Empirical Performance Evaluation of Page Segmentation Algorithms

Song Mao and Tapas Kanungo

Language and Media Processing Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742
Email: {maosong,kanungo}@cfar.umd.edu

ABSTRACT

Document page segmentation is a crucial preprocessing step in Optical Character Recognition (OCR) system. While numerous segmentation algorithms have been proposed, there is relatively less literature on *comparative* evaluation — empirical or theoretical — of these algorithms. We use the following five step methodology to quantitatively compare the performance of page segmentation algorithms: 1) First we create mutually exclusive training and test dataset with groundtruth, 2) we then select a meaningful and computable performance metric, 3) an optimization procedure is then used to automatically search for the optimal parameter values of the segmentation algorithms, 4) the segmentation algorithms are then evaluated on the test dataset, and finally 5) a statistical error analysis is performed to give the statistical significance of the experimental results. We apply this methodology to five segmentation algorithms, three of which are representative research algorithms and the rest two are well-known commercial products. The three research algorithms evaluated are: Nagy's X-Y cut, O'Gorman's Docstrum and Kise's Voronoi-diagram-based algorithm. The two commercial products evaluated are: Caere Corporation's segmentation algorithm and ScanSoft Corporation's segmentation algorithm. The evaluations are conducted on 978 images from the University of Washington III dataset. It is found that the performance of the Voronoi-based, Docstrum and Caere's segmentation algorithms are not significantly different from each other, but they are significantly better than ScanSoft's segmentation algorithm, which in turn is significantly better than the performance of the X-Y cut algorithm. Furthermore, we see that the commercial segmentation algorithms and research segmentation algorithms have comparable performances.

Keywords: Document page segmentation, OCR, comparative evaluation, performance metric, X-Y cut, Docstrum, Voronoi diagram, performance evaluation, statistical significance, paired model.

1. INTRODUCTION

Page Segmentation is the process of dividing a document image into homogeneous zones. The accuracy of most Optical Character Recognition (OCR) systems is very sensitive to the page segmentation accuracy. While many segmentation algorithms have been proposed in the literature, relatively few researchers have addressed the issue of quantitative evaluation of segmentation algorithms. In fact, a recent workshop[1] was devoted to address issues related to page segmentation and another was devoted to empirical performance evaluation.[2]

A brief survey of Page Layout Analysis can be found in Gorman and Kasturi.[3] Several page segmentation performance evaluation methods have been proposed in the past. Kanai *et al.*[4] proposed a metric that is a weighted sum of the number of edit operations (insertions, deletions and moves). The advantage of this method is that it requires only ASCII text groundtruth and hence does not require zone or text-line bounding-box groundtruth. The limitations of this method are that it can not specify the error location on the image, it is dependent on the OCR engine's recognition accuracy, and the metric can not be computed for the languages for which no OCR engine is available. Rice, Jenkins and Nartker[5] conducted a comparative evaluation of automatic zoning accuracy of four commercial OCR products using this performance metric. Vincent *et al.*[6–8] propose various bitmap-level region-based metrics. The advantages of this method are that it can evaluate both text regions and non-text regions, it is independent of zone representation schemes, the errors can be localized and categorized, and the performance metric can be customized by the users. A limitation of this method is that the metric is dependent on pixel noise. Liang, Phillips and Haralick[9] describe a region-area-based metric. The overlapping area of a groundtruth zone and segmentation zone is used to compute its performance metric. In computer vision literature, Hoover *et al.*[10] presented a methodology for evaluating range image segmentation algorithms. None of the evaluation methodologies mentioned above provide statistical significance associated with the results.

In this article, we propose a text-line based performance evaluation method. A performance metric is computed as segmentation accuracy of text-lines, i.e., the ratio of the number of correctly segmented groundtruth text-lines and the total number of groundtruth text-lines. The incorrectly segmented text-lines are defined as the horizontally merged, horizontally split and missed groundtruth text-lines because these error contribute the most OCR error. The overlapping area and the overlapping direction between groundtruth text-line bounding box and segmentation zone bounding-box are used to determine if the groundtruth text-line is correctly segmented or not. This method ignores the white space and noise regions outside text zone bodys since they contribute little OCR error. We also propose an performance evaluation methodology that is suitable for both black-box and white-box page segmentation algorithms. In this methodology, a parameter optimization procedure is conducted for each of three research algorithms, a paired model based statistical analysis is performed to provide the confidence interval of performance index and algorithm timing for each algorithm and finally an error analysis is conducted for better interpreting the functionality of algorithms.

This paper is organized as follows. In Section 2 we outline our performance evaluation methodology. In Section 3 we identify various types of OCR segmentation errors and define a performance metric based on these error measurements. In Section 4, the segmentation algorithms that we evaluated are described. In Section 5 we describe the experimental protocol used to conduct the training and testing experiments. In Section 6 we report experimental results. Finally, in Section 7, we give our conclusions.

2. PERFORMANCE EVALUATION METHODOLOGY

A large and representative dataset is desirable in any performance evaluation task in order to give an objective performance measurement of the algorithms. A typical page segmentation algorithm has a set of parameters that affect its performance index. This performance index is usually a user-defined performance metric that measures an aspect of the algorithm that the user is interested in. In order to evaluate a page segmentation algorithm on a specific dataset, a set of optimum parameters has to be used. The optimum parameter set is a *function* of the given dataset, the groundtruth and the performance metric. The set of optimum parameters for one dataset could be a non-optimal parameter set for another dataset. Hence, the choice of parameter is crucial in any performance evaluation task. When the size of dataset gets very large, a parameter set training on the whole dataset becomes computationally prohibitive and therefore a representative sample dataset of much smaller size should be used as a training dataset. After the training step, the page segmentation algorithms with the corresponding optimal parameters should be evaluated on a test dataset that is different from the training set. Finally, in order to interpret the significance of the experimental results, a statistical error analysis should be performed. The steps for our methodology for evaluating page segmentation algorithms are as follows:

1. Create mutually exclusive training and test datasets with groundtruth such that each set is a representative sample of the whole dataset.
2. Define meaningful and computable performance metrics.
3. Automatically find the optimal parameter setting for each selected segmentation algorithm using the training dataset.
4. Evaluate the segmentation algorithms with optimized parameters on the test dataset.
5. Perform a statistical error analysis to provide the significance of the evaluation results.

