentation begins by performing a bottom-up technique we call *shading*. The shading algorithm takes a binary image as input and divides the image into vertical strips (currently 100 pixels wide for 300 ppi images). For each strip, a horizontal profile (containing one position for each pixel row) is made. Each position in the profile (for each strip) is set to 1 if any image components pass through 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 (see Fig. 4). All block positions are recorded, unless they are too small (vertically), in which case the undersized blocks are considered noise and are ignored.

After all blocks from all strips are created, the blocks are connected to form lines (of address block text). Testing has shown that blocks from adjacent strips which overlap vertically are usually from the same address text line. Blocks which are horizontally adjacent and vertically overlapping are connected, forming a topological graph of the blocks, where blocks are represented as nodes and block connections are represented as node connections. Normally, the topological map requires adjustment to reflect the actual address block line (see Fig. 5). We use heuristics to separate and/or join blocks in each vertical strip and their inter-block connections, such as that shown in Fig. 6.

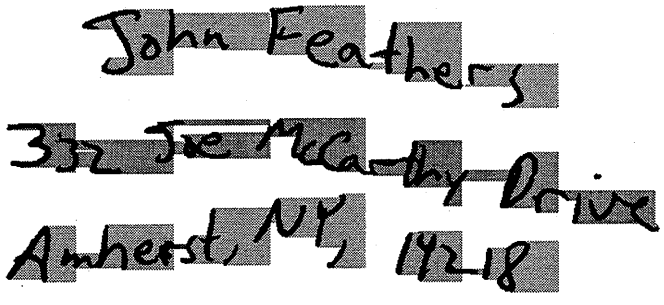


Fig. 4. Address block with shaded portions representing the blocks. Each text line in the address has shading of a slightly different density, indicating that the shading algorithm determined the line structure correctly.

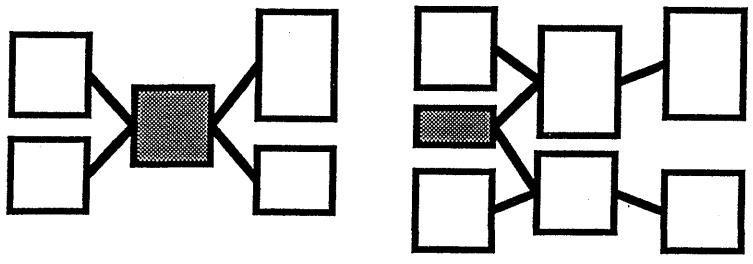
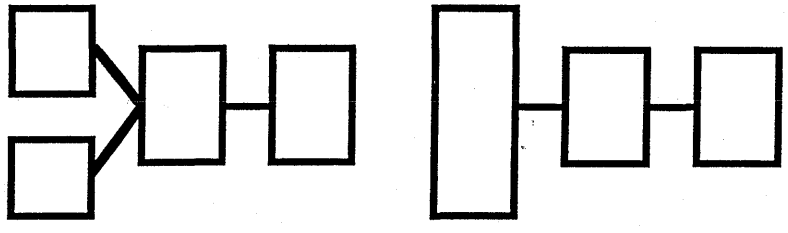


Fig. 5. Block configurations that may be ambiguous. Determining whether the textured block belongs to the upper or lower line is not straightforward.



(a) (b)

Fig. 6. Sample heuristics for combining blocks. The topological map of (a) is transformed into the topological map of (b) by combining two blocks into one.

Once the arrangements of blocks into text lines is determined, connected components that are located entirely in blocks from a single line are assigned to that line. Connected components that overlap blocks from several lines are assigned to the set of lines those components overlap. These overlapping connected components are refined in Step 4.

5.3. Feature Identification

Feature identification is performed in two stages: shape identification and spatial confirmation. In the first stage, each connected component is identified as having (or not having) the proper shape of a feature. Features are currently dashes, commas, disconnected 5-hats, and digits. During this stage, connected components may have multiple feature identifications. For instance, a short horizontal bar may be identified as having the shape of a dash and a disconnected 5-hat.

Spatial and line segmentation information is then used to determine if a component has the proper positional characteristics. For example, to be a disconnected 5-hat, the short horizontal bar must have on the same line a left-neighbor component which must not be a comma, a dash or a disconnected 5-hat and must either be identified as the digit 5 or an unrecognized digit, etc.

The constraints for shape identification and spatial orientation are not disjoint. If a component has good positioning, its shape constraints will be relaxed (and vice versa). For instance, if a component is a likely 'comma location', shape constraints are relaxed to allow odd-shaped commas to be identified as commas.

5.4. Final Line Segmentation

In some cases, line segmentation cannot be performed based solely on the positioning of connected components. Some connected components are positioned closer to another text line than the actual text line to which they belong, e.g. commas are often closer to the text line beneath their actual text line. Therefore, we have developed a set of heuristics based on empirical studies of handwritten text.

For instance, if a component has the shape of a disconnected 5-hat, and is positioned near the bottom of the text line in which it is currently placed, it is definitely not a disconnected 5-hat in that line. We then determine the likelihood of this component being a disconnected 5-hat in the text line beneath the current text line. That is, if this component were placed in the text line beneath the current text line, we determine whether it would 'fit' as a disconnected 5-hat. If the component 'fits' well as a disconnected 5-hat in the lower text line, it is placed there as part of the line segmentation.

In a similar manner, components discarded as noise can be placed back into text lines. Components with the shape of commas and dashes are sometimes initially discarded due to their small size. However, if these components have a shape and position that is consistent with a comma, a dash, or a disconnected 5-hat, they can be placed back into the text lines.

In addition to moving components to other text lines, some connected components must be forcibly segmented and placed in more than one line. These components are typically formed by characters from adjacent lines that touch. In these cases, we use connected components with high-confidence line placement to determine where a connected component should be split by forced segmentation. Using those high-confidence connected components, we can determine best-fit lines between text lines. That is, using the corner points of the bounding boxes around the high-confidence connected components, we determine a dividing line between two text lines. For example, the dividing line between line 1 and line 2 could be determined by using the upper corners of bounding boxes associated with line 1 and the lower corners of bounding boxes associated with line 2. This line can now be used to segment connected components that span line 1 and line 2 (Fig. 7).

Although we cannot guarantee perfect performance even after final line segmentation, the results are usually good enough to allow the address to be parsed into words. Furthermore, when the line segmentation process fails, it usually fails in just part of the address, i.e. most of the lines are still segmented properly. Once text line segmentation is complete, the text lines can be segmented into words.

5.5. Word Segmentation

To perform word segmentation, each text line is segmented into clusters of

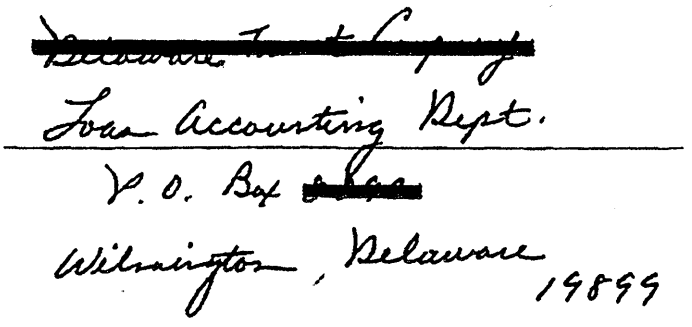


Fig. 7. An example of forced segmentation. The best-fit line(s) is (are) drawn to segment connected components that span more than one text line.

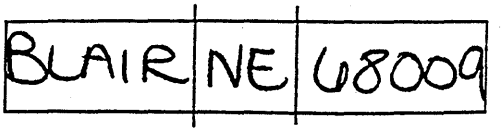
connected components, where each cluster can be considered a word. Words are formed by ordering the components of a text line from left to right and selecting word breaks between components, e.g. if components were in the order 1–8 and a word break was selected between components 3 and 4, then we would form two words: components 1–3 and components 4–8. Word breaks are hypothesized based on horizontal spaces between connected components, a shift in the mean vertical position of connected components, and the location of commas (Fig. 8). The word break locations are ordered based on their confidence, i.e. a large gap would be a high-confidence word break. Then, different word break groupings are chosen to separate the text line into 1–5 words.

