LAMP-TR-37
CAR-TR-933
CS-TR-4093

December 1999

A Methodology for Empirical Performance Evaluation of Page Segmentation Algorithms

Song Mao and Tapas Kanungo

A Methodology for 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

Abstract

Document page segmentation is a crucial preprocessing step in Optical Character Recognition (OCR) systems. While numerous page segmentation algorithms have been proposed, there is relatively less literature on *comparative* evaluation — empirical or theoretical — of these algorithms. For 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 thesis, 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 datasets 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, 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. The automatic training of algorithms is posed as an optimization problem and a direct search method — the simplex method — is used to search for a set of optimal parameter values. A paired-model statistical analysis and an error analysis are conducted to provide confidence intervals for the experimental results and to interpret the functionalities of 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 dataset. It is found that the performances 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.

This research was funded in part by the Department of Defense and the Army Research Laboratory under Contract MDA 9049-6C-1250.

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, books, newspapers, magazines, microfilms, personal notes, etc. They can be digitally created, faxed or scanned document images. The format of a document image can be handwritten or machine printed. A document can contain text, tables, figures and halftone images. The output symbolic text file from an OCR system can include only the text content of the input document image, or it can also include 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.

This report 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 thesis in Document Recognition and Retrieval VII [23].

2 Literature Survey

Page segmentation algorithms can be categorized into three classes: top-down approaches, bottom-up approaches and hybrid approaches. The Docstrum algorithm of O'Gorman [28], the Voronoi-diagram-based algorithm of Kise [19], the run-length smearing algorithm of Wahl, Wong and Casey [43], the segmentation algorithm of Jain and Yu [15], and the text string separation algorithm of Fletcher and Kasturi [7] are typical bottom-up algorithms, while the X-Y cut by Nagy [25, 26] and the shape-directed-covers-based algorithm by Baird [2, 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 [15]. A recent workshop [5] 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.* [16] 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 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. Rice, Jenkins and Nartker [38] used this performance metric in their comparative evaluation of the automatic zoning accuracy of four commercial OCR products. Vincent *et al.* [36, 37, 45] proposed various bitmap-level region-based metrics. The advantages of the Vincent *et al.* approach 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 [22] describe a region-area-based metric. The overlap area of a groundtruth zone and a segmentation zone is used to compute this performance metric. Many OCR performance evaluation case studies are discussed in [14].

In the general computer vision area, numerous researchers have presented methods for empirical performance evaluation. For example, Hoover *et al.* [13] 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.* [17] 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 automatic training of algorithms with free parameters and statistical and error 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 [3, 10].

In research segmentation algorithms that have user-specifiable parameters, typically the default parameter values are selected and no training method is explicitly specified [19, 2, 28, 15, 30, 7]. 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 [13, 17, 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 dataset. To objectively optimize a segmentation algorithm on a given training dataset, 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, Murray and Wright [8]. In our case, since the objective function corresponding to a performance metric for page segmentation algorithms is not rigorously defined mathematically, automatic training is posed as a *multivariate non-smooth nonlinear function* optimization problem. Direct search algorithms are typically used for solving optimization problems involving this kind of objective function. Powell [34] gives a detailed survey of direct search algorithms. Line search methods, discrete grid methods, simplex methods, conjugate direction methods, linear approximation methods, and quadratic approximation methods are designed to converge to a local minimum of the objective function. Additional survey literature regarding direct search algorithms can be found in [21, 44]. Many practical optimization calculations have many local minima that are not optimal. "Simulated annealing" [20] and "genetic" [9] algorithms are proposed to search for a global minimum by selecting vectors of variables using random number generators. We chose the simplex search method proposed by Nelder and Mead [27] since it is recognized as one of the most reliable and efficient methods [4, 41] for optimizing an objective function for which derivatives are not available.

3 The Page Segmentation Problem and Error Metrics

In this section, we give the definition of page segmentation. In order to evaluate the performance of page segmentation algorithms, a set of error measurements and metrics are needed. We provide the definitions of our proposed textline based error measures and metric. These definitions are based on set theory and mathematical morphology [42, 24, 12].

3.1 Page Segmentation Definition

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 be the segmentation result of algorithm A such that $R = Seg_A(I, \mathbf{p}^A)$ where $Z(R) = \{Z_k^R | k = 1, 2, \ldots, \#Z(R)\}$.

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. Some groundtruth generation methods provide only zone-level groundtruth. In our evaluation methodology, we need both zone-level and textline-level groundtruth.

3.2 Error Measurements and Metric Definitions

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. 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 rectangular region centered at $(0,0)$ with a width of $2T_X$ and a height of $2T_Y$ 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 [42, 24, 12]. Let $A, B \subseteq Z^2$. Morphological *dilation* of A by B is denoted by $A \oplus B$ and is defined as

$$A \oplus B = \left\{ c \in Z^2 | c = a + b \text{ for some } a \in A, b \in B \right\}.$$

Morphological *erosion* of A by B is denoted by $A \ominus B$ and is defined as

$$A \ominus B = \left\{ c \in Z^2 | c + b \in A \text{ for every } b \in B \right\}.$$

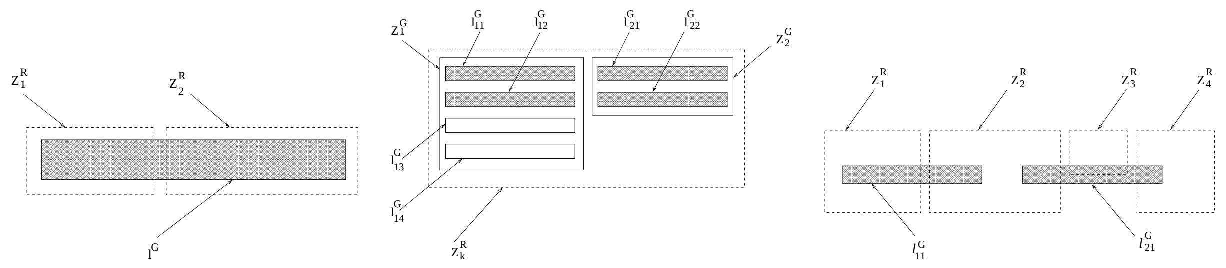
We first define correctly detected groundtruth textlines, and then define four types of textline-based error measurements.

1. Groundtruth textlines that are correctly detected:
   $D_L = \left\{ l^G \in \mathcal{L} | D(l^G) \ominus E(T_X, T_Y) \subseteq D(Z^R) \text{ for some } Z^R \in Z(R) \right\}$,
2. Groundtruth textlines that are missed:
   $C_L = \left\{ l^G \in \mathcal{L} | D(l^G) \ominus E(T_X, T_Y) \subseteq (\cup_{Z^R \in Z(R)} D(Z^R))^c \right\}$,
3. Groundtruth textlines whose bounding boxes are split:
   $$S_L = \left\{ l^G \in \mathcal{L} | (D(l^G) \ominus E(T_X, T_Y)) \cap D(Z^R) \neq \phi, \right.$$
   $$\left. (D(l^G) \ominus E(T_X, T_Y)) \cap (D(Z^R))^c \neq \phi, \text{ for some } Z^R \in Z(R) \right\}$$
4. Groundtruth textlines that are horizontally merged:
   $$M_L = \left\{ l^G_{qj} \in \mathcal{L} | \exists l^G_{q'j'} \in \mathcal{L}, Z^R \in Z(R), q \neq q', Z^G_q, Z^G_{q'} \in Z(G) \right.$$
   $$(D(l^G_{qj}) \ominus E(T_X, T_Y)) \cap D(Z_k) \neq \phi,\ (D(l^G_{q'j'}) \ominus E(T_X, T_Y)) \cap D(Z_k) \neq \phi,$$
   $$((D(l^G_{qj}) \ominus E(0, T_Y)) \oplus E(\infty, 0)) \cap D(Z^G_{q'}) \neq \phi,$$
   $$\left. ((D(l^G_{q'j'}) \ominus E(0, T_Y)) \oplus E(\infty, 0)) \cap D(Z^G_q) \neq \phi \right\}.$$
5. Noise zones that are falsely detected (false alarms):
   $F_L = \left\{ Z^R \in Z(R) | D(Z^R) \subseteq (\cup_{l^G \in \mathcal{L}} (D(l^G) \ominus E(T_x, T_Y)))^c \right\}$

Figure 1 shows an example of errors in groundtruth textlines.

