Document Structure Analysis and Performance Evaluation

Jisheng Liang

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

1999

Program Authorized to Offer Degree: Electrical Engineering

University of Washington

Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Jisheng Liang

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

____

Robert M. Haralick

Reading Committee:

____

Linda Shapiro

____

Jenq-Neng Hwang

____

Ihsin T. Phillips

Date: __________________________

University of Washington

Abstract

Document Structure Analysis and Performance Evaluation

by Jisheng Liang

Chair of Supervisory Committee

Professor Robert M. Haralick
Electrical Engineering

The goal of the document structure analysis is to find an optimal solution to partition the set of glyphs on a given document to a hierarchical tree structure where entities within the hierarchy are associated with their physical properties and semantic labels. In this dissertation, we present a unified document structure extraction algorithm that is probability based, where the probabilities are estimated from an extensive training set of various kinds of measurements of distances between the terminal and non-terminal entities with which the algorithm works. The off-line probabilities estimated in the training then drive all decisions in the on-line segmentation module. An iterative, relaxation like method is used to find the partitioning solution that maximizes the joint probability. This approach can be uniformly apply to the construction of the document hierarchy at any level. We have implemented the text line and text block extraction algorithms using this framework. The algorithms were evaluated on the UW-III database of some 1600 scanned document image pages. The text line extraction algorithm identifies and segments 99.76% of text lines correctly, while the preliminary result of the text block extraction shows 91% accuracy.

Another example of our framework is the development of a system that detects and

recognizes special symbols (Greek letters, mathematical symbols, etc.) on technical document pages, that are not handled by the current Optical Character Recognition (OCR) systems.

A large quantity of ground-truth data, varying in quality, is required in order to give an accurate measurement of the performance of an algorithm under different conditions. We constructed the University of Washington English Document Image Database-III, which contains 1600 English document images that come with manually edited ground-truth of entity bounding boxes. This database can be utilized by the OCR and document understanding community as a common platform to develop, test and evaluate their systems. Based on the ground-truth data, we can evaluate the performance of document analysis algorithms and build statistical models to characterize various types of document structures. In this dissertation, we present a set of quantitative performance metrics for each kind of information a document analysis technique infers.

We developed a protocol to evaluate the performance of graphics recognition systems on images that contain straight lines (solid or dashed), circles (solid or dashed), partial arcs of circles (solid or dashed), and text blocks. Our evaluator is designed to be used by recognition system researchers and developers for testing and enhancing their recognition algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Chapter 1

INTRODUCTION

1.1 Motivation

Document image analysis and recognition aims at the automated transformation of the information contained in paper documents into a symbolic representation which can be accessed electronically. Capturing information from paper documents is very expensive and labor intensive. It has been estimated that worldwide about $250 billion are spent annually (largely in operator salaries, etc.) in keying in information from paper documents, and this in capturing information from only 5% of the available documents [2]. Only a negligible portion of the enormous amount of material now on the Internet is the result of automated processing.

A document image is a visual representation of a printed page such as a journal article page, a business letter, a newspaper page, a mail piece, etc. A document image may contain text, graphics, pictures, or a combination of these. Document images present a more structured domain than general images. Beyond traditional Optical Character Recognition (OCR), there is often the need to preserve a document's structure and format when it is converted into electronic form: one would ideally like a document analysis system to be in a WYSIWYG - What You Scan is What You Get - fashion. The ultimate goal of a complete document image analysis system is to transform paper documents into a hierarchical representation of their structure and content (see Figure 1.1). The transformed document representation enables document interchange, editing, browsing, indexing, filing and retrieval since the document content can be accessed by its structure rather than simply by its imaged form [17]. A wide range of applications of document analysis systems are in postal service, government, health care, library, etc.

Some commercial products are already available such as OCR systems for reading pages of machine-printed text, but research is still required to improve their performance on the full range of real-world variability in typography, image quality, and context. Performance evaluation of document analysis systems requires experimental design, a large train and test database, and sophisticated analysis of the results.

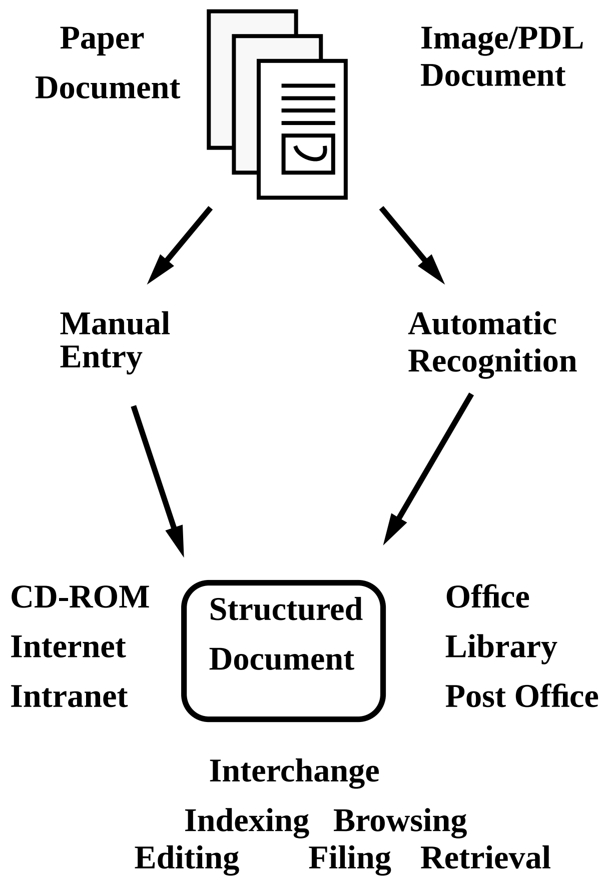


Figure 1.1: Illustrates the process of document data entry.

