Reading Handwritten Digits: A Zip Code Recognition System

Ofer Matan	(908) 949-9423	ofer@neural.att.com
Jane Bromley	(908) 949-8901	jbromley@neural.att.com
Christopher J. C. Burges	(908) 949-8525	cjcb@neural.att.com
John S. Denker	(908) 949-8128	jsd@neural.att.com
Lawrence D. Jackel	(908) 949-7773	ldj@neural.att.com
Yann Le Cun	(908) 949-4038	yann@neural.att.com
Edwin P. D. Pednault	(908) 949-1074	epdp@vax135.att.com
Wiliam D. Satterfield	(908) 949-4209	wds@neural.att.com
Charles E. Stenard	(908) 949-0630	ces@erebus.att.com
Timothy J. Thompson	(908) 949-4339	tjt@twitch.att.com

AT&T Bell Laboratories, Holmdel, N. J. 07733

KEYWORDS: Optical Character Recognition, Handwriting Recognition, Neural Networks

Abstract

In this paper we describe the problem of reading unconstrained handwritten zip-codes and outline the strategies and methods of solution. We have designed a system that reads unconstrained handwritten zip-codes taken from real U.S. Mail envelopes. This multi-digit system is an extension of a previous system that recognizes isolated handwritten digits using a multi-layered neural network. The new system has been implemented on a digital signal processing board and has achieved state of the art recognition rates.

The system is a hybrid of image processing, segmentation, and neural network classification. The part of the system carrying out segmentation ("segmenter") is "recognition driven". That is, the image may be segmented in a variety of ways, but the segmentation that is chosen is that giving the highest recognition score when the recognition scores for the individual digits are combined.

The strength of the segmenter is that it uses simple methods for easy cases and more complex techniques for harder ones. Connected components analysis handles the case of non-touching digits. Touching or disconnected digits are handled by estimating vertical cut-points and scoring the resulting segments using the neural network classifier. The best combination of cut-points is found by using dynamic programming. This approach ensures that the number of calls to the classifier is minimal, hence processing time is minimized.

We discuss the speed and accuracy achieved by the implemented system.

1 Introduction

Reading is a difficult task. Humans still outperform machines in most vision tasks, in both speed and quality. Our goal is to design machines that can read handwriting at levels approaching or exceeding human performance. We have developed algorithms able to read strings of handwritten digits using neural networks. These algorithms have been incorporated into a system that reads handwritten zip-codes appearing on real U.S mail.

The main body of this paper describes the zip-code reader we have developed. Although we have used knowledge specific to this application, the techniques we have used are applicable to many other handwriting recognition applications.

In the following subsection we introduce the motivation for the work in this area. Section 2 serves as an introduction to handwritten digit recognition. We list general requirements and techniques relevant to single digit and multi-digit recognizers. In Section 3 we describe the postal application followed by a description of the system we have developed. Sections 5 and 6 report the system's recognition performance and implementation details. We finally summarize and discuss future work.

Why is handwritten character recognition important ?

In recent years there has been a constant increase in documents on paper. The number of mail pieces sent and checks written, grows from year to year. Contrary to the belief that advances in electronic technology would help create a paperless society, they have helped increase the average amount of paper documents people and businesses handle every day. Therefore, there is a great need for machines able to read paper documents.

The field of automated text reading is known as "Optical Character Recognition (OCR)". In a typical system, an image of a document is acquired by camera or scanner ("image capture"). The system subsequently locates the section of the image relevant to the application (e.g. the dollar amount on a check, the address block on a mail piece, columns in a book, etc.). It then proceeds to recognize the characters in the field of interest. Pattern recognition techniques are used in conjunction with contextual information particular to the application (e.g., spell checkers and grammars for written prose, check digits on bank account numbers, etc...).

For machine printed documents, OCR systems have achieved considerable success. Single-font and relatively clean originals can be read at very high accuracy and speed. State of the art recognition rates of individual characters is above 99%. Typical systems that read pages of "clean" text make less than 4 mistakes per page by the use of spell checkers and other correction methods (for example see [1]). A good introduction to this field is reference [2].

The success in reading machine printed documents is due to their regularity. Since print is typeset in straight rows and columns, locating and recognizing the characters is relatively easy. This is not the case for handwriting.

Conveying the complexity of reading handwriting is not difficult; everyone has experienced not being able to read another person's (in some cases their own) handwriting.

The current "interface" between handwritten information on paper and electronic media is a human visual processing system (i.e. a data entry person). The potential savings of automating this process in terms of manual effort and speed is large.

The above motivation has generated a considerable amount of research in handwriting recognition over the past 30 years. However, only recently have the strength of the algorithms and the power and cost of the computers combined to make available, systems that are commercially feasible. For an collection of state-of-the-art research papers in handwriting recognition see reference [3].

2 Handwritten digits: from isolated digit recognition to multi-digit recognition

2.1 Introduction

This section introduces the problem of classifying images of handwritten digits. We proceed from the simple case of recognizing a single digit to the case of a multi-digit string. We give a general description of the problems addressed and the components needed to solve these problems. In Section 4 we describe in detail specific components for a zip-code recognizer based on a neural network digit recognizer.

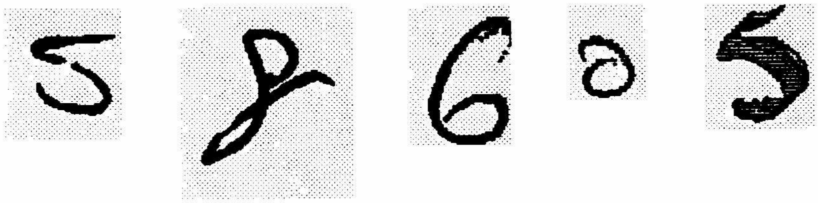


Figure 1: Examples of isolated digits.

2.2 Isolated digits

In Figure 1 some images of handwritten digits are presented[2]. We wish to associate each image presented, with one of the ten classes: 0 to 9. This problem of character recognition has been researched for over 30 years. Below, we outline some requirements and possible solutions to this task. A good introductory reference to some of the techniques we mention is [4].

Apart from assigning the input image to a class, we expect the recognition algorithm to output a score associated with the classification. This score represents the level of confidence in the classification. The score is used for rejecting patterns with low confidence levels (see Figure 2). Some recognition techniques estimate the probability of the correctness of the classification, while others do not. We shall see below that generation of probability estimates is important for multi-digit recognition.

Usually character recognition consists of three main processing stages: preprocessing, feature extraction and classification (see Figure 3). We shall briefly describe each of these stages and list some of the known techniques for each stage.

2.2.1 Preprocessing

The preprocessing is designed to separate the signal from the noise, making the following stages more robust. Noise in the case of handwriting is not only dirt and interference in the image capture. It is also variability in

[2] These images and the remainder of the images presented in this paper are black and white. However, most of the techniques we mention can be extended to grey-level images.

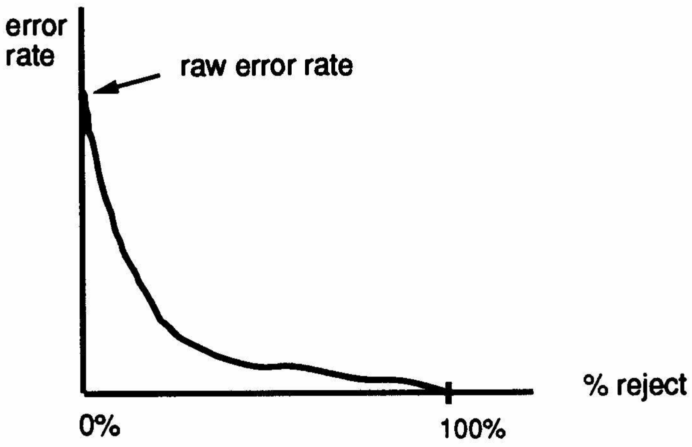


Figure 2: **Trading errors for rejects.** In many applications we wish to reject a pattern if the confidence in the classification is low. The motivation is greater if the cost of an error is much greater than the cost to reject. For example, it is more costly to send a letter by mistake to the wrong state than have a human sorter look at the envelope. The score returned by the clasifier is used for rejection. The typical form of an error versus rejection graph is shown above. The "raw error rate" is the number of classification errors the system made divided by the total number of patterns classified. If we reject some of the lowest scoring classifications the error rate of the patterns not rejected decreases. In the limit where we reject all patterns - we make no mistakes. Error rates in this paper are the number of errors divided by the number of patterns accepted (not rejected). In a real system the decision on what percent to reject will depend on the cost model: what are the costs of errors and rejections.

