DOCUMENT ZONE HIERARCHY AND CLASSIFICATION

ROBERT M. HARALICK
SU CHEN
JAEKYU HA
Electrical Engineering FT-10
University of Washington
Seattle, WA 98115
haralick@ee.washington.edu

IHSIN PHILLIPS

Computer Science and
Software Engineering
Seattle University
Seattle WA 98122
yun@george.ee.washington.edu

ABSTRACT

The purpose of zone delineation is to define the regions which cover the page such that each region, excluding its holes, (the regions which it contains), is entirely a text region or a non-text region and such that in each text region no text line is fragmented. The Zone Hierarchy determination hierarchically subdivides each text zone into text lines and each text line into word, and each word into characters.

The purpose of zone classification is to identify each delineated zone into one of the classes: text or non-text. Non-text regions may be further classified into line drawing, graph, half-tone, etc. Zones which have been classified as text regions can then be given to an OCR system to produce computer readable text for the zone.

We have done zone hierarchy determination from perfect word-processor created images and from real images with a variety of approaches. One approach uses a recursively computed morphological closing transform. Another does zone hierachy determination by a grouping of the bounding boxes of the connected components of the black pixels. Zone classification is accomplished on the basis of first and second line length moments taken in four directions of the black and the white pixels in each zone. On a database of over 12,000 zones. Text misdetection rate was 3% and non-text misdetection rate was 24%.

1. Introduction

Zone delineation consists of determining the geometric page layout of a document image page. This means a specification of the geometry of the maximal homogeneous regions and the spatial relations of these regions. A region is homogeneous if all its area is of one type: text, half-tone, figure etc. and each text line of the page lies entirely within some text region of the layout. A Manhattan page layout is one where the regions of the page layout are all rectangular and the rectangles are in the same orientation. Hence after an appropriate page rotation the sides of the rectangles will

all be either horizontal or vertical. Furthermore, each pair of rectangles either is mutually exclusive or one contains the other.

The purpose of zone delineation is to define the regions which cover the page such that each region, excluding its holes, (the regions which it contains), is entirely a text region or a non-text region and such that in each text region no text line is fragmented. The purpose of zone classification is to identify each delineated zone into one of the classes: text or non-text. Non-text regions may be further classified into line drawing, graph, half-tone, etc. Zones which have been classified as text regions can then be given to an OCR system to produce computer readable text for the zone. The purpose of zone hierarchy determination is to divide each text zone into its lines and each text line into its words and each word into its characters. The hierarchy can be thought of as a tree. The root node represents the whole document page. The nodes in succeeding levels represent zones, text lines, words and characters, respectively. Each entity is specified by the coordinates of its bounding box.

The next section is a brief literature review. In section 3 we discuss a connected component bounding box algorithm on the document images to determine the zone hierarchy. Section 4 discusses the determination of zone hierarchy by augmenting the ground truth files in the UW English Document Image Database I[15]. Section 5 the recursive closing transform alogorithm which works on document images.

2. Literature Review

Early work on zone delineation was done by Wahl et. al.[19]. They use a technique called the constrained run length smoothing algorithm. It actually consists of a morphologically closing of the document image with a horizontal structuring element of specified length (they used 300) intersected with a morphological closing of the document image with a vertical structuring element of specified length (they used 500). The intersection is then morphologically closed with a horizontal structuring element of specified length (they used 30). The bounding rectangles of the connected components of the resulting image constitute the block segments. Features of the blocks include the area of the connected component of the block, the number of black pixels in the block on the original document image. the mean horizontal black run lengths of the original image within the blocks,

Nagy and Seth[11], and Nagy et. al.[12] employ an $X-Y$ tree as the representation of a page layout. The root node of an $X-Y$ tree is the bounding rectangle of the full page. Each node in the tree represents a rectangle in the page. The children of a node are obtained by subdividing the rectangle of the parent node either horizontally or vertically, with horizontal and vertical cuts being alternately employed in successive levels in the tree. Hao et. al.[6] describe a variation on this technique.

Fisher et. al.[5] sample a 300dpi document image by a factor of 4 and use the run length smoothing algorithm. They then compute the connected components of the run length smoothed image. The connected components and their bounding boxes constitute the blocks of the geometric page layout. They extract connected component features such as component height, width, aspect ratio, density, perimeter, and area

for classifying each block as *text* or *non-text*.

Lebourgeois et.al.[10] sample the document image by a factor of 8 vertically and 3 horizontally. Each pixel on the sampled image corresponds to an 8x3 window on the original image. If any pixel on the 8x3 window of the original image is a binary one then the sampled image has a binary one in the corresponding pixel position. Then the sampled image is dilated by a horizontal structuring element to effectively smear adjacent characters into one another. Each connected component is then characterized by its bounding rectangle and the mean horizontal length of the black runs. Connected components having a vertical height within given bounds and mean horizontal run length within given bounds are then labeled as a text lines and outside the given bounds are labeled as a non-text lines. Components assigned as text regions are then vertically merged into larger blocks using rules taking into account alignment. Blocks are also subdivided to separate their horizontal peninsulas. No measure of performance is given but an indication that the method needs improvement was stated.

Saitoh and Pavlidis[16] proceed by sampling 8 vertically and 4 horizontally and then extracting the connected components. They then classify each component into *text, text or noise, diagram or table, halftone image, horizontal separator*, or *vertical separator*, using block attributes such as block height, height to width ratio, and connectivity features of the line adjacency graph, and whether there are vertical or horizontal rulings. Page rotation skew is estimated from a least squares line fit to the the center points of blobs belonging to the same block. Blocks are subdivided based on the vertical distance between lines in a block, and the height of the lines in a block. The technique was tried on 52 Japanese documents and 21 English documents. No quantitative measure of performance was given.

Pavlidis and Zhou[14] determine the geometric page layout by analyzing the white areas of a page by computing the vertical projection and looking for long white intervals from the projections. Then the column intervals are converted into column blocks, merging small blocks into larger blocks. Blocks are clustered according to their alignments and the rotation angle estimated for each cluster. The column blocks are then outlined. Finally, each block is labeled as text or non-text using features such as ratio of the mean length of black intervals to the mean length of white intervals, the number of black intervals over a certain length, and the total number of intervals. No performance results are given.

Baird[3] discusses a computational geometry technique for geometric page layout by finding the maximal rectangles covering the white ares of the page. The rectanglar regions not covered by the maximal sized white rectangles then constitute regions which can then be classified as text or non-text.

Amamoto et. al.[2] determine the geometric page layout by operating on the white space of the sampled document image. They open the white space of the sampled document image with a long horizontal structuring element and then open it with a long vertical structuring element. The union of these two openings then constitute the white space of the blocks. The blocks are then extracted from this white space. They decide that a block is a text block if the length of the longest black run length in

the vertical and horizontal directions is smaller than a given threshold. A decision is made whether the writing is horizontal or vertical based on the number N_H of blocks whose width is greater than twice its height and the number N_V of blocks whose height is greater than twice its width. If $N_H > N_V$, then the decision is horizontal writing. Else vertical writing. Each block is then assigned a class label from the set: *text, figure, image, table*, and *separation line*. No performance results are given.

O'Gorman[13] discusses what he calls the docstrum technique for determining geometric page layout. This technique involves computing the k-nearest neighbors for each of the black connected components of the page. Each pair of nearest neighbors has an associated distance and angle. By cluster the components using the distance and angle features, the geometric regions of a page layout can be determined.

Hirayama[8] develops a technique for determining the geometric layout structure of a document which begins by merging character strings into text groups. Border lines of blocks are determined by linking edges of text groups. Then blocks which have been oversegmented are merged and a projection profile method is applied to the resulting blocks to differentiate text areas from figure areas. Hirayama reports that on a data set of 61 pages of Japanese technical papers and magazines 93.3% of the text areas and 93.2% of the figure areas were correctly detected.

