Journal of Electronic Imaging 5(1), 25–36 (January 1996).

Extraction of text words in document images based on a statistical characterization

Su Chen
Robert M. Haralick
University of Washington
Department of Electrical Engineering
Seattle, Washington 98195
E-mail: suchen@caere.com

Ihsin T. Phillips
Seattle University
Department of Computer Science
Seattle, Washington 98122

Abstract. *Text structures in document images are usually laid out in a structured manner—having preferred spatial relations. These spatial relations are rarely deterministic; however, they can be modeled by probabilities. Therefore, any realistic document layout analysis algorithm should utilize this type of probabilistic knowledge to optimize its performance. We first describe a method for automatically generating a large amount of nearly perfect layout ground truth data from the LaTeX device-independent (DVI) files, where the bounding boxes for the characters, words, text lines, and text blocks are represented in hierarchies. These ground truth data enable us to construct statistical models that characterize the various layout structures in document images. We demonstrate this concept through the development of a word segmentation algorithm, which employs the recursive morphological closing transform to model word shapes in document images. We also conducted systematic experiments to evaluate the performance of our algorithm using the synthetic images generated from the LaTeX DVI files and the real images from the UW-I and UW-II English document image databases. The results indicate that the correct word detection rate is about 95% on the synthetic images and more than 90% on most of the tested real images.* © *1996 SPIE and IS&T.*

1 Introduction

Document layout analysis identifies various objects of interest on a document image and describes their spatial relations. An object is defined as a homogeneous rectangular region corresponding to one type: character, word, text line, text block, text column, or nontextual region.

Earlier work on document layout analysis can be categorically divided into two groups. One group employs the top-down or model-driven approach.[1,2] Top-down techniques start at the global image level and successively decompose the image into smaller regions. Each region has one type: character, word, text line, text block, text column, or nontextual region. Krishnamoorthy *et al.*[1] and Wang and Srihari[2] employ an X-Y tree as the representation of a document layout structure. The X-Y tree is a nested decomposition of rectangular blocks into smaller rectangular blocks. Each node in the X-Y tree corresponds to a rectangular block. The root node is the largest rectangular block, i.e., the input document image. At each level, the decomposition is induced by partitions in only one direction (horizontal or vertical), but a block may have an arbitrary number of children. In the process of partitioning, a block is segmented into subblocks by making cuts in the horizontal profile corresponding to troughs of depth and width greater than some threshold. Each resulting subblock has a vertical projection profile that can be similarly partitioned for vertical segmentation. The segmentation process may be carried out recursively to any desired depth with alternating horizontal and vertical subdivisions.

The main problems associated with this approach are: (1) At each step of the successive decompositions, the system has to select the correct decomposition model since the models for the text column, text block, text line, word, or character decomposition are inherently different. On the other hand, there are occasions when such model selections do not correspond to the levels in the decomposition tree. (2) Some popular top-down decomposition schemes, such as the above mentioned recursive X-Y cut technique, do not work for certain types of document layout topology. This is especially true when there is noise present on the document image.

The other group utilizes the bottom-up or data-driven approach,[3–5] which starts by synthesizing evidence at the black-and-white pixel level and then merges pixels into characters, characters into words, words into lines, lines into blocks, and blocks into columns, etc., until the whole document is completely labeled. These techniques are usually based on the connected component analysis. A connected component is a set of binary one pixels in a binary image that are maximally 4-connected or 8-connected. Fletcher and Kasturi's algorithm[3] assumes that each con-

Paper DDI-06 received June 9, 1995; revised manuscript received Nov. 14, 1995; accepted for publication Nov. 20, 1995.
1017-9909/96/$6.00 © 1996 SPIE and IS&T.

nected component in the image corresponds to a character or a nontextual object. It starts by extracting all the connected components in the input image. A Hough transform is applied to the centroid of the enclosing rectangles of the connected components to find collinear components. Positional relationships between collinear components, an intercharacter gap threshold, and an interword gap threshold are then used to group the components into text strings. One drawback of the method is that it is often sensitive to touching characters and fragmented characters because the underlying connectivity assumptions are violated. For some types of document images where text is printed in the dot matrix form, the algorithm breaks down significantly.

The main shortcomings of these earlier techniques are that they were developed on a trial-and-error basis and that although they provide illustrative results, hardly any have been tested on data sets of significant size. The reason is the lack of accurate document layout ground truth data of substantial size. It is clear that if we want to advance the document recognition research, any techniques that we develop must be proven on data sets of substantial size.

In addition, most papers do not give any explicit quantitative performance measure of their system. Although the appropriate performance measures for the document layout analysis are not obvious and are hard to derive, it is clear that suitable performance measures not only enable us to construct a system that optimizes its performance measures given the training data set, but also enable us to predict the system performance on the testing data set.

Section 2 describes a technique to automatically create a large amount of nearly perfect ground truth data suitable for the development of document layout analysis algorithms. Section 3 reviews the recursive morphological closing transform (RCT). Section 4 formulates the word segmentation problem as a pixel classification problem. In Sec. 5, we describe a word segmentation algorithm using RCT and a MAP classifier. Section 6 discusses an experimental protocol to train and evaluate the word segmentation algorithm. Finally, Sec. 7 summarizes our experimental results.

2 Document Layout Ground Truth Generation

The "UW English Document Image Database (I)"[6] is a data set for OCR and document image understanding algorithm development and evaluation. The database contains software to convert a DVI file from the LaTeX document processing system into bitmap images[7] and generate a so-called *character ground truth* file for each page. The file contains the bounding box, the type and size of font, and the ASCII code for each character on the page. The database provides a population of 168 such synthetically generated bitmap images. These images are manually segmented into rectangular zones. The row and column coordinates of the zone box corners are recorded.

In the following sections, we will describe a system that takes the character ground truth file and the zone box delineations of a synthetic document image and creates a tree representation of the layout structure of the document image. The root node represents the whole document image. The nodes in succeeding levels represent zones, text lines, words, and characters, respectively. Each node in the tree is specified by its bounding box.

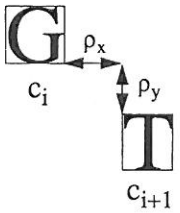


Fig. 1 Illustration of the horizontal and vertical distances between two character bounding boxes.

2.1 *Notation and Assumption*

Let a document image be denoted as $\mathcal{I}$. Let $\mathcal{Z} = \{z_1, z_2, \ldots, z_k\}$ denote the set of zones in the document image $\mathcal{I}$ where k is the total number of zones. Let the character ground truth file be modeled as a sequence of character bounding boxes $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, where n is the total number of characters.

Our assumption on the character bounding box sequence $\mathcal{C}$ is that it follows the same order as the logical reading order of the characters on the document image. We assume that spacings between two adjacent characters follow different probability distributions for the character breaks, the word breaks, and the text line breaks. The character break spacings are smaller than the word break spacings, and the word spacings are smaller than the text line break spacings.

