Detection of Courtesy Amount Block on Bank Checks

Arun Agarwal[*], Karim Hussein[*], Amar Gupta[*]
P. S. P. Wang[†]

Abstract

This paper presents a multi-staged technique for locating the courtesy amount block on bank checks. In the case of a check processing system, many of the proposed methods are not acceptable, due to the the presence of many fonts and text sizes, as well as the short length of many text strings. This paper will describe particular methods chosen to implement a Courtesy Amount Block Locator (CABL). First, the connected components in the image are identified. Next, strings are constructed on the basis of proximity and horizontal alignment of characters. Finally a set of rules and heuristics are applied to these strings to choose the correct one. The chosen string is only reported if it passes a verification test, which includes an attempt to recognize the currency sign.

Keywords: *check analysis and processing, block detection, courtesy amount recognition, image processing, heuristics rules, segmentation*

1 Introduction

Trillions of dollars change hands each year in the form of handwritten or machine printed bank checks. Currently, several steps involved in processing

[*]Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139. *agupta@mit.edu*

[†]College of Computer Science, Northeastern University, Boston, MA 02115, *pwang@ccs.neu.edu*

the check are done by computers. The number of the account from which a check has been drawn and the bank code are encoded in magnetic ink at the bottom of a check to facilitate this process. One step in the process that is, for the most part, still done by people is the identification of the dollar (or other unit of currency) amount for which the check has been written. Bank employees read the amount, usually from the courtesy dollar amount box, enter these data into a computer system, which then prints the amount in magnetic ink at the bottom of the check. From this point on, processing can be automated.

In recent years, Optical Character Recognition (OCR) technology has evolved to a level that would allow automation of the process of converting the written or printed image of the numerals in the courtesy dollar amount box to a machine understandable form [20] [7] [16]. The implementation of OCR systems tailored to the job of recognizing the courtesy amount on checks would greatly facilitate the design of a completely automated check processing system.

One reason why the use of such systems has not become widespread in this country is that checks do not conform to a single standard. Checks come in a variety of sizes, and the courtesy amount is not located in the same place on all checks. Some courtesy amount recognition systems under development, including the one being developed by MIT's OCR group [15, 25, 22, 21], will achieve reasonably accurate results when the input presented is either the image of the courtesy amount only, or an image of the entire check along with explicit information as to the exact location of the courtesy amount block.

The purpose of this paper is to describe a Courtesy Amount Block Locator (CABL) subsystem for use in a robust, completely automated amount recognition system. The CABL has been designed to construct a high level representation of the structure of a check image, and to locate the courtesy amount. Once these tasks are accomplished, the rest of the system can segment the amount into individual digits and symbols, and recognize the digits and determine the amount for which the check had been written. Actually, the CABL not only performs the task of locating the amount, but also simplifies the preliminary phase of the segmentation task, that is, identification of the connected components of the amount [25].

The processing of bank checks is one application of the broader technology of document understanding. Section 2 describes methods currently used in

Document Understanding Systems (DUS's). Section 3 provides a description of the CABL subsystem that has been designed for use in MIT's OCR group's check processing system. Section 4 discusses some of heuristics that can be applied to the specific problem of locating the courtesy amount block within a check image. Section 5 outlines the performance of the CABL on actual check data, in terms of both speed and accuracy.

2 Document Analysis

2.1 Background

There are a variety of document forms that may constitute the input to a Document Understanding System (DUS). The DUS may be designed to utilize certain known constraints on the document format. Many forms restrict the writer to using black pens. Often there are guidelines to help align written text with the margins of the page, and to fix the location where a particular piece of information is to be recorded. Sometimes there are boxes to constrain the length of a piece of data, or the height of individual characters. More constraining than this, there may be individual boxes for characters, with instructions to print one character per box. A guide to forming letters in print may appear, restricting the general shape of a letter or number, but not removing variation due to a particular person's penmanship. If the DUS were designed to understand the information a writer is trying to convey in a totally unconstrained document, there would be no restriction on the information that is presented in the document. The system would have to account for many different stylus types and colors, variations in size of print, texture of the surface, location of important data, as well as other variables. For more details of recent development in PR and OCR in document analysis, please refer to [23, 14, 2, 3, 24, 17, 28, 31, 29, 32, 30].

Most forms that are subject to machine analysis fall between these two extremes. Before such documents can be analyzed and understood, several preprocessing steps, namely binarization [27], noise reduction [13] and form line removal [5] stages have to be applied to reduce the amount of redundant information.

2.2 Extraction of Data

Since the purpose of this paper is to develop a system that recognizes the courtesy amount on checks, attention will be focused on existing methods that extract just text. Two approaches are now discussed that are available to locate bits of relevant text in an image.

2.2.1 Bottom-Up Segmentation

In a bottom-up method [26, 27, 12], characters are first located. These characters are then grouped together to form words and lines. Bottom up approaches, however, may be more robust when faced with noise, and document skew [11]. The character location process and the character grouping process are discussed in the following sub-sections.

Character Location There are two typical methods for locating characters: a stroke method and a connected component method. The rest of this sub-section describes these two methods in more details.

Stroke Method: An algorithm suggested by Srihari [26] first computes run-lengths. These runs represent a horizontal slice of a vertical or slanted handwritten stroke (assuming that most of the data on the page is handwritten or printed text). If one now examines the length of all of the horizontal runs which have lengths between 3 and 20 pixels, one can determine the average thickness of a handwritten stroke on that particular document. Next, sets of 3 or more runs that approximate this length, and are vertically connected to each other, are grouped together to form complete strokes. These primitive strokes are then grown by adding adjacent runs that satisfy a more relaxed criterion. The length of these additional runs need not be as close to the average stroke width as the runs that make up the original backbone of the stroke. This process is repeated until no stroke can be grown any further. Strokes are then grouped together to form characters, words and blocks on the basis of proximity and size.

Connected Component Method: Other common bottom-up segmenters start by identifying the connected components in the image [27, 12]. This can be accomplished using a depth first search of black pixels, using

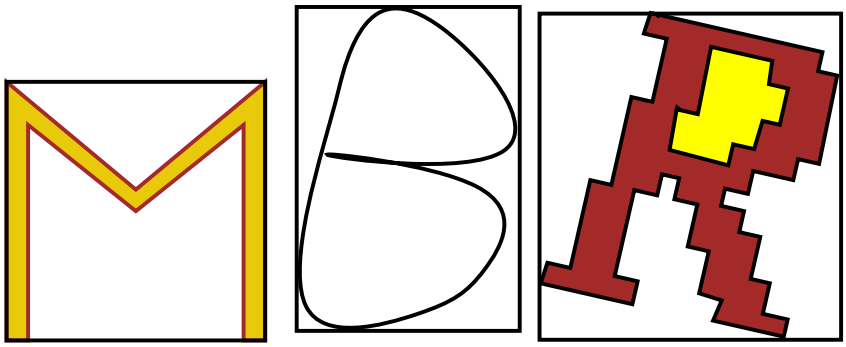

Figure 1: Minimum Bounding Rectangles (MBR's) for Several Characters

8-connectivity or 4-connectivity as an adjacency criterion. These connected components can be represented by their minimum bounding rectangles (MBR's). An MBR is the smallest rectangle whose sides are parallel to the boundaries of the image that encloses the component (See Figure 1).

