A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images

LLOYD ALAN FLETCHER AND RANGACHAR KASTURI, MEMBER, IEEE

Abstract—An automated system for document analysis is extremely desirable. A digitized image consisting of a mixture of text and graphics should be segmented in order to represent more efficiently both the areas of text and graphics. This paper describes the development and implementation of a new algorithm for automated text string separation which is relatively independent of changes in text font style and size, and of string orientation. The algorithm does not explicitly recognize individual characters. The principal components of the algorithm are the generation of connected components and the application of the Hough transform in order to group together components into logical character strings which may then be separated from the graphics. The algorithm outputs two images, one containing text strings, and the other graphics. These images may then be processed by suitable character recognition and graphics recognition systems. The performance of the algorithm, both in terms of its effectiveness and computational efficiency, was evaluated using several test images. The results of the evaluations are described. The superior performance of this algorithm compared to other techniques is clear from the evaluations.

Index Terms—Connected component analysis, document analysis, image processing, image segmentation, image understanding, text and graphics separation, text recognition.

INTRODUCTION

THE move from paper-based documentation towards computerized storage and retrieval systems has been prompted by the many advantages to be gained from the "electronic document" environment. Document update and revision is efficiently achieved in the computerized form. For efficient processing and storage of documents, however, it is necessary to generate a description of graphical elements in the document rather than a bit-map in order to decrease the storage and processing time. Thus, increasing emphasis is being placed on the need for the realization of computer-based systems which are capable of providing automated analysis and interpretation of paper-based documents [1]-[3]. Much of the attention paid to automated document analysis systems in the literature has been in relation to engineering drawings and diagrams [4], [5]. Such systems provide a means for originating technical information (text and graphics) in a digital form suitable for interactive graphics editing, reproduction, and distribution.

Manuscript received August 8, 1986; revised June 12, 1987. Recommended for acceptance by J. Kittler. This work was supported in part by the National Science Foundation under Grant ECS-8307445.

L. A. Fletcher was with the Department of Electrical Engineering, the Pennsylvania State University, University Park, PA 16802. He is now with Bell Communications Research, 331 Newman Springs Road, Red Bank, NJ 07701.

R. Kasturi is with the Department of Electrical Engineering, the Pennsylvania State University, University Park, PA 16802.

IEEE Log Number 8823861.

The initial processing stage in an automated document analysis system requires conversion of paper-based graphics/text to a digital bit-map representation. Wherever the primary goal of the automated document analysis system is interpretation of graphic data, text strings present within the digitized document must first be separated from the graphics in order that subsequent processing stages may operate exclusively on the graphic information. The extracted text may be stored separately for input to a character recognition system for later retrieval or revision. Since document types vary widely in style and content of both graphic and text data, an algorithm to perform text string removal must be able to accommodate documents containing text of various font styles and sizes. Further, the documents may contain text strings which are intermingled with graphics, and text characters which are similar in size or shape to graphics. In general, text strings may be of any orientation in the image; not simply horizontal or vertical but possibly diagonally aligned.

Several algorithms for text string separation have been reported in the literature [6]-[9]. However, many of these algorithms are very restrictive in the type of documents they can process and are therefore not useful in a general automated document analysis system. For example, the combined symbol matching algorithm [7] is sensitive to changes in text font style and size: the algorithm must be run once for each font type using different parameters, words of less than three characters which are embedded in longer strings are not removed, and the string removal is only along a specified orientation. The Block Segmentation technique [8] broadly classifies regions into text or graphics; i.e., characters which lie within predominantly graphics regions are classified as graphics. The Bley algorithm [9] is also sensitive to variations in text font style and size; the algorithm breaks connected components into subcomponents, which makes it difficult to process the components for graphics recognition. Thus there is no single algorithm which is robust enough to segment images containing mixed graphics and text, with multiple font styles and sizes and strings of arbitrary orientation. This paper describes a new robust algorithm for text string separation from mixed text/graphics images.

A ROBUST ALGORITHM FOR TEXT STRING SEPARATION

A robust algorithm for text separation has been designed to separate text strings from graphics, regardless of string orientation and font size or style. The algorithm uses simple heuristics based on the characteristics of text strings. The algorithm performs no character recognition and can only separate characters from text on the basis of size and orientation. Thus, for proper operation, the document should meet the following requirements:

1) The range in text font sizes within an image should be such that the average height of the characters belonging to the largest font is not greater than five times the average height of characters belonging to the smallest character font.

2) The interline spacing between parallel lines of characters should not be less than one quarter of the average height of characters in the line having the larger font size.

3) The resolution of the digitizer used should be such that each text character forms an isolated component not connected to graphics or other characters.

4) The largest character in any phrase should not have an area greater than five times the average area of all characters in the phrase.

5) The intercharacter gap between characters in the same word should not be greater than the average local character height. Also, the interword gap between words of the same phrase should not be greater than 2.5 times the average local character height.

If these constraints are satisfied for any particular image, the performance of the algorithm will be highly reliable. However, acceptable segmentation is obtained even if some of these constraints are not strictly satisfied. For example, the algorithm will correctly segment text strings even when some of the characters are eight connected to each other, as long as there are at least three components in each string.

