Segmentation and Recognition of Handwritten Postal Codes

Michel César

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements for the Degree of Master of Computer Science at Concordia University
Montréal, Québec, Canada

February 1988

ISBN 0-315-49071-3

ABSTRACT

Segmentation and Recognition
of Handwritten Postal Codes

Michel César

In this thesis, we present a method for segmenting handwritten postal codes. Our proposed method uses a contour-tracing algorithm in order to extract the constituent patterns making up the image of a postal code. Our method does not only segment the postal code, but also reveals important features for each of the constituent patterns. Adopting Glucksman's characteristic loci approach, these features were used to recognize the postal codes. Our segmentation method was experimentally tested and found to be robust and effective.

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my supervisor, Dr. Rajjan Shinghal, who has contributed his time, ideas, suggestions, criticisms, and encouragements for the achievement of this work.

I am also indebted to other people for their contribution in this work. I would like to thank Mr. Safwan Kabbara, my software manager, for his encouragement and permission to use the Corporation's facilities in developing this thesis; my colleague Antoine Taza, of Canada Post Corporation, for his providing the contour-tracing algorithm program; Mr. Mike Yu, of Concordia Computer Science, for digitizing the experimental data.

Finally, I wish to thank my dearest friend, Sonia Coutu, whose moral support helped in the achievement of this thesis.

<u>Table of Contents</u>

Table of Contents (cont'd)

Table of Contents (cont'd)

List of Abbreviations

CPC	Canada Post Corporation
CR	Characteristic Rectangle
FSA	Forward Sortation Area
LDU	Local Delivery Unit
NEC	Nippon Electric Company
OCR	Optical Character Reader
PR	Pattern Rectangle
TSSRS	Two-Stage Segmentation and Registration Scheme
USPS	United States Postal Service

List of Definitions

Characteristic loci: feature extraction technique based on white to dark transitions in a specified direction in a pattern.

Classifier: component of a pattern recognition system responsible for assigning an unknown pattern to the pattern class it belongs.

Expanse: space required by a postal code.

Feature extractor: component of a pattern recognition system responsible for extracting features from a pattern.

Fragment: set of connected dark pixels in a postal code image.

Framentation threshold: assigned value for maximum permissible number of fragments in a postal code image.

Image: binarized representation of a postal code.

Irregularity: in the context of this thesis, irregularity refers to one of three commonly observed problems in binary patterns of postal code images; namely, broken, touching, and overlapping patterns.

Locus: ternary code associated with a pattern's white pixel.

Pattern: binarized representation of a character within a postal code image.

Pitch: average width of a character pattern.

Preprocessor: component of a pattern recognition system responsible for "cleaning-up" the image.

Salt-and-pepper noise: isolated white pixels (salt) and isoladark pixels (pepper) that may be present in a postal code image.

<u>List of Definitions (cont'd)</u>

Segment: the segmentation procedure partitions a postal code image into entities called segments.

Segmentation: the procedure by which adjacent character patterns, in a postal code image, are isolated from each other.

Size normalization: the procedure by which a pattern is expanded or reduced in order to make it a standard size.

Skeletonization: the procedure by which a pattern is transformed into a line drawing one pixel thick, the object being, to capture the shape of the pattern.

List of Figures

List of Figures (cont'd)

List of Figures (cont'd)

List of Tables

CHAPTER I

INTRODUCTION

In this chapter, we first explain the purpose of postal codes. We then examine some fundamental notions as it pertains to segmentation of handwritten postal codes. A historical survey of segmentation is then presented. Finally, we outline the contents of this thesis.

I.1 Purpose of Postal Codes

A country may have more than one village, town, or city with the same name. For instance, in the U.S., the state of New York has two towns named Jericho. Similarly, in Canada, the province of Québec has two places named St-Bruno. If a piece of mail carries as part of its address St-Bruno, Québec, it then becomes difficult for a sorter in the post office to decide to which St-Bruno the mail should be sent. Hence, we need some indicator as part of the address to distinguish between the two St-Brunos. The person writing the address on a piece of mail would then be expected to make this indicator as part of the address; such indicators are generically called postal codes. The postal codes can help to uniquely identify villages, towns, city blocks, and even mail carriers' routes. Many countries have adopted postal codes; for example, Argentina, Australia, Brazil, Britain, Canada, France, India, Israël, Italy, Philippines, Poland, South Africa, Turkey and the United States of America. These postal codes, which are known by various names in different countries, are usually either numeric or a mixture of numeric and uppercase alohabetic characters. For instance, Canadian postal codes contain six alphanumeric characters with a blank character in the middle, such as "J3T 4A8"; the code thus requires space for seven characters; we say the expanse of the code is seven. Similarly, the expanse of a U.S. zip code such as "08836-9956" is ten. Figure I.1 shows specimens postal codes from various countries.

Argentina	4600	Italy	34123
Australia	4061	Israël	18970
Brazil	90010	Philippines	782-849
Britain	CB2 3QP	Poland	00-866
Canada	J3V 4G9	South Africa	2091
France	92130	Turkey	81040
India	500974	United States	08836-9956

Figure I.1. Specimen postal codes from various countries.

I.2 Fundamental Notions of Segmentation

Postal codes on a piece of mail provide a means for identifying uniquely a destination address, as described in the previous section. The adoption of postal codes provides one more advantage. Mail sorting can be speeded up considerably when these postal codes are recognized by optical character readers (OCRs). The optical reader usually scans the postal code on the piece of mail and stores digitized patterns of the characters as an image of dark and white pixels. The size of the image is dependent on the resolution of the optical reader. To illustrate, a digitized image of the Canadian postal code "J3T 4A8" is shown in Figure I.2.

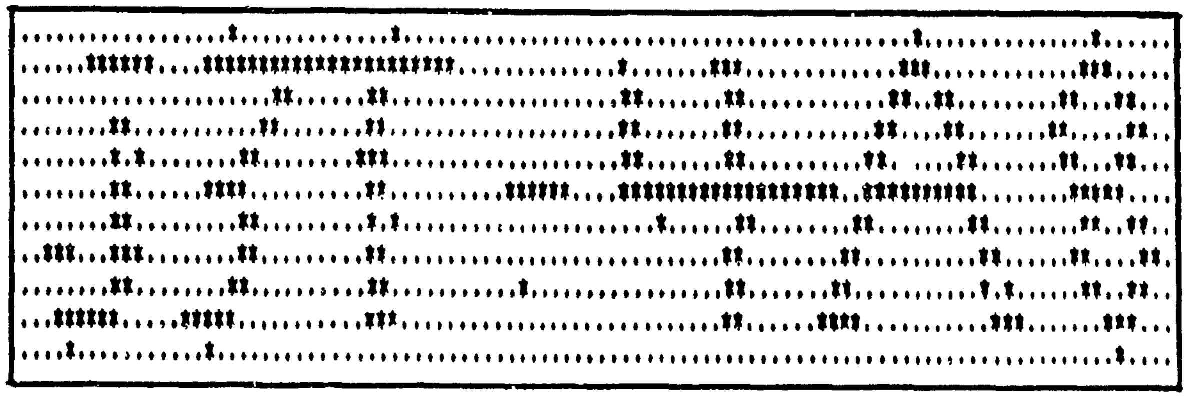


Figure I.2. Binary image of a typical Canadian postal code, "J3T 4A8". According to the format prescribed by Canada Post Corporation, there should be a blank character between the "T" and the "4" above. Nonetheless, it has been observed that in writing postal codes by hand, some people put a hyphen or dot instead of the blank character. The OCR needs to treat the hyphen or dot as noise. Salt (an isolated white pixel where there should be a dark pixel) and pepper (an isolated dark pixel where there should be a white pixel) noise may also be present in the image. As far as possible, we should attempt to eliminate all noise before the code is presented to the segmentation unit of the OCR.

In order to recognize the postal code, it is necessary to isolate each pattern from those adjacent to it in the image. In other words, we partition the image into segments, where each segment contains one pattern. For example, the image of Figure I.2 will have six segments. As mentioned in the legend below Figure I.2, the hyphen shown in the image is noise and must therefore be eliminated. If we segment the image correctly, then the successive segments would contain patterns of characters "J", "3", "T", "4", "A", and "8": no pattern would span more than one segment, and no segment would contain more than one pattern. To recognize any one pattern, we need to examine only the segment containing that pattern. Thus, if we were to recognize the six patterns individually (either by sequential or parallel processing), it would lead to the recognition of the full postal code.

The segmentation procedure becomes complicated when the image contains patterns of handwritten characters because of the following three commonly observed irregularities:

1. Broken patterns: a broken pattern is one which contains more than one fragment, where a fragment is defined to be a set of connected dark pixels. Ideally,

a pattern of a numeric or uppercase alphabetic character should be one fragment. A broken pattern, on the other hand, contains more than one fragment. To illustrate, the pattern for character "J" in Figure I.2, is made up of three fragments.

2. Touching patterns: patterns are said to be touching when a fragment contains parts of more than one pattern. In Figure I.2, for instance, "3" and "T" are touching patterns.

3. Overlapping patterns: patterns are said to be overlapping if fragments from different patterns share columns, but do not touch. To illustrate, the patterns for characters "4" and "A" in Figure I.2 are overlapping.

I.3 Historical Survey of Segmentation

In the early 1960's, some OCR systems were designed to detect a line of machine printed characters by means of a special mark at the beginning of the line. A simple segmentation algorithm was then used with special recognition logic (as in the IBM 1287 optical reader). In some applications, overly wide patterns were simply rejected by the recognition logic so that no segmentation of touching patterns was required (as in the IBM 3086 machine).

Later, more sophisticated machines were designed as those described by Hennis(1968). These machines calculated the average number of columns that a pattern occupied. If two consecutive blank columns were found, segmentation was performed along those columns. Otherwise, segmentation was forced along the columns where it was expected the pattern would end.

Hoffman & McCullough(1971) performed experiments using two different segmentation methods; the first method, called quasi-topological segmentation, based the decision to segment a character on a combination of feature-extraction and pattern-width measurements; the second method, topological segmentation, involved feature extraction alone. The experimental data consisted of machine-printed uppercase alphanumeric characters and, it was reported, that the topological method was superior to the quasi-topological approach.

A solution to character segmentation was proposed in Kori & Lin(1976) to deal with both broken and touching patterns. Since no amount of threshold adjustment could satisfactorily eliminate broken patterns, a decision was made to recognize the constituent parts of certain broken patterns as special classes. Later, during second level classification, these special classes were converted back to their original patterns. For instance, if an "M" were to be broken in the middle, a special class for both the left and right halves would exist. In the second level classification, these two classes, assuming they would be correctly recognized, would be replaced by the character "M". A similar approach was followed for touching patterns. The problem with this approach, was that, there were so many possible combinations of adjacent characters that it was impractical to recognize every pattern pair as a special class.

In Casey & Nagy(1982), the authors coupled the segmentation and classification operations in order to properly segment closely spaced printed characters. The proposed algorithm accepted a string of touching patterns only if it could decompose the string into patterns that could be positively identified (known as recursive segmentation). It was shown that the method was suitable for a document entry system capable of accepting a wide range of type styles and sizes. The segmentation method is described in the paragraph below.

A window of pre-established size was set around the pattern to be recognized. If the array, delimited by the window, contained a single pattern, the classifier could recognize it in one step. If the classifier rejected the pattern, however, then the viewing window was narrowed from the right-hand side and the classifier was tested on the truncated array. Gradual closing of the window, followed by classification, was repeated until either a truncated array was successfully recognized, or the window became so narrow that the search was abandoned.

The recursive segmentation method described above was unfortunately a very lengthy process, involving, as it did, the repeated classification of different subarrays of the given pattern. Moreover, the major condition for successful operation of the system, was that, in each document containing touching patterns of a given style, the same patterns had to also occur in isolation.

In Toyoda & Nichimura(1982), a method was proposed for extracting patterns from an unknown format document image. The proposed method included a procedure for performing adequate image segmentation, without a priori knowledge of character size and position, as is usually the case for conventional OCRs. Overall, this method was tested on five different Japanese newspapers (Japanese characters are written vertically one below another) and a 99% correct segmentation rate was reported. The major problems encountered with this approach occured when, (a) components of separated patterns were connected to the upper or lower pattern, (b) when separated patterns existed with no regularity, and (c) when two half-pitch patterns existed continuously in text line (pitch being defined as the average width of a pattern).

Pervez & Suen(1982) experimented with 1030 handwritten U.S. zip codes from the ZIPSCRIPT data base. The data base contained 5-digit zip codes which were in use in the U.S. until recently (the 5-digit codes have now been replaced by 9-digit codes). Pervez and Suen's segmentation method was based on a sequential scanning of the image. By performing two passes over the image, the different regions belonging to the patterns were recorded and then identified by means of a connectivity matrix.

In Jih(1980 and 1981), a method known as Two-Stage Segmentation and Registration Scheme (TSSRS) was developed and used in Casey & Jih(1983) to recognize machine-printed documents. By using vertical scans, touching patterns were isolated by finding out the position where the pixel density of vertical scans was a minimum. This is known as the minimum density search method. Its main drawback was that the segmentation often introduced distortions in the two subsequent segmented patterns. Moreover, this method was specifically aimed and designed for fixed-pitch typewritten documents.

A segmentation method was presented in Nishimura, Nogushi & Toyoda(1983) that could handle mixed-pitch lines of Japanese text. This proposed algorithm first examined change of pitches in the line, and, if detected, performed segmentation with the aid of "type setting rules". Traditional segmentation methods for reading text, such as the "right justification" method (where segmentation is performed by requiring scans containing no dark pixels between adjacent patterns), would not be very effective in this case because of the inherent existence of several strokes in a single Japanese character.

The basic idea of the above algorithm was to detect the change of pitch by examining the text line after a pre-defined number of scans had occured; if the pitch was found to be fixed, then the traditional pitch oriented segmentation approach was applied. However, if the pitch had been changed, the more careful segmentation method, which utilized knowledge about the type setting rules was applied. Prior to segmentation, the program detected the size of typeface used in the text. The regular length and width of the character representing each typeface were given a priori to the program. By transforming the original coded image into a set of rectangles, the program counted up the number of rectangles which were exactly the same size as the regular character of each typeface, enabling it to determine if a change of pitch had occured.

The major problem with this approach was its sensitivity to touching patterns; the program detected a change of pitch even if it had remained fixed. Even though a good rate of segmentation was obtained in this experiment (87.4%), the major source of segmenting errors occurred because of the existence of touching patterns.

In Kahan, Pavlidis & Baird(1987), touching patterns were dealt with in the following manner: the authors mentioned that the most frequent types of joins, that is, occurrences of touching patterns in typewritten characters, could be characterized; it therefore followed that the segmentation occur where the characterization was best matched. Kahan et al. said that "Most joins fall into one of two categories: serif joins and double-o joins. Serif joins are usually near the bottom or the top of the characters and are due to overlap of the extreme points of a pair of serifs. Double-o joins usually occur near the middle of the characters and are due to osculating convex contours." Therefore, segmentation of touching patterns was performed by determining the "vertical position V", defined as the function mapping a horizontal position to the number of dark pixels in the vertical column at that position. This is a derivative of the minimum density procedure discussed above.

I.4 Scope of Thesis

In this thesis, we propose an algorithm for segmenting handwritten postal codes. Our objective is to develop a segmentation algorithm which is able to resolve all of the three irregularities discussed in Section I.2; that is, broken, touching, and overlapping patterns. The algorithm should be general enough to be adopted for segmenting postal codes of many other countries. Furthermore, to verify the robustness of our segmentation algorithm, we shall recognize the postal codes.

In the following chapter, we describe our segmentation algorithm; the experimental results are discussed in Chapter III. In Chapter IV, we review recognition concepts, describe the recognition approach we adopted, and discuss its experimental performance. Finally, we give our concluding remarks in Chapter V.

CHAPTER II

PROPOSED SEGMENTATION ALGORITHM

In this chapter, we first give an informal description of our segmentation algorithm. To add clarity to our description, we present pictorial representations of our segmentation algorithm being applied to the postal code of Figure I.2. We refer to these pictorial representations at different stages in the description of our algorithm. Furthermore, to remove any ambiguity from our informal description, we delineate the algorithm in a Pascal-like language at the end of the chapter.

Before we apply our segmentation algorithm, we must first ensure that the image has been preprocessed; that is, all salt-and-pepper noise has been eliminated. Such preprocessing has been described in Unger(1959) and Doyle(1960) and will be concisely delineated in Section III.2. Figure II.1 shows the image of the postal code shown in Figure I.2 after salt-and-pepper noise has been eliminated.

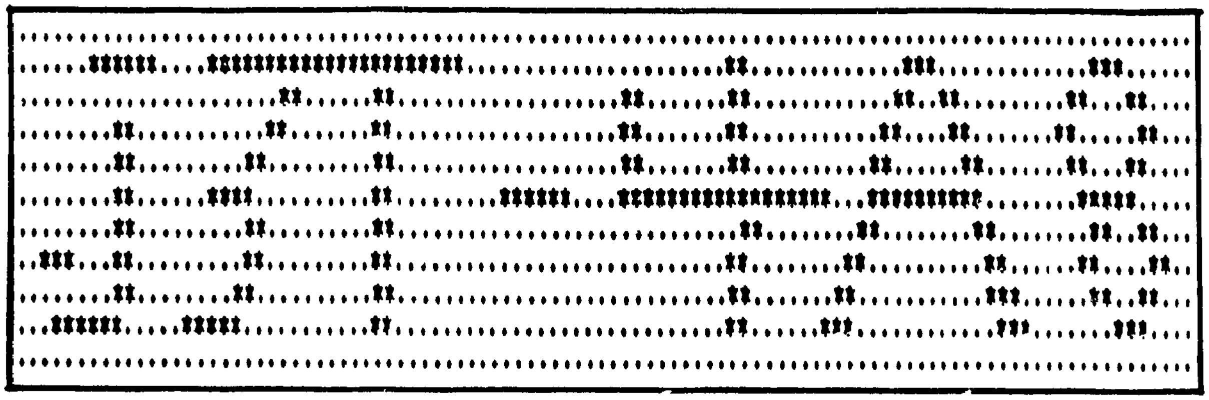


Figure II.1. The binary image of the postal code "J3T 4A8" shown in Figure I.2 after salt-and-pepper noise has been eliminated. Note that the hyphen still remains. This will be eliminated by our segmentation algorithm.

Overall, our segmentation algorithm consists of making two passes over the image. In the first pass, we create a list of the fragments present in the image. This helps us separate overlapping patterns. In the second pass, (a) all fragments that are judged to be noise are eliminated, (b) touching patterns are separated, and (c) the different fragments belonging to a broken pattern are associated together into the same segment. We now give the details of the two passes.

II.1 First Pass

In essence, in the first pass, we find a dark pixel on the edge of a fragment, and then trace its outer contour (that is, we travel along the pixels on the edge of the fragment). For tracing the outer contour, we adopt the procedure described in Toussaint & Donaldson(1970): upon meeting a dark pixel on the contour of the fragment, we travel left; otherwise, we travel right. We repeat this operation until we have travelled back to the starting contour pixel. As we trace the outer contour of a fragment, we record the following information: (a) the bounds of the fragment (that is, the number of the leftmost and rightmost columns, and the number of the topmost and bottommost rows which enclose the fragment) , (b) the area of the fragment (the number of dark pixels in it), and (c) the row and column numbers of each pixel on the contour of the fragment. We repeat this procedure for every fragment in the image. We describe next how we find a dark pixel on the edge of a fragment to begin tracing its contour.

To locate the starting contour pixel of the first fragment, we scan the image columnwise, bottom to top, the columns themselves being scanned left to right, until we meet a dark pixel. That pixel becomes the starting pixel for tracing the contour of the first fragment. Figure II.2 illustrates how the starting contour pixel of the first fragment is obtained. The figure also depicts the tracing of the outer contour of the fragment.

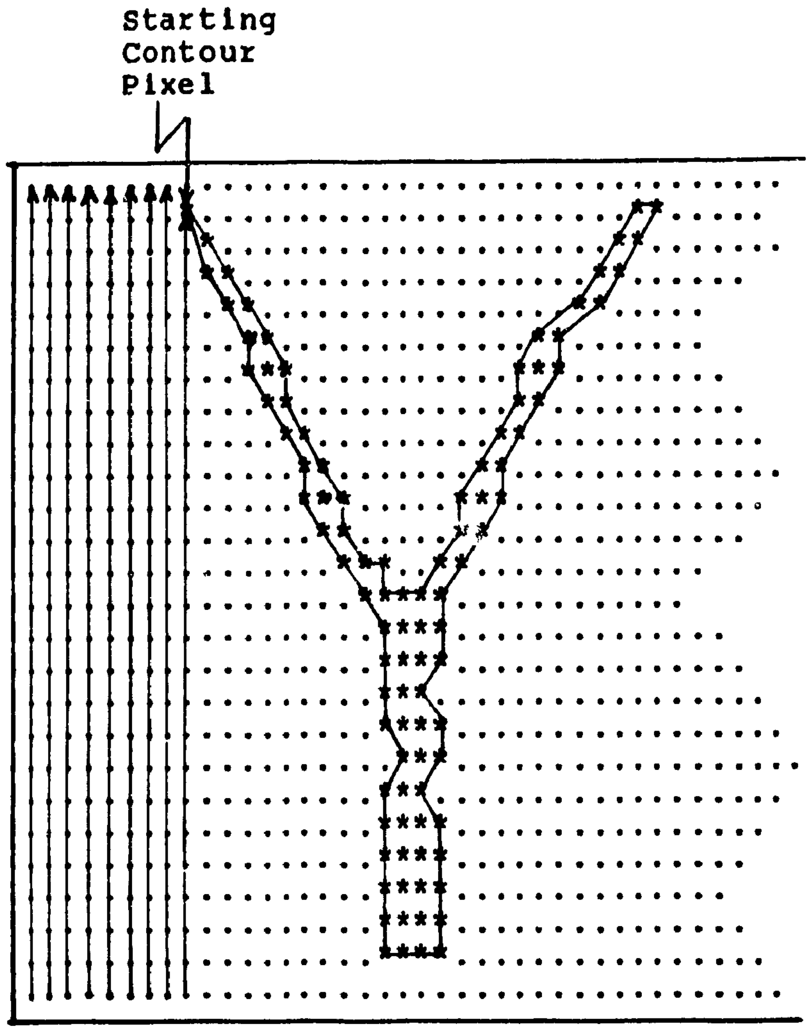


Figure II.2. The above figure illustrates how we locate the starting contour pixel of the first fragment of the postal code image. We scan the image columnwise, bottom to top, the columns themselves being scanned left to right, until we meet a dark pixel. The figure also depicts the tracing of the outer contour of the fragment from the starting contour pixel.

We now discuss how we find the starting contour pixel of the (i+1)st fragment immediately after we have traced the contour of the ith fragment, for $i \geq 1$. For every row of the ith fragment, we start from its rightmost pixel and scan the row, left to right, until we meet a dark pixel. After doing this for all the rows, the leftmost dark pixel we had met becomes the starting pixel for the contour tracing of the (i+1)st fragment. See Figure II.3. If for all the rows, we reach the rightmost bounds of the image without meeting a dark pixel, we should have extracted all the fragments.

