Performance evaluation of document layout analysis algorithms on the UW data set

Jisheng Liang, Ihsin T. Phillips[†], and Robert M. Haralick

Department of Electrical Engineering, University of Washington, Seattle, Washington 98195

[†]Department of Computer Science, Seattle University, Seattle, Washington 98122

ABSTRACT

A performance evaluation protocol for the layout analysis (page segmentation) is discussed in this paper. In the University of Washington English Document Image Database-III, there are 1600 English document images that come with manually edited ground truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text-lines, and words. We describe a performance metric for the comparison of the detected entities and the ground truth in terms of their bounding boxes. The Document Attribute Format Specification (DAFS) is used as the standard data representation. The protocol is intended to serve as a model for using the UW-III database to evaluate the document analysis algorithms. A set of layout analysis algorithms which detect different entities have been tested based on the data set and the performance metric. The evaluation results are presented in this paper.

Keyword: Document layout analysis, page segmentation, performance evaluation, document image database.

1 INTRODUCTION

Since document understanding techniques will be moving to the consumer market, they will have to perform nearly perfectly and efficiently. This means that they will have to be proved out on significant sized data sets and there must be suitable performance metrics for each kind of information a document understanding technique infers.[4] However, many of the earlier techniques on document image analysis were developed on a trial-and-error basis and only give illustrative results.

The document layout analysis (page segmentation) produces a set of entities within a region of interest. Entities can be homogeneous zones, text-blocks, text-lines, words, etc. None of the existing layout analysis techniques are optimal, and there is a wide agreement upon the need for an automatic tool to benchmark them. Randriamasy and Vincent[8] proposed a pixel-level and region-based approach to compare segmentation results and manually generated regions. An overlap matching technique is used to associate each region of one of the two sets, to the regions in the other set it has a non-empty intersection. Since the black pixels contained in the regions are counted rather than the regions themselves, this method is independent of representation scheme of regions. Quantitative

evaluation of segmentation is derived from the cost of incorrect splittings and mergings. Working on the bit-map level, their technique involves extensive computation. They assume there is only one text orientation for the whole page. Garris[2] proposed a scoring method which computes the coverage and efficiency of zone segmentation algorithm. The box distance and box similarity between zones are computed to find the matching pairs. This technique provides some numbers (scores) which are not able to help users analyze errors. Kanai et al.[5] proposed a text-based method to evaluate the zone segmentation performance. They compute an edit distance which is based on the number of edit operations (text insertions, deletions, and block moves) required to transform an OCR output to the correct text. The cost of segmentation itself is derived by comparing the costs corresponding to manually and automatically zoned pages. This metric can be used to test "black-box" commercial systems, but is not able to help users categorize the segmentation errors. This method only deals with text regions. They assume the OCR performance is independent of the segmentation performance.

A large quantity of ground truth data, varying in quality, is required in order to give an accurate measurement of the performance of an algorithm under different conditions. In the University of Washington English Document Image Database-III,[7] there are 1600 English document images that come with manually edited ground truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text-lines, and words. In this paper, we consider the comparison of ground truth and detected entities in terms of their bounding boxes. A quantitative metric is proposed to evaluate the performance of the layout analysis. The DAFS (Document Attribute Format Specification) is used as the interchange format for the representation of ground truth and algorithm output. We evaluate the performance of a set of layout analysis algorithms based on the UW-III data set and the performance metric.

2 Layout Analysis Problem

We define a hierarchical structure, called a *Polygonal Spatial Structure,* to capture the information about a document image.

