A fuzzy logic system for the detection and recognition of street number fields on handwritten postal addresses

Article · January 1995

CITATIONS
13

READS
76

3 authors, including:

Paul Gader
University of Florida
402 PUBLICATIONS **8,891** CITATIONS

SEE PROFILE

A Fuzzy Logic System for the Detection and Recognition of Handwritten Street Numbers

Paul Gader, *Member, IEEE*, James M. Keller, *Senior Member, IEEE*, and Juliet Cai, *Member, IEEE*

***Abstract*—Fuzzy logic is applied to the problem of locating and reading street numbers in digital images of handwritten mail. A fuzzy rule-based system is defined that uses uncertain information provided by image processing and neural network-based character recognition modules to generate multiple hypotheses with associated confidence values for the location of the street number in an image of a handwritten address. The results of a blind test of the resultant system are presented to demonstrate the value of this new approach. The results are compared to those obtained using a neural network trained with backpropagation. The fuzzy logic system achieved higher performance rates.**

I. Introduction

HANDWRITTEN address interpretation by a computer system is important for automatic mail processing. According to the United States Postal Service, approximately 15% of all addresses are handwritten. Almost all handwritten letter mail, approximately 25 billion pieces per year, must be processed by costly semiautomatic or manual processes [1]–[5].

Developing an effective address interpretation system is a very challenging task due to the large amount of variability and uncertainty present in handwritten addresses. The numeric fields in an address, i.e., the street number and the ZIP code, can play a crucial role in reducing the complexity of the address interpretation task [1]-[5]. If these numeric fields are correctly detected and identified, then the number of possible addresses is significantly reduced. The numeric fields supply context for higher level interpretation to reduce both the uncertainty and the ambiguity in the address.

Fig. 1 shows a common control structure for a handwritten address interpretation system. At the preprocessing level, the written matter from the address block image must be extracted from the background and noise, and each text line located and separated into image blocks. Each image block should contain at least one word from the address [3]. The search for numeric fields is confined to searching either the first image block on a subset of the address lines (for street numbers) or the last image block on a subset of the lines (for ZIP codes).

Fig. 2 displays typical address images with image blocks indicated. From these image blocks, candidates for ZIP code, street number and P.O. Box number are determined. The ZIP code and street number/P.O. Box number are then interpreted. In the case that a street number is present, the ZIP code and the street number are used to query a database to obtain a lexicon of street names. Based on these names, word recognition and verification is used to generate hypotheses for the entire address which are matched against full records in the postal database to determine the delivery point of the mail [3].

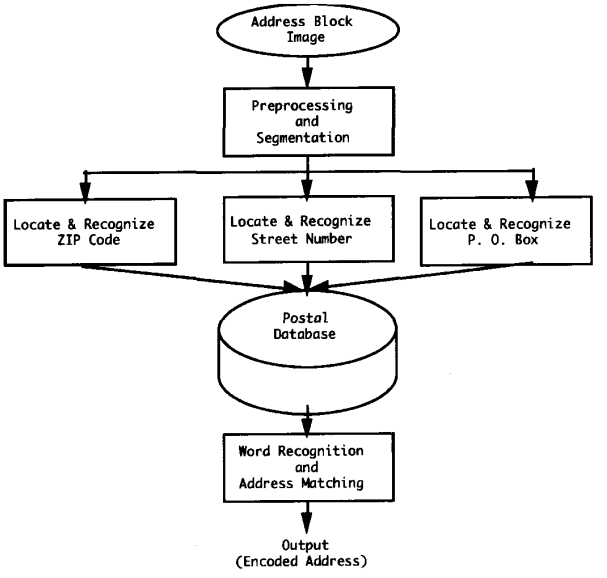

Fig. 1. Control structure for a typical handwritten address interpretation system.

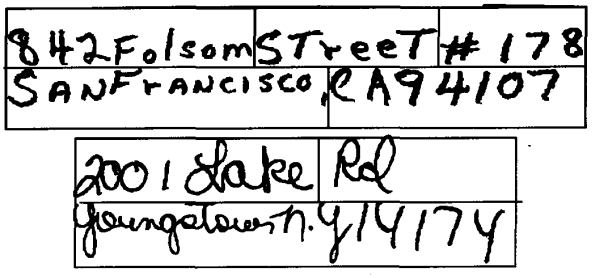

Fig. 2. Typical address images and image blocks.

Word recognition by computer is lexicon dependent; there is no program capable of reading unconstrained handwriting from static images without using a lexicon. Word recognition algorithms are able to achieve recognition rates from 85% to 95% using lexicons of size 1000 or less [5]–[8]. There are well over 100 000 unique street names in the United States,

Manuscript received August 26, 1993; revised August 19, 1994.

P. Gader and J. M. Keller are with the Electrical and Computer Engineering Department, University of Missouri-Columbia, Columbia, MO 65211 USA.
J. Cai is with Black and Veatch in Overland Park, KS 66211 USA.

IEEE Log Number 9406650.

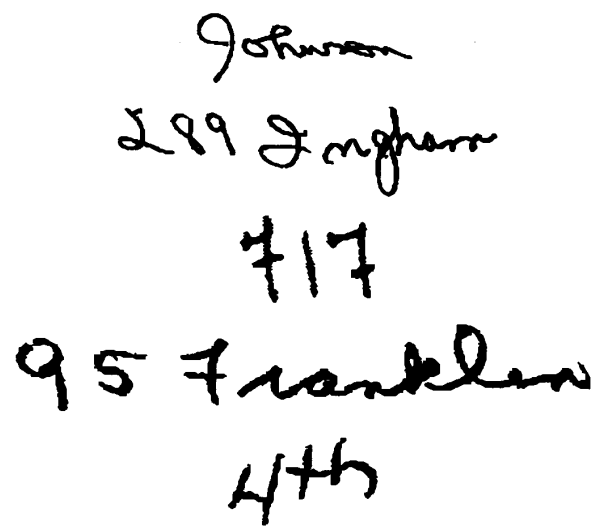

Fig. 3. Examples of confusing street numbers, a "J" that looks like a "9," an "I" that looks like a "2," "7's" that look like "F's", an "F" that looks like a "7," and a "4" that looks like an "H."

however, so an algorithm cannot be expected to read the street name directly without reducing the number of possibilities. This can be done by locating and reading ZIP codes and street numbers.

Correct location and interpretation of the street number field are crucial to the performance of the handwritten address interpretation system. Locating and reading the street numbers in addresses is difficult. There are a variable number of digits in a street number and there is ambiguity between certain numerals and alphabetic characters. For example, the "J" in the word "Johnson" in Fig. 3 can be easily misidentified as a "9" and the "o" as a "0" (zero). The ambiguity can be resolved by reading the word but as mentioned previously, without a lexicon, this is currently beyond the capabilities of any program. Segmentation of touching characters is also a problem as shown in Fig. 2. Other examples are shown in Fig. 3.

Character recognition is usually used in both numeric field recognition and word recognition. There are usually several character recognition modules working in parallel. These may include probabilistic methods, deterministic algorithms, neural networks, and fuzzy set methods to generate confidences for each upper case character class, lower case character class, and digit class [3]–[13]. These algorithms rely on numerous features which can be extracted from the current handwritten segment, all of which contain considerable uncertainty. Hence, developing a handwritten address interpretation system requires methods to analyze and manage the uncertainty present in the many parts of image processing: segmentation of an image of handwriting into characters or subcharacter components, feature extraction, and recognition modules.

In this paper, we present the incorporation of fuzzy logic into the process of detecting and recognizing street number fields in handwritten addresses. We show how fuzzy logic provides a natural mechanism to increase or decrease confidence in "street number" as a handwritten image block is processed by the computer vision modules. The fuzzy logic system uses the uncertain results obtained from segmentation and recognition modules to form hypotheses concerning the locations of street numbers in addresses. The focus of this paper is on the use of fuzzy logic; the segmentation and recognition modules have been independently described elsewhere [7]–[13]. The rules and membership functions were initially designed based on our experience and were modified using the results obtained on training images. The resultant system was then tested with several data sets taken from the real address images.