Let the number of groundtruth error textlines be $\#\{C_L \cup S_L \cup M_L\}$ (mis-detected, split or horizontally merged) , and let the total number of groundtruth textlines be $\#\mathcal{L}$. We define the performance metric $\rho(I, G, R)$ as textline accuracy:

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



(a) (b) (c)

Figure 1: This figure shows examples of textline errors. (a) shows a groundtruth textline l^G split into two segmentation zones Z_1^R and Z_2^R. (b) shows two groundtruth zones Z_1^G and Z_2^G horizontally merged into segmentation zone Z^R. Only the dark groundtruth textlines $l_{11}^G, l_{12}^G, l_{21}^G, l_{22}^G$ are considered horizontally merged since they are the only textlines that are impacted by the horizontal merge. (c) shows two textlines l_{11}^G and l_{12}^G on which multiple errors happen. l_{11}^G is split by Z_1^R and merged by Z_2^R, l_{11}^G is split by Z_3^R and Z_4^R and merged by Z_2^R. In our metric we count two instance of textline error.

In general, the performance metric $\rho(I, G, R)$ can be any function of $\mathcal{L}, D_L, C_L, S_L, M_L$ and F_L. Figure 2 gives a set of possible errors as well as an experimental example.

We consider three types of textline errors — split, missed and horizontally merged. 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, requires textline-level groundtruth.

4 Performance Evaluation Methodology

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

A large and representative dataset is desirable in any performance evaluation task in order to give objective 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 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 may be a non-optimal parameter set for another dataset. Hence the choice of parameters is crucial in any performance evaluation task. When the size of the dataset gets very large, parameter set training on the whole dataset becomes computationally prohibitive and therefore a representative sample dataset of much smaller size must be used as a training dataset. After the training step, the page segmentation algorithms with the optimal parameters should be evaluated

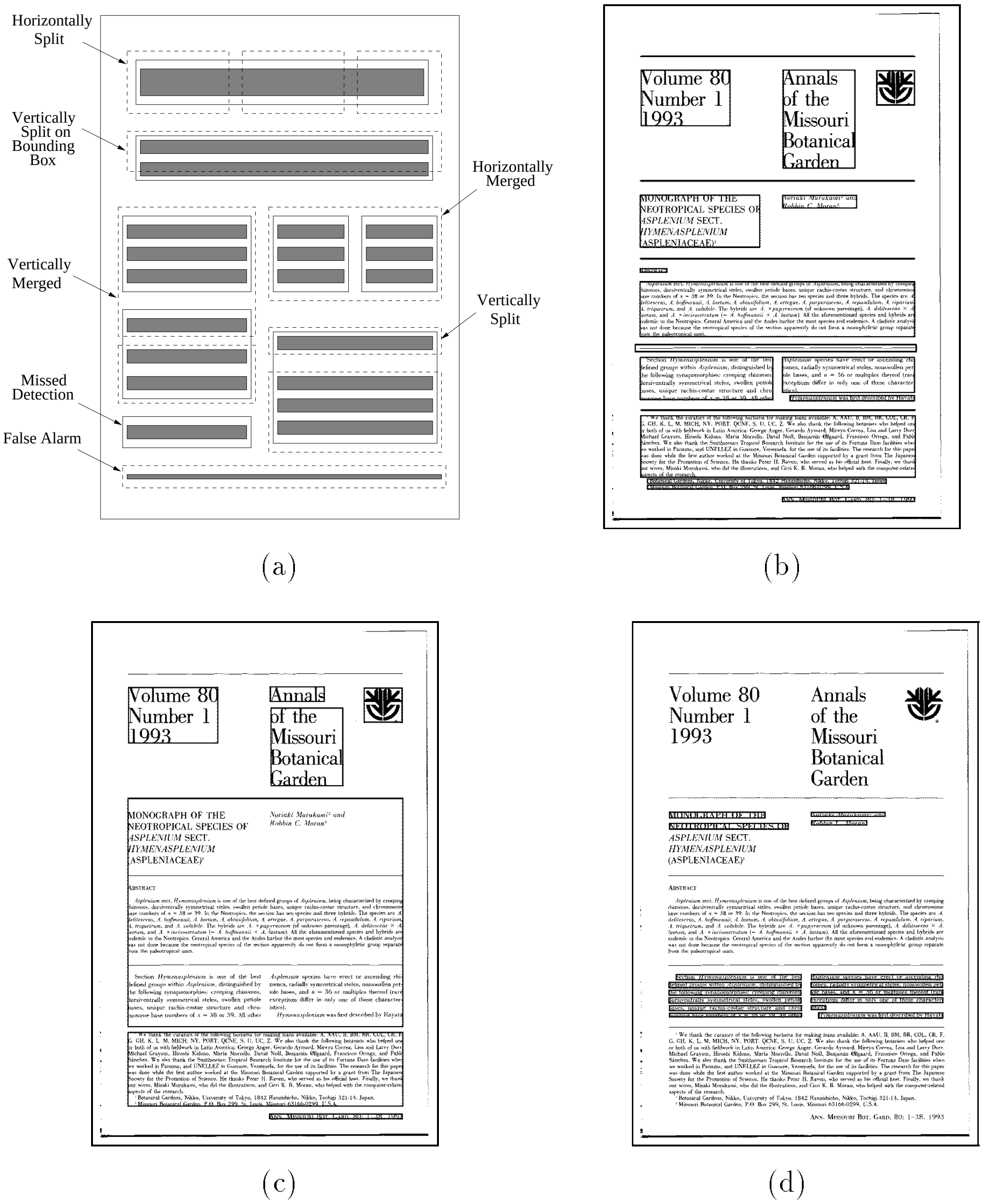


Figure 2: (a) This figure shows a set of possible textline errors. Solid line rectangles denote groundtruth zones, dashed-line rectangles denote OCR segmentation zones, dark bars within groundtruth zones denote groundtruth textlines, and dark bars outside solid lines are noise blocks. (b) This figure shows a document page image from the University of Washington III dataset with the groundtruth zones overlaid. (c) This figure shows an OCR experimental segmentation result on this document page image. (d) This figure shows 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.

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. Let $\mathcal{D}$ be a given dataset 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 dataset $\mathcal{D}$ into a mutually exclusive training dataset $\mathcal{T}$ and test dataset $\mathcal{S}$. Thus $\mathcal{D} = \mathcal{T} \cup \mathcal{S}$ and $\mathcal{T} \cap \mathcal{S} = \phi$, where ϕ is the empty dataset.

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 dataset $\mathcal{T}$.