- A polygon and the divider around it is called a Polygonal Spatial Structure (PSS).
- A basic Polygonal Spatial Structure, which is not further divided, carries a content, and the nature of the divider.
- A composite Polygonal Spatial Structure consists of one or more non-overlapping Polygonal Spatial Structures which are either basic or composite Polygonal Spatial Structures.
- We denote by Θ the set of content types (text-block, text-line, word, table, equation, drawing, halftone, handwriting, etc.).
- We denote by $\mathcal{D}$ the set of dividers (spacing, ruling, etc.).
- We denote by $\mathcal{A}$ the set of non-overlapping homogeneous polygonal areas on document image. Each polygonal area $A \in \mathcal{A}$ consists of an ordered pair (θ, I), where $\theta \in \Theta$ specifies the content type and I is the area. A polygon is homogeneous if all its area is of one type and there is a standard reading order for the content within the area.

We frame the document layout analysis problem in terms of the abstract PSS definition. A layout structure of a document image is a specification of the geometry of the polygons, the content types of the polygons, and the spatial relations of these polygons. Formally, a layout structure is $\Phi = (\mathcal{A}, \mathcal{D})$, where $\mathcal{A}$ is a set of homogeneous polygonal areas, and $\mathcal{D}$ is a set of dividers. Therefore, the evaluation problem is: given a ground truth layout $\Phi = (\mathcal{A}, \mathcal{D})$ and an automatically computed layout $\hat{\Phi} = (\hat{\mathcal{A}}, \hat{\mathcal{D}})$, determine the correspondence between the segmentation results $\hat{\mathcal{A}}$ and the ground truth $\mathcal{A}$, and report the number of correct, miss, false, merging, and splitting detections.

3 PERFORMANCE EVALUATION PROTOCOL

A performance evaluation needs a performance metric, ground-truth data, and an algorithm to match the output representation of document understanding algorithms with the ground-truth representation (See Figure 1). A standard interchange document structure representation is necessary for developers to exchange the training and test images. Statistical and display tools are also needed to help users categorize errors and analyze the cause of errors.

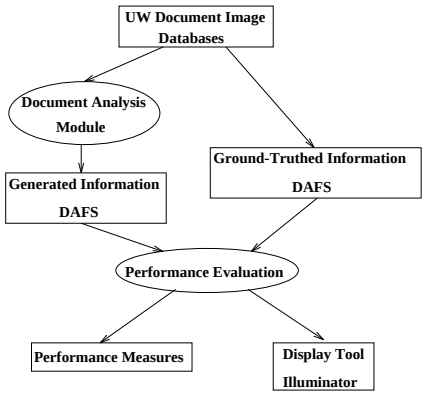

Figure 1: illustrates the process to compare the detected information with the ground truth.

3.1 Data Representation

The Document Attribute Format Specification (DAFS)[9] has been developed as a document interchange format for document decomposition and data sharing applications. The DAFS provides a format for breaking down documents into standardized entities, defining entity boundaries and attributes, and labeling their contents (text values) and attribute values. In essence, an entity is one area of image which is usually defined by a bounding box. DAFS permits the creation of parent, child and sibling relationships between entities, providing easy representation of the hierarchical structures of a document. Thus DAFS is a suitable structure for representing a Polygonal Spatial Structure. Each polygonal area of PSS is represented by an entity of DAFS. The DAFSlib provides C routines for creating, labeling, searching and manipulating all the types of entities defined under DAFS. The *Illuminator* is an editor and set of tools built on top of the *DAFSlib*. Through DAFS, developers will be able to exchange training and test documents and work together on shared problems.

3.2 Data Sets

UW-III[7] is the third in a series of UW document image databases. It contains a total of 1600 English document images randomly selected from scientific and technical journals. The documents are in DAFS format and consist of accurately ground-truthed layout and logical structure, style, and content. Each page contains a hierarchy of manually verified page, zone, text-line, and word entities with bounding box and in the correct reading order. The ASCII ground-truth and zone attributes are tagged to each zone entity. Based on these ground-truth data, we can evaluate the performance of document analysis algorithms and build statistical models to characterize various types of document structures.

We use this database as the test bed for our algorithms. DAFS structure representation is chosen as the standard representation for the ground truth data and the results of document analysis algorithms. The Illumi-

nator software[10] can be used as the display tool. For each structure that we use to describe a document, there is an associated metric that measures the difference between a structure that is automatically produced and the ground-truth structure.