Depending on a particular application, some components may be rejected based on their size or shape. For instance, a component with an aspect ratio (height / width) of .02 cannot possibly be a character. In addition, if most of the components in the image have an area of about 500 pixels, a component that has an area of 10,000 pixels can be discarded as a graphic or other non-character piece of data. Fletcher and Kasturi [12] accept or reject each component as a character on the basis of its size, black pixel density, aspect ratio, area, and position in the image.

Character Grouping:

The next step is the grouping of characters to form words and lines. One simple way is to just group together characters that are adjacent to each other horizontally, and are of similar shape and size [27]. If there is skew in the image, however, or if the document is sloppily handwritten so that neighboring characters are not necessarily directly horizontally adjacent, a different approach is necessary.

The Hough transform [10, 13] can be applied to the centroids of all of the

characters. Through this method, one can determine which characters in the image are nearly collinear. Characters which are closer than some threshold distance to each other, and are also nearly collinear can be assumed to belong to the same string, or line of text [12].

2.2.2 Top-down Approaches

In a top-down method, the system tries to determine the overall structure of a page, then breaks this down into regions, lines, words, and possibly characters. Top down approaches tend to be less computationally intensive, since they deal mainly with image blocks that are larger, but fewer in number.

Two techniques that are useful in this process are smearing and smoothing. Downton and Leedham [8, 9] have developed a system to locate the address block on a piece of mail. As a first step, the image is smeared horizontally. The effect here is that adjacent characters and words become smeared into one another. Now, when connected components are extracted, they will be few and large. Fisher et al. [11] suggest choosing the smearing factor dynamically, based on the average interline and intercharacter spacings in the image, to be sure that image items are grouped together in the optimal manner.

Smoothing is similar to smearing. If the resolution of the image is lowered, and the values assigned to the new larger pixels are determined as a function of the values of the high resolution pixels they encompass, the connected components in the low resolution image will represent the large elements of the image, such as words or blocks.

Many of these common document understanding methods discussed above assume that the input document consists mostly of dense lines of text, in a small number of type sizes. Since a check does not fit this description due to the presence of many fonts and text sizes, as well as the short length of many text strings. Therefore, modification of these methods was necessary. The next section will describe particular methods chosen to implement a Courtesy Amount Block Locator (CABL). In actuality, the CABL is a small part of a much larger system designed to determine the amount for which a check has been written, given only the image of the check as input.

3 A Courtesy Amount Block Locator

The job of locating the courtesy amount block was divided into several steps, designed to systematically decrease the amount of information required to represent a check, so that computation becomes manageable. A check is initially represented by its bitmap. If the resolution of scanning is 300 dots per inch (DPI), a typical check involves 1.5 million pieces of information. A set of connected components in the image is generated, so that there are typically only about 300 chunks to deal with. Finally, the components are grouped into strings, of which there may be about 30. Once the check is characterized by a manageable amount of data (the list of about 30 strings), the heuristics described in Section 4 can be applied.

3.1 Extraction of Components

Assuming good thresholding, most of the black pixels in a check image will correspond to text characters. Text characters will, in general, be individual connected components, islands of black pixels in a sea of white. Therefore, identifying connected components in the image is a good way to start the process of developing a high level representation of the information on the check.

The method chosen to extract connected components from the image was a depth first search. A Depth First Search (DFS) for blocks of black pixels can be computationally intensive, so an intermediate representation of the image was used. The bitmap was converted to a set of horizontal runs of black pixels, before the DFS was applied. Section 2.2.1 described the use of run length representation for a slightly different purpose by Srihari. This is a common way of reducing the amount of information used to represent the image [18]. Since black pixels on a check are relatively sparse, there might only be 10,000 runs, as opposed to the 1.5 million pixel intensities. Moreover, as DFS is linear in the number of pieces of information through which to search, computational complexity is decreased markedly through the use of this representation.

The DFS implemented is a modification of the generic Depth First Search algorithm described in [6, p. 478]. The pseudocode for implementing the DFS is shown below.

```
DFS(runlist)
{
  ListOfComponents = NULL;
  for (each run R in runlist)
    R->status = NOT_SEEN;
  for (each run R in runlist)
    if (R->status == NOT_SEEN)
      {
Component = EmptyComponent();
Visit(R, Component, runlist);
/* Sets Component to be the connected component containing R */
AddToList(ListOfComponents, Component);
      }
}

Visit(R, Component, runlist)
{
  R->status = SEEN;
  AddToComponent(Component, R);
  CurrentRow = R->row;
  for (each run R1 in runlist such that
       R1->row == CurrentRow + 1 || R1->row == CurrentRow - 1)
    /* all runs in the row above R, and the row below R */
    if (R1->status == NOT_SEEN && Adjacent(R, R1))
      {
AddToComponent(Component, R1);
Visit(R1, Component, runlist);
      }
}

Adjacent(R1, R2)
{
  startR1 = R1->left;
  startR2 = R2->left;
  finishR1 = R1->right;
  finishR2 = R2->right;
  if (finishR1 >= startR2 - 1 && startR1 <= finishR2 + 1)
```

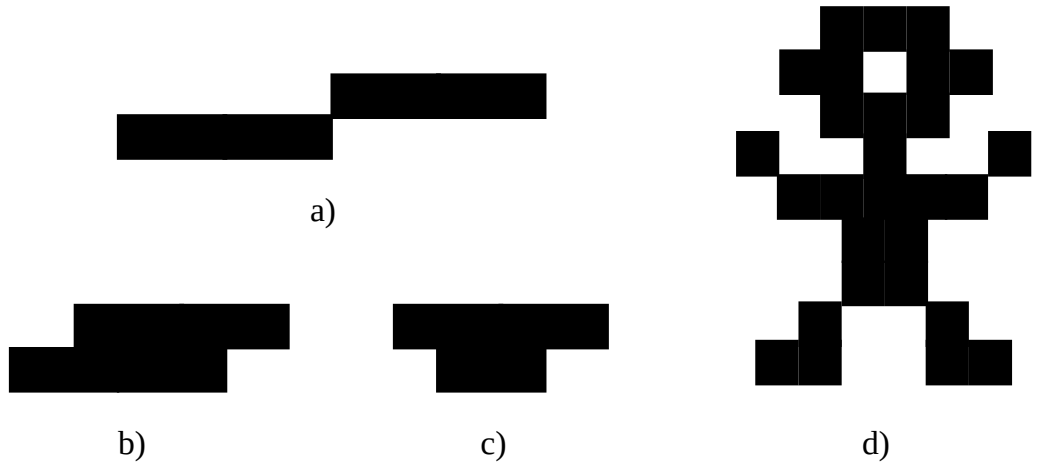

Figure 2: a,b,c) Examples of adjacent runs. d) One connected component.

```
    /* The +1 and -1 implements the check for 8-connectivity rather than
       4-connectivity */
    return (TRUE);
  else
    return(FALSE);
}
```

The horizontal rows above and below the current run are scanned for runs that are adjacent to the current run, and have not been seen yet. The adjacency criterion for two runs is described pictorially in Figure 2. A DFS through the list of runs with these criteria is equivalent to a DFS through the set of original pixels, using 8-connectivity as the adjacency criterion.