2.2 *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 sequence $\mathcal{C}$. The distance measure is defined as follows:

   $$\rho(c_1, c_{i+1}) = \rho_x(c_i, c_{i+1}) + w\rho_y(c_i, c_{i+1}),$$

   where $i = 1, 2, \ldots, n-1$. The $\rho_x(c_i, c_{i+1})$ and $\rho_y(c_i, c_{i+1})$ are the minimum horizontal and vertical distance between the edges of 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 (see Fig. 1). Here w is a weight with a typical value of $w = 2.0$.
2. Compute the histogram of the $\rho(c_i, c_{i+1})$. Normally, it contains three peaks: one for character breaks, one for word breaks, and one for text line breaks. The first two peaks are relatively stronger (see Fig. 2).
3. Text line segmentation: If $\rho(c_i, c_{i+1}) > T_2$, then the break between c_i and c_{i+1} is a text line break. Here $T_2 = \alpha S$, where S is the dominant character font size and α is a constant with a typical value of $\alpha = 10.0$. 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 text line breaks.
4. Word segmentation: 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

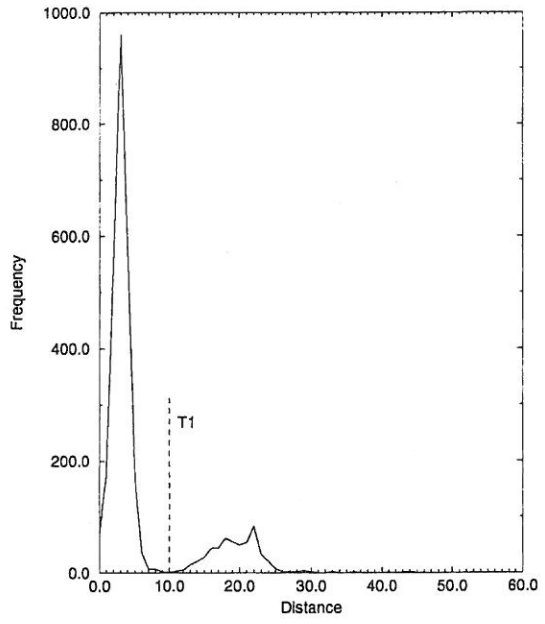


Fig. 2 Illustration of the histogram of the distance $\rho(c_i, c_{i+1})$ and the within-word character spacing threshold T_1.

$T_1 < \rho(c_i, c_{i+1}) \leq T_2$, 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 encloses all the character bounding boxes within the two adjacent word breaks. All the enclosing character bounding boxes constitute the descendents of the word bounding box. To estimate the threshold T_1 on the fly, we employ a modified Kittler automatic threshold algorithm.[8] Furthermore, the word to text line correspondence is established by finding all the word bounding boxes that are enclosed between two succeeding text line breaks.

5. Find zone correspondence: Each text line and all its descending word and character boxes are assigned to a unique zone z_j that has the maximum overlap with the text line bounding boxes. Since in the "UW English Document Image Database (I)," a zone bounding box is not necessarily the minimum zone bounding box that encloses the content of the zone, we modify the zone bounding box so that it is the minimum bounding box that encloses all the text lines assigned to the zone.

2.3 *Performance Evaluation and Examples*

For performance evaluation, we ran the algorithm on the 168 synthetic images from the "UW English Document Image Database (I)."[6] In these images, there are a total of 1366 text zones and 2423 displayed math zones. There are a total of approximately 10,000 text lines and 60,000 words.

Then we output the zone, text line, and word bounding boxes overlaid on top of the corresponding document image onto the computer screen to check for any errors. We found only four or five locations where two adjacent text lines were incorrectly merged together. The scenarios were that the second line started immediately below the end of the first line. After giving a higher weight to the vertical distance, the algorithm generated all the text line bounding boxes correctly and automatically.

We also observed that the algorithm generated incorrect results on some of the displayed math formula zones. The reason is that the placement of subsequent symbols violates our underlying assumptions (Sec. 2.1) and the usual definitions of text lines and words are no longer valid. But since it is not our purpose to provide layout ground truth data for the displayed math zones, we ignore these cases.

To quantify the accuracy of the word ground truth extraction results, we compared the word bounding boxes generated using the current approach (hereafter denoted as GT-1) with the so-called *golden ground truth* word bounding boxes, which are created in the following way: (1) compute a second set of ground truth word bounding boxes (hereafter denoted as GT-2) using the algorithm described in Ref. 9; (2) compare the word bounding boxes of GT-1 and GT-2 and mark the differences (see Sec. 6.2); and (3) check the differences and manually edit the word bounding boxes in GT-1 and GT-2 to generate the final golden ground truth word bounding boxes. We assumed that the probability that the two algorithms make the same errors is minimal.

We calculated statistics such as the rates of missed, false, correct, split, merged, and spurious detections (see definitions in Sec. 6.2). Table 1 indicates that the 60,875 golden ground truth words, 98.74% of them are extracted correctly using the current method. A total of 0.62% and 0.64% of the words are either split or merged, respectively. On the other hand, Table 2 illustrates that of the 61,289 ground truth words generated by the current algorithm, 98.07% of them correspond correctly to words in the golden ground truth. A total of 1.36% and 0.31% of the words are either split or merged, respectively. The remaining 0.26% of the words are falsely detected.

As an example, Fig. 3 illustrates one of the synthetic document images. Figure 4 gives the generated zone, text line, word, and character layout ground truth data.

Table 1 Word performance with respect to the golden ground truth.

Total golden ground truth words	Correct	Split	Merged	Missed	Spurious
60875	60108	378	388	1	0
	(98.74%)	(0.62%)	(0.64%)	(0.00%)	(0.00%)

Table 2 Word extraction performance with respect to the algorithm output.

Total ground truth words	Correct	Split	Merged	False	Spurious
61289	60108	836	187	158	0
	(98.07%)	(1.36%)	(0.31%)	(0.26%)	(0.00%)

Recursive Opening Transform

Robert M. Haralick, Su Chen and Tapas Kanungo
Department of Electrical Engineering, FT-10
University of Washington
Seattle, Washington 98195

Abstract

The opening transformation on N-dimensional discrete space Z^N is discussed. The transform is efficient to compute the binary opening (closing) with any sized structuring element. It also provides a quick way to calculate the pattern spectrum of an image. The pattern spectrum is found to be nothing more than a histogram of the opening transform. An efficient two-pass recursive opening transform algorithm is developed and implemented. The correctness of the algorithm is proven and some experimental results are given. The results have shown that the execution time of the algorithm is a linear function of n, where n is the product of the number of binary one pixels in the input binary image and the number of points in the structuring element. When the input binary image size is 256 x 256 and 50% of the image is covered by the binary one pixels, it takes approximately 250 milliseconds to do an arbitrary sized line opening and it takes approximately 500 milliseconds to do an arbitrary sized box opening on the Sun/Sparc II workstation (with C compiler optimization flag on).

1 Introduction

The mathematical morphology has drawn much attention in the computer vision community since the initial work by Serra [1]. The technique is proven to be a very powerful tool in shape analysis. There is a large body of literature addressing the theoretical aspects of the morphological operators [2] as well as their various applications [3].

However, one of the challenging problems remaining in this area is to develop efficient algorithms to perform the morphological operations. This kind of development will have a great impact on many real-time vision systems where the morphological operations are computationally intensive, especially when the size of the structuring elements becomes large.

One way out of this dilemma is to develop recursive morphological filters. The recursive morphological operator is one type of morphological operator whose output depends not only on the input pixels which are covered by the domain of the structuring element, but also on one or more previously computed output values. The recursive filters are generally computationally more efficient than their non-recursive counterparts. Haralick [5] and Bertrand [6] described one such type of a filter, the generalised distance transform (GDT) which is a generalisation of the distance transform first developed by Rosenfeld and Pfalts [4]. The GDT is very efficient in performing the binary erosion with an arbitrary sized structuring element. For a N-point structuring element, the required maximum number of operations per pixel is $N+2$.

In this paper, we will first review some of the morphological operations and the GDT. Then we will introduce the concept of the opening transform (OT) and show how it can be used to calculate the binary opening with an arbitrary sized structuring element. The opening transform also provides a quick way to compute the pattern spectrum of an image. It is found that the pattern spectrum is nothing more than a histogram of the opening transform [3]. An efficient two-pass recursive opening transform algorithm requiring about $14N$ operations per pixel for an N-point structuring element is described in detail. The theoretical proof of the alogrithm is not given due to the lack of space. Finally, some experimental results are provided.

2 Definitions and Notations

In this section, some of the morphological operations and the generalised distance transform are reviewed. Let A, K are sets in Z^N.

Definition 1: The dilation of A by a structuring element K is denoted by $A \oplus K$ and is defined by $A \oplus K =$

Fig. 3 Illustration of an example document page.