3.3 Performance Metric

To evaluate the performance of the layout analysis, the detected entities must be compared with the ground truth entities. There are two problems in making the evaluation. The first is one of correspondence: which boxes of the ground truth set correspond to which boxes of the automatically produced set. Once this correspondence is determined then a comparison based on area overlap can proceed. Suppose we are given two sets $\mathcal{G} = \{G_1, G_2, \cdots, G_M\}$ for ground-truthed entity boxes and $\mathcal{D} = \{D_1, D_2, \cdots, D_N\}$ for detected entity boxes. The comparison of $\mathcal{G}$ and $\mathcal{D}$ can be made in terms of the following two kinds of measures:

$$\sigma_{ij} = \frac{\text{Area}(G_i \cap D_j)}{\text{Area}(G_i)} \; and \; \tau_{ij} = \frac{\text{Area}(G_i \cap D_j)}{\text{Area}(D_j)}$$

where $1 \leq i \leq M$, $1 \leq j \leq N$, and Area(A) represents the area of A. The measures in the above equation constitute two matrices $\Sigma = (\sigma_{ij})$ and $T = (\tau_{ij})$. Notice that σ_{ij} indicates how much portion of G_i is occupied by D_j, and τ_{ij} indicates how much portion of D_j is occupied by G_i. Our strategy of performance evaluation is to analyze these matrices to determine correspondence and overlap. The possible errors of misdetection, false alarm, splitting, merging are detected for each entity.

These matrices have the following properties:

- If an object G_i is correctly identified by the segmentation, then there exists D_j such that $\sigma_{ij} \approx 1$ and $\tau_{ij} \approx 1$. (correct detection)
- If a certain object G_i is not detected by the segmentation, then $\sigma_{ij} \approx 0$ for all $1 \leq j \leq N$. (misdetection)
- If the segmentation detects an object D_j which is not actually present in $\mathcal{G}$, then $\tau_{ij} \approx 0$ for all $1 \leq i \leq M$. (false alarm)
- If the segmentation detects an object G_i but produces two or more objects, then $\sigma_{ij} < 1$ for all j, and $\sum_{j=1}^{N} \sigma_{ij} \approx 1$. (splitting detection)
- If two or more objects in $\mathcal{G}$ are identified as an object D_j, then $\tau_{ij} < 1$ for all i, and $\sum_{i=1}^{M} \tau_{ij} \approx 1$. (merging detection)
- Any other detections are counted as spurious detections.

An overall cost or economic gain of segmentation process can be computed based on the number of detections. Let $N_{cor}, N_{mis}, N_{fal}$ denote the number of correct, miss, and false detections. Let $N^g_{spl}, N^g_{meg}, N^g_{spu}$ be the numbers of ground-truth entities which are involved in the splitting, merging, and spurious detections, respectively. Let $N^d_{spl}, N^d_{meg}, N^d_{spu}$ be the numbers of detected entities which are involved in the splitting, merging, and spurious detections, respectively. Let $N = 2 \times N_{cor} + N_{mis} + N_{fal} + N^g_{spl} + N^d_{spl} + N^g_{meg} + N^d_{meg} + N^g_{spu} + N^d_{spu}$ be the total number of entities from both groun-truth and detected layout structure. The total cost of the segmentation process is calculated by the following formula:

$$Cost \; = \; \frac{W_{mis}N_{mis} + W_{fal}N_{fal} + W_{spl}(N^g_{spl} + N^d_{spl}) + W_{meg}(N^g_{meg} + N^d_{meg}) + W_{spu}(N^g_{spu} + N^d_{spu})}{N}$$

where $W_{mis}, W_{fal}, W_{spl}, W_{meg}, W_{spu}$ are weights for the corresponding errors. This cost function reports the weighted cost of the editing operations needed to correct the detected entity boxes into ground-truth boxes.