1.2 Contributions

The main contributions that are presented in this dissertation are:

1. A complete and precise definition of the document structure and the document structure analysis problem.

2. A probability-based and unified approach for document structure analysis.

3. An environment for evaluating document analysis algorithms and systems:

   (a) The UW-III document image database with 1600 manually ground-truthed technical journal document images. The ground-truth of line-drawing, mathematical and chemical formulae are also provided.

   (b) A wide-spectrum of quantitative performance metrics for evaluating document analysis algorithms.

   (c) A performance evaluation protocol for graphics recognition systems.

   (d) Systematic parameter estimation and model checking methodology.

4. Application of the proposed document structure analysis algorithms on a group of new and real applications:

   (a) Extract document structure from PostScript/PDF (portable Document Format) files and output to the SGML (Standard Generalized Markup Language) format.

   (b) Detect and recognize special symbols (Greek letters, mathematical symbols, etc.) from the technical document images not handled by the current OCR systems.

1.3 Overview of Dissertation

Chapter 2 reviews the state of the art of the document structure analysis, and then presents a formal document structure representation and the problem statement. Chapter 3 introduces the UW III document image database and a set of quantitative and suitable performance metrics for each kind of information a document analysis technique infers. Chapter 4 presents a probability-based and unified approach for document structure analysis and its implementation on extracting text lines and text blocks. Chapter 5 describes a system for detecting and recognizing special symbols not handled by the current OCR systems. Chapter 6 presents a performance evaluation protocol for graphics recognition systems. Finally, Chapter 7 concludes the dissertation and addresses future extension of the work.

Chapter 2

DOCUMENT STRUCTURE ANALYSIS PROBLEM

In this chapter, we first review some of the earlier work in the area of document structure analysis. In Section 2.2, we present a formal document structure model and give the formal problem statement.

2.1 Literature Review

Current document analysis systems usually consist of two processes: the segmentation process discovers various entities of interest in an input document image; the classification process identifies the detected entities' types. There are two major approaches of segmentation and classification. In the first approach, the document image is decomposed into a set of zones enclosing either textual or nontextual content portions using the global spatial properties. The statistical and textural features are extracted and used to label each zone as one of the predefined categories [22]. This top-down decomposition scheme does not work for certain types of document layout topology. This is especially true when there is noise present on the image or the document page is skewed. In the second approach, the low-level segments (connected components [23], line segments [21], etc.) are extracted and classified as text or non-text. Then the text segments are grouped into text blocks. A survey of the document physical and logical structure extraction algorithms can be found in [1].

Most of the current structure recognition algorithms assume some prior knowledge of the typographical and layout conventions of document. The required information about the document properties ranges from very specific and precise to fairly general ideas. By considering the knowledge representation and structure recognition approaches, the structure analysis algorithms can be categorized into two main types: rule or grammar based, and statistical model based. The grammar based approaches use multiple grammars to represent the recognition algorithm. The recognition task is formulated as a parsing problem using one dimensional string pattern matches for searching.

Ha *et al.* [11, 15] present an algorithm based on recursive cutting of connected component projection profiles for segmenting binary document images into zones; zones are classified as textual and non-textual; then the text zones are decomposed to text blocks, text line, and words. Empirically determined thresholds are used at each cutting step.

Nagy *et al.* [38] discussed a prototype of document image analysis system for technical journals. They integrate document layout analysis and commercial OCR to generate information for document browsing system. The specific knowledge of predetermined layout convention for a specific family of publications are encoded beforehand as block grammars in the form of a tree. Documents that satisfy the postulated layout conventions are parsed into different components.

Hu and Ingold [12] present a document description language that is similar to an attributed grammar and includes composition rules and presentation rules. The system consists of two main programs: the compiler, which compiles the description of a document class into one document graph and several fragment graphs; and the analyzer, which turns the page images of a document into their structural representations based on both kinds of graphs presented above. Matching checks the similarity between the extracted typographic attributes of a physical element and those of a node in an analysis graph. Parsing is achieved by finding the minimum cost path through an analysis graph for a list of physical elements.

Dengel [6] developed a system for partitioning raster images of business letters into logically labeled area items. It employs various knowledge resources utilizing spatial and geometric characteristics about the formal style of business letters. Due to the unsupervised learning of document model, the system allows the establishment of a specific decision tree classifier which serves as the knowledge representation.