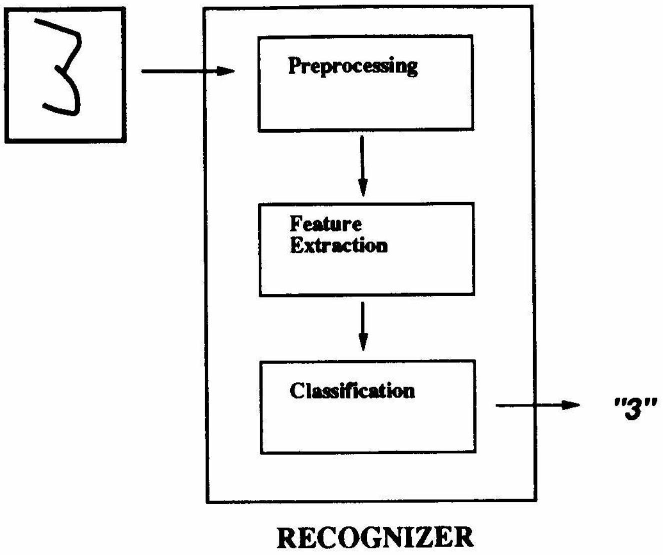


Figure 3: Typical stages in single character recognition.

handwriting styles : the size, the slant, thinkness of the stokes and other variables. Some preprocessing operations that may be taken are:

- Thresholding/Binarizing/Grey-level normalization - setting the pixel quantization levels to those most suited to the feature extraction and classification steps.
- Noise correction - corrections due to bad image capture. This may be speck removal or image enhancement.
- Size normalization - rescaling the image to a standard size needed for later stages.
- Slant correction - de-slant digits to a "normal" form.
- Thining - transform the image so the stroke width is only one pixel wide. This is necessary for some feature extractors.

2.2.2 Feature Extraction

The feature extractor generates a set of features that will enable the classification algorithm to distinguish between patterns of various characters. Examples of geometrical features that have been used: edges, end-points of strokes, crossings of strokes and closed loops. Another class of features used are generated by transformations such as the Fourier and Hough transforms.

2.2.3 Classification

The classification stage associates the feature vector[3] with a certain class, returning the confidence level of the association. We shall mention three classes of methods: template matching, syntactic classifiers and statistical classifiers.

Template matching characterizes a given pattern (or feature vector) by measuring its distance from a set of prototype patterns (feature vectors). The distance may or may not be the Euclidean distance. The pattern is classified to belong to the class of the closest prototype, and the confidence score is some function of that distance. There are various extensions of template matching such as K-nearest neighbors [4].

Syntactic methods describe each class as some specific combinations of features. The set of "legal" combinations for each class is defined by rules (a grammar). An example of a rule would be: "An eight is a loop on top of another loop".

Statistical classifiers use a set of training examples to estimate the distribution of the various classes in pixel or feature space. This is analogous to curve fitting. Using the set of training examples whose respective class is known, we can tune a set of parameters governing the classifying function to give the correct answer. Once the classifier is trained, when shown a new pattern it will try to estimate the probability of that pattern belonging to any class. A typical statistical classification technique is Parzen windows [4]. Neural networks are often considered part of this family; in Section 4.2 we describe the neural network classifier we have designed.

State-of-the-art handwritten digit recognizers have "raw recognition rates" [4] of 95% and above (depending on the quality of the data). The leading algorithms do not differ much in their raw performance. However, they do have different characteristics such as : computation complexity, memory require-

[3] Some practitioners do not apply a feature extraction, but apply the classification algorithm directly on the pixel pattern. One can view this as a feature vector where each feature is a pixel in the image. We shall term this as a vector in pixel space. An extracted feature set is termed a vector in feature space.

[4] We term the "raw recognition rate" as the recognition rate with no rejection

ments, training time and error rates at different rejection rates. The choice of one over the other is dependent on the specific needs of the application.

An isolated digit recognizer can be useful for form reading applications, where the digits are written in predefined boxes. Typical applications are credit card slips and tax forms. The recognition system can locate the box and classify the digit lying within the box. In the following subsection, we shall discuss the problem of recognizing strings of digits where the writer is not constrained to write each individual digit in a predefined box. We term this as an "unconstrained string of digits". The string may be confined to a box located on a form (e.g. the dollar amount on a personal check) or may appear anywhere in the document (e.g. the zip-code on an addressed envelope).

2.3 Multi-digit recognition

Assuming we have built a single digit recognizer, how can we use it to recognize an unconstrained string of digits ?

2.3.1 Segmentation

In order to recognize a string of digits we need a component that will separate the whole image into subimages each containing a single digit. This component is the "segmenter". Each partial image ("segment") is then in turn classified by the recognizer (see Figure 4).

A string of three digits is shown in Figure 5a. A simple segmentation algorithm can be constructed for this image; each segment will be a "blob" of black pixels (a set of contiguous black pixels). The "blobs" are called "Connected Components (CCs)". "Connected Components Analysis (CCA)" is an algorithms to locate all the CCs in an image. There are various methods to implement CCA which we will not discuss.

Each of the three CCs in Figure 5a is an individual digit. This is not always the case; in Figure 5b there are three digits and three CCs as well. Not one of the CCs is an individual digit. The "4" and "0" are combined to one CC and the "5" is disconnected into two CCs. A CCA segmenter would fail on this example. There is need for a much more sophisticated segmenter. The first CC must be bisected into two segments, and remaining two CCs joined to one segment.

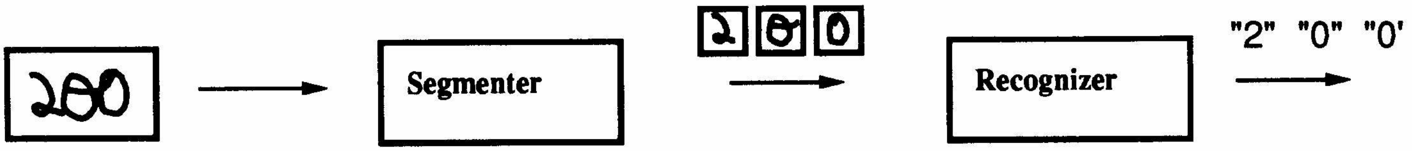


Figure 4: A simple approach to multi-digit recognition: segmentation followed by recognition.

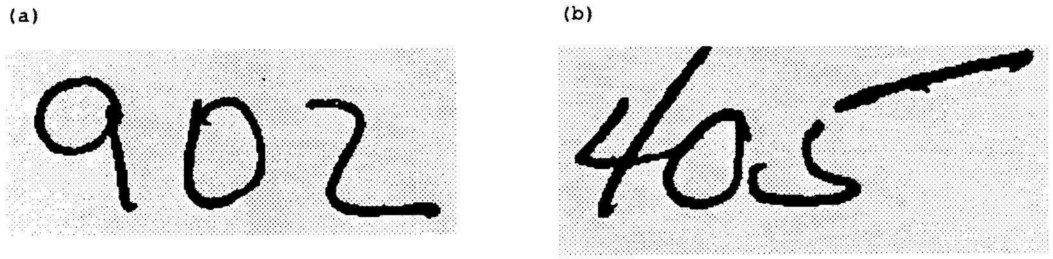


Figure 5: Multi-digit strings. In string (a) each connected component is a digit, in string (b) two digits have combined to one connected component and another digit is divided into two connected components.

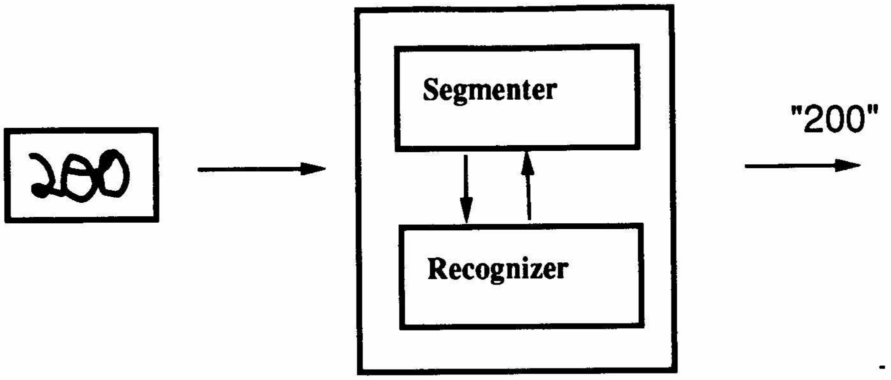


Figure 6: Recognition based segmentation: segmentation is coupled to the recognition unit.

