Empirical Performance Evaluation Methodology and Its Application to Page Segmentation Algorithms

Song Mao and Tapas Kanungo, *Member*, *IEEE*

Abstract—While numerous page segmentation algorithms have been proposed in the literature, there is lack of *comparative* evaluation—empirical or theoretical—of these algorithms. In the existing performance evaluation methods, two crucial components are usually missing: 1) automatic training of algorithms with free parameters and 2) statistical and error analysis of experimental results. In this paper, 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 data sets with groundtruth, 2) we then select a meaningful and computable performance metric, 3) an optimization procedure is then used to search automatically for the optimal parameter values of the segmentation algorithms on the training data set, 4) the segmentation algorithms are then evaluated on the test data set, and, finally, 5) a statistical and error analysis is performed to give the statistical significance of the experimental results. In particular, instead of the ad hoc and manual approach typically used in the literature for training algorithms, we pose the automatic training of algorithms as an optimization problem and use the Simplex algorithm to search for the optimal parameter value. A paired-model statistical analysis and an error analysis are then conducted to provide confidence intervals for the experimental results of the algorithms. This methodology is applied to the evaluation of five page segmentation algorithms of which, three are representative research algorithms and the other two are well-known commercial products, on 978 images from the University of Washington III data set. It is found that the performance indices (average textline accuracy) of the Voronoi, Docstrum, and Caere segmentation algorithms 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.

Index Terms—Document page segmentation, OCR, performance evaluation, performance metric, statistical significance, paired model, direct search, simplex method.

1 Introduction

Optical Character Recognition (OCR) is the automated process of translating an input document image into a symbolic text file. The input document images can come from a large variety of media, such as journals, newspapers, magazines, memos, etc. The format of a document image can be digitally created, faxed, scanned, machine printed, or handwritten, etc. The output symbolic text file from an OCR system can include not only the text content of the input document image but also additional descriptive information, such as page layout, font size and style, document region type, confidence level for the recognized characters, etc.

Page segmentation is a crucial preprocessing step in an OCR system. It is the process of dividing a document image into homogeneous zones, i.e., those zones that only contain one type of information, such as text, a table, a figure, or a halftone image. In many cases, OCR system accuracy heavily depends on the accuracy of the page segmentation algorithm. While numerous page segmentation algorithms have been proposed in the past, relatively little research effort has been devoted to the comparative evaluation—empirical or theoretical—of these algorithms.

The paper is organized as follows: In Section 2, we conduct a survey of related literature. In Section 3, we provide the problem definition for page segmentation, error measurements, and a metric. In Section 4, we outline our five-step empirical performance evaluation methodology. In Section 5, automatic algorithm training is posed as an optimization problem and a simplex algorithm is described. In Section 6, our paired model statistical analysis method is presented. In Section 7, the segmentation algorithms that we evaluated are described. In Section 8, the experimental protocol for conducting the training and testing experiments is presented. In Section 9, we report experimental results and provide a detailed discussion. Finally, in Section 10, we give our conclusions. We have reported part of the work presented in this paper in [21], [22], [23]. The software used for generating the results in this paper is described in [25], [24]. The performance metric proposed in this paper deals with text regions only. We plan to extend it to include nontext regions as well.

2 Literature Survey

Page segmentation algorithms can be categorized into three classes: top-down approaches, bottom-up approaches, and

- *The authors are with the Language and Media Processing Lab Center for Automation Research, University of Maryland, 3436 A.V. Williams, College Park, MD 20742. E-mail: {maosong, kanungo}@cfar.umd.edu.*

Manuscript received 15 Jan. 2000; revised 13 Nov. 2000; accepted 14 Nov. 2000.
Recommended for acceptance by D. Dori.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 111221.

hybrid approaches. Top-down algorithms start from the whole document image and iteratively split it into smaller ranges. The splitting procedure stops when some criterion is met and the obtained ranges constitute the final segmentation results. Bottom-up algorithms start from document image pixels and cluster the pixels into connected components which are then clustered into words, lines, or final zone segmentations. To some extent, hybrid algorithms are a mix of the above two approaches. The Docstrum algorithm of O'Gorman [28], the Voronoi-diagram-based algorithm of Kise et al. [17], the run-length smearing algorithm of Wahl et al. [37], the segmentation algorithm of Jain and Yu [13], and the text string separation algorithm of Fletcher and Kasturi [5] are typical bottom-up algorithms, while the X-Y cut by Nagy et al. [26] and the shape-directed-covers-based algorithm by Baird et al. [1] are top-down algorithms. Pavlidis and Zhou [30] proposed a hybrid algorithm using a split-and-merge strategy. A survey of OCR and page segmentation algorithms can be found in O'Gorman and Kasturi [29] and Jain and Yu [13]. A recent workshop [3] was devoted to addressing issues related to page segmentation.

While many segmentation algorithms have been proposed in the literature, relatively few researchers have addressed the issue of quantitative evaluation of segmentation algorithms. Several page segmentation performance evaluation methods have been proposed in the past. Kanai et al. [14] proposed a metric that is a weighted sum of the number of edit operations (insertions, deletions, and moves). They used this performance metric in their comparative evaluation of the automatic zoning accuracy of four commercial OCR products. The advantage of this method is that it requires only ASCII text groundtruth and, hence, does not require zone or textline bounding-box groundtruth. The limitations of this method are that it cannot specify the error location in the image, it is dependent on the OCR engine's recognition accuracy, and the metric cannot be computed for languages for which no OCR engine is available. Vincent et al. [35] proposed a bitmap-level region-based metric. The advantages of the Vincent et al. approach are that it can evaluate both text regions and nontext regions, it is independent of zone representation schemes, and the errors can be localized and categorized. A limitation of this method is that the metric is dependent on pixel noise. Liang et al. [20] described a region-area-based metric. The overlap area of a groundtruth zone and a segmentation zone is used to compute this performance metric.

In the computer vision area, numerous researchers have presented methods for empirical performance evaluation. For example, Hoover et al. [12] proposed an experimental framework for quantitative comparison of range image segmentation algorithms and demonstrated the methodology by evaluating four range segmentation algorithms. Kanungo et al. [15] described a four-step methodology for the evaluation of two detection algorithms. Phillips and Chhabra [32] presented a methodology for empirically evaluating graphics recognition systems. These methodologies have not addressed the issues of either automatic training of algorithms with free parameters or statistical analysis of experimental results. Phillips et al. [33] proposed the FERET evaluation methodology for face recognition algorithms. However, the problem addressed here is only face classification and, in particular, not face segmentation. A special issue of *IEEE Transactions on Pattern Analysis and Machine Intelligence* (vol. 21, 1999) was devoted to empirical evaluation of computer vision algorithms. Two workshops have been devoted to empirical evaluation techniques and methodologies in computer vision [2], [8].

In research segmentation algorithms that have user-specifiable parameters, typically the default parameter values are selected and no training method is explicitly specified [17], [1], [28], [13], [30], [5]. Similarly, in performance evaluation literature where the algorithm parameters can be set by evaluators, a set of parameter values are usually selected manually in the training procedure [12], [15], [32]. A common aspect of these parameter value selection methods and training methods is that a set of "optimal parameter values" are *manually* selected based on some assumption regarding the training data set. To objectively optimize a segmentation algorithm on a given training data set, a set of optimal parameter values should be *automatically* found by a training procedure. Automatic training of any algorithm with free parameters is actually an optimization problem. In the optimization area, there are a number of classes of optimization problems based on the properties of the given objective function. An in-depth discussion and classification of optimization problems can be found in Gill et al. [6]. In our case, the objective function corresponding to a performance metric for page segmentation algorithms cannot be rigorously defined mathematically. Instead, only function evaluations are possible. Hence, automatic training is posed as a *multivariate nonsmooth nonlinear function* optimization problem. Direct search algorithms are typically used for solving optimization problems involving this kind of objective function [19], [38], [34]. We chose the simplex search method proposed by Nelder and Mead [27]. "Simulated annealing" [18] and "genetic" [7] algorithms are possible global search algorithms that could have been used instead.

3 The Page Segmentation Problem and Error Metrics

We provide the definitions of our proposed textline-based error measures and metric based on set theory and mathematical morphology [10]. Let I be a document image, and let G be the groundtruth of I. Let $Z(G) = \{Z_q^G, q = 1, 2, \ldots, \#Z(G)\}$ be a set of groundtruth zones of document image I, where $\#$ denotes the cardinality of a set. Let $L(Z_q^G) = \{l_{qj}^G, j = 1, 2, \ldots, \#L(Z_q^G)\}$ be the set of groundtruth textlines in groundtruth zone Z_q^G. Let the set of all groundtruth textlines in document image I be $\mathcal{L} = \cup_{q=1}^{\#Z(G)} L(Z_q^G)$. Let A be a given segmentation algorithm and let $Seg_A(\cdot, \cdot)$ be the segmentation function corresponding to algorithm A. Let

$R = Seg_A(I, \mathbf{p}^A)$ be the segmentation result of algorithm A, where $\mathbf{p}^A$ is a parameter vector associated with algorithm A and $Z(R) = \{Z_k^R | k = 1, 2, \ldots, \#Z(R)\}$. Let $D(\cdot) \subseteq \mathcal{Z}^2$ be the domain of its argument. The groundtruth zones and textlines have the following properties: 1) $D(Z_q^G) \cap D(Z_{q'}^G) = \phi$ for $Z_q^G, Z_{q'}^G \in Z(G)$ and $q \neq q'$ and 2) $D(l_i^G) \cap D(l_{i'}^G) = \phi$ for $l_i^G, l_{i'}^G \in \mathcal{L}$ and $i \neq i'$. In our evaluation method, we evaluate deskewed document images with rectangular zones and textline groundtruth.