The algorithm described here is designed to accomplish the following without explicitly recognizing individual characters:

- Locate potential text strings and separate them from the large graphics.
- Examine the strings and separate all phrases that contain at least three characters.
- Generate two images, one each for the text strings and graphics.

The algorithm consists of the following steps:

- Connected component generation.
- Area/ratio filter.
- Collinear component grouping.
- Logical grouping of strings into words and phrases.
- Text string separation.

These processing steps are described in detail in the following sections. A test image of 2048 × 2048 pixels, shown in Fig. 1, is used to demonstrate the performance of this algorithm.

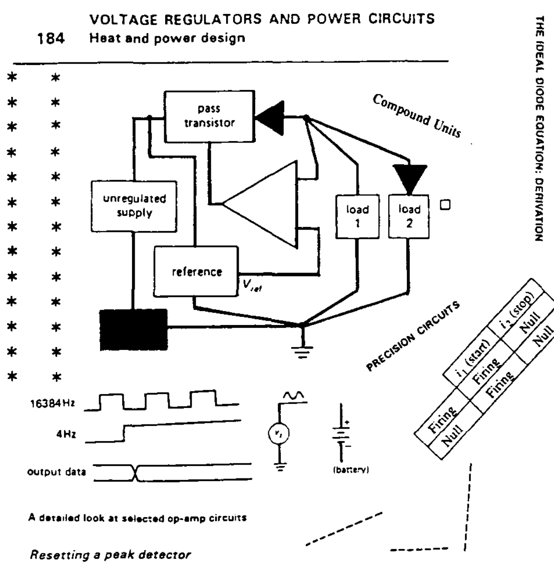


Fig. 1. Test image 1.

Connected Component Generation

Connected component generation involves grouping together black pixels which are eight connected to one another (assuming a black image on white background). In this technique, eight connected pixels belonging to individual characters or graphics are enclosed in circumscribing rectangles. Each rectangle thus identifies a single connected component. The output of the connected component generation algorithm is an information array which specifies: the maximum and minimum coordinates of the circumscribing rectangles of connected components, the coordinates of the top and bottom seeds of each connected component, and the number of black pixels. Black pixel density and dimensional ratios are easily obtained from this information. This algorithm is described in detail in [10]. Although this algorithm is similar to the algorithms described in the literature [11], [12], in our algorithm we do not change the value of each pixel to its corresponding component label, but retain the coordinates of top and bottom seeds for each component.[1] Each connected component is then either rejected or accepted as a member of a text string based on its attributes (size, black pixel density, ratio of dimensions, area, position within the image etc.).

The enclosing rectangles corresponding to the connected components of Fig. 1 are shown in Fig. 2. Note that the largest graphic is enclosed by a single rectangle. Also, the long line near the top of the image is enclosed by a rectangle. This rectangle has a height which is significantly greater than the thickness of the line since the line is oriented at a slight angle to the image axis.

Area/Ratio Filter

By initial examination of connected component attributes, the working set of connected components can be reduced to one which contains a higher percentage of

[1]An image typically contains more than 256 components. Labeling each pixel with its corresponding component number would require at least 2 bytes per pixel. This would double the memory requirement to 8 MB for a 2048 × 2048 pixel image and significantly deteriorates system performance (through increased page faults). However, this necessitates multiple passes in the text string removal stage to remove all pixels belonging to a particular connected component.

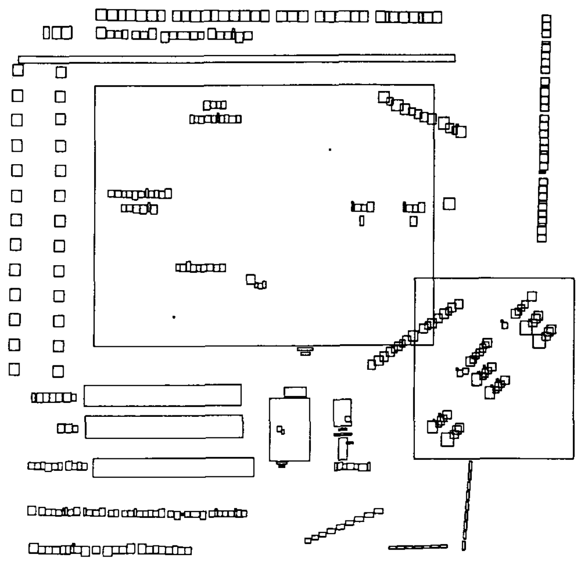


Fig. 2. Rectangles enclosing the connected components of test image 1.

characters. In general, a mixed text/graphics image will produce connected components of widely varying areas. The larger connected components represent the larger graphic components of the image. It is desirable to locate and discard large graphics in order to restrict processing to components which are candidates for members of text strings, thereby improving both accuracy and processing speed.