Ittner and Baird[9] determine a geometric layout by doing skew and shear angle corrections, partitioning the page into blocks of text, inferring the text line orientation within each block, partitioning each block into text lines, isolating symbols within each text line, and finally merging the symbols into words. Blocks are determined by the white space covering technique of Baird[3]. They report that on 100 English document image pages from 13 publishers and 22 styles, 94% of the layouts were correctly determined. The orientation of text lines in a block is determined from the minimum spanning tree of the connected components of the black pixels. The mode of the histogram of the directions of the edges in the minimum spanning tree is the orientation of the text lines in a block. Symbols in a text line are determined by taking the projection in an orthogonal direction to the text line. The projection profile is checked for a dominant frequency and the segmentation into characters is done from the projection profile with the knowledge of the dominant frequency. To determine the words in a text line, they determine a scalable word-space threshold for each text block separately. Then each text line is independently segmented to distinguish the inter-character spacing with the inter-word spacing.

Tsujimoto and Asada[18] assume that each block of the geometric page layout contains exactly one logical class. They organize the geometric page layout as a tree. Each new article in a document such as a newspaper begins with a headline which is in the head block. They find the paragraphs which belong to the head block by rules relating to the order of the geometric page layout tree and are able to assign logical structure labels of *title, abstract, sub-title, paragraph, header, footer, page number*, and *caption*. They worked on 106 document images and correctly determined the logical structure for 94 document images.

Yamashita et. al.[20] use a model-based method. Character strings, lines, and half-tone images are extracted from the document image. Vertical and horizontal

field separators (long white areas or black lines) are detected based on the extracted elements, then appropriate labels are assigned to character strings by a relaxation method. Label classes included: *header, title, author, affiliation, abstract, body, page number, column, footnote, block* and *figure*. The technique was applied to 77 front pages of Japanese patent applications. They reported that the logical structure for 59 were determined perfectly.

Saitoh et. al.[17] determine logical layout with text block labels of *body, header, footer*, and *caption*. They tested the technique on 393 document images of mainly Japanese and some English documents. To characterize performance they measured the average number of times per document image an operator has to correct the results of the automatically produced layout. They report that on the average 2.17 times per image areas not suitable for output have to be discarded, .01 times per image mis-classified areas have to be correctly labeled, and 1.09 times per image does a text area have to be reset. With respect to text ordering they report that it required moving connections .47 times per image, on the average, making new connections .11 times per image, and re-assigning type of text .36 times per image.

3. The Bounding Box Algorithm

The input to the bounding box algorithm is a deskewed binary document image. A connected component algorithm is applied to the black pixels of the deskewed binary image. For each connected component, the smallest rectangular box which circumscribes the connected component is computed. Each of the bounding boxes is considered as a single unit on the page. The algorithm does the horizontal and vertical projections of these bounding boxes. The projection profiles are analyzed and the textlines are extracted. The spatial configuration of bounding boxes within each of the extracted textlines is analyzed to extract words. Textlines are merged into text-blocks based upon the inter-textline spacing distribution and also based on the changes in the textline justification. The details are as follows.

3.1. Step 1: Bounding Boxes of Connected Components

Let I be the input binary image. A connected-component algorithm is applied to the black pixels in I to produce a set of connected components. Then, for each of the connected components, the smallest rectangular box which circumscribes the connected component is computed. Such a rectangular box is called the bounding box. A bounding box can be represented by giving the coordinates of the upper left and the lower right corners of the box. Figure 1 shows an example of the bounding box of the character 'g'.

Many kinds of symbols are used in document pages: alphanumeric characters, punctuation marks, mathematical symbols and so on. These symbols can be classified into two groups: "single-component" symbols and "multiple-component" symbols. Character 'g' is an example of single-component symbol, while character 'i' is an example of multiple-component symbol. All English alphabet characters except 'i'

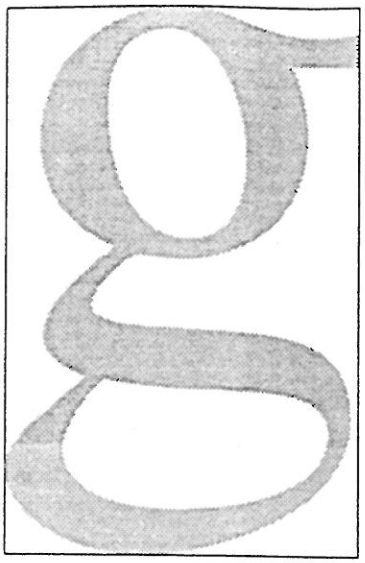


Figure 1: Character 'g' and its bounding box

and 'j' belong to the former group. Of course, a single-component symbol can be "broken" into several pieces on a degraded image. In this case, the symbol is also considered as a collection of subsymbols, that is, a multiple-component symbol.

Figure 3(a) shows a segment of an English document image (taken from the UW English Document Image Database-I, page id "L006SYN.TIF") and Figure 3(b) shows the bounding boxes produced by the algorithm in this step. Note that, the number of bounding boxes are larger than the number of symbols since multiple bounding boxes are produced for multi-component symbols. The bounding box page decomposition scheme analyzes the spatial configuration of those bounding boxes, by means of their projection profiles, to extract the textlines, words, and paragraphs.

3.2. Step 2: Projections of Bounding Boxes

Analysis of the spatial configuration of bounding boxes can be done by projecting those bounding boxes onto a straight line. Since paper documents are usually written in the horizontal or vertical direction, projections of bounding boxes onto the vertical and horizontal lines are of particular interest. While projecting bounding boxes onto the horizontal or vertical line, they will accumulate onto that line, which results in the projection profile as shown in Figure 2.

A projection profile is a frequency distribution of the projected bounding boxes on the projection line. The bounding box projection profiles provide important information about the number of bounding boxes aligned along the projection direction. Figure 3(c) and 3(d) show the horizontal and vertical projection profiles of the bounding boxes in Figure 3(b).

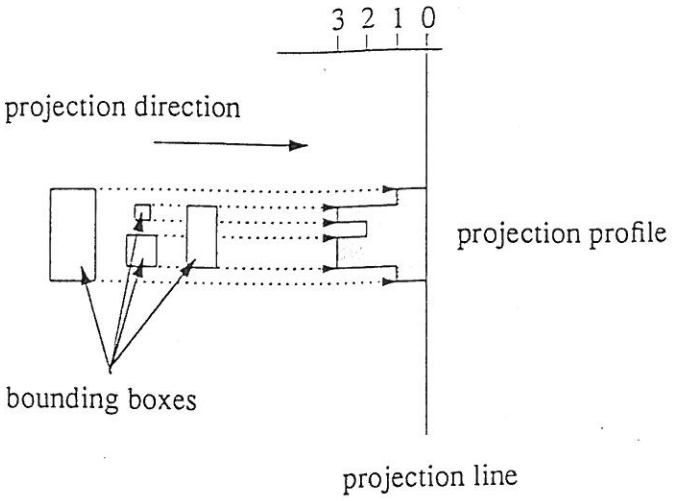


Figure 2: Horizontal Projection Profile, which is obtained by accumulating the bounding boxes onto the vertical line. It is a frequency histogram of the number of projected bounding boxes

3.3. Step 3: Extraction of Text-Lines

In this step, the bounding box algorithm first determines the textline direction of the page by analyzing both horizontal and vertical projection profiles. Once the text-line direction of the page is determined, the algorithm partitions the page bounding box into textline bounding boxes.