Various methods to segment strings of handwritten characters have been proposed. Some follow the strokes of the writer, others look at pixel projections, and others look at upper or lower contours of the image.

Whatever segmentation method is used, there are limitations in using the pipeline scheme shown in Figure 4. The underlying assumption is that segmentation and recognition can be decoupled. Except for the most simple cases this is not true.

2.3.2 Recognition-based Segmentation

The latest developments in speech recognition have demonstrated the power of using the recognition engine to score each segment in a candidate segmentation [5, 6], The individual segment scores are combined to give a global score for a specific candidate segmentation. The segmentation that gives the best combined score is chosen. This "recognition driven" segmentation is usually used in conjunction with dynamic programming to efficiently find the optimal solution. In this approach the recognition engine and the segmenter are tightly coupled (see Figure 6). This is a method we believe to be applicable to multi-character recognition.

The number of possible segmentations of a specific image is extremely large. Even if we could rely only on the recognition unit to determine the correct segmentation, it would not be feasible to try scoring all possible segments, even with an extremely fast classifier; we could not reach a solution in reasonable computing time. It is necessary to limit the number of candidate segmentations as much as possible. We would like to limit the candidates we consider, to the smallest possible set while retaining high confidence that the correct solution is a member of it. Generation of segmentation candidates can be based on segmentation methods mentioned in section 2.3.1. In the zip-code reader we have used a hybrid of CCA and vertical cuts of the image to generate the candidate segmentations.

2.3.3 Hybrid methods

Hybrid methods have been used character recognition (see for example [7]) and other fields of engineering. Most of the algorithms known for OCR have certain areas of weakness, giving rise to misclassifications of certain types of patterns. If two or more methods err on different patterns, then using some combination of their answers can decrease the overall error rate. Using a hybrid method for segmentation can be useful in the same manner.

Using multiple methods does not necessarily imply executing all of them for every image. The decision to pass to a various method may be based on the data itself or the success of a previous attempt. A reasonable rule for building an efficient hybrid solution is to use the fastest and simplest method for the easiest cases and try applying more complex method to the more difficult cases. Doing so can give an increase in performance rates while keeping processing speed minimal.

3 An introduction to the postal problem

About 15 percent of the addresses on 1st class mail are handwritten (hand-printed or cursive). This relatively small percentage translates to over ten billion mail pieces annually. Current OCR systems can read machine printed mail. However, handwritten mail pieces are rarely read. Reading even a fraction of them at high confidence will yield potential savings of millions of dollars per year.

The zip-code recognizer we have designed is only part of a total system that would read addresses from mail-pieces. An overview of other necessary components is:

1. **Image capture** – machinery is needed to take a picture of the correct side of the envelope.
2. **Address block locator** – a process to locate the area on the image containing the addressee's address, and return in the correct orientation
3. **Line and field locator** – this process locates lines and various fields in the address block. Such as city , state and zip-code.
4. **Field recognizers** – these might be general character recognizers or might be specialized for a specific field (e.g. the zip-code).
5. **Contextual analysis** – this unit may be incorporated into other parts of the system. It uses partial knowledge from the image and other information (such as mailing statistics and the postal address database) to generate hypotheses for subfields or the whole address.

Our system is built on the assumption the first three parts exist and have succeeded in providing a right-side-up image of the zip-code field that holds a 5 or 9 digits zip-code. However, there may be intruding strokes from other fields. A schematic view and data flow of a whole system are shown in figure 7.

We trained and tested our system on approximately 10,000 images, 5 and 9 digit zip-code fields taken from live mail. The images are black and white scanned at 212 pixels per inch. The zip-code database was created by contractors to the U.S. Postal Service. The zip-code fields were located by humans and extracted by drawing rectangular boxes around the field. We used 7,000 images for training purposes, and the remaining 3,000 for testing. Several examples of zip-code fields supplied to our system are shown in figure 8.

4 The zip-code reader

4.1 Overview of system

The novelty of this system is the recognition based segmenter. The segmenter is a hybrid of CCA, vertical cuts and a neural network recognizer. One of the experiments we conducted before designing our system was to measure the performance a CCA segmenter with our recognizer on a test set of zip-codes. The number of zip-codes correctly recognized was about

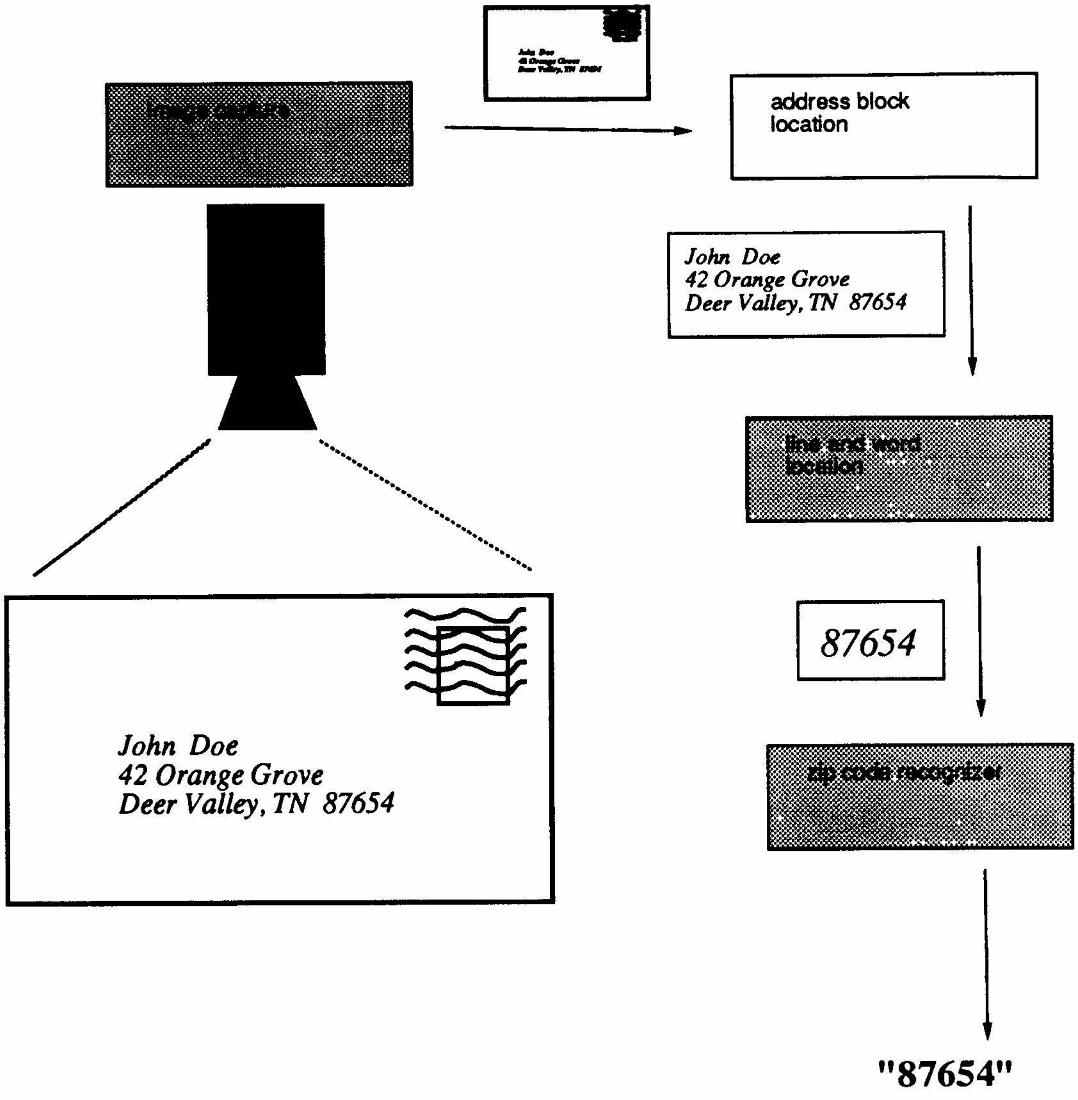


Figure 7: Data flow of images from the image capture to the zip-code recognizer.

