Perfect Document Layout Ground Truth Generation Using DVI Files and Simultaneous Word Segmentation From Document Images

Su Chen[†], Robert M. Haralick[†] and Ihsin T. Phillips[‡]
[†] Department of Electrical Engineering
University of Washington
Seattle, Washington 98195
chen@george.ee.washington.edu
[‡] Department of Computer Science
Seattle University
Seattle, Washington 98122

Abstract

In this paper, we first describe an automatic technique to efficiently generate a large amount of accurate ground truth data suitable for the development of document layout analysis algorithms. Then we describe a new word segmentation algorithm that is based on the recursive morphological closing transform. The algorithm is trainable for any given document image population and is capable of detecting all the words on a document image simultaneously. We discuss an experimental protocol to train and evaluate the word segmentation algorithm. The experimental results demonstrate that under the optimal algorithm parameter settings, the dominant correct word detection percentage is about 95% on both the training and testing image populations.

1 Introduction

Document image analysis decomposes an input document image into its various components, i.e. its content, layout structure and logical structure. The *content* is the informational part of the document, such as the text strings. The *layout structure* specifies the physical embodiment of the content on the document image, such as its appearance and location. The *logical structure* names the content-bearing parts of the document and specifies their logical relationships, such as the reading order. The three levels of decomposition are usually done separately through the processes of optical character recognition (OCR), layout analysis, and logical analysis.

This paper focuses on the document layout analysis process, which 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 [2] [3]. It starts at the global image level and successively decomposes the image into smaller regions. Each region has one type: character, word, text line, paragraph, text column, or non-textual region. Nagy [2] and Srihari [3] 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 only in 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 sub-blocks by making cuts in the horizontal profile corresponding to troughs of depth and width greater than some threshold. Each resulting sub-block 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, paragraph, text line, word, or character decomposition are inherently different. On the other hand, there are occasions when such model selections are not the direct correspondences between the object types and the levels of decomposition. 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 the case when there is noise present on the document image.

The other approach is bottom-up or data-driven [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 technique is based on a connected component analysis. A connected component is a set of binary one (zero) pixels in a binary image which are either 4-connected or 8-connected. The algorithm assumes that each connected component in the image corresponds to one text character or one non-textual 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 sensitive to touching characters and fragmented characters because the underlying connectivity assumptions are violated. For some types of document images where texts are printed in the dot matrix form, the algorithm breaks down completely.

Besides the above shortcomings, most of the earlier techniques were developed on a trial-and-error method. Little effort was placed on systematically evaluating the performance of the document layout analysis algorithms. The main reason is the lack of accurate document layout ground truth data to train and test the algorithms.

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. Section 4 discusses one experimental protocol to train and evaluate the word segmentation algo-

rithm 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 (I)" [8] is a data set for OCR and document image understanding algorithm development and evaluation. The database has software to convert a DVI file from the LaTeX document processing system into bitmap images [10]. 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 $\mathcal{I}$. Let $\mathcal{Z} = \{z_1, z_2, \cdots, z_k\}$ 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 $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$, 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. Normally, the character break spacings are smaller than the word break spacings, and the word break 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:

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

   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 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 (more populated).

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. $T_2 = \alpha S$, 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 $T_1 < \rho(c_i, c_{i+1}) \leq T_2$, 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 [15]. 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 Example document layout ground truth data

We tested our algorithm on the 168 synthetic images from the "UW English Document Image Database (I)" [8]. The algorithm performed well on all the images except on some of the displayed math formula 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. But since it is not our purpose to provide accurate layout ground truth for displayed math zones, we ignore these cases. 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 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 segmentation automatically.

As an example, Figure 1 illustrates one of the synthetic document images. Figure 2 gives the generated zone, text line, word and character layout ground truth data.

In conclusion, the technique described here provides an efficient and automatic way of creating a large amount of accu-

rately ground truthed layout ground truth data for the development and evaluation of document layout analysis algorithms. The rest of this paper will use the ground truth generated by this technique to to develop, train, and test a word segmentation algorithm that is capable of determining the bounding boxes for all words on a document image.

3 Word segmentation using recursive closing transform

Our approach to document word segmentation is based on the recursive closing transform. The recursive closing transform provides an efficient way of computing the binary morphological closings with respect to all sized structuring elements simultaneously. It is a very powerful morphological tool for image shape analysis, especially when the scale of the shape is a factor. 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. Section 3.1 contains a short overview of the transform. For details, please refer to [12] [13].

The prominent characteristics of the current word segmentation algorithm are summarized as follows:

- Most of the top-down or bottom up approaches derive the objects of interest in a recursive fashion. Our word segmentation is a one step and simultaneous process.
- The algorithm 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.
- 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.
- The algorithm is trainable to any given document image population.
- The same methodology for the word segmentation is directly applicable to both the text line and the character segmentations.