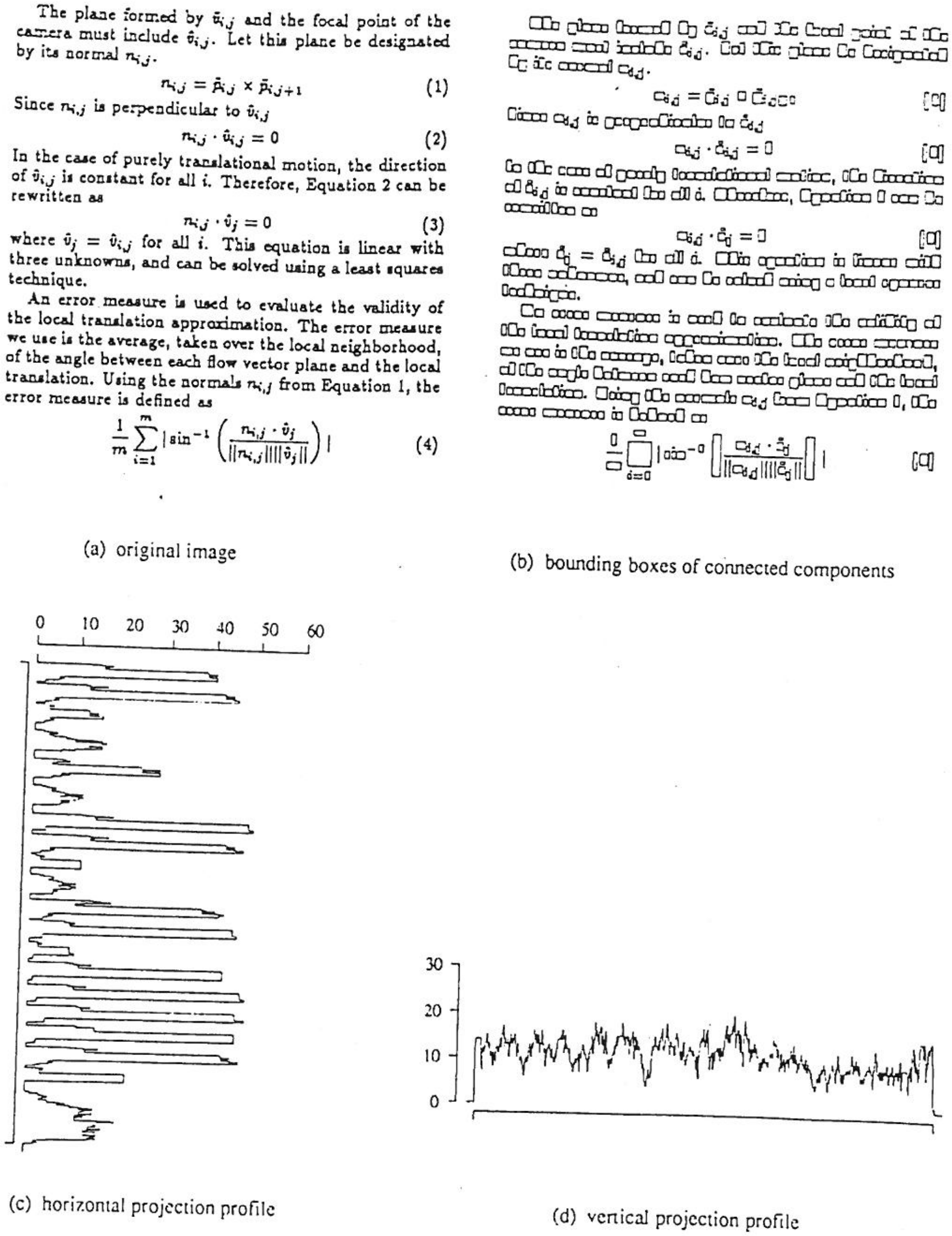
If we take a careful look at the projection profiles in Figures 3(c) and 3(d), it is clear that the text-lines are horizontally oriented: Within the horizontal projection profile, there are distinct high peaks and deep valleys at somewhat regular intervals, whereas within the vertical projection profile, there is no such distinction. Since the bounding boxes are represented by a list of coordinates of the two-points, the text-lines are easily extracted. The result is shown in Figure 3(e).

Other important information can be deduced from the horizontal projection profile: the frequency distribution of textline heights and inter-textline spacings (Figure 3(f) and 3(g)). These distributions will be employed in the text-block extraction process.

3.4. Step 4: Extraction of Words

In this step, the algorithm groups the bounding boxes on each text-line (produced from the last step) into bounding boxes of words.

Our algorithm first computes the projection profiles within each of the textline



The plane formed by $\bar{u}_{i,j}$ and the focal point of the camera must include $\hat{v}_{i,j}$. Let this plane be designated by its normal $n_{i,j}$.

$$n_{i,j} = \bar{p}_{i,j} \times \bar{p}_{i,j+1} \quad (1)$$

Since $n_{i,j}$ is perpendicular to $\hat{v}_{i,j}$

$$n_{i,j} \cdot \hat{u}_{i,j} = 0 \quad (2)$$

In the case of purely translational motion, the direction of $\hat{v}_{i,j}$ is constant for all i. Therefore, Equation 2 can be rewritten as

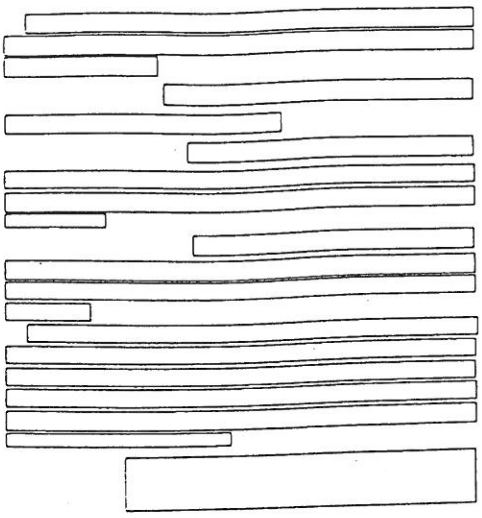
$$n_{i,j} \cdot \hat{v}_j = 0 \quad (3)$$

where $\hat{v}_j = \hat{v}_{i,j}$ for all i. This equation is linear with three unknowns, and can be solved using a least squares technique.

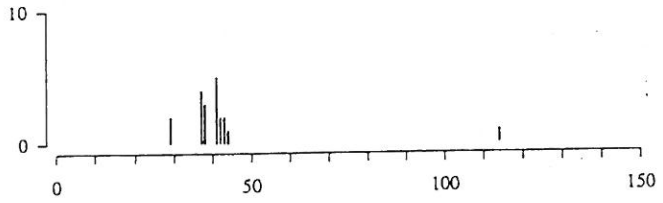
An error measure is used to evaluate the validity of the local translation approximation. The error measure we use is the average, taken over the local neighborhood, of the angle between each flow vector plane and the local translation. Using the normals $n_{i,j}$ from Equation 1, the error measure is defined as

$$\frac{1}{m}\sum_{i=1}^{m} |\sin^{-1}\left(\frac{n_{i,j} \cdot \hat{v}_j}{||n_{i,j}|| ||\hat{v}_j||}\right)| \quad (4)$$

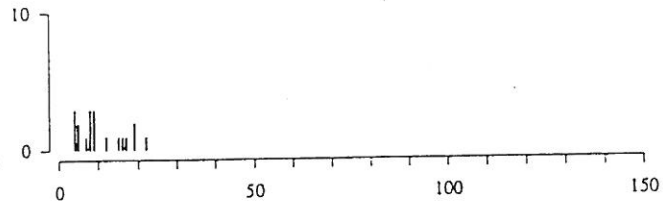
Figure 3: Bounding Boxes of Connected Components and the Projection Profiles: (a) a document image written in English, (b) bounding boxes of connected components, (c) horizontal projection profile, (d) vertical projection profile



(e) textline bounding boxes



(f) textline height distribution



(g) inter-textline spacing distribution

Figure 3 (continued): (e) text-line bounding boxes, (f) text-line height distribution, (g) inter-text-line spacing distribution. The abscissa represents the heights of text lines/inter-text-line spacings in pixel unit and the ordinate represents the number of occurrences

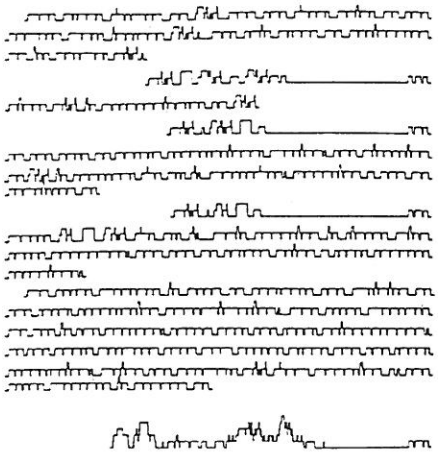
bounding boxes. Again, the units of the projection profile are bounding boxes, not pixels. Next, the algorithm considers each of the projection profiles as a one-dimensional *gray-scale image*, and threhsolds each of the images to produce a binary image. The threshold value is set to 1. Figure 3(h) shows projection profiles within textlines and Figure 3(i) shows the corresponding binarized 1-D profiles. Note that, during the binarization, a symbol (or a broken symbol) with multiple bounding boxes may be merged into one, as well as, those adjacent symbols within the same text-line whose bounding boxes are overlapping with each other. But this will not cause any problem in the result of our word extraction process, since our algorithm extracts words by merging symbols' bounding boxes to form words.