Note that a text line may be segmented into a given number of words in several ways (Fig. 9). Each word break grouping creates different words, and each configuration is assigned a confidence. Each word created by word segmentation is classified as described in the next section.

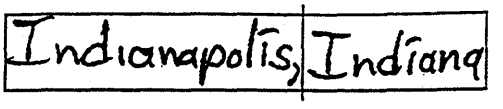
5.6. Word Classification

Words are classified based on the set of connected components which constitute that word. The features and the spatial locations of those features determine the word classification. Word categories are *text*, *city*, *state-abbreviation*, *digits* (including an estimate of the number of digits), *digits-dash*, *9-digit ZIP + 4 Codes*, *barcodes*, *P.O. Box*, and *noise*.

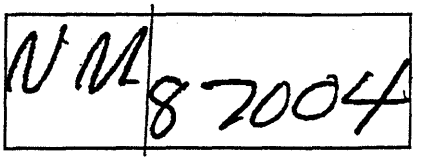
Heuristics is used to classify each word into one or more of these categories and assign a confidence to the word. For example, if a word contained four connected components (where three components were recognized as digits and the fourth was not recognized as a digit or as another feature), one word classification would be *digits*. The 'digit size' of the unrecognized component would be estimated based on the ratio of the average size of the three recognized digits and the unrecognized component size. If the unrecognized component was 2.2 times as large as the average recognized digit size, the word would be classified as a *5-digit* word (by estimating that the unrecognized component was actually two digits). The word classification confidence



(a)



(b)



(c)

Fig. 8. Word breaks can be indicated by three different types of features: (a) a horizontal gap between words, (b) a comma present between two words, (c) a vertical shift in the text line.

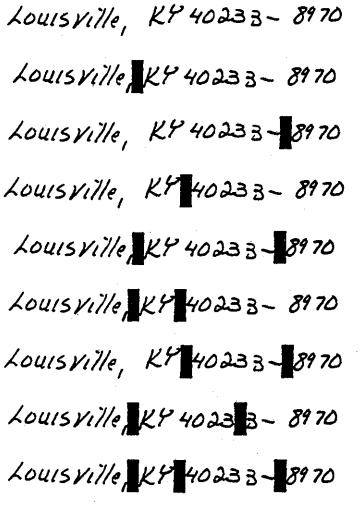


Fig. 9. Possible ways of dividing a given text line into words.

would be assigned based on the ratio of the number of recognized digits to the number of connected components in the word.

This word would also be classified as *text*, with a lower confidence. The text classification is needed, because text (particularly hand-printed text) can be misrecognized as digits. It is important that the correct classification is among the classification choices, even if extraneous classifications are made (this allows all consistent line interpretations to be examined). Naturally, too many extraneous classifications would increase the search space and hamper perception of the true classification. In our testing, each word is typically placed into three or four categories.

5.7. Grading Word Categories to Syntax

Addresses (and certain other domains) have a syntax that is generally followed. Even when the syntax is not followed precisely, portions of the address will have reasonable syntax. Using the statistical analysis[16] and empirical studies of our handwritten address database, we developed a technique to match word classifications to the most probable word syntax.

The syntax is two-dimensional, since words are horizontally ordered along a text line and the text lines are ordered vertically. One common configuration for handwritten addresses is for the bottommost line to contain three words: city, state, and ZIP Code (CSZ). To determine if the text in a handwritten address contains this configuration, we must see if the bottommost line can be divided into three words and if each of these three words can be matched to the expected form for each word. Furthermore, we must allow for some inexact matching. The first of the three words must have the characteristics of a *city* word. In the word classification step, this first word may have been classified as a *city*, and we would have found a match. However, the word may have been classified as *text* and not *city*. Although either of these classifications would be a match to *city*, the first one would be a higher reliability match. A similar approach is required for matching the state name and ZIP Code.

Grading matches between image text and syntax is achieved by ranking high-confidence syntax matches with a highly probable syntax over those with low-confidence syntax matches with less probable syntax. For example, three words classified as *city*, *state-abbreviation*, and *5-digits* would match the CSZ syntax better than three words classified as *text*, and *4-digits*. Note that if the number of digits was estimated, a word classified as having four digits might not actually contain four digits, e.g. it might contain five. Similarly, a syntax with CSZ in the bottom line would be ranked higher than a syntax with *P.O.* + *Box* + *number* in the bottom line.

When the match grading is complete, we try to locate ZIP Code candidates. The HZRS looks for the top three syntax matches that locate ZIP Code candidates. That is, if a syntax match had the word 'ZIP Code' as one of its components, and the syntax match was graded as one of the top three syntax matches containing the word 'ZIP Code', then the image text associated with that word would be a ZIP Code candidate. In addition to locating ZIP Code candidates, the syntax matches allow us to find other words (such as city name, street number, etc.) that (when recognized) can help refine the semantic interpretation of the ZIP Code.

5.8. Recognizing Words

Once we have determined the location of words, the next step is to recognize important words using techniques designed for those types of words. Our current system only recognizes ZIP Codes and state names. When a ZIP Code is located, we segment it into digits and recognize the digits. Similarly, when a printed state word is located, we segment it into characters and recognize them. For other types of words, we are developing other word recognition techniques which may not use the traditional segment-and-classify methodology. This methodology is usually most successful for machine-printed text, where the boundaries for characters or digits are either well determined or can be guessed with reasonable accuracy. However, selecting boundaries is more difficult in unconstrained handwritten addresses, where the text can be cursive or printed and where the letters or digits can be touching each other.

We now describe the details of ZIP Code recognition. Before its digits are recognized, ZIP Code recognition requires several preprocessing steps. The objective of these steps is to reduce the immense variability present from one ZIP Code to another and to give the digit recognition algorithm images that are as free from writer-dependent effects as possible. These segmentation, preprocessing, and recognition algorithms are described below.

5.8.1. ZIP Code segmentation

Given an image which contains a full string of ZIP Code digits, the segmentation algorithm will segment the whole image into regions each containing an isolated digit (see Fig. 10). The resulting images from this subsystem will be passed to an isolated digit recognition subsystem. In general, a legal ZIP Code consists of either five or nine digits with or without a dash in the middle. The segmentation algorithm applies this information to invoke its 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 (location, height, width, area size, etc.) of each connected component in the image are recorded. Other attributes based on the whole string of digits in the image are also computed and recorded. These attributes consist of its upper and lower contour profiles, the topological relationship between connected components (e.g. left of, right of, etc.), and the positions of the ZIP Code string's upper line, central line, and lower line. In addition to the creation of data structures, several operations are performed on the image, e.g. sorting connected components according to their heights, selecting digit candidates from all the connected components, detecting and removing of underlining, removing noise or insignificant components, removing long ligatures on the tails of digit candidates.

Using these attributes, the program estimates the number of digits in the image. In addition, the spacing between connected components is used to group components into digit clusters, each of which contains one or more digits. We also estimate the number of digits in each cluster.

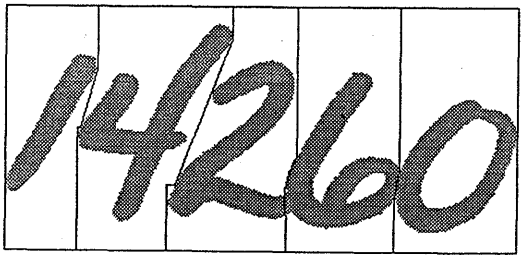


Fig. 10. An example of ZIP Code segmentation.