In conclusion, the technique described in this section provides an efficient and automatic way of creating a large amount of nearly perfect ground truth data for document layout analysis. We can use these data to develop, train, and evaluate our document layout analysis algorithms. In the following, we demonstrate a procedure to build a statistical model to characterize text words on document images, and then we describe a word segmentation algorithm using the recursive morphological closing transform.[10] to detect all text words on document images simultaneously.

3 Recursive Closing Transform: A Review

The closing transform of a set I with respect to a structuring element K generates a gray-scale image where the gray level of each pixel $x \in Z^2$ is defined as the smallest positive integer n so that $x \in I \bullet (\oplus_{n-1} K)$, where $\bullet$ denote the morphological closing operation and $\oplus_{n-1} K$ denotes the $(n-1)$-fold dilation of the base structuring element K by itself.[8] 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 defined to be zero.

Definition 1. The closing transform *of a set $I \subseteq Z^2$ by a structuring element $K \subseteq Z^2$ is denoted by CT[I,K] and is defined as:*

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

The morphological closing transform captures the shape-size content of the image background. In Ref. 10, we described an efficient recursive closing transform (RCT) to compute in constant time per pixel the morphological closing transform of a binary image.

4 Word Segmentation as a Statistical Classification Problem

Let I denote a bilevel document image. Let Σ denote the set of word bounding boxes on I. It provides a delineation of two types of regions in the image I, namely the word and nonword regions. We define a pixel as a word pixel if and only if it is on or inside the bounding box of a word. Then, a word region consists solely of word pixels, whereas a nonword region is composed of only nonword pixels.

Associated with each pixel $x \in I$, there is a random observation vector $\mathcal{Y}(x) = y$ that characterizes the image shapes around x. There is also a label $\mathcal{L}(x) = l$ that indicates whether x is a word pixel [denoted by $\mathcal{L}(x) = 1$] or a nonword pixel [denoted by $\mathcal{L}(x) = 0$]. Let $\mathcal{Y}(I)$ and $\mathcal{L}(I)$ represent images of observation vectors and labels, respectively. Hence, we can formulate the word segmentation problem as finding $\mathcal{L}(I)$ to maximize the posterior probability $P[\mathcal{L}(I) | \mathcal{Y}(I)]$, and then computing the bounding boxes for the connected word regions in $\mathcal{L}(I)$.

Our observation vector $\mathcal{Y}(x) = y$ is based on the recursive closing transform (Sec. 3). Let $K_1, K_2, \ldots, K_n$ denote n different structuring elements. Let $y = (y_1, y_2, \ldots, y_n)$, where $y_1 = CT[I,K_1](x)$, $y_2 = CT[I,K_2](x), \ldots$, and $y_n = CT[I,K_n](x)$, denote the values of the recursive closing transform at pixel $x \in I$ with respect to the structuring elements $K_1, K_2, \ldots, K_n$. The vector $\mathcal{Y}(x) = y$ provides a scale-free description of image shapes in the neighborhood of x.

In Sec. 5, we will describe a simple algorithm to solve the above problem. It has the following prominent characteristics: (1) Unlike most of the top-down or bottom-up approaches where the objects of interest are derived in a recursive fashion, our method is a one-step and simultaneous process. (2) The method utilizes the recursive morphological closing transform (RCT) to extract image shapes. The RCT provides a powerful tool to extract shape information in the image background (white space), such as the pattern spectrum. (3) The method is not sensitive to text skew because only local shape information is used. Texts can be laid out in both the horizontal and the vertical directions at the same time. (4) The method is robust under subtractive noise. Therefore, character fragmentation will not affect the performance of the algorithm. The algorithm is also tolerant to some forms of additive noise. (5) The method is trainable to any given document image population. (6) The same methodology is directly applicable to both the text line and the character segmentations.

5 Algorithm Description

The various components of the word segmentation algorithm are described next:

Step 1: *Subsampling*

Our input document images are scanned at a spatial resolution of 300 dots per inch (dpi). To process such an image, it takes a considerable amount of memory and processing time. Therefore, we subsample the original image to 150 dpi.

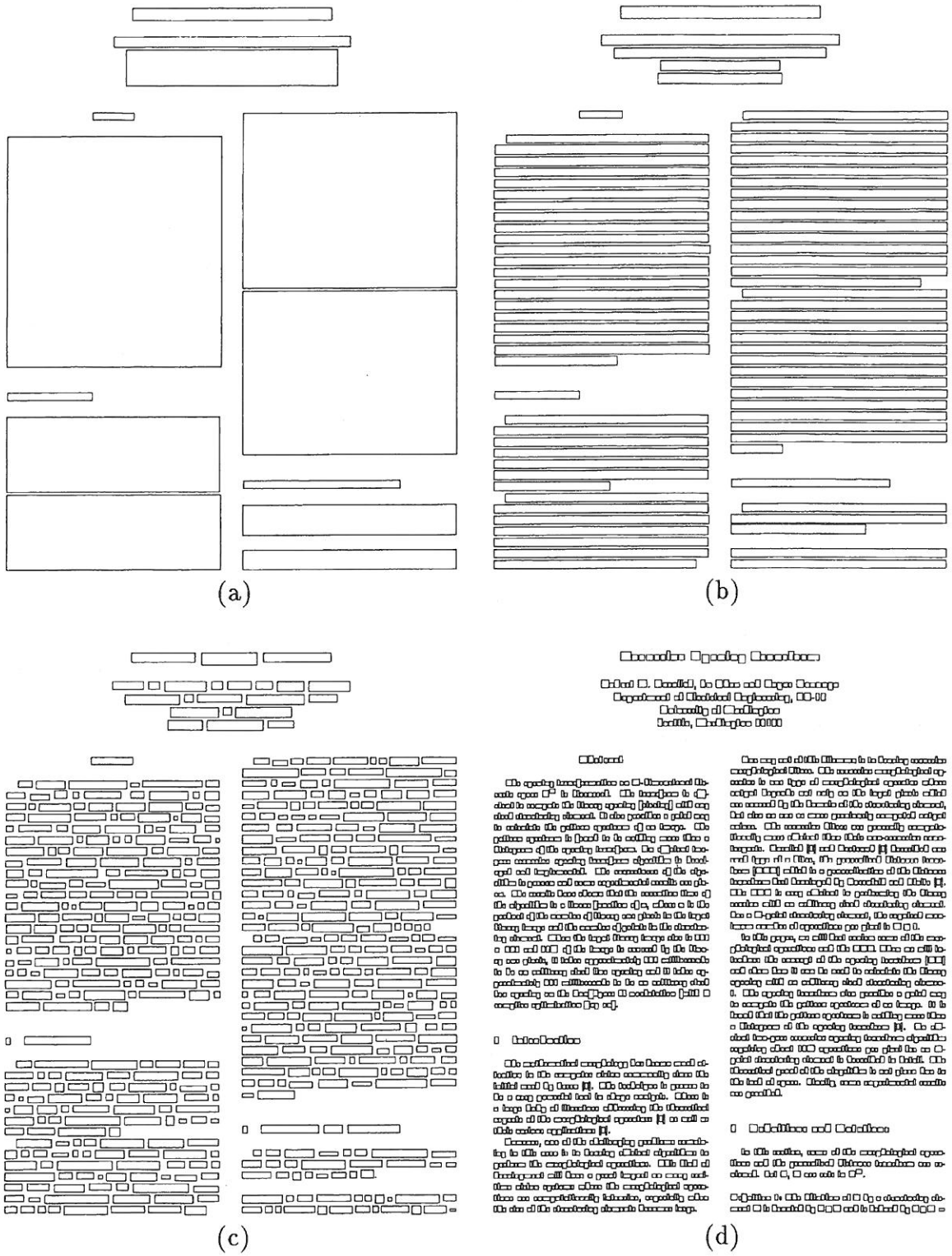


Fig. 4 Illustration of the hierarchical layout representation of the example document page: (a) zone bounding boxes, (b) line bounding boxes, (c) word bounding boxes, and (d) character bounding boxes.