The above methodology can be applied to any segmentation algorithm that has free parameters. If the algorithm does not have free parameters, as is the case with many commercial algorithms, we do not perform the training step.

3. ERROR MEASUREMENTS AND METRICS

To quantitatively evaluate algorithms, a meaningful and computable metric is essential. While different metrics may reflect researchers' different interest in their problems, a comprehensive error measurement is also desirable. Users can choose the error measurement of interest and construct computable metrics based on these error measurements, e.g., weighted sum of the error measurements. In order to avoid the shortcomings of the evaluation methods mentioned in the introduction, we define a a set of error measurements based on text-lines and propose a performance metric based on these error measurements. The error measurements we use are as follows:

l_{hm}	number of horizontally merged text-lines	l_{vm}	number of vertically merged text-lines
l_{hs}	number of horizontally split text-lines	l_{vs}	number of vertically split text-lines
n_{fl}	number of false-alarm blocks	l_{mi}	number of miss-detected text-lines
l_{bs}	number of text-lines whose bounding box is vertically split	n_{hs}	number of horizontal splits that happen on text-lines
n_{hm}	number of horizontal merges that happen between text-lines	l_{er}	number of incorrectly segmented text-lines (error type defined by users)

These error measurements account for most text-line errors. Figure 1 gives a set of possible errors as well as an experimental example.

In the paper, we define the performance metric as Text-line Accuracy $= (l_{gt} - l_{er})/l_{gt}$, where l_{gt} is the total number of groundtruth text-lines. We define l_{er} as the number of groundtruth text-lines that are horizontally split/vertical split on the bounding boxes (HVBB), horizontally merged or miss-detected since these errors contribute the most error to OCR recognition result. In this metric, no background regions is involved except groundtruth text-lines themselves. This performance metric only applies to text regions and needs text-line groundtruth for its computation.

Once we have the performance metric value for the algorithms being tested, we want to know if one algorithm is better than the other and if the performance metric difference is statistically significant. We use the paired model approach proposed by Kanungo, Marton and Bulbul[11] in their evaluation of Arabic OCR products to obtain inference about difference in the means of page segmentation algorithm performance. By using this model, the correlation of data observed on a same document image is taken into account in computing the confidence intervals. We assume that the performance results on different pages and page segmentation algorithms are statistically independent. For the five algorithms tested, we compared the difference of performance metric means and processing time mean of each possible algorithm pair and report if the difference is statistically significant or not.

4. PAGE SEGMENTATION ALGORITHMS

Page segmentation algorithms can be categorized into three classes, top-down approach, bottom-up approach and hybrid approach. Docstrum by O'Gorman,[12] Voronoi-diagram-based algorithm by Kise[13] and run-length smearing algorithm by Wahl, Wong and Casey[14] are bottom-up algorithms while X-Y cut by Nagy[15,16] and the shape-directed-covers-based algorithm by Baird[17] are top-down algorithms. We implemented O'Gorman's Docstrum algorithm and Nagy's X-Y cut algorithm. These two and Kise's Voronoi-diagram-based algorithm are representative top-down and bottom-up approaches. Two commercial, state-of-the-art, page segmentation algorithms are also evaluated. One is Caere's segmentation algorithm and the other is ScanSoft's segmentation algorithm.

4.1. The X-Y Cut Page Segmentation Algorithm

The X-Y cut segmentation algorithm[15,16] is a tree-based top-down algorithm. The root node of the tree represents the entire document page D, an interior node represents a rectangle on the page, and all the leaf nodes together represent the final segmentation. While this algorithm is easy to implement, it can only work on document pages with Manhattan layout and rectangular zones. The algorithm works as follows:

1. Create the horizontal and vertical prefix sum tables H_X and H_Y as follows:
   $H_X[i][j] = \#\{p_k \in P | X(p_k) = j, Y(p_k) \leq i, I(p_k) = 1\}$,
   $H_Y[i][j] = \#\{p_k \in P | X(p_k) \leq j, Y(p_k) = i, I(p_k) = 1\}$,
   where $P = \{p_k\}, k = 1, 2, \ldots, n$ is the set of pixels in the image and $I(p_k)$ is the binary indicator function.

2. Compute X and Y black pixel projection profile histograms at each node as follows:
   $HIS_X[i] \leftarrow H_X[Y_2(Z)][i] - H_X[Y_1(Z)][i]$,
   $HIS_Y[j] \leftarrow H_Y[j][X_2(Z)] - H_Y[j][X_1(Z)]$,
   where Z is the zone corresponding to the current node, $(X_1(Z), Y_1(Z))$ and $(X_2(Z), Y_2(Z))$ are upper-left and lower-right points of the zone.

3. Obtain the widest zero valleys V_X and V_Y in the X and Y projection profile histograms HIS_X and HIS_Y. If $V_X > T_X$ or $V_Y > T_Y$, where T_X and T_Y are two width thresholds, split at the mid-point of the wider of V_X and V_Y. Otherwise, stop splitting.