Nevertheless, there can be rare occasions when we may have failed to extract a fragment or two. As Figure II.4 illustrates, the fragments that we may not have extracted by the above procedure, are vertical strokes encompassed on two or more sides by a larger fragment, one of these sides being the right. To extract such vertical strokes, we proceed in the following manner. Suppose the fragments extracted until now have been numbered 1 to f. For every row of the jth fragment (j ranging from f down to 1), we start from its leftmost pixel and scan the row, right to left, until we meet a dark pixel. We do this for all the rows. Let us refer to the rightmost pixel we met as p. We then verify to see whether earlier, in the left to right scan, we had already extracted a fragment containing p. If we had done so, we now skip over the fragment that contains p.

Otherwise, we trace the contour of the fragment using p as a starting contour pixel. This fragment is then associated into the same segment as that of its neighbor on the right. The need for this association is apparent from Figure II.4.

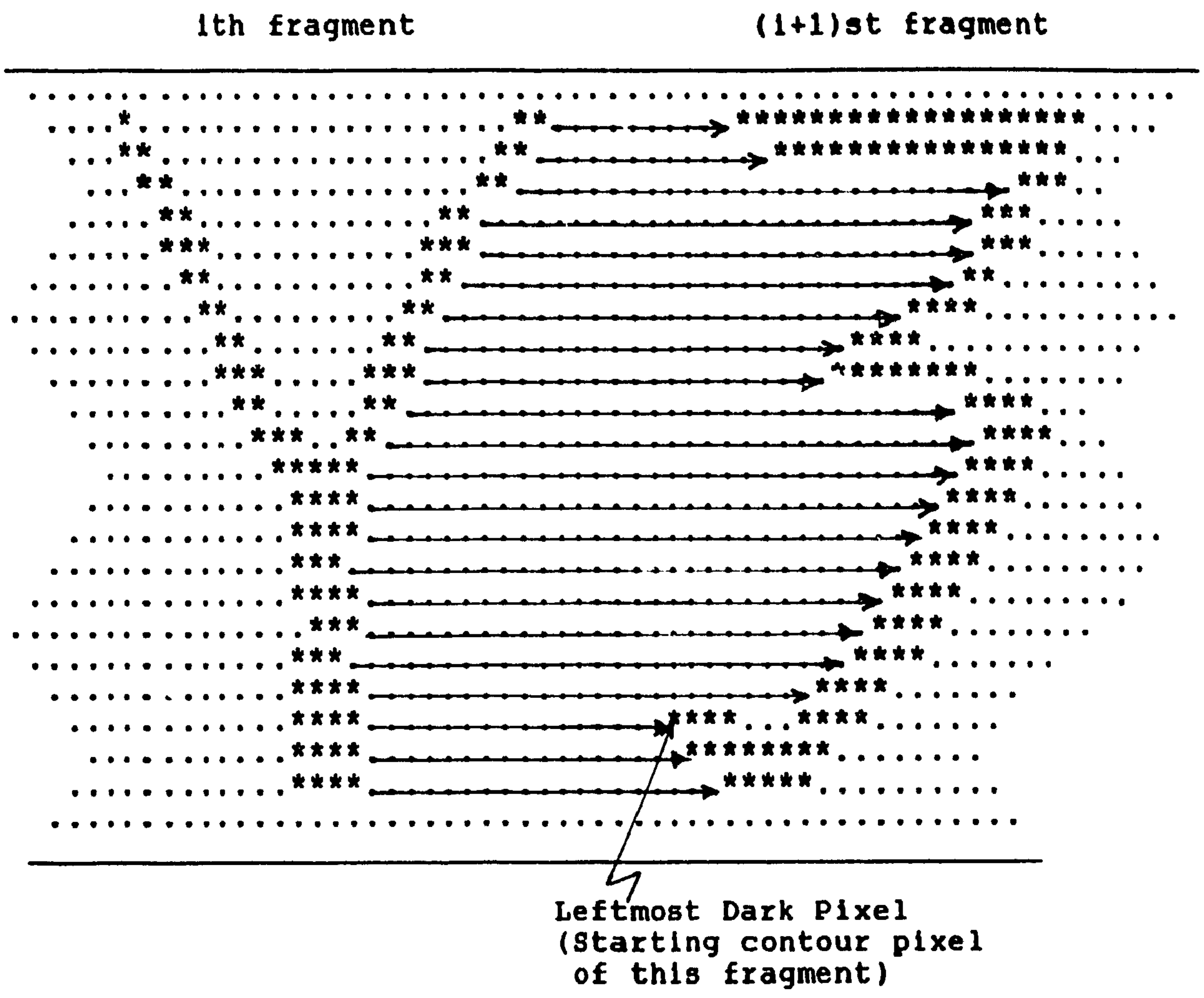

ith fragment
(i+1)st fragment
Leftmost Dark Pixel (Starting contour pixel of this fragment)

Figure II.3. The starting contour pixel of the (i+1)st fragment, i $\geq$ 1, after locating the ith fragment, is obtained in the following manner: for every row of the ith fragment, we start from its rightmost pixel and scan the row, left to right, until we meet a dark pixel. After doing this for all the rows, the leftmost dark pixel we had met becomes the starting pixel for the contour tracing of the (i+1)st fragment as the above figure illustrates.

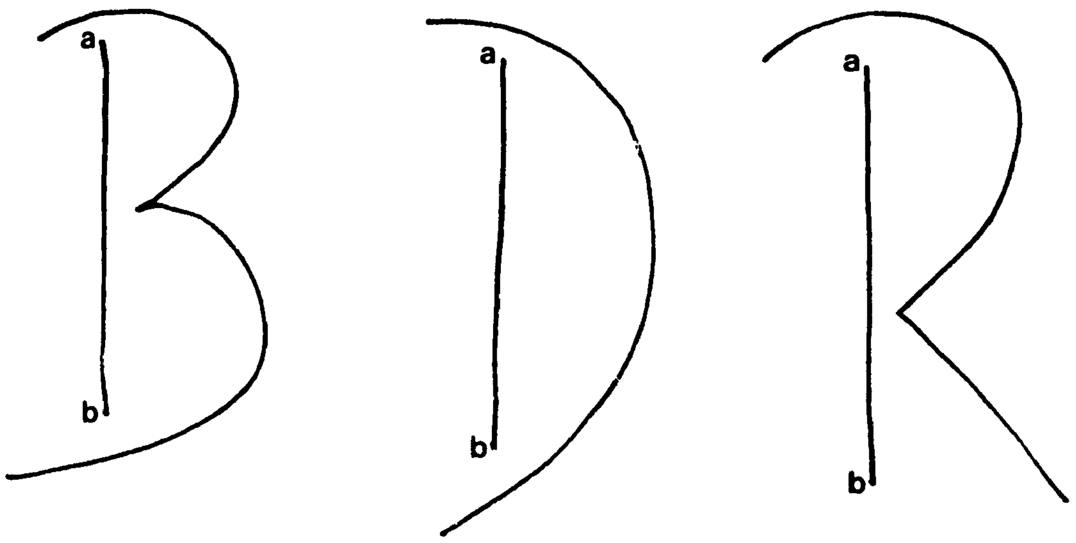


Figure II.4. The vertical strokes $\overline{ab}$ are the fragments which would not be extracted in the left to right scan of the first pass in our segmentation algorithm. Note, the fragments that would not be extracted are encompassed on two or more sides by a larger fragment, one of these sides being on the right. Therefore, in the first pass, to extract these fragments we also have to scan the image right to left. After a fragment representing a vertical stroke has been extracted, it will need to be associated into the same segment as that of the larger fragment on its right.

As can be seen from Figure II.5, by the end of this pass, we have extracted all the fragments of the image. Moreover, as the figure shows, we have also separated overlapping patterns.

If the number of fragments extracted exceeds a threshold, one may consider the image to be too fragmented, and therefore one may reject it as unsuitable for segmentation (we call this the fragmentation threshold). We now describe the second pass.

II.2 Second Pass

Note that as mentioned in Section I.2, the size of the image depends on the resolution of the optical reader. But it is quite likely that any specific postal code may fit in less space than that provided by the image. That is to say, the image may contain blank rows and blank columns surrounding the postal code. Let us define a characteristic rectangle (CR) to be the smallest rectangle that encloses the postal code. In other words, the CR contains no blank rows or blank columns surrounding the postal code. By the end of the first pass, we should have established the CR. For any further processing we need to scan within the bounds of the CR, not the full image. The second pass consists of the following three steps:

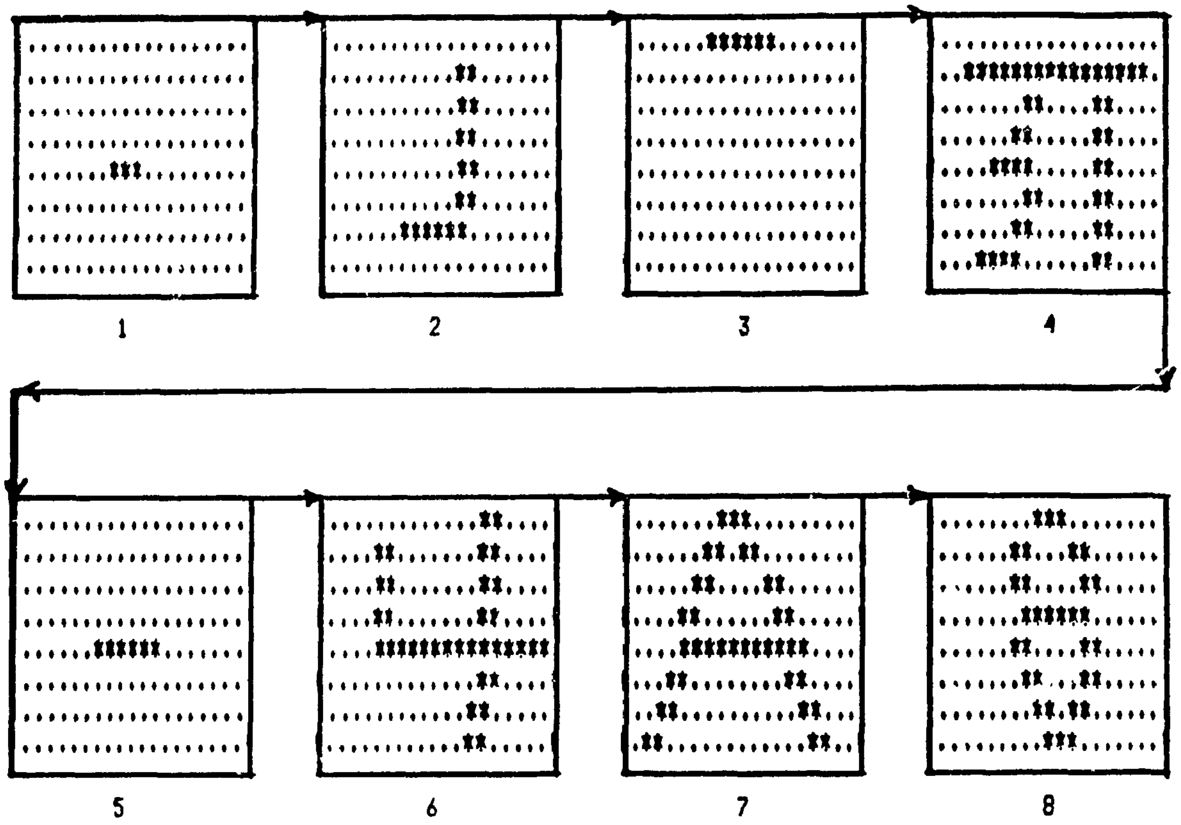


Figure II.5. The list of eight fragments extracted from the postal code image of Figure II.1 at the end of the first pass. The first three fragments are from the broken pattern "J"; the fourth fragment consists of two touching patterns "3" and "T"; the fifth fragment is obtained from the hyphen in the middle of the postal code; the sixth and seventh fragments are obtained by separating the two overlapping patterns "4" and "A"; the last fragment is obtained from the pattern "8".

<u>Step 1:</u> In the first step, we eliminate all fragments whose area is less than a threshold. This is because we consider such fragments to be noise that remained, even though we had eliminated salt-and-pepper noise before the first pass of our segmentation algorithm. (Empirically, we have found this threshold to be 1/3000 the area of the CR). This step removes most of the noise present in the image. (Nevertheless, here we expect that there can be noise dependent on the general writing habits of people in different countries. For example, as mentioned in the legend below Figure I.2, some people in Canada put a hyphen or dot in the place where there should have been a blank character. It is here that one may need to extend this step to remove noise caused by the writing habits of the people. For instance, to eliminate the hyphen or dot in a Canadian postal code, we propose the following: if along the middle columns of the CR, we find a fragment such that its height is less than half the height of the CR, we eliminate this fragment from the list.) See Figure II.6.

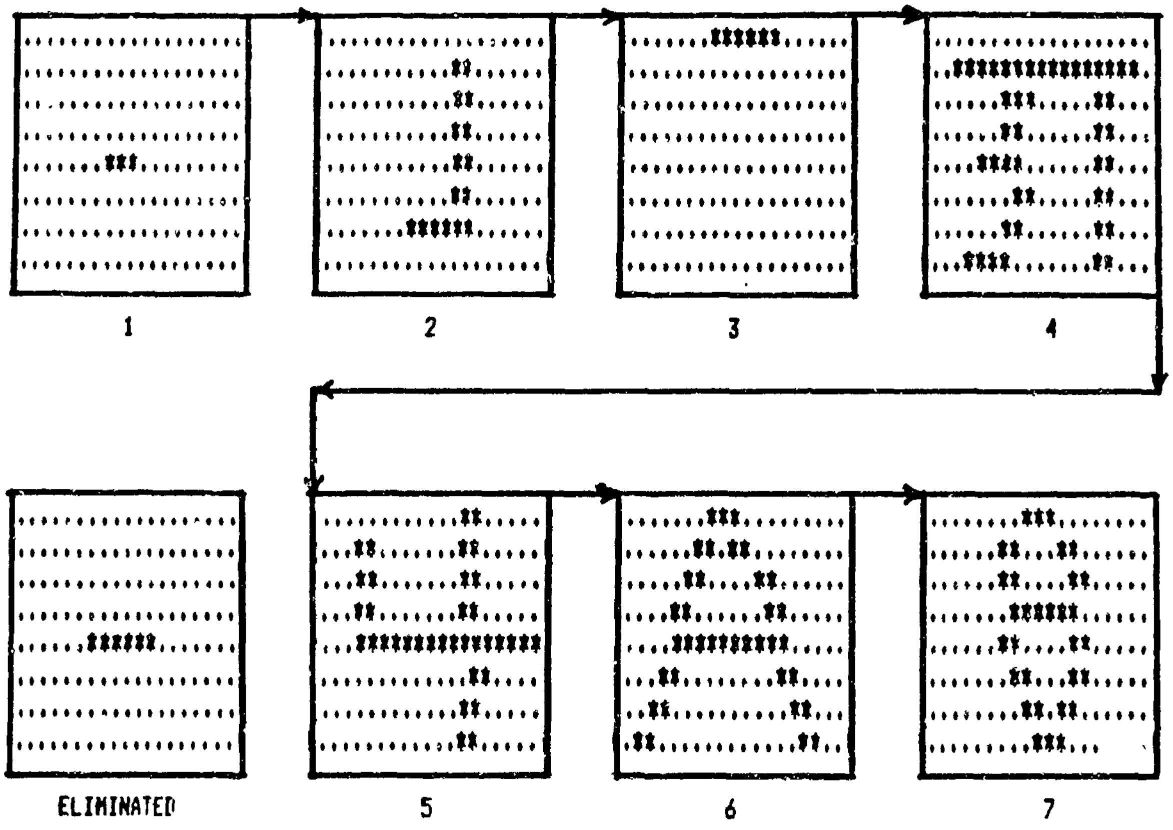


Figure II.6. At the end of the first step of the second pass, the eight fragments of Figure II.5 have now been reduced to seven fragments. This is because the fragment corresponding to the hyphen has been eliminated from the list.

<u>Step 2:</u> The objective of this step consists in separating touching patterns. To do this, we calculate the pitch of a pattern. The number of columns in the CR divided by the expanse (defined in the beginning of Section I.1) of the postal code gives the pitch of a pattern. If a fragment's width is found to be greater than 1.5 times the pitch, it is assumed that the fragment is the outcome of touching patterns. So the fragment is separated in the middle to create two fragments. See Figure II.7.

<u>Step 3:</u> In this step, the different fragments of a broken pattern are associated together into the same segment. A fragment is considered to have been broken off from a pattern if its height is less than half the height of the CR. Once we have located such a fragment, we need to associate it with the fragment that is its closest neighbor, either at the left or right or top or bottom. For instance, to locate the left neighbor F' of a fragment F, we scan the columns right to left from the left edc^ of F until we encounter a fragment. The fragment just encountered is F'. Similarly, we can locate the bottom, top and right neighbors. From these four neighbors, we associate the fragment with that neighbor which is the closest overall. (Note, a fragment does not necessarily have four

neighbors; for example, a fragment at the top of the CR may have no top neighbor as is the case for the top horizontal stroke of the pattern for character "J" in Figure I.2. In such a case, we encounter the boundary of the CR.) To clarify our explanation, we refer to Figure II.7. The first and third fragments are considered to have been broken off from pattern "J". Hence, we locate the closest neighbor of the first and third fragments respectively which is the second fragment. We associate the first, second, and third fragment into one segment (to contain pattern "J") by applying the following three steps: (i) the segment's area, that is the number of dark pixels in the segment, becomes the sum of the areas of the first, second, and third fragments; (ii) the bounds of the segment become the leftmost column number of the first fragment, the rightmost column number of the third fragment, the topmost row number of the third fragment, and the bottommost row number of the second fragment; (iii) the contour points of the segment, that is the row and column numbers of each pixel on the contour of the segment, are obtained by combining all pixels on the contour of each of the three respective fragments, that is, the first, second, and third fragments. Figure II.8 shows the postal code partitioned into six segments after this step.

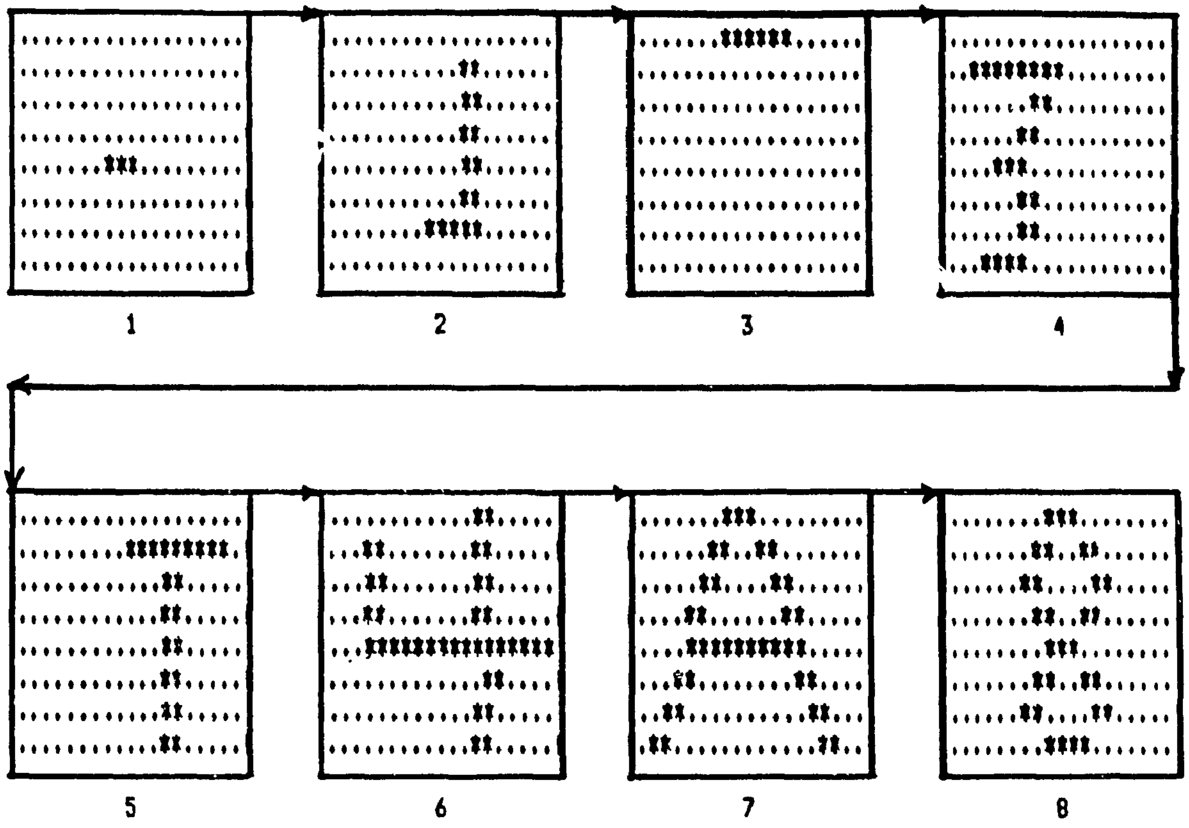


Figure II.7. At the end of the second step of the second pass, the seven fragments of Figure II.6 have now become eight fragments. This is because the touching patterns "3" and "T" have been separated.

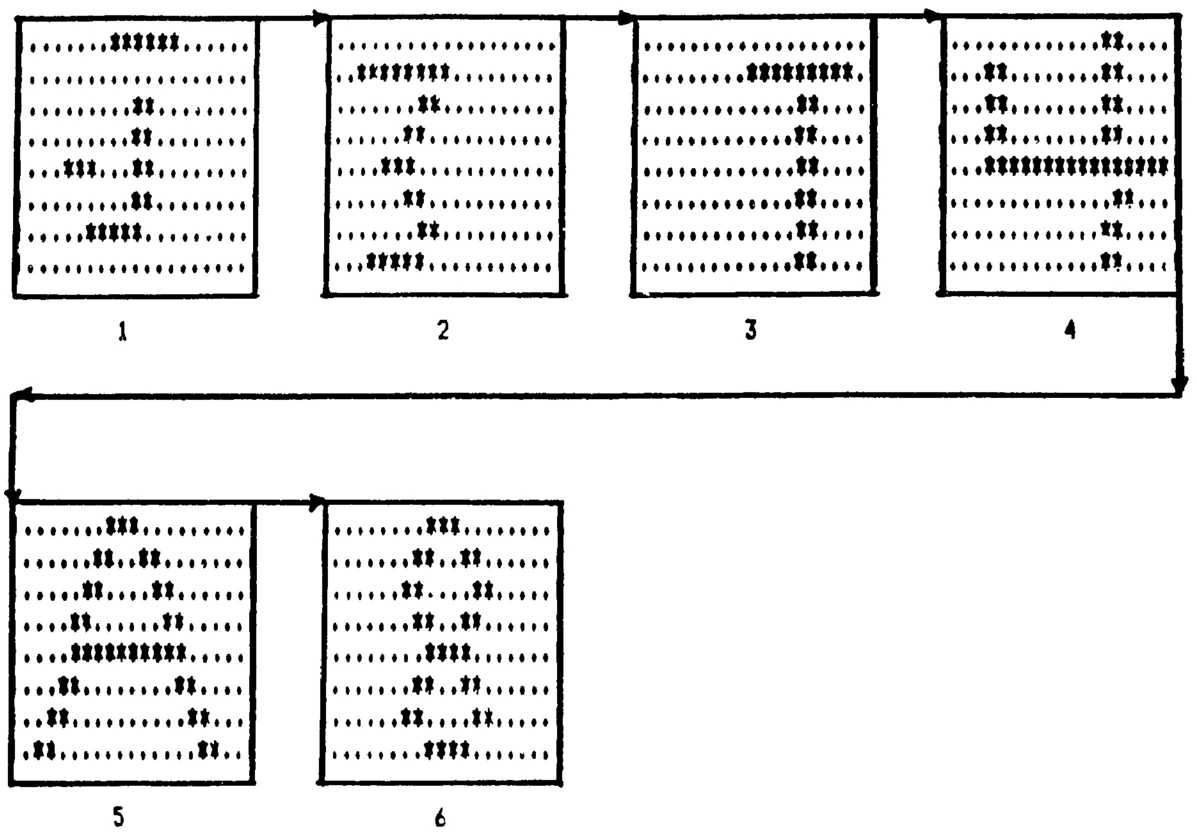


Figure II.8. At the end of the third step of the second pass (that is, at the end of our segmentation algorithm), the postal code has been partitioned into six segments. The first three fragments of Figure II.7 were associated together into the first segment above. The postal code has been correctly segmented; there are six segments and each segment contains one pattern.