Grouping and dissecting operations are performed as needed to generate the isolated digit images from the clusters. To extract non-touching digits, line segments are drawn to enclose zones that contain isolated digits. Grouping operations merge separate connected components and their containment areas into a single digit zone. Grouping is required when a digit is fragmented due to thresholding or when more than one connected component constitutes a complete digit (digits 4 and 5 are the most frequently seen cases). The grouping function includes the merging of connected components in either neighboring clusters or a single cluster. Dissecting operations split connected digits into isolated ones, and try to maintain the original digit shapes for recognition purposes. Once the number of touching digits is determined, the splitting routine selects the dissecting positions and dissects the connected component into that number of digits. Several factors influence the selection of dissecting positions: 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 profiles respectively, and the number of strokes near where touching is predicted. Based on factors found, the dissector will use different techniques to separate digits. For example, if both peak and valley are found near the predicted position, the dissector will segment the gap between two digits by directly cutting through the image from the peak to the valley. If only a peak or valley is found, a similar operation called 'hit and deflect' is performed. This technique is always guided by the slant angle. If no peaks or valleys are found, a cut is made through a 'single stroke' area.

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

5.8.2. Size normalization

The polynomial discriminant and the statistical and structural character recognition algorithms in the HZRS 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'. The result is an image of size $h \times w$.

Height normalization is given an image of size $h_i \times w_i$ and produces an image of

size $h \times w'$, where

$$w' = pw_i ,$$

and

$$p = \frac{h}{h_i} .$$

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 two cases are considered. 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 $((x/w')w, y)$ is set to black if and only if the pixel at (x, y) is black.

5.8.3. 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.[18,19] We have followed a parallel approach in order to achieve a high recognition rate and a low error rate. Our method consists of four algorithms: (i) a polynomial discriminant algorithm to make a decision based on the overall holistic characteristics of the image, (ii) a statistical approach that uses structural features of the character, (iii) a rule-based classifier that uses information about the size and placement of strokes in the image, and (iv) a rule-based algorithm based on contour features. These four 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 image of the same digit and their results are combined with a decision tree to classify the image.

The subsequent sections describe each digit recognition algorithm.

Polynomial Discriminant Method

The polynomial discriminant method for digit recognition[20] computes a second degree polynomial from the binary image of a digit after it has been normalized to 16×16. The basic calculation can be summarized as

$$d_{(1\times 10)} = X_{(1\times k)} \cdot A_{(k\times 10)}$$

where, in our implementation, $k = 1241$. This is a heuristically chosen number of

pairs of pixels. Thus, each X_i represents the product of two pixels $x_j \cdot x_l$, where j and l range from 1 to 256 are not necessarily equal. The recognition decision is the class that corresponds to the maximum value in d.

The matrix A is computed during a training phase by a least-squares calculation:

$$A = (Z^T Z)^{-1} Z^T Y$$

where Z is the $k \times N$ matrix formed by appending the N vectors as X in a set of training data. Y is the $N \times 10$ matrix formed by appending the N vectors as $y_1 \times 10$, where y_i is 1 if the class of the corresponding sample in Z is i, and is 0 otherwise.

The operation of this method is simple and cost-effective. After normalization to 16×16, the 1241 pairs of pixels that were used to develop the A matrix are accessed. If both members of the pair are black, the ten entries in A that correspond to it are added to ten accumulators. No action is taken if one or both members of the pair are white. After this step, the two accumulator cells with the maximum values and the difference between these values are returned. This is because it has been observed that this difference value is a reliable indication of the accuracy of the first choice: a high value for the difference means that the classes are well separated. The cost-effectiveness of this method is seen in that the only computation required at run-time, aside from normalization, is the access of 1241 pairs of pixels and at most 12 410 table lookups and additions. An insignificant amount of computation is needed to search for the two accumulator cells with the maximum values.

Statistical and Structural Analysis of Boundary Approximation

This method approximates the contour of a digit with a piecewise linear fit, computes features from that fit, and places them in a feature vector. The best matches between this vector and a stored set of labeled prototypes are determined. The matching is done with 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.[21] The classifier is a mixture of a structural decision tree classifier that determines in which subset of classes a given vector belongs and a modified nearest-neighbor classifier that is then used on the feature vectors in those classes.

An example of a piecewise linear contour description of a digit is shown in Fig. 11. The features extracted from these descriptions are shown in Table 1. The classifier is a mixture of a structural decision tree classifier and a k-nearest-neighbor classifier. The structural classifier determines in which subset of classes a given vector belongs. The k-nearest-neighbor classifier is used on the feature vectors in those classes. Figure 12 shows a representation of the classifier.

The mixed statistical and structural approach was trained on different numbers of digits and tested on a distinct set of 2418 digits. All digits were segmented from the ZIP Codes in mail-piece images. The best performance was 86.6% correct with a 12.5% error rate.

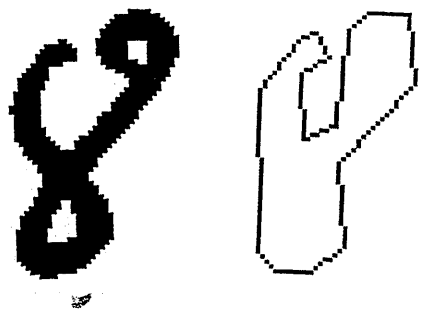


Fig. 11. An example of a digit and its piecewise linear description.

Table 1. 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, 6, 7	distance from the top of the character to the largest, second-largest and third-largest concave 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)

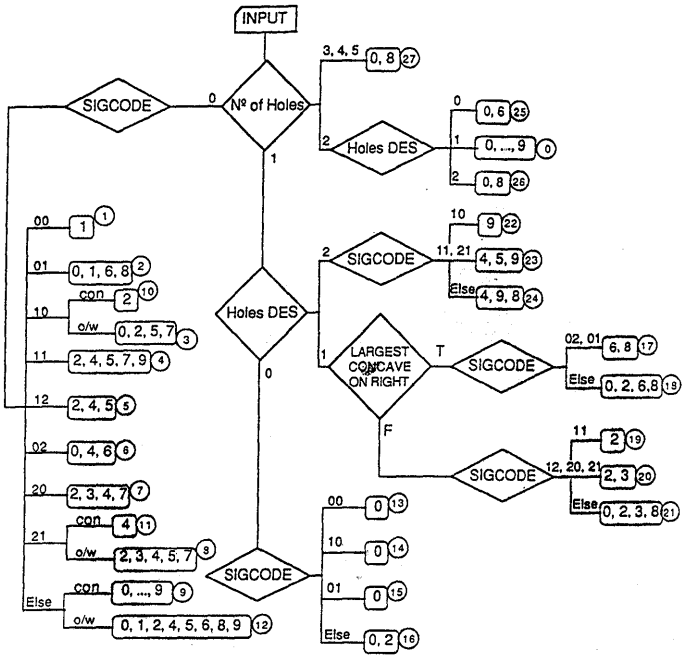


Fig. 12. The digit classifier for the mixed statistical/structural approach.

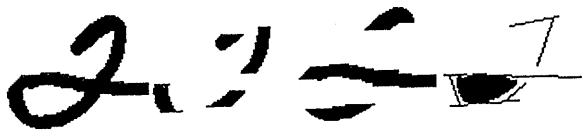
Rule-Based Stroke Analysis

This structural method of recognition decomposes a digit into strokes that are used as features for classification. The feature extractor computes the digit's left and right contour profiles, run-lengths, holes, and estimated stroke width. By using the estimated stroke width, a process is applied to extract strokes from the digit pattern. Then, the digit is completely described as a group of nearly horizontal strokes (H-strokes) and nearly vertical strokes (V-strokes). An example of stroke decomposition and features extracted from a digit is shown in Fig. 13.

The digit classifier uses the detected strokes, holes and the character contour profiles as features. The classifier consists of rules arranged in a hierarchical manner. Each rule specifies a certain type of digit. The rules describe the structure and the topological criteria of various types of digits. For instance, a rule for one type of digit 0 has the following description:

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

These rules were constructed by summarizing the results of the stroke decomposition process as applied to 1754 digit samples of training data. We categorized the results as different prototypes, then constructed the corresponding rules. These rules enabled the



(a) (b) (c) (d)

Fig. 13. An example of stroke decomposition and feature description of a digit '2': (a) original digit image, (b) its vertical strokes, (c) its horizontal strokes, and (d) the hole with the vector representation of all the strokes.

classifier to recognize 79.7% of the 2418 digits with a 4.0% substitution rate. When a digit is recognized, the classifier gives a recognition result and a confidence value associated to that particular rule which matched. Further details of the digit recognition algorithm based on structural analysis of strokes are given in Ref. 22.