The result of this computation is a set of connected components in the check image. Most of these components correspond to individual characters, either handwritten or machine printed. Some will actually be pieces of a character that were somehow not connected to the rest of the character, others will be comprised of several characters that touch each other, and still others will consist of graphics, lines, or other non-text items. It is also possible that some characters may be grouped together with blacks lines

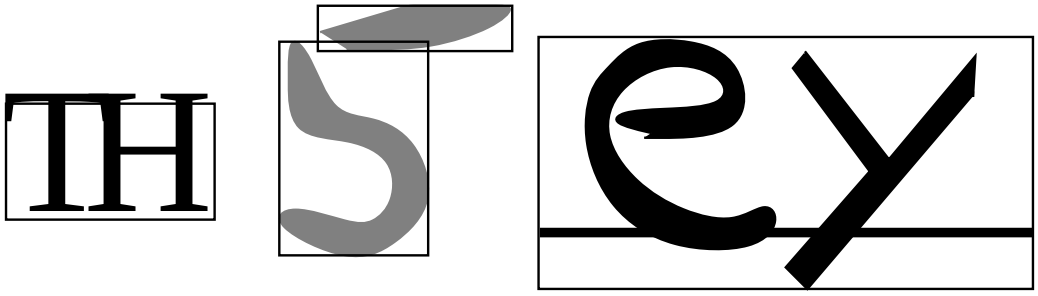

Figure 3: Non-character connected components

which they touch. Figure 3 depicts connected components corresponding to different image elements.

An attempt is made to weed out non-text components, by eliminating components with an aspect ratio (width/height) larger than 15. This should exclude the long lines on a check from consideration. This pruning value was not made too small, in order to reduce the likelihood of incorrectly removing any piece of the courtesy amount from consideration. If any digits touch a line or box around the amount, the aspect ratio of the associated component may be quite high. However, it is unlikely to be greater than 15.

It may seem as though components whose aspect ratios are too *small* should also be eliminated. This is not done, for fear of removing the digit '1' from the courtesy amount. We also do not try to remove very small components which could represent noise in the input, for fear of eliminating a decimal point from the courtesy amount, the presense of which may later be a strong indication that a particular string is the courtesy amount.

3.2 Generation of Strings

Many of the character grouping methods described in Section 2.2.1 may not be well suited to the check understanding problem. On a document such as a page of text, there are usually just one or two fonts or type sizes. There are also many lines of text, words and characters. Therefore one can use the

average intercharacter gap and interword gap to classify characters as part of distinct strings. On a check, there may be ten or more fonts, in many different sizes. In addition, handwriting size may vary depending on the amount of space given for a particular piece of handwritten data. For these reasons, the intercharacter gap and interword gaps may not be well defined entities. This makes character grouping a difficult endeavor.

Lam et al. [19] recognized a similar problem in the domain of automated reading of newspaper text. Like check images, newspaper pages contain many different type sizes. Lam et al. take a local average of the intercharacter and interword gaps in different areas of the page, and use these data to set dynamic proximity thresholds for character grouping. Even this method is not well suited to the check problem. The check understanding problem has the added difficulties that there may be some fonts which are represented only by three or four characters, and that text is sparse, as compared to text on a newspaper page.

The use of the Hough transform is suspect as well, due to the fact that many of the strings on a check are just two or three characters long. *Any* two characters will be collinear, not just the ones that together comprise a string. Moreover, a handwritten string of just three or four characters, such as the components of the date, or the courtesy amount, may not exhibit enough collinearity to be selected as part of the same string by a Hough transform based method. Mathematical methods like this are better suited to problems that involve many data points, such as characterizing a fuzzy edge, filling in the gaps in a broken line [13], or grouping together many characters of the same size and font that make up an entire line of text on a page.

The available approaches for grouping characters into strings make decisions on the bases of horizontal alignment of characters and proximity of adjacent characters. They differ only in their methods of determining these characteristics. The approach chosen for this system takes advantage of several pieces of information. First, it is not necessarily important that individual words are discovered. If a person's name appears on a check, it does not matter much whether first, middle and last names are grouped together as one string or not. The only important thing is that the digits and symbols in the courtesy amount be grouped together into one string, and that the string does not contain any characters that are *not* part of the courtesy amount.

For these reasons, the following methods were chosen. As a measure of horizontal collinearity of two characters, that is, whether or not they are

part of the same line of text, the only requirement is that the centroid of one of them horizontally projects onto some part of the other. An alternative test that was considered was, whether any part of one character horizontally projects onto any part of the other. This approach was not chosen due to the high likelihood that it will erroneously group together two strings that do not belong together.

The other variable that needed to be measured was proximity. As mentioned above, in order to be considered part of the same string, the proximity that two characters must have to each other depends on the font, and type size of the characters. Since there are so many different sizes and fonts on a check, a global proximity threshold was not used. Instead, a proximity threshold that depended on the size of the characters involved was considered. Characters in smaller typefaces need to be closer together in order to be considered part of the same string, as compared to characters in a larger font.

It was determined that the height of a character was a more accurate representation of its size than its width, for several reasons: first, certain letters or numbers, e.g. *i*, *l* and *1* have widths that do not accurately reflect the font size. Second, it is possible that some of the connected components are not really characters, but rather two or more characters that overlap or touch each other. These components will have a height, but not a width that accurately reflect the size of the font. As a result, a possible proximity criterion would be that a connected component must be within a distance equal to its height from its closest neighbor, to be considered part of the same string as that of its neighbor.

At first glance, this appears to be an appropriate proximity threshold; however, a problem exists when a period or decimal point is encountered. These components have very small height, so it is unlikely that there will be any neighboring characters within this distance from it. Given the high likelihood that a decimal point occurs within the courtesy amount, and the importance of correctly grouping the connected components that comprise the courtesy amount, this problem was treated directly.