At the end of this pass, that is, at the end of our segmentation algorithm, if the number of segments remaining is not equal to the number of alphanumeric characters in the postal code (for instance, six alphanumerics for Canada and nine numerics for U.S.), we can be certain that the image has not been correctly segmented. We then consider the image to be rejected. On the other hand, if the number of segments remaining is equal to the number of alphanumeric characters in the postal code, we then determine whether the postal code has been incorrectly or correctly segmented as follows: if the fragments belonging to any one pattern occur in more than any one segment, then the postal code has been segmented incorrectly, otherwise correctly.

In the next chapter we present the experimental performance of our segmentation algorithm. However, to remove any ambiguity from our explanation, we give, below, a Pascal-like formal description of our segmentation algorithm.

```
/..............Declaration of global variables............../
 :
 :
Const
     FRAGMENTATION_THRESHOLD = 13 ;
     NOISE_MAXIMUM_AREA = 1/3000 ;
     EXPANSE = 7 ;              / as described in Section I.1 /

Type M = 0..MAXLENGTH ;          / dimensions of binary image /
     N = 0..MAXWIDTH ;
     IMAGE_TYPE = array [M,N] of WHITE..DARK ;

        / type of the image, where M and N are its dimensions /

     ROW_LIMITS = array[M] of N ;
     COLUMN_LIMITS = array[N] of M ;
     FRAGMENT = record
                 :
                 :
                 L_LIM, R_LIM : COLUMN_LIMITS ;
                 T_LIM, B_LIM : ROW_LIMITS ;
                 FIRST_ROW, LAST_ROW : M ;
                 FIRST_COL, LAST_COL : N ;
                 AREA : real ;
                 :
                 :
               end ;
     DIRECTYPE = (UP, DOWN, LEFT, RIGHT) ;

Var MINIMUM_DIST : array[DIRECTYPE] of real ;
    D : DIRECTYPE ;
    F : array[0..FRAGMENTATION_THRESHOLD] of FRAGMENT ;
    IMAGE : IMAGE_TYPE ;
    NUM_OF_SEGMENTS : integer ;
```

```
/..................... FIRST PASS ............................/

/ ........... Extraction of first fragment ...................../

  F_NO := 1 ;                              / first fragment of list /
  NOT_FOUND := true ;
  COL := 0 ;
  While (COL < MAXWIDTH) and NOT_FOUND do
    Begin
      COL := COL + 1 ;                           / Left to right scan /
      ROW := MAXLENGTH ;
      While (ROW > 0) and NOT_FOUND do
        Begin
          ROW := ROW - 1 ;                       / Top to bottom scan /
          If IMAGE[ROW,COL] = DARK then NOT_FOUND := false ;
        end ;
    end ;

  TRACE_CONTOUR(ROW,COL) ;    / Trace contour of first fragment /


/ Procedure TRACE_CONTOUR traces the outer contour of the frag-
  ment encountered and records, (a) the bounds of the charac-
  teristic rectangle in variables FIRST_ROW, LAST_ROW,
  FIRST_COL and LAST_COL, (b) the row and column numbers of
  each pixel on the outer contour of the fragment in the
  vectors L_LIM, R_LIM, T_LIM and B_LIM respectively, and
  (c) the number of dark pixels of the fragment in AREA.       /


  END_OF_IMAGE := false ;
  While not END_OF_IMAGE do
    If F_NO > FRAGMENTATION_THRESHOLD then REJECT_POSTAL_CODE
    else EXTRACT_FRAGMENT ;

  NUM_OF_FRAGMENTS := F_NO / number of fragments extracted after
                             left to right scan.               /
```

```
/......... Right to left scan to ensure all fragments .........
 ................... have been extracted ......................../


For J := NUM_OF_FRAGMENTS downto 1 do
  Begin

     / Initialize VECTOR to current fragment's leftmost bounds /

   For ROW := 1 to MAXLENGTH do VECTOR[ROW] := F[J].L_LIM[ROW] ;
   ROW := MAXLENGTH ;
   CJ := 0 ;
   While ROW > 1 do
     Begin
       ROW := ROW - 1 ;
       NOT_FOUND := true ;
       While (VECTOR[ROW] > 0) and NOT_FOUND do
         Begin
          VECTOR[ROW] := VECTOR[ROW] - 1 ;
          If (IMAGE[ROW,VECTOR[ROW] = DARK) then / dark pixel? /
            If VECTOR[ROW] > CJ then
               Begin
                 CI  := ROW ;                   / row coordinate /
                 CJ  := VECTOR[ROW] ;/ new max column coordinate/
                 NOT_FOUND := false ;
               end ;
          end ;
      end ;

    If NOT_EXTRACTED(CI,CJ) then
      Begin
       F_NO := F_NO + 1 ;    / Trace contour of fragment if not
       TRACE_CONTOUR(CI,CJ) ;       extracted already.        /
       ASSOCIATE_FRAGMENTS(J,F_NO) ; / Associate this fragment
      end ;                            with its right neighbor /
  end ;

NUM_OF_FRAGMENTS := F_NO ;     / update # of fragments in list /
```

```
/...................... SECOND PASS ........................../

/  First step: elimination of fragments judged to be noise.   /

  MIN_AREA := HEIGHT_OF_CR * WIDTH_OF_CR * NOISE_MAXIMUM_AREA ;
  For J := 1 to NUM_OF_FRAGMENTS do
    Begin
      F_HEIGHT := F[J].LAST_ROW - F[J].FIRST_ROW ;
      If F_HEIGHT < (HEIGHT_OF_CR/2) then

/**********************************************************
*                                                        *
* ------> Special case for Canadian postal codes <------ *
*                                                        *
*     Removal of hyphen or dot often present in Canadian *
*   postal codes (discussed in Section II.2).            *
*                                                        *
*       If (F[J] lies in middle of CR) or                *
*                                                        *
**********************************************************/

          If (F[J].AREA < MIN_AREA) then ELIMINATE_FRAGMENT ;
    end ;

/ Second step: separation of touching patterns                /

   NUM_OF_FRAGMENTS := F_NO ;/ update # of fragments in list /
   PITCH := WIDTH_OF_CR/EXPANSE ;           / calculate pitch /
   for J := 1 to NUM_OF_FRAGMENTS do
     Begin
       F_WIDTH := F[J].LAST_COL - F[J].FIRST_COL ;
       If (F_WIDTH > 1.5*PITCH) then
         Begin
           F_NO := F_NO + 1 ;
           SEPARATE_FRAGMENT(J,F_NO) ;

/  Separate fragment J into fragments J and F_NO, and insert
   the latter in the list.                                     /

         end ;
     end ;

NUM_OF_FRAGMENTS := F_NO ;   / update # of fragments in list /
```

```
/ Third step: association of different fragments  belonging
              to a broken pattern, if any.                    /

/*************************************************************
*                                                           *
* ------> Special case for Canadian postal codes <------    *
*                                                           *
*     Canada Post Corporation  does not allow "I" as        *
*   leftmost character (to  be discussed  in Section        *
*   III.1). Therefore,                                      *
*                                                           *
* If (F[1].LAST_COL-F[1].FIRST_COL < PITCH/2) then          *
*                          ASSOCIATE_FRAGMENTS(1,RIGHT) ;   *
*                                                           *
*  We associate the first fragment with its right neighbor. *
*                                                           *
*************************************************************/

     For J := 1 to NUM_OF_FRAGMENTS do
      Begin

       F_HEIGHT := F[J].LAST_ROW - F[J].FIRST_ROW ;

       If (F_HEIGHT < HEIGHT_OF_CR/2) then
        Begin
          FIND_CLOSEST_NEIGHBOR_AND_ASSOCIATE ;
          F_NO := F_NO - 1 ;
        end ;

      end ;

NUM_OF_FRAGMENTS := F_NO ;    / update # of fragments in list /


If NUM_OF_SEGMENTS ≠ NUM_OF_CHARS then REJECT_POSTAL_CODE ;


/ Reject postal code if the number of segments is not equal
  to number of characters in postal code.                     /
```

```
/............. Procedure EXTRACT_FRAGMENT ..................../

  Procedure EXTRACT_FRAGMENT ;
    Begin
      For ROW := 1 to MAXLENGTH do
          VECTOR[ROW] := F[FNO-1].R_LIM[ROW] ;

/ inititialize VECTOR to previous fragment's rightmost bounds /

      ROW := MAXLENGTH ;         / initialize to bottommost row /
      CJ := MAXWIDTH ;         / initialize to rightmost column /

      While (ROW > 0) do
        Begin
          ROW := ROW - 1 ;
          NOT_FOUND := true ;
          While (VECTOR[ROW] < MAXWIDTH) and NOT_FOUND do
            Begin
              VECTOR[ROW] := VECTOR[ROW] + 1 ;
              If (IMAGE[ROW,VECTOR[ROW]) = DARK then
                     / we have found a dark pixel on that row /
                If VECTOR[ROW] < CJ then
                  Begin      / Is it the closest dark pixel ? /
                   CI := ROW ;                  / row coordinate /
                   CJ := VECTOR[ROW] ; /new column coordinate /
                   NOT_FOUND := false ;
                  end ;
            end ;
        end ;

       If CJ = MAXWIDTH then END_OF_IMAGE := true

       else            / Trace contour of newly found fragment /
         Begin
           F_NO := F_NO + 1 ;
           TRACE_CONTOUR(CI,CJ) ;
         end ;

    end ;                                  / EXTRACT_FRAGMENT /
```

```
Procedure FIND_CLOSEST_NEIGHBOR_AND_ASSOCIATE ;

 Begin
  For D := UP to RIGHT do MIN_DIST[D] = infinity ;
  F_NEAR := SEARCH(LEFT) ;

 / Function SEARCH returns  the  fragment no. in the specified
   direction if one exists, else returns 0.                  /


  If F_NEAR ≠  0 then                    / a left fragment exists /

 / ..... Find closest distance to left neighbor fragment .... /

   For X1 = F[J].FIRST_ROW to F[J].LAST_ROW do

    For X2 = F[F_NEAR].FIRST_ROW to F[F_NEAR].LAST_ROW do

     PREV_DIST:=DST(X1,X2,F[J].L_LIM[X1],F[F_NEAR].R_LIM[X2])
     If (PREV_DIST < MIN_DIST[LEFT]) then
                                   MIN_DIST[LEFT] = PREV_DIST ;

/ Function  DST  evaluates the  absolute  distance between two
  pixels in a Cartesian plane.                               /


  F_NEAR := SEARCH(RIGHT) ;
  If F_NEAR ≠  0 then                   / a right fragment exists /

/ .... Find closest distance to right neighbor fragment ...../

  For X1 = F[J].FIRST_ROW to F[J].LAST_ROW do

   For X2 = F[F_NEAR].FIRST_ROW to F[F_NEAR].LAST_ROW do

    PREV_DIST:=DST(X1,X2,F[J].R_LIM[X1],F[F_NEAR].R_LIM[X2])
    If (PREV_DIST < MIN_DIST[RIGHT]) then
                                   MIN_DIST[RIGHT] = PREV_DIST ;
```

```
Procedure FIND_CLOSEST_NEIGHBOR_AND_ASSOCIATE ; (CONT'D)

  F_NEAR := SEARCH(DOWN) ;
  If F_NEAR ≠  0 then              / a bottom fragment exists /

/ .... Find closest distance to bottom neighbor fragment .... /

  For X1 = F[J].FIRST_COL to F[J].LAST_COL do

   For X2 = F[F_NEAR].FIRST_COL to F[F_NEAR].LAST_COL do

    PREV_DIST:=DST(F[F_NEAR].T_LIM[Y1],F[J].B_LIM[Y1],Y1,Y2)
    If (PREV_DIST < MIN_DIST(DOWN)) then
                                 MIN_DIST(DOWN) = PREV_DIST ;

  F_NEAR := SEARCH(UP) ;
  If F_NEAR ≠  0 then / a top fragment exists /

/ ..... Find closest distance to top neighbor fragment ..... /

  For X1 = F[J].FIRST_COL to F[J].LAST_COL do

   For X2 = F[F_NEAR].FIRST_COL to F[F_NEAR].LAST_COL do

    PREV_DIST:=DST(F[F_NEAR].F_LIM[Y1],F[J].T_LIM[Y1],Y1,Y2)
    If (PREV_DIST < MIN_DIST[UP]) then
                                    MIN_DIST[UP] = PREV_DIST ;


 ASSOCIATE_FRAGMENTS(J,CLOSEST_NEIGHBOR(MIN_DIST))

/ Associate fragment J with closest overall neighbor fragment /


 end ; / FIND_CLOSEST_NEIGHBOR_AND_ASSOCIATE /
```

CHAPTER III

EXPERIMENTAL RESULTS OF SEGMENTATION

In this chapter, we (i) describe the data that we used in our experiment, (ii) delineate the preprocessing algorithm (both formally and informally), and (iii) present and discuss our segmentation results on Canadian postal codes.

III.1 Description of Experimental Data

To observe the performance of our segmentation algorithm, we tested it on a data set of Canadian postal codes. These postal codes are of interest to the author, since he is employed by Canada Post Corporation (CPC). Earlier, in Section I.2, we had referred to Figure I.2 to present a specimen Canadian postal code. The CPC has prescribed the following format for these codes:

$$\alpha_1 \ \nu_1 \ \alpha_2 \quad \nu_2 \ \alpha_3 \ \nu_3$$

The α's are uppercase alphabetic but cannot be "D", "F", "I", "O", "Q" or "U". Furthermore, at present, α_1 cannot even be a

"W" or a "Z". The ν's are numeric digits from "0" to "9". As mentioned in the legend below Figure I.2, there should be a blank character between α_2 and ν_2.

The segmentation algorithm we have presented in Chapter II is general enough to be used for segmenting handwritten postal codes of different countries. The algorithm may however be refined to make use of certain characteristics present in any code. For example, as explained above, the leftmost character of a Canadian postal code cannot be "I". To make use of this information, we augmented step 3 of the second pass of our segmentation algorithm with this refinement: if we found the width of the leftmost fragment to be less than half the pitch (since "I" is not present, this can only happen for broken patterns), we associated it with its right neighbor. We emphasize that this refinement may not be suitable for postal codes of other countries: it will certainly not be suitable for the U.S. zip codes.

The data set for our experiment consisted of 300 postal codes. The postal codes were written by 15 people. Since the writing styles of these people differed from one another, the postal codes varied in size ranging from 4 mm to 10 mm in height, and 20 mm to 55 mm in width. We also wanted to test whether our method was sensitive to the thickness of the pen strokes used in writing the postal codes. So, of the 300 postal codes, 112 were written with a light-stroked pen (average stroke width of 0.4 mm), and the remaining 188 were written using a heavier-stroked pen (average stroke width of 0.9 mm). Some specimen postal codes from our data set are shown in Figure III.1. The postal codes were digitized by a MICROTEK-200 auto-reader (resolution 200 pixels/inch, 7.87 pixels/mm), the image of the postal code being contained within a matrix of 75 rows and 500 columns. The images were preprocessed to eliminate salt-and-pepper noise by the techniques described in Unger(1959) and Doyle(1960). Our segmentation algorithm was implemented in PASCAL Version 3.4, the programming being intuitively as efficient as possible. In the following section, we describe the preprocessing algorithm.

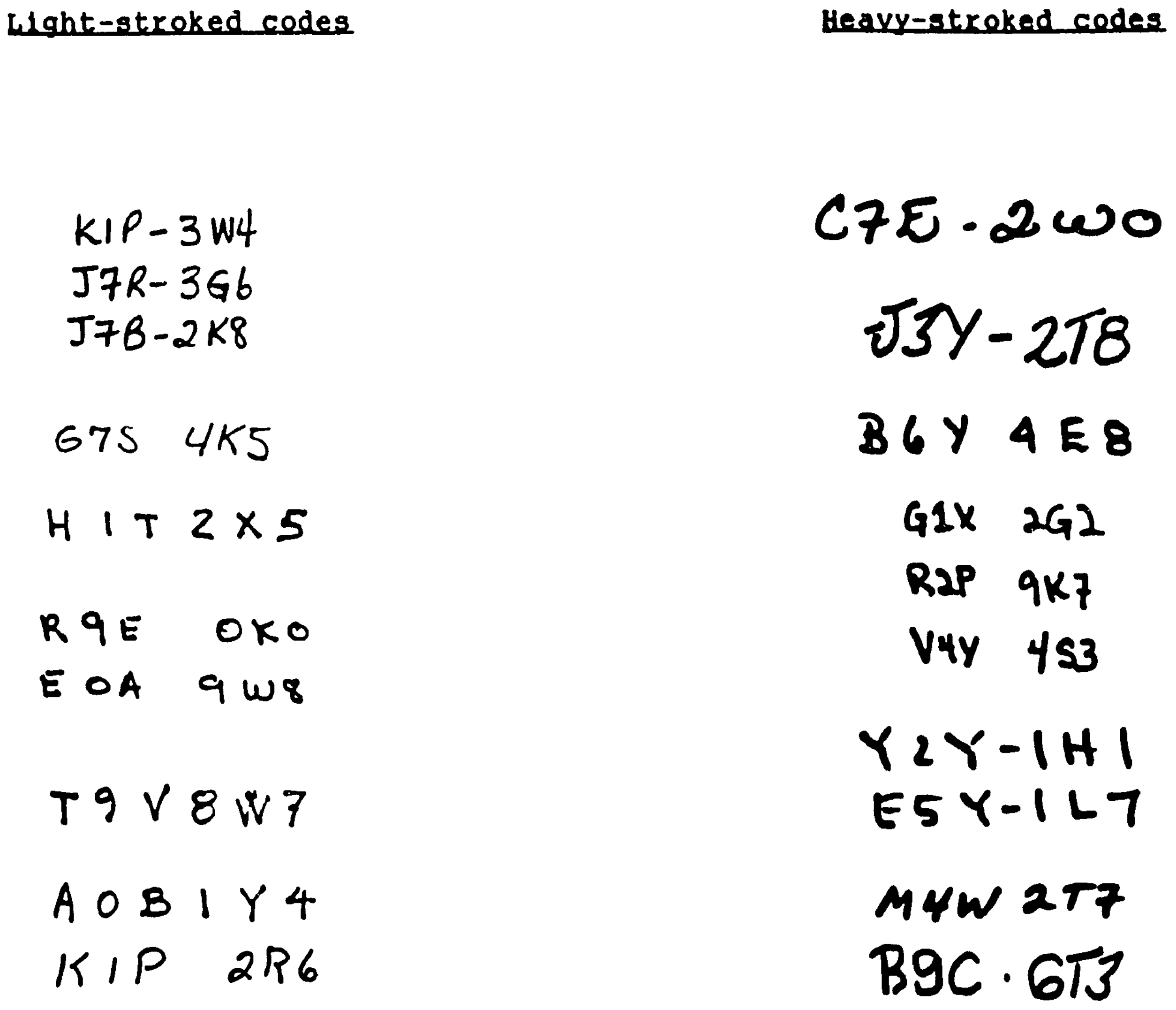


Figure III.1. Specimens of handwritten Canadian postal codes from our experimental data set.

III.2 Preprocessing Algorithm

After each postal code was digitized into a binary image of m rows, $1 \leq m \leq 75$, and n columns, $1 \leq n \leq 500$, we applied a noise removal procedure prior to the actual segmentation; the objective was to make the binary image a smoothed one; that is, removal of salt-and-pepper noise. We adopted a similar algorithm as that described in Unger(1959) and Doyle(1960).

For the sake of easy readability, a semi-formal description of the preprocessing algorithm is given first; we delineate the algorithm in a Pascal-like language afterwards.

We scan the image pixel by pixel. Let p be the pixel under consideration in the postal code image and, n0 to n7, the eight neighbors of p in a 3 x 3 window, as Figure III.2 illustrates. Pixels n0, n2, n4, and n6 are known as the orthogonal neighbors of p.

n3	n2	n1
n4	p	n0
n5	n6	n7

Figure III.2. A pixel p and its neighborhood. Pixels n0, n2, n4 and n6 are referred to as the orthogonal neighbors of p.

For each pixel p in the image we counted up its dark neighbors in a 3 x 3 window. A dark pixel p remained dark if it satisfied any of the following three conditions: i) it had more than 4 dark neighbors; ii) it lay on a diagonal, horizontal, or vertical line; or iii) it was part of a 2 x 2 dark window. See Figure III.3 for the last two conditions. A white pixel p was darkened if it satisfied either one of the following two conditions: i) it had more than 4 dark neighbors, or ii) it had at least 3 orthogonal dark neighbors. See Figure III.4. Figure II.1, which was shown earlier, was obtained after preprocessing the image of Figure I.2.

In the next section, we present our segmentation results on the Canadian postal codes that constituted our data set. To remove any ambiguity from our explanation, we present a Pascal-like formal description of the preprocessing algorithm.