By obtaining a histogram of the relative frequency of occurrence of components as a function of their area, it is possible to set an area threshold which broadly separates the larger graphics from the text components. By correct threshold selection, the largest of the graphics can be discarded, leaving only the smaller graphics and text components as members of the working set of connected components. The area threshold selection procedure must ensure that the threshold lies outside that part of the histogram which broadly represents the set of text characters. The threshold itself is determined using the area histogram (as opposed to presetting it to a value) although *a priori* knowledge about the types of documents being processed (such as the amount of text data compared to graphics), if available, is helpful in determining this threshold. For documents that contain a substantial amount of text and some large graphics, the most populated area will, in general, represent mostly text components or, at least, small graphics. By ensuring that the threshold is set above the most populated area, A_{mp}, the possibility of discarding members of the text character set is avoided. This alone is not adequate to process images that contain text strings of different sizes (it is likely that the most populated area corresponds to the smallest sized characters). Thus, a second parameter, the average area A_{avg}, is computed. The area threshold is then set at five times the larger of the two parameters A_{mp} and A_{avg}. The histogram is searched to locate components that are larger than this threshold. Since text characters of the same size may have different areas, the histogram is "blurred" such that neighboring areas are grouped together in order to determine the most populated area.

A similar type of filtering approach is taken with the connected component dimensional ratio attribute. Very long lines are unlikely to be text characters, and should therefore be discarded. Isolated straight lines within the image may be discarded on the basis of dimensional ratio. If a connected component has a ratio of less than 1 : 20 or greater than 20 : 1, it is discarded.

Collinear Component Grouping

Text strings are composed of characters which are oriented along the same straight line. In order to logically connect or group together characters into strings, it is necessary to determine which characters in the image actually lie along any given straight line. By grouping components into strings associated with a particular line, the components can be ordered according to their distance along the line. Further grouping may then take place by examining the distance between characters. By comparing the intercharacter distance with the interword gap and intercharacter gap thresholds, the string can be segmented into logical character groups, that is, into logical words and phrases. Only if components belong to a logical character group can they be regarded as members of a valid text character string. Such components are identified and are separated from the image. These steps are explained in detail in this and the following sections.

The Hough transform is a line to point transformation [13] which, when applied to the centroids of connected components in an image, can be used to detect sets of connected components that lie along a given straight line. A line in the Cartesian space (x, y) given by the equation

$$\rho = x \cos \theta + y \sin \theta \qquad (1)$$

is represented by a point (ρ, θ) in the Hough domain. Similarly, every point (x, y) in the Cartesian space maps into a curve in the Hough domain. Thus to locate all the connected components that are collinear, the Hough transform is applied to the centroids of the rectangles enclosing each connected component. All the curves corresponding to the collinear components intersect at the same point (ρ, θ), where ρ and θ specify the parameters of the line.

In the discrete case, the Hough domain is a two-dimensional array representing discrete values of ρ and θ. The resolution along the θ direction is set to one degree. The resolution along the ρ direction is treated as a variable R. Optimal selection of R is critical to correct grouping of connected components. This is because in any text string, the centroids of all characters belonging to a phrase are not necessarily collinear. For any phrase, character heights vary (upper, lower case) and character positions vary in relation to each other (ascenders, descenders). In general, the centroids of characters belonging to the same phrase will lie in the range $\pm\delta_c$ from the axis of the line connecting the phrase members. However, δ_c is a function of the text string height. Thus, it is important to select a Hough domain resolution, before applying the transform, which will cause the majority of component centroids belonging to a phrase to be grouped into the same cell. The optimal resolution is dependent on the heights and relative positions of characters within a string. Too large a value for R may cause several parallel strings to be grouped into a single cell (over-grouping). If R is too small then connected components belonging to the same string may be grouped into different cells (undergrouping). The value of R should depend on the average local character height, H_a, for a line l. However, since H_a cannot be determined until characters have actually been detected as belonging to a collinear string, and since R must be known before such grouping, an initial estimate for R is made based on the average height, H_{ws}, of all connected components in the current working set. The resolution is then set to: $R = 0.2 \times H_{ws}$.

The resolution R allows for some perturbations in the collinearity of components belonging to text strings. However, this will not be adequate to group all ascenders and descenders in a character string. Further, the value of R, which is based on average character height, will not be optimal for grouping text strings of different heights. Thus, clustering in the Hough domain may become necessary for each primary cell. Here, "primary cell" refers to the cell containing the majority of the connected components in a string. In order to group together *all* components belonging to a string, neighboring cells must be clustered into the primary string. The required degree of clustering, δ_{clus}, will depend on the local character height. However, when strings are initially extracted, 11 cells centered around the primary cell are grouped for examination. Once the string has been extracted, the local clustering factor can be determined.

The average local character height for a string H_a is used to determine the degree of clustering around the primary cell. This will ensure that all ascenders and descenders in a string are included in the text string grouping. The clustering factor is then set to H_a/R.

Since the vast majority of documents contain text strings which are parallel to the document axes, it is advantageous to remove these strings first. To allow for orientation errors during digitization, a tolerance of five degrees is allowed in processing horizontal and vertical lines. Strings at other orientations are processed after considering the horizontal and vertical strings. Furthermore, more populated cells in the Hough domain are processed before processing less populated cells. This order of string processing gives greatest weight to longer strings. If shorter phrases or words were treated before longer ones, a degradation in performance would be observed.

In summary, extraction of strings from the Hough domain is performed as follows:

1) Calculate the average height of components H_{ws} in the working set.

2) Set the Hough domain resolution R to $0.2 \times H_{ws}$. Set a counter to zero.