The approach that was finally chosen for computing a proximity threshold is outlined below.

```
Threshold(Component, ComponentList)
```

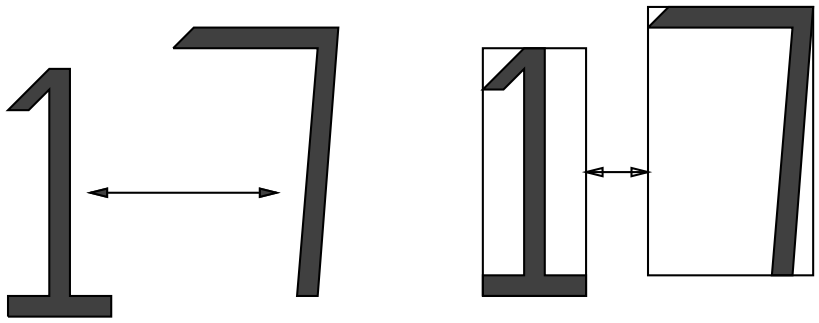

Figure 4: a) Distance between digits. b) Distance between MBR's

```
{
  Avg = Average height of all connected components in ComponentList;
  answer = 2/3 * Avg + 1/3 * Height(Component);
  return(answer);
}
```

A global proximity threshold is not used, rather, a threshold is computed for each character, as the weighted average of the character's height, and the average height of all of the characters in the image. A character C is put into a string with its neighbor if the neighboring character is horizontally aligned with it, and the distance between them is less than the proximity threshold associated with C. Therefore, characters in a larger font can be farther apart than characters in a smaller font, and still be grouped together.

Once a definition of the proximity threshold was achieved, the method of *measuring* distance between two characters was considered. It was decided that proximity would mean horizontal distance between the MBR's of the characters, as opposed to the minimum horizontal distance between pixels actually on the digits. The reason for this is shown in Figure 4. In the number *17*, if the crown of the 7 is higher than that top of the 1, the minimum horizontal distance between pixels on the digits will be much larger than the real intercharacter gap of the typeface.

The proximity and horizontal collinearity criteria described above were implemented as follows. An image was created containing only rectangular boxes representing the MBR's of the original image (see Figures 6 and 5). This image will be referred to as the highlight image. Before the highlight image is created, connected components that have a width to height ratio greater than 15 are discarded. In order to locate the strings in the image according to the horizontal alignment and proximity criteria outlined above, the following smearing algorithm was applied.

```
Smear(HighlightImage, ComponentList)
{
  for (each component C in ComponentList)
    {
      Thresh = Threshold(C, ComponentList);
      VertCenter = (C->bottom + C->top) /2;

      /* Search right */
      for (X = C->right + 1 to C->right + Thresh)
if(PixelIsSet(HighlightImage, X, VertCenter))
  {
    /* then there is another MBR aligned and close enough.
       Draw a line connected the two MBR's */
    AddRun(Image, (right+1, VertCenter) (right + Thresh, VertCenter));
  }

      /* Search left */
      for (X = C->left - 1 to C->left - Thresh)
if(PixelIsSet(HighlightImage, X, VertCenter))
  {
    AddRun(Image, (left - Thresh, VertCenter) (left - 1, VertCenter));
  }
    }
}
```

Horizontal lines were added to the highlight image connecting MBR's of characters that should be grouped into the same string. (See Figure 7). This process is reminiscent of a smearing process such as the ones described

in section 2.2.2, in that its object is to create larger connected components which represent entire strings in the image.

For each connected component in the smeared highlight image, a string is constructed by grouping connected components from the original image together, if their MBR's are contained within the connected component.

Before the strings are built up, any connected components in the smeared highlight image that have too small of a height to be anything but noise are removed. We no longer have the danger of inadvertently throwing away periods or decimal points, since those meaningful marks must occur within strings, and will be part of a connected component that is much larger.

Once the grouping process is complete, the check is represented by a list of strings. The grouping of characters into strings can be seen in Figure 8.

4 Understanding Image of Bank Check

Before implementing a system to recognize the courtesy amount block on a bank check, one must decide what assumptions can be made about the input, for what problems one must account in the process, and around what constraints one must work. Unfortunately, the format of checks is not so strictly regulated as to make the amount recognition process, or even the job of locating the courtesy amount block, trivial. Most checks fall within certain size restrictions, although there is no specific size that a check must be. In the United States, there are at least three accepted sizes for bank checks; however if one is considering checks from around the world, many other sizes are possible. This makes it impossible to just look at one particular location in the input image to obtain the courtesy amount block.

In the United States, a check is valid only if it contains the date, the amount of the check in words and numerals, a signature, and an indication as to whom the check has been written. Most checks also contain the account number of the writer, machine printed in magnetic ink by the bank who issued the checks. In other nations, these constraints may not be valid, or alternatively, may be too weak. In France, for example, the amount for which the check is being drawn must be written in numerals in two separate places. Many countries require that the courtesy amount block appear on the right half of the check. There are exceptions to this rule as well.