A meaningful and computable performance metric is essential for evaluating page segmentation algorithms quantitatively. While a performance metric is typically not unique and researchers can select a particular performance metric to study certain aspects of page segmentation algorithms, a set of error measurements is necessary. Let $T_X, T_Y \in Z^+ \cup \{0\}$ be two length thresholds (in number of pixels) that determine if the overlap is significant or not. Each of these thresholds is defined in terms of an absolute threshold and a relative threshold as:

$$T_X = \min\{HPIX, (100 - HTOL) \cdot h/100\},$$
$$T_Y = \min\{VPIX, (100 - VTOL) \cdot v/100\}.$$

The absolute thresholds, $HPIX$ and $VPIX$, are in number of pixels and the relative thresholds, $HTOL$ and $VTOL$, are in percentage, h, v are the minimum width and height (in number of pixels) of two regions that are tested for significant overlap. In our experiments, we set the thresholds as $HTOL = 90$, $VTOL = 80$, $HPIX = 11$, and $VPIX = 8$. Let

$$E(T_X, T_Y) = \{e \in Z^2 | -T_X \leq X(e) \leq T_X, -T_Y \leq Y(e) \leq T_Y\}$$

be a rectangle centered at $(0, 0)$ with a width of $2T_X + 1$ pixels and a height of $2T_Y + 1$ pixels, where $X(\cdot)$ and $Y(\cdot)$ denote the X and Y coordinates of the argument, respectively. We now define two morphological operations: dilation and erosion [10]. Let $A, B \subseteq Z^2$. Morphological *dilation* of A by B is denoted by $A \oplus B$ and is defined as $A \oplus B = \{c \in Z^2 | c = a + b \text{ for some } a \in A, b \in B\}$. Morphological *erosion* of A by B is denoted by $A \ominus B$ and is defined as $A \ominus B = \{c \in Z^2 | c + b \in A \text{ for every } b \in B\}$. We now define four types of textline-based error measurements:

1. Groundtruth textlines that are missed:

$$C_L = \\ \{l^G \in \mathcal{L} | (D(l^G) \ominus E(T_X, T_Y)) \subseteq (\cup_{Z^R \in Z(R)} D(Z^R))^c\}.$$

2. Groundtruth textlines whose bounding boxes are split:

$$S_L = \\ \{l^G \in \mathcal{L} | (D(l^G) \ominus E(T_X, T_Y)) \cap D(Z^R) \neq \phi, \\ (D(l^G) \ominus E(T_X, T_Y)) \cap (D(Z^R))^c \neq \phi, \\ \text{for some } Z^R \in Z(R)\}.$$

3. Groundtruth textlines that are horizontally merged:

$$M_L = \\ \{l_{qj}^G \in \mathcal{L} | \exists l_{q'j'}^G \in \mathcal{L}, Z^R \in Z(R), q \neq q', Z_q^G, Z_{q'}^G \in Z(G) \\ \text{such that } (D(l_{qj}^G) \ominus E(T_X, T_Y)) \cap D(Z^R) \neq \phi, \\ (D(l_{q'j'}^G) \ominus E(T_X, T_Y)) \cap D(Z^R) \neq \phi, \\ ((D(l_{qj}^G) \ominus E(0, T_Y)) \oplus E(\infty, 0)) \cap D(Z_{q'}^G) \neq \phi, \\ ((D(l_{q'j'}^G) \ominus E(0, T_Y)) \oplus E(\infty, 0)) \cap D(Z_q^G) \neq \phi\}.$$

4. Noise zones that are falsely detected (false alarm):

$$F_L = \\ \{Z^R \in Z(R) | D(Z^R) \subseteq (\cup_{l^G \in \mathcal{L}} (D(l^G) \ominus E(T_x, T_Y)))^c\}.$$

Let the number of groundtruth error textlines be $\#\{C_L \cup S_L \cup M_L\}$ (misdetected, split, or horizontally merged) and let the total number of groundtruth textlines be $\#\mathcal{L}$. A simple performance metric $\rho(I, G, R)$ is given below:

$$\rho(I, G, R) = \frac{\#\mathcal{L} - \#\{C_L \cup S_L \cup M_L\}}{\#\mathcal{L}}. \tag{1}$$

A more general metric can be defined as:

$$\rho(I, G, R) = (\#\mathcal{L} - wErr)/\#\mathcal{L},$$

where

$$wErr = \\ wC_L * \#C_L + wS_L * \#S_L + wM_L * \#M_L + wF_L * \#F_L$$

is the weighted sum of various error measurements, wC_L, wS_L, wM_L, and wF_L are the weights (between 0 and 1) of the corresponding error measurements. In this paper, we use the definition given in (1). Fig. 1 gives a set of possible errors as well as an experimental example. We see that this textline-based performance metric has the following features:

1. it is rigorously defined using set theory and mathematical morphology,
2. it is independent of zone shape,
3. it is independent of OCR recognition error,
4. it ignores the background information (white space, salt and pepper noise, etc.),
5. segmentation errors can be localized, and
6. quantitative evaluation of lower-level (e.g., textline, word, and character) segmentation algorithms can be readily achieved with little modification. This performance metric, however, does not deal with nontext regions and requires textline-level groundtruth.

4 Performance Evaluation Methodology

We now introduce a five-step methodology and identify three crucial components: automatic training, statistical analysis, and error analysis.

A large and representative data set is desirable in any performance evaluation task in order to give objective

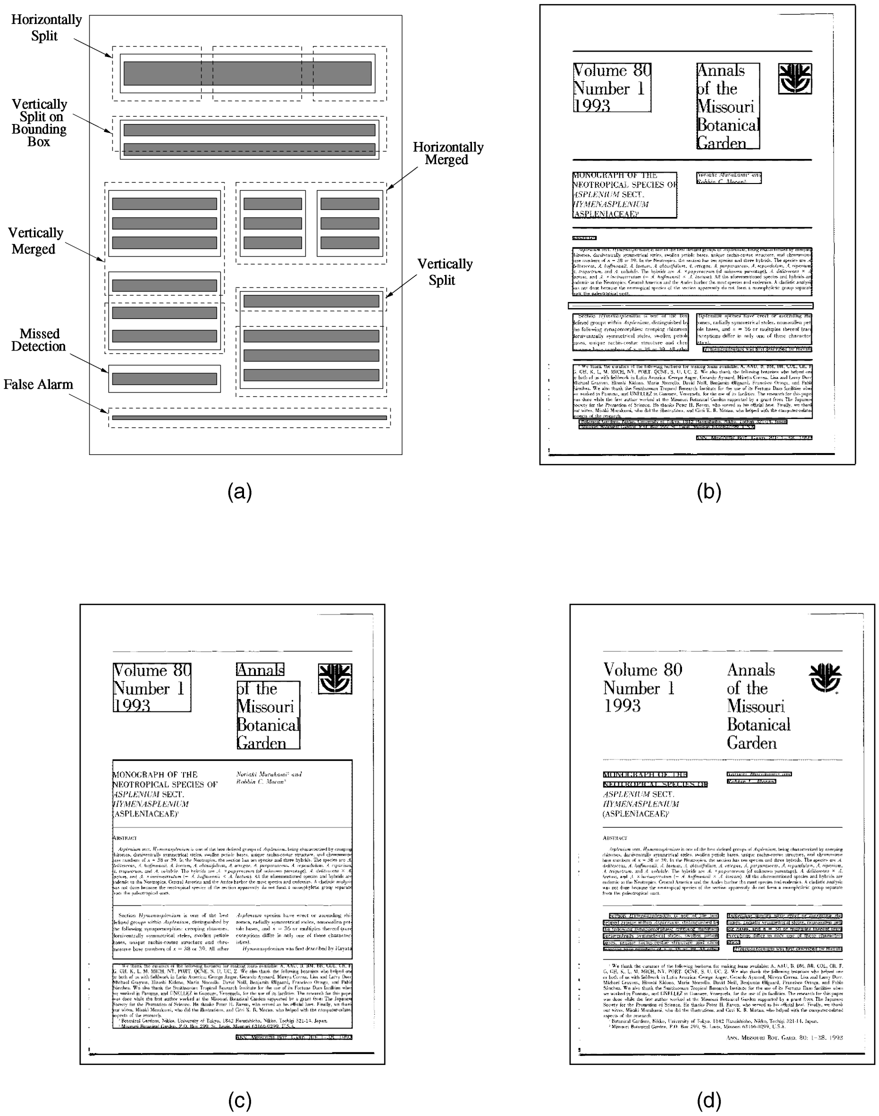


Fig. 1. (a) This figure shows a set of possible textline errors. Solid line rectangles denote groundtruth zones, dashed-line rectangles denote OCR segmentation zones, dark bars within groundtruth zones denote groundtruth textlines, and dark bars outside solid lines are noise blocks. (b) A document page image from the University of Washington III data set with the groundtruth zones overlaid. (c) OCR segmentation result on image in (b). (d) Segmentation error textlines. 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 reading order errors, whereas vertically split zones do not cause such errors.

performance measurements of the algorithms. A typical page segmentation algorithm has a set of parameters that affect its performance. The 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 data set, a set of optimum parameters has to be used. The optimum parameter set is a function of the given data set, the groundtruth, and the performance metric. The set of optimum parameters for one data set may be a nonoptimal parameter set for another data set. Hence, the choice of parameters is crucial in any performance evaluation task. When the size of the data set gets very large, parameter set training on the whole data set becomes computationally prohibitive and, therefore, a representative sample data set of much smaller size must be used as a training data set. After the training step, the page segmentation algorithms with the optimal parameters should be evaluated on a test data set that is different from the training set. Finally, in order to interpret the significance of the experimental results, statistical analysis should be performed. The relative strengths and weaknesses of the algorithms can then be understood by analyzing the errors. Let $\mathcal{D}$ be a given data set containing document image and groundtruth pairs (I, G). The steps in our methodology for evaluating page segmentation algorithms are as follows:

1. Randomly partition the data set $\mathcal{D}$ into a mutually exclusive training data set $\mathcal{T}$ and test data set $\mathcal{S}$. Thus, $\mathcal{D} = \mathcal{T} \cup \mathcal{S}$ and $\mathcal{T} \cap \mathcal{S} = \phi$, where ϕ is the empty data set.
2. Define a meaningful and computable performance metric $\rho(I, G, R)$, where I is an document image, G is

the groundtruth of I, and R is the segmentation result on I.

3. For a selected segmentation algorithm A, specify its parameter vector $\mathbf{p}^A$ and automatically find the optimal parameter setting $\hat{\mathbf{p}}^A$ for which an objective function $f(\mathbf{p}^A; \mathcal{T}, \rho, A)$ assumes the "best" measure on the training data set $\mathcal{T}$. In our case, this objective function is defined as the average textline error rate on a given data set.
4. Evaluate the segmentation algorithm A with optimized parameters $\hat{\mathbf{p}}^A$ on the test data set $\mathcal{S}$ by $\Phi(\{\rho(G, Seg_A(I, \hat{\mathbf{p}}^A)) | (I, G) \in \mathcal{S}\})$, where Φ is a function of the performance metric ρ on each document image and groundtruth pair (I, G) in the test data set $\mathcal{S}$, and $Seg_A(\cdot, \cdot)$ is the segmentation function corresponding to A. The function Φ is defined by the user. In our case,

$$\Phi(\{\rho(G, Seg_A(I, \hat{\mathbf{p}}^A)) | (I, G) \in \mathcal{S}\}) = 1 - f(\hat{\mathbf{p}}^A; \mathcal{S}, \rho, A),$$

   which is the average of the performance metric $\rho(G, Seg_A(I, \hat{\mathbf{p}}^A))$ (textline accuracy) on each document image and groundtruth pair (I, G) in the test data set $\mathcal{S}$.
5. Perform statistical analysis to find the significance of the evaluation results and analyze the errors to identify/hypothesize why the algorithms perform at the respective levels.