3) Apply the Hough transform to all components in the working set for θ in the range: $0° \leq \theta \leq 5°$, $85° \leq \theta \leq 95°$, $175 \leq \theta \leq 180°$. (In our implementation ρ is allowed to have negative values and $\theta_{\max} = 180°$.) Set the running threshold $RT_c = 20$.

4) For each cell having a count greater than RT_c, perform steps 5–10.

5) Form a cluster of 11 ρ cells (constant θ), including the primary cell, centered around the primary cell.

6) Compute the average height of components in the cluster H_a.

7) Compute the new clustering factor $f_{clus} = H_a/R$.

8) Re-cluster $\pm f_{clus}$ cells centered around the primary cell, including the primary cell.

9) Perform string segmentation (explained in detail in a later section).

10) Update the Hough transform by deleting the contributions from all components discarded in step 9.

11) Decrement RT_c by one. If RT_c is greater than 2, go to step 4.

12) If the counter is equal to 1 then stop. Otherwise compute the Hough transform for the remaining components for θ in the range: $0° \leq \theta < 180°$. Reset RT_c to 20. Increment the counter by one. Go to step 4.

Step 10 above is performed in order to "refresh" the Hough domain. If the Hough array is not continually updated, discarded characters contribute to cell counts despite having been removed. This can lead to a severe degradation in processing speed since information about discarded components is unnecessarily extracted from the Hough domain. Thus, the refreshing of the Hough domain is designed to improve performance.

Logical Grouping of Strings into Words and Phrases

Fig. 3 shows the connected components associated with the line l_{pr}. Note that the centroids of these components do not necessarily lie on the line l_{pr}. It is necessary to order these components according to their distance along the line [from a reference point, say $(0, y_0)$]. That is, for each component the distance along the line l_{pr} is calculated.

In order to locate words and phrases, the positional relationships between connected components are examined. It is the intercharacter gap and interword gap thresholds, T_c and T_w, which are used to determine component grouping. These parameters are functions of the character height. In computing this, an average of the height of several local characters should be used rather than simply using the height of the component currently under examination. For example, the component under examination may be a period at the end of a string, which would have very small height. For this reason the four neighboring (two on each side) characters are used, along with the current component, to determine the thresholds T_c and T_w. These operations are explained in the following paragraphs.

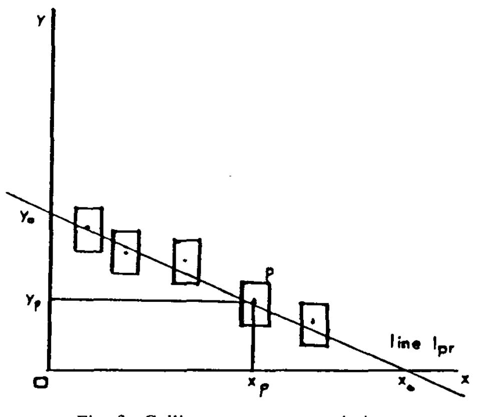


Fig. 3. Collinear component ordering.

Associated points of the line l_{pr}, once ordered, become part of a data structure containing all information about components belonging to all collinear strings. An entry is made in the string information array for each associated point, pointing to the connected component to which each point belongs. The information contained in the connected component array is then used to compute the edge-to-edge distance, D_e, for every pair of adjacent components. The magnitude of D_e belonging to the associated point i, is the distance between the two closest edges of the connected components associated with collinear component elements i and $i + 1$.

By placing a window over each connected component associated with the string information array, and considering four nearest neighbors, the average height of connected components within the window H_c is determined. In computing this average the vertical dimension of the enclosing rectangles is used for lines oriented between 0 and 45 degrees and the horizontal dimension is used for strings oriented at angles greater than 45 degrees. T_c is the intercharacter gap threshold, set at $T_c = H_c$. The interword gap threshold T_w is set at $2.5 \times H_a$. Two adjacent connected components separated by an edge-to-edge distance D_e of less than or equal to T_c belong to the same word group. Word groups separated by no more than T_w belong to the same phrase group. Connected components which belong to neither of these types are labeled as isolated characters. Note, however, that single characters which are embedded in a multiword phrase become part of the phrase group. Isolated words of less than N ($N = 3$ in our experiment) characters are not removed from the image. Further, when several isolated characters satisfy the interword gap to form a phrase containing more than three characters, the condition is checked and the characters are not removed.

The string segmentation algorithm is summarized as follows:

1) Define the data structure *String* of which each element is an associated point belonging to the same line. Define the data structure *Groups*, a vector of 100 elements. Each element of *Groups* contains four fields. The *Type* field may have the label "i," "w," or "p" (initialized to "i"), specifying the group type. The fields *Head* and *Tail* serve as pointers to the first and last connected components within the group. The last field *Num* contains the number of connected components belonging to the group. The data structure *Phrases* is identical to *Groups* except that it does not possess the *Type* field.

2) For each connected component associated with the currently active string, determine T_c and T_w based on the neighborhood average character height H_c. If $String_i(D_e) \leq T_c$ then elements i and $i + 1$ belong to the same word group. If the current group is a new one, set $Groups_{gn}(head) = i$ and $Groups_{gn}(tail) = i + 1$, where gn is the number of groups currently detected in the string. Otherwise set $Groups_{gn}(tail) = i + 1$. Set $Groups_{gn}(num) = Groups_{gn}(num) + 1$. If the group type is not "p," set $Groups_{gn}(type) =$ "w." Continue the T_c threshold test until the current group is terminated. If the T_c test fails, terminate the group entry by advancing the value of gn by one.