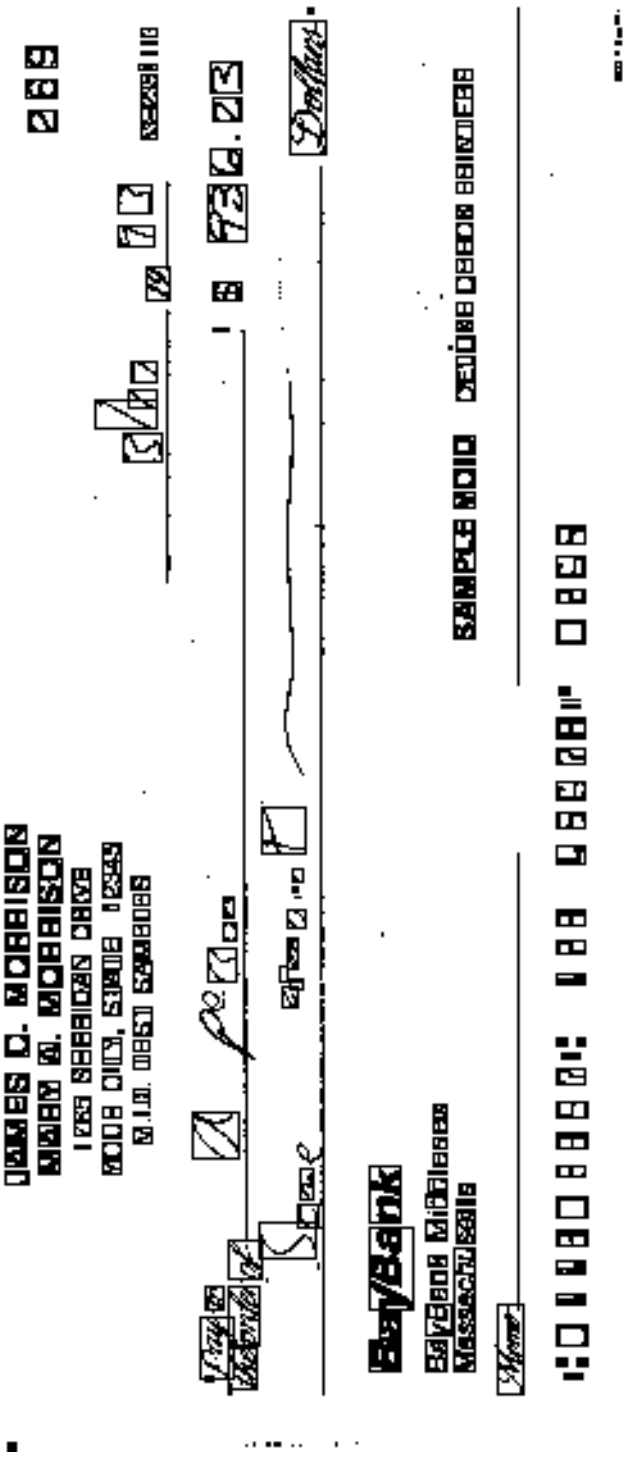

Figure 5: MBR's of Connected Components Overlaid on Original Image

Figure 6: The Highlight Image

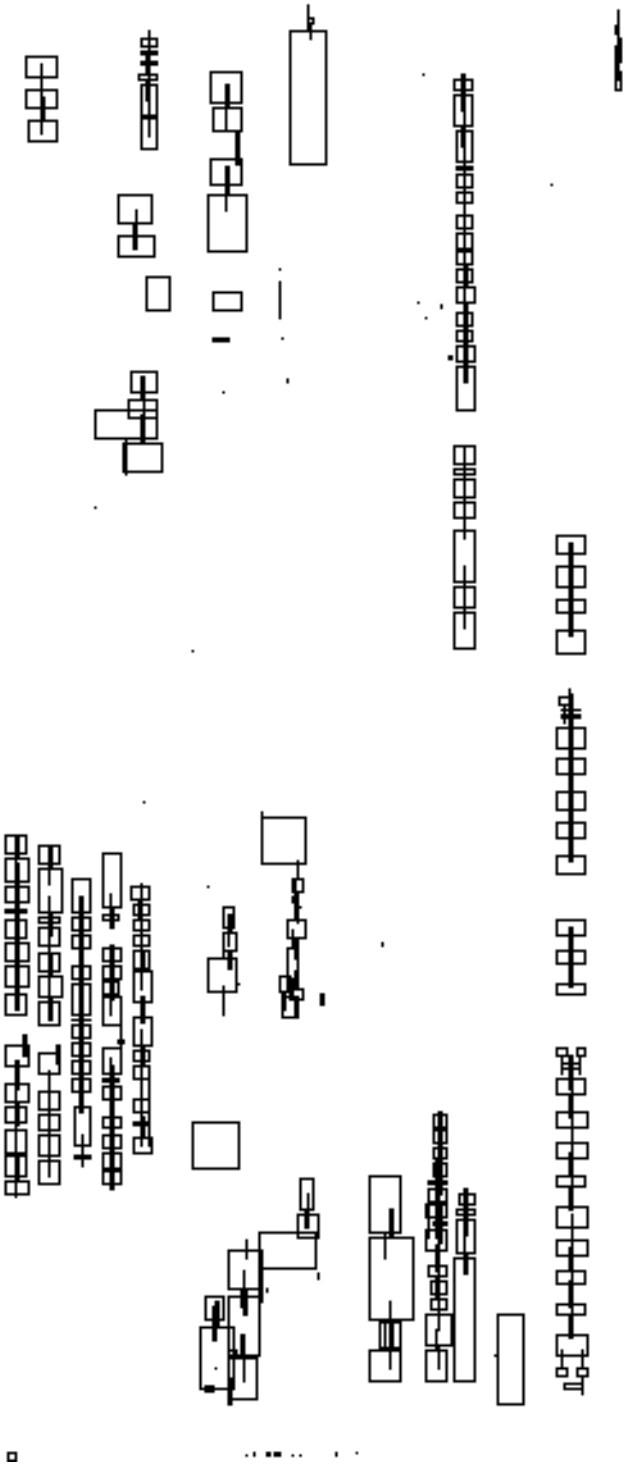

Figure 7: The 'Smeared' Highlight Image

Figure 8: Boxes around Strings of Components

4.1 Heuristic Rules

In order to simplify the implementation of a courtesy amount block recognition system, as well as to boost the accuracy of the system, it is necessary to provide a context which will guide the system's search. This context will take the form of heuristic rules.

The heuristics have been divided into two categories. The first set is for distinguishing between handwritten and machine printed text. The second set is for identifying the courtesy amount block, irrespective of whether the amount has been machine printed or handwritten. Problems foreseen in using each heuristic are also described.

4.1.1 Locating Handwritten Strings

1. *If adjacent connected components overlap, then the block is handwritten*

   Machine printed text consists of characters separated by vertical lines of white pixels. Therefore, the MBR's surrounding adjacent characters will be non-overlapping. In unconstrained handwritten numerical strings, however, the right end of a character may appear in the same vertical column as the left end of the next character (See Figure 9). In this case, the MBR's surrounding the two characters will overlap (Figure 10). This overlap is a strong indication that a particular string is handwritten.

   An exception to this occurs when there is a large skew in the input. Then, MBR's of adjacent machine printed characters may overlap (See Figure 10). To prevent this, skew correction [4] can be done prior to the application of this rule.

2. *There is less regularity in handwritten text than in machine printed text*

   This regularity is manifested in many ways. The bottommost points of all of the characters in a machine printed string will lie on a straight line. Methods for determining whether there is enough regularity in a string to say that it is indeed machine printed include the use of the Hough transform [12] and histograms [27]. Some measure of regularity may also be used with respect to the width of characters, the height of characters, and the average pixel density.

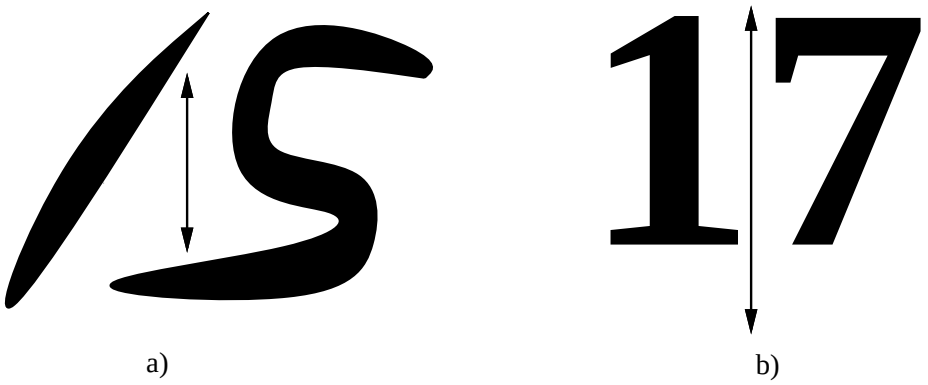

Figure 9: a) Vertical overlap in handwritten characters. b) A vertical line of white space between printed characters.