3.1 Recursive closing transform

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 $x \in I \bullet (\oplus_{n-1} K)$. 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 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:*

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

In [13], an efficient recursive closing transform (RCT) was 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 [12] [13]. 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 merged words from 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 page, this is equivalent to an input document image size of 3300×2550. To process such an image, it will take more memory and processing time. Our strategy to overcome this problem is to use a 2:1 sub-sampling and process a 150dpi image.

The sub-sampling algorithm that we have implemented is as follows: let the horizontal and vertical sub-sampling ratio be H and V, respectively. Given an input $R \times C$ bi-level image, the algorithm generates an output bi-level 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 non-overlapping $V \times H$ 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 pre-specified threshold T, its corresponding output pixel is set to binary one; otherwise, it is set to binary zero. To obtain a 150dpi sub-sampled image, we select $H = 2, V = 2$ and $T = 2$. 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). Maragos [16] indicated that image shapes can be characterized through the pattern spectrum. The recursive closing transform provides an efficient way to calculate the pattern spectrum of the image background. The pattern spectrum is nothing more than the histogram of the closing transform.

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 $K_1, K_2, \cdots, K_n$. Let $y = (y_1, y_2, \cdots, y_n)$. Then each pixel in the image I is modeled as a random observation data vector $\mathcal{Y} = y$. Furthermore, each pixel has an associated label $\mathcal{L} = l$. For the word block detection, the label could be either word ($\mathcal{L} = 1$) or non-word ($\mathcal{L} = 0$, 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 assigns a posterior probability $P(\mathcal{L} = 1 \mid \mathcal{Y} = y)$ to each pixel. The output of this step is a posterior probability map image. The posterior probability functions are estimated during the initial experimental stage. In the experiment, we choose $n = 3$ and K_1 to be a horizontal 1×2 structuring element, K_2 to be a vertical 2×1 structuring element, and K_3 to be a 2×2 square structuring element.

To introduce the correlation among the neighboring pixels in the probability map image, we morphologically close and then open the map image by a zero-height flat structuring element S. We select S to be a 2×2 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. A reasonable range for 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 $T_p = 0.96$.

Word Bounding Box Extraction

Each detected word block is modeled as an 8-connected connected component. The connected component labeling procedure described in [15] 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 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 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 $\beta = 2.0$. For each word block which 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 $f(h) = \frac{1}{W} \sum_{w=1}^{W} P[h, w]$, 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 T_c has a default value of 0.5. If the number of such detected cut points other than the two end-points ($h = 1$ 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 zero-height 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 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 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 zero-height 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 $\{h \in [1, H] \mid f_2(h) \leq T_c \text{ and } f_2(h) = f_3(h)\}$, 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. □

Once the cut points are located, the input word block is split at the cut points and the bounding boxes of the sub-word blocks are re-computed.

Figure 4 plots the word height and width probability distributions among the 168 document images described in Section 2.3. The document images were subsampled at a spatial resolution of 150dpi. From the figure, we observed that the dominant word height is $W_h = 15$, which is equivalent to a word height of about 7-8 points (1 point $\approx 1/72$ of an inch).

4 Experimental protocol

In the previous section, we outlined a word segmentation algorithm. The algorithm requires the posterior probability $P(\mathcal{L} = 1 \mid \mathcal{Y} = y)$ 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 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°, ±0.2°, ±0.4°, ±0.6°, 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° 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 $P(\mathcal{L} = 1 \mid \mathcal{Y} = y)$:

$$\begin{aligned} & P(\mathcal{L} = 1 \mid \mathcal{Y} = y) \\ &= \frac{P(\mathcal{L} = 1, \mathcal{Y} = y)}{P(\mathcal{Y} = y)} \\ &= \frac{P(\mathcal{L} = 1, \mathcal{Y} = y)}{P(\mathcal{L} = 0, \mathcal{Y} = y) + P(\mathcal{L} = 1, \mathcal{Y} = y)} \end{aligned}$$

The joint probability distributions can be substituted with the frequency counts $\#(\mathcal{L} = 0, \mathcal{Y} = y)$ and $\#(\mathcal{L} = 1, \mathcal{Y} = y)$. The counting processes are simplified in our case because the observation vectors $\mathcal{Y} = (y_1, y_2, y_3)$ 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 [13]. For word segmentation, we choose $N = 63$.