Our approach represents an innovative combination of fuzzy logic with computer vision. It is an application of fuzzy logic that differs from the traditional ones. This application convincingly demonstrates that fuzzy logic can be useful as a tool in high-level vision. It is convincing because it was performed on a real application with real data and the fuzzy rule base used input from a very complicated set of image processing, neural network, and dynamic programming algorithms that took several years to develop. There is value in this research, both for researchers in handwriting recognition and in fuzzy logic. The value to researchers in handwriting recognition is clear; we have demonstrated that they can use fuzzy logic to help solve their problems. The value to researchers in applications of fuzzy logic is that it provides a template for integrating fuzzy logic rule bases into complex decision making environments.

For comparison, we also trained a neural network to assign confidences concerning the locations of street numbers in addresses. We used the same training and testing sets and the same numerical features. The neural network performance did not perform as well as the fuzzy logic system, although it did not perform badly. There are several possible interpretations of this result: 1) The knowledge required to assign street number location confidence is not at the microstructure level: it is more coarsely grained and therefore more suitable for fuzzy logic, 2) There is not enough training data for the neural network. A fuzzy rule-based approach can incorporate human knowledge that is not statistically well represented in the data. Actually, 1) and 2) are not disjoint. It is difficult to incorporate human knowledge into a problem such as character recognition since the information (the shape of the characters) is on such a fine level, c.f. [14], [15]. On the other hand, it is relatively easy to define a neural network to perform character recognition and the results are usually very good. It is very natural to think of rules for locating street numbers (if one thinks about such things) and, as our results show, it is not so easy to train a network that will perform as well as a fuzzy logic system.

Some researchers have reported preliminary results on street number location and recognition in the context of overall address interpretation [3], [5]. There are more results reported on the related problem of ZIP code location and recognition, some of the most recent being [3]–[5]. Locating the ZIP code has some constraints that make the problem easier than street number location (although it is still quite difficult). There are either five or nine digits in a ZIP, whereas the number of digits in a street number is unknown and there is less variability in the line in which the ZIP occurs since it is supposed to be

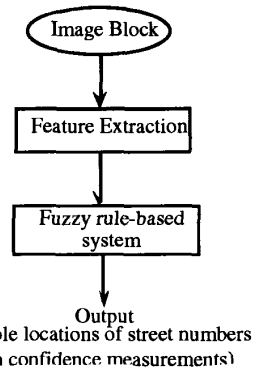

Fig. 4. Street number location system process diagram.

on the last line in the address. Our results, given in Section IV, were high enough to indicate this approach can lead to a module that could be successfully embedded in a handwritten address interpretation system.

II. Street Number Field Location and Recognition

The street number location system takes an image block as input. An image block is a subimage segmented from an address line. The output of the system is a set of possible locations of the street number field. A confidence value and recognition result is associated with each possible location. The recognition result represents the system's best guess at the identity of the street number. The street number field location system is comprised of two major components, a feature extraction module and a fuzzy rule-based system. Fig. 4 illustrates the overall process.

In feature extraction, numerical features are extracted from the preprocessed image blocks using image processing, neural networks, and dynamic programming. These features are then used as inputs to the fuzzy rule-based system, which generates multiple hypotheses for street number field location. The following sections describe each component of the street number location system in detail.

A. Feature Extraction

Feature extraction includes three major components: segmentation, neural network recognition modules, and dynamic programming. The overall control structure is shown in Fig. 5. During segmentation, each image block is divided into primitives; each primitive is a subimage of an image block that consists of a character or a part of a character. Several neural networks are activated to obtain character and digit confidence measurements on the primitives and unions of primitives. Dynamic programming is used to obtain numeric field recognition confidence measurements [7], [10]–[13]. It is also used to determine the confidence that the image block contains the string "PO," a part of "P.O. Box." The following sections describe each of the three feature extraction processes.

Segmentation: The segmentation algorithm is essentially the same as the one described in [7], [8], [10], [13]. We describe it briefly here. The segmentation process initially computes connected components and removes punctuation.

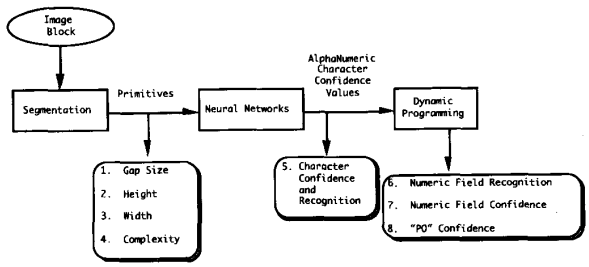

Fig. 5. Control structure for the feature extraction system.

Using simple grouping, for example grouping a multistroke "T," initial segments are generated. The initial segments consist of images of one or more connected components. The initial segments are ordered from left to right. An initial segment is said to be too complex to segment if the number of transitions from foreground to background along any horizontal line exceeds a threshold.

We process the initial segments from left to right until either there are no initial segments left or one is found to be too complex to segment. If an initial segment is not too complex to segment, then it is processed by a splitting module. The splitting module uses a distance transform to detect possible locations to split initial segments that may contain more than one character. It splits an initial segment into a sequence of smaller segments called the primitive segments associated with the initial segment. If an initial segment is too complex to segment, then the image consisting of the first 64 columns of the initial segment is created and is the primitive segment associated with that initial segment. The result of splitting and initial segmentation is a sequence of subimages of the original image block. These subimages are the primitives. Fig. 6 shows the segmentation results from three typical image blocks.

The top of each pair in Fig. 6 is the original image block and the bottom is the primitives. These results illustrate the necessity for segmentation and also the uncertainty introduced by the splitting process. The "A" in the first pair is segmented into primitives that look like "14." The "3" (or is it a lowercase cursive "z"?) in the second pair is segmented into primitives that look like "22." The last image block is segmented into primitives that could be read as "LItt*" where * represents illegible characters.

Features obtained from the segmentation include the height and the width of each primitive and the gap size between primitives. The gap size is measured by subtracting the largest x coordinate value of pixels in the left primitive from the smallest x coordinate value of pixels in the right primitive. If two primitives overlap, the gap size is set to zero.

Character Recognition Neural Networks: There are six neural networks used in the confidence assignment: two for numerals (0–9), and four for alphabetic characters. Two types of feature vectors are used as inputs to the neural networks, the transition feature vector, and the bar feature vector. The bar features are completely described in [7], [9], [12]. The transition features are completely described in [7], [13]. The neural networks were trained using backpropagation and use class-coded outputs. They also contain a class named

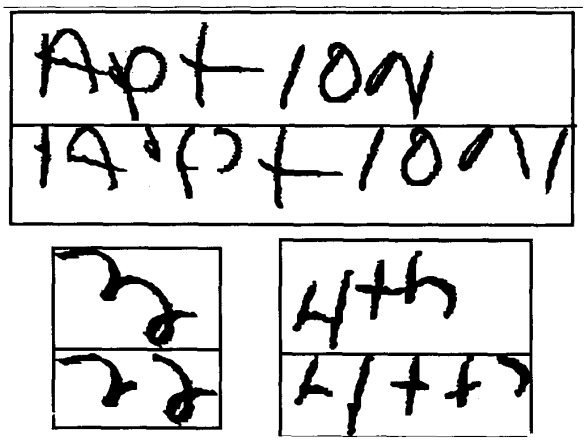

Fig. 6. Some automatically extracted, segmented image blocks and their primitive segments.

"garbage" to account for segments which do not represent any character images, such as multiple characters or pieces of characters.

The idea of the transition features is to count the locations and numbers of transitions from white pixels to black pixels along horizontal and vertical lines. The transition calculation is performed from right to left, left to right, top to bottom and bottom to top. The information is encoded as a feature vector with 100 elements. Three neural networks for the confidence measurements use transition feature vectors, one each for upper and lower case alphabetic characters, and one for digit recognition.