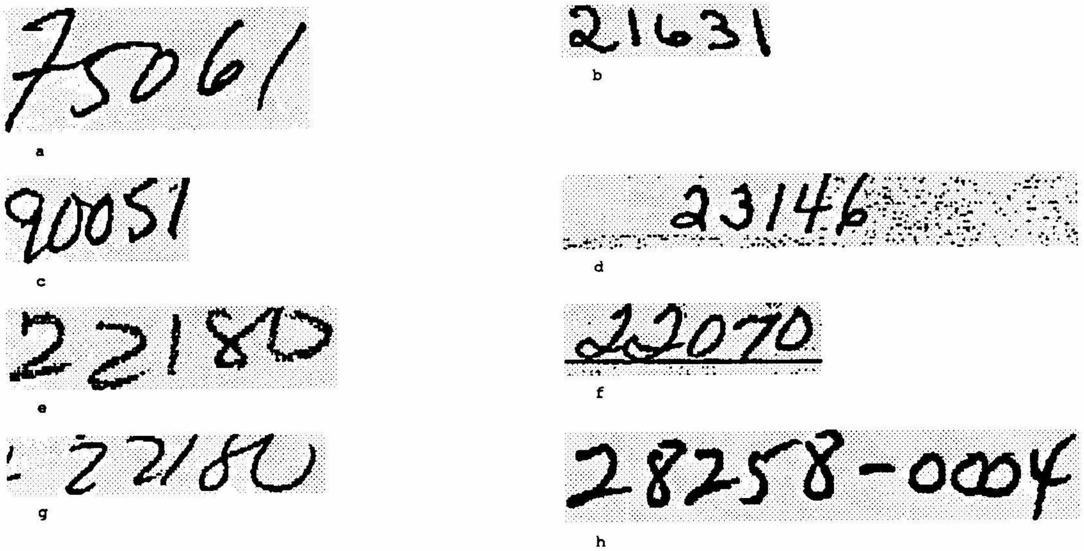


Figure 8: Examples of zip-codes.

35%. CCA is destined to fail when two or more digits are touching or one digit is disconnected. We wished to build a system that would have higher recognition performance while retaining the CCA to handle the easier cases. Our choice of an additional segmentation algorithm was vertical cuts of deskewed images.

Both segmenters may be used for a particular image; connected components that are single digits will be handle by CCA. CCs that are combined or dissected digits will be handled by the vertical cut segmenter.

The main stages of processing are the following (see figure 9):

Preprocessing – The image is converted to a standard form by removing noise and de-slanting the digits.

CCA segmentation and recognition – Each connected component is evaluated by the recognition unit. If the score is above a certain threshold the the connected component is considered solved. Segments below with scores below threshold are handled by the next level of segmentation.

Vertical cut-point estimation and segmentation – Parts of the image that have not been deemed as solved are segmented in various ways using vertical cuts. Each segment is classified and scored by the classifier. The segmentation that gives rise to the best combined score is chosen.

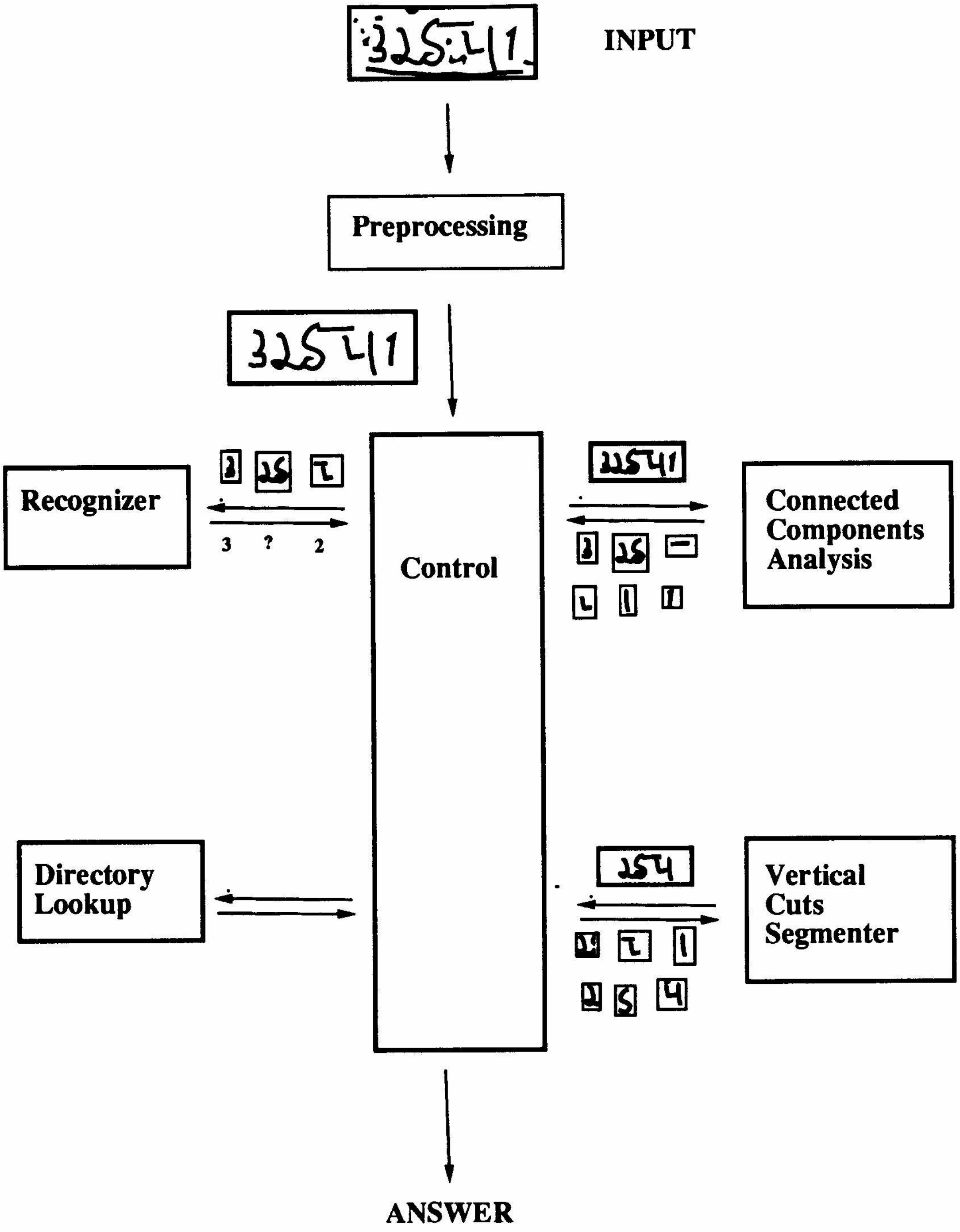


Figure 9: System overview.

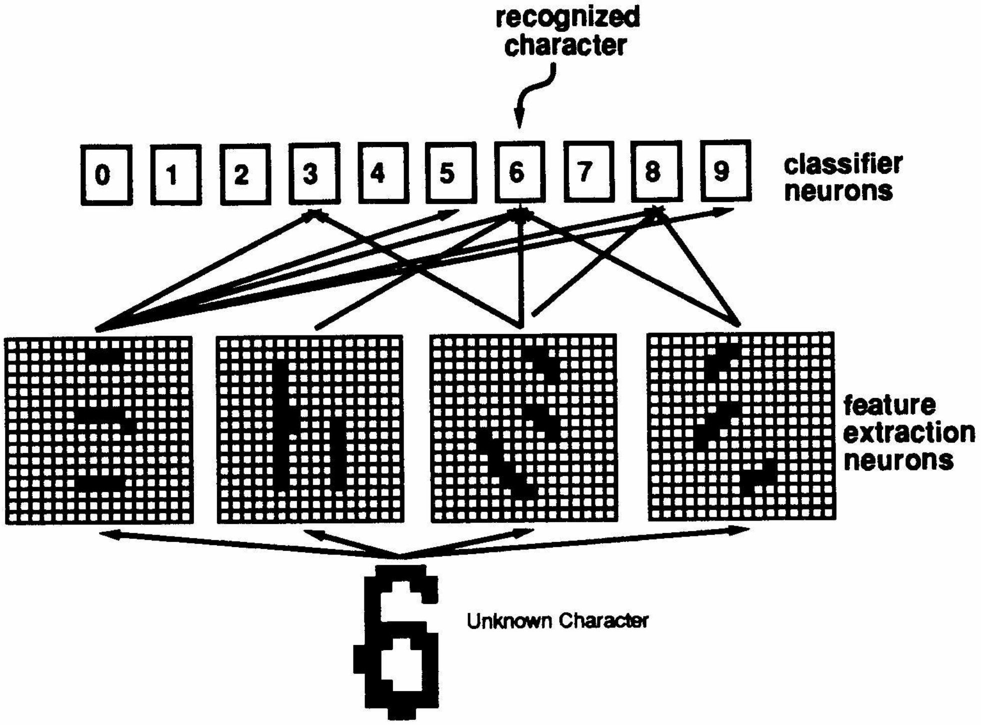


Figure 10: A schematic view of the recognizer. The network is fed a pixel image. The image signal is propagated through a set of feature extracting neurons to an output layer consisting of 10 output units, one for each class.