4. When a split decision is made, generate two child nodes. Shrink each child zone bounding box until it "tightly" encloses the the zone body. Noise removal thresholds T_X^n and T_Y^n are then used to classify and remove background noise pixels. Since noise pixels in the background are assumed to be distributed uniformly, the noise removal thresholds T_X^n and T_Y^n for a particular node are scaled linearly based on current zone's width and height.

5. Visit each newly generated zone until none of leaf node can be split further.

Pseudo-code for our implementation of the X-Y cut algorithm can be found in our forthcoming technical report.[18]

4.2. The Docstrum Page Segmentation Algorithm

Docstrum[12] is a bottom-up page segmentation algorithm that can work on document pages with non-Manhattan layout and arbitrary skew angles. However, this algorithm only applies to the segmentation of text regions. Moreover, it does not perform well when the document page contains too many joined characters and the estimates of inter-character spacing, inter-line spacing and orientation angle become inaccurate when documents contain sparse characters.

The basic steps of the Docstrum segmentation algorithm are shown below:

1. Obtain connected components C_is using a space-efficient two-pass algorithm.[19]

2. Remove small and big noise or non-text connected components using low and high size thresholds l and h.

3. Separate C_is into two group, one with dominant characters and another with characters in titles and section headings. A parameter f_d controls the clustering.

4. Find the K nearest neighbors, $\mathrm{NN}_K(i)$, of each C_i.

5. Compute the distance and angle of each C_i and its K nearest neighbors: (ρ_j^i, θ_j^i), such that $j \in \mathrm{NN}_K(i)$.

6. Compute within-line nearest-neighbor distance histogram from the following set W_ρ :
   $W_\rho = \{\rho_j^i | j \in \mathrm{NN}_K(i),\ \text{and}\ -\theta_h \le \theta_j^i \le \theta_h\}$, where θ_h is the horizontal angle tolerance threshold.
   Estimate the within-line inter-character spacing cs as the location of the peak in the histogram.

7. Compute between-line nearest-neighbor distance histogram from the set B_ρ :
   $B_\rho = \{\rho_j^i | j \in \mathrm{NN}_K(i),\ \text{and}\ 90^\circ - \theta_v \le \theta_j^i \le 90^\circ + \theta_v\}$, where θ_v is the vertical angle tolerance threshold.
   Estimate the inter-line spacing ls as the location of the peak in the histogram.

8. Perform transitive closure on within-line nearest neighbor pairings to obtain text-lines L_is using within-line nearest neighbor distance threshold $T_{cs} = f_t * cs$.

9. Perform transitive closure on L_is to obtain structural blocks or zones Z_is using parallel distance threshold $T_{pa} = f_{pa} * cs$ and perpendicular distance threshold $T_{pe} = f_{pe} * ls$. The parallel and perpendicular distances are computed as "end–end" distance, not "centroid–centroid" distance.

Pseudo-code for our implementation of the X-Y cut algorithm can be found in our forthcoming technical report.[18] In our implementation, we did not estimate orientation since all pages in the dataset have been deskewed. Furthermore, we used a resolution of 1 pixel/bin for constructing the within-line and between-line histograms, and did not perform any smoothing of these histograms.

4.3. The Voronoi-Diagram-Based Page Segmentation Algorithm

Kise's segmentation algorithm[13] is also a bottom-up algorithm based on Voronoi diagram. This method can work on document pages that have non-Manhattan layout, arbitrary skew angles as well as non-linear text-lines. A set of connected line segments are used to bound text zones. Since we evaluate all algorithms on document pages with Manhattan layouts, this algorithm has been modified to generate rectangular zones. This algorithm tends to fragment non-text regions (figures, tables and halftone images) and text zones with irregular font size and spacings. It assumes that text regions are dominant on a page and the inter-character and inter-line spacing within a text region is uniform. The algorithm steps are as follows:

1. Label connected components. A fast labeling procedure based on border following is used. The 8-connected components and sample points on their border are simultaneously obtained from the input image. The algorithm parameter sr controls the number of sample points used.
2. Remove noise connected components using maximum noise zone size threshold nm, maximum width threshold C_w, maximum height threshold C_h, and maximum aspect ratio threshold C_r for all connected components.
3. Generate Voronoi diagram. Voronoi diagram for each connected component is generated using the sample points on its border.
4. Delete superfluous Voronoi edges. Superfluous Voronoi edges are deleted to obtain text zone boundaries according to the following criteria. Let E be the Voronoi edge between two connected components C_i and C_j and $d(E)$ be the minimum distance between any two sample points from C_i and C_j. Let T_{d1} be the estimate of inter-character spacing, T_{d2} be the estimate of the inter-line spacing plus a margin controlled by a factor fr. Define $a_r(E)$ as the area ratio $\max\{\text{Area}(C_i), \text{Area}(C_j)\}/\min\{\text{Area}(C_i), \text{Area}(C_j)\}$ and T_a as the threshold of the largest area ratio between the characters of the same font and size. If a Voronoi edge satisfies $d(E)/T_{d1} < 1$ or $d(E)/T_{d2} + a_r(E)/T_a < 1$, it is deleted. In this criteria, the first inequality indicates that the Voronoi edges between C_i and C_j with a spacing smaller than the estimated inter-character spacing are deleted, regardless of their area ratio. The second inequality implies that we do not delete the Voronoi edge if 1)if C_i and C_j come from different text zones that have larger spacing than inter-line spacing plus a margin, or 2) if one is a character and the other is a non-text object that has very different area from the character.
5. Remove noise zones using minimum area threshold A_z for all zones, and using minimum area threshold A_l, and maximum aspect ratio threshold B_r for the zones that are vertical and elongated.

4.4. Commercial Segmentation Algorithms

Two commercial products, Caere's segmentation algorithm[20] and ScanSoft's segmentation algorithm,[21,22] are selected for evaluation. They are representative state-of-art commercial products. Both are black-box algorithms with no free parameters.