Different weights can also be given to horizontal and vertical splitting or merging, according to the text reading direction.

Software to compare the detected boxes with the ground truth boxes in DAFS format has been developed (see Figure 2). It produces the performance measures, and generates the DAFS files which contain the common and difference between two set of bounding boxes. This method can be applied to other entity representation schemes, such as polygons, and piecewise rectangles which are supported by DAFS.

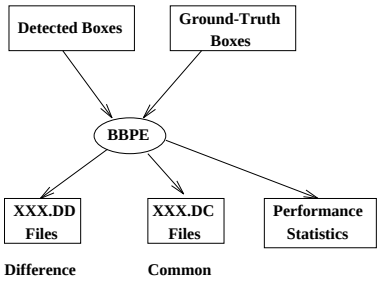

Figure 2: illustrates the process to compare the detected boxes with the ground truth boxes.

4 EXPERIMENTS

The ISL document layout analysis toolbox[6] includes a set of segmentation modules which produce the zone, text-line, word, and text-block entities respectively. We have tested all algorithms on 1600 pages in the UW-III database using the proposed performance metric. For each algorithm, the ground truth entities are provided as the input, and the default parameter values are used.

4.1 Performance of Zone Segmentation Module

A zone entity is a rectangular area that consists of homogeneous data (only one physical content type). A text zone is constrained to a single column of text and there is only one reading order for the content within the zone. Given a document image, zone segmentation is the process to segment image into a set of zone entities.

4.1.1 X-Y Cut on the projection profiles of connected component bounding boxes[3]

At each step of X–Y cut, the horizontal and vertical projection profiles of the connected component bounding boxes are calculated. Then a zone division is performed at the most prominent valley in either projection profile. The process is repeated recursively until no sufficiently wide valley is left or the height/width aspect ratio of the current region is larger than a threshold. The algorithm's most significant strength is its speed. Using connected components requires significantly less running time than previous pixel-based X-Y cutting algorithms. Table 1 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth zones as well as the algorithm output.

This algorithm is restricted to bidirectional X-Y cuttable layouts. It is also sensitive to the additive noise (false alarm and merging) and page skew (merging). To decide if applying a cut on a projection profile valley, a threshold on the width of valley is used. Therefore, the optimal value of the threshold needs to be determined for a given set of documents. When the spacing between two different regions, such as two columns, or a figure

Table 1: Performance of X-Y cut zone segmentation with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Zones	Correct	Splitting	Merging	Miss	Spurious
24243	21136	877	1446	262	522
	(87.18%)	(3.62%)	(5.96%)	(1.08%)	(2.15%)

(a)

Total Detected Zones	Correct	Splitting	Merging	False	Spurious
19477	12745	4491	523	890	828
	(66.44%)	(23.06%)	(2.69%)	(4.57%)	(4.25%)

(b)

and its caption, is smaller than the given threshold, they will be merged. The large spacing between text lines (double spacing), large sized text (title), equations, or within a single table or figure can cause splitting if a global threshold is used for all pages. Instead having a global value, the threshold should be adaptively determined by considering the width and depth of the projection profile valley, the size of nearby connected components, and the aspect ratio of current zone, etc. Smear and skew of the page can cause merging. So deskew has to be done before applying the projection. Smear usually produces some huge connected components near the boundary of the page. Some thresholds, such as the maximum size of connected components, the maximum aspect ratio of connected components, and distance of these connected components to the boundary, can be used to remove them.

4.2 Performance of Text-line Segmentation Module

Our layout analysis toolbox includes three different methods to extract text-line entities.

4.2.1 Cut the projection profile of connected component bounding boxes[3]

This algorithm computes the horizontal projection profiles of connected component bounding boxes in the given homogeneous zone (assume the horizontal reading direction), then makes a cut on wherever the profile value is zero. The maximum and minimum height of text-line are used to filter the generated text lines. Table 2 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth lines as well as the algorithm output.