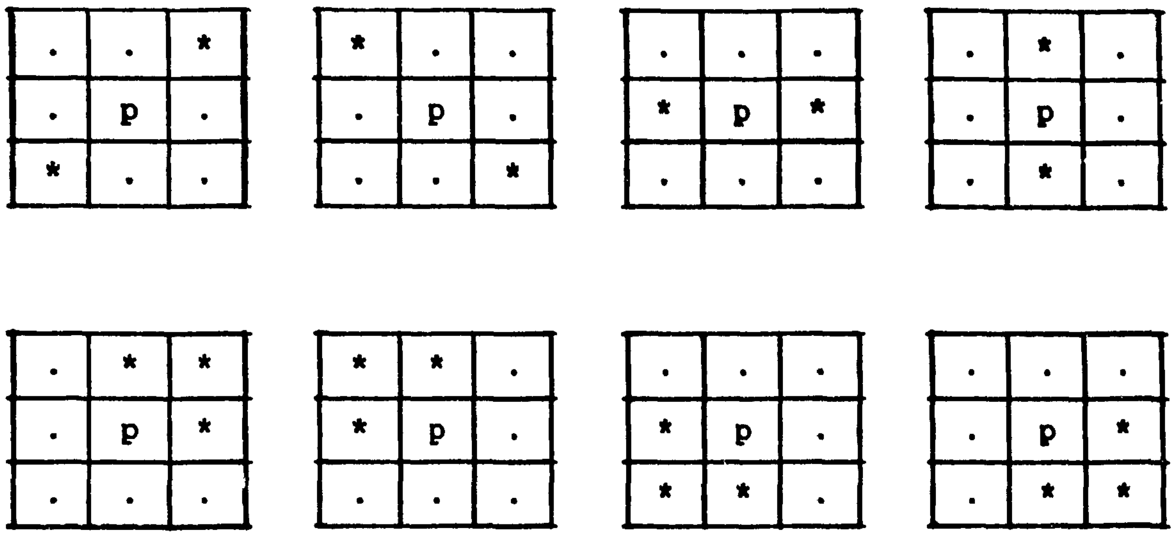


Figure III.3. If the neighborhood of a dark pixel p matched any of the windows above, or if p had more than 4 dark neighbors, p remained dark. '* ' denotes a dark pixel, '. ' a white pixel.

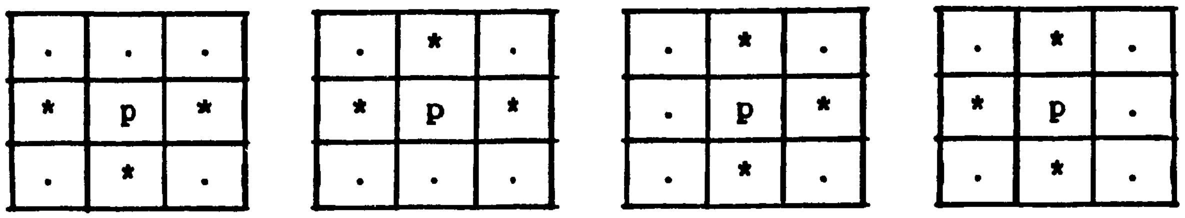


Figure III.4. If the neighborhood of a white pixel p matched any of the windows above, or if p had more than 4 dark neighbors, p was made dark in the image.

```
/...............Declaration of global variables.............../
 :
 :
 :
Type  M = 0..MAXLENGTH ;           / dimensions of binary image /
      N = 0..MAXWIDTH ;
      IMAGE_TYPE = array [M,N] of integer ;

        / type of the image, where M and N are its dimensions /
 :
 :
 :
Var IMAGE : IMAGE_TYPE ;
 :
 :
 :
Procedure PREPROCESS ;

/ Scan of entire postal code image row by row, left to right,
  top to bottom;  process each  pixel  according to its being
  WHITE or DARK.                                             /

  Var I  : M ;
      J  : N ;

  Begin
    I := 0 ;
    While I < MAXLENGTH-1 do
      J  := 0 ;
        While J < MAXWIDTH-1 do
                                                  / p is dark /
          If IMAGE(I+1,J+1) = DARK then REMOVAL_TEST(I+1,J+1)
                                            / else p is white /
          Else FILLER_TEST(I+1,J+1)
        end ;                                    / inner while /
      I  := I + 1 ;
    end ;                                        / outer while /

  end ;                                           / PREPROCESS /
```

```
Procedure FILLER_TEST(X:M; Y:N) ;

/  This procedure counts  the dark neighbors  of white pixel
   (X,Y);  depending on the  outcome of this count, (X,Y) is
   either darkened or left as is.                          /

  Var NEIGHBOR : 0..9 ;
      ROW : M ;
      COL : N ;

  Begin

    NEIGHBOR := 0 ;

    For ROW := X-1 to X+1 do      / Count dark neighbors of p /
     For COL := Y-1 to Y+1 do
      If IMAGE[ROW,COL] = DARK then NEIGHBOR := NEIGHBOR + 1 ;

    Case NEIGHBOR of

      0,1,2   : ;                          / p remains white /

      3,4     : If P has 3 orthogonal dark neighbors

                   Then IMAGE[X,Y] := DARK      / Make p dark /

      5,6,7,8 : IMAGE[X,Y] := DARK             / Make p dark /

    End ; / Case /

  End ;

End ;                                      / FILLER_TEST /
```

```
Procedure REMOVAL_TEST(X:M; Y:N) ;

/  This procedure  counts  the dark neighbors of dark pixel
   (X,Y); depending on the  outcome of this count, (X,Y) is
   either made white or left as is.                        /


  Var NEIGHBOR : 0..9 ;
      ROW : M ;
      COL : N ;


  Begin

    NEIGHBOR := 0 ;
    For ROW := X-1 to X+1 do      / Count dark neighbors of p /
      For COL := Y-1 to Y+1 do
       If IMAGE[ROW,COL] = DARK then NEIGHBOR := NEIGHBOR + 1 ;

    NEIGHBOR := NEIGHBOR - 1 ;    / p is not a dark neighbor! /


    Case NEIGHBOR of

      0,1   : IMAGE[X,Y] := WHITE ;             / Make p white /


      2,3,4 : If p lies on a diagonal,  horizontal, or vertical
              line,

                then IMAGE[X,Y] := DARK      / p remains dark /

              Else if p with 3 of its neighbors form a  2  x  2
              dark window,

                then IMAGE[X,Y] := DARK      / p remains dark /

              Else IMAGE[X,Y] := WHITE        / Make p white /


      5,6,7,8 : ;                            / p remains dark /

    End ;                                            / Case /

End ;                                         / REMOVAL_TEST /
```

III.3 Discussion of Segmentation Results

The results for our segmentation experiment are given in Table III.1. We notice from the table, the correct segmentation rate for light-stroked postal codes (94.6%) is marginally higher than that for heavy-stroked postal codes (92.0%). Moreover, we notice that after the first pass, 1.79% of the light-stroked codes and 0% of the heavy-stroked codes were rejected. The higher rejection rate for the light-stroked codes can be explained thus: because of the lightness of the stroke, some pixels were digitized as white when they really should have been dark. The outcome of this was broken patterns. Consequently, the occurrence of too many fragments was more common. The number of fragments would then exceed the fragmentation threshold and the image would be rejected. This problem was observed not to be so common for heavy-stroked postal codes.

On the other hand, after the second pass, 2.68% of the light-stroked postal codes and 4.26% of the heavy-stroked postal codes were rejected. (13 postal codes were rejected altogether during segmentation.) The higher rejection rate for the heavy-stroked postal codes can be explained thus: because of the heaviness of the stroke, patterns more often touched one another. So, at the end of the second pass, it was more likely that the number of segments was not equal to six, and the image would then be rejected. Furthermore, after the second pass, 0.89% of the light-stroked codes and 3.72% of the heavy-stroked codes were incorrectly segmented (8 postal codes were incorrectly segmented altogether). The higher rate of incorrect segmentation for the heavy-stroked codes is again due to the heaviness of the stroke.

We delved a little deeper into studying how segmentation was affected by each type of irregularity. See Table III.2. For postal codes with at least one broken pattern (23.6% of postal codes), 88.9% of the light-stroked codes and 94.1% of the heavy-stroked codes were correctly segmented. Of the postal codes with touching patterns (9% of postal codes), all light-stroked codes were correctly segmented, but only 44% of the heavy-stroked codes were correctly segmented. This is not surprising when we consider that heavy-stroked postal codes often had touching patterns, as discussed in the paragraph above. Overlapping patterns (28% of postal codes had at least one pattern overlapping another) did not affect segmentation because all such postal codes (both light and heavy-stroked) were always correctly segmented.

Table III.1. EXPERIMENTAL RESULTS ON 300 SEGMENTED CANADIAN POSTAL CODES

	Postal Codes Written with a Light-stroked Pen	Postal Codes Written with a Heavy-stroked Pen	Overall for Both Kinds of Postal Codes
% Correctly Segmented	94.6	92.0	93.0
% Rejected After 1st Pass (#)	1.79	0.0	0.67
% Rejected After 2nd Pass (&)	2.68	4.26	3.67
% Incorrectly Segmented	0.89	3.72	2.67

(#) A postal code was rejected after the first pass, if the number of fragments extracted exceeded the fragmentation threshold of 13.

(&) A postal code was rejected after the second pass if the number of segments was not equal to 6.

Table III.2. PERCENTAGE OF CORRECT SEGMENTATION FOR POSTAL CODES WITH EACH TYPE OF IRREGULARITY

	Percentage of Correctly Segmented Codes		
Type of Irregularity	Postal Codes Written With a Light-stroked Pen	Postal Codes Written With a Heavy-stroked Pen	Overall for Both Kinds of Postal Codes
71 Postal Codes With at Least One Broken Pattern	88.9	94.1	90.1
27 Postal Codes Where at Least Two Patterns Touched	100	44	48.2
84 Postal Codes Where at Least One Pattern Overlapped Another	100	100	100

Continuing on the above theme, we give Table III.3 which displays the outcome of our segmentation algorithm for some specimen postal codes that had irregularities.

Having tested our segmentation algorithm on a data set of 300 Canadian postal codes, we proceeded further and tried to recognize these segmented codes with a known recognition algorithm. The next chapter discusses the recognition aspect of our experiments.

Table III.3. THE OUTCOME OF SEGMENTATION ON SOME SPECIMEN POSTAL CODES THAT HAD IRREGULARITIES.

Postal code	Type of Stroke	Type of Irregularity	Outcome of Segmentation
X1C 4E5	Heavy	Broken	Correctly Segmented
M4W-6G6	Heavy	Touching	Correctly Segmented
K1N 3T6	Heavy	Touching	Rejected After 2nd Pass
M5S 3W2	Heavy	Touching	Incorrectly Segmented
E5Y·5T8	Heavy	Broken and Overlapping	Correctly Segmented
E8E 2W0	Light	Broken	Correctly Segmented
H0H0H0	Light	Broken	Rejected After 1st Pass
M5S-2R8	Light	Touching	Correctly Segmented
A1C-7B2	Light	Touching	Correctly Segmented
H2M 2K8	Light	Overlapping	Correctly Segmented

CHAPTER IV

RECOGNITION OF POSTAL CODES

In this chapter, we first give a brief historical background of postal automation. We then describe a typical pattern recognition system and its components; namely, preprocessor, feature extractor, and classifier. We then proceed to present the techniques we adopted for each component and conclude the chapter by discussing results of experiments performed to recognize the postal codes in our data set.

IV.1 Historical Background of Postal Automation

Attempts have been made in the past to read complete handwritten/typewritten addresses on mail, though, the general trend has been to concentrate the research on postal code recognition. Research has been going on in the United States, Japan, and Europe for the automation of postal code reading by OCRs: Genechi,Mori,Watanabe & Katsugari(1968,1970), Hoffman & McCullough(1971), Atrubin,Fursa & Malaby(1972), Rasch(1972), Schurmann(1976), Rabinow(1981), Fleming & Hemmings(1983), and Busby & Willy(1986). We now give a brief historical background on this subject.

In 1972, the United States Postal Service (USPS) put into operation the OCR II Address Reader in the Boston South Postal Annex (Focht & Burger,1976). The system sorted mail at a rate of 12 pieces/second by automatically reading a wide variety of machine-printed zip codes with an overall corect recognition percentage rate ranging from the mid 70s to the 90s. During this period, the OCR II had no reading capability for handwritten zip codes: the addition of a handwritten zip code recognition capability could have increased OCR II's usefulness, but, writing constraints would have had to be imposed.

The feasibility of using OCR techniques on unconstrained handwritten zip codes is governed by the ability to successfully solve three problems, as stated by Focht & Burger(1976):

1. the ability to detect and locate the zip code.

2. the ability to segment the individual digits of the zip code.

3. the ability to recognize the individual digits comprising the zip code.

Assuming that the zip code could accurately be located, the problem remained one of segmentation and recognition. On examination of handwritten zip codes, it became evident that the zip codes were invariably the last item of the bottom address line, and, secondly, that the individual digits were almost always separated by gaps, that is, not touching; the natural tendency of the human is to print relatively well separated digits in the zip code field. The most common exceptions to this rule are, the "2", "5" and "8", where the last stroke used in forming the digit tends to overlap with the next digit to its right. See Figure IV.1. An analysis of 200 handwritten zip codes showed that 85% of the zip codes had no touching patterns (Focht & Burger,1976).

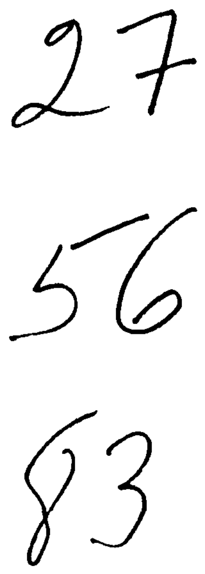


Figure IV.1. The natural tendency of the human being is to print relatively well separated digits in the zip code field. The most common exceptions to this rule are the "2", "5", and "8" as illustrated above. The last stroke in forming the left-hand side digits tends to be drawn towards the right-hand side digits.

In Kegel,Giles & Ruder(1976), a survey was made of OCR users who read numeric handprint "from forms written by the general public (uncontrolled environment)". The survey was conducted on OCRs used by some major firms, including the Japanese Postal Service and USPS. The Japanese postal codes were written within boxes that had been printed along the right-hand edge of most letter envelopes and post cards. Correct recognition rates for these OCRs were 60% when first used in 1968, but gradually went up to 90% by the mid 1970's. (Only the first 3 postal code digits, out of the 5-digit code, were actually read by the OCR).

The USPS hand print tests, however, were aimed at obtaining a better insight into situations that caused rejections of numeric character patterns (the OCR cannot identify the numeric character pattern), and substitutions (incorrect classification of a numeric character pattern). It was reported that 55% of all digit rejects were due to over-printing and erasures. Other common rejections were due to situations involving touching patterns and poor writing of a given digit as Figure IV.2 illustrates.

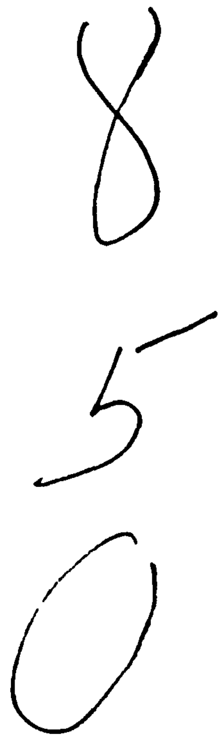


Figure IV.2. Common rejections of zip code digits usually involve touching patterns and poor writing of certain digits; such as, the open top "8", the space between the vertical and horizontal stroke at the top of a "5", and the unclosed "0".

The survey concluded by stating that operational results could be greatly enhanced by soliciting the general public's cooperation in hand-addressed mail; that is, have the public, for instance, print well-defined digits in individual boxes.

In Canada, the OCR machines used for postal code automatic reading are manufactured by Nippon Electric Company (NEC) of Japan. These machines are designed for reading typewritten multi-font postal codes, having close to a 98% correct recognition rate on production runs from one mailer, such as Bell Canada or Bank of Montreal, but may drop drastically (often to 60%) when a batch run consists of an aggregate of envelopes from several large volume mailers.

In the next section, we discuss some of the research work that has been performed on computer recognition of characters. We then present different feature extraction techniques, followed by a description of classification methods.

IV.2 Computer Recognition of Characters

Computer recognition of characters can be divided into three main categories:

1. cursive handwriting.

2. typewritten or machine-printed characters (lowercase, uppercase, or a mixture of both).

3. handwritten characters (lowercase, uppercase, or a mixture of both).

Computer recognition of cursive handwriting is complicated by the lack of character separation. The study of cursive handwriting recognition dates back to 20 years or so. A good survey of the field, up to 1972, can be found in Hannon(1972). The essential difficulty in cursive script recognition is the choice of the entities (single character, word,...) to be recognized. Individual characters would seem a natural choice; unfortunately, the shape of a character in a word usually depends upon preceding and following characters. Moreover, in most cases, a reader will easily recognize a word even if several (maybe all) characters are not recognizable, by making extensive use of contextual information. Consequently, the majority of cursive script recognition work has been approached by using a contextual method; Ehric & Kochler(1975) and Toussaint(1977) are good examples.

Much research has been devoted to the recognition of typewritten and handwritten characters. Since the recognition techniques used in these two categories are similar, we con‹idered it appropriate to merge their discussion. First, in the description that follows, we introduce a typical pattern recognition system. We then examine its components as it relates to the recognition of typewritten and handwritten characters.

A block diagram of a typical pattern recognition system is illustrated in Figure IV.3. A pattern recognition system is usually generated using a two-phased approach. In the first phase, known as the learning phase, labelled (a label consists of the identity of a pattern; for example, the pattern for character "E" would be labelled "E") samples of patterns are input to the system in order to collect a data base of commonly observed features for each pattern class. These data bases make up a set of reference patterns. In the second phase, generally referred to as the testing phase, unknown pattern samples are classified using the feature data base collected in the learning phase. In the next section, we discuss one of the important component of a pattern recognition system, namely, the feature xtractor. Major feature extraction techniques will then be presented.

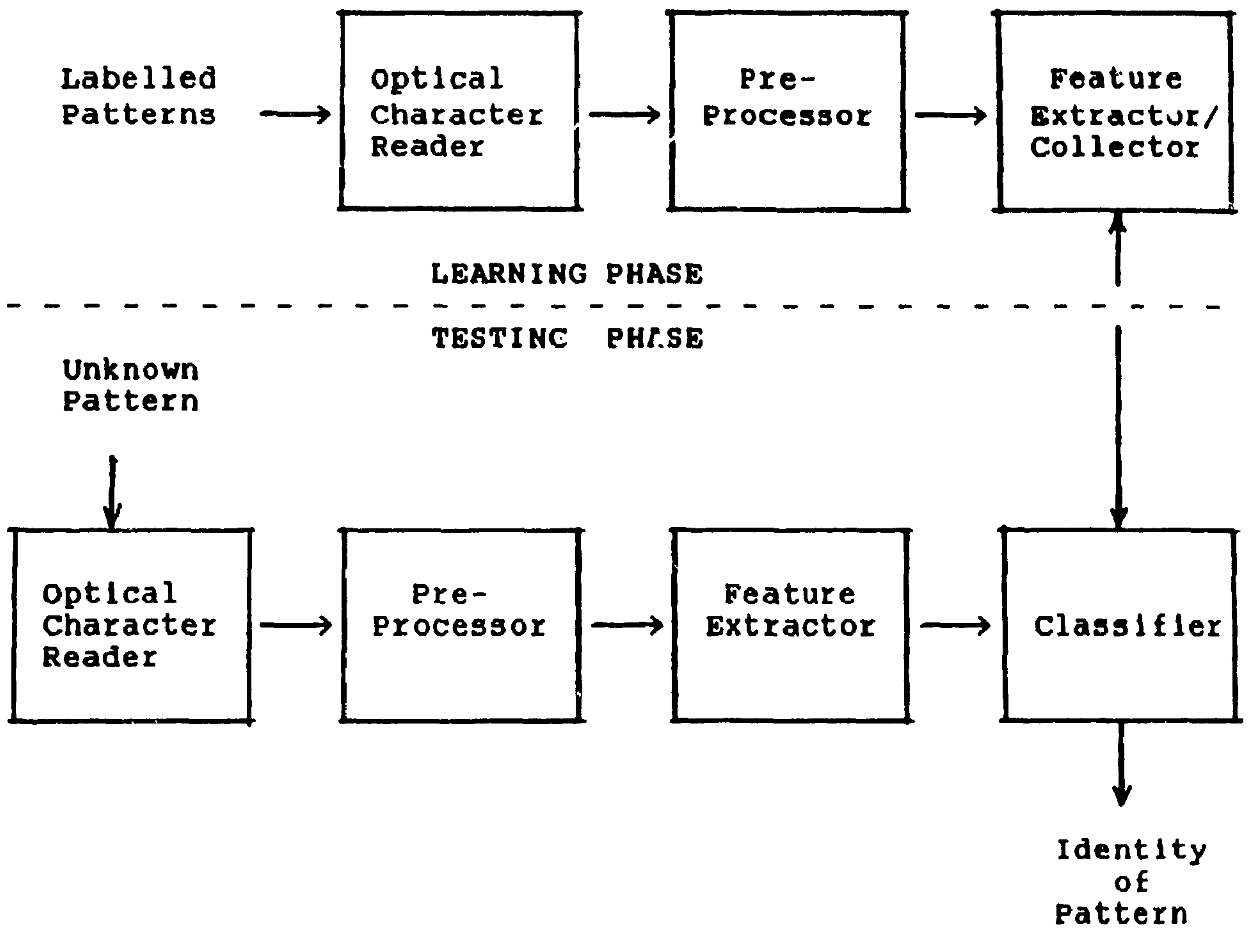


Figure IV.3. Block diagram of a typical pattern recognition system. In the learning phase, labelled pattern samples are digitized and preprocessed before features are extracted from them. These features are then collected as reference patterns. In the testing phase, the unknown pattern is identified by the classifier using a priori information gathered during the learning phase.

IV.2.1 Feature Extraction Techniques

Character patterns are recognized by the features extracted. One of the predominant obstacles in recognizing handwritten characters is the infinite number of possible character shapes generated by different writers. People recognize characters by detecting features, such as curvature of line segments, slant, topology, and overall geometry of a character (Suen,1982A). Over the years, researchers have tried to use this "human approach" with computers to automatically read handwritten and typewritten characters. Before features are extracted, it is common practice to preprocess the character pattern; the component of a pattern recognition system responsible for performing this task is appropriately called the preprocessor. This preprocessing may involve a simple noise removal procedure (as described in Section III.2), or a more elaborate procedure, such as skeletonization (pattern strokes are made one pixel thick), size normalization (pattern is made a standard size), centering (pattern is centered in a matrix), or rotating (pattern is rotated to lie on its central axis). Gudësen(1976) offers a good survey of preprocessing techniques. Preprocessing is usually dependent on the feature extraction technique used. Below, we give a brief explanation of the most commonly used feature extraction techniques.

Position: knowing the position of pixels and their state (dark or white) in the character matrix, templates are created. This method requires that all input patterns be normalized in size before they are matched against the reference patterns. Variations of this method are discussed in Bledsoe & Browning(1959), Kwan,Pang & Suen(1979), Casey & Ji(1983), Sato,Isshiki,Ohoka & Yoshida(1983), and Stentiford(1985).

Density: the density of dark and white pixels can be used to obtain a number of features, for example, the center of gravity, the symmetry or asymmetry of the pattern. This method has been tested by Hussain & Toussaint(1972), Suen & Shillman(1977), and Dasarathy & Kumar(1978).

Distance: by obtaining moments, that is, summing distances of dark pixels from a given reference point in the character matrix, features are extracted which can provide information such as the profile of the pattern. Research on the use of this method is discussed in Tucker & Evans(1974), and Spanjersberg(1974).

Crossings: the number of times a line crosses a dark region from a white region of the character matrix gives the number of compone s the character possesses along the path of the line. Various approaches of the line crossings method have been applied by Glucksman(1967), Knoll(1969), Calvert(1970), Kwon & Lai(1976), and Suen(1982B).