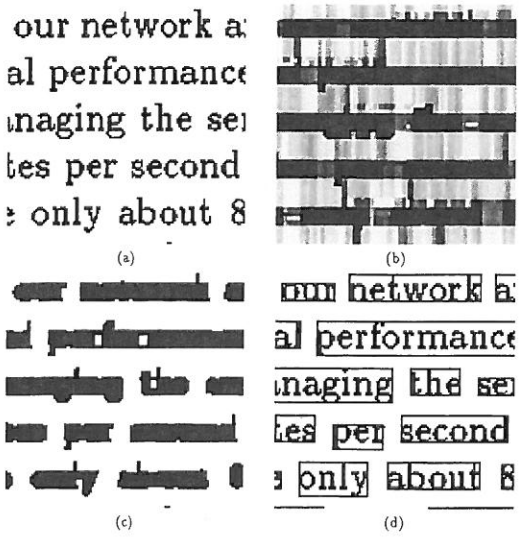


Fig. 5 Illustration of the word segmentation process: (a) subsampled 150-dpi image, (b) correlated posterior probability map image, (c) word label image, and (d) word bounding boxes.

The subsampling algorithm that we have implemented is as follows: let the horizontal and vertical subsampling ratio be H and V, respectively. Given an input $R \times C$ bilevel image, the algorithm generates an output bilevel image with a dimension of $\lfloor R/V \rfloor \times \lfloor C/H \rfloor$, where the operation $\lfloor x \rfloor$ returns the greatest integral value less than or equal to "x". Each pixel in the output image corresponds to a nonoverlapping $V \times H$ window in the input image. If the number of binary one pixels in the input $V \times H$ window is greater than or equal to a prespecified threshold T, its corresponding output pixel is set to binary one; otherwise, it is set to binary zero. To obtain a 150-dpi subsampled image, we select $H=2$, $V=2$, an $T=2$. Figure 5(a) illustrates one segment of the subsampled 150-dpi image.

Step 2: *Word region detection*

We assume that the observation vectors from different pixel locations are statistically independent, i.e.,

$$P[\mathscr{L}(I)|\mathscr{Y}(I)]=\prod_{x \in I} P[\mathscr{L}(x)|\mathscr{Y}(x)].$$

To perform the probability optimization, we assign a posterior probability $P[\mathscr{L}(x)=1|\mathscr{Y}(x)=y]$ to each pixel for being a word pixel. Then the posterior probability for each pixel to be a nonword pixel is equal to $P[\mathscr{L}(x)=0|\mathscr{Y}(x)=y]=1-P[\mathscr{L}(x)=1|\mathscr{Y}(x)=y]$. The process generates a posterior probability map image. Since all pixels are treated identically, we will leave out the index x in the following discussions for the sake of simplicity. Let $P(\mathscr{L}=1|\mathscr{Y}=y)$ denote the posterior probability for each pixel being a word pixel. It is estimated during the initial experimental stage.

To introduce the correlation among the neighboring pixels in the probability map image, we morphologically close and then open the probability map image by a flat structuring element S. We select S to be a 2×2-sized structuring element. Figure 5(b) illustrates the correlated posterior probability map image. If we threshold the probability map image at a threshold value of $T_p=0.5$, i.e., pixels that have values grater than or equal to T_p have binary one output values, we obtain the maximum *a posteriori* (MAP) classification. But in general, we could choose T_p between 0.5 and 1.0. A low threshold T_p value tends to merge several words into one block, and a high threshold T_p value tends to split a word into many blocks. Figure 5(c) illustrates the detected word label image, where $T_p=0.96$.

Step 3: *Word bounding box extraction*

We model each work in I as an 8-connected region in the word label image. The connected component labeling procedure described in Ref. 8 is used to extract the bounding box of each connected region. Figure 5(d) illustrates the extracted word bounding boxes overlaid on top of the subsampled image.

Step 4: *Hypothesis test on word height*

The presence of the character ascenders and descenders sometimes causes the merging of word blocks from two or more adjacent text lines into a single block. To automatically detect such cases and consequently split the merged word blocks into their corresponding correct words, we developed a simple postprocessing procedure to perform hypothesis testing on the height of the word blocks and test if further divisions are needed.

Let W_h denote the dominant word height of a given document image population. Then the procedure hypothesizes that all the detected word blocks whose heights exceed βW_h could be split further, where β is a real constant and has a default value of 2.0. For each word block that is hypothesized to be divided further, the algorithm will verify it by computing all possible cut points in the projection profile of the posterior probability map image along the height direction and within the bounding box of the dubious word block.

Let H and W denote the height and width of the word block. Let $P[h,w]$ represent the posterior probability map image inside the word block window, where $1 \leq h \leq H$ and $1 \leq w \leq W$. Let $f(h)$ denote the calculated probability projection profile. Then $f(h)=(1/W)\Sigma_{w=1}^{W} P[h,w]$, where $1 \leq h \leq H$. The cut points of the projection profile $f(h)$ are defined as the local minimums of $f(h)$ in a neighborhood of size W_h and whose values are less than or equal to a cut-point threshold T_c, where $0.0 \leq T_c \leq 1.0$ and T_c has a default value of 0.5 (see Fig. 6). If the number of such detected cut points other than the two endpoints ($h=1$ and $h=H$) is greater than zero, then the word block needs to be split further. The following algorithm describes the procedure to compute the cut points in the projection profile $f(h)$:

Algorithm:

1. Morphologically open the projection profile $f(h)$ by a flat structuring element of size $W_h/2$, denoted by k_1. This will remove the narrow upshoot spikes in $f(h)$. Let $f_1=f \circ k_1$.
2. Morphologically close $f_1(h)$ by a flat structuring element of size D_m, denoted by k_2. This will bridge