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

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 $\mathcal{G} = \{G_1, G_2, \cdots, G_N\}$ represent

the total of N ground truth word bounding boxes and let $\mathcal{D} = \{D_1, D_2, \cdots, D_M\}$ 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 $\mathcal{G}$ and $\mathcal{D}$. 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 $s(A, B)$:

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

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 $g : \mathcal{G} \rightarrow \mathcal{D}$ and $d : \mathcal{D} \rightarrow \mathcal{G}$:

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

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

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

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

1. If there exists a G_i such that $s(G_i, D_j) = 0$ for all $j = 1, 2, \cdots, M$, 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 $i = 1, 2, \cdots, N$, then the D_j is counted as a false detection (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 splitting detection (1-m mapping) between G_i and $\{D_{j_1}, D_{j_2}, \cdots, D_{j_m}\}$ if and only if, 1) $g(G_i) = \{D_{j_1}, D_{j_2}, \cdots, D_{j_m}\}$; 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 $\{G_{i_1}, G_{i_2}, \cdots, G_{i_m}\}$ and D_j if and only if, 1) $d(D_j) = \{G_{i_1}, G_{i_2}, \cdots, G_{i_m}\}$; 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). □

Once the element mappings between $\mathcal{G}$ and $\mathcal{D}$ has been established, we count the numbers of miss, false, correct, splitting, merging and spurious detections. Let N_{10}, N_{01} and N_{11} be the numbers of miss, false and correct detections, respectively. Let N^g_{1m}, N^g_{m1} and N^g_{mm} denote the numbers of words in the $\mathcal{G}$ that have the 1-m, m-1 and m-m mappings with words in the $\mathcal{D}$. Similarly, let N^d_{1m}, N^d_{m1} and N^d_{mm} denote the numbers of words in the $\mathcal{D}$ that have the 1-m, m-1 and m-m mappings with words in the $\mathcal{G}$. 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) $N^g_{1m} \leq N^d_{1m}$; 4) $N^g_{m1} \geq N^d_{m1}$.

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 miss, false, correct, splitting, merging 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 1.

4.3 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 5 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 $Prob[\kappa \geq \kappa_0]$, 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 $[0.5, 1.0]$.

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 $T_p = T^{opt}_p$. 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. 1.9658% and 2.5530% of the words are split or merged, respectively. The total miss and spurious detections account for about 0.3% of the total ground truth words. On the other hand, of the 434,390 words detected by the algorithm, 94.0954% of them are correctly detected as the ground truth words. There are 4.4191% and 1.1372% of the detected words are derived from either split or merged ground truth words, respectively. The total false and spurious detections account for about 0.3% of the total algorithm output.

5.2 Performance on the testing image population

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 Section 2. Each of the 96 document images and its corresponding ground truth word bounding boxes are further rotated at various degrees of $0°$, $\pm 0.2°$, $\pm 0.4°$, $\pm 0.6°$. This generates a total of 672 testing 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, 95.0667% of them are correctly detected. There are 1.6015% and 2.7573% of the words are split or merged, respectively. The total miss and spurious detections account for less than 0.6% of the total ground truth words. 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. There are 3.5896% and 1.1441% of the detected words are derived from either split or merged ground truth words, respectively. The total false and spurious detections account for less than 0.4% 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.

Figure 6 and Figure 7 illustrate two examples of the word segmentation results. From the images, we observe that the word segmentation algorithm performs almost equally well on synthetic and real document images. The whole process takes about 30 seconds per image on a Sun Sparc 10 workstation.

6 Conclusions and future work

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.

We are currently developing procedures to make the word segmentation algorithm fully automatic so that it is capable of predicting the optimal threshold parameter T_p on a per image basis. 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 [11]. Our future work will also include extending the current word segmentation technique to the text line and zone segmentation.

References

[1] 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.

[2] M.K. Krishnamoorthy, G. Nagy, S. Seth and M. Viswanathan, "Syntactic Segmentation and Labeling of Digitized Pages from Technical Journals", *IEEE Trans. on PAMI*, Vol. 15, No. 7, July 1993.

[3] D. Wang and S.N. Srihari, "Classification of newspaper image blocks using texture analysis", *Computer Vision, Graphics and Image Processing*, 47:327-352, 1989.

[4] L.A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images", *IEEE Trans. on PAMI*, Vol. 10, No. 6, 1988, pp. 910-918.

[5] A.L. Spitz, "Text Characterization by Connected Component Transformations", *Proc. of the SPIE*, vol.2181. pp. 97-105. 1994.

[6] H.S. Baird, "Background structure in document images", *Advances in Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, pp 253-269.

[7] T. Saitoh, T. Pavlidis, "Page Segmentation without Rectangle Assumption", *Proc. of 11th IAPR Int'l Conf. on Pattern Recognition*, Hague, Netherlands, Aug 30 - Sep 3, 1992, pp. 277-280.

[8] 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.

[9] S. Chen, M.Y. Jaisimha, J. Ha, I.T. Phillips and R.M. Haralick, *Reference Manual*, 1993. UW English Document Image Database - (I) Manual.

[10] T. Kanungo, *DVI2TIFF User Manual*, 1993. UW English Document Image Database - (I) Manual.

