Application Of Mathematical Morphology To Handwritten ZIP Code Recognition

Gillies, Andrew, Gader, Paul, Whalen, Michael, Mitchell, Brian

Andrew M. Gillies, Paul D. Gader, Michael P. Whalen, Brian T. Mitchell, "Application Of Mathematical Morphology To Handwritten ZIP Code Recognition," Proc. SPIE 1199, Visual Communications and Image Processing IV, (1 November 1989); doi: 10.1117/12.970049

Event: 1989 Symposium on Visual Communications, Image Processing, and Intelligent Robotics Systems, 1989, Philadelphia, PA, United States

Application of mathematical morphology to handwritten ZIP Code recognition

Andrew M. Gillies, Paul D. Gader, Michael P. Whalen, Brian T. Mitchell

Environmental Research Institute of Michigan
3300 Plymouth Road, Ann Arbor, Mi. 48107-8618

ABSTRACT

This paper describes applications of mathematical morphology to a system for recognizing handwritten ZIP Codes. It discusses morphological techniques used for preprocessing address block images, locating address block lines, splitting touching characters, and identifying handwritten numerals. These techniques combine mathematical morphology, hierarchical matching of object models to symbolic image representations, and a strategy of propagating multiple hypotheses. The various submodules of the system have been trained on over two thousand real address block images and tested on one thousand representative images. On the one thousand test images, the system correctly located 82.5 percent, correctly identified 45.6 percent, and incorrectly classified only 0.8% of the ZIP Codes. This system performance level could lead to a significant cost savings in mail piece sorting.

1. INTRODUCTION

The United States Post Office processes some 154 billion pieces of mail each year. Approximately 15% of this volume, or roughly 23 billion mail pieces, is handwritten. Postal facilities in larger cities are equipped with automated OCR (Optical Character Recognition) machines to read the ZIP Code of each mail piece and sort it accordingly. These machines perform well on typewritten mail but have difficulty with handwritten pieces. Mail pieces which are rejected by these machines and all mail pieces in postal facilities without these machines must currently be manually sorted.

This paper describes applications of mathematical morphology to ongoing research in the development of an experimental system for recognizing unconstrained handwritten ZIP Codes [1,2]. The primary goal of this research is to develop a system which correctly identifies a maximal number and incorrectly identifies a minimal number of unconstrained handwritten ZIP Codes in gray level address block images. The major issues involved with this research are segmentation, automatic location and recognition of ZIP Codes. The location of ZIP Codes involves such problems as: slanted address block lines; numerous address block formats and the resulting variability in ZIP Code location; five- and nine-digit ZIP Codes; touching, overlapping, and intersecting ZIP Code digits; and two-stroke ZIP Code digits. The recognition of ZIP Code digits involves such problems as two-stroke digits; broken digits, ill-formed digits, serif-ornamented digits, and multivariation digits.

The system described in this paper includes techniques for preprocessing address block images, locating address block lines, locating ZIP Codes, splitting touching characters, and identifying handwritten numerals. These techniques are developed in a framework that is based on a combination of mathematical morphology and object-oriented, model-based programming. The algorithm development environment used to construct the system is built in a multiprocessor environment utilizing the pixel-processing pqwer of the ERIM CytoComputer [3,4] and the symbolic processing power of a Lisp machine.

2. SYSTEM OVERVIEW

An overview of the handwritten ZIP Code system is shown in Figure 2-1. As seen in this figure, the system consists of six processing phases: 1) preprocessing to transform the input gray scale address block image into a state labeled image; 2) locating lines within the address block; 3) locating ZIP Codes within address block lines; 4) segmenting ZIP Code digits; 5) recognizing ZIP Code digits; and 6) assembling ZIP Code information to produce a system result.