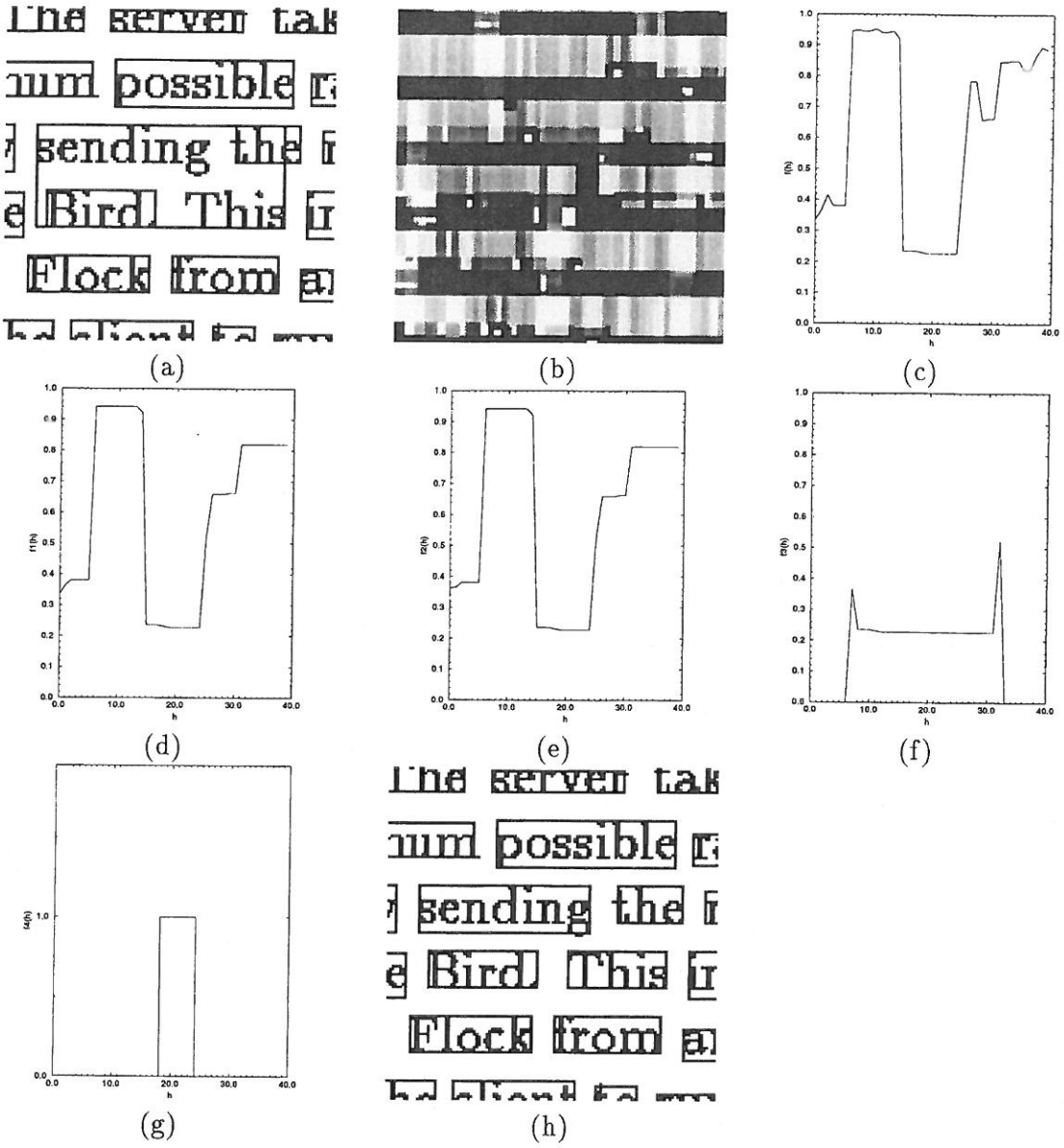


Fig. 6 Illustration of the projection profile analysis on the posterior probability map image: (a) word boxes before the analysis, (b) the posterior probability map image, (c) $f(h)$, (d) $f_1(h)$, (e) $f_2(h)$, (f) $f_3(h)$, (g) $f_4(h)$, and (h) word boxes after the analysis.

the narrow valleys in $f_1(h)$ and ensure that the cut points are at least D_m pixels wide. We select the default $D_m = 5$. Let $f_2 = f_1 \bullet k_2$.

3. Morphologically erode $f_2(h)$ by a flat structuring element of size W_h, denoted by k_3. Let $f_3 = f_2 \ominus k_3$. Then the set of possible cut points is defined as $\{H \in [1,H] | f_2(h) \leq T_c \text{ and } f_2(h) = f_3(h)\}$, which is the set of local minimums of $f_2(h)$ in a neighborhood of size W_h and whose values are less than or equal to the cut-point threshold T_c. □

The detected cut points produce a set of intervals (or run lengths) along the height direction. If the number of such intervals that do not contain the two endpoints ($h=1$ and $h=H$) is greater than zero, then the word region needs to be split further at those intervals and the bounding boxes of the subword regions are recomputed. Figure 6 gives an illustration of the projection profile analysis process.

6 Experimental Protocol

In the previous section, we outlined a word segmentation algorithm. The algorithm requires the posterior probability $P(\mathscr{L}=1|\mathscr{Y}=y)$ to be estimated. Also, to make the word segmentation algorithm fully automatic, we need to develop a procedure to estimate the optimal threshold parameter T_p on a per image basis.

6.1 *Posterior Probability Distribution Estimation*

The estimation of the posterior probability distribution is based on the 168 synthetic document images. The process

to create the ground truth layout structures for these images is described in Sec. 2. To compute the posterior probability distribution, we first generate a word masking image for each of the 168 document images. The word mask image is bilevel and has a binary one pixel if and only if the pixel is a word pixel. Each document image and its corresponding word mask image are then rotated at various degrees of 0, ± 0.2, ± 0.4, and ± 0.6 deg using a nearest neighbor interpolation algorithm. The range of rotation angles is selected because our skew estimation algorithm is capable of estimating text skew angles on document images that are within 0.5 deg of the true text skew angles at a probability of 99%.[11] This generates a total input training image population of $1176=168\times 7$ images. Each image is of size 1650×1275.

We adopt a rather brute-force method to estimate the posterior probability $P(\mathscr{L}=1|\mathscr{Y}=y)$:

$$P(\mathscr{L}=1|\mathscr{Y}=y)=\frac{P(\mathscr{L}=1,\mathscr{Y}=y)}{P(\mathscr{Y}=y)}=\frac{P(\mathscr{L}=1,\mathscr{Y}=y)}{P(\mathscr{L}=0,\mathscr{Y}=y)+P(\mathscr{L}=1,\mathscr{Y}=y)}.$$

The joint probability distributions are estimated using the frequency counts $\#(\mathscr{L}=0,\mathscr{Y}=y)$ and $\#(\mathscr{L}=1,\mathscr{Y}=y)$. In the experiment, we choose the dimensionality of $\mathscr{Y}$ to be equal to $n=3$. Let K_1 be a horizontal 1×2 structuring element, K_2 be a vertical 2×1 structuring element, and K_3 be a 2×2 square structuring element. We further bound the observation vector $\mathscr{Y}$ to lie inside the three-dimensional cube $[0,N]\times[0,N]\times[0,N]$, where N is the allowed maximum output value of the closing transform. For the word segmentation, we select $N=63$. Since the vectors $\mathscr{Y}=(y_1,y_2,y_3)$ are integer vectors, we can represent the posterior probability $P(\mathscr{L}=1|\mathscr{Y}=y)$ by a simple three-way look-up table. It also made the counting process very simple.

In this paper, we further assume that $P(\mathscr{L}=1|\mathscr{Y}=y)$ is symmetric with respect to the first two coordinates of $\mathscr{Y}$, i.e., $P[\mathscr{L}=1|\mathscr{Y}=(y_1,y_2,y_3)]=P[\mathscr{L}=1|\mathscr{Y}=(y_1,y_2,y_3)]$. This will permit the posterior probability distribution to characterize text words laid out in both the horizontal and the vertical directions. Therefore, we estimate $P(\mathscr{L}=0,\mathscr{Y}=y)$ from the frequency count $\#(\mathscr{L}=0,\mathscr{Y}=y_1,y_2,y_3)+\#[\mathscr{L}=0,\mathscr{Y}=(y_2,y_1,y_3)]$ and $P(\mathscr{L}=1,\mathscr{Y}=y)$ from the frequency count $\#[\mathscr{L}=1,\mathscr{Y}=(y_1,y_2,y_3)]+\#[\mathscr{L}=1,\mathscr{Y}=(y_2,y_1,y_3)]$.

6.2 *Word Segmentation Algorithm Evaluation*

The output of the word segmentation algorithm is a set of word bounding boxes. To evaluate its performance, we need to compare the output word bounding boxes with the ground truth word bounding boxes provided through the procedure given in Sec. 2. Let $\mathscr{G}=\{G_1,G_2,\ldots,G_N\}$ represent the total of N ground truth word bounding boxes and let $\mathscr{D}=\{D_1,D_2,\ldots,D_M\}$ denote the total of M detected word bounding boxes from the word segmentation algorithm. The evaluation problem can be formally stated as follows: Given two sets of bounding boxes $\mathscr{G}$ and $\mathscr{D}$, establish the element mappings between the two sets and report the number of misdetections (1-0 mappings), false detections (0-1 mappings), correct detections (1-1 mappings), split detections (1-m mappings), merged detections (m-1 mappings), and spurious detections (m-m mappings).

To establish the element mappings, we first define the similarity between two bounding boxes A and B, denoted by $s(A,B)$:

$$s(A,B)=\frac{\text{Area}(A\cap B)}{\text{Area}(A)},$$

where $A\cap B$ denotes the region where A and B overlap. The similarity defines the percentage area coverage of A by B.