4. Evaluate the segmentation algorithm A with optimized parameters $\hat{\mathbf{p}}^A$ on the test dataset $\mathcal{S}$ by $\Phi\left(\{\rho(G, Seg_A(I, \hat{\mathbf{p}}^A)) | (I, G) \in \mathcal{S}\}\right)$ where Φ is a function of the performance metric ρ on each document image and groundtruth pair (I, G) in the test dataset $\mathcal{S}$, and $Seg_A(\cdot, \cdot)$ is the segmentation function corresponding to A. The function Φ is defined by the user. In our case, Φ is defined as the average of the performance metric $\rho(G, Seg_A(I, \hat{\mathbf{p}}^A))$ on each document image and groundtruth pair (I, G) in the test dataset $\mathcal{S}$.

5. Perform a statistical analysis to find the significance of the evaluation results and 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.

This methodology is similar to typical methodologies used in pattern recognition. In pattern recognition, problems are usually well-defined mathematically and hence a better training strategy and optimization method can be used than in our case, where page segmentation algorithms are not rigorously defined mathematically. In the computer vision and image processing literatures, Kanungo *et al.* [17] conducted a quantitative performance evaluation of two detection algorithms. Hoover *et al.* [13] quantitatively compared four range image algorithms. In both of these papers, while a detailed methodology and experimental framework were carefully designed, two important components, automatic algorithm training and statistical analysis of the experimental results, were missing.

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. In this section, we first define the objective function in our performance evaluation of page segmentation algorithms, then we introduce a direct search algorithm to optimize the defined objective function, and finally we discuss starting point selection in our optimization problem.

5.1 The Objective Function

In this subsection, we identify the objective function. Let $\mathbf{p}^A$ be the parameter vector for the segmentation algorithm A, let $\mathcal{T}$ be a training dataset, and let $\rho(I, G, Seg_A(I, \mathbf{p}^A))$ where $(I, G) \in \mathcal{T}$ is a performance metric. 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 dataset:

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

where ρ is defined in Equation (1).

This objective function has the following properties:

- It is dependent on the values of the algorithm parameters,
- The function value is the only information available,
- The function has no explicit mathematical form and is non-differentiable,
- Obtaining a function value requires nontrivial computation.

This objective function can be classified as a *multivariate non-smooth function* [8]. 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 4.1. We choose the simplex search method proposed by Nelder and Mead [27] to minimize our objective function.

We give the notation used to describe the simplex method: Let $\mathbf{q}_0$ and $\lambda_i, i = 1, \ldots, n$ be a starting point and 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}. Figure 5.2 shows the various simplex operations.

For a segmentation algorithm with n parameters, the Nelder-Mead algorithm works as follows:

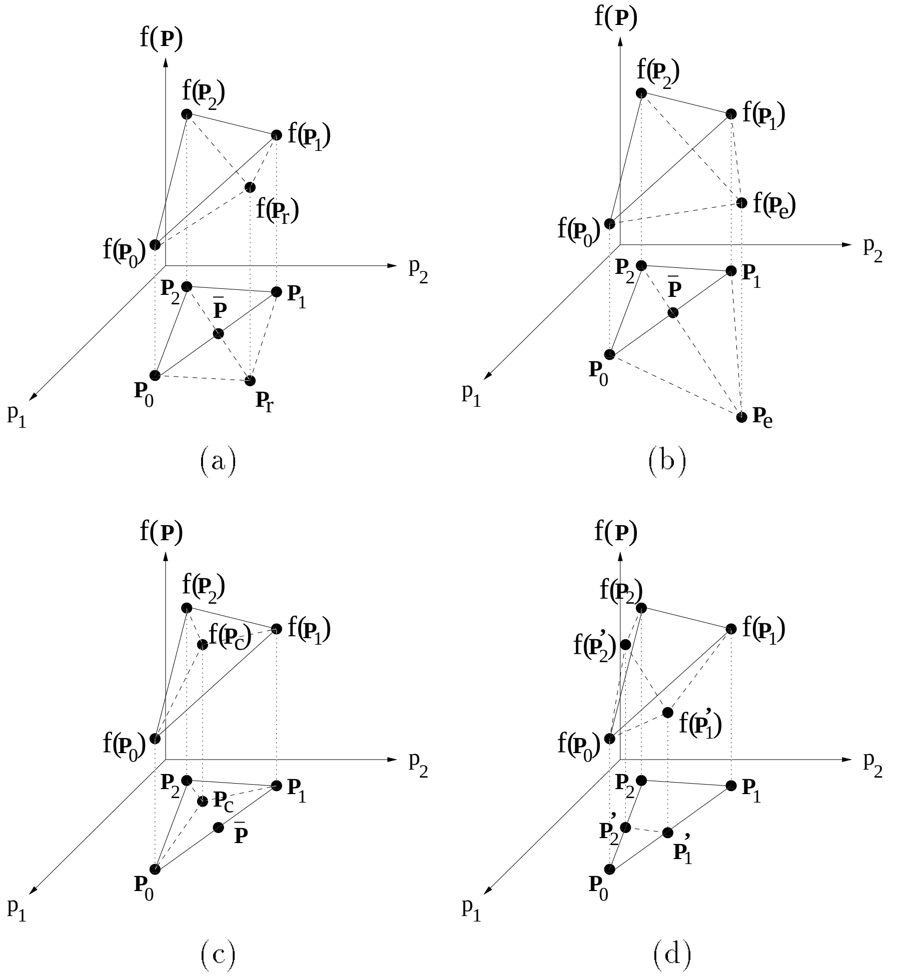


Figure 3: This figure shows four simplex operations in a two-dimensional parameter space. The solid lines denote the simplex before any operation and the dashed lines denote the simplex after the operation. $\mathbf{p}_2$ and $\mathbf{p}_0$ are the vertices for which the objective funtion $f(\cdot)$ assumes the biggest and smallest values respectively, and $\bar{\mathbf{p}} = \sum_{i=0}^{1} \mathbf{p}_i/2$ is the centroid of the two best vertices. The operations are (a) a reflection $\mathbf{p}_r$ of $\mathbf{p}_2$ with respect to the centroid point $\bar{\mathbf{p}}$, (b) an expansion $\mathbf{p}_e$ of $\mathbf{p}_2$ with respect to the centroid point $\bar{\mathbf{p}}$, (c) a contraction $\mathbf{p}_c$ of $\mathbf{p}_2$ with respect to the centroid point $\bar{\mathbf{p}}$, and (d) a shrinkage of all $\mathbf{p}_i, i \neq 0$ toward $\mathbf{p}_0$. A local minimum can be obtained after an appropriate sequence of such operations.

1. Given $\mathbf{q}_0$ and the λ_i, form the initial simplex $\tilde{\mathbf{q}}_0 = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n\}$ $\mathbf{q}_i = \mathbf{q}_0 + \lambda_i \mathbf{e}_i, i = 1, \ldots, n.$,

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. If $f(\mathbf{p}_r) \leq f(\mathbf{p}_0)$, replace $\mathbf{p}_n$ by $\mathbf{p}_r$, 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 $\gamma = [\mathbf{p}_e \bar{\mathbf{p}}]/[\mathbf{p}_n \bar{\mathbf{p}}] > 1$.

5. 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 $\mathbf{p}_r$, get a contraction point $\mathbf{p}_c$ of $\mathbf{p}_n$ by $\mathbf{p}_c = (1-\beta)\bar{\mathbf{p}} + \beta \mathbf{p}_n$ where $\beta = [\mathbf{p}_c \bar{\mathbf{p}}]/[\mathbf{p}_n \bar{\mathbf{p}}] < 1$. If $f(\mathbf{p}_c) \geq f(\mathbf{p}_n)$, shrink the simplex around the best vertices $\mathbf{p}_0$ by $\mathbf{p}_i = (\mathbf{p}_i + \mathbf{p}_0)\sigma, i \neq 0$, else replace $\mathbf{p}_n$ by $\mathbf{p}_c$.

