Extraction of text layout structures on document images based on statistical characterization

Su Chen, Robert M. Haralick
University of Washington, Department of Electrical Engineering
Seattle, Washington 98195

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

ABSTRACT

The textual structures like the characters, words, text lines, paragraphs on a document image are usually laid out in a very structured manner – having preferred spatial relations. These spatial relations are rarely deterministic; instead, they describe correlations and likelihoods. Therefore, any realistic document layout analysis algorithm should utilize this type of knowledge in order to optimize its performances.

In this paper, we first describe a method for automatically generating a large amount of almost 100% correct ground truth data for the document layout analysis. The bounding boxes for the characters, words, text lines, paragraphs are expressed in a hierarchy. Then based on these layout ground-truth, we build statistical models to model the layout structures for the words, text lines, paragraphs on document images. Finally, we described an algorithm that utilize these statistical models to extract the text words on document images. The performance of the algorithm are evaluated and reported.

Keyword: Document image layout analysis, layout ground truth generation, recursive morphological closing transform, MAP classifier, performance evaluation, optimal parameter estimation.

1 INTRODUCTION

The document layout analysis identifies various objects of interest on a document image and describes their spatial relations. In our context, an object is defined as a homogeneous rectangular region that corresponds to one type: character, word, text line, paragraph text column or non-textual region.

Earlier work on document layout analysis can be categorically divided into two groups. One group employs the top-down or model-driven approach. It starts at the global image level and successively decomposes the image into smaller regions. Each region corresponds to either characters, words, text lines, paragraphs text
columns or non-textual regions. The other group adopts the bottom-up or data-driven approach\(^4\),\(^5\). It 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 paragraphs and paragraphs into columns, etc., until the whole document is completely labeled.\(^4\)

The main problems associated with these earlier techniques are that they were developed on a trial-and-error basis and although they provide illustrative results, hardly any have been tested on significant sized data sets. The main reason is the lack of substantial sized and accurate document layout ground truth data to train and test the algorithms. It is clear that if we want to bring the computer vision research into the next higher level, any techniques that we develop must be proven out on substantial sized data sets.

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 hard to derive, it is clear that suitable performance measures not only facilitate us to construct a system that optimizes its performance measures given the training data set, but also enable us to predict the system performance based on the testing data set.

Section 2 describes a technique to automatically create a large amount of accurate ground truth data suitable for the development of document layout analysis algorithms. Section 3 describes a word segmentation algorithm using the recursive morphological closing transform and a MAP classifier. Section 5 discusses an experimental protocol to train and evaluate the word segmentation algorithm given a ground-truthed document image population. Finally, Section 5, describes our experimental results.

## 2 DOCUMENT LAYOUT GROUND TRUTH GENERATION

Document layout analysis algorithms typically decompose a document image into zones. A zone may correspond to a text block (usually a paragraph) or a figure. A text zone may contain a list of text lines; a text line may include a sequence of detached words; and a word may in turn consist of a string of characters. Therefore, the document layout ground truth data that are used for the training and testing of the document layout analysis algorithms should specify this hierarchy.

The “UW English Document Image Database (1)\(^6\)” is a data set for the OCR and the document image understanding algorithm developments and evaluations. The database has a software to convert a DVI file from the \(\LaTeX\) document processing system into bitmap images. 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. The same software also generates a so-called character ground truth file for each of the document images. The file contains the bounding box coordinates, the type and size of the font, and the ASCII code for every individual character in the image.

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.

### 2.1 Notation and assumption

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

Our assumption on the character bounding box sequence $C$ is that it follows the same order as the logical read order of the characters on the document image. The characters are laid out in such a way that they first go sequentially along one direction and then at the end of the current text line, start at a different location to begin another text line. We assume that spacings between two adjacent characters follow different probability distributions for the character breaks, the word breaks and the text line breaks. Normally, the character break spacings are significantly less than the word break spacings and the word break spacings are significantly less 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 $C$. 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 distances 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. $w$ 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 the other for text line breaks. The first two peaks are relatively stronger.

3. Text line extraction: If $\rho(c_i, c_{i+1}) > T_3$, then the break between $c_i$ and $c_{i+1}$ is a text line break. $T_3 = \alpha S$, where $S$ is the dominant character font size and $\alpha$ 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 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. $T_1$ is called the within-word character spacing threshold. 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 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 thresholding algorithm. 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 ($1$), a zone bounding box is not necessarily the minimum zone bounding box that encloses the content of the zone, therefore, the zone bounding box is modified so that it is the minimum bounding box that encloses all the text lines assigned to the zone.