Transformation: the entire matrix is transformed into a series or a vector. The advantage of this method is the significant compressing of the data. Various transformation techniques, such as Fourier transforms, have been experimented with by the following researchers: Ott(1974), Persoon & Fu(1977), Lai & Suen(1981), Moss(1983), and Shridlar & Badreldin(1984).

Physical measurements: the matrix is scanned horizontally and vertically in order to obtain the width and height of the character pattern at each row and column. Different ratios can then be calculated. This approach has been used by Ni,Ding,Gao & Liu(1980).

Line segments and edges: a n x n window (usually 3 x 3), is moved across the entire matrix in order to detect the pattern edges. Computing these edges converts the digital pattern into a line representation. Variants of this method have been used by Yamada & Mori(1978), Yamamoto & Mori(1980), Nadler(1980), and Mantas & Heaton(1983).

Outline of pattern: The dark pixels, on the edge of the pattern, are scanned and their coordinates are registered.

Valuable information can then be extracted, such as, length of line segments, end points, sharp angles and concavities. This approach has been followed by many researchers, of which Pavlidis(1976), Iwata,Yoshida & Tokunaga(1978), Belaid & Haton(1984), and Valdez & Ozkul(1985) provide a few examples.

<u>Center-line</u>: The center-line of the pattern is extracted after it has been properly skeletonized. Since each pixel in the character matrix has two vertical, two horizontal, and four diagonal neighbor pixels, an 8-direction chain code can be extracted, yielding such features as junctions, loops, diagonal lines and forks. This approach has been used by Beun(1973), Fujimoto, Kadota, Hayashi, Yamamoto, Yajima & Yasuda(1976), and Loy & Landau(1982).

We have concisely discussed most of the feature extraction techniques; this should not be construed to encompass all feature extraction techniques. Of course, feature extraction is not a goal in itself but must be coupled with an effective classification method: a successful recognition system is built on the joint operations of the feature extractor and the classifier (component of a pattern recognition system responsible for the assignment of an unknown pattern to a pattern class). With a proper selection of features, both simple and sophisticated classification methods can be implemented. On the other hand, an ineffective feature extraction technique will inevitably lead to an increase in classification errors. In the next section, we examine some of the commmon classification methods used, based on the feature extraction technique adopted.

IV.2.2 Classification Methods

The classifier is, in essence, the decision-maker of the recognition system. As mentioned above, it is largely influenced by the types of features extracted: given a set of features, it should successfully assign the unknown pattern to the class it belongs.

An intuitively appealing approach for classification is "template-matching". In this approach, a set of reference patterns, one or more per pattern class, is collected during the learning phase. The pattern to be classified is then compared with the reference pattern(s) of each class. If a match occurs within a certain tolerance, the pattern is classified, else rejected. The disadvantage of the template-matching method is that, it is rather difficult to select a good reference pattern or patterns for each pattern class, and to define appropriate matching criteria. More sophisticated methods have been adopted over the years. In the discussion that follows, we present some of these methods.

Classification methods are usually divided into two broad categories, namely, statistical and non-statistical (syntactic) classification. In the statistical approach, a set of characteristic measurements, called features, are extracted from the patterns and stored as feature vectors. Classification is then accomplished by partitioning the feature space. In the syntactic approach, after each pattern has been described in terms of simpler subpatterns, classification is performed by using a syntax analysis procedure. Below, we give, for each category, a brief summary of some classification methods.

IV.2.2.1 Statistical Classification

As mentioned above, in the statistical approach, a feature vector is extracted from a pattern. Assume N features are extracted from each input pattern. Each set of N features may be considered as a vector V, called a feature vector, or a point in a feature space. Application of a feature extraction technique such as, position, density, transformation, or crossings, to a binary pattern, usually generates such a feature vector. For example, consider the normalized pattern (defined in Section IV.2.1) for character "6" contained in a 15 x 15 matrix as shown in Figure IV.4. (Assume size normalization consisted in fitting a pattern in a 15 x 15 matrix). An element of the feature vector V, that is, a feature, could represent the number of dark pixels in a 5 x 5 window. Its feature vector V would then contain the following 9 elements: (9,5,5,1,0,0,13,10,13). The problem of classification is to assign each possible vector to its proper pattern class. This can be interpreted as partitioning the feature space into mutually exclusive regions, each region corresponding to a particular pattern class. See Figure IV.5. Mathematically, the problem of statistical classification is formulated in terms of "discriminant functions".

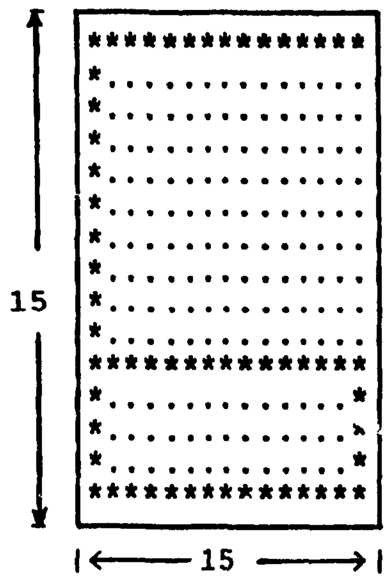


Figure IV.4. Normalized pattern of character "6" contained in a 15 x 15 matrix. A typical feature extraction technique could consist in breaking up the matrix into 5 x 5 windows. The number of dark pixels in each window could then represent a respective element, or feature, of the feature vector V. For the pattern shown above, the feature vector would contain the following 9 elements: (9,5,5,1,0,0,13,10,13). Note, the windows are numbered in increasing order, left to right, top to bottom.

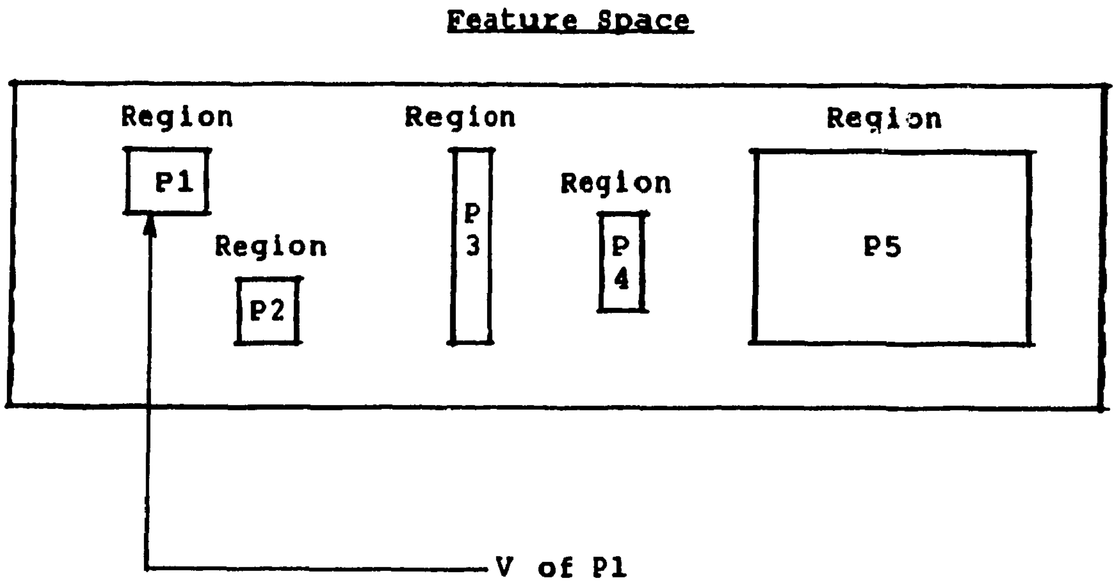


Figure IV.5. Classification of a pattern consists in mapping its feature vector V into its proper region in the feature space, each region corresponding to one of a possible m pattern classes. This mapping is usually performed by means of a "discriminant function". In the above figure, the discriminant function would be responsible for mapping the vector V of pattern P1 into the region P1 of the feature space.

In general, given m possible pattern classes, a discriminant function, $D_j(V)$, associated with pattern class C_j, $1 \leq j \leq m$, is such that, if the input pattern, represented by the feature vector V, is in class C_i, the value of $D_i(V)$ must be the largest. That is for all V belonging to class C_i,

$$D_i(V) > D_j(V), \qquad i,j=1,..,m, \qquad i \neq j \qquad \text{(IV-1)}$$

Thus, the partitions of the feature space between regions associated with pattern class C_i and C_j, respectively, is expressed by the following equation,

$$D_i(V) - D_j(V) = 0$$

Many different forms satisfying condition (IV-1) can be selected for $D_i(V)$. Below, we give a brief explanation of some important discriminant functions used in linear classifiers.

Linear Discriminant Functions: In this approach, a linear combination of features of V is selected for $D_i(V)$. A weight vector W_j for each pattern class C_j is derived from the training pattern samples. Upon satisfying the following condition, the pattern is classified in pattern class C_j.

$$(W_i - W_j) * V > 0, \; i \neq j$$

The feature space is partitioned into regions which contain all of the training patterns of one category and none of the others. This method has been used by Glucksman(1967), Knoll(1969), Loy & Landau(1982), Dinstein & Shapira(1982), Mantas & Heaton(1983), and Valdez & Ozkul(1985).

Minimum Distance Classifier: An important class of linear classifiers is that of using the distances between the input pattern and a set of reference patterns obtained during the learning phase. Suppose that m reference vectors, $R_1, R_2, \ldots, R_m$, are stored, with R_j associated with pattern class C_j. A minimum distance classification scheme with respect to $R_1, R_2, \ldots, R_m$, consists in classifying the pattern in class C_i if $|V-R_i|$ is the minimum, where $|V-R_i|$ is the distance defined between V and R_i. For instance, this distance may be the least-squared distance. Minimum distance classifiers have been used by Suen(1982B), Yoshida(1982), and Shridlar & Badreldin(1984).

Nearest Neighbor Classifier: A nearest neighbor classifier assigns a pattern to the pattern class of the nearest of a set of reference patterns. Suppose that m reference vectors, $R_1, R_2, \ldots, R_m$, are obtained and stored from the learning phase, with R_j associated with pattern class C_j. Furthermore, let the set of reference vectors in R_j be denoted $R_j^{(k)}$, i.e.,

$R_j^{(k)} \in R_j, k=1,\ldots,u_j,$

where u_j is the number of reference vectors in set R_j. A nearest neighbor classification is defined as the smallest distance between an input feature vector V and R_j.

$$d(V,R_j) = \min_{k=1,\ldots,u_j} |V - R_j^{(k)}|,$$

where $R_j^{(k)}$ represents the kth reference pattern within the R_j pattern class. The classifier will assign the pattern to the one which is associated with the nearest set. Experiments using this method have been discussed in Lai & Suen(1981), Casey & Nagy(1982), Stentiford(1985), and Porod & Ferry(1985).

Linear discriminant functions are very common methods used in statistical pattern classification. Another popular method is to use the probability density function of the feature vector V, such as in Bayes classification. In this particular approach, a probability function $P(C_j,n)$ defines the likelihood that a pattern belongs to a given class C_j given a particular feature n, $1 \leq n \leq N$. This function is maximized in order to match the pattern to its proper class. Casey & Jih(1983) and Kahan,Pavlidis & Baird(1987) describe experiments with this method. Other similar approaches use different probability functions, such as a Gaussian function. Work using this latter approach has been discussed in Fleming & Hemmings(1983) and Pavlidis(1983).

As the above discussion shows, statistical pattern classification has been adopted by many researchers. The second kind of classification, syntactic classification, uses a hierarchical (tree-like) representation of patterns. Syntactic classification has become a popular approach of research in the last two decades. We next describe the fundamental principles of syntactic classification.

IV.2.2.2 Syntactic Classification

Syntactic pattern recognition methods have been used in many cases where complex patterns can be described as an assembly of subpatterns. Feature extraction techniques such as outline of character, line segments and edges, or center-line, usually consists in decomposing a pattern into a set of subpatterns. This approach to pattern decomposition draws an analogy between the hierarchical structure of patterns and the syntax of languages. Patterns are specified as building up out of subpatterns just as sentences are built up by concatenating words, and words are built up by concatenating characters.

Evidently, for this approach to be advantageous, the simplest subpatterns selected, called pattern primitives, should be much easier to recognize than the patterns themselves. These primitives can be regions, (as small as single pixels), curve segments, or line segments, as Figure IV.6 illustrates. For instance, a curve segment might be described in terms of its beginning (head), end (tail), and curvature. Similarly, a straight line segment could be characterized by the locations of its head and tail, length, and slope.

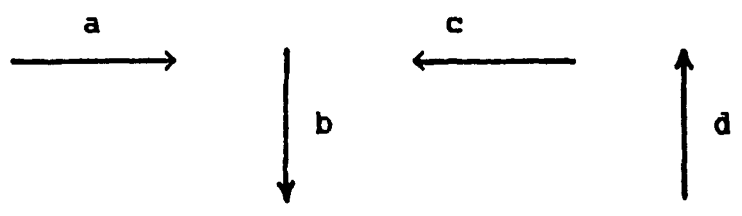


Figure IV.6. Illustrated above are a set of simple pattern primitives that may be used in describing a more complex pattern. The pattern primitives, in this example, are straight line segments with an associated direction. For instance, pattern primitives a and c have the same magnitude but differ in direction.

After primitives are selected, the next step consists in providing the structural description of a pattern in terms of the pattern primitives and their relations. The various relations defined among subpatterns can usually be expressed in terms of logical and mathematical operations. For example, if we choose "head-to-tail concatenation" as the only relation used in describing patterns, then for the pattern primitives shown in Figure IV.6, the pattern for character "6" would be represented by the string "c+c+b+b+b+a+a+d+c+c", as Figure IV.7 illustrates. The selection of a particular description is greatly influenced by the choice of pattern primitives; simple primitives may imply complex pattern descriptions, whereas the use of sophisticated primitives may result in rather simple pattern descriptions and fast classification algorithms. We now discuss some of the common classification methods used in syntactic pattern recognition.

As mentioned at the beginning of Section IV.2.2, template-matching is probably the simplest classification method. A syntactic classifier, using this method, would match an unknown pattern against the reference patterns' primitives collected during the learning phase (based on a matching threshold, the unknown pattern is classified in the same class as the reference pattern which is the "best" to match the unknown). On the other hand, if a complete pattern description is required for classification, a parsing or syntax analysis procedure is necessary; after each primitive within a pattern is identified, the classification process is accomplished by parsing the "sentence" describing a given pattern to determine whether or not it is syntactically correct with respect to pre-specified syntax rules. In between these two extreme situations, there are a number of intermediate approaches. For example, a series of tests can be designed to determine the occurrence or non-occurrence of certain primitives, or combinations of them. The result of the tests, through a table lookup, a decision tree, or a logical operation, is used for the classification decision. Experiments in syntactic pattern recognition have been discussed by: Beun(1973), Persoon & Fu(1975), Pavlidis(1976), Fu(1978), Kovalevsky(1978), Wang(1982), Loy & Landau(1982), Casey & Jih(1983), Sato,Isshiki,Ohoka & Yoshida(1983), and Belaid & Haton(1984).

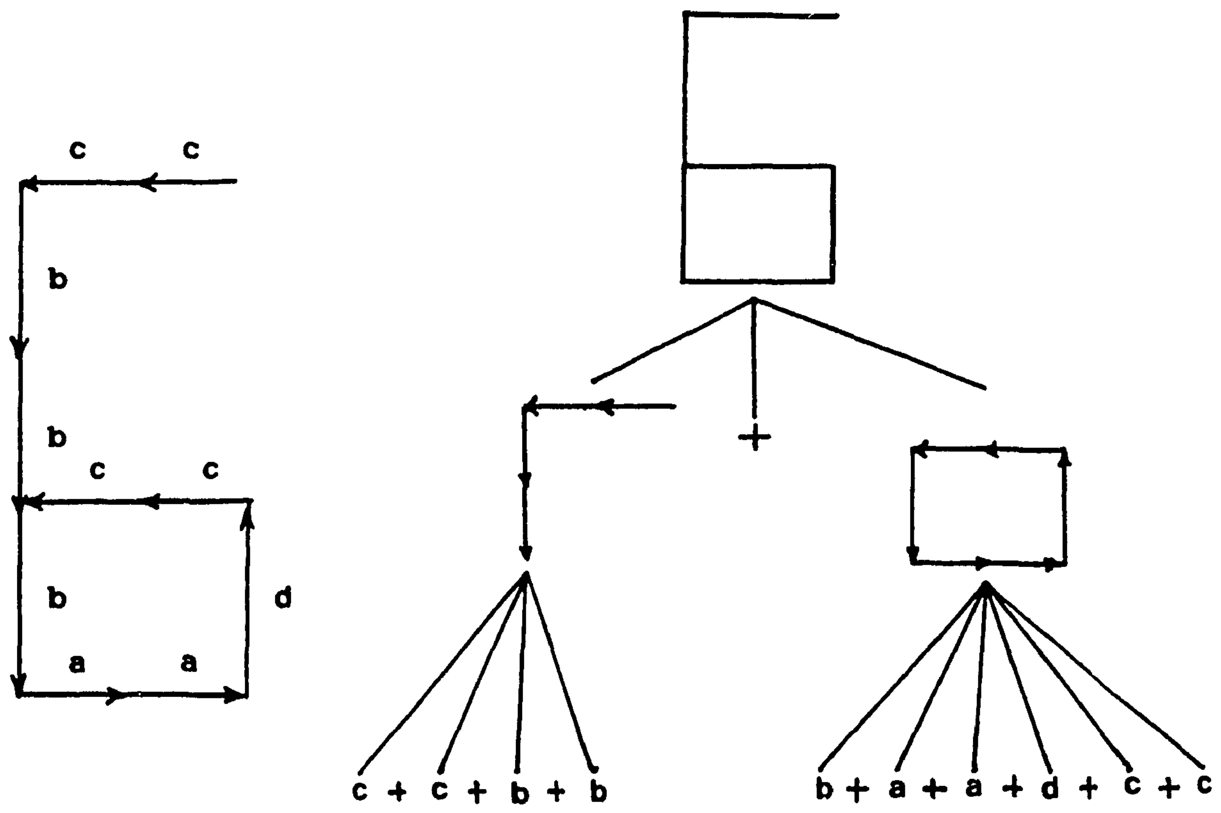


Figure IV.7. Pattern for character "6" showing its hierarchical strutural description in terms of the pattern primitives shown in Figure IV.6. The starting point for describing a pattern is arbitrary. Depending on the choice of pattern primitives, a structural description may not be unique.

We have briefly discussed the fundamental principles of computer recognition as it relates to handwritten and typewritten characters. We have introduced a typical pattern recognition system, divided it into its different components, namely, preprocessor, feature extractor, and, classifier, and discussed each component respectively. In the next section we describe the feature extraction technique and recognition method we adopted for our experiment.

IV.3 Experimental Approach to Postal Code Recognition

In this section, we first examine and discuss our selection of the feature extraction technique. We then proceed to describe the technique. Finally, we present and discuss the classification method we used for recognizing the postal codes.

We have selected characteristic loci as our feature extraction technique. This approach is a variant of the line crossings technique, first discussed by Glucksman(1967). The reasons for this choice were two-fold: on the one hand, it is a well-known method, and, on the other hand, it is relatively easy to implement. Our primary objective was to demonstrate the correctness of our segmentation algorithm.

IV.3.1 Feature Extraction: Characteristic Loci Approach

Recall that the segmentation of a postal code results in the isolation of its constituent patterns. For the Canadian postal codes comprising our data base, our segmentation algorithm confines each pattern in the smallest rectangle of r rows and c columns, $0 \leq r,c \leq 75$, which we shall henceforth refer to as the pattern rectangle (PR). The goal of the characteristic loci approach is to transform a pattern into a feature vector whose elements represents the areas of the pattern's white regions. We now describe how this is performed.

Each white pixel within the PR is considered a position from which the pattern may be observed. If we look only left, up, right, and down, in that sequence, we can count the number of line crossings in each of the four respective directions. Therefore, by observing a pattern as such, we assign a 4-digit code to each white pixel. This code is referred to as a characteristic locus. We restrict the count of line crossings to a maximum of two, any count greater than two being set to two. This restrictive approach has previously been adopted by Glucksman(1967) and Knoll(1969). These four digits form a ternary code. For example, as illustrated in Figure IV.8, the ternary code 1210 is generated for pixel X while pixel Y gives the ternary code 1111.

In the characteristic loci approach, each locus detected in the pattern is used in generating an element of the feature vector. The encoding of the white pixels into ternary codes generates 81 possible loci for each character pattern (each of the four digits of the ternary code may take one of three possible values, $3^4 = 81$). In order to reduce our storage space and computing time, the following subset of twenty-eight loci was selected since these loci were present in most of our training samples, and were considered sufficient to discriminate between two pattern classes. The following twenty-eight loci were selected:

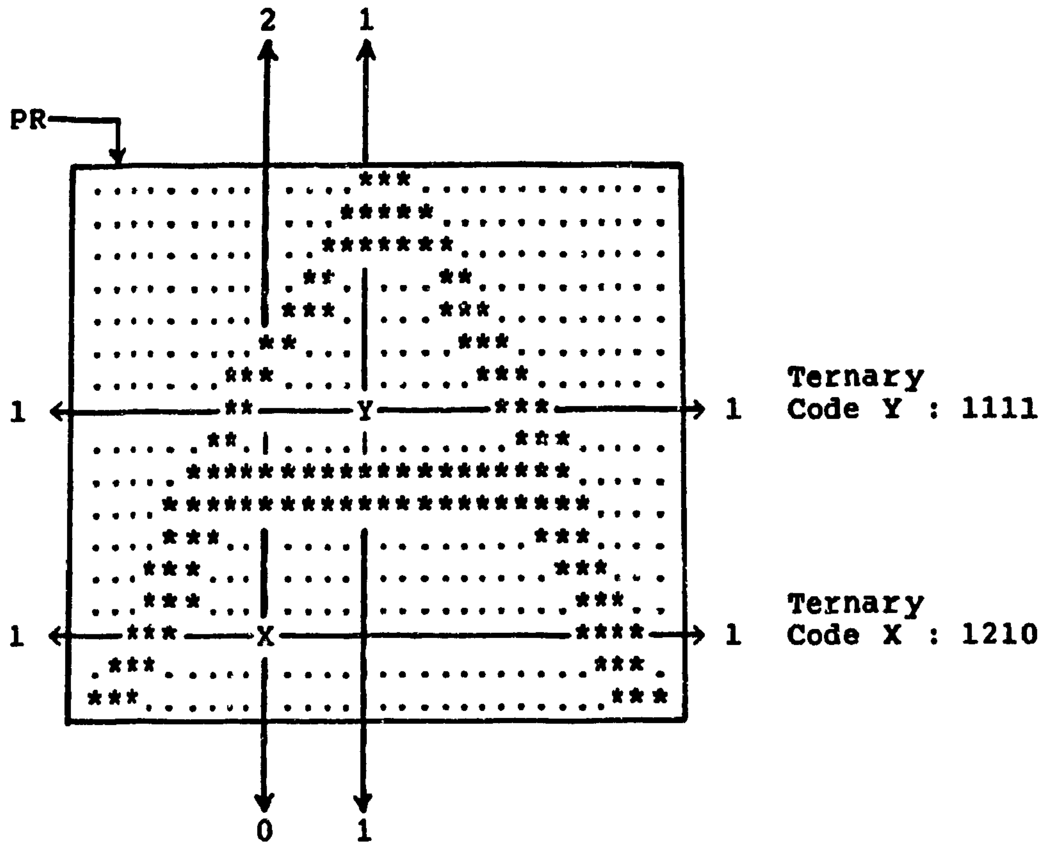


Figure IV.8. Examples of ternary codes generated by the "characteristic loci" algorithm. Pixel X gives the ternary code 1210 and ternary code 1111 is generated for pixel Y.