The bar features encode directional information from the foreground and the background. First, up to eight feature images are generated, each corresponding to one of the directions in east, northeast, north and northwest, in either the foreground or the background. Each feature image has an integer value at each location that represent the length of the longest bar which can fit at that point in that direction. For each of the eight feature images, 15 different subimage zones are created. The values in these zones are summed and normalized between zero and one. The result is a feature vector of size 120. Three neural networks for the confidence measurements use bar feature vectors, one each for upper and lower case alphabetic characters and one for digit recognition.

Primitives often contain only parts of characters. To obtain confidence measurements on characters, subimages of pairs and triples of the primitives are also used to obtain character confidence assignments using the neural networks. Hence, there are six character confidence readings and measurements at the end of each primitive, each corresponding to a single primitive, a pair or a triplet of primitives, in either upper or lower case characters. The maximum of these six confidence measurements is used as the character confidence feature for the fuzzy rule-based system. Moreover, the number and character confidence measurements and readings on primitives are also used in the dynamic programming to obtain the numeric field confidence value and recognition results, and word recognition confidence value on the string "PO."

Dynamic Programming: We use dynamic programming for word recognition and numeric field recognition. The dynamic programming algorithms find the best match between sequences of primitives and sequences of characters or digits. The best match depends upon the character or digit confidence for each of the primitives obtained from the neural networks.

The dynamic programming word recognition algorithm is a module that takes an image of a handwritten word, a string, and a list of the primitives from the image as input and returns a value between zero and one indicating the confidence that the word represents the string. The algorithm is completely described in [7], [10], [13]. We use this algorithm obtain confidence that a sequence of primitives represents the string "PO."

The dynamic programming numeric field recognition algorithm is almost the same as the word recognition algorithm. In the case of numeric field recognition, no string is used as input. The output is a string of digits with a confidence value. The confidence value represents the confidence that the image is a numeric field and the string represents the system's best guess at the identity of the field.

The basic dynamic programming algorithms produce one set of outputs for each subsequence of primitives $P_k = \{p_1, p_2, \cdots, p_k\}$, $k \leq n$, where n is number of primitives in an image block. The output consists of the confidence that P_k represents a numeric field, the hypothesized identity of the numeric field, and the confidence that the subimage block contains the character string "PO." These three features are subsequently used by the fuzzy rule-based system. The street number location neural network uses the first and third features.

We describe the dynamic programming algorithm in detail for the case of numeric field recognition. The input to the algorithm is a sequence of primitives, $P_k = \{p_1, p_2, \cdots, p_k\}$. The output is the confidence that the sequence represents a numeric field and the hypothesized identity of the numeric field. The numeric field that is produced as output can be of any length between 1 and k. For example, if $k = 3$ then the digit strings "1," "53," and "864" of length 1, 2, and 3, respectively, are all possible hypothesized identities for the sequence.

The algorithm constructs a $k \times k$ array called dparray. The columns of the array correspond to primitive segments. The rows of the array correspond to digits in the hypothesized identity. The (i, j) element of the array is the value of the best match between all possible digit strings of length i and the first j primitives. The match values come from the digit recognition neural networks. We match sequences of different lengths by forming unions of primitive segments as illustrated in Fig. 7. For each pair of integers i, j between 1 and k with $i \leq j$ we let $p_{ij} = \cup_{h=i}^{j} P_h$ denote the union of primitives i through j.

We define a function, match(s), that takes a segment s (either a primitive or a union of primitives) as input and computes the highest digit recognition confidence for any class. That is, if $\text{bf}(s, i)$ and $\text{tf}\ (s, i)$ denote the output activation values for the bar and transition feature digit recognition networks for classes $i = 0, 1, \cdots, 9$ using the appropriate features computed from

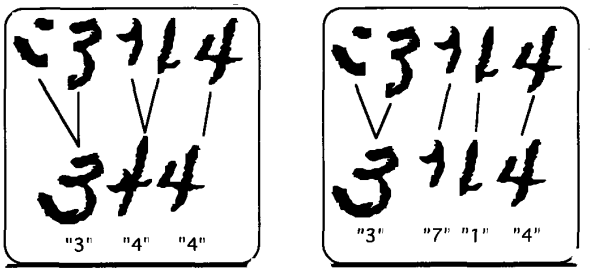

Fig. 7. Matching a sequence of primitive segments to digit strings of different lengths using different unions of primitives.

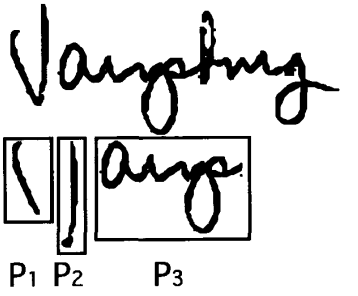

Fig. 8. An image block and the primitive segments, p_1, p_2, and p_3, used for street number location. The system should reject this image block.

input image s, then

$$\text{match}(s) = \max\{\text{bf}(s,i) + \text{tf}(s,i) \mid i = 0, 1, \cdots, 9\}.$$

Let legalp(s) be a Boolean function that returns TRUE if s is not too complex to segment and FALSE otherwise. The elements of dparray are first initialized to $-\infty$. The (i,j) element of dparray is then computed as follows

```
IF i = 1 (matching against the first digit
in the string) THEN
   IF legalp (pij) THEN
      dparray(1, j) = match(p1j)
ELSEIF legalp(pij) THEN
   dparray(i,j) =MAX{dparray(i-1,k)
   + match(pij)|
   i ≤ k < j and legalp(pkj)}.
```

Once the values of the of dparray have been computed, the best match is found by computing the maximum value in column k (the last column) after normalizing by length. That is, the output confidence is given by

$$\text{num\_field\_conf} = \underset{i}{\text{MAX}}\{\text{dparray}(i,k)/i \mid 1 \le i < k\}.$$

This formula produces the best match since dparray$(i,k)/i$ is the value of the best match between all digit strings of length i and all the input primitives. Note that we must also keep track of the classes for which the best matches occur if we want to return the highest confidence digit string.

Summary of Features: For each subsequence of primitives of the form p_1, p_2, $\cdots$, p_k, $k \le n$, where n is number of primitives in an image block, the following features are calculated:

1) *gap size*: the gap between p_k and $p_{(k+1)}$;
2) *height*: the height of p_k;
3) *width*: the width of p_k;
4) *too_comp_to_recog*: the flag indicating that p_k was extracted from a connected component that was too complex to be recognized as a digit;
5) *character confidence 1*: confidence that primitive p_k is an upper case character;
6) *character confidence 2*: confidence that primitive p_k is a lower case character;
7) *character confidence 3*: confidence that the union of primitives, $p_{(k-1)} \cup p_k$, is an upper case character;
8) *character confidence 4*: confidence that the union of primitives, $p_{(k-1)} \cup p_k$, is a lower case character;
9) *character confidence 5*: confidence that the union of primitives, $p_{(k-2)} \cup p_{(k-1)} \cup p_k$, is an upper case character;
10) *character confidence 6*: confidence that the union of primitives, $p_{(k-2)} \cup p_{(k-1)} \cup p_k$, is a lower case character;
11) *numeric field confidence*: confidence value that the image $p_1 \cup p_2 \cup \cdots \cup p_k$ is a numeric field (recognition results are also produced);
12) *"po" confidence*: confidence that the image $p_1 \cup p_2 \cup \cdots \cup p_k$ represents the string "**PO**."

For example the feature values extracted by the various submodules for the primitive segments in Fig. 8 are

		p_1	p_2	p_3
1)	gap size:	0	1	0 (pixels)
2)	height:	34	49	43 (pixels)
3)	width:	14	9	66 (pixels)
4)	too_comp_to_recog:	0	0	1
5)	character confidence 1:	"garbage"	"garbage"	"N"/0.382
6)	character confidence 2:	"l"/0.703	"j"/0.778	"garbage"
7)	character confidence 3:	N/A	"V"/0.865	"garbage"
8)	character confidence 4:	N/A	"v"/0.679	"garbage"
9)	character confidence 5:	N/A	N/A	"garbage"
10)	character confidence 6:	N/A	N/A	"garbage"
11)	numeric field confidence:	"1"/0.261	"4"/0.365	0.000
12)	"po" confidence:	0.0	0.104	0.0

In items 5)–12) the entries represent the character or digit class that was assigned the highest confidence by the neural networks and the confidence value itself. The entry "garbage" indicates that the neural networks assigned the highest confidence to the "garbage" class. These features are normalized as described in Section II–B1 before being used in the fuzzy rule base and neural network systems for street number location.