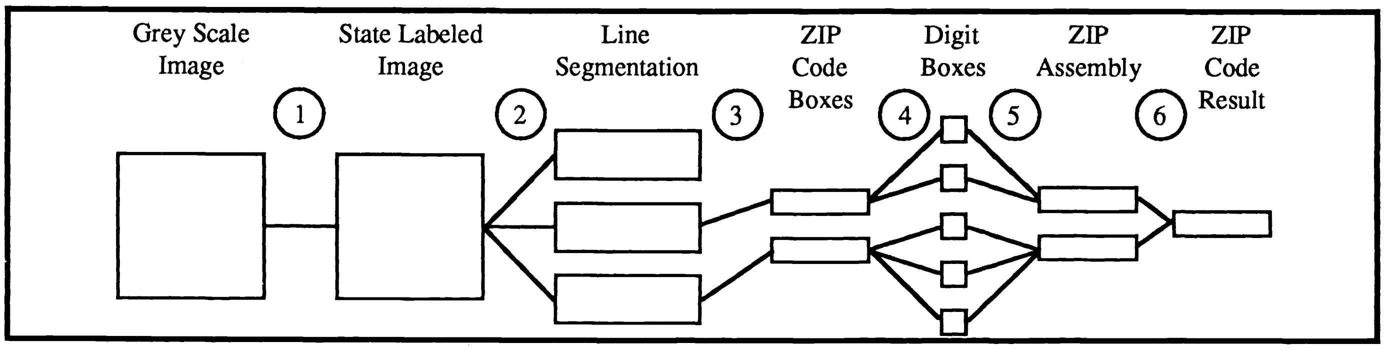


Figure 2-1. System Overview

The first phase, preprocessing, takes a gray scale address block image and uses morphological operations to perform background normalization, adaptive global thresholding, image cleaning, and stroke skeletonization. The resulting image is called a state-labeled image, since the pixel states in this image identify strokes and stroke skeletal features.

The second phase, line location, calculates eight-connected component features from the state-labeled image and groups them into address block lines represented as objects in the object-oriented environment. The third phase, ZIP Code location, takes the address block lines as inputs and locates the ZIP Code within those lines. The only use of morphology in this phase is in the calculation of bottom extremal points which are used to help detect the address block lines.

The fourth phase, digit segmentation, takes potential digit stroke objects and calculates where splitting might be needed to create isolated digit objects. Five splitting techniques are applied to each component: underline removal, weak-touch, cusp, non-horizontal bar, and horizontal ray. The underline removal process consists of a sequence of morphological operations designed to remove underlines that intersect digits within the ZIP Code without removing the bottom portion of the digits. The weak-touch splitter separates digits which touch through eight-way connectivity. The cusp, non-horizontal bar, and horizontal ray operations use combinations of morphology and heuristics to detect local handwriting stroke characteristics that indicate touching characters. Multiple hypotheses are calculated on each component based on the number of digits recognized thus far, the number of components sent to the splitting subsystem, and the ZIP Code length hypothesis (i.e. five or nine). The result of this processing phase is an enriched set of potential digit objects.

The fifth processing phase, digit recognition, analyzes each potential digit to see if it matches handwritten digit models. Initial recognition features, called cavity features, are generated using mathematical morphology. For potential digits that resulted from digit segmentation (or splitting), a tree-like structure is maintained throughout this process to retain the relationships among the original stroke object, the split objects, and their digit recognition results. For potential digits that contain an ancillary component in their immediate neighborhood, digit recognition is performed with and without the ancillary component to cover both the hypothesis that the potential digit requires grouping and that it does not. Morphological tests are used throughout the digit matching process. The sixth and last processing phase, ZIP Code assembly, takes the digit recognition results and attempts to piece together a best-guess ZIP Code decision.

The following sections describe applications of morphology in the preprocessing, digit splitting, and digit recognition phases of the system and results of experiments that evaluated selected component and system performance.

3. PREPROCESSING

The ZIP Code reading system starts with a gray scale image of the address block from the mail piece. This image must be processed to segment the characters of the address block from the background of the envelope. The segmentation process must take into account variations in background white level, writing implement characteristics, and possibly textured backgrounds. The image preprocessing routines produce a binary image of the segmented characters. The process also performs a skeletonization of this binary image, producing features to be used by the character splitter. The preprocessing described in this paper is based on morphological operations and is implemented on the ERIM Cyto HSS. The steps in the raw image preprocessing are described below.