- out of the sixteen possible loci containing only 0's and 1's (mixture of 0's and 1's), all were selected except, the locus 0000 (which can only be generated by a white pixel in a blank PR), and loci having three 0's (loci which can only be generated by white pixels outside the PR). Therefore, this reduced the number of such loci to eleven.

- the following seventeen loci were also selected because we observed their presence in our training samples. We will enumerate them and give a typical character pattern where the locus was observed.

1201: "E"	1211: "8"
1112: "B"	1120: "N"
2011: "N"	2110: "M"
1210: "A"	1102: "E"
0211: "9"	0112: "2"
2111: "6"	1021: "W"
2112: "R"	1121: "G"
1221: "2"	2211: "6"
1122: "2"	

Hence, the subset we have selected consists of twenty-eight loci. This defines a much smaller feature vector dimensionality. This approach was previously adopted by Glucksman(1967). Figure IV.9 depicts the selected loci.

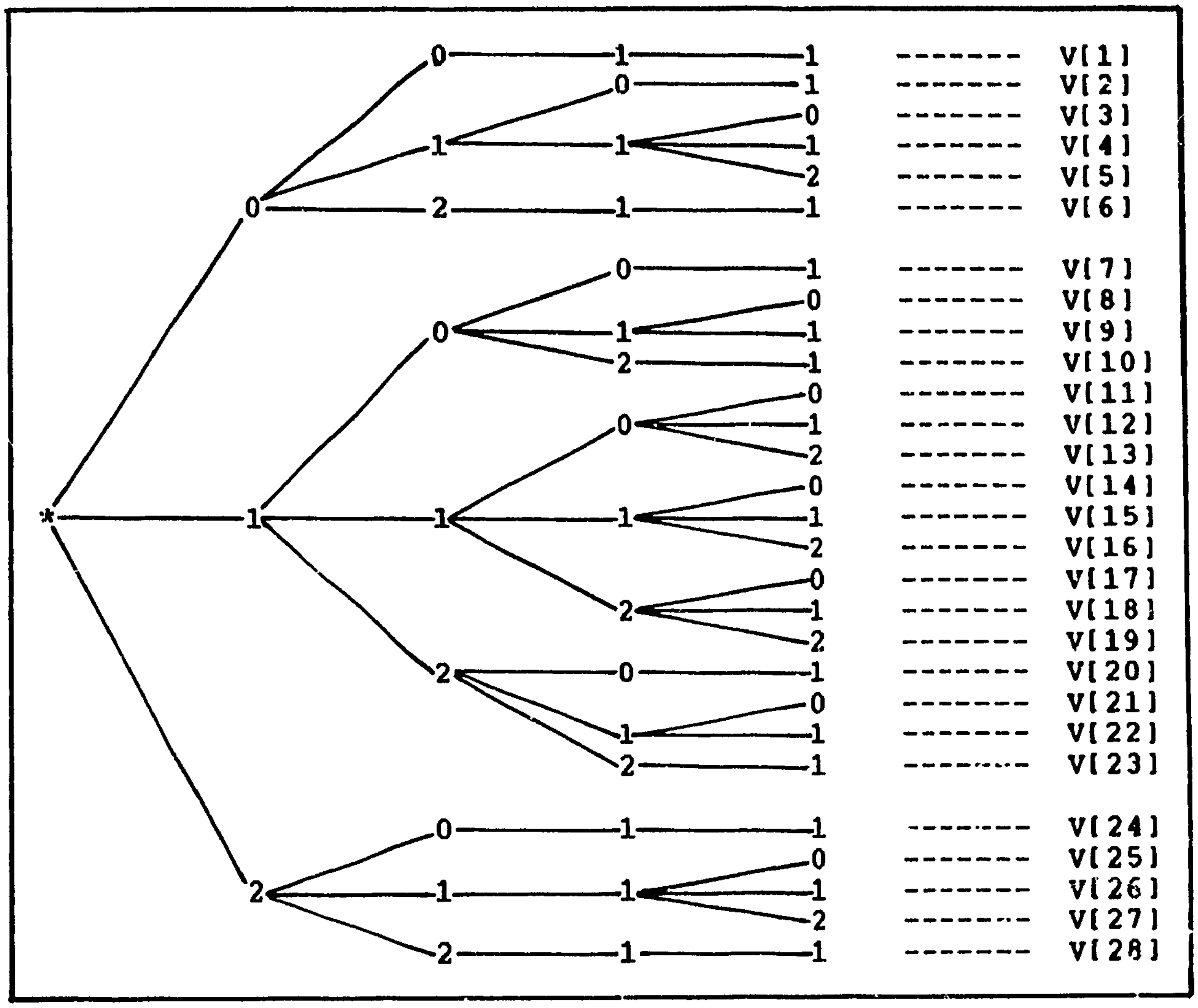


Figure IV.9. Subset of twenty-eight loci (selected out of eighty one possible loci) used in our experiment. The numbers on the right-hand-side (in square brackets) correspond to the feature vector elements.

Next, we explain how a feature vector is generated given the subset of loci. We first initialize to zero the twenty-eight elements of the pattern's feature vector. As mentioned above, the object of the characteristic loci approach is to transform the pattern into a feature vector whose elements represent the areas of the pattern's white regions. By scanning the PR, we obtain a locus for each white pixel within the PR. We count the number of times each respective locus is encountered and express the total count as that element of the feature vector which corresponds to the locus, as defined in Figure IV.9. Thus, if 25 white pixels had locus 0011, then feature vector V[1] would equal 25. Each element of the feature vector may be considered as a "view" of the character pattern from its white background. More precisely it corresponds to the absolute area of each white region (encoded as a locus). As an example, the pattern "S" of Figure IV.10 would generate the feature vector shown in Table IV.1.

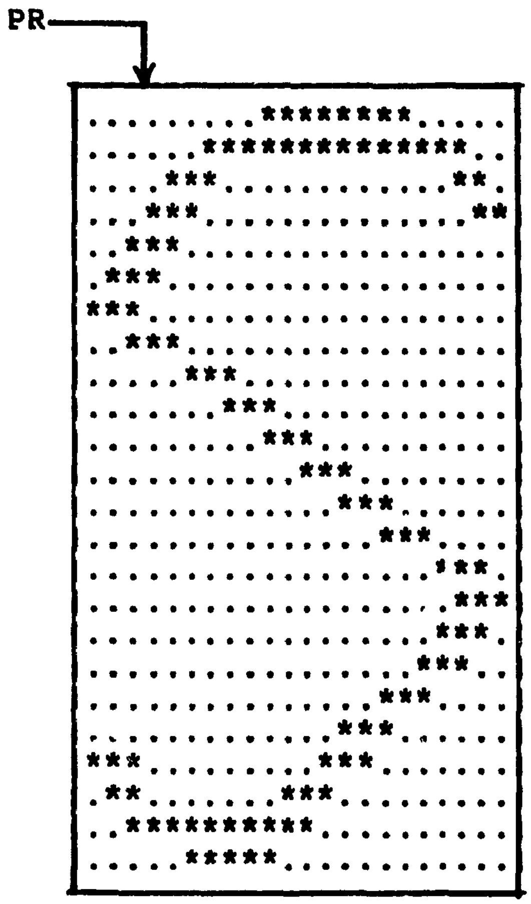


Figure IV.10. The pattern for "S" contained in its PR. The feature vector of above pattern is given in Table IV.1.

Table IV.1. FEATURE VECTOR OBTAINED FOR PATTERN "S" BY CHARACTERISTIC LOCI ALGORITHM

Element number of feature vector	Locus	Number of white pixels having locus	Normalized feature vector
1	0011	25	6.09
2	0101	0	0.00
3	0110	8	1.95
4	0111	2	0.49
5	0112	0	0.00
6	0211	152	37.07
7	1001	8	1.95
8	1010	0	0.00
9	1011	0	0.00
10	1021	0	0.00
11	1100	51	12.44
12	1101	1	0.24
13	1102	121	29.51
14	1110	0	0.00
15	1111	0	0.00
16	1112	26	6.34
17	1120	0	0.00
18	1121	0	0.00
19	1122	0	0.00
20	1201	0	0.00
21	1210	0	0.00
22	1211	16	3.90
23	1221	0	0.00
24	2011	0	0.00
25	2110	0	0.00
26	2111	0	0.00
27	2112	0	0.00
28	2211	0	0.00

In order to make the feature vector independent of the PR's size, we normalize it. This is performed by dividing each element of the feature vector by the total number of white pixels in the PR. (For the pattern of Figure IV.10, the total number of white pixels is 410). We further multiply each element by one hundred in order to express the loci as relative percentage areas of white regions in the PR (see last column in Table IV.1).

The feature vector thus extracted is found to have the following desirable characteristics:

1. normalizing the feature vector, we make the feature extraction method independent of the pattern's size.

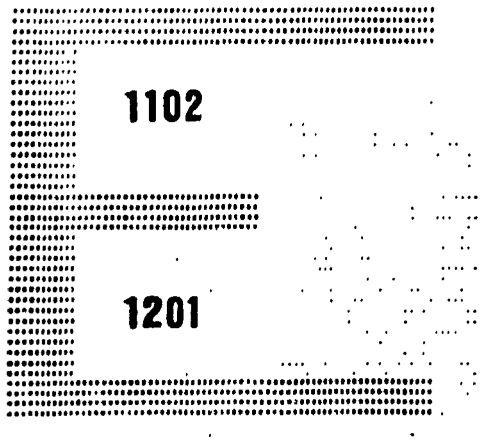
2. The operations on distinct rows and columns are completely independent and can proceed in any time sequence desired. A great deal of flexibility in the amount of parallelism can thus be incorporated when implementing the algorithm (since no parallel computing facilities were available at the time of the experiment, parallelism was not included in the feature extraction algorithm).

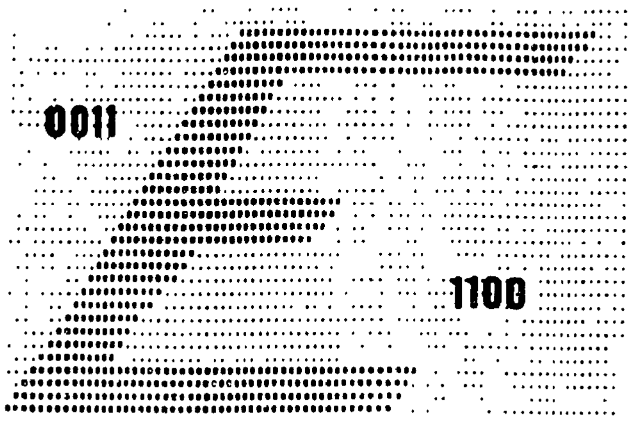
In the learning phase of our pattern recognition system, we extracted feature vectors from our training postal code samples by the method delineated above. Our 150 training samples consisted of an arbitrary selection of some of the codes written from each of the 15 people. Upon careful examination of the features extracted from these training samples, we selected our subset of twenty-eight loci. We also gathered a set of reference vectors, one or more per character class, which was subsequently used in building the decision trees to be described in the next section.

IV.3.1.1 Classification Using Decision Trees

To classify a pattern we need to assign it to the class it belongs using the feature vector extracted. Our classification scheme consists in using decision trees. We use a combination of a syntactic and statistical approach to classification; each locus detected in a pattern is considered a pattern primitive (syntactic), and the size (magnitude) of the elements of the feature vector helps discriminate different pattern classes (statistical). The classification trees are divided into two major classes, namely, numeric and alphabetic (recall that a Canadian postal code is alphanumeric). Each subclass is further divided into two decision trees: a top-level decision tree designed to examine the presence and magnitude of feature vector elements expected in good-quality patterns (described in the next paragraph); a low-level decision tree designed to examine the presence and magnitude of certain feature vector elements that occurred in our training samples. If a pattern cannot be classified by the top-level decision tree, the low-level decision tree is invoked. In the discussion that follows, we explain how the decision trees were designed.

We designed our top-level decision trees to handle all good-quality patterns, that is, patterns that were obtained from characters that had well-defined shapes. See Figure IV.11. It is our estimate that 20-25% of our experimental data set contained good-quality patterns. Based on this assumption, we designed a top-level decision tree for both the numeric and alphabetic characters. This approach was adopted in order to minimize the classifying time; a good-quality pattern could potentially be classified after a few tests only as discussed below. However, our classification scheme would be quite inefficient if we did not make provision for poor-quality patterns which are rejected from our top-level decision tree. The poor-quality patterns require a low-level decision tree. Having adopted the same approach for designing both the alphabetic and numeric decision trees, we next describe their overall conceptual design.





Good-quality pattern

Poor-quality pattern

Figure IV.11. The left-hand side pattern illustrates what we refer to as a good-quality pattern. Such patterns are obtained from characters that are not slanted, and whose shapes are well-defined. The right-hand side pattern illustrates what we refer to as a poor-quality pattern. Such patterns are usually obtained from characters that are slanted and do not possess well-defined shapes. The pattern for character "E", on the left-hand side, would generate a feature vector having loci "1102" and "1201". However, the pattern for character "E", on the right-hand side, would likely generate some unexpected loci due to its being slanted. These unexpected loci ("0011" and "1100") hinder the classification process; such poor-quality patterns must be classified using more elaborate tests.

The top-level decision trees are designed to examine the occurrence of loci expected from good-quality patterns; if the feature vector elements corresponding to these loci exceed a certain magnitude (as would be expected from good-quality patterns), a pattern is usually assigned to its proper class without difficulty. Moreover, the decision trees test the occurrence of broken strokes within a pattern. This information is obtained from our segmentation algorithm and helps us classify patterns. For instance, given that a pattern is broken, the top-level decision tree first examines the occurrence of certain loci expected from good-quality patterns to classify such a pattern into the most likely classes which tend to have broken strokes, that is, "5", for numeric, and, "E", "H", "J", "T", and "Y", for alphabetic. We observed such broken patterns on postal code samples.

Thus, in the learning phase, by repeatedly submitting our training samples to our top-level decision tree and comparing their classification outcome, (correctly recognized, misrecognized, or rejected) gradual refinements were made. Figures IV.12 and IV.13 depict a refined version of the alphabetic and numeric top-level decision trees respectively.

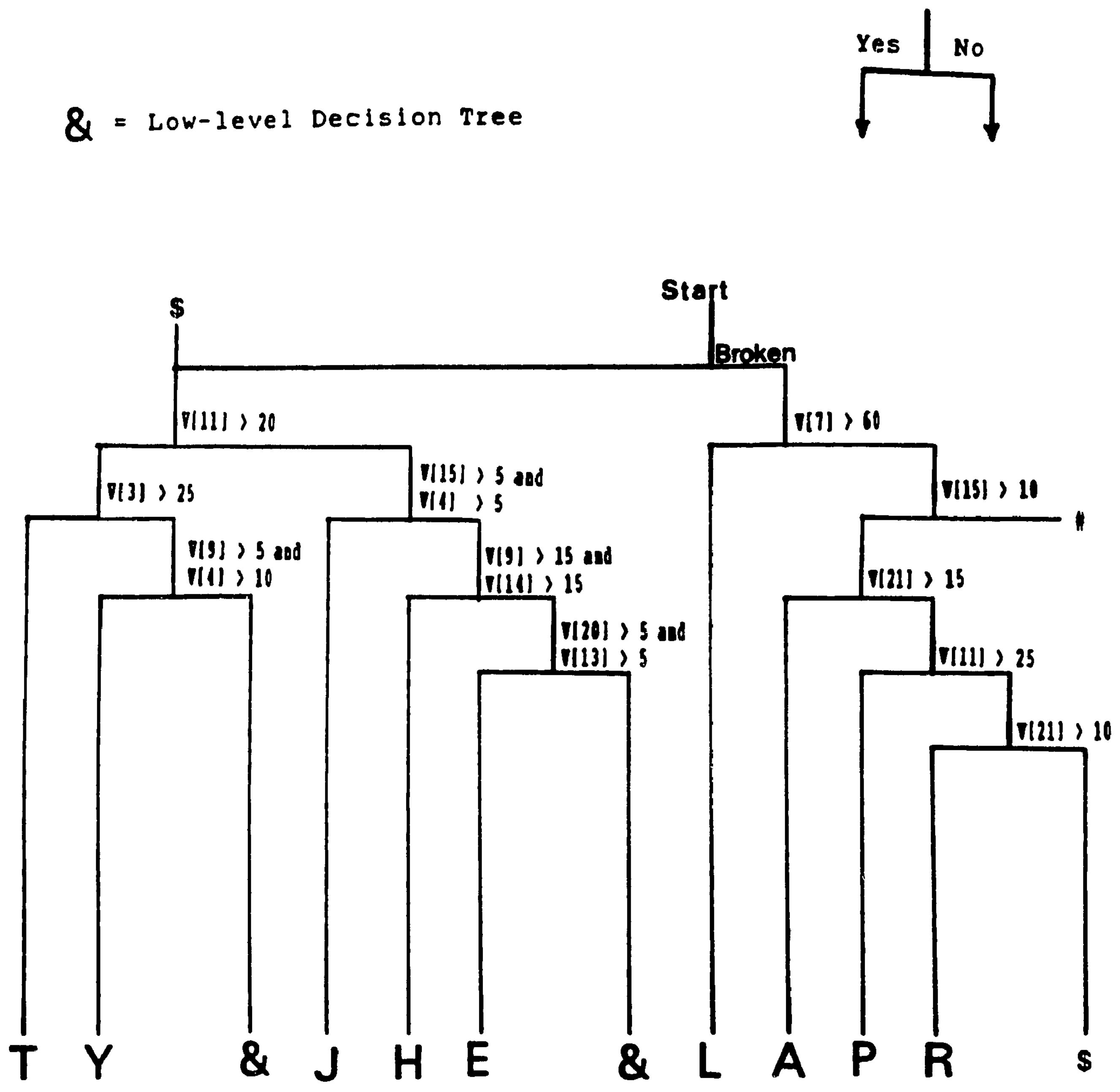


Figure IV.12. Top-level decision tree for alphabetic characters used in our experiment.

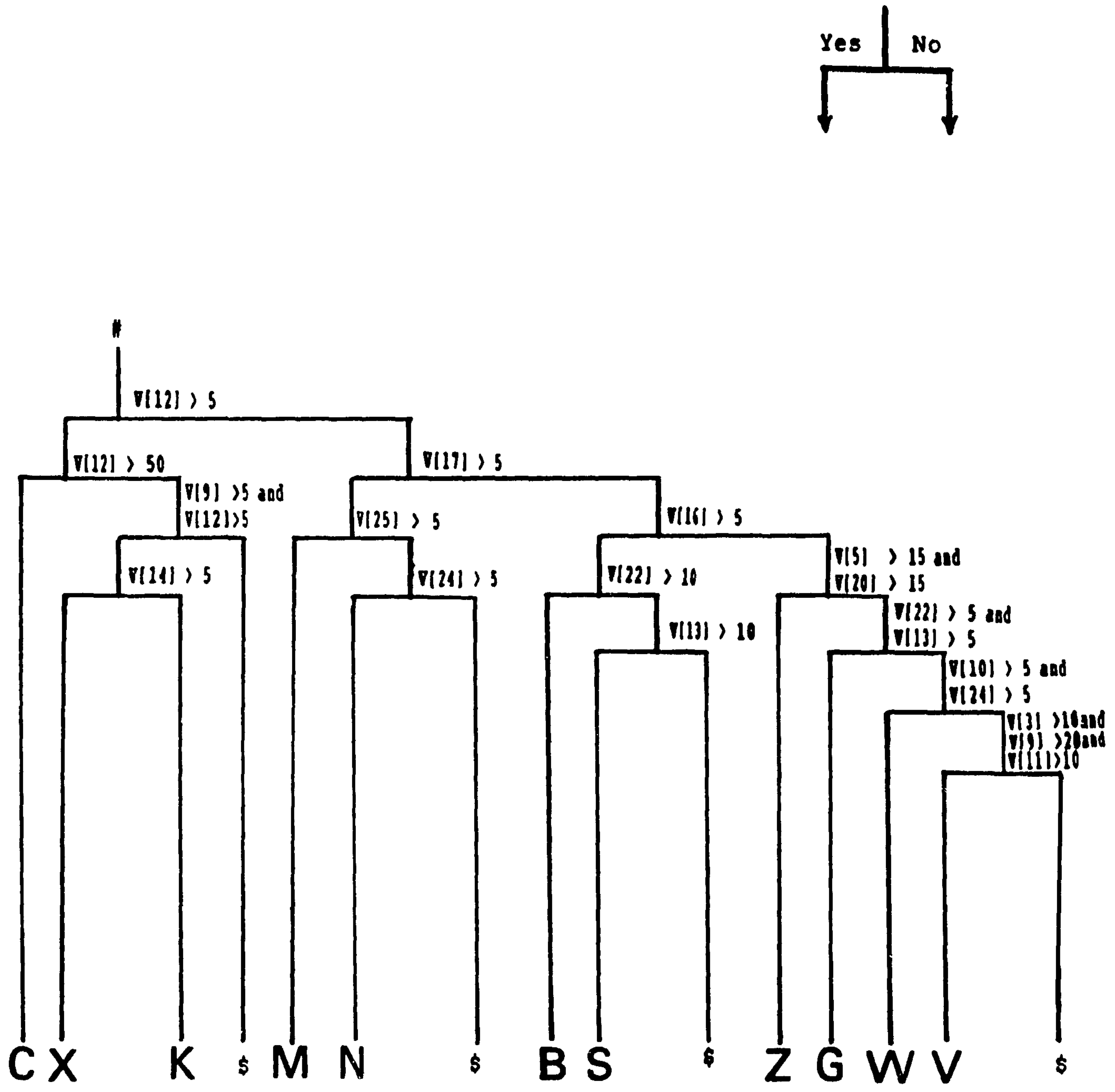


Figure IV.12 (cont'd). Top-level decision tree for alphabetic characters used in our experiment.