B. Fuzzy Rule-Based System

Fuzzy logic represents a form of approximate reasoning whereby propositions containing imprecise or vague terms are modeled by possibility theory [16]–[18] and more general forms of inference are utilized [19]. The fuzzy rule base developed for the street number location system contains 48 rules and is implemented using the software package, CubiCalc [21], which utilizes correlation-min inference [21]. The input to the fuzzy rule base is a set of features associated with a sequence of primitives $p_1, p_2, \cdots, p_k$. The output of the rule

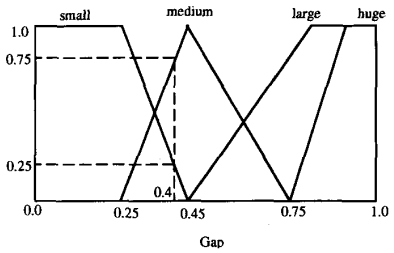

Fig. 9. Membership functions for variable "gap."

base is a number representing the confidence that the subimage block $p_1 \cup p_2 \cup \cdots \cup p_k$ is a street number field. In addition, three assumptions are made in the development of the street number location rule base:

1) A street number always starts from the beginning of an address line; thus it is always contained in the first image block segmented from the line image;
2) No street number starts with a zero;
3) No street number is longer than 5 digits.

All rules and membership functions used in our system are provided in Appendix A and in [22].

Input/Output Variable Definition: There are eight input variables and one output variable associated with each subimage block $p_1 \cup p_2 \cup \cdots p_k$. The input variables are gap_k, height_k, width_k, too_comp_to_recog (complex_k), character confidence reading (conf_k), numeric field reading, numeric field confidence measurement (n_k), and "PO" confidence (po_k), while the output variable (the street number confidence) is called adjconf. Each variable has several adjectives, also called the term set. Each adjective is represented by a membership function. Each membership function maps the value of the variable to a membership value of that adjective. For instance, one of the input variables is "gap," the normalized spatial distance between two consecutive primitives. The adjectives used to describe gap are " huge," "large," "medium," and "small." The membership functions for these adjectives are represented in Fig. 9.

All the input variables for the fuzzy rule-based system are shown in Table I. In addition to the variables in Table I, a variable po-set$_k$ = MAX{ confidence that $p_1 \cup p_2 \cup \cdots p_j$ represents the string "PO" $\mid j \leq k\}$ is computed internally and used by the system.

The output variable is "adjconf," which indicates the adjustment of the confidence value for a street number field. It is initialized to zero at beginning of each image block. The range of "adjconf" falls between -1.0 and 1.0. The final output is calculated as $0.5 + (1/2)$*adjconf, so that the result is mapped between 0 and 1. The adjectives for adjconf are "neglarge," "negsmall," "negtiny," "nearzero," "postiny," "possmall," and "poslarge," similar to those adjectives in control applications.

Rule Description: There are 48 rules used in this fuzzy rule-based system. Each rule consists of one or more antecedents and one consequence. An example is the rule

IF complex_n is large AND n is large AND gap is medium
THEN adjconf is poslarge;

TABLE I
INPUT VARIABLE DEFINITIONS INPUT VARIABLE

Input variable	Description	Value range	Adjectives
gap_k	The distance between p_k and p_{k+1} normalized by: $C = \frac{1}{p+q}\left(\sum_{i=1}^{p} \text{width}_i + \sum_{\left[\text{gap}_i>=5\right]}^{p} \text{gap}_i\right)$ p is the number of primitives in the image block, q is the number of gaps with size no less than 5 pixels, width_i is the width of the ith primitive, and gap_i is the distance in pixels between the ith and (i+1)th primitive as described in section 2.1.1. *It is set to 1 if the* normalized result is larger than 1.	[0, 1]	{huge, large, medium, *small*}
height_k	The height of the primitive p_k divided by the fixed height of an address line, 64 pixels.	[0, 1]	{large, medium, small, tiny}
width_k	The width of subimage $p_1 \cup p_2 \cup ... p_k$ divided by 420.	[0, 1]	{large, medium, small}
complex_k	The flag set for primitives that are too complex to be *digits. It is a fuzzy variable* that has only two adjectives, high and small, corresponding to 1 and 0.	[0, 1]	{large, small}
conf_k character confidence	The maximum value of the character confidence i, i =1 . . . 6, as given in section 2.2.4. The class that corresponds to the value is also used.	[0, 1]	{large, medium, small}
numeric *field* reading k	The highest confidence digit string returned by the *dynamic programming* algorithm given the primitives $p_1, p_2, \ldots, p_k$ as input.	N/A	N/A
n_k	Confidence that $p_1 \cup p_2 \cup ... \cup p_k$ is a numeric field.	[0, 1]	{large, medium, small}
"PO" confidence	The dynamic programming match value between p_1, $p_2, \ldots, p_k$ and the string "PO".	[0, 1]	{large, medium, small}

which can be interpreted as
IF the next primitive is too complex to be recognized as digits
AND the numeric field confidence, n, is large
AND the gap size between this primitive and next primitive, gap, is medium
THEN the street number confidence should be adjusted to positive and large.

A weighting factor preceding a rule alters the relative importance of rules that are simultaneously active. The weighting factor is applied to the result fuzzy set by scaling the set proportional to the weighting factor prior to the summation process. Weighting values span the range zero through one.

A confidence value is produced for each subimage block, indicating the degree of confidence that the subimage block is a street number field. The final procedure of the street number location system is thresholding of the final confidence. Locations that have confidence values above a threshold are interpreted as ending positions of street number fields. Image blocks for which all confidence values are below the threshold are interpreted as non-street number blocks.

III. EXPERIMENTAL APPROACH

A. Project Data Set

The data used for this project were binary image blocks, supplied by the Environmental Research Institute of Michigan

(ERIM). The image blocks were selected to be the first word block in each address line except the first and last lines in an address block. Initially, 71 image blocks were used for training, with 41 having street number fields. Another 78 image blocks were used as an initial test set; 40 of them contained street number fields. These images were then combined into a single training set that was used to train the final system. A blind test of the final system was conducted using another 155 image blocks containing 79 street number fields. The training and initial test sets are referred to as train1 and test1. The combination of the train1 and test1 is referred to as btrain, and the blind test set is referred to as btest.

B. Procedure

The development of the fuzzy rule base followed the usual development path of rule based systems: an iterative cycle of rule definition, testing, and rule refinement. The rules in the fuzzy rule base were initially written based on pictures of address blocks [23]. The system was then trained with the train1 set. The training process was iterated until the results were satisfactory. Following each training cycle, the system was adjusted based on the analysis of the results, especially the errors. System adjustment methods are discussed in this section.

Upon completion of the training, initial testing was performed with the test1 set. A few adjustments were then performed based on the test results. Additional rules were added to the rule base, and a few rules were changed. The final system was applied to the btest set one time, constituting the blind test. Final decisions concerning the location of the street numbers were made by thresholding the output confidence. Those locations for which the confidence was above the threshold were labeled as locations of street numbers by the system. The threshold used on the blind test set, btest, was selected by applying the algorithm described in this section to the training set, btrain. The system was scored using the number of street number locations assigned by the system that were true street locations. The recognition rates achieved by the system were also scored. The results of training and testing are described in Section IV.