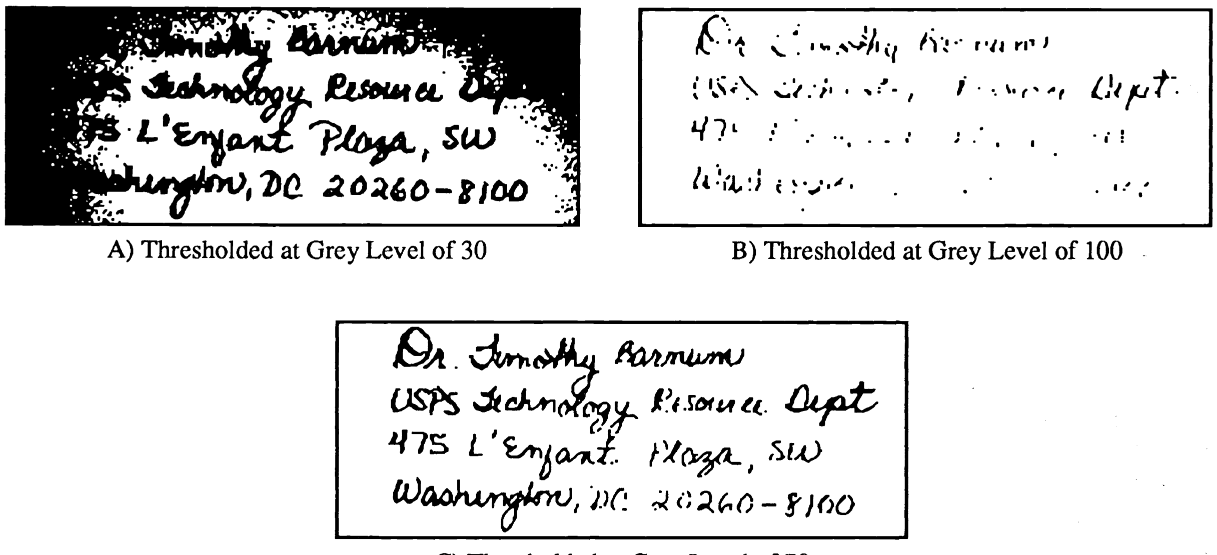
The first step in image preprocessing is normalization of the background white level. This step removes variations in background white level from mail piece to mail piece. It also removes any smooth gradients that exist in the background white level within a single image. Background normalization is done by first estimating the white level and then subtracting this level from the raw image. The white-level estimation is accomplished by closing the raw image with a cylindrical structuring element of radius 20. The result is an inverted image, in which the characters appear in lighter gray scale values against an essentially black background.

The segmentation of characters from background is accomplished by applying a global threshold to the white-level normalized image. This threshold is selected automatically by an adaptive method.

The adaptive threshold selection method uses ten preselected threshold values as candidates. (For our images, with gray-level values in the range 0 to 255, the ten preselected thresholds were 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100.) The image is sliced at each of these ten levels, and the resulting binary image is analyzed for noise content. The image with the least noise content is selected.

The noise content measurement is done by finding components in the image that are small enough to be removed by five passes of skeletonization (thinning). This skeletonization is done in such a way that endpoints are removed from totally thinned lines. Thus, any component which is shorter and thinner than ten pixels (twice the number of skeletonization passes) is labeled as noise. The total area of these noise components provides a measure of the global noise content of the image. Figure 3-1 illustrates this process on a sample image at three levels of thresholding. Part A of the figure shows the image thresholded at too low a level. The noise measurement would be high in this image due to the large connected components around the border of the image. Part B shows the result from thresholding an image at too high a grey level. In this image, the actual stroke components are severed into multiple pieces which contribute to a high global noise content. Finally, part C of the figure shows the result from thresholding the image at the correct threshold. The overall noise measurement is low for this image due to the absence of large connected components present in part A. Also, the stroke components do not contribute to the noise measurement in this image since they are too long to be removed by 5 passes of thinning. Note that skeletonization, rather than erosion, must be used to detect noise components because the strokes themselves are thin.

The next step in preprocessing is to clean up the binary image produced by the thresholding step. First, vertical and horizontal components longer than 150 pixels are removed. Then connected components that are too small to contain the five-point elemental neighborhood are removed using erosion.


A) Thresholded at Grey Level of 30

B) Thresholded at Grey Level of 100

C) Thresholded at Grey Level of 70

Figure 3-1. Adaptive Global Threshold

The next step in the preprocessing is skeletonization. In this step, the skeletal components of each binary feature are marked in the image. The skeletonization is performed so as to preserve eight-way connectivity. Once the skeleton itself is formed, the junctions between thinned components and the endpoints of thinned components are marked. The resulting image has five states: background, thinned components, endpoints, junctions, and "flesh." The flesh is that part of the binary image removed during skeletonization. The binary image itself can be easily reconstructed from the skeletonized image by taking the union of the thinned components, endpoints, junctions, and flesh regions (i.e., everything but the background). This five-state image forms the input to the ZIP Code location module.