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.

5 Automatic Algorithm Training: The Optimization Problem

Any automatic training or learning problem can be posed as an optimization problem. An optimization problem has three components: the objective function that gives a single measure, a set of parameters that the objective function is dependent on, and a parameter subspace that defines acceptable or reasonable parameter values. The acceptable or reasonable parameter subspace defines the constraints on the optimization problem. The purpose of an optimization procedure is to find a set of parameter values for which the objective function gives the "best" (minimum or maximum) measure values.

5.1 The Objective Function

In this section, we identify the objective function. Let $\mathbf{p}^A$ be the parameter vector for the segmentation algorithm A, let $\mathcal{T}$ be a training data set, and let $\rho(I, G, Seg_A(I, \mathbf{p}^A))$ be a performance metric where $(I, G) \in \mathcal{T}$. We define the objective function $f(\mathbf{p}^A; \mathcal{T}, A, \rho)$ to be minimized as the average textline error rate on the training data set:

$$f(\mathbf{p}^A; \mathcal{T}, A, \rho) = \frac{1}{\#\mathcal{T}} \left[\sum_{(I,G) \in \mathcal{T}} 1 - \rho(G, Seg_A(I, \mathbf{p}^A)) \right], \quad (2)$$

where ρ is defined in (1). This objective function has the following properties:

1. It is dependent on the values of the algorithm parameters.
2. The function value is the only information available.
3. The function has no explicit mathematical form and is nondifferentiable.
4. Obtaining a function value requires nontrivial computation.

This objective function can be classified as a *multivariate nonsmooth function* [6]. In the following section, we describe an optimization algorithm to minimize this objective function.

5.2 The Simplex Search Method

Direct search methods are typically used to solve the optimization problem described in Section 5.1. We choose the simplex search method proposed by Nelder and Mead [27] to minimize our objective function. Let $\mathbf{q}_0$ be a starting point in segmentation algorithm parameter space and let $\lambda_i, i = 1, \ldots, n$ be a set of scales. Let $\mathbf{e}_i, i = 1, \ldots, n$ be n orthogonal unit vectors in n-dimensional parameter space, let $\mathbf{p}_0, \ldots, \mathbf{p}_n$ be $(n+1)$ ordered points in n-dimensional parameter space such that their corresponding function values satisfy $f_0 \leq f_1 \leq, \ldots, \leq f_n$, let $\bar{\mathbf{p}} = \sum_{i=0}^{n-1} \mathbf{p}_i / n$ be the centroid of the n best (smallest) points, let $[\mathbf{p}_i \mathbf{p}_j]$ be the n-dimensional Euclidean distance from $\mathbf{p}_i$ to $\mathbf{p}_j$, let α, β, γ, and σ be the *reflection, contraction, expansion,* and *shrinkage* coefficient, respectively, and let T be the threshold for the stopping criterion. We use the standard choice for the coefficients: $\alpha = 1$, $\beta = 0.5$, $\gamma = 2$, $\sigma = 0.5$. We set T to 10^{-6}. For a segmentation algorithm with n parameters, the Nelder-Mead algorithm works, as shown in Fig. 2.

5.3 Starting Point Selection

The objective function corresponding to each segmentation algorithm need not have a unique minimum. Furthermore, direct search optimization algorithms are *local* optimization algorithms. Thus, for each (different) starting point, the optimization algorithm could converge to a different optimal solution. We constrain the parameter values to lie within a reasonable range and randomly choose six starting locations within this range. The optimal solution corresponding to the lowest optimal value is chosen as the best optimal parameter vector.

6 Statistical Analysis: A Paired Model Approach

In comparative performance evaluation frameworks, statistical analysis plays a crucial role in objectively interpreting the experimental results. In our experiments, we compare the performance metric values (average textline accuracy) of page segmentation algorithms against each other. In doing so, some basic questions are immediately raised: 1) If the performance metric of one algorithm is better than that of another algorithm, is the result statistically significant? 2) What is the uncertainty in the estimated performance metric? To answer such questions, a statistical model needs to be constructed for the experimental observations. In this

```
1 Given $\mathbf{q}_0$ and the $\lambda_i$, form the initial simplex as $\mathbf{q}_i = \mathbf{q}_0 + \lambda_i \mathbf{e}_i, i = 1, \ldots, n.$,
2 Relabel the $n+1$ vertices as $\mathbf{p}_0, \ldots, \mathbf{p}_n$ with $f(\mathbf{p}_0) \leq f(\mathbf{p}_1) \cdots \leq f(\mathbf{p}_n)$,
3 Get a reflection point $\mathbf{p}_r$ of $\mathbf{p}_n$ by $\mathbf{p}_r = (1+\alpha)\bar{\mathbf{p}} - \alpha \mathbf{p}_n$ where $\alpha = [\mathbf{p}_r\bar{\mathbf{p}}]/[\mathbf{p}_n\bar{\mathbf{p}}]$.
4.1 If $f(\mathbf{p}_r) \leq f(\mathbf{p}_0)$, replace $\mathbf{p}_n$ by $\mathbf{p}_r$ and $f(\mathbf{p}_n)$ by $f(\mathbf{p}_r)$, get an expansion point
    $\mathbf{p}_e$ of $\mathbf{p}_n$ by $\mathbf{p}_e = (1-\gamma)\bar{\mathbf{p}} + \gamma \mathbf{p}_n$ where $\gamma = [\mathbf{p}_e\bar{\mathbf{p}}]/[\mathbf{p}_n\bar{\mathbf{p}}] > 1$.
    If $f(\mathbf{p}_e) < f(\mathbf{p}_n)$, replace $\mathbf{p}_n$ by $\mathbf{p}_e$ and $f(\mathbf{p}_n)$ by $f(\mathbf{p}_e)$. Go to step 5.
4.2 Else if $f(\mathbf{p}_r) \geq f(\mathbf{p}_{n-1})$, if $f(\mathbf{p}_r) < f(\mathbf{p}_n)$ replace $\mathbf{p}_n$ by $\mathbf{p}_r$ and $f(\mathbf{p}_n)$ by $f(\mathbf{p}_r)$,
    get a contraction point $\mathbf{p}_c$ of $\mathbf{p}_n$ by $\mathbf{p}_c = (1-\beta)\bar{\mathbf{p}} + \beta \mathbf{p}_n$, $\beta = [\mathbf{p}_c\bar{\mathbf{p}}]/[\mathbf{p}_n\bar{\mathbf{p}}] < 1$.
    If $f(\mathbf{p}_c) \geq f(\mathbf{p}_n)$, shrink the simplex around the best vertex $\mathbf{p}_0$ by $\mathbf{p}_i = (\mathbf{p}_i + \mathbf{p}_0)\sigma$,
    $i \neq 0$, else replace $\mathbf{p}_n$ by $\mathbf{p}_c$ and $f(\mathbf{p}_n)$ by $f(\mathbf{p}_c)$, go to step 5.
4.3 Else, replace $\mathbf{p}_n$ by $\mathbf{p}_r$ and $f(\mathbf{p}_n)$ by $f(\mathbf{p}_r)$ .
5 If $\sqrt{\sum_{i=0}^{n}(f(\mathbf{p}_i) - f(\bar{\mathbf{p}}))^2/n} < T$, stop else go to step 2.
```

Fig. 2. The Nelder-Mead optimization algorithm.

section, we describe a paired model analysis approach proposed by Kanungo et al. [16] for their evaluation of Arabic OCR engines and adapt it to analyze our experimental results.

6.1 Modeling Experimental Data Using the Paired Model

Let $A_1, A_2 \ldots, A_k$ denote the k algorithms we evaluate and let $X_{ij}, i = 1, \ldots, k,\ j = 1, \ldots, n$ be the observation (textline accuracy) corresponding to algorithm A_i and document image I_j in test data set $\mathcal{S}$. In our experiment, the number of algorithms k is five and the total number of images n in test data set $\mathcal{S}$ is 878. We assume that the observations from different images are statistically independent, i.e., that X_{ij} and $X_{i'j'}$ are independent when $j \neq j'$. We also assume for a fixed algorithm A_i, that observations $X_{ij}, j = 1, \ldots, n$, are iid random variables with finite mean μ_i and finite variance σ_i^2. However, the observations X_{ij} and $X_{i'j}$ corresponding to two different algorithms, i.e., $i \neq i'$, on the *same* image I_j are statistically dependent since the two algorithms use the same image as input. We assume that the correlation coefficient $\rho_{ii'}$ of observations of algorithm A_i and $A_{i'}$ on the same page is constant. This $\rho_{ii'}$ is positive since a document image that causes an algorithm to generate a bad performance metric generally will also cause other algorithms to generate bad performance metrics. Finally, let $cov(X_{ij}, X_{i'j}) = \rho_{ii'}\sigma_i\sigma_{i'}$ be the covariance of observations X_{ij} and $X_{i'j}$, where $i \neq i'$.