[11] S. Chen and R.M. Haralick, "An automatic algorithm for text skew estimation in document images using recursive morphological transforms", *ICIP*, Austin, November, 1994.

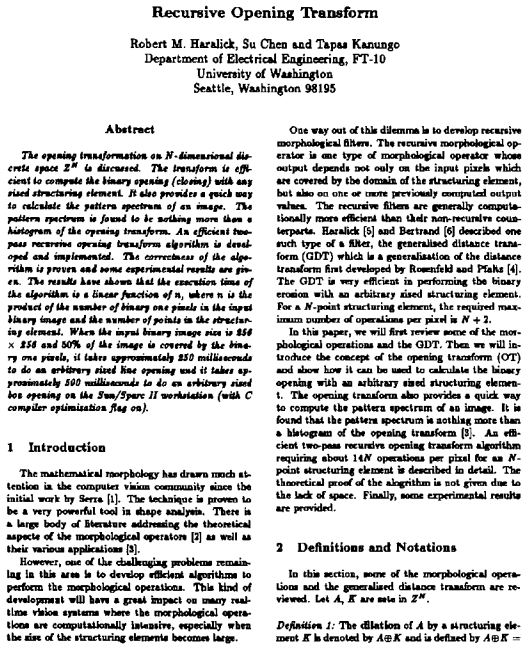
[12] R.M. Haralick, S. Chen and T. Kanungo, "Recursive Opening Transform", *CVPR, Champaign*, June 1992.

[13] S. Chen and R.M. Haralick, "Recursive erosion, dilation, opening and closing transforms", *IEEE Trans on Image Processing*, Vol.4, No. 3, March 1995.

[14] S. Chen, R.M. Haralick and I.T. Phillips, "Extraction of text layout structures on document images based on statistical characterization", *Proc. of IS&T / SPIE Symposium on Electronic Imaging: Science and Technology*, San Jose, February, 1995.

[15] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, 1991.

[16] P. Maragos, "Pattern Spectrum and Multiscale Shape Representation", *IEEE Transactions on PAMI*, Vol. 11, No. 7, July 1989, pp. 701-716.



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 × 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 generalized distance transform (GDT) which is a generalization of the distance transform first developed by Rosenfeld and Pfaltz [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 algorithm 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 generalized 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 $A \oplus K =$

Figure 1: Illustrates an example document page.