The final step in the preprocessing is the calculation of the bottom extremal points. This is accomplished using the structuring elements R = { (0 ,0) ,(0 -1)} and B = { (-1,0) (0,0) ,(1,0) } in the following operations:

(a) Inew = Iold - (Iold $\ominus$ R)

(b) Iout = Ispan-until ((Iold $\ominus$ R) Inew B) where Ispan-until(A B S) denotes the successive application of $(A \oplus rS) \cap B$ for r = 1,2,... until $(A \oplus rS) \cap B - (A \oplus (r-1)S) \cap B = 0$. This formulation allows for finding the "local minima" on a stroke while avoiding those local minima that are not at the bottom of the stroke. These bottoms extremal points are very useful in locating the bottoms of address block lines. Computation of the bottom extremal points is illustrated in Figure 4. The initial bottom extremal points (the white pixels in the middle 2) are found by operation (a). Those that are not truly bottom minima are then deleted by spanning the black pixels over the white pixels with a horizontal bar B.

Once the preprocessing is completed, the system locates the address block lines and then one or more regions in the image that could contain the ZIP Code. These regions are then passed to the digit splitting and recognition modules for further analysis. In the next sections, we discuss how morphology is used as a tool for splitting and recognition.

4. DIGIT SPLITTING

One of the many difficulties encountered by the OCR machines in tackling handwritten address blocks is the occurrence of two or more isolated digits being merged into one connected component. This parent component must be dissected into its associated children before the children can be correctly recognized. The subsystem designed to accomplish this task consists of two main phases. The first phase is composed of three splitting modules that work independently on a candidate component. Their sole concern is to locate plausible areas on the component where it could be split into two children. They use heuristically determined tests to assign a confidence value to each candidate split. The second phase of the segmentation subsystem is the control module which accepts the low level splits and propagates forward the most likely children.

Several key issues were addressed in the design of this subsystem. First, it has the capability of dissecting a component into more than two children. Second, the lower levels of the system are a parallel construction of morphological splitters, each targeted at a particular class of connected digits. Third, the subsystem propagates forward multiple splitting hypotheses so that the correct dissection has a high probability of being considered. Finally, the system is designed so that other low level dissection submodules can be developed independently and easily inserted into the overall segmentation subsystem.

4.1 Control module

The control module accepts two sets of information; the hypothesized splits returned from the low level morphological splitters, and auxiliary information generated by digit recognition on the initial components in the potential ZIP code box. This auxiliary information includes how many more digits must be recognized in order for a ZIP code hypothesis to be completed, how many components are available to be split, and how many split hypotheses the system would like to propagate forward. It uses the auxiliary information to create a target list. Each element of the target list represents the number of children the system will attempt to split a parent component into. The length of the target list represents the number of hypotheses the system would like to propagate forward.

For elements of the target list that equal 2 (i.e. require the parent component to be split into 2 children), the available split with the highest confidence is taken. However, since the three splitting modules only return locations where the parent can be split into 2 children, for elements of the target list greater than 2, the split must be constructed from more than 1 split location. In this case, the split is constructed using the top most available splits whose intersect area is zero.

4.2 Morphological splitting modules

The first phase of splitting connected components consists of three splitting modules that work independently on a candidate component. Each of these modules searches for plausible locations where the component could be split into two children components and associates a confidence value with each location. This confidence value is based on feature and image domain properties of the potential split. Feature domain characteristics that affect the confidence of a split include attributes such as height, width , and location. Image domain properties affecting confidence include the component area in the vicinity of a split and the height ratio of the resulting children components. These properties were ascertained heuristically through visual examination of the characteristics of both correct and incorrect splits.

4.2.1 Cusp splitter

The cusp splitter is designed to separate rounded digit strokes that touch. It uses matched filters to locate cusp features, namely, vee shaped areas in the stroke which border on background. Figure 4.2.1-1 shows a pictoral representation of the cusp splitter. The top row of the figure shows the foreground, background, and combined structuring elements that are used to detect a top cusp feature. Similarly, the second row shows the structuring elements used to locate bottom cusp features. The bottom cusp feature structuring elements are merely the top cusp structuring elements inverted and vice-versa.

The latter part of Figure 4.2.1-1 shows actual sample instances of connected components that were correctly dissected by the cusp splitter. When a potential pair of a top and bottom cusp are identified by heuristic tests, the actual dissection is made by dilating an image of the top cusp in the direction of the bottom cusp and intersecting this image with the original image. This process is shown in part C of the figure. The final part of the figure demonstrates how the individual stroke components are recovered once a split has been made. To recover the leftmost component, the break image is first spanned over the stroke image to the right to establish a "barrier" which opposes the recovery of any stroke component to the right of the break. The break image is then spanned repeatedly over the difference between the original image and the barrier image until the entire left component is recovered.