Contour Analysis Method

A contour analysis method[23] has been fully developed for digit recognition and is being modified for character recognition. It calculates the curvature at every point on the outer and inner contours of a binary image. Eight feature types are defined based on the amount of curvature present at any point. Three features are used for concave curvature, and five for convex curvature. Each feature is also associated with its direction and location. The feature string extracted from an unknown character is matched against a rule base to achieve recognition.

Figure 14 shows examples of the eight features as well as how direction and location are defined. Direction is quantized to eight compass points and location is quantized to a 4 × 4 cartesian grid with the origin in the upper left corner.

An example is shown of a feature string extracted for an image of a '3':

B5x2y2 S4x1y1 B5x3y0 S4x0y0 A0x3y0

B0x2y1 A0x3y2 S3x0y3 B3x2y3

Each entry describes the presence of one feature. It gives the type, direction, and location in a clockwise fashion. For example, the first feature extracted was a bay facing southwest in the middle of the image. Then, a spur was detected facing west near the top, and then the next feature was another bay, and so on. The extraction continues until the whole image has been analyzed.

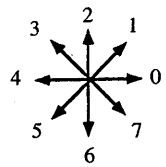
Following extraction, the feature string is compared against a rule base of over 130 rules. Classification is done by a decision tree. Each rule narrows down the possible choices for the next branch of the tree. If no rule is matched in the current branch, the classifier backtracks to the previous branch and tries again for a match.

The rules are generalized to match many feature strings with one rule. Each rule is made up of clauses, and one clause will match one feature string. A clause can contain either positive or negative information. Positive information requires the feature to be

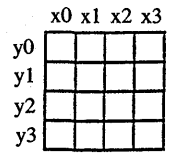
Border Features

Spur	highly convex sharp endpoint
Stub	small highly convex
Wedge	convex with far sides
Curl	short round convex
Arc	long round convex
Null	very small concave
Inlet	small concave
Bay	long concave

Compass Directions



4 × 4 Grid



Lake Features

Small	0–10% of bounding rectangle
Medium	10–30% of bounding rectangle
Large	Greater than 30% of rectangle

Fig. 14. Examples of the eight features and their direction and location definitions.

present, while with negative information the feature must be absent.

The final rule that matched the above feature string is:

Class: 3 Rule: Plain '3' digit

clause 1:IBN 017 x1x2x3y1y2 Y
clause 2:IBN 067 x1x2x3y1y2y3 N
clause 3:BI 23456 x0x1x2x3y1y2y3 Y
clause 4:BI 23456 x0x1x2x3y0y1y2 Y
clause 5:SWCA 23456 x0x1x2y0y1y2 Y

This rule describes many images that fall into the class of 3's. Once the first clause matches a part of the feature string, all other clauses must follow in order. Clause 2 denotes negative information; it cannot match any part of the feature string between clause 1's and clause 3's invocations.

The first clause of a rule tries to define a unique feature of its class. In this case,

the first clause is described as:

feature type — Inlet, Bay, or Null
direction — facing East (directions 0, 1, and 7)
location—on the right side, in the middle (x-coordinate: x1, x2, and x3, y-coordinate: y1 and y2)
positive information — Yes, these features must be present

The first clause matched feature B0x2y1. Clause 2 is negative and so should not match any features until clause 3 is matched. Clause 3 matches B3x2y3, and the rest of the clauses find matches also. The classifier skips information in the feature strings that is not needed. This enables general rules to be written for varying handwriting styles in the same class.

This method has been developed and tested on handwritten digits. The classifier was trained on 2456 unconstrained handwritten digits and tested on 540 test digits. A correct rate of 85% has been achieved with a 9% substitution rate.

Combination of Results

The results of applying the four digit-recognition algorithms discussed above are combined with the decision tree shown in Fig. 15.

The first test in the tree is: If the distance of the polynomial discriminant method is greater than a threshold and the best decisions of the four techniques agree, then output that decision. Subsequent tests check other combinations of decisions. There are 17 such tests in the tree.

Application of the combined approach to the same 2711 digits that the polynomial discriminant, structural and statistical, stroke, and contour analysis methods were tested on resulted in a correct rate of 95.54% and an error rate of 1.99%. Also, a 'normalized' performance is shown in which the error rate was adjusted to be below 1.5%. This was accomplished by rejecting decisions on certain branches. The best normalized performance was 95.02% correct with a 1.33% error rate. We also tested the digit recognition system on a 'good' set of digits. These were the 1762 digits that were successfully segmented. Thus, these were digits that were free of artifacts and did not touch other digits. We achieved a 97.1% recognition rate with an error rate of 0.96% on these digits.

The state name recognition algorithm is in an early development stage. The HZRS currently only recognizes hand-printed state names and state abbreviations. Two character recognition algorithms are used to identify isolated characters. The recognition results are then compared to a dictionary of state names (full and abbreviated). This straightforward approach allows us to recognize 22% of the state names which contribute 1.8% to the 76.4% total correct rate. We fully expect the state name recognition algorithms to be replaced by more sophisticated ones. However, for completeness, we will briefly describe the state name segmentation and recognition algorithms currently used in the system.

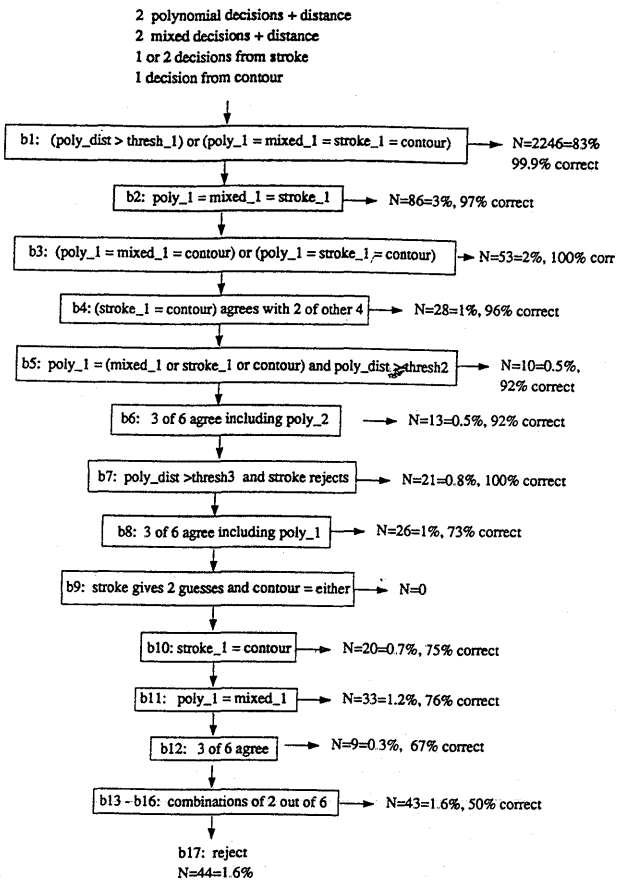


Fig. 15. Decision tree for combining the results of three digit-recognition algorithms.

State Name Segmentation

The segmentation algorithm accepts a state word image as input and outputs a set of character images. The state name segmentation algorithm performs a connected component analysis and groups isolated connected components together to form characters. No forced segmentation of connected components is performed. The grouping algorithm joins components that are too small to be characters with the nearest components. This scheme does not work when characters touch (such as in cursive writing), but it does work for hand-printed text even when individual characters may consist of several components.