6. Else replace $\mathbf{p}_n$ by $\mathbf{p}_r$.

7. If $\sqrt{\sum_{i=0}^{n} (f(\mathbf{p}_i) - f(\bar{\mathbf{p}}))^2/n} < T$, stop.

8. Else go back to step 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? 3) Are the algorithms in one class performing significantly better than those in another class? 4) What are the sources of performance metric variance? To answer such questions, a statistical model needs to be constructed for the experimental observations.

In this section, we describe a paired model analysis approach proposed by Kanungo *et al.* [18] for their evaluation of Arabic OCR engines and adapt it to analyze our experimental results. We use the paired model to model our experimental observations and to provide underlying theory for creating confidence intervals and testing hypotheses.

6.1 Modeling Experimental Data Using the Paired Model

In this section, we set up the notation and fit the paired model to our experimental observation data. Let $A_1, A_2 \ldots, A_k$ denote the k algorithms we evaluate, and let $X_{ij}, i = 1, \ldots, k, \quad j = 1, \ldots, n$ be the observation (textline accuracy) corresponding to algorithm A_i and document image I_j in test dataset $\mathcal{S}$. In our experiment, the number of algorithms k is 5 and the total number of images n in test dataset $\mathcal{S}$ 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. Let $cov(X_{ij}, X_{i'j}) = \rho_{ii'}\sigma_i\sigma_{i'}$ where $i \neq i'$ is the covariance of observation $X_{ij}, X_{i'j}$.

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

$$E[\hat{\Delta}_{ii'}] = E[\bar{W}_{ii'}] = E[\bar{X}_i - \bar{X}_{i'}] = \mu_i - \mu_{i'} = \Delta_{ii'}. \tag{3}$$

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

$$Var[\hat{\Delta}_{ii'}] = Var[\bar{W}_{ii'}] = Var[\bar{X}_i - \bar{X}_{i'}] = \frac{\sigma_i^2 + \sigma_{i'}^2 - 2\rho_{ii'}\sigma_i\sigma_{i'}}{n}. \tag{4}$$

6.2 Confidence Intervals and Hypothesis Testing

In this section we address two important issues: i) How does one characterize uncertainty of the performance metric estimates? and ii) If the average textline accuracy of one algorithm is better than another, how do we verify that the result is statistically significant and not just due to chance? Let us 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

$$\lim_{n\to\infty} \frac{\hat{\Delta}_{ii'} - \Delta_{ii'}}{\sigma_{ii'}/\sqrt{n}} = \lim_{n\to\infty} \frac{\bar{W}_{ii'} - (\mu_i - \mu_{i'})}{\sigma_{ii'}/\sqrt{n}} \sim N(0,1), \tag{5}$$

where $\sigma_{ii'}$ is the true standard deviation of $\hat{\Delta}_{ii'}$. For large samples, it is sufficient for us to assume that

$$\frac{\hat{\Delta}_{ii'} - \Delta_{ii'}}{\sigma_{ii'}/\sqrt{n}} = \frac{\bar{W}_{ii'} - (\mu_i - \mu_{i'})}{\sigma_{ii'}/\sqrt{n}} \sim N(0,1). \tag{6}$$

When $\sigma_{ii'}$ is not available as it is in our case, the sample standard deviation $V_{ii'}$ is typically used in place of $\sigma_{ii'}$ on the left side of Equation (6). The new formula has an approximate t distribution with $n-1$ degrees of freedom, i.e.,

$$\frac{\hat{\Delta}_{ii'} - \Delta_{ii'}}{V_{ii'}/\sqrt{n}} = \frac{\bar{W}_{ii'} - (\mu_i - \mu_{i'})}{V_{ii'}/\sqrt{n}} \sim t_{n-1}. \tag{7}$$

Thus, for a given significance level α, we can compute a confidence interval as

$$\Delta_{ii'} \in \hat{\Delta}_{ii'} \pm \frac{t_{\alpha/2,n-1}V_{ii'}}{\sqrt{n}}. \tag{8}$$

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:

- Null hypothesis:
$$H_0 : \Delta_{ii'} = \mu_i - \mu_{i'} = 0,$$
- Alternative hypothesis:
$$H_a : \Delta_{ii'} = \mu_i - \mu_{i'} \neq 0,$$
- Test statistic:
$$T = T(X_{i1}, \ldots, X_{in}, X_{i'1}, \ldots, X_{i'n}) = (\hat{\Delta}_{ii'} - 0)/(V_{ii'}/\sqrt{n}) \tag{9}$$
- Approximate distribution of the test statistic T under the null hypothesis H_0:
$$t \text{ distribution with } n-1 \text{ degrees of freedom}$$
- Rejection region significance level α test:
$$\text{Define} \quad P_{val} = \int_{-\infty}^{-T} f(t)dt + \int_{T}^{\infty} f(t)dt \tag{10}$$
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 the correlation of observations on the same image is considered, a 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 $\Delta_{ii'}$. This variance is smaller than that 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 the later case, since two samples are independent, the variance of the estimator $\hat{\Delta}_{ii'}$ is given by

$$Var[\hat{\Delta}_{ii'}] = \frac{\sigma_i^2 + \sigma_{i'}^2}{n}. \tag{11}$$

Since $\rho_{ii'} > 0$, it can be easily seen that

$$\frac{\sigma_i^2 + \sigma_{i'}^2}{n} > \frac{\sigma_i^2 + \sigma_{i'}^2 - 2\rho_{ii'}\sigma_i\sigma_{i'}}{n}. \tag{12}$$

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. 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. If there are word clustering or line clustering procedures, the final zone segmentations are obtained by clustering the words or lines. The commercial products usually are "black-box" algorithms from which no algorithm structure information can be inferred.

7.1 The X-Y Cut Page Segmentation Algorithm

The X-Y cut segmentation algorithm [25, 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 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:
   $H_X[i][j] = \#\{p \in D(I) | X(p) = j, Y(p) \leq i, I(p) = 1\}$,
   $H_Y[i][j] = \#\{p \in D(I) | X(p) \leq j, Y(p) = i, I(p) = 1\}$,
   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 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 $V_X > T_X$ 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 and generate two child nodes Otherwise, make the current node a leaf node.

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. However, this algorithm only applies to the segmentation of text regions. Moreover, it does not perform well when the document page image contains too many joined characters and the estimates of inter-character spacing, inter-line spacing and orientation angle become inaccurate when document images contain sparse characters.

The basic steps of the Docstrum segmentation algorithm are as follows:

1. Obtain connected components (C_is) using a space-efficient two-pass algorithm [11].

2. Remove small and large noise or non-text 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.

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 a within-line nearest-neighbor distance histogram from the following set W_ρ : $W_\rho = \{\rho_j^i | j \in \mathrm{NN}_K(i), \text{ and } -\theta_h \leq \theta_j^i \leq \theta_h\}$, 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_ρ : $B_\rho = \{\rho_j^i | j \in \mathrm{NN}_K(i), \text{ and } 90° - \theta_v \leq \theta_j^i \leq 90° + \theta_v\}$, 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.