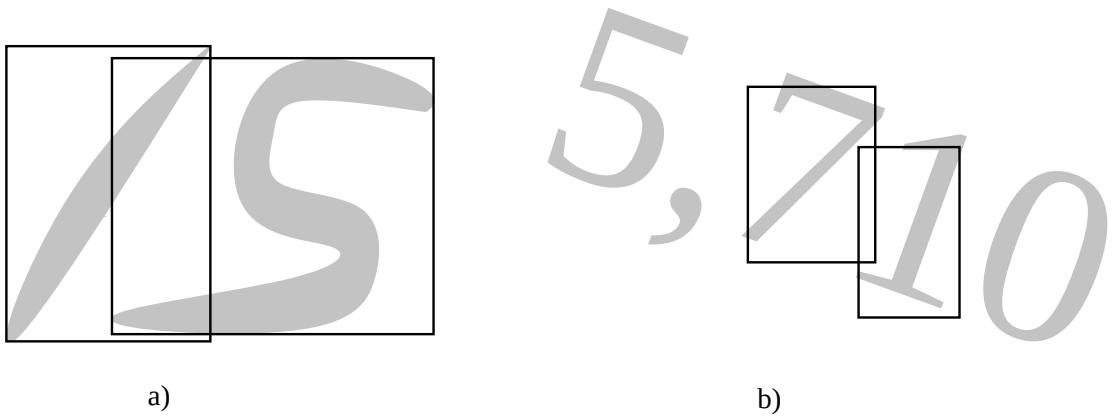

Figure 10: a) Intersecting MBR's. b) Intersecting MBR's of printed characters with large skew.

3. *Handwritten text is in a different color than machine printed text*

   Any check contains material preprinted on it by machine. This includes the guidelines on which to write, the number of the check, the century indicator on the date line, and the bank account information at the bottom. In a gray scale image, one can obtain a histogram to determine the most common intensity level of text on the check. If one assumes that the most common intensity level will correspond to the color of the machine printed text, text that has a significantly different intensity level can be assumed to be handwritten.

4. *Some characters are connected to each other in handwritten text*

   Unconstrained handwriting can be very sloppy. When writing quickly, adjacent characters can be inadvertently connected to each other. If this occurs, the two will be identified together as one character by a connected component extraction algorithm. This *character* will have a width that is much larger than the widths of the other characters in the string.

   This heuristic is risky as well. Due to imperfections in the scanning, smoothing and other preprocessing of a check image, it is possible for two adjacent machine printed characters to appear connected when they are in fact merely very close to each other.

4.1.2 Identifying the Correct Block

In this section, heuristics are outlined for labeling one of the strings on the check as the courtesy amount block. If the check is handwritten, the heuristics above can be used to locate all of the handwritten strings. Then the following heuristics can be applied to only those strings that are handwritten. If the check is machine printed, they may be applied to all of the strings, since the handwriting locators above would have failed to eliminate any string from consideration.

1. *The courtesy amount should have at least 2 characters*

   In US, the courtesy amount usually includes the dollar and cent values for which the check has been written. Therefore the amount contains at least two characters, usually at least three, since checks are seldom

written for less than a dollar. Even when other units of currency are used, there is usually an integer part and a fractional part.

2. *A box may surround the courtesy amount*

   On many check formats, a rectangle denotes the area on the check where the courtesy amount should be written. If this rectangle is a black box that is dark enough to be identified as a connected component, the courtesy amount block can be chosen as the one string that appears completely within another connected component. The difficulty with this heuristic arises when the written amount overlaps, departs from, or makes contact with the box. Then, the offending character(s) will be grouped together with the box as one connected component. In this case, the form lines must be removed [4].

   Usually, the rectangle will be light enough so that it will be considered to be a part of the background by the thresholding algorithm. In this case, the rectangle can still be identified if the original gray scale image is used.

3. *A decimal point may appear in a courtesy amount*

   A decimal point is fairly easy to distinguish within a string. It may be a connected component within a string that has smaller dimensions than the others. It also must appear in the bottom portion of the string. Unfortunately, small stray marks or noise may be identified as decimal points as well.

4. *There are two digits after the decimal point in courtesy amount*

   This heuristic combats, to some extent, the problem described with the last heuristic. The decimal point in a courtesy amount is not in a random location, as a stray mark may be, but rather to the left of the rightmost two characters in the string.

5. *Some digits may be smaller than others*

   Very often, a person will write the numerals denoting the fraction amount in a smaller size than the whole amount. For example, the cents amount (or fractions of other units of currency) may also occur

above other characters, e.g., $\frac{32}{100}$. By analyzing the positions of the various characters in the string, and their relative sizes, one can determine whether a fraction appears in a string.

6. *Other Blocks May Be In Cursive*

   Handwritten portions of the checkin areas other than the courtesy amount block may be written in cursive style. Examples are the signature, the payee, and the courtesy amount in words. Cursive strings will be extracted as single connected components, rather than as strings of adjacent components. Since the courtesy amount will be a string of adjacent characters, these longer cursive components can be rejected.

7. *The date will appear above the courtesy amount*

   On most checks, the date appears above the courtesy amount block. Even if it is not directly above the amount, the date is usually closer to the top of the check. Therefore if two strings are both likely to be numeric strings about 6 characters long, the one further down from the top of the check is more likely to be the courtesy amount.

8. *The date has distinguishing characteristics*

   The date on a check has two characteristics that distinguish it from the courtesy amount. First, the date may contain several short handwritten strings of one to four digits, separated by a machine printed string. (Many checks print the first two digits of the year on the check.) In addition, the date may be written in a smaller size than the courtesy amount, and generally contains shorter strings.

9. *The courtesy amount is in a particular general location*

   This heuristic will only work for checks of certain countries. Many countries, including the US, have the courtesy amount appearing on the right half of the check, and near the center of the check in the vertical direction.

4.2 Using Redundant Information

4.2.1 Arriving at a Consensus

The preceding section listed many heuristics that can be applied when attempting to locate the courtesy amount block. Though only few of them may suffice to identify the correct block, the use of redundant information serves to improve the chances that the chosen block is the correct one.

The likelihood that a particular string is indeed the courtesy amount depends on the number of tests that the string passes. It is possible that one, or even several of the heuristics will point to an incorrect choice. When many rules are applied, however, the hope is that the correct choice will be clear. If no block is clearly of a higher likelihood than the others, the check can be transmitted for analysis by human operators, without a choice being made. The use of redundant information is necessary because significant variability exists, and the consequences of an incorrect choice are severe.

As the system is tested on check data, it can be determined what heuristics are most accurate at locating the correct block. The results of those heuristics can be given a greater weight in choosing the courtesy amount block.

4.2.2 Verification

Once a choice has been made, further verification can take place. Again, this is warranted by the fact that these heuristic methods are subject to a great degree of variability, headed by the problems described above with each rule. The following rule can be used to verify the choice of string: **A dollar sign, or other special character will appear to the left of the courtesy amount**

Most checks will have a machine printed dollar sign, (or pound mark, etc.) on the check. In addition, a number of individuals write their own dollar sign out of *force of habit.*

When a string has been chosen, the system can find the string to the left of the chosen string, and determine if that string contains only one character. If only structural information is to be used, the verification routine can use the knowledge that there is indeed a lone symbol to the left of the chosen block. The following algorithm may be used to determine which string is the one to the left of the chosen string.

```
GetLeft(String, Stringlist)
{
  LeftString = NULL;
  VertCenter = (String->top + String->bottom) / 2;
  for (each string S in Stringlist)
    {
      if (VertCenter is Between S->top and S->bottom &&
  S->right < String->left)
if (LeftString == NULL ||
    S->right > LeftString->right)
  LeftString = S;
    }
  return(LeftString);
}
```

Alternatively, a simple symbol recognizer can be used on that lone symbol to determine if it is indeed a currency notation symbol such as a dollar sign. A neural network can be trained to recognize a dollar sign, as well as currency symbols from other nations. The neural network can be of the same variety that will eventually be used to recognize individual digits of the courtesy amount.