4.2.2 Horizontal ray splitter

The Horizontal Ray splitter is designed to dissect digits that are connected through a three-way junction with one of the digits coming in to the junction at a ninety degree angle. Junctions are one of the skeletal features calculated using morphology during preprocessing and are locations on a component where three or more thinned strokes intersect. A pictoral overview of this splitter is shown in Figure 4.2.2-1.

The splitter uses morphology to first detect regions in the component where a vertical bar will fit. The length of this bar is equal to four times the component's stroke width. Stroke width is computed using a morphological thinning operation. These regions are intersected with regions of the component where a three stroke width ray will fit. Initially, it was required that these two regions intersect in the vicinity of a junction feature of the component's skeleton, in order for the junction to become a candidate for dissection. In the splitters current confidence-based configuration, these two areas merely boost the junction's confidence for possessing a correct horizontal ray split.

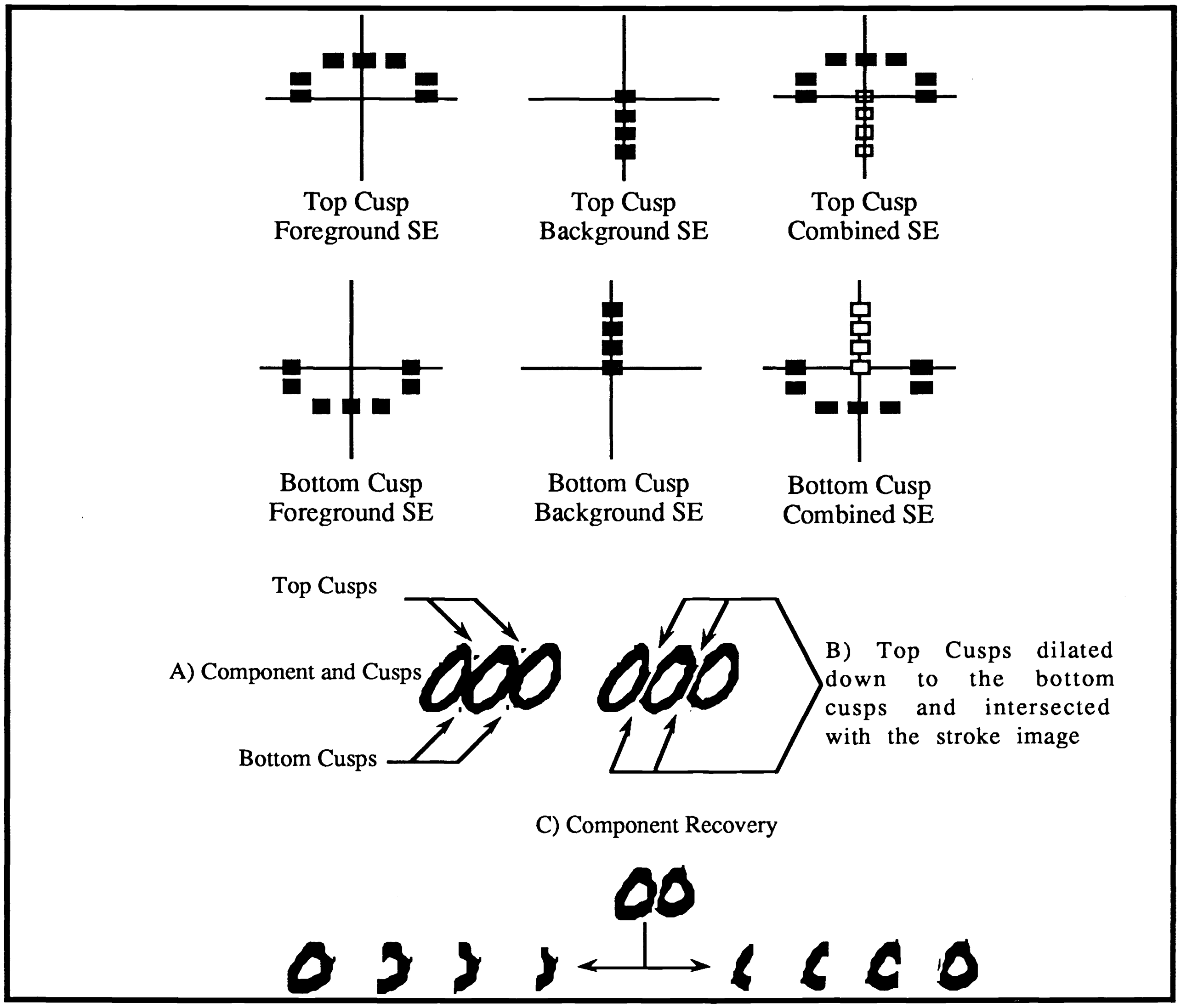


Figure 4.2.1-1. The Cusp Splitter

Once a decision has been made to dissect at a candidate junction, the splitting is accomplished by moving over approximately one stroke-width from the junction in the direction of the horizontal ray and snipping the component with a one pixel wide vertical bar. The height of the bar is allowed to reach slightly more than two stroke widths.