3) Compare $String_i(D_e)$ with T_w. If $D_e < T_w$ then the previous group and the current connected component belong to the same phrase. Make entries in *Phrases* to point to the member groups. Set $Groups_{gn}(type) =$ "p" and $Groups_{gn-1}(type) =$ "p." Begin a new group entry for the associated connected component i. Repeat step 2. If the T_w threshold is not satisfied then the adjacent connected components currently under consideration do not belong to the same phrase: terminate any active entry in *Phrases*. Begin a new group by repeating step 2.

4) Once all entries in *String* have been classified, search for two or more consecutive "i" groups present within a phrase. There should never be more than two consecutive isolated characters embedded within the same phrase. Remove any such sequences by splitting or truncating groups as necessary.

The results of the string segmentation are two data structures which provide information about words belonging to a particular phrase and information about the connected components which make up any words within a phrase.

The problem of "destructive line overlap" occurs if shorter strings, which overlap longer strings, are processed before the longer strings. For example, consider the strings shown in Fig. 4. All the components in Fig. 4(a) belong to at least one phrase. Thus, all of these components must be removed. However, if short strings are removed first, we get Fig. 4(b). Finally, after considering the longer strings, we get Fig. 4(c). Thus, it is important to consider longer strings before shorter strings for grouping and removal. However, these strings belong to different collinear component clusters in the Hough domain. Therefore, it is necessary to examine all clusters to separate longer strings before removing shorter strings.

The string processing procedure thus gives weight to a phrase group depending on the number of connected components belonging to the particular group. In examining cells for the extraction of collinear component sets, the

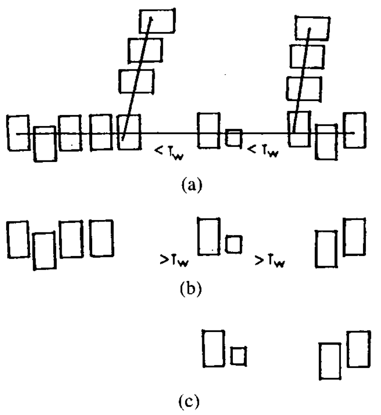


Fig. 4. Effect of processing shorter strings before longer strings: (a) characters oriented along three lines; (b) result of processing shorter strings first; (c) final result: some of the characters are not removed.

running threshold RT_c is used. The running threshold is also used as the threshold for the phrase population. Character groups are only processed if the group population exceeds both RT_c and N. This avoids the confusion which can result from destructive line overlap. Further refinements are also applied to discard disproportionately larger connected components, possibly due to graphics, from text strings.

Words or phrases which consist of several repeated characters, such as dotted or dashed lines, should not be removed from the image. Such components are in fact part of the graphics rather than actual text strings. Groups of such repeated characters possess a high degree of consistency in both ratio and black pixel density. Hence, repeated strings can be detected and discarded by calculating the variance in ratio and density for each phrase of a collinear component set. If both the variance in ratio and density fall below a given threshold for three or more consecutive characters, then these repeated characters of the phrase group are discarded from the working set and the current collinear component set, and are not deleted from the image.

Text String Separation

Once all strings have been extracted from the Hough transform, all connected components corresponding to these strings are deleted from the image array. This involves replacing black pixels, belonging to marked connected components, with white pixels. It is important that only those black pixels which originally formed a particular connected component should be deleted, not simply all black pixels which lie within the area of the circumscribing rectangle. It is for this reason that the seeds of the connected components are retained during connected component generation. Knowing the location of $Seed_t$ and $Seed_b$ of the connected component allows all black pixels which are eight connected to them to be detected and marked for deletion. Using a line by line "scan" method similar to that used in the connected component formation process described earlier, runs of eight connected black pixels are merged together. If a black pixel belongs to a group to which one of the seeds also belongs then the pixel is marked for deletion. The scanning procedure is repeated four times, twice for each seed. The procedure is begun with the top seed, moving south; then starting from the bottom seed and scanning north; then east from the top seed again; finally the component is scanned west from the bottom seed. This multidirectional scan ensures that all possible eight connected groups are detected. All black pixels marked during this procedure are then replaced with white pixels.

The algorithm is implemented in Pascal. The program was run on a DEC VAX 11/785 running the VMS operating system. The complete program includes over 1000 lines of code. The following section presents the results obtained by application of the algorithm to several test images.

Experimental Results

In order to test and evaluate the performance of the algorithm, several test images were required. Several documents were digitized to 2048 × 2048 pixels using a Microtek Image Scanner (interfaced to an IBM PC-AT) with a resolution of 300 pixels per inch. The scanned images in compressed form were transfered to the VAX using KERMIT communications software. The results of the algorithm application to each image are illustrated and discussed in [10]. In this paper we limit the discussion to two test images.