Su Chen, Robert M. Haralick and Ihsin T. Phillips

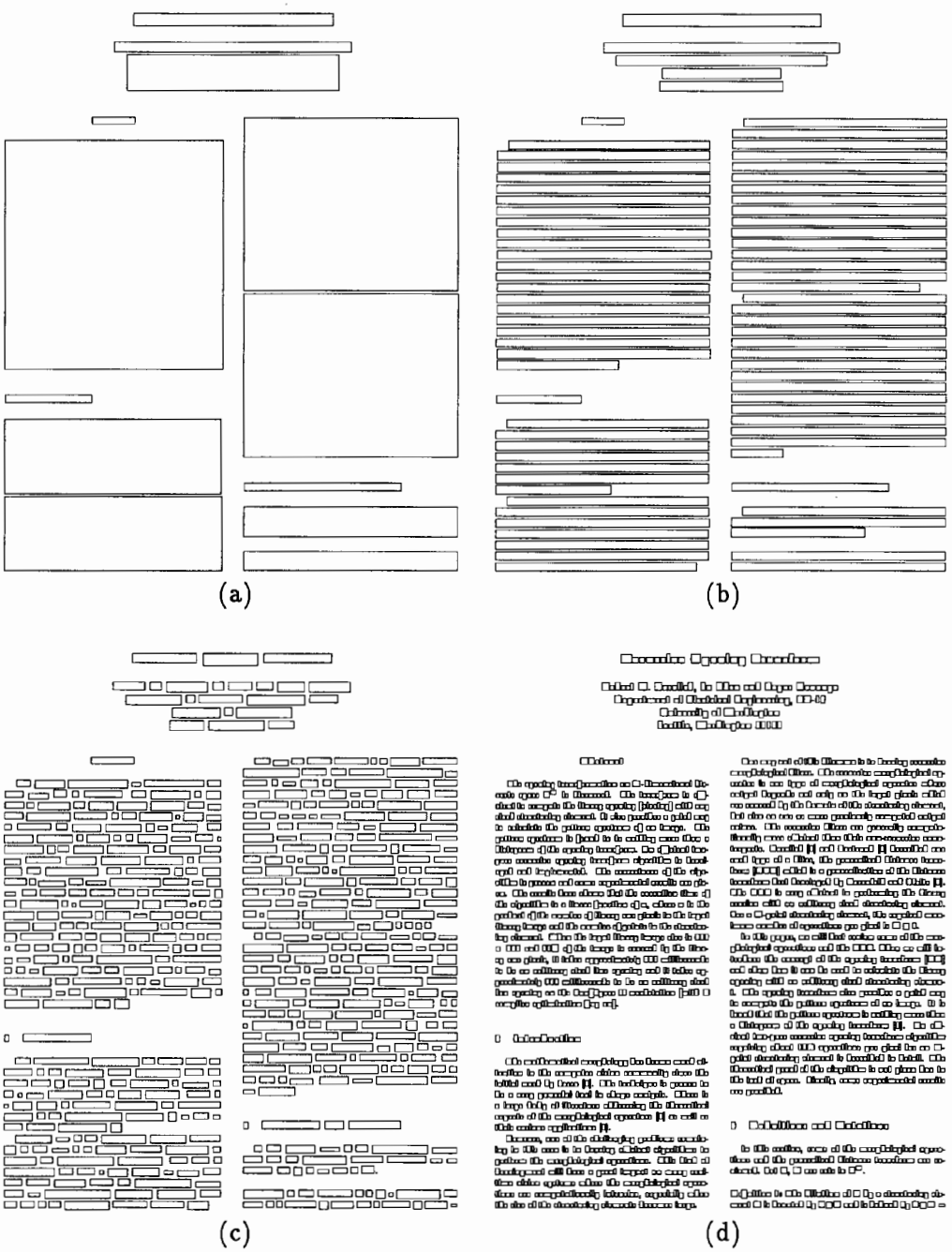


Figure 2: **Illustrates the hierarchical layout representation of the example document page. (a) Zone bounding boxes; (b) Line bounding boxes; (c) Word bounding boxes; (d) Character bounding boxes.**

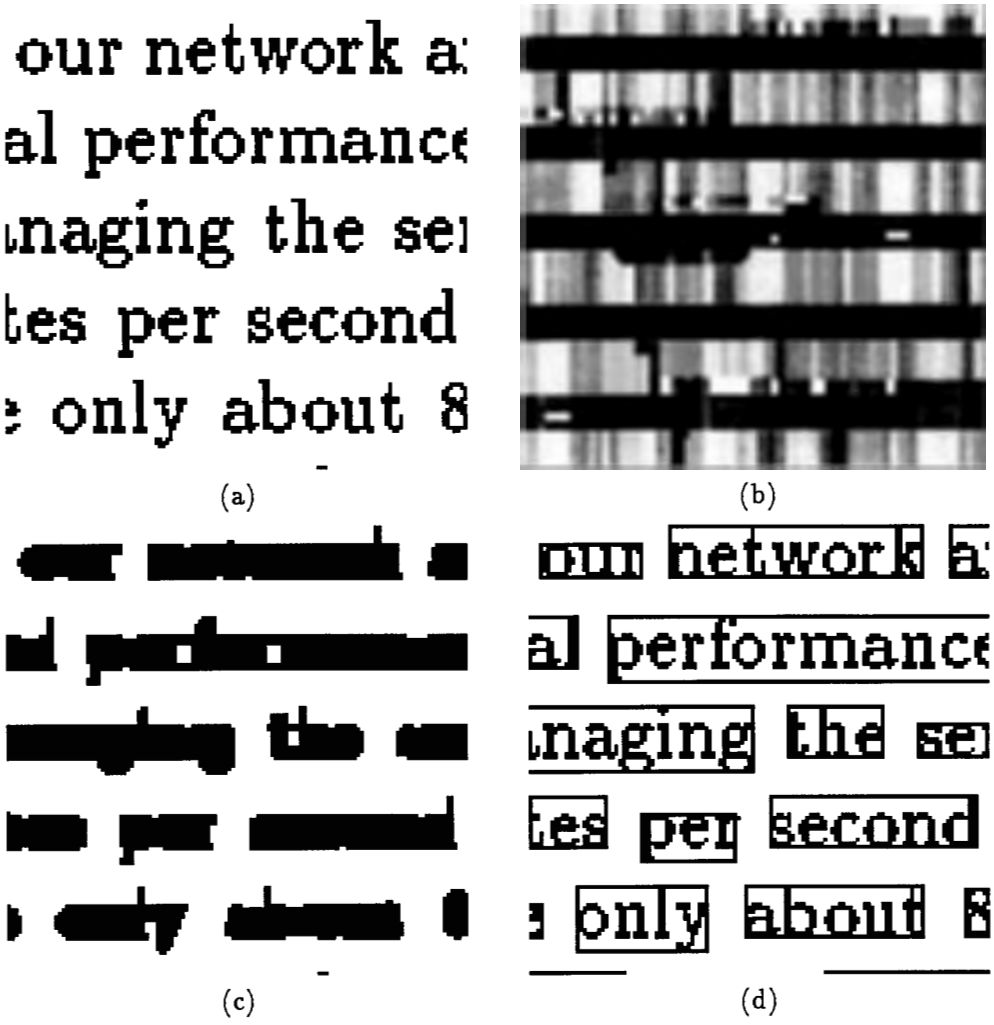
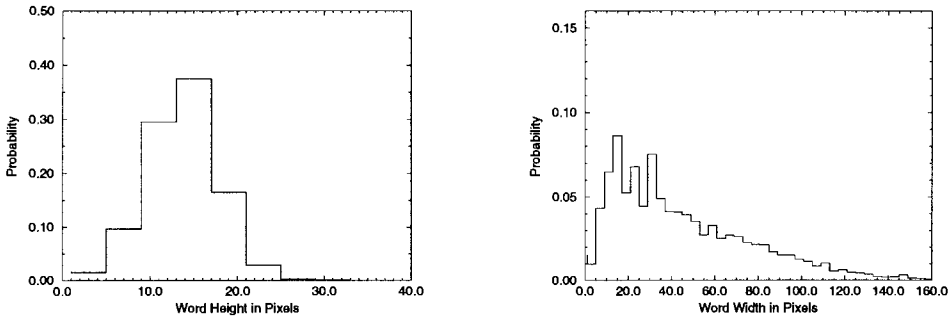
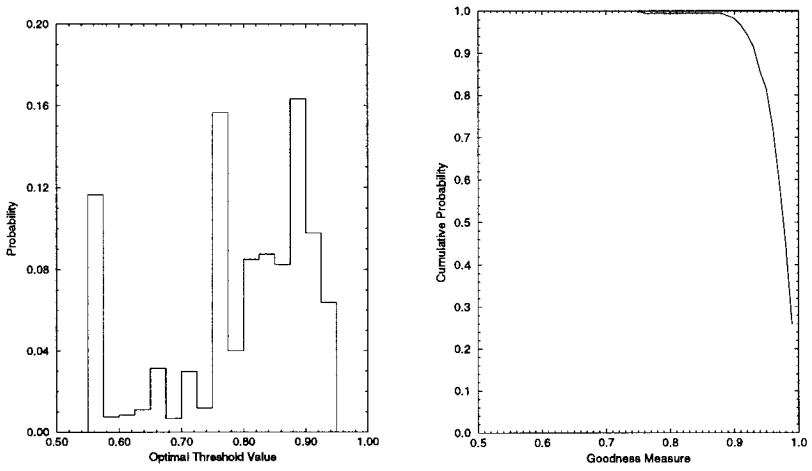