After the binarization, the algorithm performs a morphological closing operation on each of the binarized text-line projection profile with structuring element of appropriate size. The length of the structuring element is determined by analyzing the distribution of the run-length of the zero's on the binarized text-line projection profile. In general, such a run-length distribution is bi-modal. One mode represents the inter-symbol spacing, and the other represents the inter-word spacing. Figure 3(j) shows the frequency distribution of inter-character spacings, The length of the structuring element is chosen between the two modes. As the result, the morphological closing operation closes spaces between symbols, but not the spaces between words. (See Figure 3(i) and 3(k)). Figure 3(l) shows the results of this step.

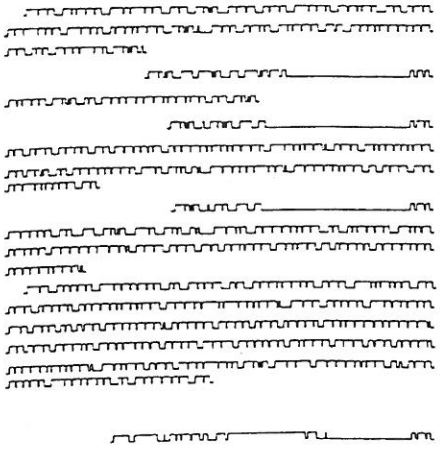
In a printing process, each symbol is associated with its width, the amount of space it occupies when it appears in a text-line. In some fonts this spacing is constant, i.e., it does not vary from character to character. These fonts are called fixed pitch or monospaced fonts; they are used mainly for typewrite-style printing. Most fonts used for high quality typography, however, associate a different width with each character. Such fonts are called variable pitch fonts. In either case, most of alphabetic languages are usually printed in such a way that less spacing is put between consecutive characters in a word while more spacing is put between neighboring words on a text line. This fact can easily be observed from the vertical projection profile for each text line (Figure 3(h)).

Figure 3(j) shows the frequency distribution of inter-character spacings, which are from the example text shown in Figure 3(a). As is expected from the fact that there is more spacing between words than between characters, the inter-character spacing distribution is usually bimodal: The mode on the left side corresponds to the inter-character spacings and the mode on the right side to the inter-word spacings. In fixed pitch typefaces such as the Courier typeface the separation of two modes is outstanding while it is not so in variable pitch typefaces such as the "Times Roman" typeface. Anyway, some value between the two modes will function as a threshold value for word extraction.

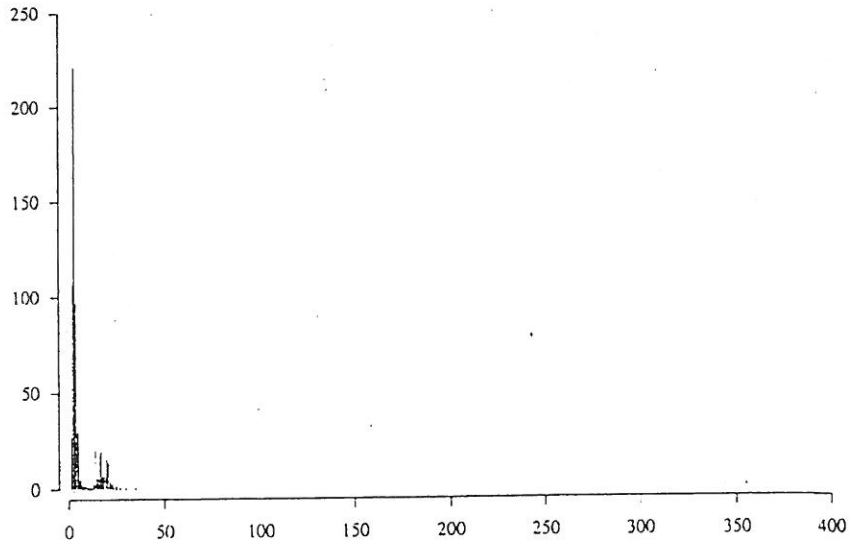
Characters may be kerned to improve their visual spacing. Kerning is a modification to inter-character spacing of text. Kerning is applied with proportionally spaced typefaces for purely visual reasons: either because it looks better, or because it improves the legibility of the text. When kerning occurs, the bounding boxes of the characters to which kerning is applied will overlap, which is reflected on the vertical



(h) vertical projection profiles

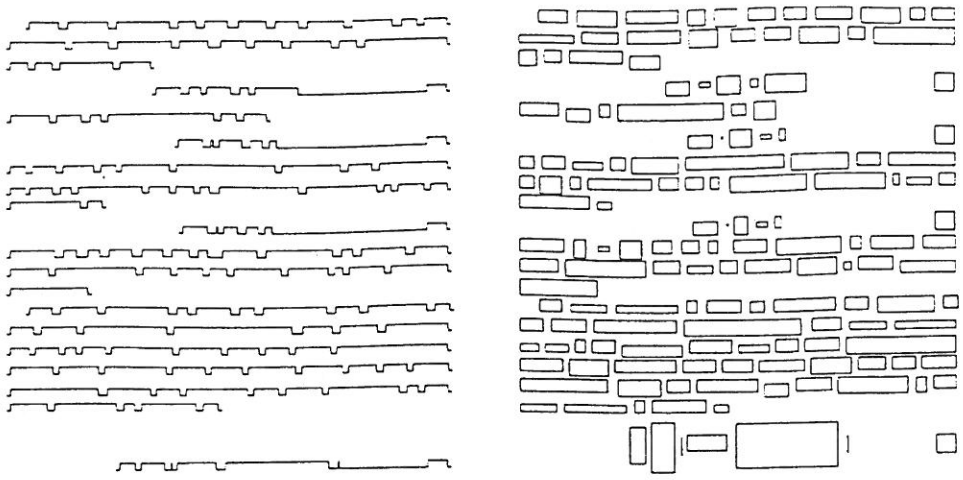


(i) binarized profile



(j) inter-character spacing distribution

Figure 3 (continued): (h) vertical projection profiles, (i) binarized vertical projection profiles, (j) inter-character spacing distribution



(k) closed binary vertical profiles (l) word bounding boxes

Figure 3 (continued): (k) vertical projection profiles, (l) binarized vertical projection profiles

projection profile (Figure 3(h)).

Sometimes, characters are *decorated* or *accented* as can be seen, for example, in *é*. Such an accent is usually found in a variety of foreign languages as well as in mathematical expressions. Accented symbols are composed of multiple components whose bounding boxes are aligned in the vertical direction. Also, accentuation is reflected on the vertical projection profile at a text line.

3.5. Step 5: Extraction of Paragraphs

In this step, the algorithm groups the bounding boxes of the extracted text-lines into bounding boxes of text-blocks.

There are four basic types of text line layout that are in common use: centered, flush left, flush right, and justified. These are also frequently referred to as centered, left-justified, right-justified, and fully-justified.

Though a paragraph is a unit in the logical hierarchy, its physical appearance is noticeable in a document image. Whatever justification has been used in the preparation of a document, paragraphs are usually made either by changing the justification of the current text line or by putting more space between two text lines, one of which is from the previous paragraph and the other of which from the current one. In the former case, one might usually indent the first line of a paragraph of text.

Extraction of paragraphs should be primarily based on the above two basic paragraph-breaking methods. When a significant change in text-line heights or in inter-text-line spacings occurs, we might say that a new paragraph begins. The distributions of text-line heights and inter-text-line spacings (Figure 3(f) and 3(g)) together with the horizontal projection profile (Figure 3(c)) give us a clue for this kind of paragraph breaking. If there is a change in the text-line justification, we might say that a new type of text block begins.