Directory lookup - Answers that do not appear in the zip-code directory are rejected and the vertical cut segmenter is recalled to generate the next highest scoring candidate.

4.2 The digit recognizer

Both segmentation components call the digit recognizer. This is a neural network that has been trained using back-propagation. This recognizer has been described in detail in [8]. A schematic figure of the network is shown in Figure 10

The main feature of this neural network is that it is directly fed with a pixel image, rather than feature vectors. This eliminates the need to design a complex feature extraction stage. The only preprocessing done is normalizing the input image to a 20 by 20 pixel grey scale image using a linear transformation. The grey levels are scaled to fall within the range -1 to $+1$.

The output is composed of 10 units, one for each digit. The desired output for a pattern belonging to class i is is $+1$ for the ith output unit, and -1 for the other output units.

Back-Propagation networks are composed of several layers of interconnected elements (units) arranged in a feed-forward architecture: connections can only go from lower layers to higher layers. Each unit resembles a "soft" linear classifier, computing a weighted sum of its input, and transforming this sum through a non-linear squashing function (usually a sigmoid function such as tanh). Learning is performed by iteratively modifying the weights on each connection so as to minimize an objective function. A popular objective function is the mean squared error between the actual output of the network and a desired output. Minimizing the objective function is performed by a gradient descent procedure which requires the computation of the gradient of the objective function with respect to connection weights. Back-Propagation is just an efficient way to compute this gradient.

The design of the connections between the computing layers must be guided by our knowledge about shape recognition. Because there are well-known advantages to performing shape recognition by detecting and combining local features, our network has only *local connections* in all but the last layer. Furthermore, salient features of a distorted character might be displaced slightly from their position in a typical character, or the same feature can appear at different locations in different characters. Therefore a feature detector that is useful on one part of the image, is likely to be useful on other parts of the image as well. Specifying this knowledge can be performed by forcing a set of units, located at different places on the image, to have identical weight vectors. The outputs of such a set of neurons constitute a *feature map.* This operation is equivalent to a convolution with a small size kernel, followed by a squashing function. Figure 11 depicts two convolutional type feature maps operating on an input map.

The idea of local, convolutional feature maps can be applied to subsequent hidden layers as well, to extract features of increasing complexity and abstraction. Interestingly, higher level features require less precise coding of their location. Reduced precision on the position is actually advantageous, since a slight distortion or translation of the input will have reduced effect on the representation. Thus, each feature extraction layer in our network is followed by an additional layer which performs a local averaging and a subsampling, reducing the resolution of the feature map. This layer introduces a certain level of invariance to distortions and translations, due to the smoothing nature of the averaging. Our designed architecture is a "bi-pyramid": The loss of spatial resolution in the feature maps (due to subsampling) is

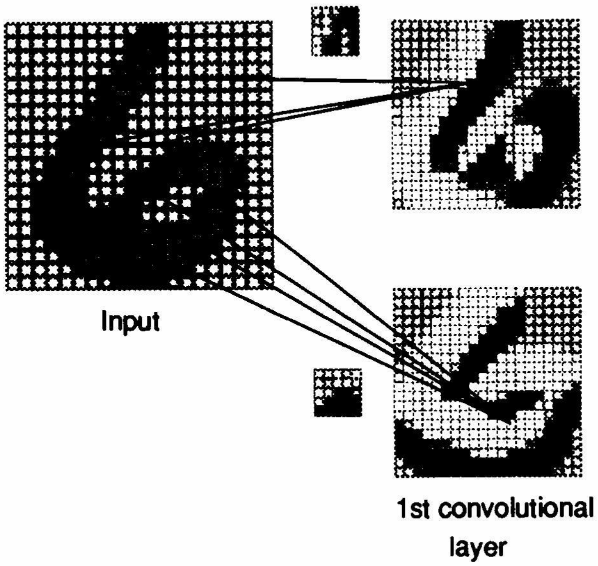


Figure 11: Two convolutional feature maps from the first layer in the network. The size of each unit specifies the magnitude of the activation and the color the sign. The values of the connection weights are plotted above the connection.

partially compensated by an increase in the number of feature types.

The network architecture is composed of 5 layers of processing. The first is a convolutional layer followed by a subsampling layer. The third is an additional convolutional layer, again followed by a subsampling layer. The fourth layer is fully connected to the 10 units of the output layer. There are 6465 nodes in the network. There are approximately 150,000 connections. Since many of them are constrained to have the same value the number of free parameters is only 3658. All these are free to adjust during training. The internal states of all the units in the network when presented with an example "6" are shown in Figure 12.

Training was done using back-propagation. The network used for the zip-code recognizer was bootstraped from a network trained on about 10,000 isolated digits described in [8]. After we were able to segment a considerable number of zip-codes using the whole system, the network was retrained on digits that had been segmented by the system, thus tuning it to recognize isolated digits generated by the segmenter it has to cooperate with.

A problem we encountered while developing the system was that the

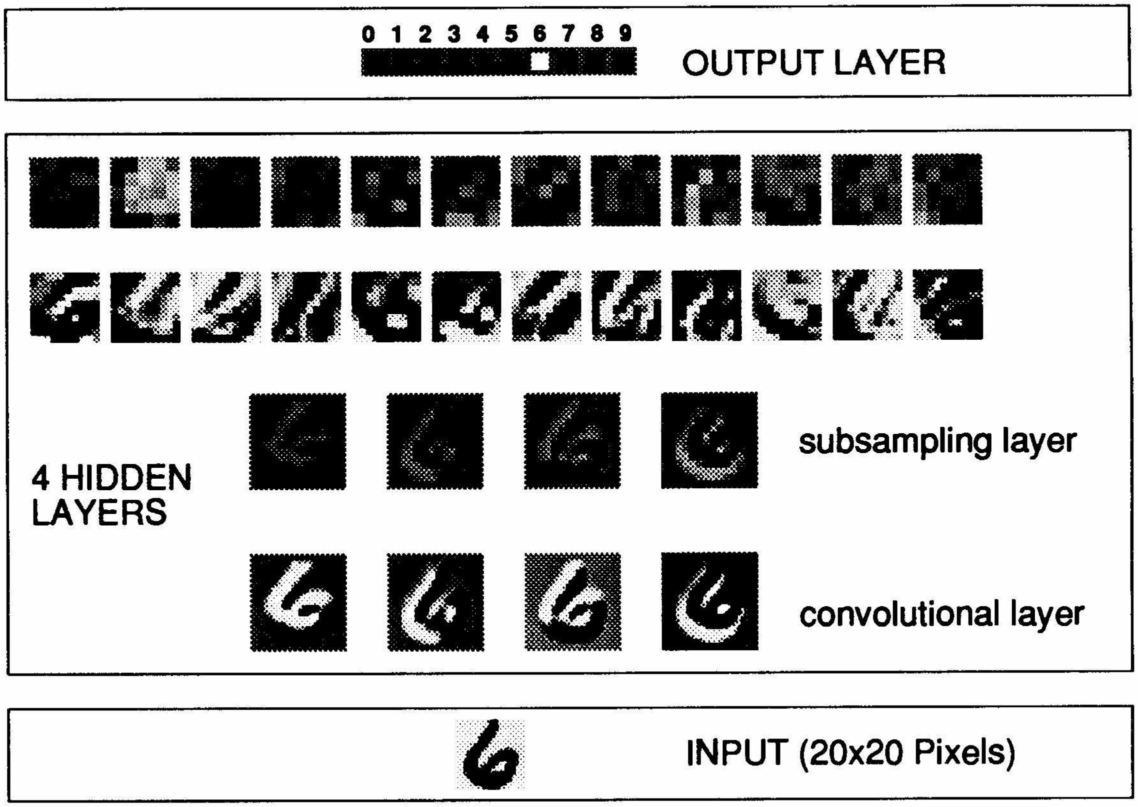


Figure 12: The internal states of the network for a specific pattern.

recognizer would assign reasonable score to segments that were not digits. A solution we applied to this problem was found by training these non-digit segments as counter-examples. That is, setting their target to "none of the above". This was achieved by setting the desired state for these examples to -1 for all ten output units.