Table 2: Performance of line segmentation (projection cut) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Lines	Correct	Splitting	Merging	Miss	Spurious
105439	99935	125	5035	299	45
	(94.78%)	(0.12%)	(4.78%)	(0.28%)	(0.04%)

(a)

Total Detected Lines	Correct	Splitting	Merging	False	Spurious
102680	99935	503	1755	31	456
	(97.33%)	(0.49%)	(1.71%)	(0.03%)	(0.44%)

(b)

The advantages of this algorithm are that it is very simple and fast, and it produces very low misdetection rate. But this algorithm is sensitive to skew (global or local), smear, warping, and noise. If the inter-text-line spacing is very small and the superscript, subscript, or the ascender and descender of adjacent lines overlap or

touch with each other, the text lines are usually merged. To solve this problem, we need to cut the projection profile where the non-zero deep valley appears. It means an adaptive decision for cutting the profile has to be made instead of using a global threshold. This decision depends on the width of the valley, the profile values, and the height of the valley boundaries. The morphological closing and opening operations can be used for this purpose. Skew and warping of text-lines can cause merging of text lines. The spurious errors are usually caused by the rotated text lines. The text-line reading direction detection needs to be done beforehand. We can check if there are distinct high peaks and deep valleys at somewhat regular intervals in the projection profile of one direction, while no such features are seen in the other direction.

4.2.2 Merging and splitting of connected components[6]

This algorithm merges a set of connected component bounding boxes into line bounding boxes. Boxes are merged if and only if the horizontal distance between them is small, and they overlap vertically. The set of connected component bounding boxes is filtered to eliminate noise and non-textual components before applying the segmentation algorithm. In severally degraded documents, characters in vertically adjacent text lines may be merged. A splitting process is applied to split the possible merged connected components. Then the segmentation algorithm is re-run on the set of connected components. Table 3 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth lines as well as the algorithm output.

Table 3: Performance of line segmentation (merging and splitting) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Lines	Correct	Splitting	Merging	Miss	Spurious
105439	102861 (97.56%)	554 (0.53%)	1329 (1.26%)	630 (0.60%)	65 (0.06%)

(a)

Total Detected Lines	Correct	Splitting	Merging	False	Spurious
105200	102861 (97.78%)	1260 (1.20%)	585 (0.56%)	33 (0.03%)	461 (0.44%)

(b)

A set of rules and the corresponding thresholds have been used for the merging and splitting process. The optimal values for the thresholds should be determined from the training data. This algorithm is slow. The splitting procedure is able to split the merged connected components and warped or skewed text-lines, but it may also cause some splitting errors, i.e. the descender of characters, dot of character "i" or "j", or the superscript and subscript are split from the text-line..

4.2.3 Comparing two line segmentation algorithms

We compare the performance of two line segmentation algorithms described above. The protocol diagram is shown in Figure 3. The output will be the performance statistics of the common correct detection (XXX.DC.DC files), the different correct detection (XXX.DC.DD files), the common errors (XXX.DD.DC files), and the different errors (XXX.DD.DD files).

Table 4 illustrates the numbers and percentages of common and specific detections with respect to the correct line detection made by the first and second algorithms. If we combine two line segmentation algorithms, the total number of correct detections is 103686, and the correct detection percentage is 98.51%.

Table 5 illustrates the numbers and percentages of common, specific, splitting, merging and spurious detections

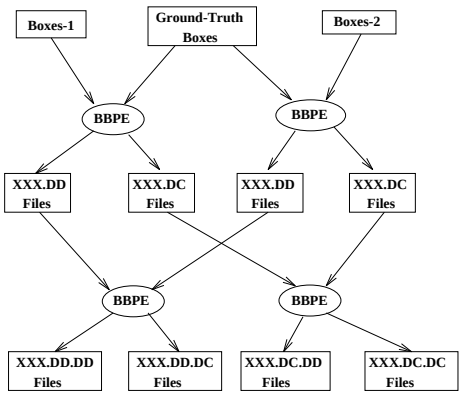

Figure 3: illustrates the process to compare the performance of two different segmentation algorithms with the ground truth boxes.