The image shown in Fig. 1 has several characteristics found in different types of mixed text/graphics images: various text font styles and sizes; text strings enclosed by graphics; text strings with various orientations; graphics of various size and shape. In general, most document types do not possess such a wide variation in text font styles, sizes, and orientations within the same document. However, if the algorithm could be made robust to such variations in image characteristics, then it would also be robust when applied to more uniform images.

The result of the removal of all phrases of three or more characters is shown in Fig. 5. Note that several of the characters in the diagonally oriented table, such as "g" and "p" are not removed. These letters are in fact eight connected to the table itself, as can be seen from the connected components in Fig. 4. Thus, these characters are part of the graphics and therefore cannot be isolated for identification as characters. Note that the series of asterisks have not been removed since they constitute repeated "isolated character" phrases. Although the inter-component gap satisfies the interword gap threshold, it does not satisfy the inter-character gap threshold. As a result, the string is discarded since it is a phrase made up of single character words. Of the three dashed lines in the bottom right of the image, two strings have been partially removed. The ratio consistency of these strings was not sufficient to prevent their removal as text. The consistency in both ratio and density of the other dashed line

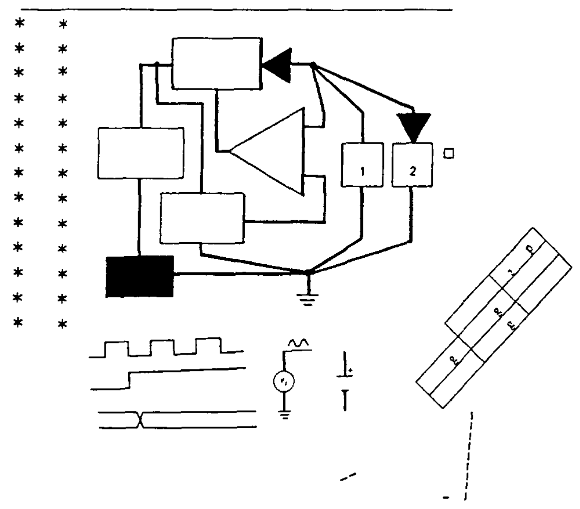
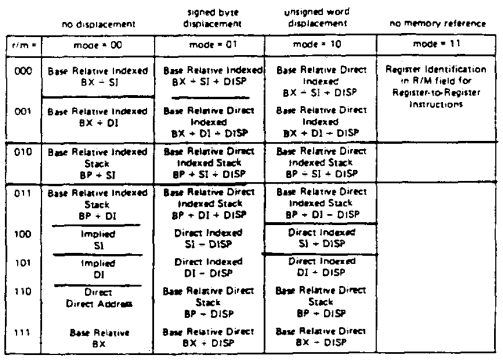


Fig. 5. Output after text string separation.



	no displacement	signed byte displacement	unsigned word displacement	no memory reference
r/m =	mode = 00	mode = 01	mode = 10	mode = 11
000	Base Relative Indexed BX + SI	Base Relative Indexed BX + SI + DISP	Base Relative Direct Indexed BX + SI + DISP	Register Identification in R/M field for Register-to-Register Instructions
001	Base Relative Indexed BX + DI	Base Relative Direct Indexed BX + DI + DISP	Base Relative Direct Indexed BX + DI + DISP	
010	Base Relative Indexed Stack BP + SI	Base Relative Direct Indexed Stack BP + SI + DISP	Base Relative Direct Indexed Stack BP + SI + DISP	
011	Base Relative Indexed Stack BP + DI	Base Relative Direct Indexed Stack BP + DI + DISP	Base Relative Direct Indexed Stack BP + DI + DISP	
100	Implied SI	Direct Indexed SI + DISP	Direct Indexed SI + DISP	
101	Implied DI	Direct Indexed DI + DISP	Direct Indexed DI + DISP	
110	Direct Direct Address	Base Relative Direct Stack BP + DISP	Base Relative Direct Stack BP + DISP	
111	Base Relative BX	Base Relative Direct BX + DISP	Base Relative Direct BX + DISP	

Figure 2.9 Format of the I8086/8088 two-operand instruction such as ADD with variations. Memory-to-memory instructions are not provided. Note that the instruction can occupy two, three, or four bytes. (Reprinted by permission of Intel Corporation. Copyright 1980.)

Fig. 6. Test image 2.

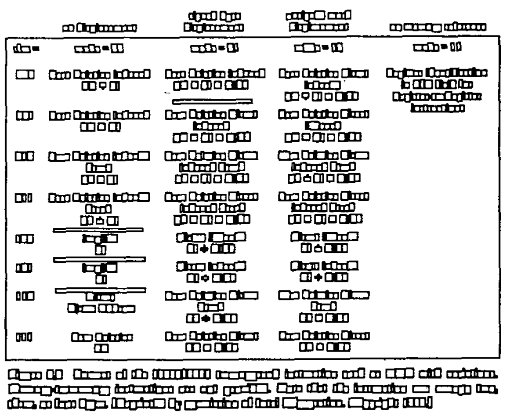


Fig. 7. Connected components of test image 2.