2.3 Example ground truth layout data

We tested our algorithm on the 168 synthetic images from the "UW English Document Image Database ($1$)". The algorithm performed almost perfectly on all the images except on some of the displayed math zones where the
placement of subsequent symbols violates our underlying assumptions (Section 2.1). In this situation, the usual definitions of text lines and words are no longer valid. It is not of our current interest to provide the accurate layout ground truth for displayed math zones. For the 168 synthetic images, there are a total of 1366 text zones and 243 displayed math zones. There are a total about 10,000 text lines and 60,000 words.

To ensure that the above automatic procedure works correctly on all the 168 synthetic images, we actually displayed each document image overlaid with the zone, text line and word bounding boxes and checked if there were any errors. On all the images, we found merely 4 or 5 locations where two adjacent text lines are merged together. The scenarios were that the next text line started immediately below the end of the previous text line. After giving a larger weight to the vertical distance parameter, the algorithm generated the correct text lines automatically.

Figure 1 illustrates one synthetic document image. Figure 2 gives the generated zone, text line, word and character layout ground truth data.

Figure 1: illustrates an example document image.

3 WORD SEGMENTATION

Our approach to the document word segmentation is based on the recursive closing transform. The recursive closing transform provides an adequate tool for the image shape analysis in the image background, especially when the scale of the shape is a factor. The recursive closing transform efficiently computes the binary morphological closings with respect to all sized structuring elements simultaneously. Hence, it provides a characterization of image shapes at all scales. It is extremely 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 section, we will give a short overview of the transform.

The prominent characteristics of the current word segmentation algorithm are summarized as follows: 1) It is un-like most of the top-down or bottom up approaches where the objects of interest are derived in a recursive fashion. Our word segmentation is a one step and simultaneous process. 2) The algorithm is not sensitive to text skews because only the local shape information is used. Texts can be laid out in both the horizontal and the vertical directions at the same time. 3) The algorithm 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. 4) The algorithm is trainable to any given document image population.
3.1 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 \( z \in Z^2 \) is defined as the smallest positive integer \( n \) so that \( z \in I \circ (\Theta_{n-1}K) \). If no such \( n \) exists, where \( z \notin I \circ (\Theta_{n-1}K) \) for all \( n \), then the closing transform at \( z \in Z^2 \) is defined to be zero.

**Definition 3.1.** 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:

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

Figure 2: 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.
An efficient recursive closing transform (RCT)\(^9\) is developed to compute in constant time per pixel the closing transform of a binary image.

### 3.2 System overview

In this section, an algorithm for the word segmentation on document images is described. The algorithm first sub-samples the input document image and then detects the block areas that correspond to words. The word block detection is based on the recursive closing transform described in\(^8\).\(^9\) Each of the detected word block areas is then modeled as an 8-connected connected component. The bounding box of each of the connected components is computed. As a final step, the algorithm performs a hypothesis test on the heights of the detected word blocks to handle merging words among adjacent text lines. The various components of the word segmentation algorithm are described next:

#### Sub-sampling

Assume that our input document images are scan-digitized at a spatial resolution of 300dpi. For a standard 11" \(\times\) 8.5" page, it is equivalent to an input document image size of 3300 \(\times\) 2550. To process such an image, it will take more memory and processing time. Our strategy to overcome this problem is to use sub-sampling. A sampling algorithm is used to reduce the spatial resolution to 150dpi. Figure 3 (a) illustrates one segment of the sub-sampled 150dpi image.

#### Word block detection

The word block detection is based on the recursive closing transform. The recursive closing transform is useful in extracting shape information in the image background (white-space).

Let \(K_1, K_2, \ldots, K_n\) denote \(n\) structuring elements. Let \(y_1 = CT[I, K_1](x), y_2 = CT[I, K_2](x), \ldots, y_n = CT[I, K_n](x)\) denote the values of the closing transform at pixel \(x \in I\) with respect to the structuring elements \(K_1, K_2, \ldots, K_n\). Let \(y = (y_1, y_2, \ldots, y_n)\). Then each pixel in the image \(I\) is modeled as a random observation data vector \(Y = y\). Furthermore, each pixel has an associated label \(L = l\). For the word block detection, the label could be either word (\(L = 1\)) or non-word (\(L = 0,\) white-space). A pixel is defined to be a word pixel if and only if it is on or inside the bounding box of a word. A pixel is defined to be a non-word pixel if it is outside the bounding boxes of all words.