5. EXPERIMENTAL PROTOCOL

In this section we provide the details of our experimental setup — datasets used, algorithm parameter values, hardware and software environments etc. The experiment we conduct has a training phase and a testing phase for the three research algorithms, and only a testing phase for the two commercial products.

The machines used for training and testing are Sun workstations running Solaris 2.6 operating system. We used text-line accuracy as our performance metric, for each document page, we obtained a performance metric value, we then compute a average performance metric value over all document pages in the training dataset or test dataset and report it as the final algorithm performance index.

5.1. Dataset Specification

We select University of Washington Dataset[23] for the performance evaluation task. All pages in the dataset are journal pages over a large variety of journals from diverse subject areas and publishers. The dataset also has geometric text-line and zone groundtruth for each page. The text-line and zone groundtruth are represented by non-overlapping rectangles. University of Washington III dataset has 1601 deskewed binary document images at 300 dpi resolution. We choose a subset of 978 pages that corresponds to the University of Washington I dataset pages as our experimental dataset. We evaluate the chosen algorithms only on text regions. The non-text regions are ignored in evaluation process. A training dataset of 100 document pages are randomly sampled from the selected 978 documents; the remaining 878 document pages are considered as the test dataset.

5.2. Parameter Specification

We specify a parameter set for each of the three research algorithms, fix the ones that the corresponding algorithm is insensitive to and only train the ones that the corresponding algorithm is sensitive to. The training procedure is conducted on the 100-page training dataset. The machines we use are Ultra 1,2 and 5 Sun workstations running Solaris 2.6 operating system. After the training step, a set of optimal parameter values are found for each research algorithm. Since the two commercial products are black-box algorithms without any parameters, we do not perform training step for them.

5.2.1. X-Y Cut Algorithm Parameters

The X-Y cut algorithm has four free parameters. Since the algorithm is very sensitive to all the four parameters, we search for the optimal value for each of the four parameters over the ranges given below,

1. X widest zero valley width threshold T_X^C: {0-400 pixels};
2. Y widest zero valley width threshold T_Y^C: {0-400 pixels};
3. Vertical noise removal threshold T_X^n: {0-400 pixels};
4. Horizontal noise removal threshold T_Y^n: {0-400 pixels}.

5.2.2. Docstrum Algorithm Parameters

O'Gorman in his paper specifies eight parameters for the Docstrum algorithm. We add two more: 1) superscript-subscript character distance threshold factor for correctly handing text-line segmentation, and 2) character size ratio threshold to separate larger characters from dominant characters. The algorithm is insensitive to six of the parameters. We fix these six parameters as the follows: number of nearest connected components for clustering $K = 9$; low connected component size-threshold $l = 2$ pixels; high connected component size-threshold $h = 200$ pixels; horizontal angle tolerance threshold $\theta_h = 30°$; vertical angle tolerance threshold $\theta_v = 30°$; superscript and subscript character distance threshold factor $f_s = 0.4$. The values for the parameters that the algorithm is sensitive to were searched from the ranges given below:

1. Nearest neighbor threshold factor f_t: {0-20};
2. Parallel distance threshold factor f_{pa}: {0-20};
3. Perpendicular distance threshold factor f_{pe}: {0-20}
4. Character size ration factor f_d: {0-20}.

5.2.3. Voronoi-Diagram-Based Algorithm Parameters

Kise's algorithm has eleven free parameters and is insensitive to seven of them, six of which are related to removing noise connected component and blocks. We fix the seven parameters as follows: maximum height and width thresholds of a connected components $C_h = 500$ pixels and $C_w = 500$ pixels, maximum connected component aspect ratio threshold $C_r = 5$, minimum area threshold of a zone $A_z = 50$ pixels^2 for all zones, and minimum area threshold $A_l = 40000$ pixels and maximum aspect ratio threshold $B_r = 4$ for the zones that are vertical and elongated. The last parameter is the size of the smoothing window $sw = 2$. The optimal values for the other four parameters are searched from the following ranges recommended by Kise:

1. sampling rate sr: {4-7};
2. Max size threshold of noise connected component nm: {10-40};
3. Margin control factor for Td2 fr: {0.01-0.5};
4. Area ratio threshold ta: {40-200}.

5.3. Algorithm Training

The parameters that segmentation algorithm are sensitive to are automatically selected by training the algorithm on the 100-page training dataset. A downhill simplex optimization procedure[24] is used to search for the optimal parameter value for each segmentation algorithm. A starting point in the parameter space is necessary for the optimization procedure. We choose five different starting points within the reasonable working parameter subspace for each research algorithm and obtained five locally optimal results. Then we select the parameter values corresponding to the maximum result as the optimal parameter value. The optimization procedure details are described in our forthcoming technical report.[18]

5.4. Algorithm Testing

All five algorithms are tested on 878 pages test dataset. In order to be able to compare the timing information of each algorithm, we test them using their optimized parameters only on a Ultra 5 Sun workstation running Solaris 2.6 operating system. The CPU speed reported by the "fpversion" UNIX command is 333 MHz. The two commercial products are run on Gateway PC with a 400 MHz Pentium II CPU running Windows 95 operating system. We normalize the PC timing to UNIX timing using the relation $t_{UNIX} = 400 \cdot t_{PC}/333$ for comparison with the timing of the research algorithms.