Now, construct a new random variable

$$W_{ii'j} = X_{ij} - X_{i'j}, i \neq i',$$

where $W_{ii'j}$ and $W_{ii'j'}$ are independent. Based on our assumptions, it is easily seen that $W_{ii'j}$s are iid random variables for fixed i and i'. Let $\bar{W}_{ii'}$ and $V_{ii'}^2$ be the sample mean and sample variance of $W_{ii'j}$ and let $\Delta_{ii'}$ be true mean difference such that $\Delta_{ii'} = \mu_i - \mu_{i'}$. An unbiased estimator of $\Delta_{ii'}$ is $\hat{\Delta}_{ii'} = \bar{W}_{ii'} = \bar{X}_i - \bar{X}_{i'}$ since

$$E[\hat{\Delta}_{ii'}] = E[\bar{W}_{ii'}] = E[\bar{X}_i - \bar{X}_{i'}] = \mu_i - \mu_{i'} = \Delta_{ii'}.$$

The variance of the estimator $\hat{\Delta}_{ii'}$ is

$$Var[\hat{\Delta}_{ii'}] = $$
$$Var[\bar{W}_{ii'}] = Var[\bar{X}_i - \bar{X}_{i'}] = (\sigma_i^2 + \sigma_{i'}^2 - 2\rho_{ii'}\sigma_i\sigma_{i'})/n.$$

6.2 Confidence Intervals and Hypothesis Testing

We first address the issue of uncertainty in performance estimates. Since $W_{ii'j}$ are iid random variables, for fixed i and i', where $i \neq i'$, by the Central Limit Theorem, we have (3), where $\sigma_{ii'}$ is the true standard deviation of $\hat{\Delta}_{ii'}$. When $\sigma_{ii'}$ is not available as it is in our case, the sample standard deviation $V_{ii'}$ is typically used in place of $\sigma_{ii'}$ in (4). The new formula has an approximate t distribution with $n-1$ degrees of freedom. Thus, for a given significance level α, we can compute a confidence interval, as shown in (5).

$$\lim_{n\to\infty} \frac{\hat{\Delta}_{ii'} - \Delta_{ii'}}{\sigma_{ii'}/\sqrt{n}} = \lim_{n\to\infty} \frac{\bar{W}_{ii'} - (\mu_i - \mu_{i'})}{\sigma_{ii'}/\sqrt{n}} \sim N(0,1), \tag{3}$$

$$\frac{\hat{\Delta}_{ii'} - \Delta_{ii'}}{V_{ii'}/\sqrt{n}} = \frac{\bar{W}_{ii'} - (\mu_i - \mu_{i'})}{V_{ii'}/\sqrt{n}} \sim t_{n-1}, \tag{4}$$

$$\Delta_{ii'} \in \hat{\Delta}_{ii'} \pm \frac{t_{\alpha/2,n-1}V_{ii'}}{\sqrt{n}}. \tag{5}$$

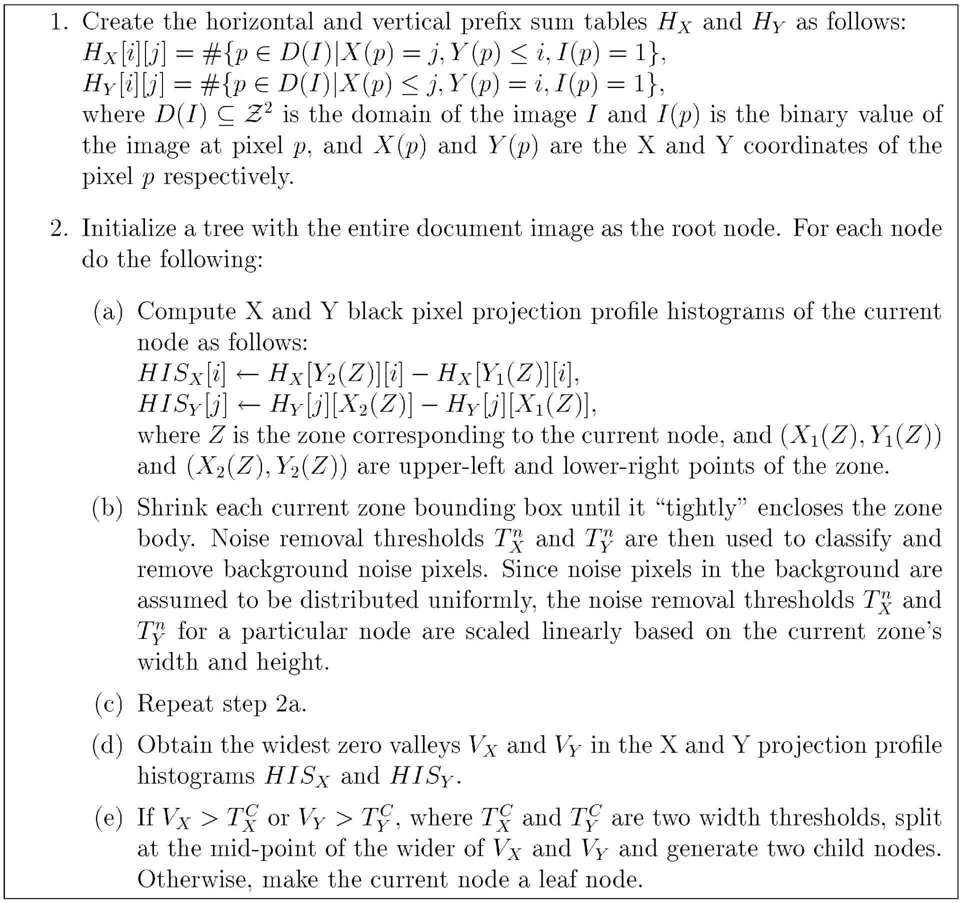
The second problem we want to address is whether or not one algorithm is performing significantly better than another. That is, we want to test the hypothesis that the true means of the observations from two different algorithms are significantly different. Let $f(t)$ be the probability density function (pdf) of the t distribution with $n-1$ degrees of freedom. Let $T(X_{i1}, \ldots, X_{in}, X_{i'1}, \ldots, X_{i'n})$ be the test statistic, which is a function of the observations. For a given significance level α, the corresponding hypothesis test can be formulated as follows: Let the null hypothesis be $H_0 : \Delta_{ii'} = \mu_i - \mu_{i'} = 0$, the alternative hypothesis be $H_a : \Delta_{ii'} = \mu_i - \mu_{i'} \neq 0$, and the test statistic be

$$T = T(X_{i1}, \ldots, X_{in}, X_{i'1}, \ldots, X_{i'n}) = (\hat{\Delta}_{ii'} - 0)/(V_{ii'}/\sqrt{n}).$$

Under the null hypothesis H_0, the test statistic T is distributed approximately as a t distribution with $n-1$ degrees of freedom. Now, define $P_{val} = \int_{-\infty}^{-T} f(t)dt + \int_{T}^{\infty} f(t)dt$. For a test with a significance level of α, reject the null hypothesis H_0 if $P_{val} < \alpha$.

6.3 Advantages of the Paired Model Analysis

This paired test is valid even if $\sigma_i^2 \neq \sigma_{i'}^2$, where $i \neq i'$. We do not need to assume a distribution for observation X_{ij}. Since



```
1. Create the horizontal and vertical prefix sum tables $H_X$ and $H_Y$ as follows:
   $H_X[i][j] = \#\{p \in D(I) | X(p) = j, Y(p) \leq i, I(p) = 1\}$,
   $H_Y[i][j] = \#\{p \in D(I) | X(p) \leq j, Y(p) = i, I(p) = 1\}$,
   where $D(I) \subseteq \mathcal{Z}^2$ is the domain of the image $I$ and $I(p)$ is the binary value of
   the image at pixel $p$, and $X(p)$ and $Y(p)$ are the X and Y coordinates of the
   pixel $p$ respectively.

2. Initialize a tree with the entire document image as the root node. For each node
   do the following:

   (a) Compute X and Y black pixel projection profile histograms of the current
       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, and $(X_1(Z), Y_1(Z))$
       and $(X_2(Z), Y_2(Z))$ are upper-left and lower-right points of the zone.
   (b) Shrink each current zone bounding box until it "tightly" encloses 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 the current zone's
       width and height.
   (c) Repeat step 2a.
   (d) Obtain the widest zero valleys $V_X$ and $V_Y$ in the X and Y projection profile
       histograms $HIS_X$ and $HIS_Y$.
   (e) If $V_X > T_X^C$ or $V_Y > T_Y^C$, where $T_X^C$ and $T_Y^C$ are two width thresholds, split
       at the mid-point of the wider of $V_X$ and $V_Y$ and generate two child nodes.
       Otherwise, make the current node a leaf node.
```

Fig. 3. The X-Y Cut segmentation algorithm.

the correlation of observations on the same image is considered, a smaller variance of $(\sigma_i^2 + \sigma_{i'}^2 - 2\rho_{ii'}\sigma_i\sigma_{i'})/n$ is obtained for the estimator $\hat{\Delta}_{ii'}$ of $\Delta_{ii'}$ than the variance of $(\sigma_i^2 + \sigma_{i'}^2)/n$ in the case where this correlation is ignored, i.e., the two samples $X_{i1}, \ldots, X_{in}$ and $X_{i'1}, \ldots, X_{i'n}$ are assumed to be independent. In other words, a more precise estimate of $\Delta_{ii'}$ is obtained if we use the paired model.

7 Page Segmentation Algorithms

Page segmentation algorithms can be categorized into three classes: top-down approaches, bottom-up approaches, and hybrid approaches. We implemented the X-Y cut algorithm (a top-down algorithm) and the Docstrum algorithm (a bottom-up algorithm). Kise et al. [17] provided us with a C implementation of their Voronoi-based algorithm (a bottom-up algorithm). Two commercial products, Caere's segmentation algorithm [4] and ScanSoft's segmentation algorithm [36], were selected for evaluation. They are representative state-of-art commercial products. Both are black-box algorithms with no free parameters. In the following sections, we describe the three research algorithms.

7.1 The X-Y Cut Page Segmentation Algorithm

The X-Y cut segmentation algorithm [26] is a tree-based, top-down algorithm. The root node of the tree represents the entire document page image I, 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 deskewed document pages with Manhattan layout and rectangular zones. The algorithm works, as shown in Fig. 3.

7.2 The Docstrum Page Segmentation Algorithm

Docstrum [28] is a bottom-up page segmentation algorithm that can work on document page images with non-Manhattan layout and arbitrary skew angles. This algorithm is not designed to handle nontext regions and text zones with irregular font sizes and spacings and tends to fragment them. Moreover, it does not perform well when document images contain sparse characters. The basic steps of the Docstrum segmentation algorithm are shown in Fig. 4. In our implementation, we did not estimate orientation since all pages in the data set were 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.

7.3 The Voronoi-Diagram-Based Page Segmentation Algorithm

The segmentation algorithm in [17] is also a bottom-up algorithm based on the Voronoi diagram. This method can work on document page images that have non-Manhattan layout, arbitrary skew angles, or nonlinear textlines. A set of connected line segments are used to bound text zones. Since we evaluate all algorithms on document page images with

1. Obtain connected components (C_is) using a space-efficient two-pass algorithm [9].
2. Remove small and large noise or nontext connected components using low and high thresholds l and h.
3. Separate the C_is into two groups, one with dominant characters and the other 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 using the sorting approach [28].
5. Compute the distance and angle of each C_i and its K nearest neighbors: (ρ^i_j, θ^i_j), such that $j \in \mathrm{NN}_K(i)$.
6. Compute a within-line nearest-neighbor distance histogram from the following set $W_\rho : W_\rho = \{\rho^i_j | j \in \mathrm{NN}_K(i), \text{ and } -\theta_h \leq \theta^i_j \leq \theta_h\}$, 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 a between-line nearest-neighbor distance histogram from the set $B_\rho : B_\rho = \{\rho^i_j | j \in \mathrm{NN}_K(i), \text{ and } 90^\circ - \theta_v \leq \theta^i_j \leq 90^\circ + \theta_v\}$, 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 textlines L_is using within-line nearest neighbor distance threshold $T_{cs} = f_t \cdot cs$.
9. Perform transitive closure on the L_is to obtain structural blocks or zones Z_is using parallel distance threshold $T_{pa} = f_{pa} \cdot cs$ and perpendicular distance threshold $T_{pe} = f_{pe} \cdot ls$. The parallel and perpendicular distances are computed as "end–end" distance, not "centroid–centroid" distance.