The word block detection algorithm first compute the posterior probability for each pixel being a word pixel \(P(L = 1 \mid Y = y)\). Then the posterior probability for each pixel to be a non-word pixel is equal to \(P(L = 0 \mid Y = y) = 1 - P(L = 1 \mid Y = y)\). The output of this step is called the posterior probability map image. The posterior probability distribution is estimated during the initial experimental phase. In the experiment, we choose \(n = 3\) and \(K_1\) is a horizontal \(1 \times 2\) structuring element, \(K_2\) is a vertical \(2 \times 1\) structuring element, \(K_3\) is a \(2 \times 2\) square structuring element.

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 zero-height flat structuring element \(S\). We select \(S\) to be a \(2 \times 2\) square structuring element. Figure 3 (b) illustrates one segment of the correlated posterior probability map image.

Finally, the correlated probability map image is thresholded to output the binary word block image. Input...
pixels that have values greater than or equal to $T_p$ output a binary one value. The reasonable range of the threshold $T_p$ is 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 3 (c) illustrates one segment of the detected word block image, where $T_p = 0.96$.

Figure 3: Illustrates the word segmentation process. (a) sub-sampled 150dpi image; (b) correlated posterior probability map image; (c) thresholded word block image; (d) word bounding boxes.

Word bounding box extraction

Each detected word block is modeled as an 8-connected connected component. The connected component labeling procedure described in $^{10}$ is performed on the binary word block image. The bounding box of each of the connected components is calculated. Figure 3 (d) illustrates one segment of the sub-sampled image overlaid with the extracted word bounding boxes.

Hypothesis test on word height

The presence of the character ascenders and.descenders sometimes causes the merging of many word blocks from two or more adjacent text lines into one big block. In order to automatically detect such cases and consequently split the merged word blocks into their corresponding correct words, we developed a simple post-processing
procedure to perform a hypothesis test 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 $\beta W_h$ could be split further, where $\beta$ is a real constant and has a default value of $\beta = 2.0$.

Then, for each word block which is hypothesized to be divided further, the algorithm will verify it by computing intervals of cut points in the projection profile of the posterior probability map image along the height direction 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 $f(h) = \frac{1}{W} \sum_{w=1}^{W} P(h, w)$, 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 has a default value of $T_c = 0.5$. 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 zero-height flat structuring element of size $W_h/2$, denoted by $k_1$. This will remove the narrow up-shoot spikes in $f(h)$. Let $f_1 = f \circ k_1$.

2. Morphologically close $f_1(h)$ by a zero-height flat structuring element of size $D_m$, denoted by $k_2$. This will bridge the narrow valleys in $f_1(h)$ and ensure that the cut points are at least $D_m$ wide. We select the default $D_m = 5$. Let $f_2 = f_1 \circ k_2$.

3. Morphologically erode $f_2(h)$ by a zero-height flat structuring element of size $W_h$, denoted by $k_3$. Let $f_3 = f_2 \circ k_3$. Then the set of possible cut points is defined as the set $\text{cut} = \{ h \in [1, H] \mid f_2(h) \leq T_c \}$ and $f_3(h) = f_3(h)$, 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 $T_c$. We use a binary function $f_4$ to characterize the set $\text{cut}$, i.e., $f_4(h) = 1$ if and only if $h \in \text{cut}$. □

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 end-points ($h = 1$ and $h = H$) is greater than zero, then the word block needs to be split further at those intervals and the bounding boxes of the sub-word blocks are re-computed.

## 4 EXPERIMENTAL PROTOCOL

In the previous section, we outlined a word segmentation algorithm. The algorithm requires the posterior probability $P(L = 1 \mid Y = y)$ 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.

### 4.1 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 Section 2. To compute the posterior probability distribution, we first generate a so-called word mask image for each of the 168 document
images. The word mask image is bi-level 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 $0^\circ$, $\pm 0.2^\circ$, $\pm 0.4^\circ$, $\pm 0.6^\circ$, using a nearest neighbor interpolation algorithm. The range of rotation angles is selected because our skew estimation algorithm is capable of detecting text skew angles on document images which are within $0.5^\circ$ of the true text skew angles at a probability of 99%. This generates a total input training image population of $1176 = 168 \times 7$ images. Each image is of size $1650 \times 1275$.