5 Results and Discussion

5.1 Choosing a Block

Once the set of strings is determined Section 3, application of the heuristics described in Section 4 can begin. Associated with each string is a likelihood value, representing the likelihood that that particular string is the courtesy amount block. Initially, all likelihood values are set to zero. As the heuristics are applied to each string, the string's likelihood value is incremented for each test passed by the string.

Heuristics 1 and 2 from the handwritten string location set have been implemented from the data structures involved. For Heuristic 2, the test for regularity involves determining whether the bottom of any character in the string deviates from the bottom of any other by more than 1/4 of the average

character height. Heuristic 3 could not be applied, since color or gray scale images were never used. Heuristic 4 was determined not to be reliable enough for use in the system. Machine printed characters too often appeared to be connected due to low image resolution.

From the correct block identification set, the result of Heuristic 6 is achieved implicitly through the implementation of Heuristic 1. Strings written cursively should contain only one connected component. This string will fail the test of having at least 2 or 3 components. Most of these heuristics are based on the structure of the string, or of the connected components contained within it, and therefore involve simple computation of sizes, positions, and numbers of characters. The heuristics involving distinguishing characteristics of the date were not implemented due to the variations in the appearance of the date. Depending on how it is written, it may appear as one, two or three separate strings.

5.2 Verification of Block Choice

After all of the heuristics above are applied, the string with the highest likelihood value is extracted. This string S becomes the subject of the verification process described here. If the choice is verified, S is returned by the CABL as the correct courtesy amount string. If it is not, NULL is returned.

```
Verify(String, StringList)
{
  /* String has a higher likelihood than any other in StringList */
  Left = GetLeft(String, StringList);
  if (Left->NumberOfCharacters == 1)
    {
      TheChar = FirstCharacter(Left);
      if (AspectRatio(TheChar) is between .5 and 2.0)
{
  /* there is a character to the left of the chosen string
     that has the dimensions of a currency sign */
  String->likelihood = String->likelihood + 1;
  if (IsCurrencySign(TheChar))
    return(VERIFIED);
}
```

```
    }

  /* Check if the first character of String is a currency sign */
  First = FirstCharacter(string);
  if (IsCurrencySign(First))
    return(VERIFIED);
  else
    {
      /* Can't find a currency sign.  Verify only if no other string has
 a likelihood within 3 of String's likelihood */
      for (each S in Stringlist besides String)
{
  if(String->likelihood - S->likelihood < 3)
    return(NOT_VERIFIED);
}
      return(VERIFIED);
    }
}
```

First, the string closest to the left of S is determined, using the algorithm described in Section 4.2.2. If this string contains only one character, and that character has an aspect ratio that makes it likely to be a dollar sign or other special currency symbol, the likelihood of that string is incremented.

Now, one would like to verify whether or not this single character is indeed a dollar sign. A neural network recognizer was trained to recognize dollar and other currency signs. If the results suggest that it is indeed a correct sign, the choice is verified.

If the recognizer can find no currency sign, another decision must be made. Is the CABL confident enough in its answer even without this verification, to report that string S is the courtesy amount? If the difference between the likelihood of string S and any other string is less than 3, the confidence level is not very high. String S is not a *clear* choice so the string is rejected. However, if no other string has a likelihood within 3 of that of string S, S is verified.

5.3 Testing

Preliminary testing of the CABL described in Section 3 was conducted in the following manner. Test images of US and British Checks were input to the system. Key features of the image were varied among the tests. For example, some of the courtesy amounts included decimal points, while others included fractions. The dates were written in several different fashions. The dollar amount in words sometimes included a fraction, and sometimes contained a line at the end of it. Often, words or digits extended below form lines on the check. The payee was written in either cursive style or in print.

Results of these preliminary tests were very promising. Out of 5300 U.S. checks, the dollar amount was correctly located 5199 times. On the checks that failed, the string generator did not correctly group together all digits in the dollar amount. The four digits of the integer dollar amount were grouped together, and that string was reported as the dollar amount. The cents part, which was written in the form of a fraction, was grouped as a separate string.

The CABL correctly located the courtesy amount block on 90% of tested British checks. The decimal point followed by two digits was a strong cue in that check, however only an integer amount appeared on the check which was not correctly analyzed. The CABL reported that a choice could not be made from this image. In no case was a string reported as the courtesy amount that had been incorrectly identified.

5.4 Performance

The images were originally scanned at 300 DPI. A typical check is approximately 2.75 inches by 6 inches, so that at least 1.4 million bits are needed to store the image. If gray scale is used, that number increases to about 1.5 million bytes. The CABL was implemented in C on UNIX workstations. The time necessary just to read an image file 1.5 megabytes long from disk was improved to approximately 2 seconds. The application of the all of the algorithms and heuristics took several seconds to complete. Since it takes a human an average of 4-8 seconds to locate *and* recognize the courtesy amount [20], these times are comparable.

In order to further decrease both computational intensity and storage required for the image files, the images can be converted to 100 DPI. The CABL required a more reasonable faster speed to correctly locate the cour-

tesy amount in an image of this size. Some failure examples are shown in Figure 11.

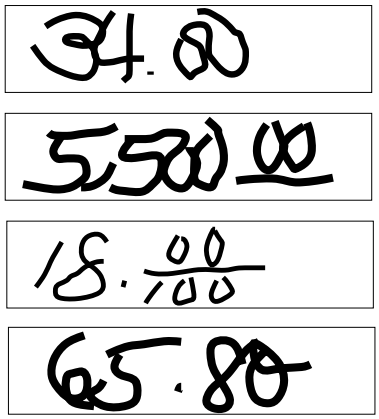

Figure 11. Some examples of failures.

At this moment, the database of check images and the corresponding courtesy amounts are not available for other researchers yet.

5.5 Future Improvements

5.5.1 Generation of Strings

Improvements can be made on many of the methods used in the implementation of the CABL. Currently, the only way these lines are eliminated from consideration is by discarding connected components that have a width/height ratio greater than a theshold value. However, if some text character touches a line, or passes through it, it will be part of a connected component with the line, and may be discarded as well. Therefore, line removal as a preprocessing step would add to the robustness of the system.

One place where improvement may be necessary is in the grouping of connected components to form strings. An optimal solution to this problem must involve feedback from a character recognizer. This can be verified by looking at the highlight image of a check, that is, an image containing only MBR's of characters that appeared in the original check image. When one looks at such an image, which is devoid of semantic information, it is often impossible for a human to decide which MBR's should be grouped together into a string.