Table 4: Comparison of correct detections with respect to (a) the first algorithm (projection cut); the second algorithm (merging and splitting).

Total Correct Lines	Common	Specific	Splitting	Merging	Spurious
99935	99110 (99.17%)	825 (0.83%)	0 (0.00%)	0 (0.00%)	0 (0.00%)

(a)

Total Correct Lines	Common	Specific	Splitting	Merging	Spurious
102861	99110 (97.13%)	3751 (3.65%)	0 (0.00%)	0 (0.00%)	0 (0.00%)

(b)

with respect to the errors made by the first and second algorithms. Due to the different nature of two line segmentation algorithms, it is unlikely that they make the same mistakes. Therefore, we can either combine the outputs generated by two algorithms to get the better performance, or apply the projection profile algorithm first, since it requires less computation and has very few misdetection errors, then apply the merging and splitting algorithm to the suspicious lines.

Table 5: Comparison of errors with respect to (a) the first algorithm (projection profile); the second algorithm (merging and splitting).

Total Errors	Common	Specific	Splitting	Merging	Spurious
2184	684 (31.32%)	1223 (56.00%)	181 (8.29%)	20 (0.92%)	76 (3.48%)

(a)

Total Errors	Common	Specific	Splitting	Merging	Spurious
1687	684 (40.55%)	415 (24.60%)	494 (29.28%)	10 (0.59%)	84 (4.98%)

(b)

4.2.4 Group words into lines based on a Linear Displacement Structure Model[1]

The extraction problem is posed as finding the text lines and text blocks that maximize the Bayesian probability of the text lines and text blocks by observing the word bounding boxes. The probabilistic linear displacement model (PLDM) is derived to model the text line structures from word bounding boxes. An iterative algorithm

is developed that utilizes the probabilistic models to extract text lines and text blocks. Table 6 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth lines as well as the algorithm output.

Table 6: Performance of line segmentation (PLDM) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Lines	Correct	Splitting	Merging	Miss	Spurious
105439	101736 (96.49%)	1486 (1.41%)	2136 (2.03%)	11 (0.01%)	70 (0.07%)

(a)

Total Detected Lines	Correct	Splitting	Merging	False	Spurious
106120	101736 (95.87%)	3235 (0.30%)	988 (0.93%)	0 (0.00%)	161 (0.15%)

(b)

This algorithm works well when the models are close approximations of the actual situations. A large number of model parameters also need to be determined given the training data. The algorithm is unsuitable for documents that do not fit its model. It assumes the documents have homogeneous layout, i.e., similar font size and spacing between text entities. Therefore, the algorithm produces poor results on pages with various layout attributes.

4.3 Performance of Word Segmentation Module

Two different word segmentation (top-down and bottom-up) methods are provided in our tool box.

4.3.1 Cut on projection profiles of connected component bounding boxes[3]

Given a text line, the vertical projection profile of connected component bounding boxes is computed. The algorithm considers each such profile as a one-dimensional *gray-scale image*, and thresholds it at 1 to produce a binary image. The binarization is followed by morphological closing with a structuring element of appropriate length to close spaces between symbols, but not between words. The length is determined by analyzing the distribution of the run-lengths of 0's in the binarized profile. In general, such a run-length distribution is bi-modal. One mode corresponds to the inter-character spacings within words, and the other to the inter-word spacings. The bottom of the valley between these modes gives the desired structuring element length. Table 7 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth words as well as the algorithm output.