Let us take a look at Figure 3(a). The fourth line is a displayed math zone, which is right-justified in this example, and in the fourteenth text line, some amount of indentation was placed, and therefore, a new paragraph begins, and so on. Figure 3(m) shows the text block bounding boxes for our example text, which can be obtained merely by taking into consideration the above two basic text block breaking methods. Figures 3(n)- 3(p) show the bounding boxes obtained as above superimposed on the original image.

3.6. Summary For Bounding Box Algorithm

The bounding box algorithm is just at the stage where we are ready to begin evaluating it on a large database. This will be done within the next few months. From the results we have illustrated we can expect almost perfect results when the quality of the input image is high. How much it degrades as the input image degrades is the crucial question to be answered by the evaluation.

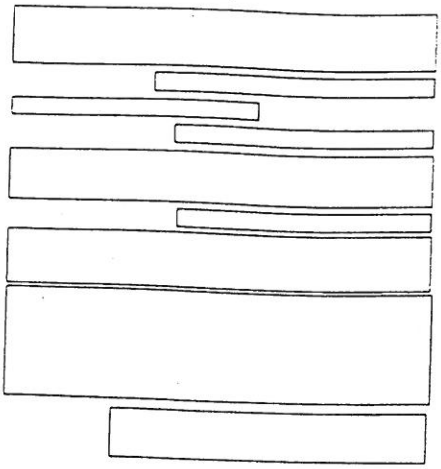
4. Page Decomposition Using the Recursive Closing Transform

In this section we discuss the recursive closing transform approach to zone hierarchy determination from document image pages. The recursive closing transform is a powerful morphological tool for image background shape analysis. For it provides an efficient way of computing the binary morphological closings with respect to any sized structuring element. It is especially useful in areas where the choice of the size of the structuring element needs to be determined after a morphological examination of the content of the image. In the following, we will give a short overview of the transform[4].

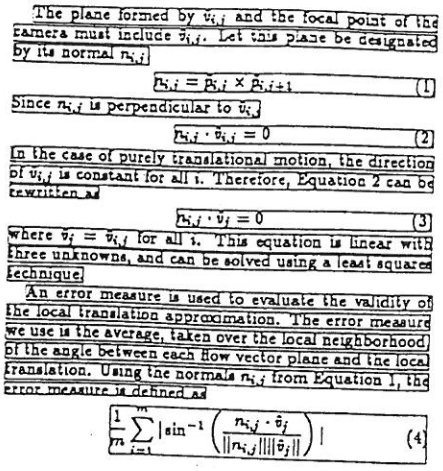
4.1. Recursive Closing Transform: A Review

The closing transform of a set I with respect to a structuring element K generates a grayscale image where the gray level of each pixel $x \in Z^2$ is defined as the smallest positive integer n so that $x \in I \bullet (\ominus_{n-1} K)$. If no such n exists, where $x \notin I \bullet (\oplus_{n-1} K)$ for all n, then the closing transform at $x \in Z^2$ is designated as zero.

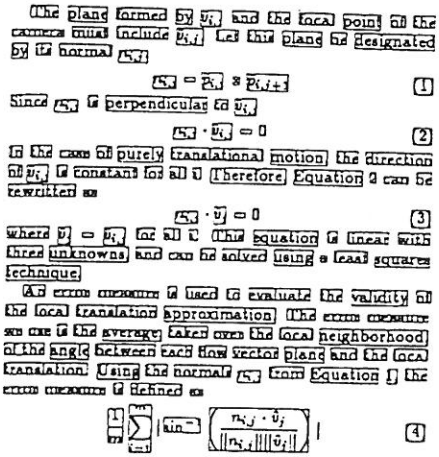
Definition 1 *The* closing transform *of a set* $I \subseteq Z^2$ *by a structuring element*



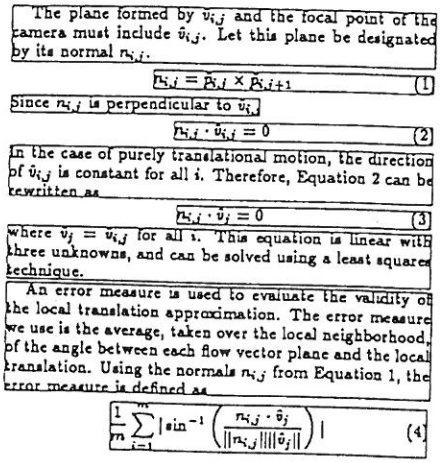
Text block bounding boxes



Text Lines



Words



Paragraphs

Figure 3 (continued): (m) text block bounding boxes, (n) bounding boxes and the original image, (o) text word bounding boxes and the original image, (p) text block bounding boxes and the original image

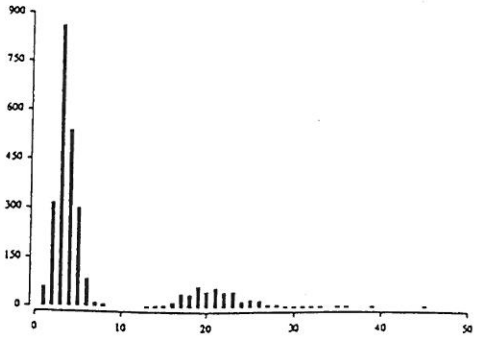


Figure 4: Inter-character Spacing Distribution for a 12-pt font size document: The abscissa ,measured in the pixel unit, represents the horizontal distance between the bounding boxes of two adjacent connected components in a text-line and the ordinate denotes the number of occurrences. The image resolution is 300 dpi and the number of characters used in the document image is 2935

$K \subseteq Z^2$ is denoted by $CT[I, K]$ and is defined as:

$$CT[I,K](x) = \begin{cases} min\{n \mid x \in I \bullet (\oplus_{n-1} K)\} & if\ \exists n, x \in I \bullet (\oplus_{n-1} K) \\ 0 & if\ \forall n, x \notin I \bullet (\oplus_{n-1} K). \end{cases}$$

In [4], an efficient recursive algorithms, namely the recursive closing transform (RCT), is developed to compute in constant time per pixel the closing transforms.

4.2. Algorithm Description

The algorithm relies on the recursive closing transform to extract image background (white-space) informations. Let $K_1, K_2, \cdots, K_n$ denote n structuring elements. Let $y_1 = CT[I, K_1](x), y_2 = CT[I, K_2](x), \cdots, y_n = CT[I, K_n](x)$ denote the values of the closing transform at pixel $x \in I$ with respect to the structuring elements $K_1, K_2, \cdots, K_n$. Then each pixel in the image I is model as a random observation data vector $\mathcal{Y} = (y_1, y_2, \cdots, y_n)$. Furthermore, each pixel has an associated label $C = c$. The page segmentation algorithm assigns a label to each pixel to maximize the posterior probability:

$$P[C = c \mid \mathcal{Y} = (y_1, y_2, \cdots, y_n)] = \frac{P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = c] P[C = c]}{P[\mathcal{Y} = (y_1, y_2, \cdots, y_n)]}.$$

The same Bayesian formulation is used for both the text line and the word segmentations except that the maximum likelihood probability distribution $P[\mathcal{Y} =$