Then based on the similarity measure, we define two mappings $g:\mathscr{G}\rightarrow\mathscr{D}$ and $d:\mathscr{D}\rightarrow\mathscr{G}$:

$$g(G_i)=\{D_j\in\mathscr{D}|G_i=\arg\max_{X\in\mathscr{G}} s(D_j,X)\},$$

$$d(D_j)=\{G_i\in\mathscr{G}|D_j=\arg\max_{X\in\mathscr{D}} s(G_i,X)\},$$

where $g(G_i)$ denotes the set of $D_j\in\mathscr{D}$ that has the highest percentage area coverage by G_i among all other boxes in $\mathscr{G}$, and $d(D_j)$ denotes the set of $G_i\in\mathscr{G}$ that has the highest percentage area coverage by D_j among all other boxes in $\mathscr{D}$. Therefore, we establish links from G_i to $g(G_i)$ and from D_j to $d(D_j)$.

Based on the two functions $g:\mathscr{G}\rightarrow\mathscr{D}$ and $d:\mathscr{D}\rightarrow\mathscr{G}$, we can establish mappings between the elements of $\mathscr{G}$ and $\mathscr{D}$. The rules are described as follows:

1. If there exists a G_i such that $s(G_i,D_j)=0$ for all $j=1,2,\ldots,M$, then the G_i is counted as a misdetection (1-0 mapping).
2. If there exists a D_j such that $s(D_j,G_i)=0$ for all $i=1,2,\ldots,N$, then the D_j is counted as a false detction (0-1 mapping).
3. There is a correct detection (1-1 mapping) between G_i and D_j if and only if $g(G_i)=\{D_j\}$ and $d(D_j)=\{G_i\}$.
4. There is a split detection (1-m mapping) between G_i and $\{D_{j_1},D_{j_2},\ldots,D_{j_m}\}$ if and only if (1) $g(G_i)=\{D_{j_1},D_{j_2},\ldots,D_{j_m}\}$; (2) there exists one $D_0\in g(G_i)$ such that $d(D_0)=\{G_i\}$ and for all $D\in g(G_i)$ but $D\neq D_0$, $d(D)=\emptyset$; and (3) for all $D\notin g(G_i)$, $G_i\notin d(D)$.
5. There is a merged detection (m-1 mapping) between $\{G_{i_1},G_{i_2},\ldots,G_{i_m}\}$ and D_j if and only if (1) $d(D_j)=\{G_{i_1},G_{i_2},\ldots,G_{i_m}\}$; (2) there exists one $G_0\in d(D_j)$ such that $g(G_0)=\{D_j\}$ and for all $G\in d(D_j)$ but $G\neq G_0$, $g(G)=\emptyset$; and (3) for all $G\notin d(D_j)$, $D_j\notin g(G)$.
6. Any other detections are counted as spurious detections (m-m mappings).

Table 3 Economic gain coefficients.

γ_{10}	γ_{01}	γ_{11}	γ_{1m}	γ_{m1}	γ_{mm}
0.0	0.0	1.0	0.5	0.5	0.0

Once the element mappings between $\mathscr{G}$ and $\mathscr{D}$ have been established, we count the numbers of missed, false, correct, split, merged, and spurious detections. Let N_{10}, N_{01}, and N_{11} be the numbers of missed, false, and correct detections, respectively. Let N^g_{1m}, N^g_{m1}, and N^g_{mm} denote the numbers of words in the $\mathscr{G}$ that have the 1-m, m-1, and m-m mappings with words in the $\mathscr{D}$. Similarly, let N^d_{1m}, N^d_{m1}, and N^d_{mm} denote the numbers of words in the $\mathscr{D}$ that have the 1-m, m-1, and m-m mappings with words in the $\mathscr{G}$. Then the following relations satisfy: (1) $N=N_{10}+N_{11}+N^g_{1m}+N^g_{m1}+N^g_{mm}$; (2) $M=N_{01}+N_{11}+N^d_{1m}+N^d_{m1}+N^d_{mm}$; (3) $N^g_{1m} \leqslant N^d_{1m}$; and (4) $N^g_{m1} \geqslant N^d_{m1}$.

The performance of the word segmentation algorithm can be measured through a goodness function. Let it be denoted as κ. It is defined by

$$\kappa = \min(\kappa_1, \kappa_2),$$

where

$$\kappa_1 = (\gamma_{10}N_{10} + \gamma_{11}N_{11} + \gamma_{1m}N^g_{1m} + \gamma_{m1}N^g_{m1} + \gamma_{mm}N^g_{mm})/N,$$

$$\kappa_2 = (\gamma_{01}N_{01} + \gamma_{11}N_{11} + \gamma_{1m}N^d_{1m} + \gamma_{m1}N_{d_{m1}} + \gamma_{mm}N^d_{mm})/M,$$

and the γ_{10}, γ_{01}, γ_{11}, γ_{1m}, γ_{m1}, and γ_{mm} are economic gain coefficients for the missed, false, correct, split, merged, and spurious detections. The larger the goodness measure κ, the better the performance of the word segmentation algorithm. In the experiment, we choose the economic gain coefficients as in Table 3.

6.3 *Optimal Threshold Determination*

In the word segmentation algorithm, there is a threshold value T_p that needs to be computed on a per image basis. Therefore, it is necessary to develop an automatic procedure to predict the optimal threshold value on the fly. Our approach to this problem is to first determine the optimal threshold values for each of the training document images and then construct a regression function to predict the optimal threshold value given the histogram of the posterior probability map image.[11]

Given an input document image, κ is a function of the threshold value T_p, i.e., $\kappa = \kappa(T_p)$. The optimal T_p is defined as the value that produces the best word segmentation goodness measure. Let T^{opt}_p denote the optimal threshold value. Then,

$$T^{opt}_p = \arg[\max_{T_p \in [0,1]} \kappa(T_p)].$$

Figure 7 illustrates the probability distributions of the optimal threshold values T^{opt}_p and the corresponding goodness measures for the 1176 training document images. The cumulative probability is defined as the Prob[$\kappa \geqslant \kappa_0$], i.e., the probability that the goodness measure κ is no less than κ_0. We observe that the optimal threshold values lie approximately in the range of [0.5, 1.0].

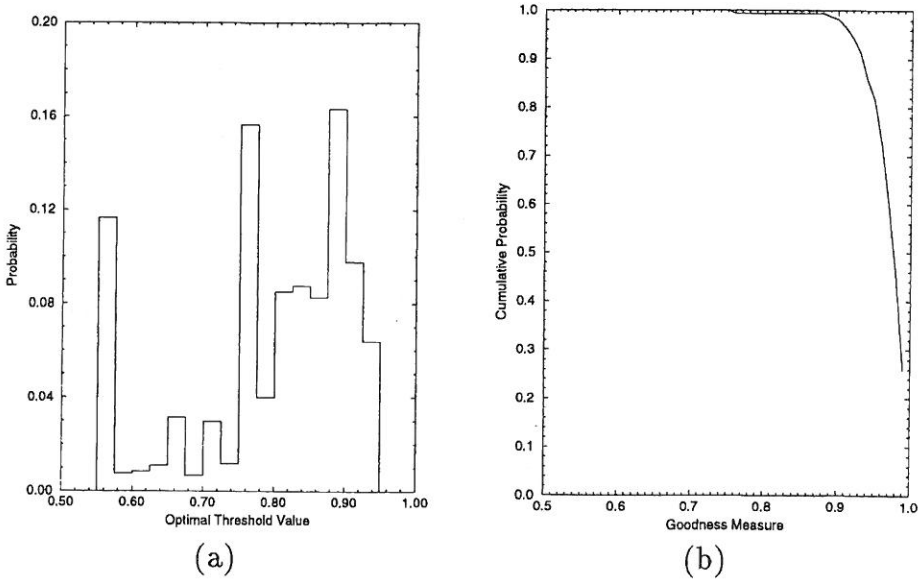


Fig. 7 Illustration of the probability distributions of the optimal threshold values T^{opt}_p and the corresponding goodness measure for the training document images.