Fig. 4. The Docstrum segmentation algorithm.

Manhattan layouts, this algorithm has been modified to generate rectangular zones. This algorithm has design limitations similar to those of the Docstrum algorithm. The algorithm steps are shown in Fig. 5.

8 Experimental Protocol

The experiment we conducted has a training phase and a testing phase for the three research algorithms and only a testing phase for the two commercial products since they do not have user-specifiable free parameters. We used textline accuracy as our performance metric. For each document page, we obtained a performance metric value. In the training phase, we computed an average error rate, which is equal to 1 minus the average performance index, over all document pages in the training data set $\mathcal{T}$. In the testing phase, we computed an average performance index over all document pages in the test data set $\mathcal{S}$ and report it as the algorithm performance index.

1. Label connected components. Sample points on their borders. The 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. The Voronoi diagram for each connected component is generated using the sample points on its border.
4. Delete superfluous Voronoi edges to generate text zone boundaries according to a spacing and area-ratio criteria [17].
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.

Fig. 5. The Voronoi-based segmentation algorithm.

8.1 Data Set Specification

We selected the University of Washington Data Set [31] for the performance evaluation task since it is the only data set currently available that has textline-level groundtruth for each document page. All pages in the data set are journal pages from a large variety of journals in diverse subject areas and from different publishers. The data set also has geometric textline and zone groundtruth for each page. The textline and zone groundtruth are represented by nonoverlapping rectangles. The University of Washington III data set has 1,601 deskewed binary document images at 300 dpi resolution. We chose a subset of 978 pages that correspond to the University of Washington I data set pages as our experimental data set. A training data set $\mathcal{T}$ of 100 document pages was randomly sampled from the selected 978 documents; the remaining 878 document pages are considered as the test data set $\mathcal{S}$.

8.2 Algorithm Training and Testing

We fix the parameters that the algorithm is insensitive to and automatically train the ones that the algorithm is sensitive to on the 100-page training data set $\mathcal{T}$. Nelder-Mead simplex optimization procedure [27] is used to search for the optimal parameter value for each algorithm. Based on information about the document page style, a reasonable working range can be selected for each parameter of each algorithm. Six different starting points within the reasonable working parameter subspace for each research algorithm were randomly selected and the corresponding six convergence points were obtained. Then, we selected the parameter values corresponding to the minimum of the six optimal values attained in the six searches. Since the two commercial products are black-box algorithms without any parameters, we do not perform the training step for them. All five algorithms were tested on the 878-page test data set $\mathcal{S}$. We trained the three research algorithms on the same type of UNIX machine and tested all five segmentation algorithms on the same type of PC machine.

8.2.1 X-Y Cut Algorithm Parameters

The X-Y cut algorithm [26] has four free parameters. Since the algorithm is very sensitive to all four parameters, we searched for the optimal value for each of the four parameters over the reasonable working ranges: Vertical noise removal threshold T_X^n: {20-250 pixels}; Horizontal noise removal threshold T_Y^n: {20-200 pixels}; X widest zero valley width threshold T_X^C: {20-100 pixels}; Y widest zero valley width threshold T_Y^C: {20-100 pixels}. In most cases, since the vertical cut is longer than the horizontal cut, we set the maximum of T_X^n to be larger than that of T_Y^n. Furthermore, since most interline gaps are less than 100 pixels, we set the maximum of T_X^C and T_Y^C to 100 pixels. We set the simplex scales to be $\lambda_i = 20$, where $i = 1, 2, 3, 4$.

8.2.2 Docstrum Algorithm Parameters

O'Gorman, in his paper, specified eight parameters for the Docstrum algorithm [28]. We introduced two additional parameters for textline segmentation control and character grouping: 1) a superscript-subscript character distance threshold factor for correctly handling textline segmentation and 2) a character size ratio threshold to separate larger characters from dominant characters. The algorithm is insensitive to six of the ten parameters. We fixed these six parameters as follows: number of nearest connected components for clustering, $K = 9$; low and high connected component size-thresholds (height or width), $l = 2$ pixels, $h = 200$ pixels; horizontal and vertical angle tolerance thresholds, $\theta_h = 30^\circ$, $\theta_v = 30^\circ$; superscript and subscript character distance threshold factor, $f_s = 0.4$. The values for the four parameters that the algorithm is sensitive to were searched for in the reasonable working ranges: nearest-neighbor threshold factor f_t: {1-5}, Parallel distance threshold factor f_{pa}: {2-10}, perpendicular distance threshold factor f_{pe}: {0.5-5}, and character size ratio factor f_d: {2-10}. We set the simplex scales to be $\lambda_i = 1$, where $i = 1, 2, 3, 4$.

8.2.3 Voronoi-Diagram-Based Algorithm Parameters

The Voronoi diagram-based algorithm has eleven free parameters and is insensitive to seven of them [17]. Six of these eleven parameters are related to removing noise connected components and blocks. The algorithm is insensitive to another of these eleven parameters, sw. We fixed the seven parameters as follows: maximum height and width thresholds of a connected component, $C_h = 500$ pixels and $C_w = 500$ pixels, maximum connected component aspect ratio threshold, $C_r = 5$, minimum area threshold of a zone, $A_z = 50\ \mathrm{pixels}^2$ for all zones and minimum area threshold, $A_l = 40,000\ \mathrm{pixels}^2$ 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, which is fixed at $sw = 2$. The optimal values for the other four parameters are searched for in the following ranges recommended by Kise:[1] sampling rate sr: {4-7}, max size threshold of noise connected component nm: {10-40 pixels}, margin control factor for Td2 fr: {0.01-0.5}, and area ratio threshold ta: {40-200}. We set the simplex scales to be $\lambda_1 = 1, \lambda_2 = 10, \lambda_3 = 0.1$, and $\lambda_4 = 40$.

8.3 Hardware and Software Environments

We implemented the X-Y cut and Docstrum algorithms based on [26], [28]. We used the PSET software package [25], [24] for training and testing the algorithms. The platform used for the implementation and algorithm training was Ultra 10 Sun workstations running the Solaris 2.6 operating system. The clock rate of the machine reported by the command "fpversion" was 440MHz. The main memory is 512 MB. The compiler used was GNU gcc 2.7.2. Kise et al. [17] provided us with a C implementation of the Voronoi-based segmentation algorithm. The platform used for the testing was a Gateway PC with a 400 MHz Pentium II CPU and 128 MB main memory. To test the three research algorithms, we ported the PSET software package to the PC running Linux 7.0 operating system. To test the two products on the PC, we wrote programs in the Visual C++ 5.0/Windows 95 environment to extract zone coordinates from the OCR output.

[1] Personal communication, October 20, 1999.

TABLE 1
Optimal Parameter Values for Each Research Algorithm

Algorithm	Optimal Parameter Value	Error Rate (percent)	Function Evaluations	Timing (hours)	Operating System	Platform Type
X-Y cut	$(78, 32, 35, 54)$	14.71	135	8.84	Solaris 2.6	Ultra 10 Sun Workstation, clock rate 440 MHz, main memory 512 MB.
Docstrum	$(2.577, 2.341, 0.597, 9.928)$	5.00	168	6.85		
Voronoi	$(6, 11, 0.083, 199)$	4.74	80	7.06		

9 Experimental Results and Discussion

In this section, four aspects of the experimental results are reported: training, test, statistical analysis, and error analysis.

9.1 Training Results

Three research algorithms were trained on a 100-page training data set $\mathcal{T}$. Table 2 reports the optimum parameters, optimum performance index (average textline accuracy) value, and training time corresponding to each randomly selected starting point for the X-Y cut, Docstrum, and Voronoi algorithms. We consider the parameter values that give the lowest error rate as a set of optimal parameter values for each research algorithm, as shown in Table 1. Fig. 6 shows the convergence characteristics for the X-Y cut, Docstrum, and Voronoi algorithms. The findings from the training results for each research algorithm are summarized as follows:

- The X-Y cut algorithm. From Table 2a and Fig. 6a, we can make the following observations:

1. The error rates for all starting points converge in the range of 14.71 percent to 18.96 percent.
2. The convergence rate before the first 30 function evaluations is much faster than that beyond 30 function evaluations.
3. Most optimum values of parameter T_X^n are larger than those of parameter T_Y^n.
4. All optimum values of parameter T_X^C are smaller than those of parameter T_Y^C.
5. There is smaller variation in the optimal values of parameter T_X^C than that in optimal values of other parameters.
6. There is a fair amount of variation in the optimal parameter values.

From the above observations, we can see that the X-Y cut algorithm objective function has multiple local minima and the performance at these local

TABLE 2
Optimization Results of (a) the X-Y Cut Algorithm, (b) the Docstrum Algorithm, and (c) the Voronoi-Based Algorithm for Six Randomly Selected Starting Points within a Reasonable Working Parameter Subspace

Starting Parameter Values $(T_X^n, T_Y^n, T_X^C, T_Y^C)$	Optimal Parameter Values $(T_X^n, T_Y^n, T_X^C, T_Y^C)$	Error Rate (percent)	Number of Function Evaluations	Timing (hours)
$(140, 80, 50, 70)$	$(128, 62, 23, 91)$	18.96	147	9.62
$(120, 120, 10, 80)$	$(97, 107, 22, 97)$	18.57	105	6.85
$(80, 40, 70, 50)$	$(82, 64, 21, 89)$	15.52	85	5.56
$(60, 120, 10, 20)$	$(67, 55, 21, 70)$	14.96	140	9.18
$(100, 80, 100, 50)$	$(78, 58, 22, 70)$	15.79	147	9.55
$(80, 20, 70, 50)$	$(78, 32, 35, 54)$	14.71	135	8.84

(a)