System Adjustment: There are number of ways to adjust fuzzy rule-based systems. One is adding new rules. For example, the two rules in the rule base specifically for "P.O. Box" were added after observing that the string "PO" was often misidentified as "90." Other adjustments for individual rules include changing the adjectives that are used in a rule. The effects of adjective adjustments are limited because they are only effective when the rules are activated. In addition, the effectiveness of a rule can be adjusted by its weight. The relative importance of the rule can be reflected by their weights. Finally, changing the shapes of membership functions can dramatically change output results. Such adjustments affect overall performance of the system. It is a technique usually applied when the values of a variable are generally shifting up or down.

Our approach to improving the system was by trial and error and iterative improvement. The system adjustments that were performed were based on the judgement of the human algorithm developers. After each trial of the system, the errors were analyzed, adjustments were identified to correct for the errors, and the system was tried again. This approach can lead to a nonrobust system; our results on blind test data suggest that if care is given not to make rules too specific, then a robust system can be developed.

Threshold Selection Algorithm: A simple algorithm was used to obtain the best threshold for the training sets. The best threshold was defined as the threshold that produced the best success rate using the top three numeric field location hypotheses with the highest confidence values. The algorithm looped through a quantized number of possible thresholds and found the one which gave the highest success rate on the training set. That threshold was then used for the blind tests.

IV. Experimental Results

A. Definition of Performance Measurements

This section defines all the terms used to evaluate the performance of the street number location system. They are: number of correctly located street numbers (L), number of incorrectly located street numbers (I), top n truth containment number $T(n)$, number of correctly identified nonstreet numbers (S), number of correctly identified (located and recognized) street numbers (R), success rate, top n truth containment rate, false positive number, recognition rate, and field recognition rate. Let N_s denote the number of image blocks with street numbers, N_n the number of image blocks without street numbers, and N the total number of image blocks, $N_s + N_n$. Definitions of these performance measures are given in Table II.

B. Results

The results have been segregated into several parts: the location results for the initial training and testing experiments, location results for the blind test, and numeric field recognition rates for the btrain and btest sets.

Initial Training and Testing Results: Initial training was conducted on the train1 set consisting of 71 image blocks, with 41 containing street number fields. Initial testing was conducted on the test1 set consisting of 78 image blocks, with 40 containing street number fields.

The best threshold automatically found using the training set was 0.535. For comparison, results for thresholds of 0.5 and 0.75 are also shown in the following tables. Table III compares the top 3 success rates and Table IV shows the truth containment rates.

Blind Testing Results: The blind test set, btest, consists of 155 image blocks, with 79 having street number fields. The union of train1 and test1 was used as the reference training set, btrain; the best threshold computed from btrain was used as the threshold for the blind test.

Fig. 10 shows the top 1 and top 3 truth containment rate, false positive, and top 3 success rate for the blind test results. The curves in Fig. 10 can be viewed as operating curves. The street number location module is useful as a submodule in a larger, address interpretation system. Therefore, the performance measure that is most important is driven by

TABLE II
DEFINITION OF PERFORMANCE MEASUREMENTS

Performance Measurement	Definition
number of correctly located street numbers, L:	The number of image blocks which contain the street number and for which the correct location of the street number is assigned the highest confidence value of all potential locations in that block and for which the highest confidence value is above the threshold.
truth containment number, T(n):	The number of image blocks which contain the street number and for which the correct location of the street number is assigned a confidence value above the threshold and the value is among the n highest confidence values from the block. T(1) is the same as L.
number of incorrectly located street numbers, I:	The number of image blocks which contain the street number for which the highest confidence value is above the threshold but is assigned to the wrong location.
number of correctly identified non street numbers, S:	The number of image blocks that do not contain any street number, and are identified as such by the system.
number of correctly identified street numbers, R:	The number of street number images that have been correctly located and recognized.
top n success rate (topn_suc_rate):	The total number of correctly located street number blocks and correctly identified non street number blocks, divided by the total number of image blocks: (T(n)+S)/N. Top 1 success rate is (L+S)/N and is referred to as the success rate.
top n truth containment rate:	This is calculated as: $T(n)/N_s$. Top 1 truth containment rate is L/N_s and is referred to as the truth containment rate.
false positive number (false_pos):	The number of non street number blocks that have been identified as street number blocks, i.e. the highest confidence value of the image block is above the threshold. It is calculated as: N_n-S.
recognition rate:	The number of street numbers that have been correctly located and read, divided by the total number of street number blocks: R/N_s.
field recognition rate:	The number of street number blocks that have been correctly located and read, divided by the number of correctly located street number blocks: R/L.

TABLE III
TOP n SUCCESS RATES FOR INITIAL TRAINING AND TESTING

	thresh=	0.5	thresh=	0.535	thresh=	0.75
n	train	test	train	test	train	test
1	85%	85%	93%	90%	77%	78%
2	86%	87%	94%	91%	77%	78%
3	86%	87%	94%	91%	77%	78%

TABLE IV
TOP n TRUTH CONTAINMENT RATES FOR INITIAL TRAINING AND TESTING

	thresh=	0.5	thresh=	0.535	thresh=	0.75
n	train	test	train	test	train	test
1	93%	93%	93%	88%	61%	60%
2	95%	98%	95%	90%	61%	60%
3	95%	98%	95%	90%	61%	60%

the way in which the submodule will be used in the larger system. For example, it may be very important for the address interpretation system that the submodule achieve high truth containment, that is, it may be important that the correct location is almost always one of the hypotheses. In this case, operating with a threshold level of 0.35 would be appropriate. On the other hand, it may be more desirable to operate at a level that maximizes the success rate, which implies that a threshold level of around 0.55 would be appropriate.

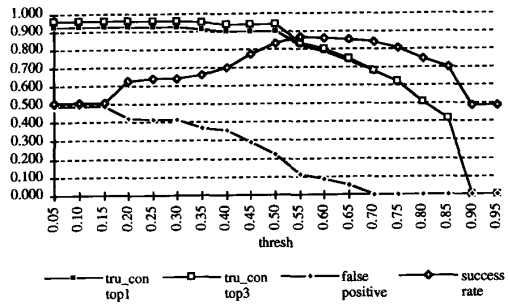

Fig. 10. Blind test results.

TABLE V
TOP n SUCCESS RATE FOR THE BLIND TEST

	thresh= 0.5		thresh= 0.535		thresh= 0.75	
n	train	test	train	test	train	test
1	85%	81%	91%	86%	77%	81%
2	87%	83%	93%	88%	77%	81%
3	88%	83%	93%	88%	77%	81%

TABLE VI
SHOWS TRUTH CONTAINMENT RATES

	thresh= 0.5		thresh= 0.535		thresh= 0.75	
rank	train	test	train	test	train	test
1	94%	90%	91%	87%	59%	62%
2	98%	95%	94%	90%	59%	62%
3	99%	95%	95%	90%	59%	62%

TABLE VII
NUMERIC FIELD RECOGNITION RATES

	R	L	avg_chr /field	N_s	R/L	R/N_s
btrain	56	74	3.08	81	76%	69%
btest	57	69	3.07	79	83%	72%

The best threshold obtained from the training was previously shown to be 0.535. Tables V and VI show the comparison between the combined training set and the blind testing results at thresholds of 0.5, 0.535, and 0.75. Table V shows success rates.

Table VI shows truth containment rates. The best threshold was then calculated for the blind test set, and was found to be 0.5355, a value only 0.0005 away from the primary threshold. This demonstrates the robustness of the original threshold selection procedure.

C. Recognition Rates

Table VII shows the recognition rates and associated information from the training set, btrain, and the blind test set, btest. The threshold used was the best threshold obtained from the training set, btrain. Avg_chr/field is the average number of digits in a street number field for all the correctly located street number image blocks. Field_rec_rat is the field recognition rate defined in Section IV-A, and rec_rate is the recognition rate.