Table 4 Algorithm performance with respect to the ground truth on the training image set.

Total ground truth words	Correct	Split	Merged	Missed	Spurious
429,338	408,741 (95.20%)	8440 (1.97%)	10961 (2.55%)	376 (0.09%)	820 (0.19%)

Table 6 Algorithm performance with respect to the ground truth on the testing image set.

Total ground truth words	Correct	Split	Merged	Missed	Spurious
258,328	245584 (95.07%)	4137 (1.60%)	7123 (2.76%)	846 (0.33%)	638 (0.25%)

7 Experimental Results

7.1 *Performance on the Training Images*

To benchmark the optimal performance of our word segmentation algorithm, we tested the algorithm on the 1176 training document images described in Sec. 6.1 under the optimal threshold setting $T_p = T_p^{opt}$. Tables 4 and 5 illustrate the percentages of missed, false, correct, split, merged, and spurious detections with respect to the ground truth (not the manually edited golden ground truth discussed in Sec. 2.3) as well as the algorithm output. The word boxes from displayed math zones are excluded during the evaluation because the ground truth word boxes for mathematical formulas are not accurate. Of the 429.338 ground truth words, 95.20% of them are correctly detected and 1.97% and 2.55% of the words are split or merged, respectively. The total missed and spurious detections account for about 0.28% of the total ground truth words. On the other hand, of the 434,390 words detected by the algorithm, 94.10% of them are correctly detected as the ground truth words, and 4.42% and 1.14% of the detected words are derived from either split or merged ground truth words, respectively. The total false and spurious detections account for about 0.35% of the total algorithm output.

7.2 *Performance on the Testing Images*

To assess the optimal performance of the algorithm on other document image population, we first prepared a new set of 96 LaTeX document pages, and then created the corresponding TIFF images and the ground truth word bounding boxes using the programs described in Sec. 2. Each of the 96 document images and its corresponding ground truth word bounding boxes are further rotated at various degrees of 0, ± 0.2, ± 0.4, and ± 0.6 deg. This generates a total of 672 testing document images.

Under the optimal threshold settings ($T_p = T_p^{opt}$), Tables 6 and 7 illustrate the percentages of missed, false, correct, split, merged, and spurious detections with respect to the ground truth as well as the algorithm output. Of the 258,328 ground truth words, 95.07% of them are correctly detected, and 1.60% and 2.76% of the words are split or merged, respectively. The total missed and spurious detections account for about 0.58% of the total ground truth words. On the other hand, of the 258,802 words detected by the algorithm, 94.89% of them are correctly detected as the ground truth words, and 3.59% and 1.14% of the detected words are derived from either split or merged ground truth words, respectively. The total false and spurious detections account for about 0.37% of the total algorithm output. The evaluation does not exclude the word boxes from the displayed mathematical formula. This explains the slight changes in the percentages for the split, merged, and spurious detections. But otherwise, the performance of the word segmentation algorithm on the testing document images is not significantly different from that on the training document images because the training set is sufficiently large.

7.3 *Performance on the Real Images*

We applied the same algorithm on a set of 763 real document images randomly selected from the "UW English Document Image Database (I) & (II)." We first constructed the ground truth word bounding boxes for these real images using a methodology similar to the one described in Sec. 2.3. The work is part of our efforts in constructing the UW English Document Image Database (III), which will be available to the public in 1996. There are a total of about 380,000 ground truth words in this set of images.

Figure 8 plots the probability distributions of the correct word detection rate of our word segmentation algorithm on a per image basis. We fixed $T_p = 0.95$ for all the images in the experiment. It shows that on most of the images, the correct word detection rate is more than 90%.

Figures 9 and 10 illustrate two examples of the word segmentation results. The whole process takes about 20 s per image on a Sun Sparc 10 workstation.

8 Conclusions and Future Work

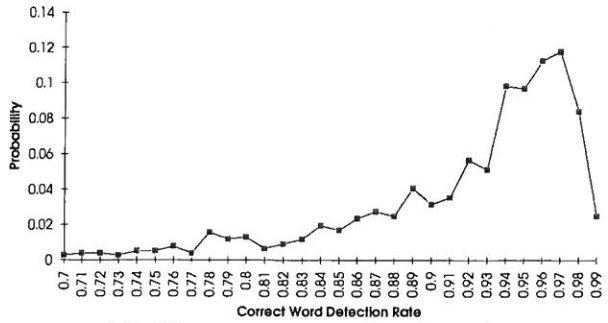
In this paper, we discuss an engineering approach for text word extraction in document images. We first presented an automatic method to generate a large amount of nearly perfect document layout ground truth data from the LaTeX DVI files. Then, we used the layout ground truth data to build a statistical model to characterize the word structures

Table 5 Algorithm performance with respect to the algorithm output on the training image set.

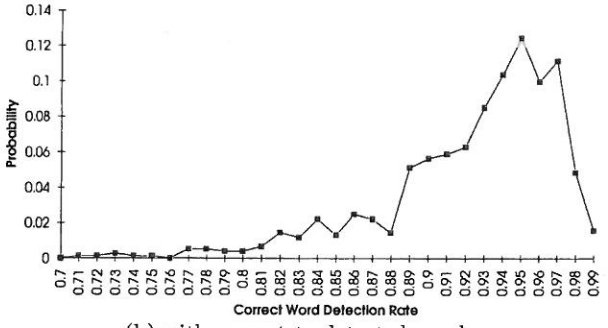
Total detected words	Correct	Split	Merged	False	Spurious
434,390	408741 (94.10%)	19196 (4.42%)	4940 (1.14%)	763 (0.18%)	750 (0.17%)

Table 7 Algorithm performance with respect to the algorithm output on the testing image set.

Total detected words	Correct	Split	Merged	False	Spurious
258,802	245584 (94.89%)	9290 (3.59%)	2961 (1.14%)	313 (0.12%)	654 (0.25%)



(a) with respect to ground truth words



(b) with respect to detected words

Fig. 8 Illustrations of the probability distributions of the corect word detection rate on real document images (T_p=0.95).

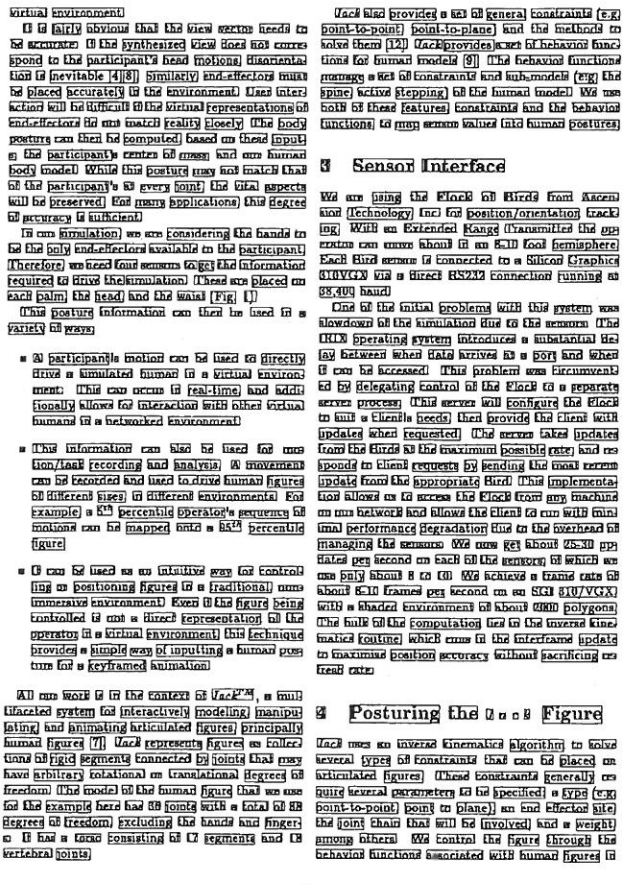


Fig. 9 Illustration of a synthetic document image overlaid with the extracted word bounding boxes.