4.2.3 Non-horizontal bar splitter

Similar to the Horizontal Ray splitter, the Non-Horizontal Bar splitter is designed to separate digits that touch through a three-way junction. However, as its name suggests, it is aimed at dissecting components at non-horizontal angles. Specifically, in its current design, the splitter requires that a junction have either a 45, 90, or 135 degree bar of length equal to 4 stroke widths fit in its general area in order for the junction to be considered a candidate for dissection.

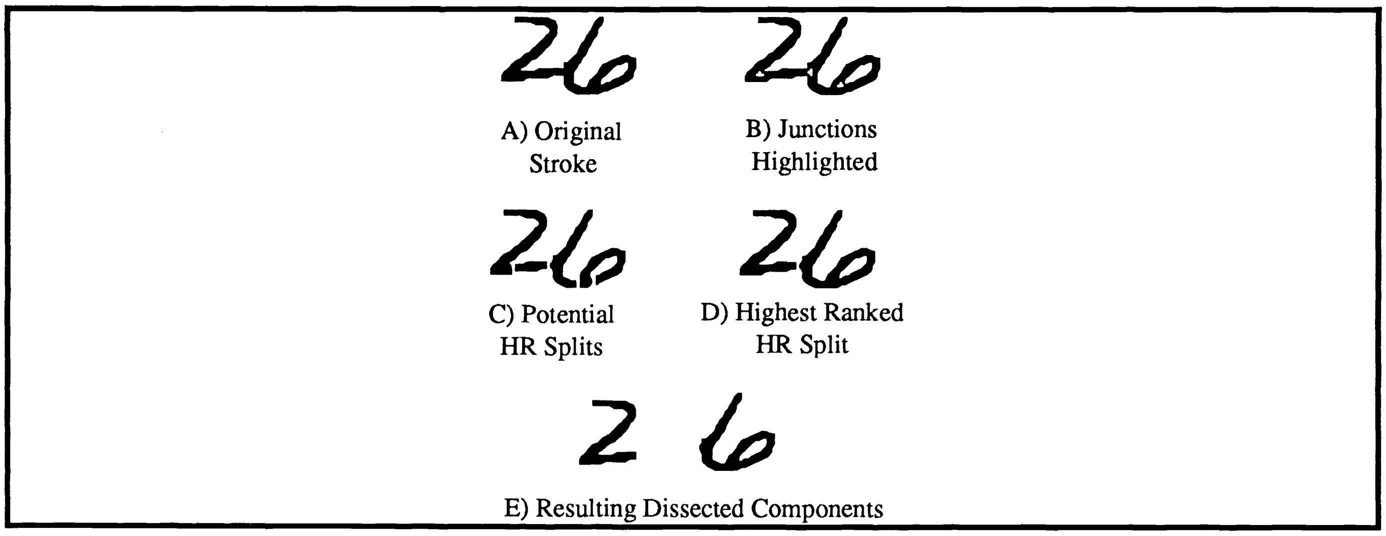


Figure 4.2.2-1 Horizontal Ray Splitter

Figure 4.2.3-1 shows the operation of this splitter. Parts A of the figure show a candidate stroke component for this splitter and part B shows the junctions found during the morphological preprocessing of the address block image. Part C of the figure shows how the parent component is split into two children once a junction has been targeted. The method of dissection begins by taking a dissection bar that is sloped at the same angle as the bar that fit into the candidate junction and intersecting it into the feature. It then moves the dissection bar away from the candidate junction and intersects it with the stroke again. This process is repeated until the dissection bar is approximately one stroke-width away from the junction. The spot where the intersection of the dissection bar and the feature yield the least area is chosen for the break location. This is the dissection area shown in C1. This area is sent to the background producing the final features shown in part D.