In our implementation, we did not estimate orientation since all pages in the dataset 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

Kise's segmentation algorithm [19] 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 non-linear textlines. A set of connected line segments are used to bound text zones. Since we evaluate all algorithms on document page images 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 sizes and spacings. It assumes that text regions are dominant on a page and that the inter-character and inter-line spacing within a text region are 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 borders 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 a Voronoi diagram. The Voronoi diagram for each connected component is generated using the sample points on its border.

4. Delete superfluous Voronoi edges. These 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 let $d(E)$ be the minimum distance between any two sample points from C_i and C_j. Let T_{d1} be the estimate of the inter-character spacing, and let 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 $\max\{\mathrm{Area}(C_i), \mathrm{Area}(C_j)\}/\min\{\mathrm{Area}(C_i), \mathrm{Ar}$ and the threshold T_a as 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 criterion, 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) C_i and C_j come from different text zones that have larger spacing than inter-line spacing plus a margin, or if 2) 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.

   A C implementation of this algorithm was provided to us by Professor Koichi Kise.

7.4 Commercial Segmentation Algorithms

Two commercial products, Caere's segmentation algorithm [6] and ScanSoft's segmentation algorithm [39, 40], were selected for evaluation. They are representative state-of-art commercial products. Both are black-box algorithms with no free parameters.

8 Experimental Protocol

In this section we provide the details of our experimental setup so that other researchers can replicate our experiments. 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. We then computed an average performance metric value over all document pages in the training dataset $\mathcal{T}$ or test dataset $\mathcal{S}$ and report it as the final algorithm performance index.

In Section 8.1, we specify the dataset set used in our experiments. In Section 8.2, we describe the training procedure and specify the parameters for each research algorithm. In Section 8.3, we briefly describe the testing procedure. In Section 8.4, we give the details of the hardware and software environments.

8.1 Dataset Specification

We selected the University of Washington Dataset [31] for the performance evaluation task since it is the only dataset that has textline-level groundtruth for each document page. All pages in the dataset are journal pages from a large variety of journals in diverse subject areas and from different publishers. The dataset also has geometric textline and zone groundtruth for each page. The textline and zone groundtruth are represented by non-overlapping rectangles. The University of Washington III dataset has 1601 deskewed binary document images at 300 dpi resolution. We chose a subset of 978 pages that correspond to the University of Washington I dataset pages as our experimental dataset. We evaluated the chosen algorithms only on text regions since they carry the most information about a document image. The non-text regions were ignored in the evaluation process. We plan to extend our work to the evaluation of non-text regions in the future. A training dataset $\mathcal{T}$ of 100 document pages was randomly sampled from the selected 978 documents; the remaining 878 document pages are considered as the test dataset $\mathcal{S}$.

8.2 Algorithm Training

The parameters that a segmentation algorithm is sensitive to are automatically selected by training the algorithm on the 100-page training dataset $\mathcal{T}$. A direct search optimization procedure [27] is used to search for the optimal parameter value for each algorithm. A starting point is necessary for the optimization procedure. Based on information about the document page style, a reasonable working range can be selected for each parameter of each algorithm. We chose a relatively conservative range to make sure that the true optimum parameter values fell within the range. 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 maximum of the six optimal parameter values attained in the six searches. In the following sections, we specify the parameters that we optimized for each algorithm and the corresponding reasonable working ranges. We fix the parameters that the algorithm is insensitive to and only train the ones that the algorithm is sensitive to. The training procedure is conducted on a randomly selected 100-page training dataset $\mathcal{T}$.

8.2.1 X-Y Cut Algorithm Parameters

The X-Y cut algorithm [25, 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 given below:

1. X widest zero valley width threshold T_X^C: {20-250 pixels};
2. Y widest zero valley width threshold T_Y^C: {20-200 pixels};
3. Vertical noise removal threshold T_X^n: {20-100 pixels};
4. Horizontal noise removal threshold T_Y^n: {20-100 pixels}.

Since in most cases the vertical cut is longer than horizontal cut, we set the maximum of T_X^C be larger than that of T_Y^C. Furthermore, since most inter-line gaps are less than 100 pixels, we set the maximum of T_X^n and T_Y^n to 100 pixels.

8.2.2 Docstrum Algorithm Parameters

O'Gorman in his paper specifies 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 handing 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 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 four parameters that the algorithm is sensitive to were searched for in the reasonable working ranges given below:

1. Nearest neighbor threshold factor f_t: {1-5};
2. Parallel distance threshold factor f_{pa}: {2-10};
3. Perpendicular distance threshold factor f_{pe}: {0.5-5};
4. Character size ratio factor f_d: {2-10}.

8.2.3 Voronoi-Diagram-Based Algorithm Parameters

Kise's algorithm has eleven free parameters and is insensitive to seven of them. 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, $A_z = 50$ 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, 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};
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}.

8.3 Algorithm Testing

All five algorithms were tested on the 878-page test dataset $\mathcal{S}$. In order to be able to compare the timing information for each algorithm, we tested all algorithms on the same machine.

8.4 Hardware and Software Environments

In this section, we provide the details of the hardware and software environments used for the implementation and training of the research algorithms and the testing of both the research algorithms and the commercial products.

8.4.1 Implementation

We implemented the X-Y cut and Docstrum algorithms based on [25, 26, 28]. The platform used for the implementation was an Ultra 1 Sun workstation running the Solaris 2.6 operating system. The compiler used was GNU gcc 2.7.2. In our implementation of the two research algorithms and the benchmarking algorithm, a DAFS library of Release 1.0 developed by RAF Technology Inc. [35] was used. Kise [19] provided us with an implementation of his Voronoi-based segmentation algorithm. We wrote programs in the Visual C++ 5.0 environment to extract zone coordinates from the OCR output of the two commercial products.

8.4.2 Training Phase

The machines we used for training were Ultra 1,2 and 5 Sun workstations running the Solaris 2.6 operating system. We used a direct search simplex algorithm for searching for a set of optimal parameter values for each research algorithm.

8.4.3 Testing Phase

In order to be able to compare the timing information for the algorithms, we tested them on a single machine, an Ultra 1 Sun workstation running the Solaris 2.6 operating system. The CPU speed reported by the `fpversion` UNIX command was 167 MHz. The two commercial products were tested on a Gateway PC with a 400 MHz Pentium II CPU running the Windows 95 operating system. We normalized the PC timing to UNIX timing using the relation $t_{UNIX} = 400 \cdot t_{PC}/167$ for comparison with the timing of the research algorithms.

9 Experimental Results and Discussion

In this section, four aspects of the experimental results are reported: training, test, statistical analysis and error analysis. In the training section, we report optimal parameter values and training times, show the convergence curves, and discuss the convergence rate for each research algorithm on the training dataset $\mathcal{T}$. In the testing section, we report the performance metric and timing results for all five algorithms on test dataset $\mathcal{S}$. Furthermore, a confidence interval is calculated for the results. In the statistical analysis section, we compare the performance metric and timing of each possible algorithm pair using the paired model. In the error analysis section, we report error analysis results on three different error categories for each algorithm and identify the possible sources of these errors for each research algorithm. Finally, based on the discussion in the error analysis section, we provide some recommendations for users to choose appropriate page segmentation algorithms for their purpose.

9.1 Training Results

Three research algorithms were trained on a 100-page training dataset $\mathcal{T}$. Table 2, Table 3 and Table 4 report the optimum parameters, optimum performance index (textline accuracy) value, and training time corresponding to each randomly selected starting point for the X-Y cut, Docstrum and Voronoi algorithms respectively. 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. Figure 4, Figure 5 and Figure 6 show the convergence characteristics for the X-Y cut, Docstrum and Voronoi algorithms respectively.