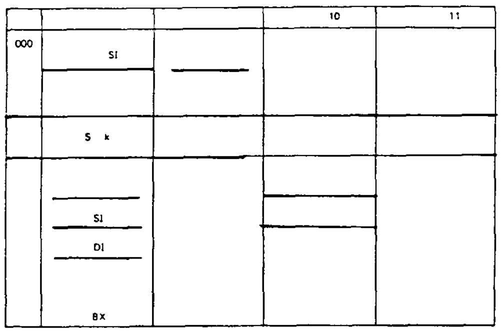


Fig. 8. Output after text string separation.

was high enough for it to be retained as a graphic. Note that three of the small graphics in the neighborhood of the battery figure have been deleted. These were grouped together as a three component phrase since the inter-word gap threshold was satisfied.

The image of Fig. 6 has a large number of text strings. In this respect this image complements the first test image. Due to ink spread in the original document several characters are connected to each other. For example, the characters s and e in most of the occurrence of the word "Base" are connected to each other as can be easily verified in the connected components diagram shown in Fig. 7. In several instances the word contains only two components. The output image after text extraction is shown in Fig. 8. All character strings, including those that are eight connected to each other, have been removed with one notable exception: the letters "S" and "k" in the work "Stack" in second column. In this word the letters a and c are connected to each other making it a string of four characters. The two connected components in the middle of the string are removed as parts of vertical strings (longer than 4 characters) before the algorithm looks for strings which contain four characters. Thus the letters S and k are left behind. The reasons for other components which are not removed are as follows: two of the 0's in the string "000" are connected to each other making it a two character string: the intercharacter gap between the string "SI" at the top of the second column and its previous character is large compared to the height of the previous character (the + sign is made up of two disjoint components of relatively small heights; see Fig. 7). Similar comments hold good for the strings 10 and 11 that are next to "=" signs. Other strings which have not been removed contain less than three characters.

Performance Evaluation

The results of the application of the text string separation algorithm, given in the previous section, demonstrates the robustness of the algorithm to images of various types. In this section the algorithm is evaluated in terms of processing speed.

The time required to generate connected components depends not only on the number of components within the image, but also on the complexity of the information represented by the black pixels. In general, however, processing time increases in direct proportion to the number of connected components in the image. The time required for the application of the Hough transform to the active set of components also increases with the number of connected components. However, the time for a single Hough transform consumes only about five percent of the total processing time for an entire image.

The algorithm is implemented in several distinct steps. In Table I, a breakdown of the processing times required for several operations applied to Test Image 1 is given. Times for operations on both the whole image (reading, writing, connected component generation and deletion) and operations on character strings (segmentation, area thresholding) are given. Steps 1–7 and step 12 in the table are operations performed on the whole image array. Steps 8–11 are performed on a single collinear component set. The CPU time for each of these operations is given for a collinear component set of 15.

Several string operations typically require processing times of only a few tenths of a second. These processing times vary with the number of connected components in a collinear component set. Although individually the string processing steps require relatively small processing time, when several hundred collinear component sets are processed (as in a typical image) the cumulative processing time becomes significant and is directly related to the number of components in an image and the complexity of the image. The processing time increases as the cell population threshold decreases, since more cells become available for extraction. Many more collinear component sets of lower population are processed than sets of higher population. As the population threshold RT_c decreases, the processing time for the working set rapidly increases.

TABLE I
Breakdown of CPU Times for Processing Test Image 1 and for Processing Steps for a Collinear Component Set of Population 15

Processing step	CPU time (secs.) for each operation	CPU time (secs.) for whole image
1. Read 2K image	-	48.5
2. Write 2K image to disk	-	50.8
3. Generation of CC's (395)	-	47.7
4. Initialize *Info* array	-	0.14
5. Filter out CC's based on area and ratio	-	0.47
6. Hough transform (393 CC's, 0°, 90°)	-	9.3
7. Hough transform (270 CC's, all angles)	-	18.9
8. *Clustering* and *String* extraction	0.54	243.3
9. *String* element initialization	0.04	94.8
10. String grouping	0.42	148.5
11. String refinement	0.11	94.3
12. Connected component deletion	-	8.50
Total		764.31

Algorithm Improvements

The algorithm is highly CPU intensive. In order to be useful in automated document processing, the processing time should be minimized. A hardware implementation of the algorithm would achieve a reduction in the required CPU time. However, certain modifications might be made to the algorithm itself to improve performance. For example, collinear component set extraction from the Hough domain involves, effectively, reapplication of the Hough transform to all active components for a given cell with values of ρ and θ. The required processing time for these steps could be reduced drastically if each Hough domain cell contained information about the components which belong to it. This would require cells to maintain a list of possibly up to one hundred component entries. Due to the limited memory on our computer, it was not possible to implement this feature.[2]

It is possible that the algorithm may be made even more robust to rapid changes in image characteristics over small areas of the image. Several of the processing steps can be made more adaptive by using several iterations in order to gather more accurate statistics about strings within the image (e.g., calculation of the thresholds T_w and T_c based on local component height). Such modifications will result in a degradation in processing speed. There is a definite tradeoff between the robustness of the algorithm and the processing speed. Making the algorithm more adaptable will result in longer processing times.

Some problems are encountered when dealing with dotted or dashed lines. The consistency in density and ratio in such graphics strings is not always as high as might be expected. Also, the intercharacter gap is not always consistent, resulting in the classification of some dashed lines as character strings. Some further processing may be required in order to fully accommodate the detection of repeated graphic strings.