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1. Training Results

Three research algorithms are trained on 100-page training dataset. Table 1 reports the optimum parameters, optimum performance index (text-line accuracy) value and training time for each research algorithm. The findings from training results for each research algorithm are summarized as the following:

1. Kise's Area-Voronoi-diagram-based algorithm: $sr = 6$ implies that for every six pixels on a connected component contour we should take one sample. Intuitively the smaller this parameter is, the more sample points we get and hence the better estimation can be obtained, this result indicts this parameter has interaction with other parameters, $nm = 16$ implies that we should not delete any connected component with a size (perimeter) greater than 16 pixels, otherwise we may delete small characters as noise blocks, $fr = 0.08$ implies that we should choose a very conservative interline spacing threshold, this indicates there is a large variation of interline spacing among test dataset pages, ta=116 implies that the non-text connected components are well separated from text components in training dataset pages, we can choose a larger value safely, an infinite value for this parameter indicates that we can select voronoi edges based only on connected component distance. This algorithm gives the best performance index for the training dataset.

2. X-Y cut algorithm: $T_X^n = 61$ and $T_Y^n = 8$ indicate that vertical cut is generally longer than horizontal cuts, hence a smaller noise removal threshold is needed in horizontal direction. $T_X^C = 34$ and $T_Y^C = 42$ imply that the vertical gap and horizontal gap between text zones are generally similar, the vertical gap is slightly wider than the horizontal gap. This algorithm gives the poorest performance index. By eyeballing a large amount of training results, we found that this algorithm is sensitive to the presence of non-text region such as tables, figures and halftones, also the thresholds becomes invalid for pages with thick noise bars on the edges.

3. Docstrum algorithm: $f_t = 2.545$ indicates that the nearest neighbor distance threshold T_{cs} should be 2.545 times of estimated intercharacter spacing cs, $f_{pa} = 2.336$ indicates that parallel distance threshold T_{pa} between ends of text-lines for blocking should be 2.336 times of estimated inter-character spacing cs, $f_{pe} = 0.613$ indicates that the perpendicular distance threshold T_{pe} between ends of text-lines should be 0.613 time of estimated inter-line spacing ls, and $f_d = 8.050$ implies that the size ratio between the dominant character group and larger character group should be 8.05. These values indicate that there is a fair amount of variation in character size, text-line spacings. Also since the intercharacter and interline spacing estimation becomes very inaccurate for sparse large character group, the size ratio threshold is large. This algorithm gives a similar performance index as that of Kise's algorithm.

Table 1. Algorithm training results. For each algorithm we report the optimal parameters and the corresponding optimal performance index value. The experiments are computationally quite intensive and training each algorithm took 2-4 days.

	Optimum Parameters	Performance Index (percent)	Training Time (hours)
Voronoi	sr=6, nm=16, fr=0.08, ta=116	95.21	78.41
Docstrum	f_t= 2.55, f_{pa}=2.34, f_{pe}=0.61, f_d=8.05	95.06	54.03
X-Y cut	$T_X^C = 34$, $T_Y^C = 42$, $T_X^n = 61$, $T_Y^n = 8$ (in pixels)	88.08	95.50

Table 2. Algorithm testing results and the corresponding 95% confidence intervals. The average time per page is also reported. The time taken by the two commercial products were normalized for the processor speed differences between the PC and the SUN.

	Performance Index (percent)	Average Processing Time (seconds)
Voronoi	94.38 ± 0.78	4.71 ± 0.09
Docstrum	93.88 ± 1.03	9.60 ± 0.22
X-Y cut	84.55 ± 1.70	3.64 ± 0.05
Caere	93.97 ± 0.85	2.02 ± 0.01, (normalized) 2.42 ± 0.02
ScanSoft	87.29 ± 1.35	3.13 ± 0.04, (normalized) 3.76 ± 0.05

6.2. Testing Results

All five algorithms are tested on 878-page test dataset with their corresponding optimum parameters. Table 2 reports the testing performance index (text-line accuracy) and average algorithm timing. Figure 2 gives a bar-chart representation of the testing results for each evaluated algorithm.

From the testing results, we see that Voroni-based algorithm, Docstrum and the Caere's algorithm have similar performance index and are better than that of ScanSoft's algorithm, which in turn is better than that of the X-Y cut algorithm. Caere's segmentation algorithm has the least average processing time, Kise's algorithm, X-Y cut and ScanSoft's segmentation algorithm have the similar average processing time. Docstrum has the most average processing time. The connected component labeling we used for Docstrum may not be the optimum one and hence its timing may be further improved. For comparison purposes, an evaluator always likes to know if the performance index and processing time differences between algorithms is statistically significant or not, especially for those algorithms with similar performance index values. This is addressed in the following subsection.

6.3. Statistical Analysis of Results

We employed a paired model[11] to compare the performance index and testing time difference between each possible algorithm pair, then compute their confidence intervals. The analysis results for performance index and processing timing are reported in a matrix in Table 3 and Table 4 respectively. If we denote T_{ij} as the value of table cell at ith row and jth column, $T_{ij} = a_i - a_j$ where a_i is the performance index (algorithm timing) value of algorithm on the ith row, a_j is the performance index (algorithm timing) value of algorithm on the jth column. Note that the normalized processing timing is used for two commercial products.

From Table 3, we can find that the performance indexes of Kise's algorithm, Caere's segmentation algorithm and Docstrum are not statistically different, but they are statistically better than those of ScanSoft's segmentation algorithm and X-Y cut algorithms, the performance index of ScanSoft's segmentation algorithm is statistically better than that of X-Y cut. From Table 4, we can find that all algorithms have statistically significant processing time from each other. From the least processing time to the most processing time, the algorithms are ranked as Caere's segmentation algorithm, X-Y cut, ScanSoft's segmentation algorithm, Kise and Docstrum. For Docstrum, a better connected component labeling algorithm can improve it timing performance.