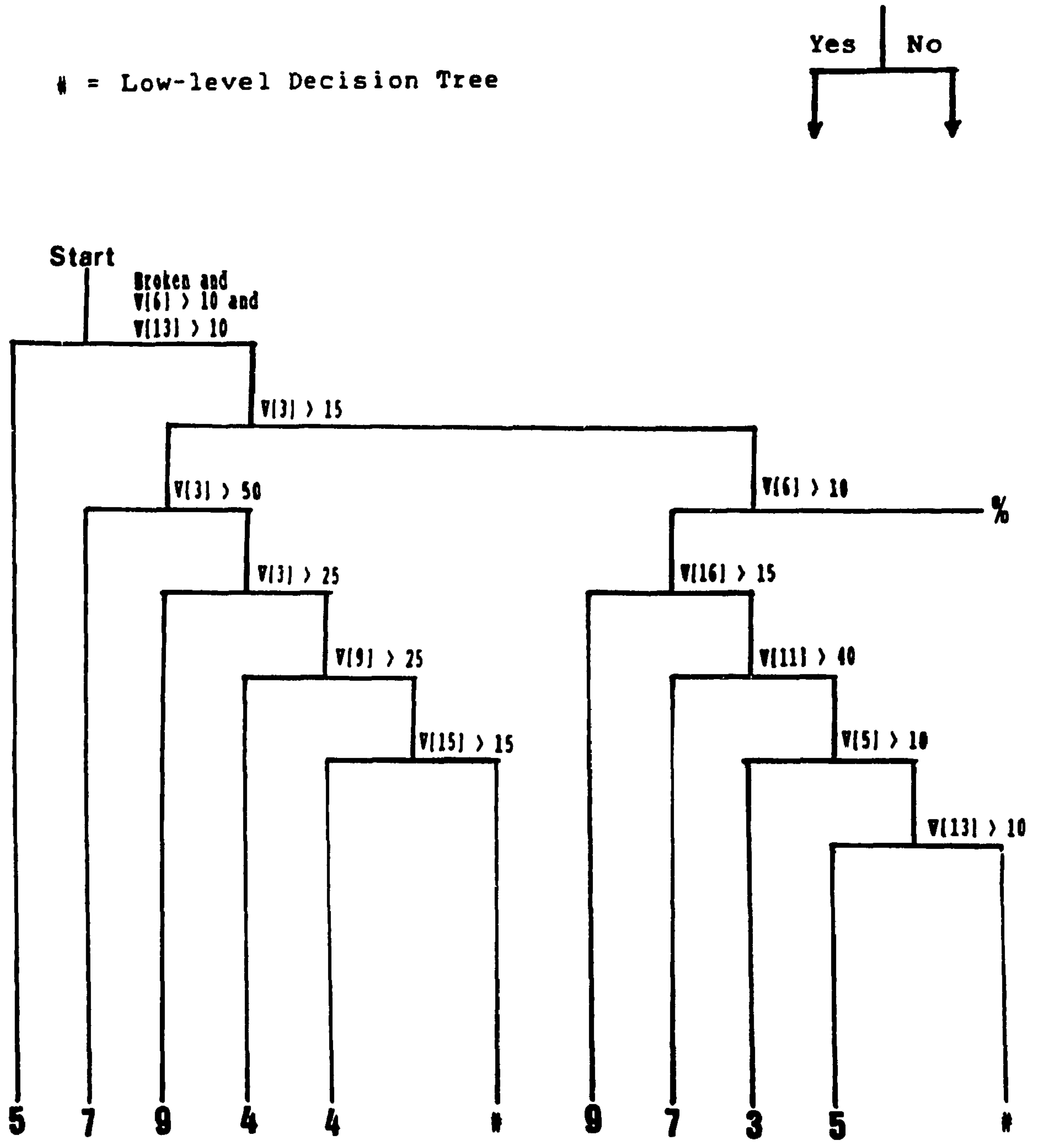
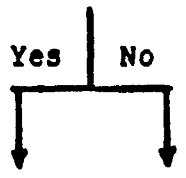


Figure IV.13. Top-level decision tree for numeric characters used in our experiment.

= Low-level Decision Tree



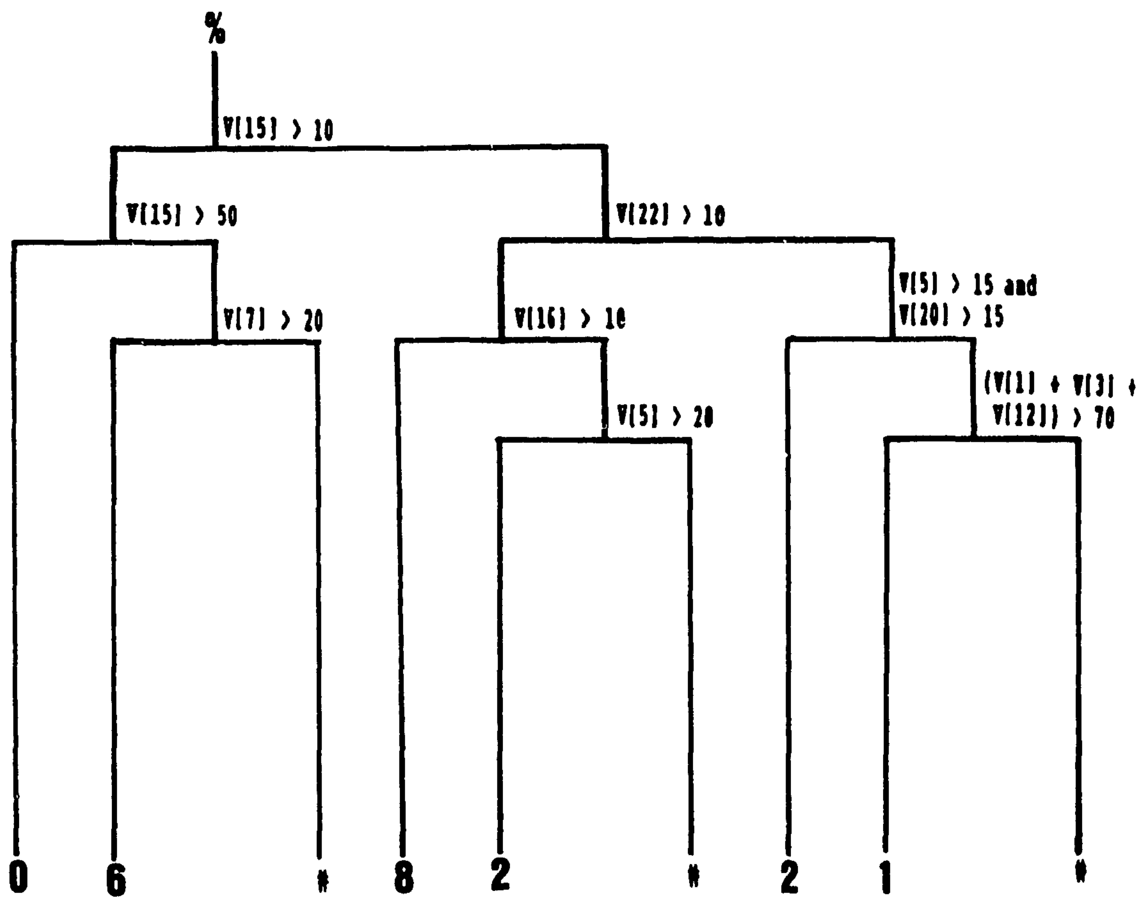


Figure IV.13 (cont'd). Top-level decision tree for numeric characters used in our experiment.

As mentioned earlier, poor-quality patterns require further processing to be classified. Hence, if a pattern cannot be classified using the top-level decision tree, the pattern is subjected to more rigorous tests in the low-level decision tree. We next explain how these low-level decision trees (alphabetic and numeric) were designed.

The learning phase enabled us to build these low-level decision trees; patterns that were rejected by the top-level decision tree had their feature vectors collected into a data base, one per each pattern class. These reference vectors were then used in order to build the low-level decision trees whose complexities make it virtually impossible to depict pictorially. To exemplify the type of testing required in the low-level decision trees, we refer to Figure IV.11. As mentioned in the legend, the right-hand side pattern of character "E" would inevitably generate a vector having loci "0011" and "1100". Hence, since this pattern would not be expected to be classified by the top-level decision tree, the low-level decision tree would incorporate the examination of these specific loci. The expected loci from a good-quality pattern of character "E", such as the one on the left-hand side of Figure IV.11, would also be examined, their presence being expected in lesser magnitude. See Figure IV.14. Finally, if a pattern reaches the low-level decision tree and cannot be classified, it is rejected.

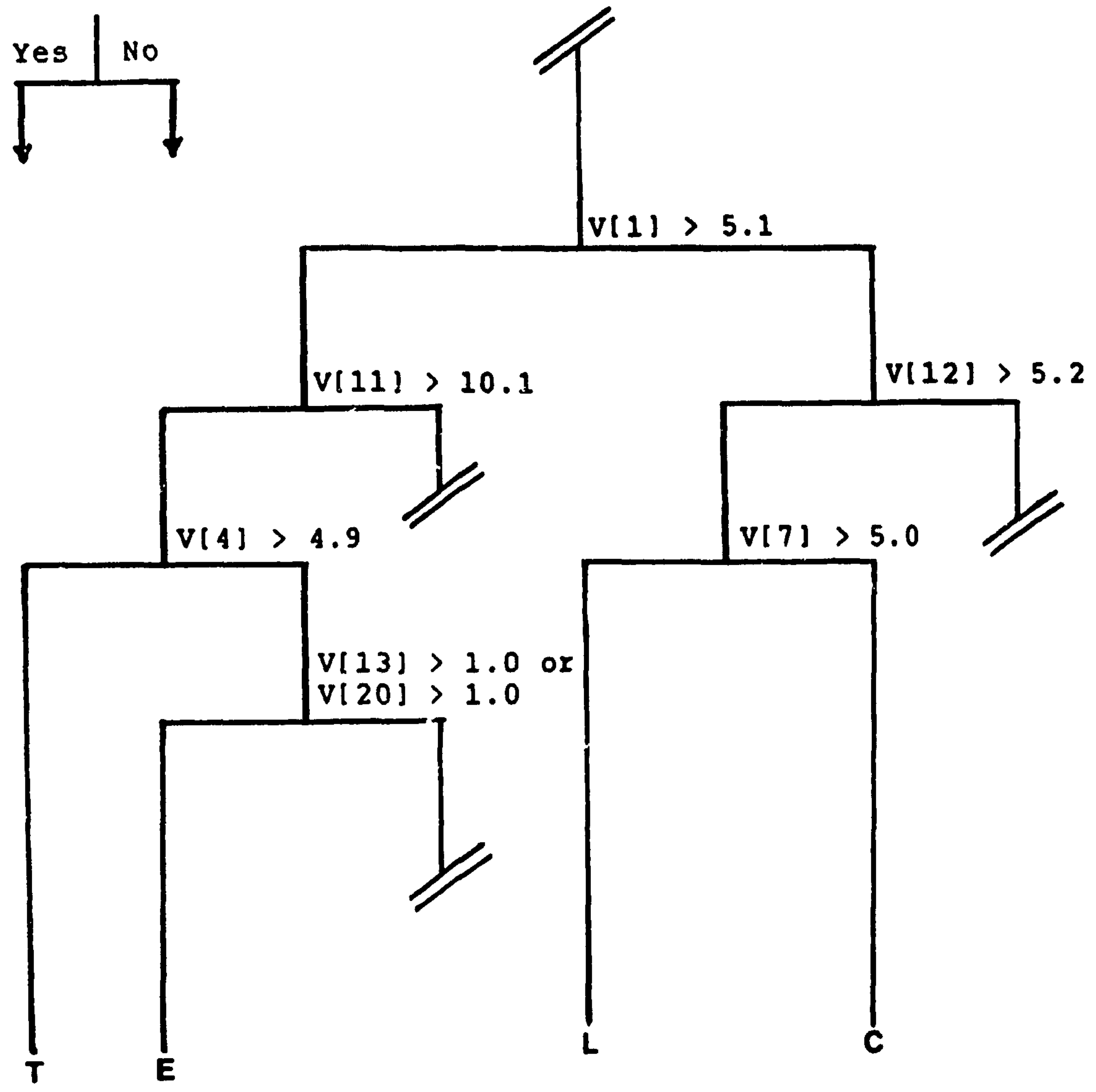


Figure IV.14. Section of low-level decision tree for alphabetic characters. This branch of the tree tests the presence and magnitude of certain loci detected in poor-quality patterns from our training samples. As the figure illustrates, these tests are designed to discriminate patterns "C", "E", "L", and "T".

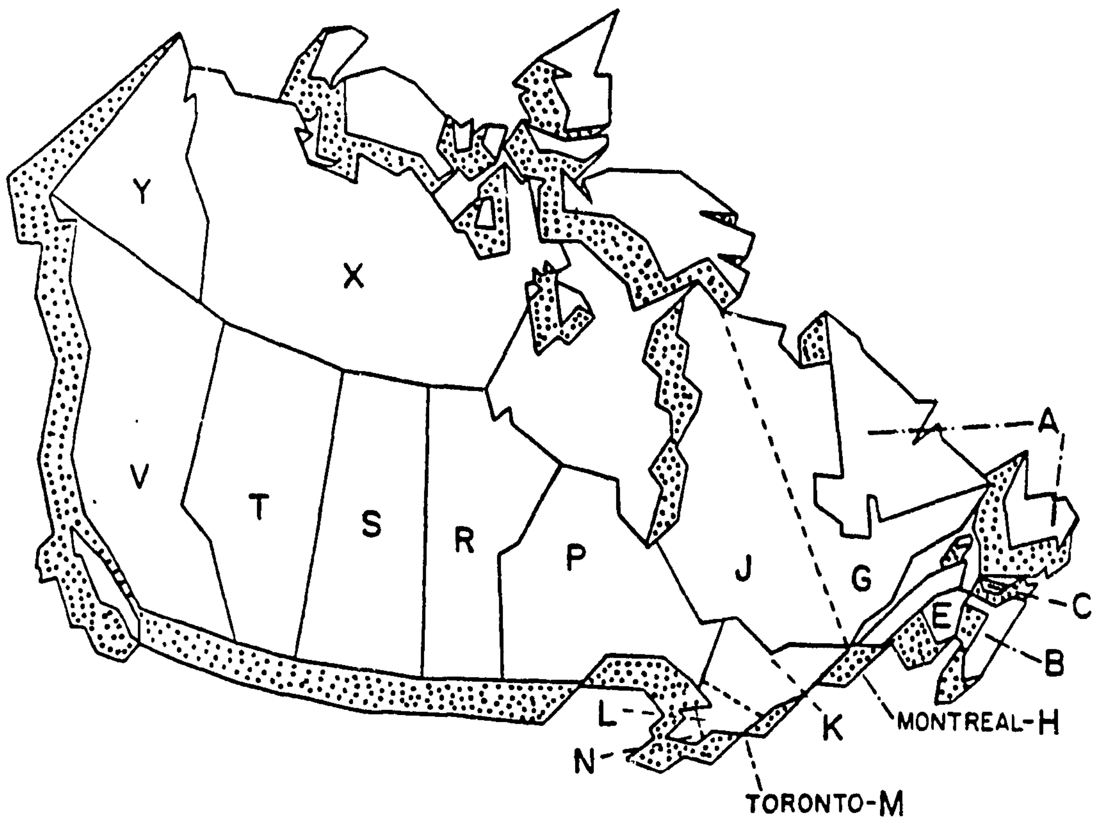
We have explained our motivation for adopting the characteristic loci approach and reviewed main feature extraction techniques. We further surveyed major classification methods and discussed the one we have adopted. In the next section, we present the experimental performance of our classification method.

IV.4 Discussion of Recognition Results

In Section III.1 we presented the format prescribed by Canada Post Corporation for Canadian postal codes. In order to clarify the following presentation of our experimental results, we shall first introduce some terminology pertaining to postal code structure.

A Canadian postal code consists of a FSA and LDU. The leftmost three characters are referred to as the FSA, Forward Sortation Area, and usually indicate a city or rural area. The rightmost three characters are referred to as the LDU, Local Delivery Unit, and usually indicate a street block or a major building. The postal code allocation was initiated in the early 1970's and there are now presently over 6,000 "official" destinations in Canada residing in the ten provinces and the two territories. Figure IV.15 depicts the provincial geographic boundaries along with the allocated geographic designators, that is, the leftmost character of the postal code. We now present our recognition results.

The overall system recognition results for the postal codes are given in Figure IV.16. We obtained a 60% recognition rate, with 9.3% misrecognitions and 30.7% rejections. We shall examine the results in more detail, but first, we must acknowledge the fact that, out of the 300 original postal codes, 13 were rejected by our segmentation algorithm (see Section III.3). Therefore, in further analyzing our results, we shall only consider the 287 remaining postal codes.



A	Newfoundland/Terre-Neuve
B	Nova Scotia/Nouvelle-Écosse
C	Prince Edward Island/Île-du-Prince-Édouard
E	New Brunswick/Nouveau-Brunswick
G	Quebec East/Est du Québec
H	Montreal Metropolitan/Montréal Métro
J	Quebec West/Ouest du Québec
K	Eastern Ontario/Est de l'Ontario
L	Central Ontario/Centre de l'Ontario
M	Toronto Metropolitan/Toronto Métro
N	Southwestern Ontario/ Sud-Ouest de l'Ontario
P	Northern Ontario/Nord de l'Ontario
R	Manitoba
S	Saskatchewan
T	Alberta
V	British Columbia/Colombie-Britannique
X	Northwest Territories/ Territoires du Nord-Ouest
Y	Yukon

Figure IV.15. The Canadian postal coding system in relation to the geographic boundaries. Starting in the East, the leftmost character of the postal code (the geographic designator), "A", was allocated to Newfoundland/Labrador. The coding was then extrapolated westwards, Quebec and Ontario being the only two provinces to be allocated more than one geographic designator because of their higher population density.

We observe from Figure IV.17, that the recognition rate rises to 62.7%, with 9.8% misrecognitions and 27.5% rejections, when only the 287 codes that were submitted to the recognition unit are considered. Note, that, although similar results for the postal codes written with a light-stroked pen and a heavy-stroked pen were obtained during segmentation (see Table III.1), a significant difference can be observed in the recognition results; the light-stroked codes had a better recognition rate, that is, less misrecognitions, and less rejections.

Figure IV.18 displays the postal code recognition results by individual writer, from best to worst, that is, from the highest recognition rate to the lowest. Note that the worst five writers used a heavy-stroked pen, each having a recognition rate of 50% or less. Moreover, these five writers accounted for 47.8% of all rejections. These five writers produced poor-quality patterns (discussed in Section IV.3) that accounted for the majority of postal codes having one or more of the three types of irregularities discussed in Section I.2 and illustrated in Table III.2.

Number of postal codes	Number of codes correctly recognized	Number of postal codes misrecognized	Number of postal codes rejected
300	60.0%	9.3%	30.7%

Figure IV.16. The above table displays the overall system recognition results. Out of 300 digitized postal codes, 60% were correctlly recognized, 9.3% were misrecognized, and 30.7% were rejected (including the 13 codes rejected during segmentation).

	Number of postal codes	Number of codes correctly recognized	Number of postal codes misrecognized	Number of postal codes rejected
Light-Stroked Codes	107	74.8%	6.5%	18.7%
Heavy-Stroked Codes	180	55.6%	11.7%	32.8%
Total	287	62.7%	9.8%	27.5%

Figure IV.17. The above figure displays the recognition error rate excluding the 13 postal codes rejected during segmentation. Of the 28 misrecognized codes, 19 had 1 misrecognized character while 9 had 2. Also, of the 8 incorrectly segmented codes (see Section III.1), 6 were rejected and the other 2 were misrecognized.

Writer	Light Stroked	Heavy Stroked	Number of Codes Recognized	Number of Codes Misrecognized	Number of Codes Rejected
1	22		95.5%	4.5%	0.0%
2	19		84.2%	5.3%	10.5%
3		21	81.0%	14.3%	4.8%
4	17		76.5%	5.9%	17.7%
5		32	68.8%	12.5%	18.8%
6	12		66.7%	16.7%	16.7%
7	18		66.7%	11.1%	22.2%
8		20	65.0%	20.0%	15.0%
9		20	55.0%	20.0%	25.0%
10	19		52.6%	0.0%	47.4%
11		14	50.0%	0.0%	50.0%
12		21	47.6%	14.3%	38.1%
13		21	47.6%	4.8%	47.6%
14		21	38.1%	9.5%	52.4%
15		10	20.0%	0.0%	80.0%
TOTAL	107	180			

Figure IV.18. Postal code recognition rate by individual writer (best to worst). The worst five writers accounted for 47.8% of rejections.

Figure IV.19 displays the character recognition rates for the alphabetic and numeric characters respectively (recall, a Canadian postal code has 3 alphabetic and 3 numeric characters). Each subclass of characters is further divided into light and heavy-stroked characters. We point out two important observations from figure IV.19. First, the light-stroked characters had close to a 10% better recognition rate than the heavy-stroked characters. Second, the heavy-stroked numeric characters had 8.9% rejections while the light-stroked numeric characters had only 1.9% rejections. The overall recognition rate is given at the bottom of the figure. We observe a 86.9% recognition rate, with 6.4% misrecognitions and 6.7% rejections.

		Number of patterns	Number of patterns recognized	Number of patterns misrecognized	Number of patterns rejected
	Light-Stroked	321	91.9%	2.8%	5.3%
	Heavy-Stroked	540	82.2%	9.6%	8.2%
ALPHABETICS		861	85.8%	7.1%	7.1%
	Light-Stroked	321	94.1%	4.1%	1.9%
	Heavy-Stroked	540	83.7%	6.7%	8.9%
NUMERICS		861	88.0%	5.0%	6.6%
	Light-Stroked	642	93.0%	3.4%	3.6%
	Heavy-Stroked	1080	83.3%	8.2%	8.5%
OVERALL		1722	86.9%	6.4%	6.7%

Figure IV.19. Recognition rates. The table displays the alphabetic, numeric, and the overall recognition error rates.

Figure IV.20 gives the confusion matrices for the alphabetic and numeric characters respectively. Some important observations should be pointed out from the top matrix: (a) no "V"'s were rejected, but nine were misrecognized as "Y", (b) twelve "R"'s were rejected, eight of which written with a heavy-stroked pen, and, (c) seven "M"'s were misrecognized as "H", all of them being written with a heavy-stroked pen. Similarly, from the bottom matrix of Figure IV.20, we observe the following: (a) out of the ten "2"'s that were rejected, nine were written with a heavy-stroked pen, and, (b) the eight "3"'s that were rejected were written with a heavy-stroked pen.

ALPHABETIC

	A	B	C	E	G	H	J	K	L	M	N	P	R	S	T	V	W	X	Y	Z	Rej.	Tot.
A	40																				2	42
B		34												1							8	43
C			39				1		2												2	44
E				40	1		2															43
G		1			34				1												4	40
H	2					59					3								1			65
J							38								3						2	43
K			1			2		31			1										4	39
L									41												2	43
M						7				37											2	46
N						3		1			30		1				1				1	37
P												36	1								2	39
R							3						21								12	36
S					1									40							3	44
T															48						3	51
V																36			9			45
W							1				1					1	36					39
X						1						1						36			4	42
Y							3									1			40		3	47
Z			1						1									1		23	7	33
	A	B	C	E	G	H	J	K	L	M	N	P	R	S	T	V	W	X	Y	Z	Rej.	Tot.

NUMERIC

	0	1	2	3	4	5	6	7	8	9	Reject	Total
0	79										1	80
1		117	2								9	128
2		3	85	1			1				10	100
3			2	71	1	4			2		8	88
4		3			83						3	89
5			2			75	2	1			1	81
6							68				9	77
7		3	2	1	2			65	2	1	9	85
8					1		1		65	3		70
9		2	1					1	5	50	4	63
	0	1	2	3	4	5	6	7	8	9	Reject	Total

Figure IV.20. The alphabetic and numeric character confusion matrices. Thus, out of 43 "B"'s, 34 were recognized correctly, 1 was recognized as an "S", and 8 were rejected.