Character Recognition

After segmentation, each isolated character image is passed to two separate character recognition algorithms. The first is a template matching algorithm that uses a nearest-neighbor classifier. The second uses a Bayesian classifier with a set of structural features.

The two recognition algorithms had the same correct rates, with 90.1% correct and 9.9% error. They were combined by accepting a result only if the top choices in them agreed. If the top choices disagreed, the result was rejected. This combination resulted in an 80.4% correct rate with a 5.0% error rate.

The character recognition results are compared against a dictionary containing 143 state names and abbreviations. If all the characters in the state name image are recognized and they match a dictionary word, or if three or more recognized characters match characters in the corresponding positions in a dictionary word, the state name from the dictionary is accepted as an answer. Otherwise, the state name word recognition is rejected. By the use of this algorithm, 22.0% of the automatically located state words were recognized with a 2.8% error rate.

5.9. Using Semantic Information to Constrain Word Recognition

Once all relevant words have passed through the initial recognition phase, semantic information is used to further improve recognition performance. We have just begun developing handwritten word recognition algorithms; therefore, we have not fully implemented all our techniques for using semantic information to read address information. Still, preliminary results show that semantic information can be applied in a useful manner.

One straightforward semantic check is to determine if the recognized ZIP Code is in a United States Postal Service (USPS) directory of all valid ZIP Codes. The USPS has assigned approximately 40% of all possible 5-digit ZIP Codes. Therefore, if errors are generated randomly in the first five digits, it is expected that 60% of the errors will be detected (and removed) by this test.

A more sophisticated use of semantic information is in our reading of the street address line. Reading the street address line allows us to convert a 5-digit ZIP Code to a 9-digit ZIP + 4 Code. The extra four digits give a more accurate semantic evaluation of the delivery address for the mail piece.

The street address line is located by searching for a line in the address that contains *digits* followed by *text*. Based on statistical analysis of address images, we know that the street address line must be located beneath the top address text line and above the text line containing the city name (or state name if the city name does not exist). If a street address is found (street address lines occur in 75% of letter mail), the system tries to recognize the street number. If the street number is recognized with high confidence, we create a dictionary of possible street names.

The dictionary is created using the National Carrier Walk Sequence (NCWS) directory. This directory lists all valid mail delivery stops and includes a listing of all valid street addresses. By knowing the 5-digit ZIP Code and the street number, the NCWS can provide all the street names that have that ZIP Code and street number. Figure 16(a) shows a sample address and the located street address line (Fig. 16(b)). The street number *369* is recognized and we are able to create a preliminary dictionary, shown in Fig. 16(c). This dictionary contains the official USPS abbreviation for the street address. However, we must expand the dictionary to include all

abbreviations that people normally use for that street address. For instance, *AV* is expanded to *AV*, *AVE*, *AVEN*, *AVENU*, *AVENUE*, and *AVN*. Expanding all street names from the preliminary street names creates the final dictionary shown in Fig. 16(d).

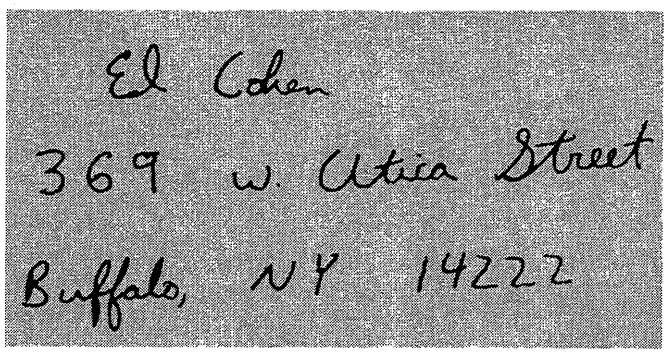
Next, we pass the dictionary and the street name image to the handwritten word recognition algorithm. In our current algorithm, everything to the right of the street number (on the same text line) is assumed to be part of the street name. The handwritten word recognition algorithm extracts features, e.g. ascenders, holes, sharp curvature inflections, lower contour, and recognizes isolated characters from the street name image. It then determines the degree of the match between features extracted and features expected to be found in dictionary words (recognized characters are similarly matched). This word recognition algorithm is described in more detail in Ref. 24.

If a street name has a significantly better match than the others, then it is considered to be correct. Figure 16(e) shows the list of dictionary names followed by the number of features and recognized characters that match the dictionary word. Since *West Utica Street* is the highest matched name, we consider that to be correct. We are then able to determine the ZIP + 4 Code for the mail piece to be *14222-1909*. This example shows how in using semantic information we were able to make the word recognition task easier (by supplying a small dictionary of valid words) and encode a precise semantic description of the address (the ZIP + 4 Code). Preliminary results show that our system is able to locate and recognize over 20% of the street address lines correctly.

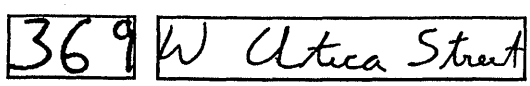
State name recognition is also used to improve system performance. In a valid address, the state name and the first two ZIP Code digits strongly constrain each other. If the state name is known, we can uniquely determine the first digit and constrain the second digit to be one of at most five digits. If the first digit is known, the number of possible state names can be reduced from 50 to at most 5. If both of the first two digits are known, the state name can be restricted to three possibilities. These constraints are used to help us determine the ZIP Code when its digits are poorly recognized.

ZIP Code recognition results are normally accepted only if all its digits are recognized at a sufficient confidence level. However, if a state name is recognized and either of the first two digits is not recognized with high confidence, the digit recognition results are replaced with digits taken from a directory that matches the state names to ZIP Codes. For instance, if only the first digit in a ZIP Code was not recognized (*?4215*), and the state name was recognized as *New York*, by knowing that all New York ZIP Codes begin with the digit *1*, we would be able to determine the ZIP Code to be *14215*.

Currently, state name recognition improves system performance from 74.6% to 76.4% correct recognition of 5-digit ZIP Codes. We expect this figure to improve as we further develop character and word recognition capabilities. We also plan to develop city name recognition which will help in recognizing the last three digits of a 5-digit ZIP Code. Furthermore, if no ZIP Code is present (or it is unrecognizable), the



(a)



(b)

14222-1713 369 RICHMOND AVE #1
14222-1713 369 RICHMOND AVE #2
14222-1713 369 RICHMOND AVE #3
14222-1713 369 RICHMOND AVE #4
14222-1909 369 W UTICA ST
14222-1703 369 NORWOOD AVE

(c)

NORWOODAV	RICHMONDAVN
NORWOODAVE	WESTUTICAST
NORWOODAVEN	WESTUTICASTR
NORWOODAVENU	WESTUTICASTREET
NORWOODAVENUE	WESTUTICASTREETS
NORWOODAVN	WESTUTICASTRT
RICHMONDAV	WUTICAST
RICHMONDAVE	WUTICASTR
RICHMONDAVEN	WUTICASTREET
RICHMONDAVENU	WUTICASTREETS
RICHMONDAVENUE	WUTICASTRT

(d)

Fig. 16. Stages in reading a street address line. (a) Original image, (b) street address line extracted, (c) preliminary dictionary of street names, (d) expanded dictionary of street names.

WUTICASTRT	10
WUTICASTREETS	10
WUTICASTREET	10
WESTUTICASTREET	10
WESTUTICASTRT	10
WESTUTICASTREETS	10
WUTICAST	9
WUTICASTR	9
WESTUTICAST	9
WESTUTICASTR	9
RICHMONDAVE	3
RICHMONDAVEN	3
RICHMONDAVENUE	3
RICHMONDAV	3
RICHMONDAVENU	3
RICHMONDAVN	3
NORWOODAVN	2
NORWOODAVENUE	2
NORWOODAVENU	2
NORWOODAVEN	2
NORWOODAVE	2
NORWOODAV	2

(e)

Fig. 16. Cont'd. Stages in reading a street address line. (e) Ordered list of dictionary words returned from the handwritten word recognition algorithm.

ZIP Code can usually be determined if the city name and state name are recognized.