6.4. Error Analysis

Error analysis is crucial to interpret the functionality of the evaluated algorithms. Different algorithm has different weakness. Figure 3 shows error analysis results for each algorithm. X-Y cut has the most horizontally merged

Table 3. Paired model statistical analysis results on the *difference* between a pair of performance indexes (in percent) and the corresponding 95% confidence intervals. A (*) indicates that the difference is statistically significant at $\alpha = 0.05$, and no (*) indicates that the difference is not significant. We see that there is no significant difference between Voronoi, Docstrum and Caere algorithms. However this group is significantly better than Scansoft, which is inturn is better than XY-cut.

	Caere	Docstrum	ScanSoft	X-Y cut
Voronoi	0.41 ± 0.86 $P_{val} = 0.170082$	0.49 ± 0.97 $P_{val} = 0.158157$	7.09 ± 1.36 (*) $P_{val} = 1.51512E-23$	9.82 ± 1.76 (*) $P_{val} = 2.02491E-26$
Caere	–	0.08 ± 1.11 $P_{val} = 0.443731$	6.67 ± 1.38 (*) $P_{val} = 1.26375E-20$	9.40 ± 1.78 (*) $P_{val} = 4.14161E-24$
Docstrum	–	–	6.59 ± 1.61 (*) $P_{val} = 1.69193E-15$	9.32 ± 1.89 (*) $P_{val} = 2.12715E-21$
ScanSoft	–	–	–	2.72 ± 2.01 (*) $P_{val} = 0.0040104$

Table 4. Paired model statistical analysis results on the *difference* in processing times (seconds) and the corresponding 95% confidence intervals. A (*) indicates the difference is statistically significant at $\alpha = 0.05$ and no (*) implies the difference is not significant. We see that from the least to the most averge processing time, the algorithms are ranked as: Caere, X-Y cut, ScanSoft, Voronoi and Docstrum.

	Caere	Docstrum	ScanSoft	X-Y cut
Voronoi	2.29 ± 0.08 (*) $P_{val} = 0$	-4.89 ± 0.15 (*) $P_{val} = 0$	0.95 ± 0.10 (*) $P_{val} = 0$	1.07 ± 0.07 (*) $P_{val} = 0$
Caere	–	-7.18 ± 0.21 (*) $P_{val} = 0$	-1.34 ± 0.05 (*) $P_{val} = 0$	-1.22 ± 0.04 (*) $P_{val} = 0$
Docstrum	–	–	5.84 ± 0.23 (*) $P_{val} = 0$	5.95 ± 0.19 (*) $P_{val} = 0$
ScanSoft	–	–	–	0.12 ± 0.07 (*) $P_{val} = 0.00038$

text-line error rate. This occurs in pages that have thick and long noise blocks at the edges, which can not be cut through. Many text regions under these noise blocks are merged together. The Voronoi-based algorithm has the least such errors. This is partially due to the fact that the algorithm uses not only the connected component spacing but also their area ratio to generate zone boundary, and hence few lines or noise blocks between text regions do not effect the results. Since Docstrum only uses spacing information to block text regions, it has more horizontally merged text-lines than that of the Voronoi-based algorithm. In the test dataset pages, there are titles with wide inter-character and inter-word spacings, numbered text lists as well as text-lines with irregular character spacings. These features make the spacing parameter estimation inaccurate in the Voronoi-based and Docstrum algorithms. Both algorithms have about 3-4.5% HVBB split text-line error. The X-Y cut algorithm tends to cut text-lines such as header, footer and author that are not aligned with text blocks. All three research algorithms have a small number of missed text-lines. ScanSoft has the most HVBB split and miss-detected text-line error rates and the second most horizontally merged text-line error rate. The Caere algorithm has the least HVBB split text-line error rate, the third most horizontally merged text-line error rate and the second least miss-detected text-line error rate.

7. CONCLUSIONS

We proposed a performance evaluation methodology for evaluating page segmentation algorithms. Under this methodology, we introduced a text-line based performance metric that has the following features: 1) independent of shape of zones, 2) independent of OCR recognition error, 3) ignores the background information (white space, salt and pepper noise etc.), 4) segmentation errors can be localized, and 5) quantitative evaluations on lower level (e.g. text-line, word and character) segmentation algorithms can be readily achieved with little modifications. We trained the three research algorithms on the training dataset and tested the trained research algorithms and the two commercial algorithms on the test dataset. In training phase, a set of optimal parameter values is automatically found for each research algorithm using the downhill simplex algorithm. A paired model statistical analysis is performed on the experimental results to provide significance levels for both performance index and algorithm timing for each algorithm. The three research algorithms are analyzed in three error types. We found that the performance of Voronoi, Docstrum and Caere's segmentation algorithm are not significantly different from each other, but they are significantly better than that of ScanSoft's segmentation algorithm which in turn is significantly better than that of X-Y cut. We intend to extend this work to evaluation of tables, graphs and half-tone images.

ACKNOWLEDGMENTS

We would like to thank Steve Dennis and Glenn Van Doren of Department of Defense and Melissa Holland and Jeff DeHart of Army Research Laboratory for funding this work; Koichi Kise of Osaka Prefecture University for providing us the software implementation of his algorithm and modifying his algorithm for our evaluation purpose; Mindy Bokser of Caere Corporation for providing us with the Caere page segmentation software; Henry Baird of Xerox Corporation for his comments and Greg Marton of University of Maryland for his help in obtaining experiment data.