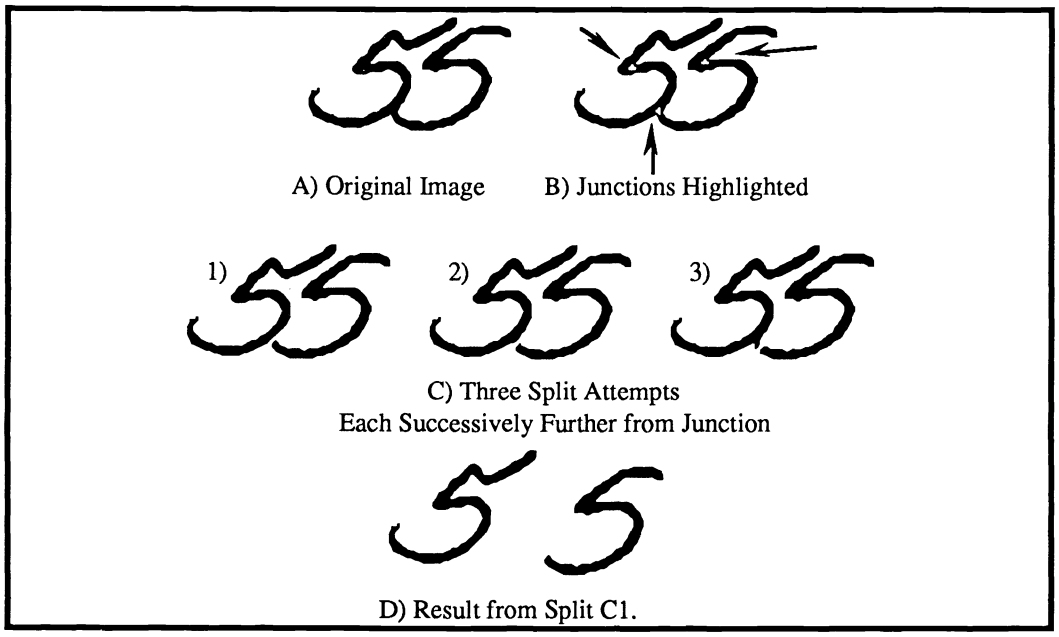
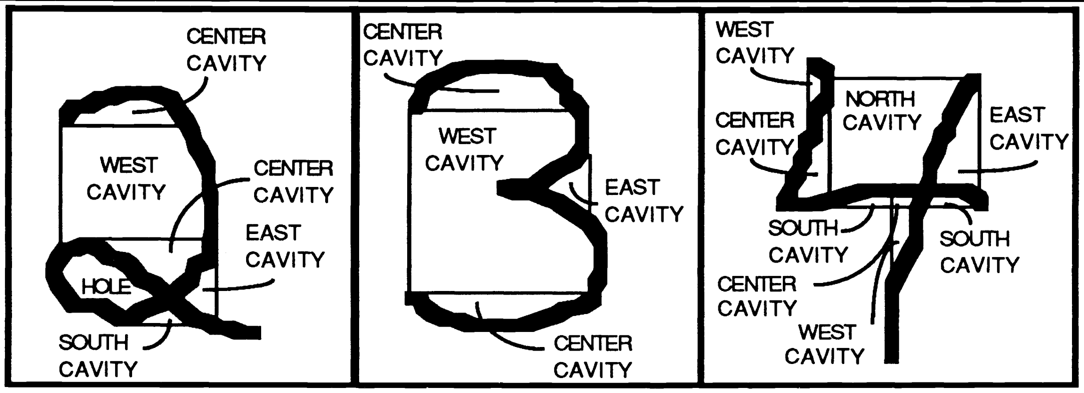
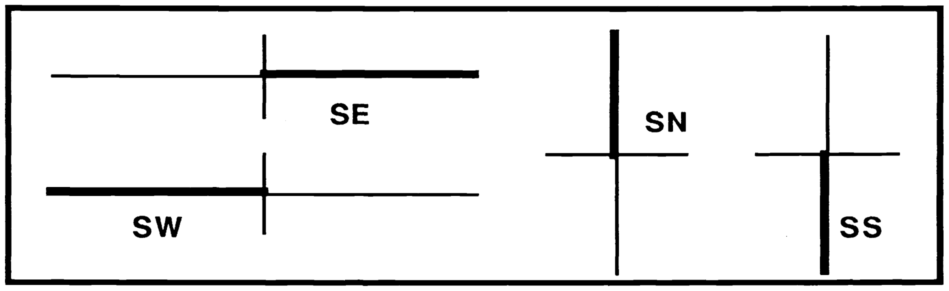