Semantic information is also available from the P.O. Box number. If a 9-digit ZIP + 4 Code is recognized, its 4-digit add-on can be confirmed by recognizing the P.O. Box number. If a 5-digit ZIP Code is recognized, its 4-digit add-on can be determined by recognizing the P.O. Box number. The process of locating and recognizing P.O. Box numbers is similar to that of reading street addresses; however, P.O. Box numbers do not require the use of handwriting recognition algorithms. At present we are able to correctly locate and recognize over 50% of the P.O. Box numbers.

6. TWO EXAMPLES OF THE HZRS ON ADDRESS IMAGES

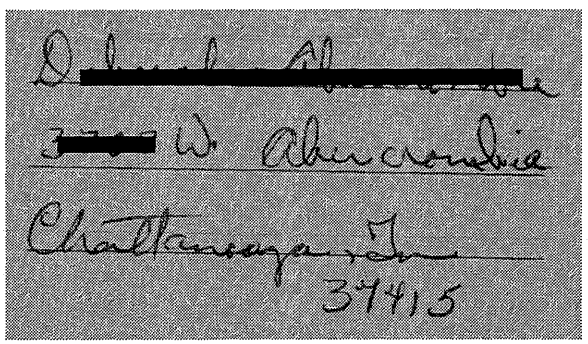
Two examples of the operation of the complete system are shown in Figs. 17 and 18. Figure 17 shows a straightforward example of ZIP Code recognition. Figure 18 gives an example where the initial ZIP Code location failed, but the system correctly located the proper candidate and processed the second candidate correctly. In the following paragraphs, we first describe the results of Fig. 17 and then discuss Fig. 18.

In Fig. 17 some of the intermediate processing steps of the address image are shown. The gray-scale image is seen in Fig. 17(a), and the thresholded image in Fig. 17(b). Underlining was removed (Fig. 17(c)) and connected component analysis

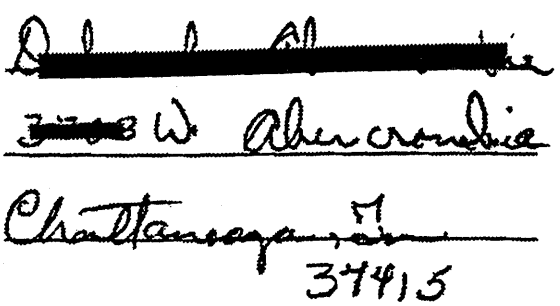
was performed to segment the address into text lines and words. The personal name and street address were intentionally blacked out, in accordance to our agreement with USPS.

Each text line was segmented into several different word groupings and the selected word segmentation combined with syntax choice is shown in Fig. 17(d). In this example, the first word was classified as *text* with confidence 0.9. The second word was classified as *box* (as in 'P.O. Box number') with confidence 0.9 and as *text* with confidence 0.9. Since we do not currently have full character recognition capabilities, we had to rely on the size of the words and punctuation for text classification, which resulted in a belief that the state name was actually the word 'box'. The third word was classified as *5-digits* with confidence 0.49 and as *text* with confidence 0.2.

The syntax *city-state-ZIP* matches to *text-text-5-digits*. Based on this match, we selected the 5-digit word as the ZIP Code candidate. The ZIP Code candidate (Fig. 17(e)) was segmented and the isolated digits were passed to the digit recognition algorithm. The results from digit recognition and their confidences are shown in

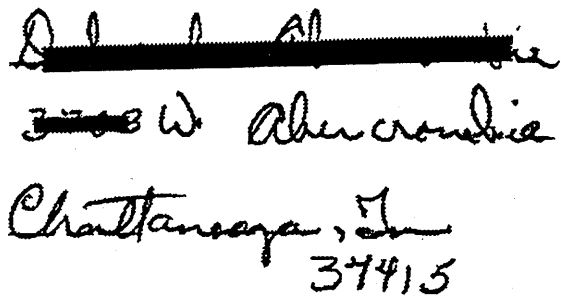


(a)

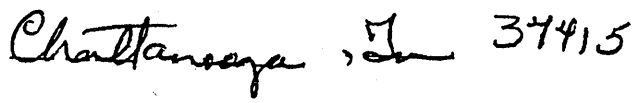


(b)

Fig. 17. Steps in processing a handwritten address (first example). (a) Gray-level image, (b) thresholded image.

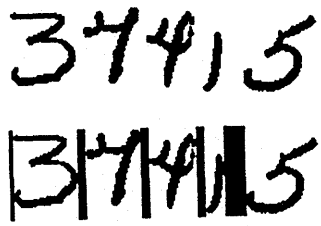


(c)



0.90	text	0.90	box	0.49	5-digits
		0.90	text	0.20	text

(d)



3	0.999
7	0.667
4	0.970
1	0.999
5	0.999

(e) (f)

Fig. 17. Cont'd. Steps in processing a handwritten address (first example). (c) Image with underlines removed, (d) bottom lines are segmented and matched to *city-state-ZIP* syntax, (e) ZIP Code candidate and its segmentation, (f) digit recognition results of ZIP Code candidate shown as pairs (recognized digit, confidence).

Fig. 17(f). The state name recognition was unsuccessful because the state name was cursively written. The recognized ZIP Code *37415* was checked against the USPS ZIP Code directory and found to be valid. The HZRS then returned *37415* as the answer.

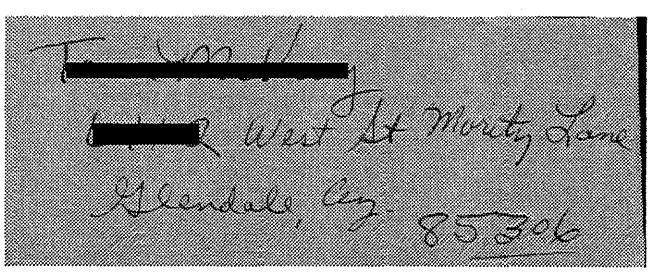
In Fig. 18, the processing to locate the correct ZIP Code is slightly more complex. The gray level image is shown in Fig. 18(a), and the thresholded image in Fig. 18(b).

Our underline removal algorithm was run, but since no underlining was found, no action was taken. Connected component analysis was performed to segment the address into text lines and words. As before, the personal name and street address were intentionally blacked out.

In this example, the first match to the *city-state-ZIP* syntax was incorrect. The HZRS determined that the match was incorrect and tried another syntax match. The first syntax match is shown in Fig. 18(c). We see that the state name includes part of the ZIP Code. The reason for this syntax selection is that the digits *5* and *3* in the ZIP Code touch, thus causing the digit estimator to be in error. Note that the touching digits are large compared to the other digits in the ZIP Code. Since the touching digits were not recognized during the connected component analysis (the only segmentation performed up to this point was inter-line segmentation), the digit estimator determined that this 2-digit connected component contained three digits. At the same time, we noted that the confidence assigned to the *5-digits* classification for the ZIP Code was only 0.24. However, the high classification confidences for city and state abbreviation matched this word grouping to the *city-state-ZIP* syntax.

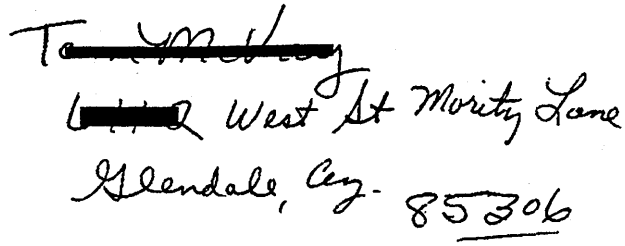
This ZIP Code candidate was segmented (Fig. 18(d)) and the digit recognition results are shown in Fig. 18(e). The segmentation results show the digit locations that required forced segmentation, i.e. in this example segmentation was required between the first three digits of the image. This ZIP Code candidate was rejected since one of the digits was not recognized, so the HZRS selected the next-best candidate.