The findings from the training results for each research algorithm are summarized as the follows:

Table 1: Optimal parameter values for each research algorithm.

algorithm values	optimal parameter value	error rate (percent)	function evaluations	timing (hours)
X-Y cut	$(78, 32, 35, 54)$	14.71	86	12.70
Docstrum	$(2.578, 2.345, 0.600, 9.930)$	5.00	108	7.00
Voronoi	$(6, 11, 0.083, 200)$	4.73	52	6.99

1) The X-Y cut algorithm. From Table 2 and Figure 4, we can make the following observations:

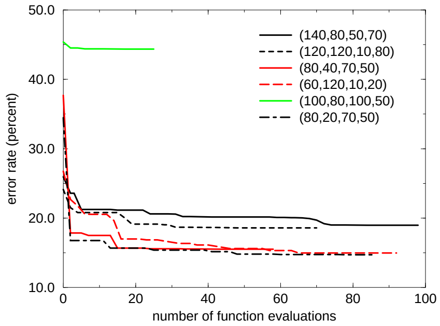


Figure 4: Convergence curves corresponding to six randomly selected starting points in the training of the X-Y cut algorithm.

Table 2: Optimization results of the X-Y cut algorithm for six randomly selected starting points within a reasonable working parameter subspace.

starting parameter values	optimal parameter value	error rate (percent)	number of function evaluations	timing (hours)
$(140, 80, 50, 70)$	$(128, 62, 23, 91)$	18.96	98	25.88
$(120, 120, 10, 80)$	$(97, 107, 22, 97)$	18.57	70	10.45
$(80, 40, 70, 50)$	$(82, 64, 21, 89)$	15.52	58	11.32
$(60, 120, 10, 20)$	$(67, 55, 21, 70)$	14.96	92	17.64
$(100, 80, 100, 50)$	$(100, 79, 100, 49)$	44.38	25	4.33
$(80, 20, 70, 50)$	$(78, 32, 35, 54)$	14.71	86	12.70

- The error rates for all starting points (except one) converge in the range of 14.71% to 18.96%,
- The convergence rate before the first 20 function evaluations is much faster than that beyond 20 function evaluations,
- Most values of parameter T_X^n are larger than those of parameter T_Y^n,

- All values (except those of the outlier point) of parameter T_X^C are smaller than those of parameter T_Y^C,
- Except for the unusual point, the variance of the number of function evaluations is small,
- 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 minima is not very stable. The algorithm only needs about 20 function evaluations to reach stable performance. The vertical cuts are generally longer than horizontal cuts. The vertical inter-zone gaps are generally wider than horizontal inter-zone gaps.

2) Docstrum algorithm. From Table 3 and Figure 5, we can make the following observations:

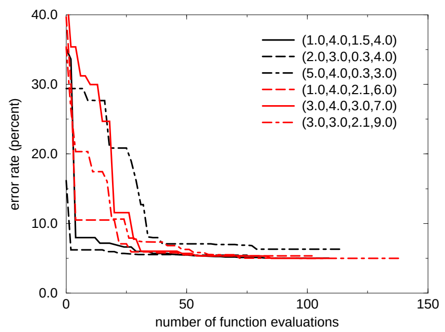


Figure 5: Convergence curves corresponding to six randomly selected starting points in the training of the Docstrum algorithm.

Table 3: Optimization results of the Docstrum algorithm for six randomly selected starting points within a reasonable working parameter subspace.

starting parameter values	optimal parameter value	error rate (percent)	number of function evaluations	timing (hours)
(1.0, 4.0, 1.5, 4.0)	(2.327, 2.344, 0.597, 5.223)	5.01	109	15.08
(2.0, 3.0, 0.3, 4.0)	(2.362, 2.129, 0.597, 4.383)	5.44	75	10.31
(5.0, 4.0, 0.3, 3.0)	(3.072, 2.138, 0.302, 3.602)	6.30	115	12.74
(1.0, 4.0, 2.1, 6.0)	(2.537, 1.973, 0.645, 7.551)	5.34	102	13.66
(3.0, 4.0, 3.0, 7.0)	(2.578, 2.345, 0.600, 9.930)	5.00	108	7.00
(3.0, 3.0, 2.1, 9.0)	(2.521, 2.336, 0.595, 10.375)	5.00	139	12.77

- The error rates for all starting points except an outlier converge in the range of 5.00% to 6.30%,
- The convergence rate before the first 50 function evaluations is much faster than that beyond 50 function evaluations,

- The parameters f_t, f_{pa} and f_{pe} converge to very similar values from all starting points,
- There is a large variation in the optimal values of parameter f_d,
- The total number of function evaluations is larger than those for the X-Y cut and Voronoi algorithms.

From the above observations, we can see that the performance of the algorithm stabilizes after about 50 function evaluations, which is much larger than 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 inter-character and inter-line gap estimation is not accurate, and hence more errors will occur. However, for the other three parameters, the fact that the optimal values are very close implies the objective function may have a single "valley" in the neighborhood of these parameter values.

3) Kise's Area-Voronoi-Diagram-Based algorithm. From Table 4 and Figure 6, we can make the following observations:

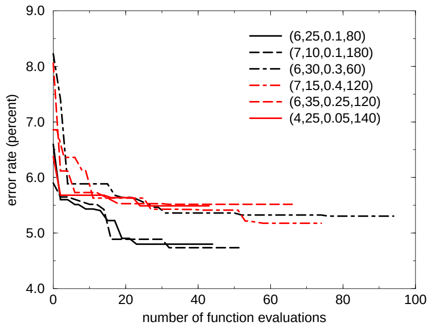


Figure 6: Convergence curves corresponding to six randomly selected starting points in the training of the Voronoi algorithm.

Table 4: Optimization results of the Voronoi algorithm for six randomly selected starting points within a reasonable working parameter subspace.

starting parameter values	optimal parameter value	error rate (percent)	number of function evaluations	timing (hours)
$(6, 25, 0.1, 80)$	$(6, 15, 0.084, 108)$	4.80	44	8.77
$(7, 10, 0.1, 180)$	$(6, 11, 0.083, 200)$	4.73	52	6.99
$(6, 30, 0.3, 60)$	$(6, 11, 0.146, 149)$	5.31	94	24.11
$(7, 15, 0.4, 120)$	$(8, 11, 0.0977, 191)$	5.17	74	19.17
$(6, 35, 0.25, 120)$	$(6, 11, 0.246, 193)$	5.52	66	8.69
$(4, 25, 0.05, 140)$	$(4, 11, 0.134, 161)$	5.49	43	7.17

- The error rates for all starting points converge in the range of 4.73% to 5.52%,

- The convergence rate before the first 20 function evaluations is much faster than that beyond 20 function evaluations,
- The value parameter nm for most (five) starting points converges to 11 pixels,
- There is a relatively small variance in the convergence values of parameters sr and ta,
- There is a relatively large variance in the convergence values of parameter fr,
- There is a relatively large variance in the number of function evaluations corresponding to the six starting points,

From the above observations, we can see that the Voronoi algorithm objective function has multiple local minima, but the performance at these local minima is stable. The algorithm needs only about 20 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 non-text 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 dataset $\mathcal{S}$ with their respective optimum parameters. Table 5 reports the performance index (textline accuracy) and average algorithm timing on the test dataset $\mathcal{S}$. Figure 7 gives a bar-chart representation of the testing results for each evaluated algorithm.