Converting the output activation layers to scores

We wish the output of the classifier to be an estimate of the probability of correct classification. In oreder to achieve this we apply the Softmax normalization scheme for classifiers with N mutually-exclusive outcomes [9]. This scheme is applied as a post-processor to the "raw outputs" of the neural network. Softmax's appeal is that its outputs are positive and sum to 1, thus satisfying the axioms of probability theory. Each Softmax output is an increasing function of the corresponding raw output (when the other outputs are held constant) and preserves the ordering of the classes. There are other interesting theoretical qualities of Softmax, such as its connection to the entropy of the system [9]. The modified form of Softmax, we use, is the following:

$$S_i = \frac{e^{\beta O_i}}{e^{\alpha} + \sum_k e^{\beta O_k}}$$

Where O_i is the activation level of output unit i and S_i is the Softmax score for class i. α and β are adjustable parameters.

The term involving α essentially represents the activation level of the artificial $N+1$st category, the "none of the above" category. It will cause reduction of the score when the highest active unit has a low absolute value.

Experiments we have conducted show that the Softmax scores are good estimations of the probability of correct classification[10]. This is of great importance, since it enables us to combine the individual digits' scores into a combined score for a given segmentation.

Dash recognition

Dashes are not recognized by the neural network. A separate module was designed to recognize dashes that are located, more or less, in the vertical center of the image.

4.3 Image preprocessing

The preprocessing stage is essential to ensure robustness of the system. The goal is to transform the input image to a standard form with minimum noise. We shall describe the preprocessing by following the transformation of an example image. Figures 13b-13e show the preprocessing stages for the zip-code shown in 13a.

Underline removal (13b)

A line underneath the zip-code is very common. We have found that it is beneficial to remove it first since other stages are effected by its existence and it is more difficult to remove at later stages. The line is detected as a strong peak in the pixel projection onto the vertical axis. The removal is governed by the projection profile and by neighboring pixel analysis. In cases where the line is touching the digits above it, two actions may be taken, part of the digits may be lost (in many cases the loss is not enough to disrupt correct recognition) or the line may not be removed.

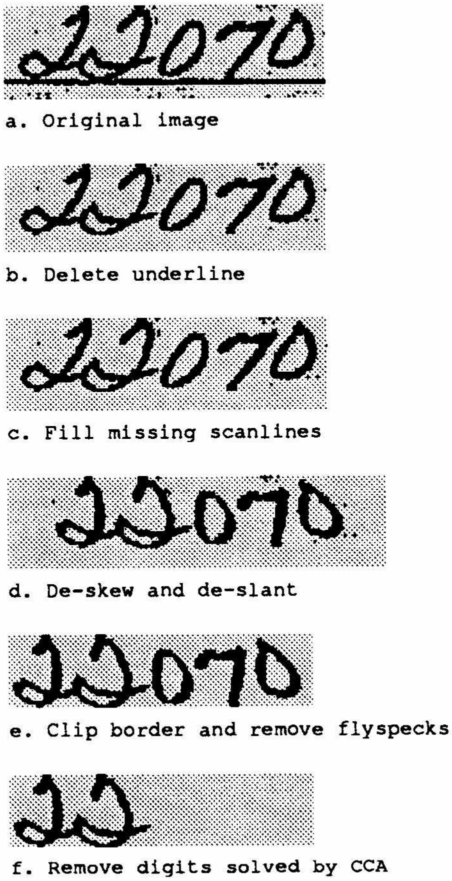
Image enhancement (13c)

In many cases the images captured have missing scan lines. A simple algorithm that fills missing pixels is applied.

Slant and skew adjustment (13d)

Many people tend to write so that the characters are slanted. For most right handed people the **slant** is to the right (figures 8a, 8d, 8f and 8g), in less typical cases the slant is to the left (figure 8b). In other cases there is no noticeable slant (figures 8e and 8h). We wish to de-slant the images so that vertical cuts will be sufficient to separate neighboring characters.

The **skew** of an image in OCR applications is usually a rotation of the original image, and the correction is to align the image's horizontal axis parallel to the internal horizontal axis. This definition of de-skewing is more suited for machine print where one can define a baseline for a line of text. In handwriting the characters need not be on a straight line due to the writer shifting his hand during writing (e.g. figure 8c). In that case a rotation will not place the digits on one line. We wish to transform the image so that the



a. Original image

b. Delete underline

c. Fill missing scanlines

d. De-skew and de-slant

e. Clip border and remove flyspecks

f. Remove digits solved by CCA

Figure 13: Preprocessing stages and CCA.

character will lie more or less on a horizontal line. Our assumption (based on the data we worked with) is that the skew is relatively small, therefore the operation we perform for slant correction, applied along the vertical axis, should supply adequate skew correction.

Slant correction is done by rotating one axis while keeping the other fixed. The best new angle for the vertical axis is found by generating pixel projections for each rotation (see figure 14). The best angle is when the peaks are sharpest, corresponding to the dominant direction of the strokes. To find the corresponding angle we maximize the "entropy" [5] $S = \sum_i p_i ln(p_i)$. Where p_i is the normalized value of the pixel projection at site i.

Noise cleanup (13e)

Flyspecks (small connected components) and intruding marks from other fields are removed. This is done by the CCA, which is presented in the following section, but we consider it part of the preprocessing. Following the cleanup, surrounding white space around the image is clipped.

Parameter estimation

After preprocessing is done several estimates about the image can be made. Using the aspect ration between the width and height of the image the system decides if this is a 5 or 9 digit zip-code. This decision can be reconsidered at various other stages of the process.

Given the number of estimated digits, we can estimate the expected width of each one (the "pitch"). We can estimate the stroke width of the digits by measuring the average number of contiguous black pixels in the horizontal direction (the average "run-length").

4.4 The segmenter

4.4.1 Connected Component Analysis (CCA)

In this process each set of adjoining pixels is grouped to a connected component (CC). Two connected components can be united into one if they are