The second-best match to the *city-state-ZIP* syntax is shown in Fig. 18(f). This was selected second since the last word *6-digits* was not a good match to an expected ZIP Code (no ZIP Codes have six digits). However, the system realized that digit estimation error might have occurred and allowed the syntax match to proceed. The ZIP Code candidate was selected and segmented (Fig. 18(g)). The digit recognition and segmentation results are shown in Fig. 18(h). The segmentation results show that forced segmentation was required between the second and third digits of the ZIP Code image. Still, the digit recognition confidence is high and the ZIP Code is valid compared to the USPS ZIP Code directory, so the system returned *85306* as the result.



(a)

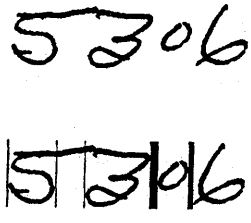
Fig. 18. Steps in processing a handwritten address (second example). (a) Gray-level image.



(b)

1.00	city	1.00	state abbrev.	0.33	text
0.67	text	0.50	text	0.24	5-digits
		0.13	5-digits		

(c)

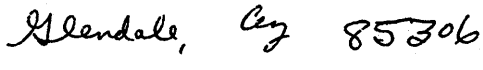


(d)

5 0.735
Reject
3 0.500
0 0.999
6 0.999

Segmentation resul
1 2 3

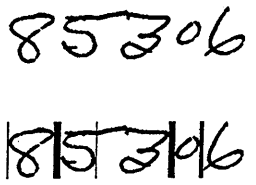
(e)



1.00	city	0.80	state abbrev.	0.25	text
0.67	text	0.67	text	0.24	6-digits

(f)

Fig. 18. Cont'd. Steps in processing a handwritten address (second example). (b) Thresholded image, (c) first match for segmented bottom line to *city-state-ZIP* syntax, (d) first ZIP Code candidate and its segmentation, (e) digit recognition results of first ZIP Code candidate shown as pairs (recognized digit, confidence), (f) second match for segmented bottom line to *city-state-ZIP* syntax.



(g)

8 0.999
5 0.970
3 0.500
0 0.999
6 0.999

Segmentation results
2 3

(h)

Fig. 18. Cont'd. Steps in processing a handwritten address (second example). (g) Second ZIP Code candidate and its segmentation, (h) digit recognition results of second ZIP Code candidate shown as pairs (recognized digit, confidence).

7. PERFORMANCE EVALUATION

The results shown in Table 2 describe the system performance as of November 29, 1990. As we have described, the HZRS research is ongoing and system performance is being improved on a (nearly) day-to-day basis.

The results (in Table 2) are from a set of 508 images chosen by the USPS from the Postal Address Image Database. For each image, the HZRS returned one of two responses: a ZIP Code, a ZIP + 4 Code or a rejection message. A rejection message indicated that the system did not find sufficient information to determine its encoding.

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 performance 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 means that the recognition system response matches the index entry ZIP Code or that the index entry indicates no ZIP Code is present and the recognition system returns a rejection message. A REJECT response means that the recognition system returns a rejection message and the index entry indicates 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.

A further analysis was done to determine the types of failure (both rejects and errors).

8. DISCUSSION

We have presented a systematic approach to extracting semantic information from

Table 2. System performance

Group	Code	Condition	Count	%
1	1.1	ZIP Code correctly read	388	76.38
(Correct)	1.2	No ZIP Code present	11	2.17
		Subtotal	399	78.54
2	2.1	Invalid ZIP Code correctly read	5	0.98
(Reject)	2.2	Rejection due to location or recognition	98	19.29
		Subtotal	103	20.28
3	3.1	Location failure (no ZIP Code)	0	0.00
(Error)	3.2	Location or recognition failure (with ZIP Code)	6	1.18
		Subtotal	6	1.18
		TOTAL	508	100.00

1.1 ZIP Code present and correctly read.
1.2 No ZIP Code present and image rejected.
2.1 ZIP Code present and correctly recognized, but ZIP Code is not in USPS directory of valid ZIP Codes.
2.2 ZIP Code present but image rejected due to location or recognition failure.
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 digits were misrecognized.

unconstrained handwritten text in limited domains. This approach uses bottom-up information to develop a global description and then uses the global description to suggest hypotheses which can be verified.

We have also shown how this approach is used in a system that determines ZIP Codes from handwritten postal addresses. On a test set of 508 address images, the system has a correct sort rate of 76.4% with an error rate of 1.2%. The system is an evolving one as efforts to improve its performance are continuing. We are actively developing more sophisticated handwritten character recognition algorithms and expect to add these to our HZRS. Additional character and word recognition information will allow us to use much more of the semantic information available in the addresses.

Our future research efforts will include extending our approach to other domains. A system developed for another domain would probably need different preprocessing steps to generate a binary image. Also, different features may be extracted, depending on the type of handwriting present in the image. Currently, the features we use are dashes, commas, disconnected 5-hats, and digit recognition. It is reasonable to assume that other features (such as character recognition results) may be used. For the address-reading application, we divide the text lines into 1–5 words. Other domains may expect different numbers of words in each line. Naturally, the syntax and semantics would need to reflect the new domain, but the framework used by our approach could direct control of their use.

ACKNOWLEDGEMENTS

The authors are deeply indebted to Carl O'Connor, Timothy Barnum and Gary Herring of the Technology Resource Department of the United States Postal Service

and John Tan of Arthur D. Little, Inc. for their guidance and support of this work. In addition, they would like to thank Alan Commike, Peter Cullen, Chih-Chau Kuan, and John Favata for contributing ideas and code towards portions of the HZR project.

REFERENCES

1. J. Alston and J. Taylor, *Handwriting: Theory, Research and Practice*, Nichols Publishing Company, New York, 1987.
2. C. Y. Suen, "Handwriting generation, perception, and recognition", *Acta Psychol.* **54** (1983) 295–312.
3. C. C. Tappert, C. Y. Suen and T. Wakahara, "On-line handwriting recognition—A survey", *Proc. 9th Int. Conf. on Pattern Recognition, Vol. 2*, 1988, pp. 1123–1132.
4. P. Hogarth, "Materials and tools", *Creative Ink Drawing*, Watson-Guptill Publications, New York, 1968, pp. 19–26.
5. M. A. Tinker, *Legibility of Print*, Iowa State University Press, 1986.
6. J. Rabinow, "The present state of the art in reading machines", *Pattern Recognition*, Ed. L. N. Kanal, Thompson Book Company, 1968, pp. 3–30.
7. S. Kahan, T. Pavlidis and H. S. Baird, "On the recognition of printed characters of any font and size", *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 2 (1987) 274–288.
8. M. A. Just and P. A. Carpenter, *The Psychology of Reading and Language Comprehension*, Allyn and Bacon, Inc., 1987.
9. S. N. Srihari, *Computer Text Recognition and Error Correction*, IEEE Computer Society Press, Silver Spring, Maryland, 1984.
10. H. S. Baird and K. Thompson, "Reading chess", *IEEE Trans. Pattern Anal. Mach. Intell.* **12**, 6 (1990) 552–559.
11. J. M. Brady and B. J. Wielinga, "Reading the writing on the wall", *Computer Vision Systems*, Eds. A. R. Hanson and E. Riseman, Academic Press, New York, 1978, pp. 283–299.
12. R. O. Duda and P. E. Hart, "Experiments in the recognition of hand-printed text: Part II context analysis", *AFIPS Conf. Proc.* **33**, 1968, pp. 1139–1149.
13. J. H. Munson, "Experiments in the recognition of hand-printed text: Part 1—Character recognition", *Proc. Fall Joint Computer Conference* **33**, 1968, pp. 1125–1136.
14. R. Bornat and J. M. Brady, "Using knowledge in the computer interpretation of handwritten FORTRAN coding sheets", *Int. J. Man-Mach. Stud.* **8**, 1 (1976) 13–27.
15. P. W. Palumbo and S. N. Srihari, "Text parsing using spatial information for recognizing addresses in mail pieces", *Proc. Int. Conf. Pattern Recognition*, Paris, France, Oct. 1986, pp. 1068–1070.
16. J. J. Hull, D. 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, Mar. 1988.
17. N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Trans. Syst. Man. Cybern.* **9**, 1 (1979) 63–66.
18. 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 Recogn.* **12**, 3 (1980) 189–199.
19. P. Ahmed and C. Y. Suen, "Computer recognition of totally unconstrained handwritten ZIP codes", *Int. J. Pattern Recogn. Artif. Intell.* **1**, 1 (1987) 1–15.