REFERENCES

1. T. Breuel and M. Worring, eds., *Document Layout Interpretation and its Applications*, (Bangalore, India), September 1999.
2. K. W. Bowyer and P. J. Phillips, eds., *Empirical Evaluation Techniques in Computer Vision*, (Santa Barbara, CA), June 1998.
3. L. O'Gorman and R. Kasturi, *Document Image Analysis*, IEEE Computer Society Press, Los Alamitos, CA, 1995.
4. J. Kanai, S. V. Rice, T. A. Nartker, and G. Nagy, "Automated evaluation of OCR zoning," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**, pp. 86–90, 1995.
5. S. V. Rice, F. R. Jenkins, and T. A. Nartker, "The fifth annual test of OCR accuracy," Tech. Rep. TR-96-01, University of Nevada, Las Vegas, NV, 1996.
6. S. Randriamasy and L. Vincent, "Benchmarking page segmentation algorithms," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 411–416, (Seattle, WA), June 1994.
7. S. Randriamasy, L. Vincent, and B. Wittner, "An automatic benchmarking scheme for page segmentation," in *Proceedings of SPIE Conference on Document Recognition*, vol. 2181, pp. 217–230, (San Jose, CA), February 1994.
8. B. A. Yanikoglu and L. Vincent, "A complete environment for ground-truthing and benchmarking document page segmentation," *Pattern Recognition* **31**, pp. 1191–204, September 1998.
9. J. Liang, I. T. Phillips, and R. M. Haralick, "Performance evaluation of document layout analysis algorithms on the UW data set," in *Proceedings of SPIE Conference on Document Recognition*, vol. 3027, pp. 149–160, (San Jose, CA), February 1997.
10. A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldof, K. W. Bowyer, D. W. Eggert, A. Fizgibbon, and R. B. Fisher, "An experimental comparison of range image segmentation algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18**, pp. 673–689, 1996.
11. T. Kanungo, G. A. Marton, and O. Bulbul, "OmniPage vs. Sakhr: Paired model evaluation of two Arabic OCR products," in *Proceedings of SPIE Conference on Document Recognition*, vol. 3651, pp. 109–120, (San Jose, CA), January 1999.
12. L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**, pp. 1162–1173, 1993.
13. K. Kise, A. Sato, and M. Iwata, "Segmentation of page images using the area Voronoi diagram," *Computer Vision and Image Understanding* **70**, pp. 370–382, 1998.
14. F. Wahl, K. Wong, and R. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Vision, Graphics, and Image Processing* **20**, pp. 375–390, 1982.
15. G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," in *Proceedings of International Conference on Pattern Recognition*, vol. 1, pp. 347–349, (Montreal, Canada), July 1984.
16. G. Nagy, S. Seth, and M. Viswanathan, "A prototype document image analysis system for technical journals," *Computer* **25**, pp. 10–22, 1992.
17. H. Baird, "Background structure in document images," *International Journal of Pattern Recognition and Artificial Intelligence* **8**, pp. 1013–1030, 1994.
18. S. Mao and T. Kanungo, "Empirical evaluation of document segmentation algorithms." Forthcoming technical report, 1999.
19. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley Publishing Company, Reading, MA, 1992.
20. Caere Co., *Caere Developer's Kit 2000*.
21. ScanSoft Co., *TextBridge: Application Programmer's Interface*.
22. ScanSoft Co., *XDOC Data Format*.
23. I. Guyon, R. M. Haralick, J. J. Hull, and I. T. Phillips, "Data sets for OCR and document image understanding research," in *Handbook of Character Recognition and Document Image Analysis*, H. Bunke and P. Wang, eds., pp. 779–799, World Scientific Publishing Co. Pte. Ltd., Singapore, 1997.
24. J. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal* **7**, pp. 308–313, 1965.