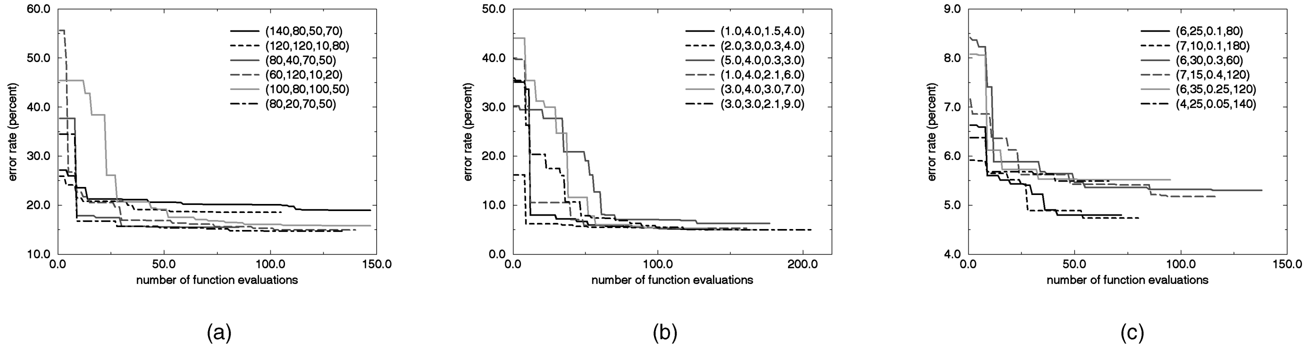
Starting Parameter Values $(f_t, f_{pa}, f_{pe}, f_d)$	Optimal Parameter Values $(f_t, f_{pa}, f_{pe}, f_d)$	Error Rate (percent)	Number of Function Evaluations	Timing (hours)
$(1.0, 4.0, 1.5, 4.0)$	$(2.330, 2.341, 0.600, 5.236)$	5.01	168	7.09
$(2.0, 3.0, 0.3, 4.0)$	$(2.354, 2.132, 0.600, 4.379)$	5.44	113	4.70
$(5.0, 4.0, 0.3, 3.0)$	$(3.072, 2.138, 0.303, 3.603)$	6.30	177	7.23
$(1.0, 4.0, 2.1, 6.0)$	$(2.533, 1.975, 0.647, 7.547)$	5.34	161	6.58
$(3.0, 4.0, 3.0, 7.0)$	$(2.577, 2.341, 0.597, 9.928)$	5.00	168	6.85
$(3.0, 3.0, 2.1, 9.0)$	$(2.521, 2.336, 0.595, 10.375)$	5.00	208	8.58

(b)

Starting Parameter Values (sr, nm, fr, ta)	Optimal Parameter Values (sr, nm, fr, ta)	Error Rate (percent)	Number of Function Evaluations	Timing (hours)
$(6, 25, 0.1, 80)$	$(6, 15, 0.079, 106)$	4.80	72	7.13
$(7, 10, 0.1, 180)$	$(6, 11, 0.083, 199)$	4.74	80	7.06
$(6, 30, 0.3, 60)$	$(6, 11, 0.147, 148)$	5.31	138	12.32
$(7, 15, 0.4, 120)$	$(8, 11, 0.098, 190)$	5.18	116	9.93
$(6, 35, 0.25, 120)$	$(6, 11, 0.246, 193)$	5.52	95	8.46
$(4, 25, 0.05, 140)$	$(4, 11, 0.138, 160)$	5.49	66	7.47

(c)

Note that the timings in (b) do not include the processing time for generating connected components. Since the generated connected components are the same for each algorithm run in the training procedure, they are generated only once before starting the training procedure.



(a) (b) (c)

Fig. 6. Convergence curves corresponding to six randomly selected starting points in the training of (a) the X-Y cut algorithm, (b) the Docstrum algorithm, and (c) the Voronoi-based algorithm. Note that the scales for error rate in the three figures are not the same.

minima varies a lot. The algorithm only needs about 30 function evaluations to reach stable performance. The vertical cuts are generally longer than horizontal cuts. The vertical interzone gaps are generally wider than horizontal interzone gaps. There is relatively small variation in the horizontal interzone gaps.

- **Docstrum algorithm.** From Table 2b and Fig. 6b, we can make the following observations:

  1. The error rates for all starting points converge in the range of 5.00 percent to 6.30 percent
  2. The convergence rate before the first 70 function evaluations is much faster than that beyond 70 function evaluations.
  3. The parameters f_t, f_{pa}, and f_{pe} converge to very similar values from most starting points.
  4. There is a relatively large variation in the optimal values of parameter f_d.
  5. The number of function evaluations is generally larger than those for the X-Y cut and Voronoi algorithms.

  From the above observations, we can see that the Docstrum algorithm objective function has multiple local minima and the variation in the optimal error rates is the smallest among the three research algorithms. The performance of the algorithm stabilizes after about 70 function evaluations, which is much larger than those for the X-Y cut and Voronoi algorithms. The performance of the Docstrum algorithm is insensitive to large (> 5) values of parameter f_d, since for small f_d, more connected components are grouped into the sparse connected component group where the intercharacter and interline gap estimation is not accurate and, hence, more errors will occur. However, for the other three parameters, the fact that most of the optimal values are very close implies the objective function may have a single "valley" in the neighborhood of these parameter values.

- **The Area-Voronoi-Diagram-Based algorithm.** From Table 2c and Fig. 6c, we can make the following observations:

  1. The error rates for all starting points converge in the range of 4.74 percent to 5.52 percent.
  2. The convergence rate before the first 40 function evaluations is much faster than that beyond 40 function evaluations.
  3. The value of the parameter nm for most starting points converges to 11 pixels.

  From the above observations, we can see that the Voronoi algorithm objective function has multiple local minima. The variation in performances at these local minima is much smaller than that of the X-Y cut algorithm. The algorithm needs only about 40 function evaluations to reach a stable performance. The optimal algorithm performance is insensitive to the value of parameter fr. The fact that the optimal value of parameter ta is large implies that the text and nontext connected components are well-separated. The fact that the values of parameter fr are generally small indicates that we should choose a conservative (large) interline spacing threshold.

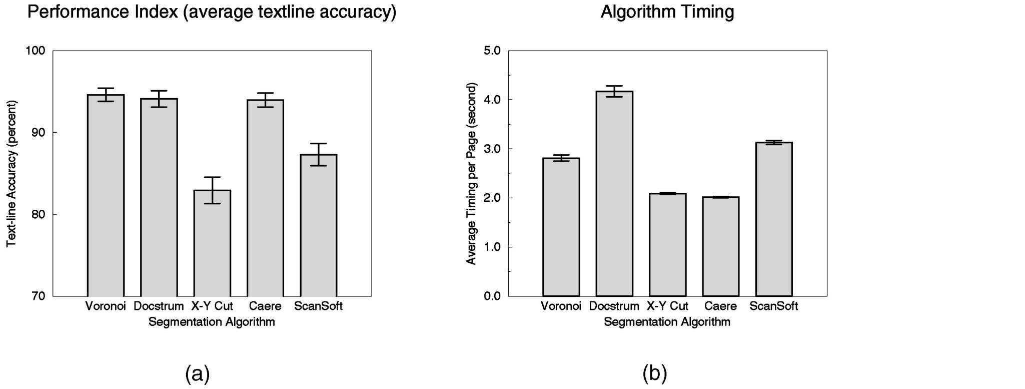
9.2 Testing Results

All five algorithms were tested on a 878-page test data set $\mathcal{S}$ with their respective optimum parameters. Table 3 reports the performance index (average textline accuracy) and

TABLE 3
Algorithm Testing Results and the Corresponding 95 Percent Confidence Intervals

Algorithm	Performance Index (percent)	Average Processing Time (seconds)	Operating System	Platform Type
Voronoi	94.60 ± 0.80	2.81 ± 0.06	Linux 7.0	Gateway PC, 400 MHz Pentium II CPU, main memory 128 MB.
Docstrum	94.09 ± 1.00	4.17 ± 0.11		
X-Y cut	82.92 ± 1.62	2.09 ± 0.02		
Caere	93.96 ± 0.85	2.01 ± 0.01	Windows 95	
ScanSoft	87.28 ± 1.36	3.13 ± 0.04		

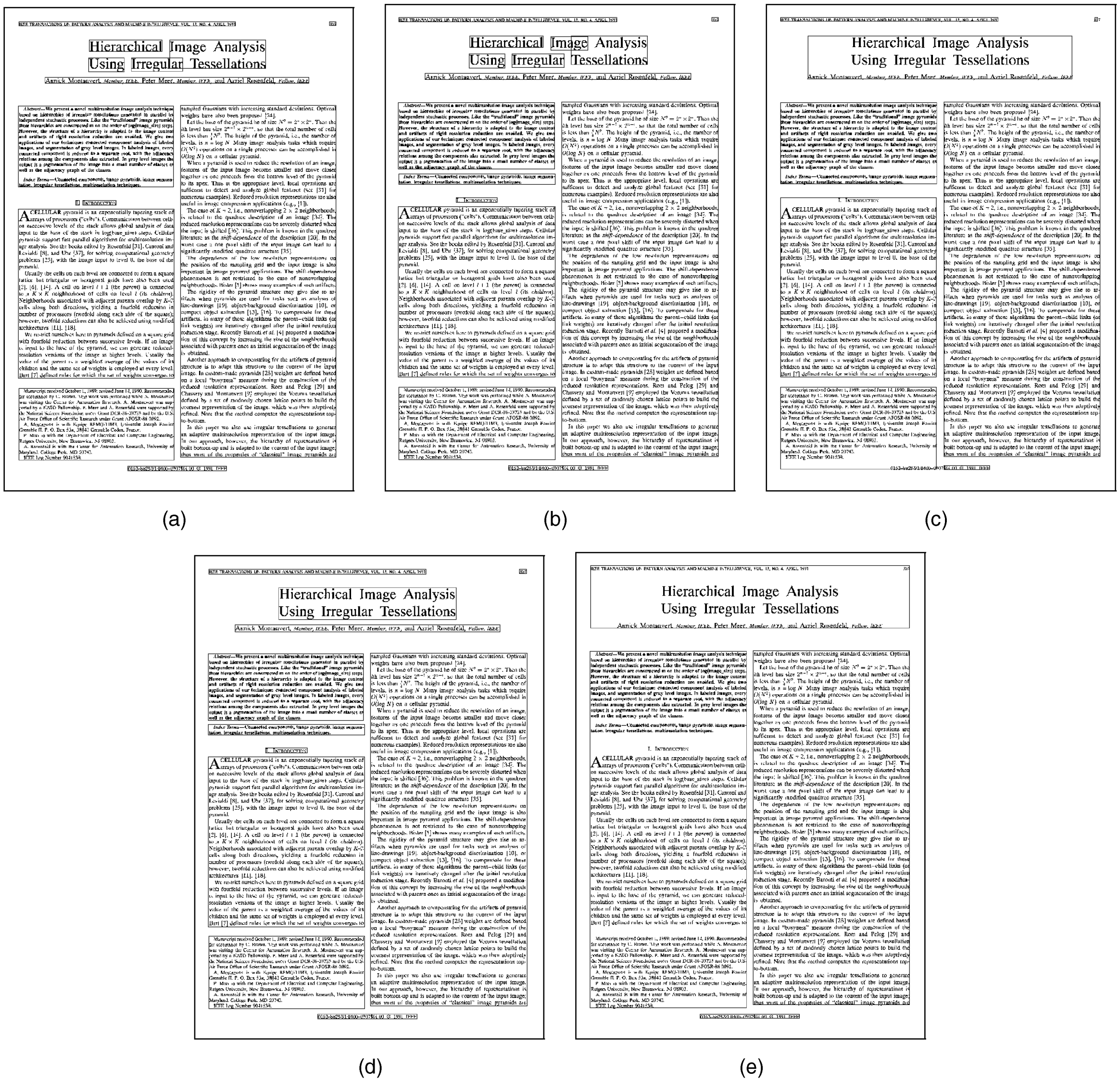
The average time per page, the platform type, and the operating system are also reported.