20. J. Schurmann, "A multifont word recognition system for postal address reading", *IEEE Trans. Comput.* **27**, 8 (1978) 721–732.
21. T. Pavlidis and F. Ali, "A hierarchical shape analyzer", *IEEE Trans. Pattern Anal. Mach. Intell.* **1**, 1 (1979) 2–9.
22. C. Kuan and S. N. Srihari, "A stroke-based approach to handwritten numeral recognition", *Proc. USPS Advanced Technology Conference*, Washington D.C., May 3–5, 1988, pp. 1033–1041.
23. D. D'Amato, L. Pintsov, H. Koay, D. Stone, J. Tan, K. Tuttle and D. Buck, "High speed pattern recognition system for alphanumeric handprinted characters," *Proc. IEEE-CS Conf. Pattern Recog. Image Proc.*, Las Vegas, 1982, pp. 165–170.
24. J. Favata and S. N. Srihari, "Recognition of Cursive Words for Address Reading", *Fourth Advanced Technology Conference*, Washington, D.C., Nov. 1990.

APPENDIX. POSTAL ADDRESS IMAGE DATABASE

In order to develop a practical system it is necessary to have a representative database of images. For this purpose a database of digitized images of handwritten addresses was collected by our group. The database consists of more than 2500 images of complete handwritten address blocks digitized at 300 pixels per inch and with eight bits of gray scale.

To carry this out, we assembled an image collection station that consisted of an Eikonix model 850 CCD image digitizing camera, a SUN 3/260G gray-scale workstation, a 689 megabyte disk, and a half-inch nine-track tape drive. This was set up at the United States Postal Service Sectional Center Facility in Buffalo, New York from February to April 1987. Some examples of these addresses appear in other parts of this paper.

A designed sampling procedure was used to choose most of the other pieces of mail that were digitized. The objective of this procedure was to gather a database that contained many cases that would be difficult to recognize by a simple approach and would still contain a geographic distribution of addresses. The database contains 200 images from each of the ten ZIP Code zones. The images in this category were divided up so that each state within a zone contributed the same number of pieces. The number taken from each state is shown in Table A1.

We also took images of approximately 20 pieces of mail from each of 25 different cities. Ten of these images were cursive and ten were handwritten. Approximately ten of the pieces from each city had hand-printed addresses and ten had cursive addresses. The cities, listed in Table A2, were chosen to provide a cross section from the entire country.

Altogether, there are 2636 images in the designed sample set. The 136 extra images include cases without ZIP Codes, foreign mail, and so on. These were taken to aid in the development of algorithms to handle these cases as well as good addresses. We also digitized 120 images of machine-printed addresses. These were randomly chosen with no particular constraints on the destination address and are being used to develop a portion of our system that determines whether an address is machine-printed and recognizes the ZIP Code or just rejects the machine-printed piece.

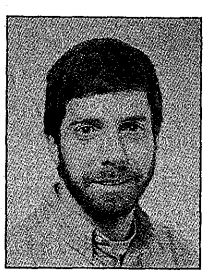
Table A1. Distribution of handwritten addresses by states.

State	No.	State	No.	State	No.
Alabama	63	Alaska	47	Arizona	31
Arkansas	57	California	39	Colorado	31
Connecticut	30	Delaware	69	D. C.	37
Florida	41	Georgia	33	Hawaii	45
Idaho	29	Illinois	38	Indiana	74
Iowa	33	Kansas	59	Kentucky	50
Louisiana	49	Maine	30	Maryland	52
Massachusetts	31	Michigan	40	Minnesota	33
Mississippi	39	Missouri	63	Montana	33
Nebraska	51	Nevada	33	New Hampshire	30
New Jersey	30	New Mexico	32	New York	66
North Carolina	31	Ohio	47	Oklahoma	50
Oregon	39	Pennsylvania	70	Rhode Island	32
South Carolina	32	South Dakota	32	Tennessee	40
Texas	47	Utah	34	Vermont	32
Virginia	39	Washington	46	West Verginia	36
Wisconsin	33	Wyoming	29	North Dakota	32
Puerto Rico	19	Unknown	1		

Table A2. Cities for which 20 samples were taken.

Atlanta, Ga.	Baltimore, Md.	Boston, Ma.
Boulder, Co.	Buffalo, N.Y.	Charlotte, N.C.
Cleveland, Oh.	Dallas, Tx.	Des Moines, Ia.
El Paso, Tx.	Ft. Lauderdale, Fl.	Hartford, Ct.
Indianapolis, In.	Kansas City, Mo.	Las Vegas, Ne.
Los Angeles, Ca.	Minneapolis, Mn.	New Orleans, La.
New York, N.Y.	Philadelphia, Pa.	Portland, Me.
San Francisco, Ca.	Sioux Falls, S.D.	St. Louis, Mo.
St. Paul, Mn.		

Received 15 December 1990.



Edward Cohen received his B.S. in electrical engineering from Cornell University in 1982 and his M.S. in computer science from the State University of New York (SUNY) at Buffalo in 1988. He is currently pursuing a Ph.D. in computer science at SUNY Buffalo. Prior to entering graduate school, he worked at RCA Technology Transfer Laboratories, where he obtained three patents on image processing techniques. Since 1987, he has been a research assistant in the CEDAR (Center of Excellence for Document Analysis and Recognition) group at SUNY Buffalo, working on USPS (United States Postal Service) related projects. His research interests include computer vision, image processing, parallel algorithms, and knowledge-based systems.

Jonathan J. Hull is a Research Assistant Professor in the Department of Computer Science at the State University of New York (SUNY) at Buffalo. He received a B.A. degree in computer science and statistics in 1980, an M.S. degree in computer science in 1983, and a Ph.D. degree in computer science in 1988, all from SUNY Buffalo. He is presently the Associate Director of the Center of Excellence for Document Analysis and Recognition at SUNY Buffalo and has technical responsibility on a project that is developing the next generation address recognition unit for the United States Postal Service.

Sargur N. Srihari is Professor of Computer Science and Director of the Center of Excellence for Document Analysis and Recognition. The center's work is in developing methodologies, algorithms, software, and hardware with a particular focus on reading machines. The goal of such machines is to read text from images of documents containing ordinary handwriting and even poor quality machine-printing. The center, largely funded by the USPS, has over 50 members including graduate students, undergraduate students, full-time research scientists and technical and secretarial staff.

Dr. Srihari has been successively Assistant, Associate and full Professor at the State University of New York at Buffalo since 1978. He was Acting Chairman of the Computer Science Department from 1987–88. He teaches in several areas of computer science, including pattern recognition and artificial intelligence. He has supervised six completed doctoral dissertations. He is author of over 125 papers and an IEEE tutorial on Computer Text Recognition and Error Correction.

Dr. Srihari received a B.Sc. degree in physics and mathematics from Bangalore University in 1967, a B.Engg. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, in 1970, and M.S. and Ph.D. degrees in computer and information science from the Ohio State University, Columbus, in 1971 and 1976 respectively. He is a senior member of IEEE, a member of the American Association for Artificial Intelligence, the Pattern Recognition Society and the Association for Computing Machinery. He received a New York State/United University Professions Excellence Award for 1991.

World Scientific Series in Computer Science – Vol. 30

CHARACTER & HANDWRITING RECOGNITION

Expanding Frontiers

Editor

P S P Wang

College of Computer Science
Northeastern University
Boston, MA 02115, USA

With Foreword by C Y Suen

World Scientific
Singapore • New Jersey • London • Hong Kong