Figure 3: Illustrates the word segmentation process. (a) sub-sampled 150dpi image; (b) correlated posterior probability map image; (c) thresholded word block image; (d) word bounding boxes.



(a) (b)

Figure 4: Illustrates the word height and width probability distributions. (a) word height; (b) word width. The image resolution is 150dpi.



(a) (b)

Figure 5: Illustrates the probability distributions of the optimal threshold values T_p^{opt} and the corresponding goodness measures for the training document images.

Table 1: 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

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

Total Ground Truth Words	Correct	Splitting	Merging	Miss	Spurious
429338	408741	8440	10961	376	820
	(95.2026%)	(1.9658%)	(2.5530%)	(0.0876%)	(0.1910%)

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

Total Detected Words	Correct	Splitting	Merging	False	Spurious
434390	408741	19196	4940	763	750
	(94.0954%)	(4.4191%)	(1.1372%)	(0.1756%)	(0.1727%)

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

Total Ground Truth Words	Correct	Splitting	Merging	Miss	Spurious
258328	245584	4137	7123	846	638
	(95.0667%)	(1.6015%)	(2.7573%)	(0.3275%)	(0.2470%)

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

Total Detected Words	Correct	Splitting	Merging	False	Spurious
258802	245584	9290	2961	313	654
	(94.8926%)	(3.5896%)	(1.1441%)	(0.1209%)	(0.2527%)