Figure 4.2.3-1 Non Horizontal Bar Splitter



DETECTED CONCAVITY FEATURES



STRUCTURING ELEMENTS

$$\text{NORTH-CAVITY} = (I \oplus N) \cap \overline{(I \oplus S)} \cap (I \oplus E) \cap (I \oplus W) \cap \overline{I}$$

$$\text{SOUTH-CAVITY} = \overline{(I \oplus N)} \cap (I \oplus S) \cap (I \oplus E) \cap (I \oplus W)\, \overline{I}$$

$$\text{EAST-CAVITY} = (I \oplus N) \cap (I \oplus S) \cap (I \oplus E) \cap \overline{(I \oplus W)} \cap \overline{I}$$

$$\text{WEST-CAVITY} = (I \oplus N) \cap (I \oplus S) \cap \overline{(I \oplus E)} \cap (I \oplus W) \cap \overline{I}$$

$$\text{HOLE} = \overline{I} \cap \overline{\text{SPAN[BORDER}, \overline{I}, \text{4-WAY]}}$$

$$\text{CENTER-CAVITY} = (I \oplus N) \cap (I \oplus S) \cap (I \oplus E) \cap (I \oplus W) \cap \overline{I} \cap \overline{\text{HOLE}}$$

MORPHOLOGICAL EXPRESSIONS
FIGURE 5-1 - Concavity Feature Detection

5. DIGIT RECOGNITION

The first step in isolated digit recognition is feature detection. Feature detection is performed in the image domain by applying mathematical morphology to images consisting of potential digits. The resulting features are then represented as feature objects and a model-based approach is used to match the collection of feature objects (potential digits plus recognition features) to digit models. The digit recognizers use cavity features, as depicted in Figure 5-1, as their starting input. Other features can be detected during the actual model matching process.

Figure 5-1 illustrates the cavity features and their detection. A cavity is a region of points surrounded by the stroke on at least three sides. The cavities are named by the direction in which they open, that is, the side on which they are not surrounded. A center cavity is a region that is surrounded on all four sides but is not a hole. A hole is a region completely enclosed by the stroke.

The cavity features are detected using the tools of mathematical morphology The image that results from the feature detection phase has seven states: background, stroke, north cavity, south cavity, east cavity, west cavity, center cavity, and hole.

The recognition of isolated digits is performed by the same model matcher used by the model-based location module. The matcher uses specially constructed models which describe the requirements for each of the ten numerals (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9). Each of the ten numeral models is painstakingly crafted using an iterative refine-and-test methodology. The models are used by a Lisp function called match. The match function takes two arguments-a model, and a feature list. The function returns nil if the feature list does not match the supplied model. Otherwise, it returns a binding structure which associates each of the features in the feature list with a variable (or slot) of the model.

6. RESULTS

This system was trained on over 2000 images and tested on over 1000 untrained images. The results can be summarized as follows. For overall performance, the system correctly processed 45.7%, rejected 53.5%, and incorrectly classified a false ZIP Code only 0.8% of the time. For locating unconstrained ZIP Codes in these images, the system achieved an 83.1% detection rate, a 6.0% reject rate, and a 10.9% false alarm rate. The system correctly segmented the digits in the ZIP Codes 88.8% of the time. And, finally, on the satisfactorally segmented digits, the recognition subsystem correctly classified digits 88.7% of the time, rejected 10.8% of the digits, and incorrectly classified only 0.5% of the total samples.

7. ACKNOWLEDGEMENTS

This work was funded by the Office of Advanced Technology, United States Postal Service under contract 104230-86-H-0042.

8. REFERENCES

[1] Mitchell, B. T. and Gillies, A. G. A System for Feature-Model-Based Image Understanding, *VISION '87*, 81-90, June 1987.

[2] Gillies, A.M. and Mitchell, B.T. A Model-Based Approach to Handwritten Digit Recognition, *Machine Vision and Applications*, (to appear).

[3] Lougheed, R.M. and Sampson, R.E. 3-D Imaging Systems and High-Speed Processing for Robot Control. *M. Vision and Applications* 1:41-57, (1988) .

[4] Lougheed, R. M., McCubbrey, D., and Jain, R. Applying Iconic Processing in Machine Vision. In Process.