Summary and Conclusions

A new algorithm for text string separation from mixed text/graphics images has been presented. The algorithm is robust to changes in text font style and size within an image. The algorithm also accommodates the separation of text strings of any orientation. The algorithm adapts to changes in text characteristics within the image in order to achieve optimal performance. The algorithm is highly reliable provided that documents conform to several constraints on their characteristics. The algorithm presented has several advantages over previously applied techniques. With improvements made to the algorithm in terms of processing speed and efficiency of data representation, the application of the algorithm will become highly appropriate for use in document analysis systems.

[2]Our system has 8 MB of memory. Executing a program containing more than one 2048 × 2048 byte variable array resulted in significant deterioration in performance of the system due to increased page faults. With a system having more memory, larger data structures could be used to greatly enhance performance.

References

[1] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," *IBM J. Res. Develop.*, vol. 6, pp. 642–656, Nov. 1982.

[2] R. N. Slater, "Automating data base capture for CAD/CAM," *Comput. Graphics World*, vol. 48, pp. 45–53, Oct. 1984.

[3] M. Karima, K. S. Sadhal, and T. O. McNeil, "From paper drawings to computer aided design," *IEEE Comput. Graphics Applications*, vol. 5, pp. 24–39, Feb. 1985.

[4] C. L. Huang and J. T. Tou, "Knowledge based functional symbol understanding in electronic circuit diagram interpretation," *Aplications of Artificial Intell. III, Proc. SPIE*, vol. 635, pp. 288–299, 1986.

[5] H. Bunke, "Automatic interpretation of lines and text in circuit diagrams," in *Pattern Recognition Theory and Applications*, J. Kittler, K. S. Fu, and L. F. Pau Eds. Boston, MA: D. Reidel, 1982, pp. 297-310.

[6] L. T. Watson, K. Arvind, A. W. Ehrich, and R. M. Haralick, "Extraction of lines and regions from grey tone line drawing images," *Pattern Recognition*, vol. 17, pp. 493-506, 1984.

[7] W. H. Chen, W. K. Pratt, E. R. Hamilton, R. H. Wallis, and P. J. Capitant, "Combined symbol matching facsimile data compression system," *Proc. IEEE*, vol. 68, pp. 786-796, 1980.

[8] F. M. Wahl, M. K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Comput. Vision, Graphics, Image Processing*, vol. 20, pp. 375-390, 1982.

[9] H. Bley, "Segmentation and preprocessing of electrical schematics using picture graphs," *Comput. Vision, Graphics, Image Processing*, vol. 28, pp. 271-288, 1984.

[10] L. A. Fletcher, "Text string separation from mixed text/graphics images," M.S. thesis, Dep. Elec. Eng., Pennsylvania State Univ., Aug. 1986.

[11] A. Rosenfeld and A. C. Kak, *Digital Picture processing*, vol. 2, 2nd ed. New York: Academic, 1982.

[12] J. P. Foith, C. Eisenbarth, E. Enderle, H. Geisselmann, H. Ringshauser, and G. Zimmermann, "Real-time processing of binary images for industrial applications," in *Digital Image Processing Systems*, L. Bolc and Z. Kulpa, Eds. Berlin: Springer-Verlag, 1981.

[13] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1978, pp. 523-525.

Lloyd Alan Fletcher was born in Birmingham, England, in 1962. He received the B.S. degree in physics with microelectronics and computing from the University of Leicester, England, in 1984. From August 1984 until August 1986 he attended the Pennsylvania State University, where he was a teaching and research assistant, receiving the M.S. degree in electrical engineering in 1986. At Penn State his research interests included computer vision and digital image analysis.

Since September 1986 he has been a Member of Technical Staff with Bell Communications Research, Red Bank, NJ. He is currently working in the Switching Analysis and Reliability Technology Center, where he is involved with the development of criteria to assure the reliability and quality of digital switching systems and telecommunications equipment deployed by the regional Bell telephone companies.

Rangachar Kasturi (M'82) was born in Bangalore, India, in 1949. He received the B.E. degree in electrical engineering from Bangalore University in 1968 and the M.S.E.E. and Ph.D. degrees from Texas Tech University, Lubbock, in 1980 and 1982, respectively.

Dr. Kasturi is an Associate Professor of Electrical Engineering at the Pennsylvania State University, where he was an Assistant Professor from 1982 to 1986. From 1978 to 1982 he was a Research Assistant at Texas Tech University and was engaged in research in multiplex holography and digital image processing. From 1976 to 1978 he was the Engineering Officer at the Visvesvaraya Industrial and Technological Museum, Bangalore, and from 1969 to 1976 he was with the Bharat Electronics Ltd., Bangalore, as a Research and Development Engineer. His current research interests are in the applications of image analysis and artificial intelligence techniques for map data processing, graphics recognition, and document analysis. He has directed several projects sponsored by NSF, Digital Equipment Corporation, AT&T, and the Applied Research Laboratory. He has published a number of papers in journals and conference proceedings. He is the author of a book chapter and is coeditor of a book on image analysis applications to be published by Marcel Dekker.

Dr. Kasturi is a member of OSA, SPIE, Eta Kappa Nu, and Sigma Xi.