Figure IV.21 displays the recognition rate of the leftmost character of the postal code (geographic designator), and Figure IV.22 gives the confusion matrix. We give this figure because the geographic designator is considered to be the most crucial character in the postal code (the geographic designator identifies a province within Canada as Figure IV.15 illustrated). The results are essentially the same for the heavy and light-stroked characters. We note an important observation from the confusion matrix; seven "R"'s were rejected, four of them being from the same writer, that is, writer #10 from Figure IV.18.

Figure IV.23 displays the recognition rate of the leftmost three characters of the postal code (Forward Sortation Area-FSA). These, recall from Section III.1, give a sort to city level (or a specific area within a city), or a rural area. From the table, we observe that 81.3% of the light-stroked FSAs were recognized as opposed to only 66.7% for the heavy-stroked FSAs.

	Number of patterns	Number of patterns recognized	Number of patterns misrecognized	Number of patterns rejected
Light-Stroked	107	87.9%	2.8%	9.4%
Heavy-Stroked	180	86.1%	7.2%	6.7%
Total	287	86.7%	5.9%	7.3%

Figure IV.21. Recognition rate of the leftmost character of the postal code. The results are essentially the same for both light-stroked and heavy-stroked patterns except that the number of heavy-stroked codes misrecognized were 7.2% as opposed to 2.8% for the light-stroked codes.

	A	B	C	E	G	H	J	K	L	M	N	P	R	S	T	V	X	Y	Reject	Total
A	13																			13
B		5																	3	8
C			8																	8
E				17																17
G					18														3	21
H	1					35					1							1		38
J							23								1				1	25
K			1			1		18			1									21
L									13										1	14
M						1				23									1	25
N											6									6
P												7	1						1	9
R													9						7	16
S														12					2	14
T															7					7
V																16		3		19
X						1				1							10		2	14
Y						1										1		9	1	12
	A	B	C	E	G	H	J	K	L	M	N	P	R	S	T	V	X	Y	Reject	Total

Figure IV.22. Confusion matrix for the leftmost character of the postal code.

	Number of FSAs	Number of FSAs recognized	Number of FSAs misrecognized	Number of FSAs rejected
Light-Stroked	107	81.3%	5.6%	13.1%
Heavy-Stroked	180	66.7%	13.9%	19.4%
Total	287	72.1%	10.8%	17.1%

Figure IV.23. Recognition error rate of the leftmost three characters of the postal code (FSA). Observe, that, whereas the recognition error rate for the light-stroked and heavy-stroked patterns of the geographic designator are essentially similar, the FSA recognition rate is significantly affected by pen stroke thickness.

In presenting our experimental results, we have uniformly discriminated light-stroked characters from heavy-stroked ones. To conclude our discussion and to give an intuitive feeling for the types of common problems which may arise during recognition of heavy-stroked characters, we give Figure IV.24. In order to make our discussion more pertinent, we have included, for comparison's sake, some of the postal codes written by the best writer, that is, writer #1 of Figure IV.18.

We have presented our experimental results and discussed the performance of the recognition method we adopted. In the next chapter, we give our concluding remarks and propose some refinements for further experimental work on postal code segmentation and recognition.

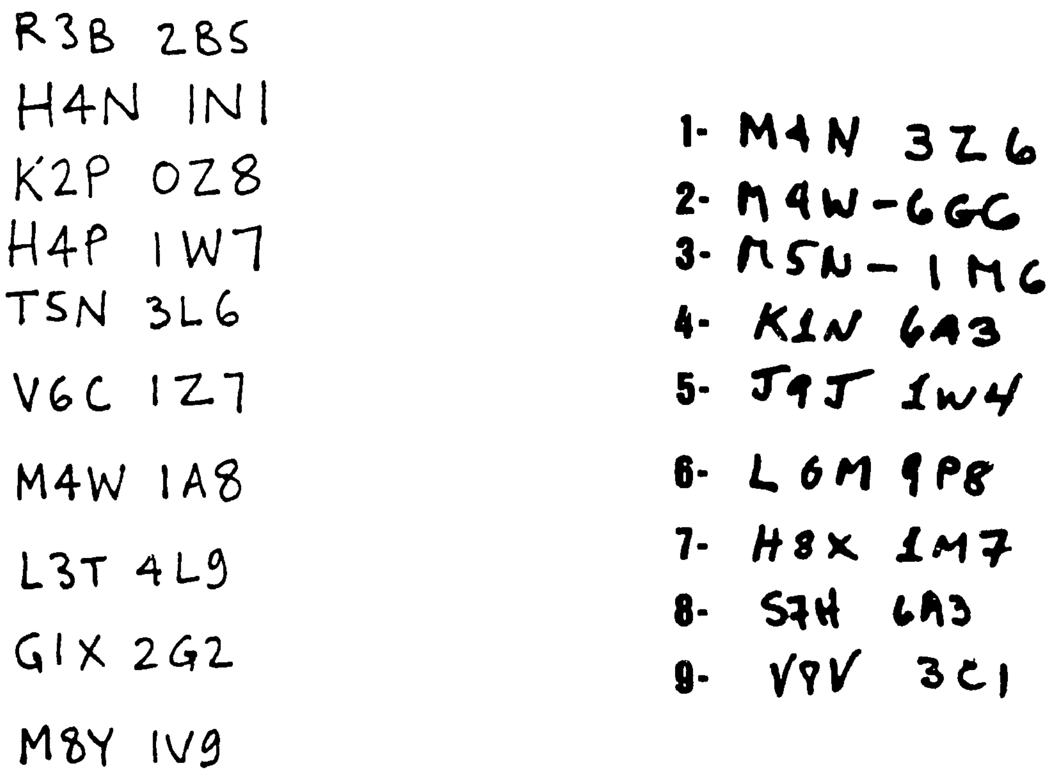


Figure IV.24. Specimens of handwritten postal codes taken from our experimental data set. The left-hand side codes were written with a light-stroked pen and were taken from the best writer, writer #1 of Figure IV.18. The right-hand side codes show some postal codes written with a heavy-stroked pen. The first code has a "4" with a filled-in hole; the second, third and fourth codes have "6"'s with their holes filled in and distorted "M"'s and "W"'s; the fifth and sixth codes each have a "9" with a filled hole; the seventh code has a distorted "3" and "M"; the eighth code has the hole in the "A" filled in; the ninth code has the bottom hole of the "8" filled in.

CHAPTER V

CONCLUDING REMARKS

V.1 Overall Assessment

We have presented an algorithm for segmenting handwritten postal codes. We tested our algorithm on a data set consisting of 300 handwritten Canadian postal codes. Moreover, to examine the sensitivity of our segmentation algorithm to the thickness of pen stroke, our data base consisted of codes written by both light-stroked and heavy-stroked pens. The experimental performance of our segmentation algorithm revealed a 93% correct segmentation rate, in spite of the codes having broken, touching, and overlapping patterns. (the light-stroked codes had a marginally higher correct segmentation rate than the heavy-stroked codes). The algorithm uses contour tracing, which had been previously used mainly for feature extraction. Thus, the information gathered during segmentation can be used again later for feature extraction. This would add efficiency to an optical character recognition system. Moreover, our algorithm is general enough to be adopted for postal codes of many other countries, for example, Argentina (numeric), Australia (numeric), Brazil (numeric), Britain (alphanumeric), France (numeric), India (numeric), Italy (numeric), Israël (numeric), Philippines (numeric), Poland (numeric), South Africa (numeric), Turkey (numeric), and the United States of America (numeric). The algorithm has been presented in a modular fashion: any module may be refined to handle specific characteristics that may be local to the postal codes of a particular country.

To test the correctness of our segmentation algorithm, we used a well-known feature extraction method, that is, characteristic loci (Glucksman,1967), and a two-level decision tree classifier to recognize the segmented postal codes. We obtained an overall 60% correct postal code recognition rate, with 30.7% rejections, and 9.3% misclassifications. Our initial intent on verifying the sensitivity of pen stroke thickness revealed important implementation considerations which will be discussed in the next section.

What clearly pervades through our experimental results is that although our segmentation method is quite insensitive to pen stroke thickness, the recognition process is definitely affected by the thickness of the pen stroke. We may give our overall observations thus: if a good writer uses a heavy-stroked pen and prints big characters, the recognition error rate is minimally affected. On the other hand, if the same or worse writer is confined to print his characters in a smaller area, then, there is a higher likelihood that, (a) characters will be touching, thus complicating segmentation, or, (b) characters such as, "A", "B", "P", "R", "4", "6", "8", and "9", will have their loops filled in, thus impeding the recognition process. However, if a good writer prints his characters with a light-stroked pen, experimental results demonstrate that the recognition rate is not affected by the size of the character.

V.2 Suggestions for Future Research

To conclude our discussion, we suggest enhancements and propose research work for future experiments. Acknowledging the fact that heavy-stroked characters slightly complicate segmentation and substantially hinder recognition, an experiment could be repeated using solely light-stroked codes. If the experiment is to be repeated using different thickness of pen strokes, we suggest the following enhancement: improve the segmentation of heavy-stroked touching patterns since they were responsible for the majority of segmentation errors (see Table III.2). Furthermore, touching patterns that are correctly segmented should be preprocessed further before feature extraction since they usually tend to contain residual noise.

We propose the following for feature extraction: given that our segmentation algorithm reveals the outer contour of each pattern, we suggest the use of a feature extraction technique such as, transformation (Fourier transforms), or outline of pattern (see Section IV.2). Using such a technique would speed up the feature extraction process (since the information is already available from segmentation) and hence make the OCR system more efficient.

Conclusively, we give two important suggestions extrapolated from our experimental results, intended at guiding Canada Post Corporation towards a national implementation of OCR automatic reading of handwritten postal codes:

- make the public aware of the sensitivity of OCR software to pen stroke thickness from handwritten postal codes.

- as a first step towards OCR automatic reading of handwritten postal codes, consider pre-sorting handwritten mail by FSA (leftmost three characters of postal code), leaving the final sort to manual workers. Referring to Figure IV.23, we observe that the FSA correct recognition rate is approximately 10% higher than the correct recognition rate of the postal code (Figure IV.17). Hence, if a piece of mail is pre-sorted to city level or rural area (that is by FSA), then the final sort could be done manually on the LDU only (that is the rightmost three characters of the postal code). As the recognition rate percentage would improve gradually by using state-of-the-art techniques, the migration towards full OCR automatic reading of postal code would minimally impede production during the transition phase.

References

[1] Atrubin, A.J., Fursa, R.S. & Malaby, D.L.(1972). Mail piece address simulation. Proceedings International Conference on Cybernetics and Society. Washington, U.S., 522-526.

[2] Belaid, A. & Haton, J.-P.(1984). A syntactic approach for handwritten mathematic formula recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-6, No.1, 105-111.

[3] Beun, M.(1973). A flexible method for automatic reading of handwritten numerals. Philips Technical Review. Vol. 33, 89-101, 130-137.

[4] Bledsoe, W.W. & Browning, I.(1959). Pattern recognition and reading by machine. Proceedings Eastern Joint Computer Conference, 225-232.

[5] Busby, J.L. & Willy, K.D.(1986). The development of the E40 letter sorting machine. International Postal Engineering Conference. London, England, 131-144.

[6] Calvert, T.W.(1970). Nonorthogonal projections for feature extraction in pattern recognition. IEEE Transactions on Computers. Vol. 19, 447-452.

[7] Casey, R.G. & Nagy, G.(1982). Recursive segmentation and classification of composite character patterns. Proceedings Sixth Conference on Pattern Recognition. Munich, Germany, Vol. 2, 1023-1026.

[8] Casey, R.G. & Jih, C.R.(1983). A processor-based OCR system. IBM Journal of Research and Development. No. 4, 386-399.

[9] Dasarathy, B.V. & Kumar, K.P.B.(1978). CHITRA: cognitive handprinted input-trained recursively analyzing system for recognition of alphanumeric characters. International Joint Computing Information Science. Vol. 7, 253-282.

[10] Dinstein, I. & Shapira, Y.(1982). Ancient hebraic handwriting identification with run-length histograms. IEEE Transactions on Systems, Man and Cybernetics. SMC-12, 405-409.

[11] Doyle, W.(1960). Recognition of sloppy, handprinted characters. Proceedings Western Joint Computer Conference. San Francisco, U.S.A., Vol. 17, 133-142.

[12] Ehric, R.W. & Kochler, K.J.(1975). Experiments in contextual recognition of cursive script. IEEE Transactions on Computers. Vol. C-24, No. 2, 182-194.

[13] Fleming, J. F. & Hemmings, R.F.(1983). A method of recognition for handwritten block capitals. Pattern Recognition Letters 1. Numbers 5,6, 457-464.

[14] Focht, L.R. & Burger, A.(1976). A numeric script recognition processor for postal zip code application. Proceedings International Conference on Cybernetics and Society. Washington, U.S., 489-492.

[15] Fu, K.S.(1978). Recent advances in syntactic pattern recognition. Proceedings Fourth International Joint Conference on Pattern Recognition. Kyoto, Japan, 13-18.

[16] Fujimoto, Y., Kadota, S., Hayashi, S., Yamamoto, M., Yajima, S. & Yasuda, M.(1976). Recognition of hand-printed characters by nonlinear elastic matching. Proceedings Third International Joint Conference on Pattern Recognition. Coronado, California, U.S., 113-118.

[17] Genechi, H., Mori, K.I., Watanabe, S. & Katsugari, S. (1968). Recognition of handwritten numeral characters for automatic letter sorting. Proceedings IEEE. Vol. 56, 1291-1301.

[18] Genechi, H., Mori, K.I., Watanabe, S. & Katsugari, S. (1970). Automatic reader-sorter for mail with handwritten or printed postal code numbers. TOSHIBA Review, 7-11.

[19] Glucksman, H.(1967). Classification of mixed font alphabetics by characteristic loci. Digest of First Annual IEEE Conference, 138-141.

[20] Güdesen, A.(1976). Quantitative analysis of preprocessing techniques for the recognition of hand-printed characters. Pattern Recognition. Vol. 8, 219-227.

[21] Hannon, L.D.(1972). Automatic recognition of print and script. IEEE Proceedings. Vol. 60, No. 10, 1165-1176.

[22] Hennis, R.B.(1968). The IBM 1975 optical page reader part 1: System design. IBM Journal of Research and Development, 346-353.

[23] Hoffman, R.L. & McCullough, J.W.(1971). Segmentation methods for recognition of machine printed characters. IBM Journal of Research and Development. Vol. 15, No.1, 153-165.

[24] Hussain, A.B.S., Toussaint, G.T. & Donaldson, R.W.(1972). Results obtained using a simple character recognition procedure on Munson's handprinted data. IEEE Transactions on Computers. Vol. 21, 201-205.

[25] Iwata, K., Yoshida, M. & Tokunaga, Y.(1978). High speed OCR for handprinted characters. Proceedings Fourth International Joint Conference on Pattern Recognition. Kyoto, Japan, 645-649.

[26] Jih, C.R.(1980). Segmentation method for fixed-pitch, machine-printed documents. IBM Technical Disclosure Bulletin. No. 23, 1194, August, 1980.

[27] Jih, C.R.(1981). Optical character recognition using baseline information. U.S. Patent no. 4, 251, 799. February 17, 1981.

[28] Kahan, S., Pavlidis, T. & Baird, H.S.(1987). On the recognition of printed characters of any font and size. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-9, No. 2, 274-288.

[29] Kegel, A.G., Giles, J.K. & Ruder, A.H.(1976). Observations on selected application of optical character readers for constrained, numeric handprint. Proceedings International Conference on Cybernetics and Society. Washington, U.S., 471-475.

[30] Knoll, A.L.(1969). Experiments with characteristic loci for recognition of hand-printed characters. IEEE Transactions on Computers. Vol. 18, 366-372.

[31] Kori, R. & Lin, W.C.(1976). An outline mini-computer based system for reading printed text aloud. Proceedings International Conference of Cybernetics and Society. Washington, U.S., 509-513.

[32] Kovalevsky, V.A.(1978). Recent advances in statistical pattern recognition. Proceedings Fourth International Joint Conference on Pattern Recognition. Kyoto, Japan, 2-12.

[33] Kwan, C.C., Pang, L. & Suen, C.Y.(1979). A comparative study of some recognition algorithms in character recognition. Proceedings International Conference on Cybernetics and Society. Denver, Colorado, U.S., 530-535.

[34] Kwon, S.K. & Lai, D.C.(1976). Recognition experiments with handprinted numerals. Proceedings Joint Workshop on Pattern Recognition and Artificial Intelligence, 74-83.

[35] Lai, M. & Suen, C.Y.(1981). Automatic recognition of characters by Fourier descriptors and line encoders. Pattern Recognition. Vol. 14, Nos 1-6, 383-393.

[36] Loy, W.W. & Landau, I.D.(1982). An on-line procedure for recognition of handprinted alphanumeric characters. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-4, No. 4, 422-427.

[37] Mantas, J. & Heaton, A.G.(1983). Handwritten character recognition by parallel labelling and shape analysis. Pattern Recognition Letters I, 465-468.

[38] Moss, R.H.(1983). On Line Recognition (OLREC): A novel approach to visual pattern recognition. Pattern Recognition. Vol. 16, No. 6, 535-550.

[39] Mori, S., Yamamoto, K. & Yasuda, M.(1984). Research on machine recognition of handprinted characters. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-6, 386-405.

[40] Nadler, M.(1980). Structural codes for omnifont and hand-printed characters. _Proceedings Fifth International Conference on Pattern Recognition_. Miami Beach, U.S., 848-852.

[41] Ni, G., Ding, J., Gao, Z. & Liu, J.(1980). A structural method for handprinted alphanumeric and other symbols. _Proceedings Fifth International Conference on Pattern Recognition_. Miami Beach, U.S., 726-728.

[42] Nishimura, Y., Nogushi, Y. & Toyoda, J.(1983). Segmentation of machine-printed text. _Proceedings International Conference on Text Processing with a Large Character Set_, 79-84.

[43] Ott, R.(1974). On feature selection by means of principal axis transform and nonlinear classification. _Proceedings Second International Joint Conference on Pattern Recognition_. Copenhagen, Denmark, 220-222.

[44] Pavlidis, T.(1976). Syntactic feature extraction for shape recognition. Proceedings Third International Joint Conference on Pattern Recognition. Coronado, California, U.S., 95-99.

[45] Pavlidis, T. & Ali, F.(1979). A hierarchical syntactic shape analyzer. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-1, No. 1, 2-9.

[46] Pavlidis, T.(1983). Effects of distortions on the recognition rate of a structural OCR system. IEEE 1983 Computer Vision and Pattern Recognition, 303-309.

[47] Persoon, E. & Fu, K.S.(1975). Sequential classification of strings generated by SCFG's. International Journal of Computer and Information Sciences. Vol. 4.

[48] Persoon, E. & Fu, K.S.(1977). Shape discrimination using Fourier descriptors. IEEE Transactions on Systems, Man and Cybernetics. Vol. 7, 170-179.

[49] Pervez, A. & Suen, C.Y.(1982). Segmentation of unconstrained handwritten numeric postal zip codes. Proceedings Sixth International Conference on Pattern Recognition. Munich, Germany, Vol. 1, 545-547.

[50] Porod, W. & Ferry, D.K.(1985). Pattern recognition in highly integrated circuits. Pattern Recognition. Vol. 18, No. 2, 179-189.

[51] Rabinow, J.(1981). Postal automation: It can be done. IEEE Spectrum, 50-57.

[52] Rasch, P.J.(1972). Contextual recognition of mail pieces address information. Proceedings International Conference on Cybernetics and Society. Washington, U.S., 615-621.

[53] Sato, K., Isshiki, I., Ohoka, A. & Yoshida, K.(1983). Hand-scan OCR with a one-dimensional image sensor. Pattern Recognition. Vol. 16, No. 5, 459-467.

[54] Schurmann, J.(1976). Multifont word recognition system with application to postal address reading. Proceedings Third International Joint Conference on Pattern Recognition. Coronado, California, U.S., 658-662.

[55] Shridlar, M. & Badreldin, A.(1984). High accuracy character recognition algorithm using Fourier and topological descriptors. Pattern Recognition. Vol. 17, No. 5, 515-524.

[56] Spanjersberg, A.A.(1974). Combinations of different systems for the recognition of handwritten digits. Proceedings Second International Joint Conference on Pattern Recognition. Copenhagen, Denmark, 208-209.

[57] Stentiford, F.W.M.(1985). Automatic feature design for optical character recognition using an evolutionary search procedure. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-7, No. 3, 349-355.

[58] Suen, C.Y. & Shillman, R.J.(1977). Low error rate optical character recognition of unconstrained handprinted letters based on a model of human perception. _IEEE Transactions on Systems, Man and Cybernetics_. Vol. 7, 491-495.

[59] Suen, C.Y.(1982A). Distinctive features in automatic recognition of hand-printed characters. _Signal Processing_. Vol. 4, 193-207.

[60] Suen, C.Y.(1982B). The role of multi-directional loci and clustering in reliable recognition of characters. _Proceedings Sixth International Conference on Pattern Recognition_. Munich, Germany, Vol. 2, 1020-1022.

[61] Tou, J.T. & Gonzalez, R.C.(1972). Recognition of handwritten characters by topological feature extraction and multilevel categorization. _IEEE Transactions on Computers_. Vol. 21, 776-785.

[62] Toussaint, G.T. & Donaldson, R.W.(1970). Algorithms for recognizing contour-traced handprinted characters. _IEEE Transactions on Computers_. Vol. 19, 541-546.

[63] Toussaint, G.T.(1977). The use of context in pattern recognition. Proceedings International Conference on Pattern Recognition & Image Processing, IEEE. Troy, N.Y., U.S.A., June 6-8.

[64] Toyoda, N. & Nichimura, Y.(1982). Study of extracting Japanese newspaper article. Proceedings Sixth International Conference on Pattern Recognition. Munich, Germany, 1113-1115.

[65] Tucker, N.D. & Evans, F.C.(1974). A two-step strategy for character recognition using geometrical moments. Proceedings Second International Joint Conference on Pattern Recognition. Copenhagen, Denmark, 223-225.

[66] Unger, S.H.(1959). Pattern detection and recognition. Proceedings Institute of Radio Engineers. Vol. 47, 1737-1752.

[67] Valdez, M.E. & Ozkul, T.(1985). A new procedure for recognizing handwritten and typewritten alphabetical characters. Proceedings of the Conference on IEEE, 233-237.

[68] Wang, P.S.-P.(1982). A new character recognition scheme with lower ambiguity and higher recognizability. Proceedings Sixth International Conference on Pattern Recognition. Munich, Germany, 37-39.

[69] Yamada, H. & Mori, S.(1978). Line-wise parallel operations and their application to handprint recognition. Proceedings Fourth International Conference on Pattern Recognition. Kyoto, Japan, 789-793.

[70] Yamamoto, K. & Mori, S.(1980). Recognition of handprinted characters by an outermost point method. Pattern Recognition. Vol. 12, 229-236.

[71] Yoshida, K.(1982). Online handwritten character recognition for a personal computer system. IEEE Transactions on Consumer Electronics. Vol. CE-28, No.3, 202-209.