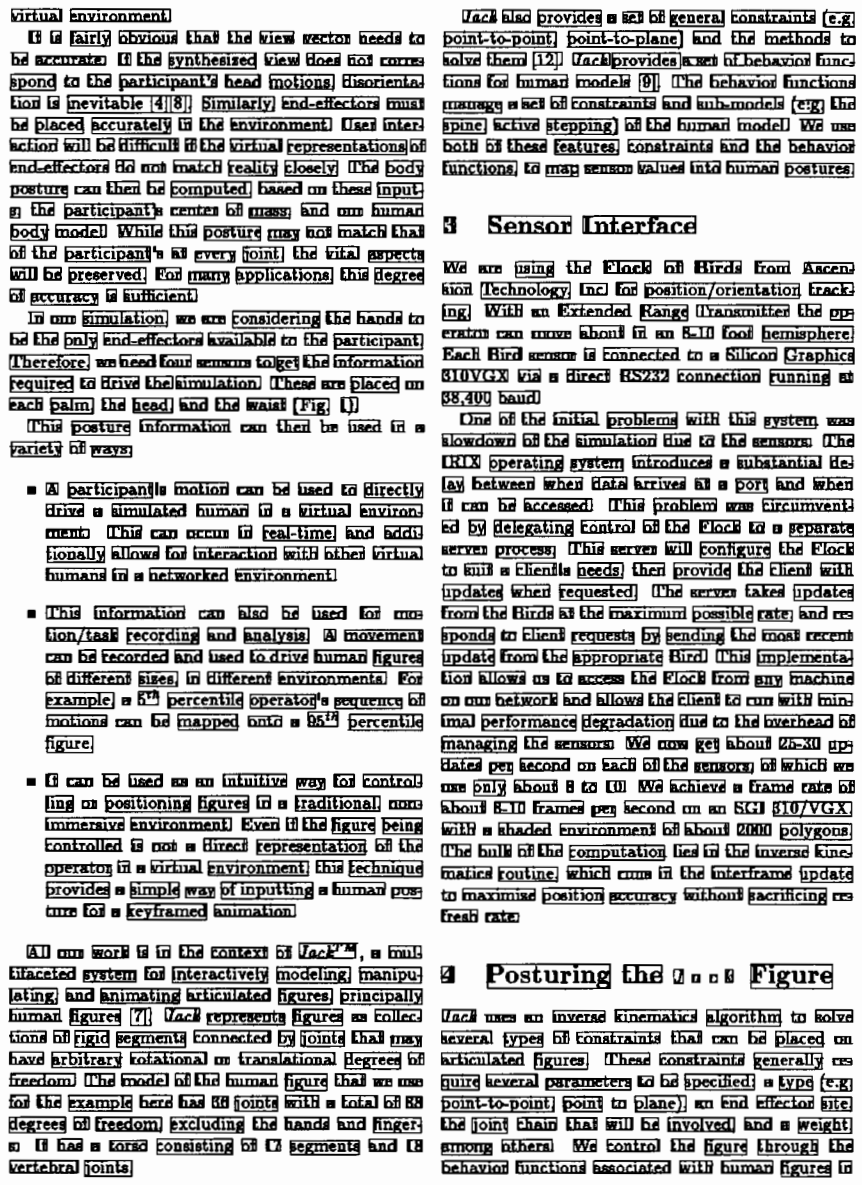
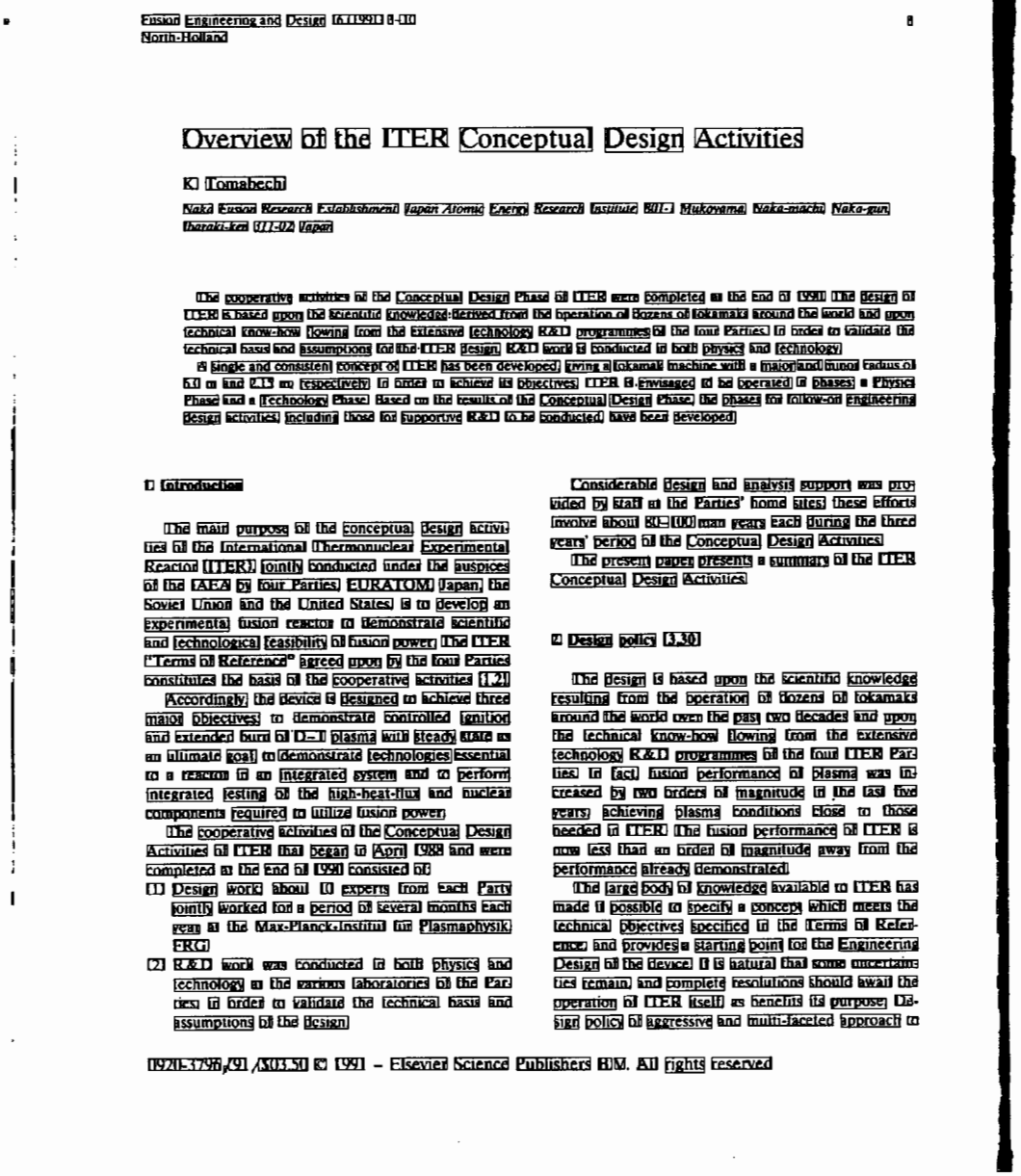