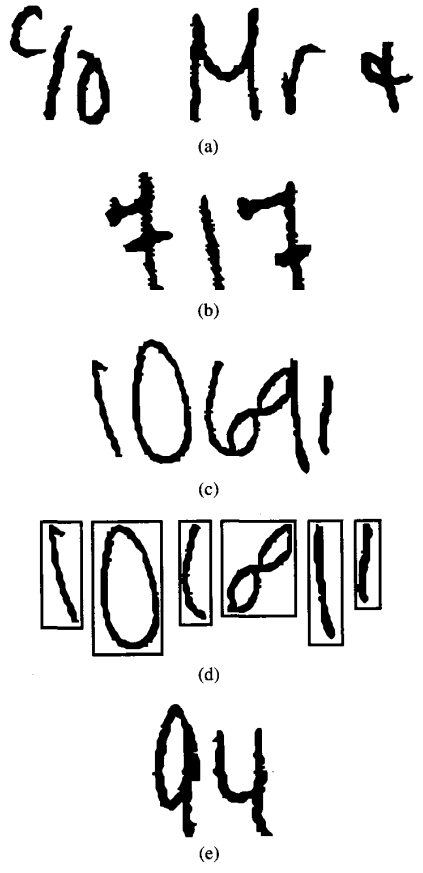

Fig. 11. Error examples.

C. Error Analysis

Training and testing yielded three major error categories: errors caused by numeric confidence assignments, by character confidence assignments, and by incorrect segmentations. The digit recognition neural networks sometimes assign high number confidences to characters, parts of characters, and unions of characters. Since the rules are heavily dependent on the numeric field confidence measurements, characters are occasionally mistaken for street number fields. An example is shown in Fig. 11(a) where the street number reading on the string "c/o" is read as a "90" with confidence of 0.782, a relatively high confidence.

On the other hand, some digits resemble characters. The character confidence can be high on digits, causing misidentification of street numbers as non-street number fields. In Fig. 11(b), the character reading on the first digit "7" is "*F*" with confidence value of 0.609, compared to the digit confidence reading of a "4" with a value of 0.443.

If the primitives are not computed correctly, then it is very unlikely that a high numeric field confidence will be obtained. For example, the primitive segments for the image block in Fig. 11(c) are shown in Fig. 11(d). Since the fourth primitive contains pieces of two different characters, there is no way to obtain correct recognition results on this image block.

In addition to these errors, some errors are caused by individual rules; however, changing the rule to correct one mistake often causes one or more other mistakes. An example is contained in Fig. 11(e). The number reading on the image is "04" with confidence value of 0.648. Since one of the system assumptions is no street number starts with a zero, however, the system assigns a low value for the street number field based one of its rules. Changing that rule would cause more errors on other image blocks, especially those having words starting with a "*O*." Therefore, optimal adjustment of the rules remains a subject for further study.

V. Street Number Location Using Neural Networks

We trained a wide variety of multilayer feedforward networks for locating the street numbers using backpropagation. We used the numeric input variables that were used in the fuzzy logic system as given in Table I, Section II–B1. We were able to train the networks to perform reasonably well but not as well as the fuzzy logic system.

Multilayer feedforward networks can perform very well for pattern classification tasks. For example they are extremely good tools for character recognition. One difficulty in using them, however, is that there are several parameters, such as the number of hidden units and the learning rate, to choose. We performed experiments using a wide range of values for these parameters.

The street number location neural networks used features from the blocks $p_1 \cup p_2 \cup \cdots \cup p_k$ and $p_1 \cup p_2 \cup \cdots \cup p_k \cup p_{k+1}$. The 16 input features are: gap_k, height_k, width_k, complex_k, conf_k, n_k, po_k, po-set_k, gap_{k+1}, height_{k+1}, width_{k+1}, complex_{k+1}, conf_{k+1}, n_{k+1}, po_{k+1}, po-set_{k+1}. Note that, because of how the features are defined, these inputs actually use information from all primitives $p_1, p_2, \cdots, p_{k+1}$. Thus, contextual information is available to the network. The network has one output which represents the confidence that the primitives $p_1 \cup p_2 \cup \cdots \cup p_k$ form a complete street number.

We used exactly the same data sets for training and testing that we used for the fuzzy logic system. Using the training set train1, we varied the learning rate and the number of hidden units. We also trained with and without balancing the training set. A balanced training set is one which contains the same number of samples per class. If a training set is not balanced, then a least squares error criteria can lead to a classifier which simply ignores a class which has very few training samples. For each variation, we evaluated the performance of the network on both the training set, train1, and the initial testing set, test1. Table VIII shows the results on obtained with different numbers of hidden units and with and without balanced training. The correct rate is the number of correctly located street numbers, or top 1 truth containment rate, as defined in Table II. The success rate is also defined in Table II. These results represent the best that we could obtain by varying the learning rate.

The testing correct and success rates are both lower than those obtained using the fuzzy logic system.

We then trained 15 different networks using the combined training set, btrain. We balanced the combined training set

TABLE VIII
BEST NEURAL NETWORK STREET NUMBER LOCATION PERFORMANCE RESULTS ON THE INITIAL TRAINING AND TEST SETS

# Hidden Units	Training Correct Rate	Testing Correct Rate	Training Success Rate	Testing Success Rate
4	76%	70%	85%	83%
4 (bal)	98%	85%	89%	86%
14	76%	70%	86%	83%
14(bal)	98%	85%	89%	86%
16	73%	73%	83%	85%
20	76%	70%	83%	82%
21	76%	70%	85%	83%

TABLE IX
BEST NEURAL NETWORK STREET NUMBER LOCATION PERFORMANCE RESULTS ON THE COMBINED TRAINING AND BLIND TEST SETS

# Hidden Units	Training Correct Rate	Testing Correct Rate	Training Success Rate	Testing Success Rate
2	93%	86%	85%	78%
4,6, 8,10,12,14, 16	90%	87%	83%	79%
18	93%	87%	84%	78%
20, 22, 24, 30, 32, 50	91%	87%	83%	79%

before training. Each network had a different number of hidden units. The networks were trained for 1500 epochs. Every 25 epochs the performance of the network was evaluated on the blind test set, btest. If the success rate of the network was better than the previous best success rate achieved by that network, the weights were saved. Thus, the performance results achieved were essentially the best achievable on the blind test set in 1500 epochs of training. This is in contrast to the fuzzy logic system experiment. In that experiment, the blind test set was actually used as a blind test set. Thus, the neural network approach was given a better chance to succeed. The performance of the networks on the blind test set always either remained constant or deteriorated for the last several hundred epochs of training, indicating that no further increases in performance were likely to occur with those networks.

The results of training on the balanced combined training set and evaluating on the blind test set are shown in Table IX. It is interesting to note that the networks trained to exactly or about the same rates, regardless of the number of hidden units.

Recall that the fuzzy rule base achieved a testing success rate of 86%, which is significantly better than that of 79% achieved with the neural network. The correct rate of 87% is essentially identical to that achieved by the fuzzy rule base. One may argue that if we tried 100 hidden units we may get a better success rate, however, we may not. But we may get a better success rate if we try 150 hidden units. This, of course, is a problem with neural networks; it is very difficult to know what the correct parameters are.

In summary, although we performed numerous experiments and used the same numeric features, we were unable to train the multilayer feedforward neural networks to perform as well as the fuzzy logic system. We conjecture that the reason is that the granularity of knowledge required to locate street numbers is "coarser" than that required to perform tasks such as character recognition. Tasks that require knowledge about the world that is not statistically represented in the data are difficult or impossible for neural networks to learn but the knowledge can be encoded into rules.

TABLE X
COMBINATIONS OF ADJECTIVES USED IN TYPE 1 NUMERIC FIELD AND GAP SIZE RULES

n_k (A)	gap_k (B)	adjconf (C)
small	small	neglarge
small	medium	neglarge
small	large	nearzero
medium	small	nearzero
medium	medium	postiny
medium	large	poslarge
large	small	possmall
large	medium	poslarge
large	large	poslarge

VI. CONCLUSION

We developed a fuzzy rule-based system for locating street number fields. The performance of this system illustrates the capacity for locating street number fields using a fuzzy rule-based system. The fuzzy rule base was compared to a neural network approach. The results of this system are highly dependent on preprocessing results, in this case, the results of neural networks and dynamic programming, as well as segmentation. Overall, the results of testing indicate that such a street number location system can be successfully embedded into a bigger address interpretation system.