Table 5: Algorithm testing results and the corresponding 95% confidence intervals. The average time per page is also reported. The times 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	94.64 ± 0.78	9.09 ± 0.18
Docstrum	94.11 ± 0.99	15.43 ± 0.32
X-Y cut	82.94 ± 1.61	6.37 ± 0.07
Caere	93.97 ± 0.85	2.02 ± 0.01, (normalized) 4.84 ± 0.04
ScanSoft	87.29 ± 1.35	3.13 ± 0.04, (normalized) 7.52 ± 0.10

From the testing results, we see that the Voronoi-based, Docstrum, and Caere algorithms have similar performance indices which 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, whereas Docstrum has the greatest average processing time. The connected component labeling method 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 are statistically significant or not,

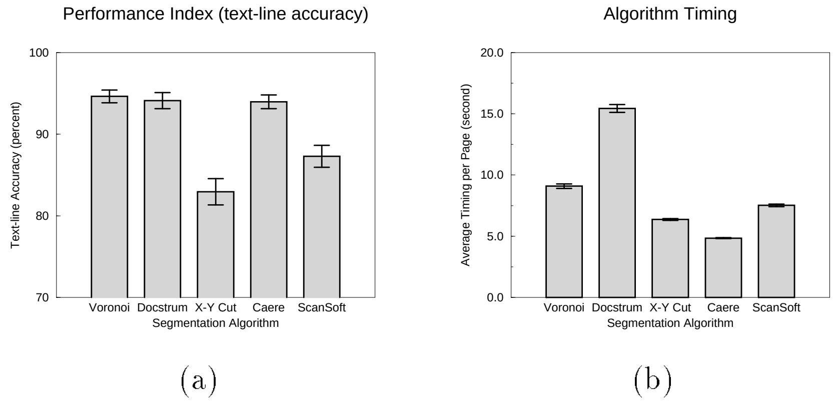


Figure 7: The first three algorithms in the bar chart are reseach algorithms, and the last two algorithms are commercial products. (a) shows the testing results of the performance index (textline accuracy) for each algorithm. A 5% level t-test indicates that the performances of Voronoi, Docstrum and 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 the algorithm testing time results for each algorithm. A 5% level t-test indicates that each algorithm's timing is significantly different from that of any other algorithm. From fastest to slowest, the algorithms are ranked as: Caere, X-Y cut, ScanSoft, Voronoi and Docstrum.

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 [18] to compare the performance index and testing time differences between each possible algorithm pair, and then compute their confidence intervals. The analysis results for performance index and processing time are reported in matrix form in Table 6 and Table 7 respectively. If we denote by T_{ij} the value of the table cell in the ith row and jth column, $T_{ij} = a_i - a_j$ where a_i is the performance index (processing time) value of the algorithm in the ith row, and a_j is the performance index (processing time) value of algorithm in the jth column. Note that the normalized processing time is used for the two commercial products.

From Table 6, we find that the differences between the performance indices of Kise's 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. From Table 7, we can 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's segmentation algorithm, X-Y cut, ScanSoft's segmentation algorithm, Kise, and Docstrum. For Docstrum, a better connected component labeling algorithm might improve its timing performance.

Table 6: 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 the Voronoi, Docstrum and Caere algorithms. However, this group is significantly better than Scansoft, which is in turn is better than XY-cut.

	Caere	Docstrum	ScanSoft	X-Y cut
Voronoi	0.66 ± 1.17 $P_{val} = 0.13$	0.52 ± 1.23 $P_{val} = 0.20$	7.33 ± 1.55 (*) $P_{val} = 9.02E-20$	11.69 ± 1.80 (*) $P_{val} = 2.29E-34$
Caere	–	-0.13 ± 1.09 $P_{val} = 0.40$	6.67 ± 1.38 (*) $P_{val} = 1.26E-20$	11.04 ± 1.67 (*) $P_{val} = 1.21E-35$
Docstrum	–	–	6.81 ± 1.59 (*) $P_{val} = 8.06E-17$	11.18 ± 1.79 (*) $P_{val} = 4.61E-32$
ScanSoft	–	–	–	4.36 ± 1.87 (*) $P_{val} = 2.79E-06$

Table 7: 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 greatest averge processing time, the algorithms are ranked as: Caere, X-Y cut, ScanSoft, Voronoi and Docstrum.

	Caere	Docstrum	ScanSoft	X-Y cut
Voronoi	4.25 ± 0.16 (*) $P_{val} = 0$	-6.35 ± 0.18 (*) $P_{val} = 0$	1.57 ± 0.20 (*) $P_{val} = 1.91E-48$	2.72 ± 0.13 (*) $P_{val} = 0$
Caere	–	-10.59 ± 0.29 (*) $P_{val} = 0$	-2.68 ± 0.10 (*) $P_{val} = 0$	-1.53 ± 0.05 (*) $P_{val} = 0$
Docstrum	–	–	7.91 ± 0.32 (*) $P_{val} = 0$	9.06 ± 0.26 (*) $P_{val} = 0$
ScanSoft	–	–	–	1.15 ± 0.12 (*) $P_{val} = 1.06E-66$

9.4 Error Analysis

Error analysis is crucial to interpreting the functionalities of the evaluated algorithms. Each algorithm has different weakness. Figure 8 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 inter-character and inter-line spacings are estimated for each individual document image. Titles with wide inter-character and inter-word 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 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. Also, Docstrum estimates inter-character and inter-line spacing for each individual document image. 3) In the Voronoi-based algorithm's implementation, in addition to what has been done for Docstrum, Kise uses not only the spacing of the connected components but also their area ratios to generate zone boundaries. Hence a few lines or noise blocks between text regions do not cause horizontal merges, whereas they do cause horizontal merges in the Docstrum algorithm. This is the main reason why Docstrum has more horizontally merged textlines than the Voronoi-based algorithm. We can see that among the research algorithms, the X-Y cut has the highest mis-detection error rate while Voronoi and Docstrum have negligible error rates. This is again due to the global thresholds of the X-Y cut algorithm which cause textlines such as headers, footers, authors and 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:

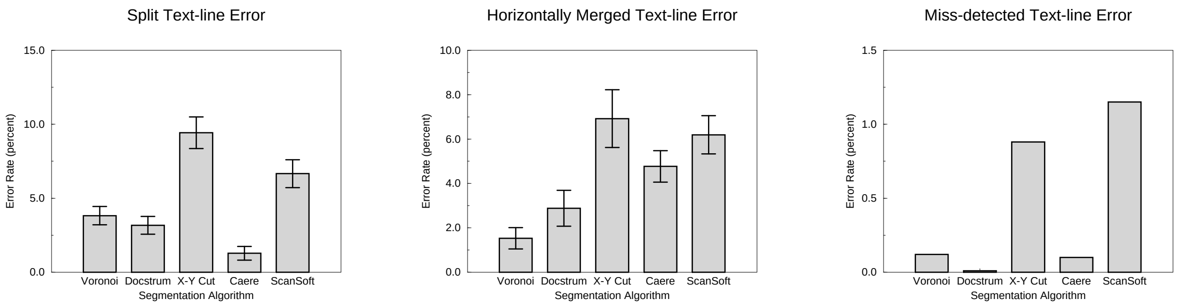
- For segmentation of document pages with large skew angles or large noise blocks (especially "L"-shaped, "U"-shaped or close-shaped thick noise bars), the X-Y cut algorithm is a bad choice.

- 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.

- For segmentation of document images with a large font size range, irregular inter-character or inter-line spacing, few noise blocks, and negligible skew angles, X-Y cut is a better choice than either the Voronoi-based or Docstrum algorithms.