(a)

(b)

Fig. 7. The first three algorithms in the bar chart are reseach algorithms, and the last two algorithms are commercial products. (a) Shows the testing results of the performance index (average textline accuracy) for each algorithm. (b) Shows the algorithm testing time results for each algorithm.


Hierarchical Image Analysis
Using Irregular Tessellations
Annick Montanvert, Peter Meer, and Azriel Rosenfeld

(a)

(b)

(c)

(d)

(e)

Fig. 8. This figure shows the segmentation results of the Voronoi algorithm (a), the Docstrum algorithm (b), the X-Y cut algorithm (c), the Caere segmentation algorithm (d), and the ScanSoft segmentation algorithm (e) using their corresponding optimal parameters on a document page. Note that the Voronoi and the Docstrum algorithms split titles due to inaccurate parameter estimation, the X-Y cut algorithm splits page number and footer that are not aligned with the main text columns, and the ScanSoft segmentation algorithm merged the header and page number.

average algorithm timing on the test data set $\mathcal{S}$. Fig. 7 gives a bar-chart representation of the testing results for each evaluated algorithm.

From the testing results, we see that the Voronoi-based, Docstrum, and Caere algorithms have similar performance indices (average textline accuracy) which are better than

TABLE 4
Paired Model Statistical Analysis Results

	Caere	Docstrum	ScanSoft	X-Y Cut
Voronoi	0.64 ± 0.90 $P_{val} = 0.08$	0.51 ± 0.98 $P_{val} = 0.16$	7.31 ± 1.39 (*) $P_{val} = 4.75E - 24$	11.68 ± 1.63 (*) $P_{val} = 7.75E - 41$
Caere	–	-0.13 ± 1.10 $P_{val} = 0.41$	6.68 ± 1.38 (*) $P_{val} = 1.26E - 20$	11.04 ± 1.67 (*) $P_{val} = 1.21E - 35$
Docstrum	–	–	6.81 ± 1.59 (*) $P_{val} = 9.05E - 17$	11.17 ± 1.80 (*) $P_{val} = 4.64E - 32$
ScanSoft	–	–	–	4.37 ± 1.88 (*) $P_{val} = 2.79E - 06$

(a)

	Caere	Docstrum	ScanSoft	X-Y Cut
Voronoi	0.80 ± 0.05 (*) $P_{val} = 0$	-1.36 ± 0.06 (*) $P_{val} = 0$	-0.32 ± 0.07 (*) $P_{val} = 2.17E - 17$	0.72 ± 0.05 (*) $P_{val} = 0$
Caere	–	-2.16 ± 0.10 (*) $P_{val} = 0$	-1.12 ± 0.04 (*) $P_{val} = 0$	-0.07 ± 0.01 (*) $P_{val} = 1.49E - 31$
Docstrum	–	–	1.04 ± 0.12 (*) $P_{val} = 3.94E - 58$	2.08 ± 0.10 (*) $P_{val} = 0$
ScanSoft	–	–	–	1.04 ± 0.04 (*) $P_{val} = 0$

(b)

Paired model statistical analysis results and the corresponding 95 percent confidence intervals on the difference between a pair of performance indexes (in percent) (a) and the difference in processing time (seconds) (b). A (*) indicates that the difference is statistically significant at $\alpha = 0.05$, and no (*) indicates that the difference is not significant.

that of ScanSoft's algorithm, which in turn is better than that of the X-Y cut algorithm. From the fastest to the slowest, the algorithms are ranked as: Caere, X-Y cut, Voronoi, ScanSoft, and Docstrum. The connected component labeling method we used for Docstrum may not be the optimum one and, hence, its timing may be further improved. Fig. 8 shows the segmentation results of all algorithms on a document page. For comparison purposes, an evaluator always likes to know if the performance index (average textline accuracy) and processing time differences between algorithms are statistically significant or not, especially for those algorithms with similar performance index values. This is addressed in the following section.

9.3 Statistical Analysis of Results

We employed a paired model [16] to compare the performance index and testing time differences between each possible algorithm pair and then computed their confidence intervals. The analysis results for performance index (average textline accuracy) and processing time are reported in matrix form in Table 4. If we denote by T_{ij}, the value of the table cell in the $i\text{th}$ row and $j\text{th}$ column, $T_{ij} = a_i - a_j$, where a_i is the performance index or processing time of the algorithm in the $i\text{th}$ row, and a_j is the performance index or processing time of the algorithm in the $j\text{th}$ column. Algorithm timing performance depends on hardware and software factors, such as machine type, operating system, memory size, network protocol [11], etc. In our case, we assume that the timing difference contributed by different compiler and operating systems is negligible.

From Table 4, we find that a 5 percent level t-test indicates that the differences between the performance indices (average textline accuracy) of the Voronoi diagram-based algorithm, Caere's segmentation algorithm, and Docstrum are not statistically significant, but they are significantly better than those of ScanSoft's segmentation algorithm and the X-Y cut algorithm. Moreover, the performance index of ScanSoft's segmentation algorithm is significantly better than that of the X-Y cut algorithm. We find that the processing times of all algorithms differ significantly from one another. From the fastest processing time to the slowest processing time, the algorithms are ranked as Caere, X-Y cut, Voronoi, ScanSoft, and Docstrum.

9.4 Error Analysis

Error analysis is crucial to interpreting the functionalities of the evaluated algorithms. Each algorithm has different weaknesses. Fig. 9 shows the error analysis results of three error types for each algorithm.

We can see that among the research algorithms, X-Y cut has a much larger split textline error rate than the Voronoi and Docstrum algorithms. This is mainly due to the fact that the two zone cut thresholds (or widest zero valley thresholds) T_X^C and T_Y^C and the two noise removal thresholds T_X^n and T_Y^n are global thresholds that are *fixed* for each document image, whereas in the Voronoi and Docstrum algorithms, the intercharacter and interline spacings are estimated for each individual document image. Titles with wide intercharacter and interword spacings, numbered text lists, and textlines with irregular character spacings in some document images make the spacing parameter estimation inaccurate in both the Voronoi-based and Docstrum algorithms and, hence, contribute to the split textline error rates in these two algorithms. However, these split textline error rates are much smaller than that of X-Y cut. We can see that among the research algorithms, X-Y cut has the largest horizontally merged textline error rate, Docstrum has the second highest such error rate, and Voronoi has the lowest. This occurs primarily for the following reasons: 1) There are pages that have "L"-shaped thick, long noise blocks at the edges, which cannot be cut through in either the X or Y direction by the noise removal thresholds T_X^n and T_Y^n of the X-Y cut algorithm, so that many text regions under these noise blocks are merged together. 2) In Docstrum's implementation, the huge noise blocks encountered by the X-Y cut algorithm are filtered out in a preprocessing step, so that they do not affect connected component and textline clustering procedures. 3) In the Voronoi-based algorithm's implementation, in addition to what has been done for Docstrum, Kise et al. not only use the spacing of the connected components, but also their area ratios to generate zone boundaries. Hence, lines or large noise blocks between text regions do not cause horizontal merges, whereas they do cause horizontal merges in the Docstrum and the X-Y cut algorithm. We can see that among the research algorithms, the X-Y cut has the highest misdetected textline error rate while Voronoi and Docstrum have negligible such error rates. This is again due to the global thresholds of the X-Y cut algorithm which cause

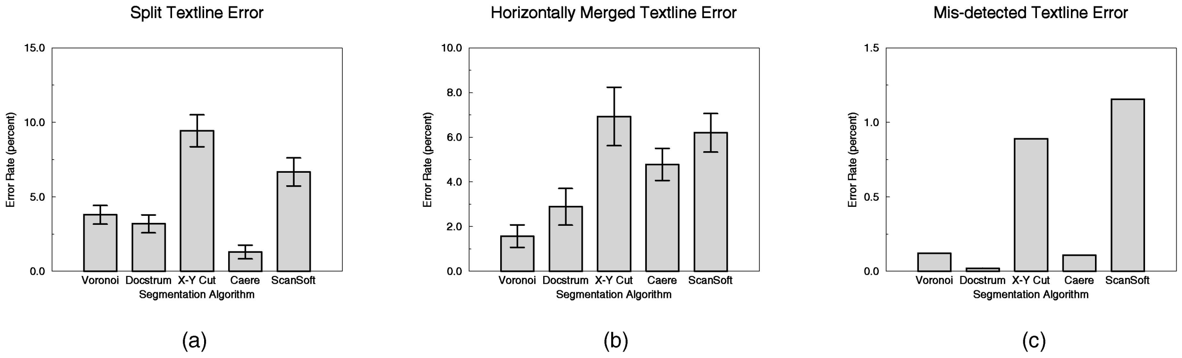


Fig. 9. This figure shows three types of errors. (a) Shows the page error rate as the ratio of the number of groundtruth textlines whose bound boxes are split and the total number of groundtruth textlines. We denote this error category as split textline error. A 5 percent level t-test indicates that the split textline error rates of all algorithms differ significantly from one another. (b) Shows the page error rate as the ratio of the number of groundtruth textlines that are horizontally merged and the total number of groundtruth textlines. We denote this error category as horizontally merged textline error. A 5 percent level t-test indicates that the horizontally merged textline error rates of X-Y cut and ScanSoft are not significantly different, but they are significantly higher than those of the other three algorithms. Moreover, the horizontally merged textline error rates of Voronoi, Docstrum, and Caere are significantly different from each other. (c) Shows the page error rate as the ratio of the number of groundtruth textlines that are missed and the total number of groundtruth textlines. We denote this error category as misdetected textline error. We can see that Caere has the smallest split textline error rate, Voronoi has the smallest horizontally merged textline error rate, and Docstrum has the smallest misdetected textline error rate. Note that the misdetected textline error rates are much smaller than the other two types of error rate for all algorithms and the scales for different type of error rate in the three figures are not the same.

textlines such as headers, footers, authors, or page numbers that are not aligned with text blocks to be considered as noise regions and, hence, not to be detected.