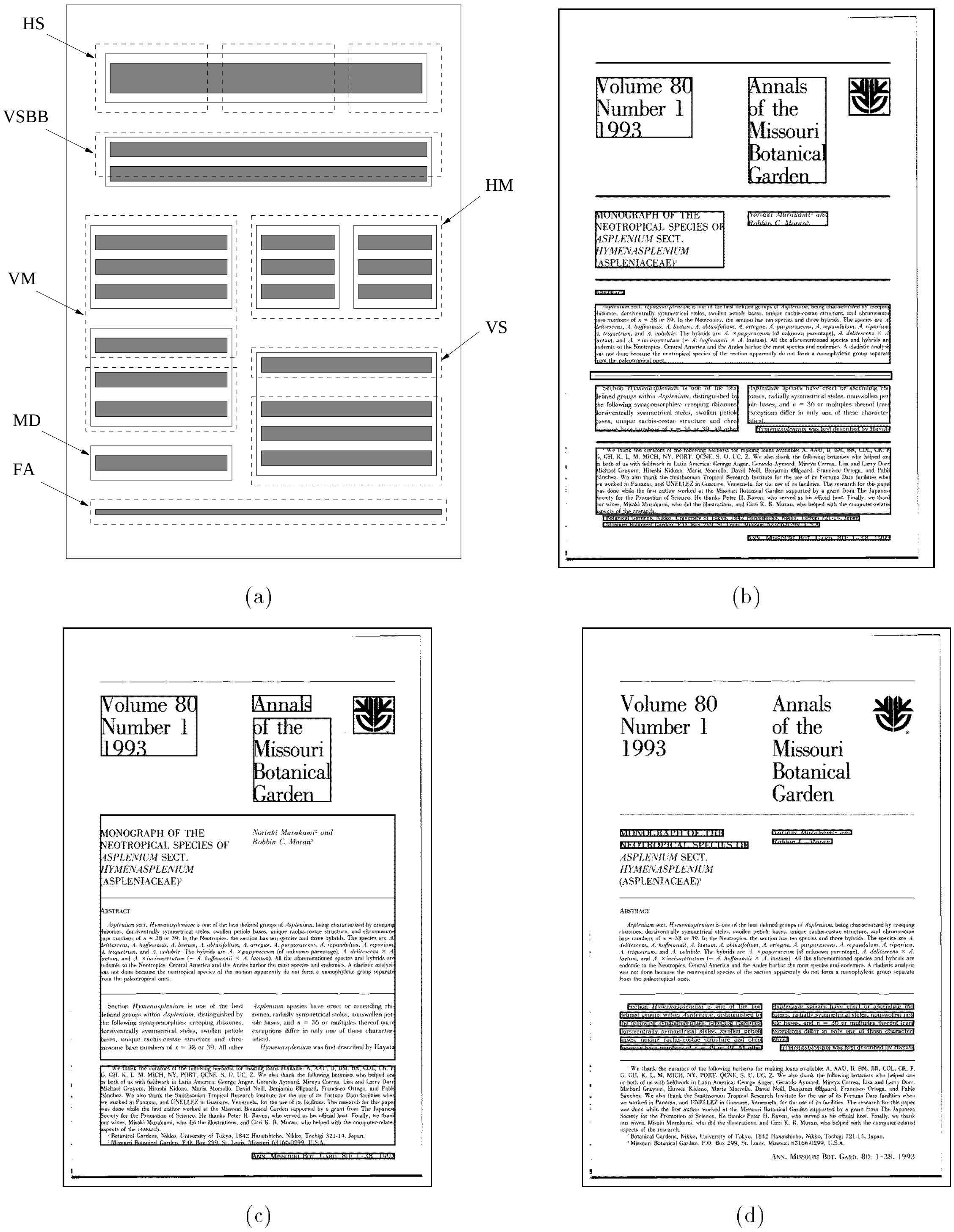
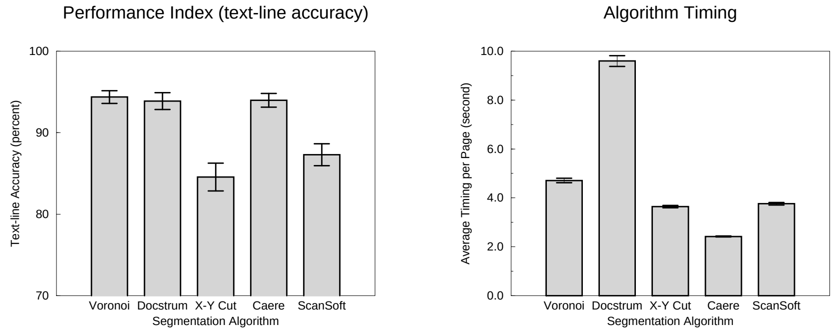
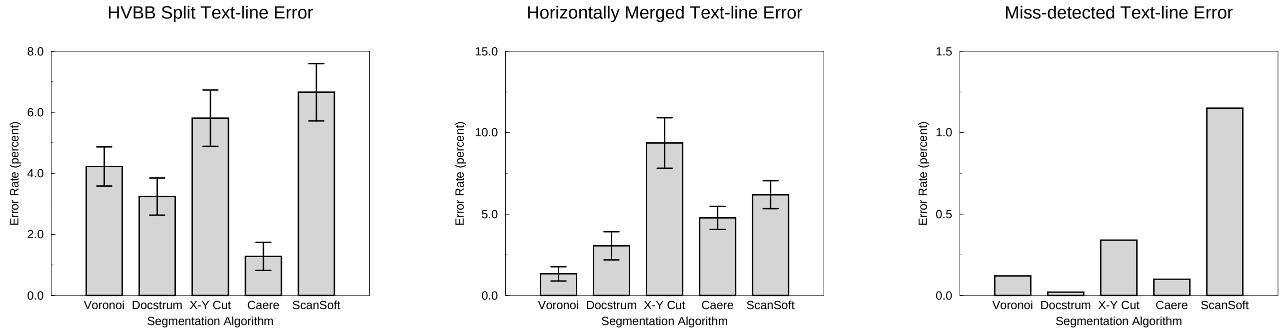


Figure 1. (a) This figure shows a set of possible text-line errors. Solid line rectangles denote groundtruth zones, dashed-line rectangles denote OCR segmentation zones and dark bars within groundtruth zones denote groundtruth text-lines, dark bars outside solid lines are noise blocks. HS is Horizontal Split, VSBB is Vertical Split on Text-line Bounding Box, VM is Vertical Merge, MD is Miss-detection, FA is False Alarm, HM is the Horizontal Merge and VS is Vertical Split. (b) This figure shows a document page with groundtruth zones. (c) This figure shows an OCR experimental segmentation result on this document page. (d) This figure shows segmentation error text-lines. Notice that there are two horizontally merged zones just below the caption and two horizontally merged zones in the middle of the text body. In OCR output, horizontally split zones cause error in terms of reading order whereas vertically split zones do not cause such errors.



(a) (b)

Figure 2. The first three algorithms in the bar chart are reseach algorithms, the last two algorithms are commercial products. (a) shows the testing results of performance index (text-line accuracy) for each algorithm. A 95% confidence t-test indicates that the performance of Voronoi, Docstrum and Caere are not significantly different, but the three are significantly better than ScanSoft, which in turn is significantly better than X-Y cut. (b) shows testing results of algorithm timing for each algorithm. A 95% confidence t-test indicates each algorithm's timing is significantly different from that of any other algorithm. From the least to most algorithm timing, algorithms are ranked as: Caere, X-Y cut, ScanSoft, Voronoi and Docstrum.



(a) (b) (c)

Figure 3. Page error rate is the ratio of the total number of error groundtruth text-lines and the total number of groundtruth text-lines. In this figure, we show the average page error rate. A 95% confidence t-test indicates: 1) for HVBB split text-line error shown in (a), the error rates of ScanSoft and X-Y Cut are not significantly different, but they are siginificantly higher than those of the other three algorithms. Moreover, the error rate of Voronoi, Docstrum and Caere are significantly different from each other; 2) for horizontally merged text-line error shown in (b), the error rate of all five algorithms are significantly different from each other.