- The Voronoi-based algorithm is preferred over the Docstrum algorithm in general.

- For the X-Y cut algorithm, first remove large noise blocks by labeling connected components and then removing the larger ones.



(a) (b) (c)

Figure 8: This figure 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 error. We can see that a 5% level t-test indicates that 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 rates of Voronoi, Docstrum and Caere are significantly different from each other. (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 horizontal merge error. We can see that a 5% level t-test indicates that the error rate of all five algorithms 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 mis-detection error. We can see that the rate of this error type is much smaller than those of the other two error types. From the lowest to the highest error rate, the algorithms are ranked as: Docstrum, Voronoi, Caere, X-Y cut and ScanSoft.

10 Conclusions

We have proposed a five-step performance evaluation methodology for evaluating page segmentation algorithms: 1) First we randomly partition the dataset $\mathcal{D}$ into a mutually exclusive training dataset $\mathcal{T}$ and test dataset $\mathcal{S}$ with both textline-level and zone-level groundtruth, 2) we then define textline accuracy as the performance metric, 3) the Nelder-Mead simplex search algorithm is then used to search automatically for the optimal parameter values of the segmentation algorithms, 4) the segmentation algorithms are then evaluated on the test dataset $\mathcal{S}$ using their corresponding optimal parameter values, and finally 5) a paired-model statistical analysis and an error analysis are performed to provide confidence intervals and to test hypotheses regarding the performance indices and algorithm timings. The errors of three research algorithms were analyzed in terms of mis-detection, split and horizontal merge error types. We found that the performances 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 the algorithms are significantly different from one another. From the fastest to the slowest, the algorithms are ranked as Caere, X-Y cut, ScanSoft, Voronoi and Docstrum. In the error analysis, we found that X-Y cut has the most split and horizontal merge errors due to its global thresholds, Voronoi has the least horizontal merge errors due to its usage of area ratio information of connected components, and Caere has the least split error. We intend to extend this work to evaluation of tables, graphs and half-tone images.

Acknowledgement

We would like to thank Dr. Paul Smith of Department of Mathematics for discussions on various statistical issues; Dr. Kise of the Osaka Prefecture University for providing us with a software implementation of his segmentation algorithm and modifying it for our evaluation purposes; Mindy Bokser of Caere Corporation for providing us with the Caere page segmentation software; Dr. Henry Baird of Xerox Corporation for introducing us to Kise's algorithm; Greg Marton of the University of Maryland for his help in obtaining experimental data; and Dr. Azriel Rosenfeld of the University of Maryland for his comments.

We would also like to thank Steve Dennis and Glenn Van Doren of the Department of Defense and Melissa Holland and Jeff DeHart of the Army Research Laboratory for supporting this work. This research was funded in part by the Department of Defense and the Army Research Laboratory under Contract MDA 9049-6C-1250.

References

[1] H. Baird. Background structure in document images. *International Journal of Pattern Recognition and Artificial Intelligence*, 8:1013–1030, 1994.

[2] H. S. Baird, S. E. Jones, and S. J. Fortune. Image segmentation by shape-directed covers. In *Proceedings of International Conference on Pattern Recognition*, pages 820–825, Atlantic City, NJ, June 1990.

[3] K. W. Bowyer and P. J. Phillips, editors. *Empirical Evaluation Techniques in Computer Vision*, Santa Barbara, CA, June 1998.

[4] M. J. Box. A comparison of several current optimization methods, and the use of transformations in constrained problems. *Computer Journal*, 9:67–77, 1966.

[5] T. Breuel and M. Worring, editors. *Document Layout Interpretation and its Applications*, Bangalore, India, September 1999.

[6] Caere Co. *Caere Developer's Kit 2000.*

[7] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:910–918, 1988.

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

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

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

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

[12] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:523–550, 1987.

[13] 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 Transactions on Pattern Analysis and Machine Intelligence*, 18:673–689, 1996.

[14] J. J. Hull. Performance evaluation for document analysis. *International Journal of Imaging Systems and Technology*, 7:357–362, 1996.

[15] A. K. Jain and B. Yu. Document representation and its application to page decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:294–308, 1998.

[16] J. Kanai, S. V. Rice, T. A. Nartker, and G. Nagy. Automated evaluation of OCR zoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:86–90, 1995.

[17] T. Kanungo, M. Y. Jaisimha, J. Palmer, and R. M. Haralick. A methodology for quantitative performance evaluation of detection algorithms. *IEEE Transactions on Image Processing*, 4:1667–1674, 1995.

[18] 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*, volume 3651, pages 109–120, San Jose, CA, January 1999.

[19] K. Kise, A. Sato, and M. Iwata. Segmentation of page images using the area Voronoi diagram. *Computer Vision and Image Understanding*, 70:370–382, 1998.

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

[21] R. M. Lewis, V. Torczon, and M. W. Trosset. Why pattern search works. *OPTIMA*, 59:1–7, 1998.

[22] 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*, volume 3027, pages 149–160, San Jose, CA, February 1997.

[23] S. Mao and T. Kanungo. Empirical performance evaluation of page segmentation algorithms. In *Proceedings of SPIE Conference on Document Recognition*, San Jose, CA, January 2000, to appear.

[24] G. Matheron. *Random Sets and Integral Geometry.* Wiley, New York, 1975.

[25] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proceedings of International Conference on Pattern Recognition*, volume 1, pages 347–349, Montreal, Canada, July 1984.

[26] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25:10–22, 1992.

[27] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[28] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1162–1173, 1993.

[29] L. O'Gorman and R. Kasturi. *Document Image Analysis.* IEEE Computer Society Press, Los Alamitos, CA, 1995.

[30] T. Pavlidis and J. Zhou. Page segmentation and classification. *Graphical Models and Image Processing*, 54:484–496, 1992.

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

[32] I. T. Phillips and A. K. Chhabra. Empirical performance evaluation of graphics recognition systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:849–870, 1999.

[33] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss. The FERET evaluation methodology for face-recognition algorithms. Technical Report NISTIR 6264, National Institute of Standards and Technology, Gaithersburg, MD, 1999.

[34] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, 1998.

[35] RAF Technology, Inc. *DAFS Library Programmer's Guide and Reference.*

[36] S. Randriamasy and L. Vincent. Benchmarking page segmentation algorithms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 411–416, Seattle, WA, June 1994.

[37] S. Randriamasy, L. Vincent, and B. Wittner. An automatic benchmarking scheme for page segmentation. In *Proceedings of SPIE Conference on Document Recognition*, volume 2181, pages 217–230, San Jose, CA, February 1994.

[38] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fifth annual test of OCR accuracy. Technical Report TR-96-01, University of Nevada, Las Vegas, NV, 1996.

[39] ScanSoft Co. *TextBridge: Application Programmer's Interface.*

[40] ScanSoft Co. *XDOC Data Format.*

[41] B. K. Schnabel. An investigation into the effects of random error on a selection of current minimization methods. Master's thesis, University of Leeds, UK, 1966.

[42] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, New York, 1982.

[43] F. Wahl, K. Wong, and R. Casey. Block segmentation and text extraction in mixed text/image documents. *Graphical Models and Image Processing*, 20:375–390, 1982.

[44] M. H. Wright. Direct search methods: Once scorned, now respectable. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1995*, pages 191–208. Addison-Wesley, Longman (Harlow), 1996.

[45] B. A. Yanikoglu and L. Vincent. A complete environment for ground-truthing and benchmarking document page segmentation. *Pattern Recognition*, 31:1191–204, 1998.