Table 7: Performance of word segmentation (projection cut) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Words	Correct	Splitting	Merging	Miss	Spurious
828201	800135 (96.61%)	3288 (0.40%)	16548 (2.00%)	7892 (0.95%)	338 (0.04%)

(a)

Total Detected Words	Correct	Splitting	Merging	False	Spurious
813419	800135 (98.37%)	7782 (0.96%)	5174 (0.64%)	147 (0.02%)	181 (0.02%)

(b)

This algorithm is fast and simple and able to produce reasonable high accuracy. It is robust for different

layout and conditions since the word spacing is determined adaptively. Small skew and warping are tolerable. One of the limitations of this algorithm is that the text-line is required as input. It assumes the inter-character and inter-word spacing is bi-model which is not right for some cases, such as the list item, text line with only one word, etc. This algorithm uses Otsu's threshold algorithm which requires the fractions of spaces in each model are approximately equal. It is sensitive to noise, italic, underlined text, etc. Recursive cuts may be applied to the text-lines with more than two space modes, such as the list item and section heading. The words which end with "f", with italic font, or has mathematical symbol, is very likely to be merged with the adjacent words. We can find the base line and x-height first, and only use the portion of connected component between those two lines for computing the spacing. It may get rid of the effect of "f" and other special symbols. In-line math, sub-script or super-script, punctuation, and brackets cause merging and splitting. Severally skewed, warped, or degraded text-lines are merged and split. The knowledge of typographical features, such as font size, normal or italic, proportional or fixed space, etc., will be very helpful to decide if a space is word space or character space.

4.3.2 Merging of black pixels using morphological closing transform[1]

The word block detection is based on the recursive closing transform. It first computes the posterior probability for each pixel being a word pixel and produces a posterior probability map image. This image is thresholded to output the binary word block image and the word bounding boxes are extracted. As a final step, the algorithm performs hypothesis test in the height of the detected word blocks to handle merging words among adjacent text lines. Table 8 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth words as well as the algorithm output.

Table 8: Performance of word segmentation (morphological closing) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Words	Correct	Splitting	Merging	Miss	Spurious
828201	656067 (79.22%)	19024 (2.30%)	100851 (12.18%)	47008 (5.68%)	5251 (0.63%)

(a)

Total Detected Words	Correct	Splitting	Merging	False	Spurious
786713	656067 (83.40%)	46654 (5.93%)	35699 (4.54%)	44007 (5.60%)	4286 (0.54%)

(b)

This algorithm is a one-step and simultaneous process. The input can be the whole page and it can be applied to either horizontal or vertical text. Since only local information is used, the algorithm is not sensitive to skew and subtractive noise. There are many parameters need to be determined. It is necessary to train the algorithm and get the posterior probability map from the ground-truthed real images. It assumes the homogeneous spatial relations among text entities. The performance should be much better if it is applied to the homogeneous text blocks, given the algorithm is trained on the text regions with the similar textual characteristics. It means a locally adaptive method should be developed instead of using the global threshold and probabilities.

4.4 Performance of Text-Block Segmentation Module

A text-block is a text entity that can be assigned a functional label (paragraphs, section heading, captions, etc.). Given a set of text-line entities, text-block extraction module is the process to merge text-lines into text-blocks.

4.4.1 Grouping of text-lines based on the alignment of neighboring lines[3]

The beginning of a text block, such as paragraph, math zone, section heading, etc., are usually marked either by changing the justification of the current text-line or by putting extra space between two text-lines: one from the previous paragraph and the other from the current one or by changing text height. So when a significant change in text-line heights, inter-text-line spacings, or justification occurs, we say that a new text block begins. Table 9 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth text-blocks as well as the algorithm output.

Table 9: Performance of text-block segmentation (text line alignment) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Text-Blocks	Correct	Splitting	Merging	Miss	Spurious
21738	16442 (75.64%)	1613 (7.42%)	3283 (15.10%)	2 (0.01%)	398 (1.83%)

(a)

Total Detected Text-Blocks	Correct	Splitting	Merging	False	Spurious
22958	16442 (71.62%)	4928 (21.47%)	1218 (5.31%)	0 (0.00%)	370 (1.61%)

(b)

This algorithm utilizes the paragraph formatting attributes as the cues for text block segmentation. A few rules and the corresponding thresholds have been developed. Instead of the global threshold for the inter-line spacing, a local threshold should be adaptively determined within a homogeneous region. By carefully studying different cases, the rules for determining the text line justification and the text block segmentation can be expanded to improve the performance. The information of text font style can be used. But the hand-crafted rules are ad-hoc and not robust.