5.5.2 Application of Heuristics

The likelihood value associated with each string depends on which tests the string passes. A weighting factor was given to each heuristic, based on an informal survey of types of checks and types of writing that are likely to be encountered. A more rigorous method of assigning weighting factors to the heuristics would make the heuristics more reliable.

There is also room for improvement in the heuristics themselves. More complicated heuristics can be developed to try to classify each string on the check as part of some structural piece of the image, e.g. the date, courtesy amount, signature. In this way, the CABL can be less likely to return a partial string, incorrect string, or string that includes parts of more than one structural element of a check.

The neural net that was trained to recognize dollar and pound signs could be further trained to recognize currency symbols from other nations, or other special symbols. In this way, verification would be possible for more checks.

5.6 Summary

This research work is a continuation of other related effort on check analysis and recognition by the Imaging Lab of MIT SLoan School. Part of earlier results were represented and published in [1, 14, 15, 21, 30]. It has been shown that bottom up information from connected component extraction was combined with the top down method of smearing, in order to properly segment the check image into text strings, while consuming minimum time. Dynamic information was used in the grouping of characters into strings. Top level information about the likely format of the check and the handwritten courtesy amount was encoded in the heuristics that were applied to the strings.

The methods used by the CABL were chosen to be general, wherever possible, so that changes in the problem domain do not necessitate implementing an entirely new system; however, any constraints specific to the check understanding problem were accounted for at each step, so as to make the system accurate as well as robust.

References

[1] A.Agarwal, A.Gupta, K.Hussein, and P.Wang. Bank check analysis and

recognition by computers. In H.Bunke and P.Wang, editors, *Handbook on OCR and Document Image Analysis*. World Scientific Publishing Co., 1996.

[2] H. Bunke, H. Baird, and K. Yamamoto(ed). *Structured Document Image Analysis*. Springer-Verlag, 1992.

[3] H. Bunke, P.S.P. Wang, and H. Baird(ed). *Advances in Document Image Analysis and Recognition*. WSP, 1995.

[4] Richard Casey, David Fergueson, K. Mohiuddin, and Eugene Walach. Intelligent forms processing system. *Machine Vision and Applications*, 5(3):143–155, Summer 1992.

[5] Edward Cohen, Jonathan J. Hull, and Sargur N. Srihari. Understanding handwritten text in a structured environment: Determining zip codes from addresses. In P.S.P. Wang, editor, *Character and Handwriting Recognition: Expanding Frontiers*, pages 221–264. World Scientific Publishing Co., 1991.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, and McGraw-Hill Book Company, 1991.

[7] Y. Le Cun, O. Matan, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, and H.S. Baird. Handwritten ZIP code recognition with multilayer networks. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 35–40, Atlantic City, New Jersey, June 1990.

[8] A. Downton and C. Leedham. Preprocessing and presorting of envelope images for automatic sorting using OCR. *Pattern Recognition*, 23(3/4):347–362, 1990.

[9] A. Downton, R. Tresidso, and C. Leedham. Recognition of handwritten British postal addresses. In *From Pixels to Features III*, pages 129–144. Elsevier Science Publishers, 1992.

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

[11] J. Fisher, S. Hinds, and D. D'Amato. A rule based system for document image segmentation. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 567–572, Atlantic City, New Jersey, June 1990.

[12] L. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE PAMI*, 10(6):910–918, November 1988.

[13] Rafael C. Gonzales and Paul Wintz. *Digital Image Processing.* Addison-Wesley, 1987.

[14] A. Gupta, M.V. Nagendraprasad, A. Liu, P.S.P. Wang, and Ayyadurai. An integrated architecture for recognition of totally unconstrained handwritten numerals. *IJPRAI*, 7(4):757–773, 1993.

[15] Amar Gupta. A collection of papers on handwritten numeral recognition. Technical Report IFSRC 219-92, MIT Sloan School of Management, 1992.

[16] I. Guyon. Applications of neural networks to character recognition. In P.S.P. Wang, editor, *Character and Handwriting Recognition, Expanding Frontiers*, pages 353–382. World Scientific Publishing Co., 1991.

[17] I. Guyon and P.S.P. Wang(ed). *Advances in Pattern Recognition Systems using Neural Network Technologies.* WSP, 1994.

[18] S. Hinds, J. Fisher, and D. D'Amato. A document skew detection method using run length encoding and the hough transform. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 464–468, Atlantic City, New Jersey, June 1990.

[19] Stephen Lam, Dacheng Wang, and Sargur Srihari. Reading newspaper text. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 703–708, Atlantic City, New Jersey, June 1990.

[20] Jean-Vincent Moreau. A new system for automatic reading of postal checks. In *From Pixels to Features III*, pages 171–184. Elsevier Science Publishers, 1992.

[21] L. Mui, A. Agarwal, A. Gupta, and P.S.P. Wang. An adaptive modular neural network with application to unconstrained character recognition. *IJPRAI*, 8(5), 1994.

[22] M.V. Nagendraprasad, Alexis Liu, and Amar Gupta. A system for automatic recognition of totally unconstrained handwritten numerals. Technical Report IFSRC 218-92, MIT Sloan School of Management, 1992.

[23] M.V. Nagendraprasad, P.S.P. Wang, and A. Gupta. Algorithms for thinning and rethickening digital patterns. *Journal of Digital Signal Processing, Academic Press*, 3(2):97–102, 1993.

[24] L. O'Gorman and R. Kasturi(ed). *Document Image Analysis*. IEEE-CS Press, 1994.

[25] Peter Sparks, M.V. Nagendraprasad, and Amar Gupta. An algorithm for segmenting handwritten numerical strings. Technical Report IFSRC 214-92, MIT Sloan School of Management, 1992.

[26] S. Srihari. Feature extraction for locating address blocks on mail pieces. In *From Pixels to Features*, pages 261–275. North Holland, 1987.

[27] S. Srihari, C. Wang, P. Palumbo, and J. Hull. Recognizing address blocks on mail pieces. *The AI Magazine*, 8(4):25–40, December 1987.

[28] C.Y. Suen and P.S.P. Wang(ed). *Thinning Methodologies for Pattern Recognition*. WSP, 1994.

[29] P.S.P. Wang. Learning, representation, understanding and recognition of words - an intelligent approach. In S. Impedovo, editor, *Fundamentals in Handwriting Recognition*, pages 81–112. Springer-Verlag, 1994.

[30] P.S.P. Wang and A. Gupta. An improved structured approach for automated recognition of handprinted characters. *IJPRAI*, 5(2):97–121, 1991.

[31] P.S.P. Wang(ed). *Intelligent Chinese Language Pattern and Speech Processing*. WSP, 1988.

[32] P.S.P. Wang(ed). *Character and Handwriting Recognition - Expanding Frontiers (preface by C.Y.Suen)*. WSP, 1991.