[5] Maximizing any convex function of p_i would serve the same purpose

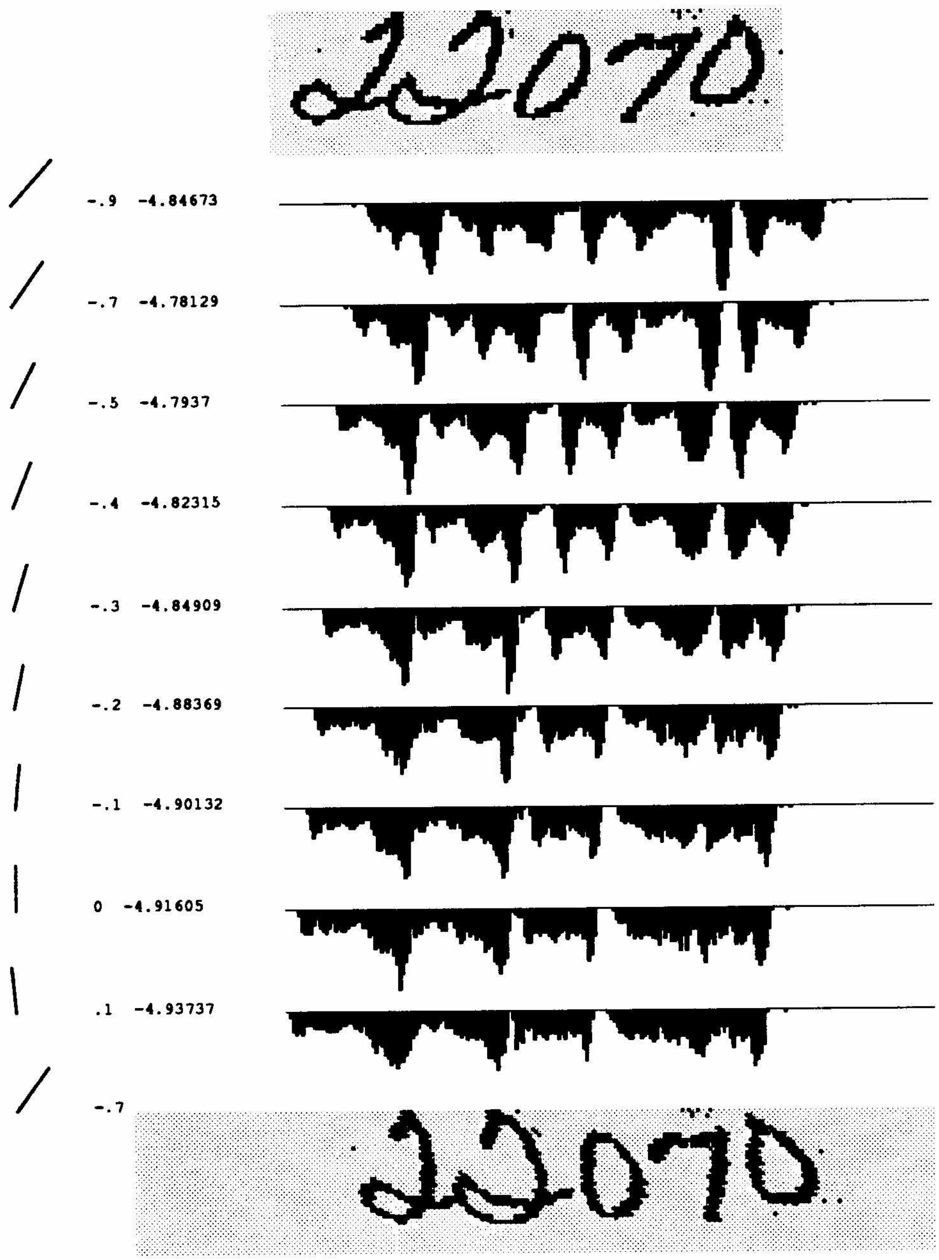


Figure 14: The pixel projections through various angles for the example zip-code. The entropy is printed next to the projection. The image is de-slanted according to the angle (0.7) with the highest entropy. The de-slanted image is shown at the bottom of the figure.

sufficiently close one to another.

Each CC is checked to see if it is too small or too large to be a reasonable digit. If it is very small it is regarded as a "flyspeck" and discarded. If it is too large it probably consists of two or more connected digits and will be handled by the next level of segmentation.

CCs that pass the validity check are evaluated by the recognizer that scores them. Those scored above a certain threshold are designated as solved. The rest are passed on to the next level. Figure 13f shows the remaining parts of an image that have not been solved by CCA. The estimate number of digits determined at preprocessing is re-examined in light of the CCs found.

In the case when all CCs are recognized with scores above threshold and the number of segments completed is 5 or 9 (or 9 and a dash) then the segmentation is concluded.

4.4.2 Vertical cut-point estimation and segmentation

At this stage the segmenter must recognize all parts of the image not solved by the CCA. The CCs marked as solved are erased from the image (see Figure 13f) and the vertical pixel projection of the remaining image is calculated.

The number of remaining digits is determined by subtracting the number of digits recognized by CCA from the estimated number of digits. The and the estimate average pitch for the remaining digits is calculated by dividing the remaining width by the number of remaining digits. The vertical cut segmenter will try to find the best cuts that will give the correct number of remaining digits. The number of necessary cuts is one less than the number of remaining digits. Figure 15 follows the vertical cut segmentation process for the zip-code shown in Figure 8. For the sake of the explanation the CCA segmenter has been de-activated. Had it been active the approach to the correct solution would be much simpler.

The vertical pixel projection of the remaining image is passed through a filter with a decay parameter of $\frac{\textit{projection height}}{\textit{estimated width of stroke}}$. The resulting function is close to value +1 at areas of white space and gives a score close to zero when the projection value is well over the estimated stroke width. We term this function the "prior probability" of cuts. This function is smoothed and the maxima points are located. These will be the candidate cut points.

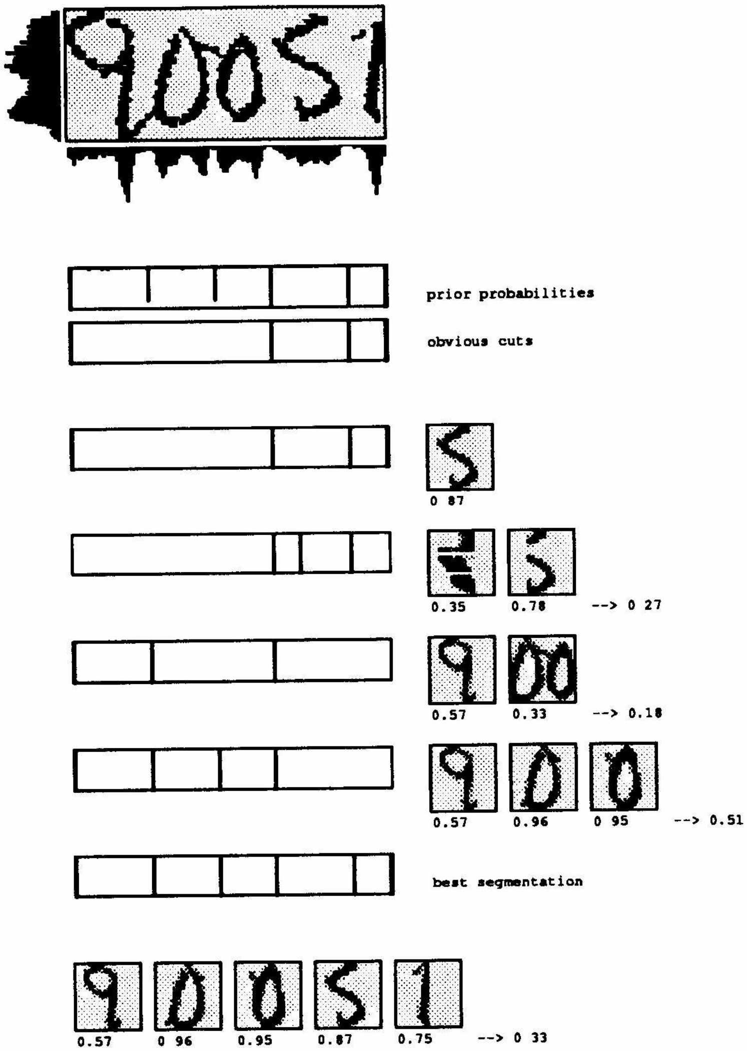


Figure 15: Vertical cut segmentation. The preprocessed image with its horizontal and vertical projections is shown on top. Candidate cuts with related scores inferred from the vertical projection are shown imediately beneath. Though only four are with relatively high scores there are others that are hard to see at this figure's scale. Two of the candidate cuts are marked as obvious, leaving two additional ones to find. Each line shows a various segmentation of the two leftmost segments. The images passed to the recognizer and the resulting scores are shown to the right. The best combined segmentation is given at the bottom.

Each cut-point has a score associated with it.

A candidate cut-point with a prior probability greater than a certain threshold (we have set this to 0.9) is termed an "obvious cut". Cuts that are too close to the sides of the image or one to another are deleted from the set of obvious cuts.

If the number of remaining obvious cuts is equal to the number of necessary cuts then the set of obvious cuts is the solution. If the number is less than needed the system proceeds to find the additional cuts using the recognizer. In the case that there are more obvious cuts than needed, it is assumed that there is a problem in this image and all obvious cuts will be discarded, and the system will proceed to find all missing cuts using the recognizer.

The cuts determined implicitly by the CCA and the obvious cuts divide the image into a number of segments. (if there are no cuts there is only one segment). Using the estimated pitch we can get an upper and lower estimate on the number of digits in each segment. In the the example shown in Figure 15 there are three segments. Therefore two cuts are missing. The first is estimated to have 2 or 3 digits, the second to have 1 or 2 digits, and the third is estimated to have one only. A candidate segmentation is generated for each segment for each possible number of digits in it.