Figure 6: Illustrates a synthetic document image overlaid with the extracted word bounding boxes.



Fusion Engineering and Design 18 (1991) 3–10
North-Holland

Overview of the ITER Conceptual Design Activities

K. Tomabechi

Naka Fusion Research Establishment, Japan Atomic Energy Research Institute, 801-1, Mukoyama, Naka-machi, Naka-gun, Ibaraki-ken 311-02, Japan

The cooperative activities of the Conceptual Design Phase of ITER were completed at the end of 1990. The design of ITER is based upon the scientific knowledge derived from the operation of dozens of tokamaks around the world and upon technical know-how flowing from the extensive technology R&D programmes of the four Parties, in order to validate the technical basis and assumptions for the ITER design, R&D work is conducted in both physics and technology.

A single and consistent concept of ITER has been developed, being a tokamak machine with a major and minor radius of 6.0 m and 2.15 m, respectively, in order to achieve its objectives. ITER is envisaged to be operated in 3 phases: a Physics Phase and a Technology Phase. Based on the results of the Conceptual Design Phase, the phases for follow-on engineering design activities, including those for supportive R&D to be conducted, have been developed.

1. Introduction

The main purpose of the conceptual design activities of the International Thermonuclear Experimental Reactor (ITER), jointly conducted under the auspices of the IAEA by four Parties, EURATOM, Japan, the Soviet Union and the United States, is to develop an experimental fusion reactor to demonstrate scientific and technological feasibility of fusion power. The ITER "Terms of Reference" agreed upon by the four Parties constitutes the basis of the cooperative activities [1,2].

Accordingly, the device is designed to achieve three major objectives: to demonstrate controlled ignition and extended burn of D–T plasma with steady state as an ultimate goal; to demonstrate technologies essential to a reactor in an integrated system and to perform integrated testing of the high-heat-flux and nuclear components required to utilize fusion power.

The cooperative activities of the Conceptual Design Activities of ITER that began in April 1988 and were completed at the end of 1990 consisted of:

(1) Design work, about 10 experts from each Party jointly worked for a period of several months each year at the Max-Planck-Institut für Plasmaphysik, FRG;

(2) R&D work was conducted in both physics and technology at the various laboratories of the Parties in order to validate the technical basis and assumptions of the design.

Considerable design and analysis support was provided by staff at the Parties' home sites; these efforts involve about 80–100 man years each during the three years' period of the Conceptual Design Activities.

The present paper presents a summary of the ITER Conceptual Design Activities.

2. Design policy [3,30]

The design is based upon the scientific knowledge resulting from the operation of dozens of tokamaks around the world over the past two decades and upon the technical know-how flowing from the extensive technology R&D programmes of the four ITER Parties. In fact, fusion performance of plasma was increased by two orders of magnitude in the last five years, achieving plasma conditions close to those needed in ITER. The fusion performance of ITER is now less than an order of magnitude away from the performance already demonstrated.

The large body of knowledge available to ITER has made it possible to specify a concept which meets the technical objectives specified in the Terms of Reference and provides a starting point for the Engineering Design of the device. It is natural that some uncertainties remain and complete resolutions should await the operation of ITER itself as benefits its purpose. Design policy of aggressive and multi-faceted approach to

0920-3796/91/$03.50

Figure 7: Illustrates a real document image overlaid with the extracted word bounding boxes.