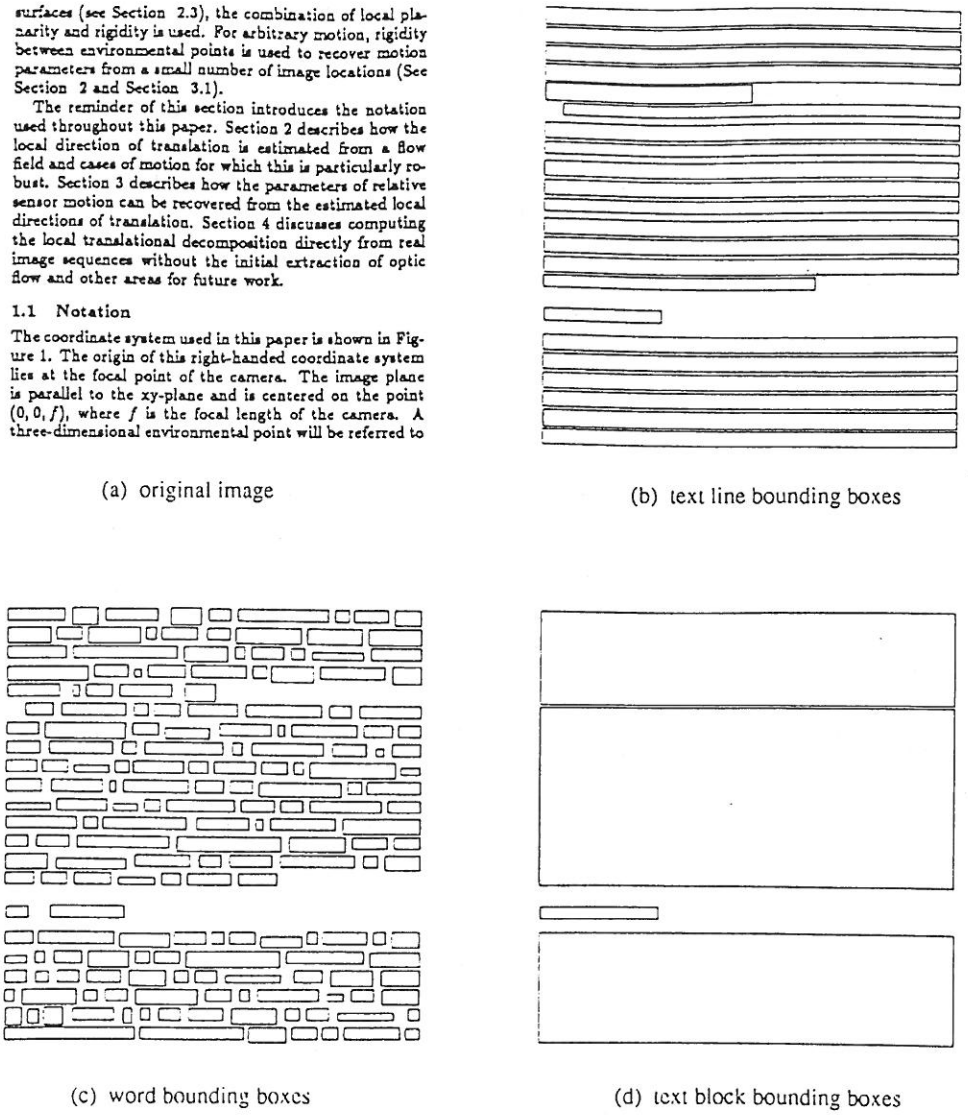


Figure 5: Example of Document Image Decomposition: (a) a document image written in English, (b) text line bounding boxes, (c) word bounding boxes, (d) text block bounding boxes

$(y_1, y_2, \cdots, y_n) \mid C = c]$ and prior probability distribution $P[C = c]$ are computed differently. In the initial experimental stage, we assume uniform prior probability distribution of class labels. In this case, the above Bayesian formulation reduces to the maximum likelihood formulations.

4.2.1. Text Line Extraction

To perform text line extraction, we generate a binary text line mask image. A pixel in the text line mask image has a binary one value if and only if it is part of a text line, i.e. lies within a text line bounding box. Otherwise, it has a binary zero value.

In the algorithm training stage, our current system adopts a rather brute force algorithm to to compute the empirical maximum likelihood probability distribution for text lines. We directly calculate the frequency counts of $P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = 1]$ and $P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = 0]$ given the set of training ground-truthed document images that we generated in Section . In our experiment, we select $n = 3$ and K_1 is the horizontal 1×2 structuring element, K_2 is the vertical 2×1 structuring element, K_3 is the 2×2 square structuring element. The value of $\mathcal{Y} = (y_1, y_2, \cdots, y_n)$ is bounded within the hyper-cube $[1, N] \times [1, N] \times [1, N]$, where N is the maximum integer value of the closing transform.

A pixel is classified as a text line pixel if and only if $P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = 1] > P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = 0]$.

4.2.2. Text Word Extraction

Similarly, to perform text word extraction, we generate a binary text word mask image. A pixel in the text word mask image has a binary one value if and only if it is part of a text word, i.e. lies within a text word bounding box. Otherwise, it has a binary zero value.

To characterize each pixel, the same brute force algorithm is used to compute the empirical maximum likelihood probability distribution for text words.

A pixel is classified as a word pixel if and only if $P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = 1] > P[\mathcal{Y} = (y_1, y_2, \cdots, y_n) \mid C = 0]$.

4.3. Experimental Results

To evaluate the recursive transform algorithm, we trained the algorithm on the 168 noise-free document images to obtain the empirical maximum likelihood probability distributions for text line and word segmentations.

Figure 6 shows an example noise-free synthetic document image. Figure 7 illustrates the extracted text line and word images using the recursive transform algorithm. Figure 8 plots the calculated text line and word bounding boxes.

Although we have not yet done complete evaluation of the algorithm we have tried it on a few real document images. Figure 9 illustrates an example real document

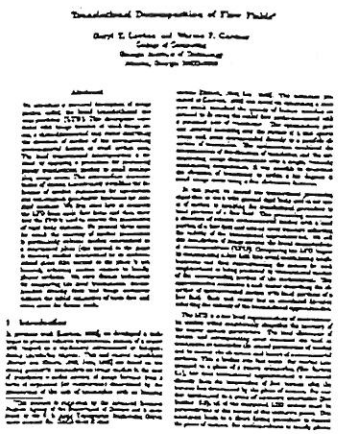


Figure 6: Illustrates an example synthetic document page

image. The image is contaminated with noise. Figure 10 shows the text line and word images. Figure 11 shows the text line and word bounding boxes.

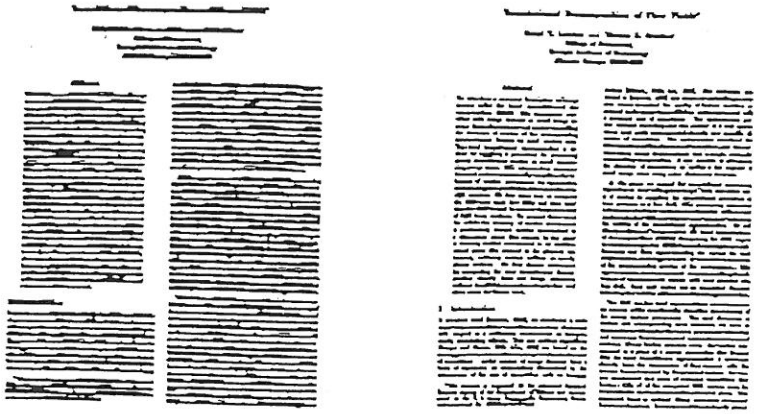
5. Zone Hierarchy Ground Truth Annotation

The previous sections discussed algorithms which began with document images and produced the zone hierarchy. In this section we discuss an approach to determine the zone hierarchy using only existing ground truth annotation files associated with document images created from word processor system files. The purpose here is for developing accurate ground truth zone hierarchies to aid in the use of large training sets for algorithms which operate directly on the document images such as those discussed in sections 3 and 4.

The "UW English Document Image Database (I)" [15] is a data set for the optical character recognition (OCR) and the document image understanding algorithm developments and evaluations. The database has software to convert a DVI file from the LaTeX document processing system into bitmap images. The program also generates a so-called character ground truth file for each document image. The file contains the bounding box coordinates, the font type and size, and the ASCII code for each individual character in the images.

In addition, the database provides a total of 168 such synthetically generated bitmap images and their associated character ground truth file. The document images are manually segmented into rectangular zones whose row and column coordinates are given in the database.

The algorithm discussed in this section takes the character ground truth file and



(a) (b)

Figure 7: Illustrates the segmented images: (a) text line image; (b) word image

the document page zone box delineations of a document page to create the zone zone hierarchy.