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

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

The joint probability distributions can be substituted with the frequency counts $\#(L = 0, Y = y)$ and $\#(L = 1, Y = y)$. The counting processes are simplified in our case because the observation vectors $Y = (y_1, y_2, y_3)$ are integer vectors and bounded within the 3-dimensional cube $[0, N] \times [0, N] \times [0, N]$, where $N$ is the allowed maximum output integer value of the closing transform. For word segmentation, we choose $N = 63$.

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

### 4.2 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 Section 2. Let $G = \{G_1, G_2, \ldots, G_N\}$ represent the total of $N$ ground truth word bounding boxes and let $D = \{D_1, D_2, \ldots, D_M\}$ 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 $G$ and $D$, establish the element mappings between the two sets and report the numbers of miss detections (1-0 mappings), false detections (0-1 mappings), correct detections (1-1 mappings) and splitting detections (1-m mappings), merging 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 $g : G \rightarrow D$ and $d : D \rightarrow G$:

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

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

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

Based on the two functions \( g : G \rightarrow D \) and \( d : D \rightarrow G \), we could establish mappings between the elements of \( G \) and \( D \). The rules are summarized 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 miss detection (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 detection.
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 splitting 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 \( D \neq D_0, d(D) = \emptyset \); 3) For all \( D \notin g(G_i), G_i \notin d(D) \).
5. There is a merging 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 \( G \neq G_0, g(G) = \emptyset \); 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).

Let \( N_{10}, N_{01} \) and \( N_{11} \) be the numbers of miss, false and correct detections, respectively. Let \( N_{1m}^f, N_{m1}^f \) and \( N_{mm}^f \) denote the numbers of words in the \( G \) that have the 1-m, m-1 and m-m mappings with words in the \( D \). Similarly, let \( N_{1m}^d, N_{m1}^d \) and \( N_{mm}^d \) denote the numbers of words in the \( D \) that have the 1-m, m-1 and m-m mappings with words in the \( G \). Then the following relations satisfy: 1) \( N = N_{10} + N_{11} + N_{1m}^f + N_{m1}^f + N_{mm}^f; \) 2) \( M = N_{01} + N_{11} + N_{1m}^d + N_{m1}^d + N_{mm}^d; \) 3) \( N_{1m}^d \leq N_{1m}^f; \) 4) \( N_{m1}^f \geq N_{m1}^d \). The performance of the word segmentation algorithm can be measured through a goodness function. Let it be denoted as \( \kappa \). It is defined by:

\[
\kappa = \min(\kappa_1, \kappa_2)
\]

where

\[
\kappa_1 = (\gamma_{10}N_{10} + \gamma_{11}N_{11} + \gamma_{1m}N_{1m}^f + \gamma_{m1}N_{m1}^f + \gamma_{mm}N_{mm}^f)/N
\]

\[
\kappa_2 = (\gamma_{01}N_{01} + \gamma_{11}N_{11} + \gamma_{1m}N_{1m}^d + \gamma_{m1}N_{m1}^d + \gamma_{mm}N_{mm}^d)/M
\]

and the \( \gamma_{10}, \gamma_{01}, \gamma_{11}, \gamma_{1m}, \gamma_{m1} \) and \( \gamma_{mm} \) are economic gain coefficients for the miss, false, correct, splitting, merging and spurious detections. The larger the goodness measure \( \kappa \), the better the performance of the word segmentation algorithm. In the experiment, we choose the economic gain coefficients as in Table 1:

<table>
<thead>
<tr>
<th>( \gamma_{10} )</th>
<th>( \gamma_{11} )</th>
<th>( \gamma_{1m} )</th>
<th>( \gamma_{m1} )</th>
<th>( \gamma_{mm} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.6</td>
<td>1.0</td>
<td>0.9</td>
<td>0.4</td>
</tr>
</tbody>
</table>

4.3 Optimal threshold determination

In the word segmentation algorithm, there is a threshold value \( T_0 \) 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.
Given an input document image, $\kappa$ 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 $T_p^{opt}$ denote the optimal threshold value. Then,

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

# 5 EXPERIMENTAL RESULTS

## 5.1 Performance on the training image population

To benchmark the optimal performance of our word segmentation algorithm, we tested the algorithm on the 1176 training document images described in Section 4.1 under the optimal threshold setting $T_p = T_p^{opt}$. Table 2 and Table 3 illustrate the percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth 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 formula (displayed or inline) are not accurate (Section 2.3). Of the 429,338 ground truth words, 95.2026% of them are correctly detected.