APPENDIX A
RULES AND MEMBERSHIP FUNCTIONS

The entire set of 48 rules and associated membership functions are described in this appendix. We first describe the rules and then the membership functions.

Rules: There are 35 numeric field and gap size rules. Most possess one of the two basic structures

Type 1

IF complex_{k+1} is **large** AND n_k is **A** AND gap_k is **B**
THEN adjconf is **C**

and

Type 2

If complex_{k+1} is **small** AND n_k is **A** AND n_{k+1} is **B**
and gap_k is **C**
THEN adjconf is **D**.

The combinations of adjectives used in rules of Type 1 are given in Table X and those used in rules of Type 2 in Table XI.

Other rules involving numeric field confidence are

If complex_{k+1} is **small** AND n_k is **large** AND n_{k+1} is **small** and gap_k is **medium**
THEN adjconf is **possmall**.

TABLE XI
COMBINATIONS OF ADJECTIVES USED IN TYPE 2 NUMERIC FIELD AND GAP SIZE RULES

n_k (A)	n_{k+1} (B)	gap_k (C)	adjconf (D)
large	large	large	possmall
large	large	medium	postiny
large	large	small	nearzero
large	medium	large	possmall
large	medium	medium	postiny
large	medium	small	nearzero
large	small	large	poslarge
large	small	medium	possmall
medium	large	large	negsmall
medium	large	medium	negsmall
medium	large	small	neglarge
medium	medium	large	postiny
medium	medium	medium	nearzero
medium	medium	small	negtiny
medium	small	large	postiny
medium	small	medium	postiny
medium	small	small	postiny
large	large	large	nearzero
large	large	medium	negsmall
large	large	small	neglarge
large	medium	large	nearzero
large	medium	medium	neglarge
large	medium	small	neglarge
large	small	large	postiny
large	small	medium	negsmall
large	small	small	negsmall

If complex_{k+1} is **small** AND height_{k+1} is **tiny** and gap_k is **small**
THEN adjconf is **poslarge**.

If complex_{k+1} is **small** AND height_{k+1} is (**medium or large**) and n_k is **large** AND n_{k+1} is **small** and gap_k is **small**
THEN adjconf is **postiny**.

If complex_{k+1} is **small** AND height_{k+1} is **tiny** and n_k is **medium** AND n_{k+1} is **small** and gap_k is **small**
THEN adjconf is **possmall.**

If complex_{k+1} is **small** AND height_{k+1} is (**medium or large**) and n_k is **medium** AND n_{k+1} is **small** and gap_k is **small**
THEN adjconf is **negtiny**.

There are three rules involving recognition of the string "PO":
IF po_k is **medium** then adjconf is **negsmall**
IF po_k is **large** then adjconf is **neglarge**
IF po-set_k is **large** then adjconf is **neglarge**.

These rules state that if the confidence that any subblock of the current image block has a high confidence as the string "PO," then the street number confidence should be decreased significantly.

The final set of rules have weighting factors associated with them as discussed previously. These rules use the character and digit class information to turn rules off that could be invalid in the presence of ambiguous characters. For example, the character "O" and the digit "0" should both yield high character and numeric confidence values so there little information to be gained from the confidence values.

TABLE XII
VALUES OF X AND Y IN THE DEFINITION OF W3

X	Y
"g"	"9"
"o"	"0"
"b"	"6"
"l"	"1"
"s"	"5"
"z"	"2"
"z"	"3"
"I"	"1"
"O"	"0"
"S"	"5"
"Z"	"2"

The weighting factors are defined as follows: Let char_k be the character class associated with conf_k and digit_k the last digit in the digit string associated with n_k (the string which the dynamic programming algorithm returns as the highest confidence numeric string for the block $p_1 \cup p_2 \cup \cdots \cup p_k$). Let L_k be the length of this string. Define weights w_1, w_2, w_3, w_4, w_5 by the following (crisp) rules

$w_1 =$ 0 if $\text{char}_k \in$ { "O", "Z", "S", "I", "Z", "o", "z", "s", "l", "g", "b"}
1 otherwise.

$w_2 =$ 0 if $L_k \leq 5$
1 otherwise.

$w_3 =$ 0.1 if $\text{char}_k =$ "P"
0 if $\text{char}_{k+1} = X$ and $\text{digit}_{k+1} = Y$ where X and Y are given in Table XII
1 otherwise.

$w_4 =$ 0 if $\text{char}_{k-1} \in$ { "O", "Z", "S", "I", "o", "z", "s", "l", "g", "b"}
1 otherwise.

$w_5 =$ 1 if $c1$ = "0" (The numeral zero)
0.5 otherwise.

The rules that use these weights are
(w_1) IF n_k is **medium** and conf_k is **large**
THEN adjconf is **negsmall**.

This rule is active unless the character class is easily confused with a digit. It states that a medium number confidence and a large character confidence should result in a small negative decrease in street number confidence.

(w_2) IF width_k is **medium**
THEN adjconf is neglarge.

This rule is active if the most confident street number read so far is longer than five digits. This rule implies that if the width of the subimage is large and the street number reading is longer than five digits, then the street number confidence is decreased by a significant amount.

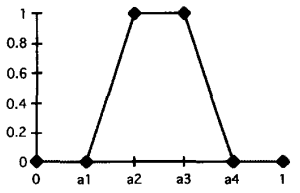

Fig. 12. Standard form for a trapezoidal number.

(w_3) IF gap_k is **huge** and c_{k+1} is **large**
THEN adjconf is **possmall**.

This rule is active if the character class being used is not easily confused with a digit class.

(w_4) IF n_k is **medium** and c_{k-1} is **large**
THEN adjconf is **negsmall**.

This rule is also active if the character class being used is not easily confused with a digit class.

(w_5) IF width is **large**
THEN adjconf is **neglarge**.

This rule discourages the system from reading street numbers that begin with "0" (zero).

B. Adjectives for Input/Output Variables

Each input/output variable has several adjectives. Each adjective is represent by a membership function. The domains of these functions are [0, 1] for input and [-1, 1] for output variable; and the ranges are [0, 1]. The following adjectives are defined as trapezoidal numbers and represented by the four points a_1, a_2, a_3, shown in Fig. 12. Note that $a_1 = a_2$ or $a_3 = a_4$ at the edges of the respective domains, e.g., for "large" and "small." Also, if $a_2 = a_3$, then the trapezoidal fuzzy set is actually a triangular fuzzy set.