5.1. Notation and Assumptions

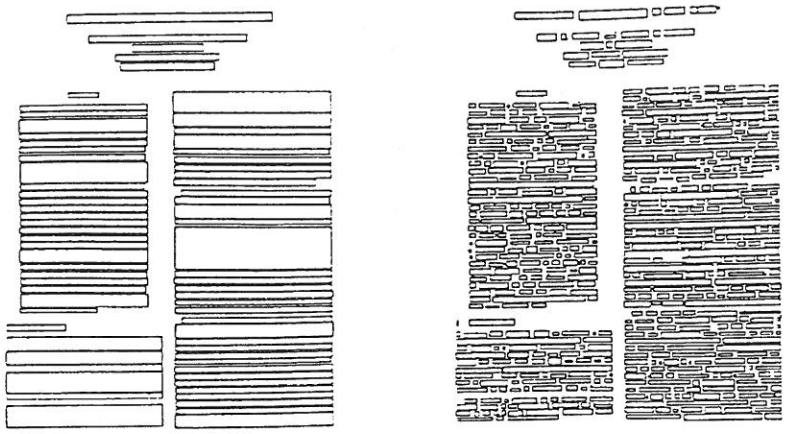
Let a document page be denoted as $\mathcal{P}$. Let $\mathcal{Z} = \{z_1, z_2, \cdots, z_k\}$ denote the set of zone of the document page $\mathcal{P}$. Let the character ground truth file be modeled as a sequence of character bounding boxes $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$.

Our assumption on the character bounding box sequence $\mathcal{C}$ is that it follows the same logical reading order, i.e. the correct character reading sequence are preserved. The characters are laid out in such a way that they first go sequentially from left to right and then at the end of a line, start at a different vertical or horizontal position. We assume that spacings between two adjacent characters in the $\mathcal{C}$ follow different distributions for the character breaks, the word breaks and the line breaks.

5.2. Algorithm

The following procedure describes the algorithm for extracting ground truth layout information from the character ground truth files.

1. Compute spacings between any two adjacent characters in the bounding box



(a) (b)

Figure 8: Illustrates the (a) Line bounding boxes; (b) Word bounding boxes

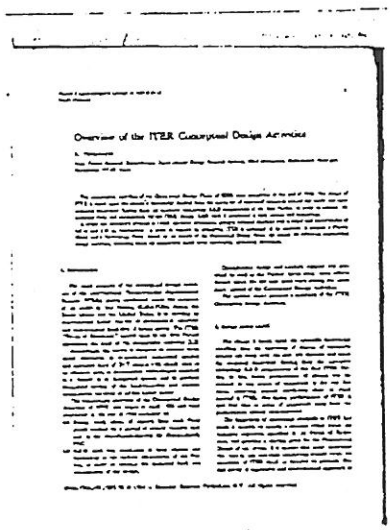
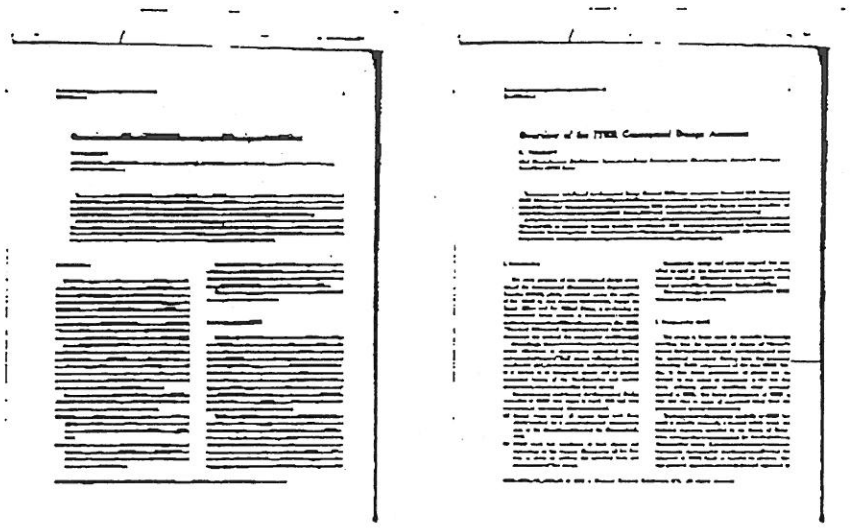
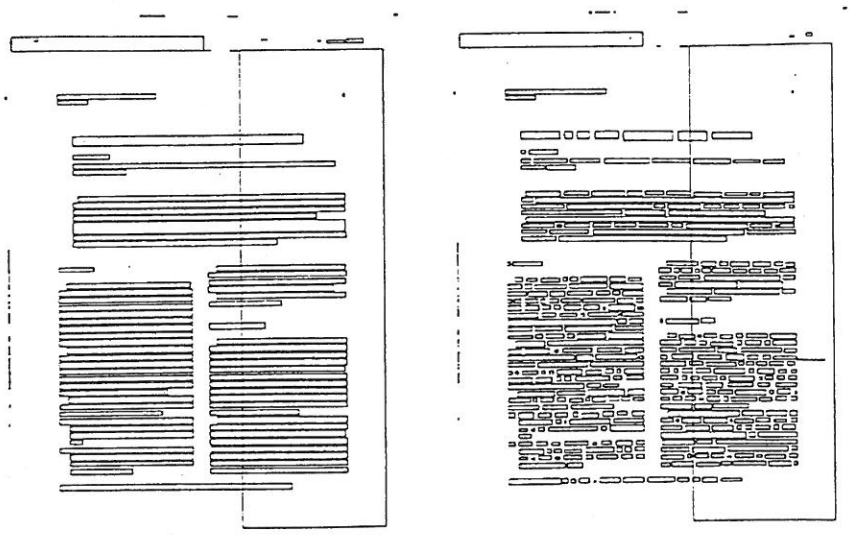


Figure 9: Illustrates an example real document page



(a) (b)

Figure 10: Illustrates the segmented images: (a) text line image; (b) word image



(a) (b)

Figure 11: Illustrates the (a) Line bounding boxes; (b) Word bounding boxes

sequence C. The distance measure is defined as follows:

$$\rho(c_i, c_{i+1}) = \rho_x(c_i, c_{i+1}) + w\rho_y(c_i, c_{i+1})$$

where $i = 1, 2, \cdots, n-1$. $\rho_x(c_i, c_{i+1})$ and $\rho_y(c_i, c_{i+1})$ are the minimum horizontal and vertical distance between the two bounding boxes, respectively. The $\rho_x(c_i, c_{i+1})$ is zero when c_i, c_{i+1} overlap horizontally. Likewise, the $\rho_y(c_i, c_{i+1})$ is zero when c_i, c_{i+1} overlap vertically. w is a weight.

2. Compute the histogram of the $\rho(c_i, c_{i+1})$. Typically, it contains three peaks: one for character breaks, one for word breaks and the other for line breaks. The first two peaks are relatively stronger.

3. Text line extraction: If $\rho(c_i, c_{i+1}) > T_2$, then the break between c_i and c_{i+1} is a line break. $T_2 = \alpha S$, where S is the dominant character font size and α is a constant and typically set at 10. The bounding box of a text line is calculated by finding the minimum bounding box that includes all the character bounding boxes within the two adjacent line breaks.

4. Text word extraction: If $\rho(c_i, c_{i+1}) \leq T_1$, then the break between c_i and c_{i+1} is a character break. If $T_1 < \rho(c_i, c_{i+1}) \leq T_2$, then the break between c_i and c_{i+1} is a word break. The bounding box of a word is calculated by finding the minimum bounding box that includes all the character bounding boxes within the two adjacent word breaks. We employ an automatic thresholding technique to estimate the threshold T_1 [7].

5. Find zone correspondence: Each text line and its associated word and character boxes are assigned to a unique zone z_j that has the maximum overlap with the text line bounding boxes. The zone bounding box are modified so that it is the minimum bounding box that encloses all the text lines assigned to the zone.

5.3. Experimental Results

We tested our algorithm on the 168 images from the "UW English Document Image Database (I)" [15]. The algorithm performed perfectly on all the images. Figure 12 illustrates an example document page image. Figure 13 illustrates its zone, text line, word and character level layout structures.