The cuts are chosen in the following way: The prior probabilities of the candidate cuts are multiplied by a Gaussian centered around equi-distant partitions of the segments. The cut with the maximum score near each equi-distant cut is chosen. All these cuts are checked for consistency , that is that they are not too close and that they enclose a reasonable amount of ink. If they are not consistent, another set is generated.

Each of the sub-segments are sent to the recognizer for scoring and the results are multiplied[6] The final result is a score for each segment given it has a certain number of digits within it.

The system then evaluates all combinations of the various sub-segments giving rise to the correct number of digits. The best score is the chosen segmentation. In our simple example there are only two combinations: two cuts in the first segment and none in the second or one cut in each. The first possibility has a higher combined score therefore it is chosen as the answer.

[6] The scores returned by the recognizer are estimated probabilities. We make the assumption that the probabilities are independent and therefore multiplying the individual scores gives an estimated probability for correctness of the digit string.

The process is designed so that the same partial image is not evaluated twice by the recognizer, but the result is kept for other solutions encorporating the same sub-solution.

In many cases more possibilities are explored than in the example shown. Still due to the fact we use both CCA and obvious cuts, enables the system to prune the search and explore more possibilities for the remaining cuts. The overall number of calls to the digit recognizer is small (in our example 9 call where made). For 5 digit zip-codes the average number of calls is 10. This is a factor of two over the minimum number of calls needed (If we knew a priori what the correct segmentation was).

We have added a small amount of contextual processing by comparing the first five digits in the resulting answer to a zip-code validation table. In case the resulting zip-code is not valid, the segmenter will generate the next highest ranking interpretation, until a valid zip-code is found.

5 Segmentation & recognition performance results

The total system was tested on 2585 image of 5 and 9 digit zip-codes. The results for various tests are summarized in Table 1, The best performance this system has achieved without use of the zip-code table is 73.85% correct and 26.15% incorrect.

In a real application, errors are traded for rejections, according to the cost of an error versus the cost of a rejection. In the postal world this translates to the cost of sending the mail to the wrong post office versus the cost of a human looking at the envelope in question. A cost model is usually much more complex since the error cost depends on the error itself. For example an error in the first digit of the zip-code is more "serious" than one in the last digit.

We have not tried to incorporate cost into our rejection scheme. The rejection parameter we use is a function of the digit recognizer scores. The score assigned to a recognized image is the multiplication of all the individual digit scores. In order to reject both the 5 digit and 9 digit zip-codes on the same scale, the rejection parameter was set to the 5th or 9th root of the zip-code score. A benchmark we have used is the error rate when we reject 40% of the lowest scoring images. The error rate of the remaining images at this rejection rate was was 5.47%. ($error\ rate = errors/(1 - reject\ rate)$).

For the postal application the crucial digits are the first five. When the measure of correctness is only on the first five digits, the raw error rate (errors when none where rejected) decreases to 24.53

The zip-code table was used to discard illegal answers. The raw error when the validation table was used, was 23.83% and was 3.22% at 40% reject. Given the fact that there are roughly only 40% legal 5 digit zip-codes out of the possible 10,000 one would expect a greater change.

An interesting question is what is the contribution of the hybrid segmentation approach. We have already stated that during the design we took into consideration that CCA could solve about 35% of the cases. What would be the performance of the vertical cut segmenter if the CCA component was deactivated ? After checking this case we found the raw error to be 35.59% and 6.95% at 40% reject.

We have analyzed the errors with the highest scores. About 50% are recognition errors, 30% are segmentation errors and the remaining errors are due to bad preprocessing. These numbers are for the highest scoring errors the distribution is probably different over all errors, but since low scoring patterns are rejected, we are less interested in them.

The recognition errors are cases where the segmentation into individual digits were correct, but at least one digit was misclassified. Some cases are ambiguous even to the human eye. Others are due to lack of resolution or training data.

The segmentation errors are cases were the zip-code was incorrectly segmented. Since most of the segmentation is governed by the recognizer - this is also a problem with the recognizer giving reasonable scores to wrong segmentations.

The preprocessing errors are cases were the preprocessor was unable to remove noise or intruding marks, that caused the segmenter or recognizer to err. On the other hand there are cases where important parts of the image were incorrectly removed as noise.

system	raw error	error at 40% reject
whole zip-code	26.15	5.47
first 5	24.53	4.64
first 5 + zip directory	23.83	3.22
first 5 , zip directory , no CCA	35.59	6.95

Table 1: Raw errors and errors at 40% rejection for various versions of the system.

6 Implementation

The system was initially prototyped on a Sun4 workstation using the neural network simulator SN2[11]. The system was then developed on a C++ test bed version on a Sun4. The code has been ported with minor changes to run on a AT&T DSP32C resident on an ARIEL DSP32C board that resides on a AT&T PC 386 bus.

The system's average time per zip-code image is 3.1 seconds. The processing time is approximately shared between preprocessing, segmentation and recognition . However, the code is not fully optimized and we expect it to be able to run at a higher rate. The total time (especially preprocessing) is dependent in the size of the input image. The preprocessing time could be reduced considerably with a modest degradation in recognition performance, if the input images are reduced in size scale.

7 Summary & future work

We have presented a system that recognizes unconstrained strings of digits at state of the art performance levels. The system is is based on what we term "recognition based segmentation" The segmenter's decision is governed by the scores returned by the recognizer for the individual segments. The interpretation chosen is the one that give the best **global** score.

This scheme is not necessarily computationally expensive. If the candidates evaluated are chosen carefully the additional computation can be small. We have chosen a hybrid solution of connected components analysis and vertical cuts to generate the candidate segmentation set. The average call to the recognizer in the system is roughly a factor of two of number of digits in the image.

This is a first stab at this problem and can, with no doubt be improved. We have identified that the majority of high scoring errors was due to misclassifications of the recognizer. Many of the segmentation errors are also due to the recognizer. It is quite clear that we need classifiers with performance exceeding approximately 95% correct at the single digits level.

We have chosen recognizing unconstrained digits as a first problem, because they are the easiest problem in their class. Our goal is to recognize unconstrained alpha-numeric handwriting. The methods described in this paper are dependent on various heuristic that are hard to optimize together. It is not clear if they can be extended to more general problems.

We are currently exploring techniques that rely less on heuristics to generate the cuts and are more strongly coupled to the recognizer during training. These methods use a Maximum A Posteriori approach to reach the best interpretation. Preliminary results of combining our new approach with the one described above have shown remarkable improvement. The raw error rate has gone down to 19% and the error at 40% reject is less than 1.5%.

Acknowledgements

This work has been partially funded by the U.S. Postal Service, task order 104230-90-C-2456.

References

[1] H. S. Baird. Anatomy of a page reader. In *IAPR Workshop on Machine Vision Applications*, (Tokyo, 1990), 1990.

[2] G. Nagy. Optical character recognition - theory and practice. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of statistics, Volume II: Classification, Pattern Recognition and Reduction of Dimensionality*. North Holland, 1982.

[3] C. Y. Suen, editor. *Frontiers in Handwriting Recognition*. CENPARMI, Concordia University, Montréal, 1990.

[4] R.O. Duda and P.E. Hart. *Pattern Classification And Scene Analysis.* Wiley and Son, 1973.

[5] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Processing*, ASSP-26(1):43–49, 1978.

[6] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5:179–190, 1983.

[7] T. K. Ho, J. J. Hull, and S. N. Srihari. Combination of structural classifiers. In *IAPR Workshop on Syntactic and Structural Pattern recognition*, pages 123–136, (Murray Hill, NJ, USA), 1990.

[8] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and H. S. Baird. Handwritten zip code recognition with multilayer networks. In *Proceedings of the 10th International Conference on Pattern Recognition*, (Atlantic City, NJ, 1990), 1990. IEEE Computer Society Press.

[9] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs with relationships to statistical pattern recognition. In F. Fougelman-Soulie and J. Hérault, editors, *Neuro-computing: algorithms, architectures and applications.* Springer-Verlag, 1989.

[10] O. Matan, R. K. Kiang, C. E. Stenard, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and Y. Le Cun. Handwritten character recognition using neural network architectures. In *Proceedings of the 4th United States Postal Service Advanced Technology Conference*, volume 2, (Washington D.C., 1990), 1990.

[11] L.-Y. Bottou and Y. Le Cun. *SN2: A Simulator for Connectionist Models.* Neuristique SA, Paris, France, 1989.