<table>
<thead>
<tr>
<th>Total Ground Truth Words</th>
<th>Correct</th>
<th>Splitting</th>
<th>Merging</th>
<th>Miss</th>
<th>Spurious</th>
</tr>
</thead>
<tbody>
<tr>
<td>429338</td>
<td>408741</td>
<td>14440</td>
<td>10281</td>
<td>325</td>
<td>420</td>
</tr>
<tr>
<td>(95.2026%)</td>
<td>(2.9884%)</td>
<td>(2.3680%)</td>
<td>(2.3976%)</td>
<td>(0.0750%)</td>
<td>(0.1910%)</td>
</tr>
</tbody>
</table>

Table 2: Algorithm performance with respect to the ground truth on the training image set.

<table>
<thead>
<tr>
<th>Total Detected Words</th>
<th>Correct</th>
<th>Splitting</th>
<th>Merging</th>
<th>False</th>
<th>Spurious</th>
</tr>
</thead>
<tbody>
<tr>
<td>434390</td>
<td>433761</td>
<td>18165</td>
<td>15537</td>
<td>762</td>
<td>762</td>
</tr>
<tr>
<td>(99.3314%)</td>
<td>(4.4159%)</td>
<td>(3.5121%)</td>
<td>(3.5973%)</td>
<td>(0.1756%)</td>
<td>(0.1757%)</td>
</tr>
</tbody>
</table>

Table 3: Algorithm performance with respect to the algorithm output on the training image set.

## 5.2 Performance on the test image population

To assess the optimal performance of the algorithm on new document images, we first prepared a new set of 96 LaTeX document pages, created the corresponding TIFF images and the ground truth word bounding boxes using the programs described in Section 2. Then each of the 96 document images and its corresponding ground truth word bounding boxes are rotated at various degrees of $0^\circ$, $\pm 0.2^\circ$, $\pm 0.4^\circ$, $\pm 0.6^\circ$. These become a total of 672 test document images.

Under the optimal threshold settings ($T_p = T_p^{opt}$), Table 4 and Table 5 illustrate the percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth as well as the algorithm output. Of the 258,328 ground truth words, 95.0967% of them are correctly detected. On the other hand, of the 258,802 words detected by the algorithm, 94.8926% of them are correctly detected as the ground truth words. The evaluation does not exclude the word boxes from from displayed mathematical formula. The performance of the word segmentation algorithm on the test document images is not significantly different from that on the training document images because the training set is sufficiently large.

138 / SPIE Vol. 2422
Table 4: Algorithm performance with respect to the ground truth on the test image set.

<table>
<thead>
<tr>
<th>Total Ground Truth Words</th>
<th>Correct</th>
<th>Splitting</th>
<th>Merging</th>
<th>Mix</th>
<th>Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Detected Words</td>
<td>255328</td>
<td>248584</td>
<td>676</td>
<td>7123</td>
<td>864</td>
</tr>
<tr>
<td></td>
<td>(100.00%)</td>
<td>(97.07%)</td>
<td>(20.34%)</td>
<td>(27.82%)</td>
<td>(0.327%)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Total Detected Words</th>
<th>Correct</th>
<th>Splitting</th>
<th>Merging</th>
<th>False</th>
<th>Spacing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Detected Words</td>
<td>255328</td>
<td>248584</td>
<td>676</td>
<td>7123</td>
<td>864</td>
</tr>
<tr>
<td></td>
<td>(100.00%)</td>
<td>(97.07%)</td>
<td>(20.34%)</td>
<td>(27.82%)</td>
<td>(0.327%)</td>
</tr>
</tbody>
</table>

6 CONCLUSIONS

We presented an automatic method for generating a large amount of accurate document layout ground truth data from \LaTeX files. The generated layout ground truth data is then used to train and evaluate a word segmentation algorithm which is capable of simultaneously detecting all the words on a document image and is trainable to any given document image population. We described an experimental protocol on how to train and evaluate the word segmentation algorithm. The experimental results demonstrate that under the optimal algorithm parameter settings, the correct word detection percentage is about 95% on both training and testing document images (a total of about 600,000 words). We achieve this performance even with the presence of some small amount of skews.

7 REFERENCES