9.5 Recommendations

Based on the discussion in the last section, we feel that some recommendations may be useful to users who can make a choice among page segmentation algorithms. We summarize our recommendations about the three research algorithms as follows:

1. For segmentation of document pages with large skew angles or large noise blocks (especially "L"-shaped or "U"-shaped thick noise bars), the X-Y cut algorithm is a bad choice.
2. For segmentation of document pages with lines separating zones, the Voronoi-based algorithm is a better choice than either the Docstrum, or X-Y cut algorithm.
3. For an easy to implement algorithm that is also fast, the X-Y cut algorithm is a good choice.
4. For the X-Y cut algorithm, first remove large noise blocks by labeling connected components and then removing the larger ones.
5. If the given data set is similar to the data set used in this paper, users can choose segmentation algorithms according to their error characteristics shown in Fig. 9.
6. A fast connected component generation algorithm can make the Docstrum faster.

10 Conclusions

We have proposed a five-step performance evaluation methodology for evaluating page segmentation algorithms. We identify three crucial components of this methodology: automatic training posed as an optimization problem, paired model statistical analysis of experimental results, and error analysis of experimental results in terms of misdetection, split, and horizontal merge error types. We found that the performance indices (average textline accuracy) of the Voronoi, Docstrum, and Caere segmentation algorithms are not significantly different from one another, 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 also found that the timings of all algorithms are significantly different from one another. From the fastest to the slowest, the algorithms are ranked as Caere, X-Y cut, Voronoi, ScanSoft, and Docstrum. In the error analysis, we found that X-Y cut has the most split and horizontally merged textline errors due to its global thresholds, Voronoi has the least horizontally merged textline errors partly due to its usage of area ratio information of connected components, and Caere has the least split textline error. We intend to extend this work to evaluation of tables, graphs, and half-tone images.

Acknowledgments

The authors would like to thank Dr. Paul Smith for discussions on various statistical issues, Dr. Kise for providing us with a software implementation of his segmentation algorithm, Mindy Bokser for providing us with the Caere page segmentation software, Greg Marton for his help in obtaining experimental data, and Dr. Henry Baird, Dr. Azriel Rosenfeld, and Dr. Jeffrey K. Hollingsworth for their comments. This research was funded in part by the US Department of Defense and the Army Research Laboratory under contract MDA 9049-6C-1250, Lockheed Martin under Contract 9802167270, the Defense Advanced Research Projects Agency under contract N660010028910, and the US National Science Foundation under grant IIS9987944.

References

[1] H.S. Baird, S.E. Jones, and S.J. Fortune, "Image Segmentation by Shape-Directed Covers," *Proc. Int'l Conf. Pattern Recognition*, pp. 820-825, June 1990.

[2] *Empirical Evaluation Techniques in Computer Vision,* K.W. Bowyer and P.J. Phillips, eds., Santa Barbara, Calif. June 1998.

[3] *Document Layout Interpretation and Its Applications,* T. Breuel and M. Worring, eds., Bangalore, India, Sept. 1999.

[4] *Caere Developer's Kit 2000.* Caere Co. http://www.caere.com/ 1998.

[5] L.A. Fletcher and R. Kasturi, "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images" *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, pp. 910-918, 1988.

[6] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization,* chapter 4, London and New York: Academic Press, 1993.

[7] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Mass.: Addison-Wesley, 1989.

[8] *Performance versus Methodology in Computer Vision,* R.M. Haralick and P. Meer, eds., Seattle, June 1994.

[9] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision.* Reading, Mass.: Addison-Wesley, 1992

[10] R.M. Haralick, S.R. Sternberg, and X. Zhuang, "Image Analysis Using Mathematical Morphology," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 9, pp. 523-550, 1987.

[11] J.K. Hollingsworth, E. Guven, and C. Akinlar, "Benchmarking a Network of PCs Running Parallel Applications," *Proc. Int'l Performance, Computing, and Communications Conf.,* pp. 447-453, Feb. 1998.

[12] A. Hoover, G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D.B. Goldof, K.W. Bowyer, D.W. Eggert, A. Fitzgibbon, and R.B. Fisher, "An Experimental Comparison of Range Image Segmentation Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, pp. 673-689, 1996.

[13] A.K. Jain and B. Yu, "Document Representation and Its Application to Page Decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, pp. 294-308, 1998.

[14] J. Kanai, S.V. Rice, T.A. Nartker, and G. Nagy, "Automated Evaluation of OCR Zoning," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, pp. 86-90, 1995.

[15] T. Kanungo, M.Y. Jaisimha, J. Palmer, and R.M. Haralick, "A Methodology for Quantitative Performance Evaluation of Detection Algorithms," *IEEE Trans. Image Processing,* vol. 4, pp. 1667-1674, 1995.

[16] T. Kanungo, G.A. Marton, and O. Bulbul, "OmniPage vs. Sakhr: Paired Model Evaluation of Two Arabic OCR Products," *Proc. SPIE Conf. Document Recognition and Retrieval VI,* vol. 3651, pp. 109-120, Jan. 1999.

[17] K. Kise, A. Sato, and M. Iwata, "Segmentation of Page Images Using the Area Voronoi Diagram," *Computer Vision and Image Understanding,* vol. 70, pp. 370-382, 1998.

[18] V. Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications.* Dordrecht, The Netherlands: Reidel Publishing, 1989.

[19] R.M. Lewis, V. Torczon, and M.W. Trosset, "Why Pattern Search Works," *OPTIMA,* vol. 59, pp. 1-7, 1998.

[20] J. Liang, I.T. Phillips, and R.M. Haralick, "Performance Evaluation of Document Layout Analysis Algorithms on the UW Data Set," *Proc. SPIE Conf. Document Recognition IV,* vol. 3027, pp. 149-160, Feb. 1997.

[21] S. Mao and T. Kanungo, "A Methodology for Empirical Performance Evaluation of Page Segmentation Algorithms," Technical Report CAR-TR-933, Univ. of Maryland, College Park, Dec. 1999. http://www.cfar.umd.edu/~kangungo/pubs/trsegeval.ps.

[22] S. Mao and T. Kanungo, "Automatic Training of Page Segmentation Algorithms: An Optimization Approach," *Proc. Int'l Conf. Pattern Recognition,* pp. 531-534, Sept. 2000.

[23] S. Mao and T. Kanungo, "Empirical Performance Evaluation of Page Segmentation Algorithms," *Proc. SPIE Conf. Document Recognition and Retrieval VII,* vol. 3967, pp. 303-314, Jan. 2000.

[24] S. Mao and T. Kanungo, "PSET: A Page Segmentation Evaluation Toolkit," *Proc. Fourth IAPR Int'l Workshop Document Analysis Systems,* pp. 451-462, Dec. 2000.

[25] S. Mao and T. Kanungo, "Software Architecture of PSET: A Page Segmentation Evaluation Toolkit," Technical Report CAR-TR-955, Univ. of Maryland, College Park, Sept. 2000. http://www.cfar.umd.edu/~kanungo/pubs/trpset.ps. Software is available at http://www.cfar.umd.edu/~kanungo/software/software.html.

[26] G. Nagy, S. Seth, and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," *Computer,* vol. 25, pp. 10-22, 1992.

[27] J. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer J.,* vol. 7, pp. 308-313, 1965.

[28] L. O'Gorman, "The Document Spectrum for Page Layout Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, pp. 1162-1173, 1993.

[29] L. O'Gorman and R. Kasturi, *Document Image Analysis.* Los Alamitos, Calif.: IEEE CS Press, 1995.

[30] T. Pavlidis and J. Zhou, "Page Segmentation and Classification," *Graphical Models and Image Processing,* vol. 54, pp. 484-496, 1992.

[31] I. Phillips, *User's Reference Manual,* CD-ROM, UW-III Document Image Database-III, July 1996.

[32] I.T. Phillips and A.K. Chhabra, "Empirical Performance Evaluation of Graphics Recognition Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, pp. 849-870, 1999.

[33] P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, pp. 1090-1104, 2000.

[34] M.J.D. Powell, "Direct Search Algorithms for Optimization Calculations," *Acta Numerica,* vol. 7, pp. 287-336, 1998.

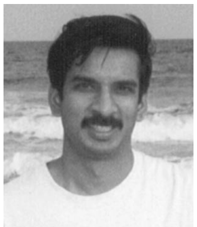
[35] S. Randriamasy, L. Vincent, and B. Wittner, "An Automatic Benchmarking Scheme for Page Segmentation," *Proc. SPIE Conf. Document Recognition,* vol. 2181, pp. 217-230, Feb. 1994.

[36] *Application Programmer's Interface,* ScanSoft Co., Dec. 1997. http://www.scansoft.com.

[37] F. Wahl, K. Wong, and R. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," *Graphical Models and Image Processing,* vol. 20, pp. 375-390, 1982.

[38] M.H. Wright, "Direct Search Methods: Once Scorned, Now Respectable," *Numerical Analysis 1995,* pp. 191-208, D.F. Griffiths and G.A. Watson, eds., Addison Wesley, Longman (Harlow), 1996.

Song Mao received the BEng and MEng degrees from the Department of Precision Instrument Engineering, Tianjin University, Tianjin, China in 1993 and 1996, respectively, and the MS degree in electrical and computer engineering from the University of Maryland at College Park in 1999. He is currently pursuing a PhD degree in electrical and computer engineering at the University of Maryland at College Park. His research interests include document image analysis, pattern recognition, and computer vision.



Tapas Kanungo received the MS and PhD degrees in electrical engineering from the University of Washington, Seattle, in 1990 and 1996, respectively. From March 1996 to October 1997, he worked at Caere Corporation, Los Gatos, California, on their OmniPage OCR product. During the summer of 1994, he worked at Bell Labs, Murray Hill, New Jersey, and during the summer of 1993, he worked at the IBM Almaden Research Center, San Jose, California. Prior to that, from 1986 to 1988, he worked on speech coding and online handwriting analysis in the Computer Science group at Tata Institute for Fundamental Research, Bombay, India. He currently serves as a codirector of the Language and Media Processing Lab at the University of Maryland, College Park, where he conducts research in the areas of document image analysis, OCR-based cross-language information retrieval, pattern recognition, and computer vision. He cochaired the 1999 IAPR Workshop on Multilingual OCR, was a coguest editor of the *International Journal of Documnent Analysis and Recognition* special issue on performance evaluation, and has been program committee member of several conferences. He is a member of the IEEE.