6. Zone Classification

We have begun to explore a technique for zone classification distinguishing text zones from non-text zones. Non-text zones include half-tones, graphs, tables, figures, drawings etc. Text zones include ordinary text areas and display mathematics.

For our initial experiments, we used the zone delineations given in the ground truth files of the UW English Document Image Database I. For each zone we computed

Figure 12: Illustrates an example document page

the first two runlength moments for the black pixels and the white pixels in the zone, taken in the horizontal direction, the vertical direction, and the two diagonal directions. The database has 979 document images plus a few more additional scans of the same images. There were 14,343 text zones of one sort or another and 1,616 non-text zones.

To determine the classification accuracy, we divided the data set in thirds, trained on two thirds and then tested on the last third. Then we repeated this for each of the three thirds. We used a classification tree for the decision rule. We observed that 13,842 text zones of the 14,343 text zones were classified correctly yielding a 97% correct accuracy for text zone classification. And there were 1,234 non-text zones of the 1,616 non-text zones that was classified correctly, yielding a accuracy of 76%. The total classification accuracy was 94%. Just to understand the significance of the accuracy results we note that if all zones were classified as text zones the classification accuracy would be 90

We are now in the process of going through all zones which were misclassified so that we can design additional features that would help correctly classify them.

1. N. Amamoto, S. Torigoe, Y. Hirogaki, "Block Segmentation and Text Area Extraction of Vertically/Horizontally Written Document," *2nd ICDAR*, Tsukuba, 1993, p739-742.
2. H.S. Baird, "Background Structure in Document Images," *Advances In Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, p253-269.
3. S. Chen and R.M. Haralick, "Recursive Erosion, Dilation, Opening and Closing

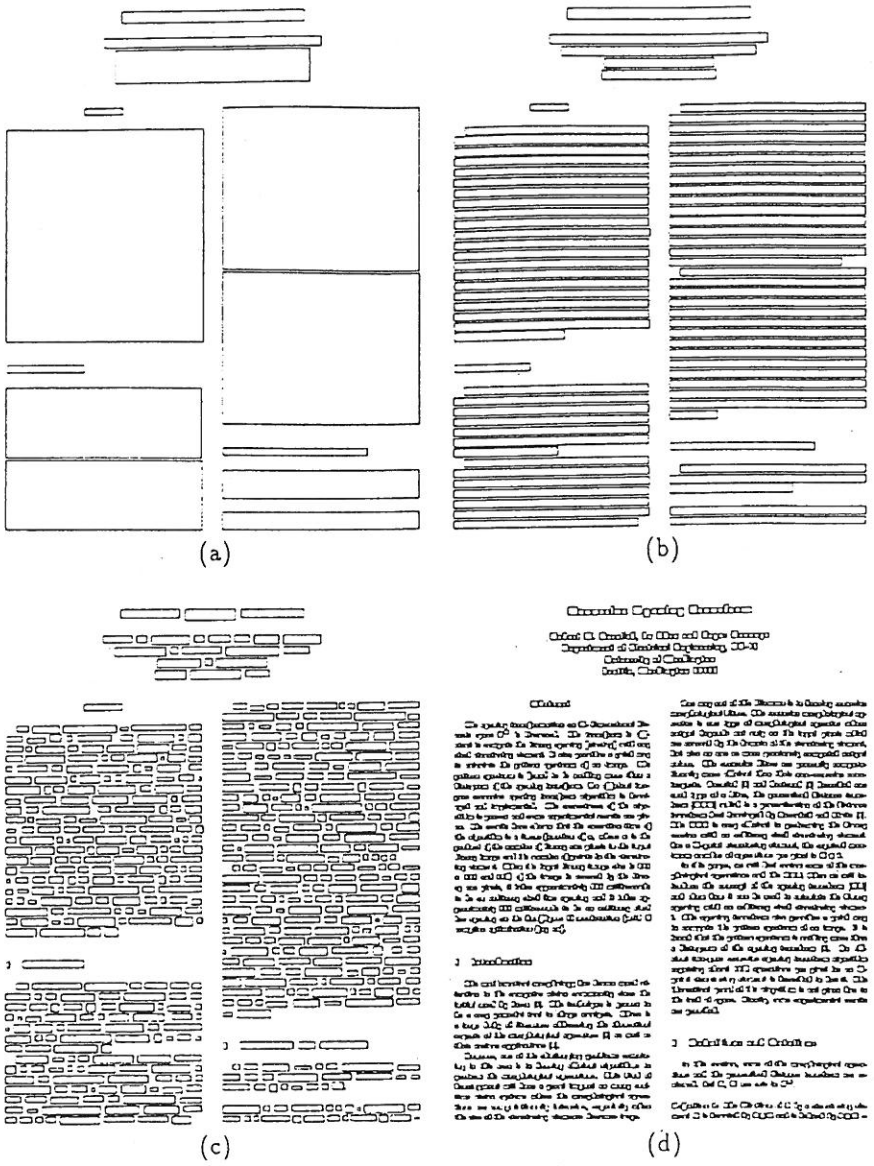


Figure 13: Illustrates the hierarchical layout representation of the example document page. (a) Zone bounding boxes; (b) Line bounding boxes; (c) Word bounding boxes; (d) Character bounding boxes

Transforms", to appear in *IEEE Trans on Image Processing*, Mar. 1995.
4. J.L. Fisher, S.C. Hinds, and D.P. D'Amato, "A Rule-Based System For Document Image-Segmentation," *10th ICPR*, Atlantic City, 1990, p567-572.
5. X. Hao, J.T.L. Wang, P.A. Ng, "Nested Segmentation: An Approach for Layout Analysis in Document Classification," *2nd ICDAR*, Tsukuba, 1993, p319-322. Atlantic City, 1990, p464-468.
6. R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, 1991.
7. Y. Hirayama, "A Block Segmentation Method For Document Images with Complicated Column Structures," *2nd ICDAR*, Tsukuba, 1993, p91-94.
8. D.J. Ittner and H.S. Baird, "Language-Free Layout Analysis," *2nd ICDAR*, Tsukuba, 1993, p336-340.
9. F. Lebourgeois, Z. Bublinski, and H. Emptoz, "A Fast and Efficient Method For Extracting Text Paragraphs and Graphics from Unconstrained Documents," *11th ICPR*, The Hague, 1992, p272-276.
10. G. Nagy and S.C. Seth, "Hierarchical Representation of Opically Scanned Documents," *7th ICPR*, Montreal, 1984, p347-349.
11. G. Nagy, S.C. Seth, S.D. Stoddard, "Document Analysis With An Expert System," *Pattern Recognition in Practice II*, E.S. Gelsema and L.N. Kanal editors, North Holland, Amsterdam, 1986, p149-159.
12. L. O'Gorman, "The Document Spectrum for Bottom-Up Page Layout Analysis," *Advances In Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, p270-279.
13. T. Pavlidis and J. Zhou, "Page Segmentation by White Streams," *1st ICDAR*, Saint-Malo, 1991, p945-953.
14. I.T. Phillips, S. Chen and R.M. Haralick, "English Document Database Standard", *Proc. of the Second International Conference on Document Analysis and Recognition* , Japan, October 20-22, 1993, pp. 478-483.
15. T. Saitoh and T. Pavlidis, " Page Segmentation without Rectangle Assumption," *11th ICPR*, The Hague, 1992, p 277-280.
16. T. Saitoh, M. Tachikawa, T. Yamaai, "Document Image Segmentation and Text Area Ordering," *2nd ICDAR*, Tsukuba, 1993, p323-329.
17. S. Tsujimoto and H. Asada, "Understanding Multi-articled Documents," *10th ICPR*, Atlantic City, 1990, p551-556.
18. F.M. Wahl, K.Y. Wong, and R.G. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," *Computer Graphics and Image Processing* Vol 20, 1982, p375-390.
19. A. Yamashita, T. Amasno, H. Takahashi, and K. Toyokawa, "A Model Based Layout Understanding Method for the Document Recognition System," *1st ICDAR*, Saint-Malo, 1991, p130-138.