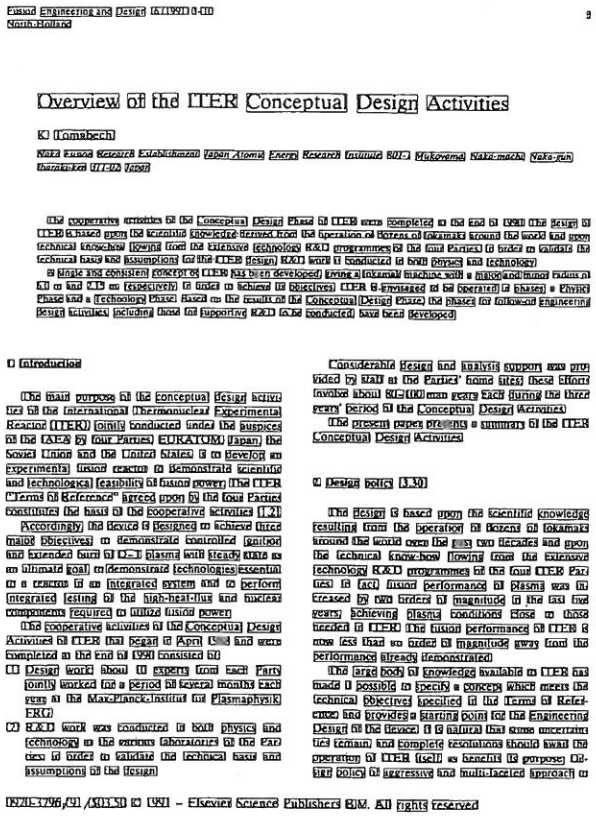


Fig. 10 Illustration of a real document image overlaid with the extracted word bounding boxes.

on document images. We developed and evaluated a word segmentation algorithm that is capable of simultaneously detecting all the words on a document image. The experimental results demonstrate that the correct word detection percentage is about 95% on synthetic document images and more than 90% on most of the real images.

As future work, we want to optimize the word segmentation algorithm on real document images, which may include training the algorithm on real document images and developing procedures to allow the word segmentation algorithm to predict the optimal threshold parameter T_p on the fly. A regression tree function can be constructed to predict the T_p^{opt} given the histogram of the posterior probability map image, similar to the process described in Ref. 11.

References

1. M. K. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," *IEEE Trans Patt. Anal. Mach. Intell.* **15** (7) (July 1993).
2. D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics and Image Processing* **47**, 327–352 (1989).
3. L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Trans. Patt. Anal. Mach. Intell.* **10** (6), 910–918 (1988).
4. L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Trans. Patt. Anal. Mach. Intell.* **15** (11) (Nov. 1993).
5. A. L. Spitz, "Text characterization by connected component transformations," *Proc. SPIE* **2181**, 97–105 (1994).
6. I. T. Phillips, S. Chen, and R. M. Haralick, "English Document Database Standard," *Proc. Second Int. Conf. Document Analysis and Recognition*, pp. 478–483, Japan, Oct. 20–22, 1993.

7. T. Kanungo, *DVI2TIFF User Manual, UW English Document Image Database–(I) Manual* (1993).
8. R. M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Reading, MA (1991).
9. J. Ha, I. T. Phillips, and R. M. Haralick, "Document page decomposition by the bounding-box projection techniques," *ICDAR'95: Third Intl. Conf. Document Analysis and Recognition*, Montreal, Canada, Aug. 14–16, 1995.
10. S. Chen and R. M. Haralick, "Recursive erosion, dilation, opening and closing transforms," *IEEE Trans. Image Processing* **4** (3) (March 1995).
11. S. Chen and R. M. Haralick, "An automatic algorithm for text skew estimation in document images using recursive morphological transforms," *Proc. ICIP-94: 1994 IEEE Int. Conf. Image Processing*, Vol. I, pp. 139–143, Austin, TX, Nov. 13–16, 1994.
12. 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* **20**, 375–390 (1982)
13. H. S. Baird, "Background structure in document images," in *Advances in Structural and Syntactic Pattern Recognition*, pp. 253–269, World Scientific, Singapore (1992).
14. T. Saitoh and T. Pavlidis, "Page segmentation without rectangle assumption," *Proc. 11th IAPR Intl. Conf. on Pattern Recognition*, pp. 277–280, Hague, Netherlands, Aug. 30–Sep. 3, 1192.
15. S. Chen, M. Y. Jaisimha, I. T. Phillips, and R. M. Haralick, *Reference Manual, UW English Document Image Database–(I) Manual*(1993).
16. R. M. Haralick, S. Chen, and T. Kanungo, "Recursive opening transform," *CVPR*, Champaign (June 1992).
17. S. Chen, R. M. Haralick, and I. T. Phillips, "Extraction of text layout structures on document images based on statistical characterization," *Proc. SPIE* **2242**, 128–139 (1995).
18. S. Chen, R. M. Haralick, and I. T. Phillips, "Perfect document layout ground truth generation using DVI files and simultaneous text word detection from document images," *Proc. 4th Annual Symp. Document Analysis and Information Retrieval*, Las Vegas (April 1995).
19. P. Maragos, "Pattern spectrum and multiscale shape representation," *IEEE Trans. Patt. Anal. Mach. Intell.* **11**, 701–716 (July 1989).
20. J. Woods, "Two-dimensional discrete Markov random fields," *IEEE Trans. Information Theory* **IT-18**, 232–240 (1972).

Su Chen received BS and MS degrees in electrical engineering from Tsinghua University, Beijing, China, in 1987 and 1989, respectively. In 1990, he joined the Intelligent Systems Laboratory at the University of Washington and obtained his PhD degree in electrical engineering in 1995. Dr. Chen has published more than two dozen articles in the general areas of computer vision, pattern recognition, and image processing. He also helped design and implement the UW-I, UW-II, and UW-III document image databases. His current research interests include mathematical morphology, document image analysis, natural language understanding, and information management and retrieval. Dr. Chen is currently affiliated with Caere Corporation, Los Gatos, California.

Robert M. Haralick is the Boeing Clairmont Egtvedt Professor in Electrical Engineering at the University of Washington. His recent work is in shape analysis and extraction using the techniques of mathematical morphology, robust pose estimation, techniques for making geometric inferences from perspective projection information, propagation of random perturbations through image analysis algorithms, and document analysis. Dr. Haralick served on the faculty of the Electrical Engineering Department at the University of Kansas as a professor from 1975 to 1978. In 1979 Dr. Haralick joined the Electrical Engineering Department at Virginia Polytechnic Institute and State University where he was a professor and director of the Spatial Data Analysis Laboratory. From 1984 to 1986, Dr. Haralick served as vice president of research at Machine Vision International, Ann Arbor, Michigan. He is a fellow of IEEE for his contributions in computer vision and image processing. He serves on the editorial board of *IEEE Transactions on Pattern Analysis and Machine Intelligence,* is an associate editor for the *IEEE Transactions on Image Processing,* and is an associate editor for *Pattern Recognition.* He received a BA in mathematics from the University of Kansas in 1964, a BS in electrical engineering in 1966, and a MS in electrical engineering in 1967. He completed his PhD at the University of Kansas in 1969.

Ihsin T. Phillips received her BS in 1979, MS in 1981, and PhD in 1984, all in computer science, from the University of Maryland, College Park. She is currently an associate professor in the Department of Computer Science and Software Engineering at Seattle University and an affiliate professor in the Department of Electrical Engineering at the University of Washington, Seattle, Washington. Her areas of research interest include digital image processing, pattern recognition, artificial intelligence, and software engineering. Her latest research interest is in the field of document image analysis and recognition. She has published many research articles and book chapters in this area. She has also been credited for the UW-I, UW-II, and UW-III document image databases.