4.4.2 Extract the text-block structure based on APLDM[1]

An augmented PLDM is developed to characterize the text block structures from text line bounding boxes. An iterative algorithm is developed that utilizes the probabilistic models to extract text lines and text blocks. Table 10 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth text-blocks as well as the algorithm output.

Table 10: Performance of text-block segmentation (APLDM) with respect to (a) the ground truth; (b) the algorithm output.

Total Ground Truth Text-Blocks	Correct	Splitting	Merging	Miss	Spurious
21738	15861 (72.96%)	1414 (6.50%)	3312 (15.24%)	8 (0.04%)	1143 (5.26%)

(a)

Total Detected Text-Blocks	Correct	Splitting	Merging	False	Spurious
22370	15861 (70.90%)	3933 (17.58%)	1456 (6.51%)	0 (0.00%)	1120 (5.01%)

(b)

Th performance of this algorithm depends on if the documents fit its model and on the training data by which the model parameters are determined.

5 DISCUSSIONS

A performance evaluation protocol for the layout analysis has been developed. A set of algorithms have been tested based on the UW-III data set and the performance metric. The free parameters of the algorithms can be estimated for the different documents if the ground truth is provided. Currently, we are using the default parameter values that are generated by heuristics or from a small set of training samples. We are now in the process of determining the values of all the parameters of all the algorithms to optimize the end performance. Each algorithm in the toolbox has its advantage and limitations for documents with different layout and conditions. Therefore it is necessary to characterize these algorithms by evaluating their performance on different kinds of document. For example, we want to know on what kind of documents an algorithm is successful or fails. Then we may be able to choose the right algorithms for a given set of documents and task. An interesting question here is, given a set of documents, how to classify them according to their quality and layout. This paper presents the results of layout analysis modules given the ground-truth input. The performance of different sequences of routines also needs to be evaluated.

6 REFERENCES

[1] S. Chen. *Document Layout Analysis Using Recursive Morphological Transforms.* Ph.D. thesis, Univ. of Washington, 1995.

[2] M.D. Garris. *Evaluating Spatial Correspondence of Zones in Document Recognition Systems.* Proc. Int. Conf. on Image Processing, pp 304-307, vol. 3. Washington, DC, Oct. 1995.

[3] J. Ha, R.M. Haralick, and I.T. Phillips. *Document Page Decomposition using Bounding Boxes of Connected Components of Black Pixels.* Document Recognition II, SPIE Proceedings, Vol. 2422, pp 140–151, San Jose, February 1995.

[4] R.M. Haralick. *Document Image Understanding: Geometric and Logical Layout.* Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 385-90, 21-23 June 1994.

[5] J. Kanai, S V. Rice, T.A. Nartker, and G. Nagy. *Automated Evaluation of OCR Zoning.* IEEE Trans. on PAMI, Vol 17, No. 1, pp 86-90, Jan. 1995.

[6] J. Liang, J. Ha, R. Rogers, B. Chanda, I.T. Phillips, and R.M. Haralick. *The Prototype of a Complete Document Understanding System.* IAPR Workshop on Document Analysis Systems, pp. 131-154, 1996.

[7] I.T. Phillips. *User's Reference Manual for the UW English/Technical Document Image Database III.* UW-III English/Technical Document Image Database Manual, 1996.

[8] S. Randriamasy and L. Vincent. *Benchmarking Page Segmentation Algorithms.* Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 411-416, 21-23 June 1994.

[9] RAF Technology, Inc., *DAFS: Document Attribute Format Specification.* 1995.

[10] RAF Technology, Inc., *Illuminator User's Manual.* 1995