TABLE XIII
ADJECTIVE MEMBERSHIP FUNCTIONS VARIABLE

Variable	Adjective	a1	a2	a3	a4
conf_{k-1}	large	0.50	0.75	1.00	1.00
conf_{k-1}	medium	0.25	0.50	0.50	0.75
conf_{k-1}	small	0.00	0.00	0.25	0.50
conf_k	large	0.60	0.80	1.00	1.00
conf_k	medium	0.25	0.50	0.75	1.00
conf_k	small	0.00	0.00	0.20	0.40
conf_{k+1}	large	0.60	0.80	1.00	1.00
conf_{k+1}	medium	0.25	0.50	0.50	0.75
conf_{k+1}	small	0.00	0.00	0.20	0.40
complex_k	large	0.50	1.00	1.00	1.00
complex_k	small	0.00	0.00	0.00	0.50
complex_{k+1}	large	0.50	1.00	1.00	1.00
complex_{k+1}	small	0.00	0.00	0.00	0.50
gap_k	huge	0.75	0.90	1.00	1.00
gap_k	large	0.45	0.70	1.00	1.00
gap_k	medium	0.25	0.45	0.45	0.75
gap_k	small	0.00	0.00	0.25	0.45
gap_{k+1}	large	0.50	0.75	1.00	1.00
gap_{k+1}	medium	0.25	0.50	0.50	0.75
gap_{k+1}	small	0.00	0.00	0.25	0.50
height_k	large	0.50	0.75	1.00	1.00
height_k	medium	0.25	0.50	0.75	1.00
height_k	small	0.00	0.25	0.25	0.50
height_k	tiny	0.00	0.00	0.10	0.25
height_{k+1}	large	0.50	0.75	1.00	1.00
height_{k+1}	medium	0.25	0.50	0.50	0.75
height_{k+1}	small	0.00	0.25	0.25	0.50
height_{k+1}	tiny	0.00	0.00	0.10	0.25
width_k	large	0.50	0.80	1.00	1.00
width_k	medium	0.25	0.50	0.50	0.75
width_k	small	0.00	0.00	0.00	0.50
n_k	large	0.60	0.80	1.00	1.00
n_k	medium	0.38	0.62	0.62	0.76
n_k	small	0.00	0.00	0.24	0.50
n_{k+1}	large	0.60	0.85	1.00	1.00
n_{k+1}	medium	0.38	0.62	0.62	0.76
n_{k+1}	small	0.00	0.00	0.24	0.50
po_k	large	0.50	0.70	1.00	1.00
po_k	medium	0.20	0.40	0.40	0.70
po_k	small	0.00	0.00	0.10	0.40
po_set_k	large	0.40	0.70	1.00	1.00
po_set_k	medium	0.20	0.40	0.40	0.70
po_set_k	small	0.00	0.00	0.10	0.40
adjconf	poslarge	0.40	1.00	1.00	1.00
adjconf	possmall	0.16	0.37	0.37	0.60
adjconf	postiny	0.00	0.10	0.10	0.21
adjconf	nearzero	-0.10	0.00	0.00	0.10
adjconf	negtiny	-0.20	-0.10	-0.10	0.00
adjconf	negsmall	-0.50	-0.30	-0.30	-0.10
adjconf	negtiny	-0.20	-0.10	-0.10	0.00
adjconf	negsmall	-0.50	-0.30	-0.30	-0.10
adjconf	neglarge	-1.00	-1.00	-0.75	-0.30

ACKNOWLEDGMENT

We would like to thank the reviewers for their many helpful comments. We would also like to acknowledge the many researchers who contributed to the development of the systems used to generate input for the fuzzy rule base.

REFERENCES

[1] D. J. Hepp, M. J. Ganzberger, M. P. Whalen, T. M. Peurach, B. D. Forester, and A. M. Gillies, "A system for the analysis of machine printed address block images using contextual information," in *Proc. U.S. Postal Serv. Adv. Technol. Conf.*, Nov. 1992, pp. 353–366.

[2] P. B. Cullen, J. J. Hull, and S. N. Srihari, "A constraint-based approach to postal address interpretation," in *Proc. U.S. Postal Serv. Adv. Technol. Conf.*, Nov. 1992, pp. 367–377.

[3] M. Ganzberger, R. Rovner, D. Hepp, A. Gillies, C. Lake, and C. Wood, "A system for handwritten address interpretation," in *Proc. of the U.S. Postal Serv. Adv. Technol. Conf.*, Nov. 1992, pp. 337–351.

[4] S. N. Srihari, E. Cohen, V. Govindaraju, and A. Shekhawat, "Determining delivery point codes on handwritten addresses," in *Proc. U.S. Postal Serv. Adv. Technol. Conf.*, Nov. 1992, pp. 321–336.

[5] V. Govindaraju, A. Shekhawat, and S. N. Srihari, "Interpretation of handwritten addresses in US mail stream," in *Proc. Third Int. Workshop Frontiers Handwriting Recogni.*, May 1993, pp. 207–217.

[6] F. Kimura, M. Shridhar, and N. Narasimhamurthi, "Lexicon directed segmentation—Recognition procedure for unconstrained handwritten words," in *Proc. Third Int. Workshop Frontiers Handwriting Recogni.*, May 1993, pp. 122–132.

[7] A. Gillies, D. Hepp, and P. Gader, "A system for recognizing handwritten words," *ERIM Tech. Rep.* submitted to U.S. Postal Service, Nov. 1992.

[8] P. D. Gader, M. P. Whalen, M. Ganzberger, and D. Hepp, "Handprinted word recognition on a NIST data set," *Machine Vision and Applications.*, 1994 (accepted for publication).

[9] P. D. Gader, M. Mohamed, and J.-H. Chiang, "Fuzzy and crisp handwritten alphabetic character recognition using neural networks," in *Proc. Art. Neural Networks Eng.*, St. Louis, MO, Nov. 1992, pp. 421–427.

[10] ——, "Segmentation-based handprinted word recognition," in *Proc. of U.S. Postal Serv. Adv. Technol. Conference.*, Nov. 1992, pp. 215–225.

[11] ——, "Comparison of crisp and fuzzy character neural networks for handwritten word recognition," in *Proc. N. Am. Fuzzy Infor. Process. Soc., NAFIPS '92*, Dec. 1992, pp. 257–266.

[12] ——, "Crisp and fuzzy character neural networks for handwritten word recognition," *IEEE Trans. Fuzzy Syst.*, under review.

[13] ——, "Handwritten word recognition with character and inter-character neural networks," submitted to *IEEE Trans. Syst. Man Cybern.*, 1993.

[14] P. D. Gader, B. Forester, M. Ganzberger, A. Gillies, B. Mitchell, M. Whalen, and T. Yocum, "Recognition of handwritten digits using template and model matching," *J. Patt. Recogni.*, vol. 24, no. 5, 1991.

[15] A. M. Gillies and B. T. Mitchell, "A model-based approach to handwritten digit recognition," *Machine Vision Applicat.*, vol. 2, 1989, pp. 231–243.

[16] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Inform. Sci.*, Part I, vol. 8, pp. 199–249; Part II, vol. 8, pp. 301–357; Part III, vol. 9, pp. 43–80, 1975.

[17] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Syst.*, vol. 1, 1978, pp. 3–28.

[18] L. Zadeh, "The theory of approximate reasoning," in *Machine Intell.*, vol. 9, J. Hayes, D. Michie and L. Mikulich, Eds. New York: Halstead, 1979, pp. 149–194.
[19] D. Dubois and H. Prade, "Fuzzy sets in approximate reasoning, part 1: Inference with possibility distributions," *Fuzzy Sets Syst.*, vol. 40, 1991, pp. 143–202.
[20] CubiCalc, *Computer Software Manual.* HyperLogic Corporation, IBM-PC, 1990.
[21] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1992.
[22] J. Cai, "A fuzzy rule-based system for locating street numbers," M.S. thesis, Univ. of Missouri–Columbia, 1993.
[23] "SUNY Buffalo postal address image database," State Univ. of New York at Buffalo, Dept. of Comp. Sci., 1989.

Paul Gader (M'87) received the Ph.D. in applied mathematics from the University of Florida in 1986.

Dr. Gader has held positions as Section Manager and Research Engineer at the Environmental Research Institute of Michigan, Senior Research Scientist at Honeywell Systems and Research Center and as Assistant Professor of Mathematics at the University of Wisconsin–Oshkosh. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Missouri–Columbia. His current research interests are in a variety of topics related to handwritten and machine-printed address interpretation, medical image analysis, automatic target recognition, obstacle detection, scene analysis, image algebra and mathematical morphology, and applied mathematics.

James M. Keller (M'79–SM'92) received the Ph.D. in mathematics in 1978.

He has had faculty appointments in the Bioengineering/Advanced Automation Program, the Research Reactor, and the Electrical and Computer Engineering Department at the University of Missouri-Columbia, where he currently holds the rank of Professor. He is also the E.A. Logan Research Professor in the College of Engineering. His current research interests include computer vision, pattern recognition, fractal geometry for natural scene analysis, neural networks, and the modeling and management of uncertainty.

Dr. Keller is a national lecturer for the Association for Computing Machinery (ACM) and is currently the President of the North American Fuzzy Information Processing Society (NAFIPS). He is an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the *International Journal of Approximate Reasoning* and is on the editorial board of the *Journal of Intelligent and Fuzzy Systems.*

Juliet Cai (S'90–M'91) received the M.S. degree in electrical engineering from the University of Missouri–Columbia in 1993.

She is currently an Engineer at Black and Veatch in Kansas City, MO.