Most works deal with specific documents for which the information about the document style is a priori well-defined. One way of knowledge acquisition is manually generating the heuristic rules, or grammars, by carefully looking through a set of representative document pages. This task is tedious and requires a great expertise. Hand-crafted rules and model tend to be brittle and only work for a specific kind of document. This means that these approaches analyze successfully only those documents having previously defined structure and they fail when a new kind of form has to be analyzed or the documents have some part of variations.

Most of the rule or grammar based algorithms use fixed, global distance thresholds for merging components into words, or words into text lines, etc. These thresholds should be determined locally based on the adjacent text size.

Chen *et al.* [53] describe a text word, line and block segmentation algorithm for horizontal rectangular layouts. The morphological closing transform is applied to the binary document image. The extracted segments are classified as words or non-text according to their size. The algorithm groups extracted words into text lines based on statistical models of the colinearity and equal spacing of words within the text lines. The text lines are then merged into text blocks according to a statistical model of the homogeneity of height, width, leading, and justification within text blocks.

Since the measurements made on document may have errors, and the knowledge about the document style may be ambiguous, the structure extraction algorithm is required to be error-tolerant and robust. The problem of uncertainty in knowledge and evidence and the information integration of multiple evidences have to be solved.

Srihari *et al.* [18] developed a postal automation system that locates and interprets destination address blocks on letter mail pieces with a high success rate and high speed. The system is based on data-driven processing. It extracts primitive information from the image and groups the information into possible address blocks. The evidence combination tool integrates pieces of evidence generated for a block into a single block labeling hypothesis. The best candidate block is selected by verifying the consistency of labeling hypotheses by using spatial relations.

The approximate reasoning, based on fuzzy logic, is used by Hu and Ingold [12] to improve the system robustness.

In order to build a near-perfect system, it is necessary to replace heuristics by systematics and rely on mathematical optimization rather than intuition. Only the algorithm of Chen *et al.* [53] takes a systematic approach to integrating the evidence sources they use. Their algorithm computes Bayesian estimates of the text lines and text blocks based on statistical models. However, their algorithm's underlying models ignore the fact that some of the parameters are dependent on the text's font size. Thus the algorithm may achieve poor results for documents containing a wide variety of fonts and sizes.

The well-defined and unambiguous performance measures, combined with reproducible experiments, are necessary in every area of science and technology. However, it is clear that many of the early works on document analysis systems provide illustrative results and hardly any have their techniques tested on significant sized data sets and give quantitative performance measures [1]. The main reasons are the lack of precise and concise mathematical document model, the lack of accurate document ground truth data to train and test the algorithms, and the lack of appropriate and quantitative performance metrics and evaluation protocol. Since document analysis techniques are moving to the consumer market, they have to perform nearly perfectly. This means that they have to be proved out on significant sized data sets and that there must be suitable performance metrics for each kind of information a document analysis technique infers. From the user's point of view, the appropriate measure of a document recognition system is the total cost of document conversion, typically dominated by the cost of correcting residual errors in the output. However, such a measure is not necessarily appropriate for a researcher who seeks to measure progress in developing an algorithm. A developer might use a number of metrics at an early stage of development, and an overall cost to evaluate the final system. Different performance metrics may be important at different stages of system developed.

Currently, no standard testing procedures exist for measuring and comparing algorithms within a document analysis system. Randriamasy and Vincent [39] proposed a pixel-level and region-based approach to compare segmentation results and manually generated regions. An overlap matching technique is used to associate each region of one of the two sets, to the regions in the other set it has a non-empty intersection. Since the black pixels contained in the regions are counted rather than the regions themselves, this method is independent of representation scheme of regions. Quantitative evaluation of segmentation is derived from the cost of incorrect splittings and mergings. Working on the bit-map level, their technique involves extensive computation. They assume there is only one text orientation for the whole page. Kanai *et al.* [28] proposed a text-based method to evaluate the zone segmentation performance. They compute an edit distance which is based on the number of edit operations (text insertions, deletions, and block moves) required to transform an OCR output to the correct text. The cost of segmentation itself is derived by comparing the costs corresponding to manually and automatically zoned pages. This metric can be used to test "black-box" commercial systems, but is not able to help users categorize the segmentation errors. This method only deals with text regions. They assume the OCR performance is independent of the segmentation performance. Garris [29] proposed a scoring method which computes the coverage and efficiency of zone segmentation algorithm. The box distance and box similarity between zones are computed to find the matching pairs. This technique provides some numbers (scores), which are not able to help users analyze errors.

In this dissertation, we adopt an engineering approach to systematically characterizing the document structures based on a large document image database, and develop statistical methods to extract the document structure from the image. We develop suitable quantitative performance measures to evaluate our document structure analysis algorithms.

2.2 Document Structure Model and Problem Statement

A *polygonal area* on a document page is given by a pair (θ, I), where $\theta \in \Theta$ specifies the label that designates the physical type of content, i.e. text block, text line, word, table, equation, drawing, halftone, etc., and I is the area enclosed by boundary of the polygon. The boundary of the polygon is given as the sequence of line segments connecting the successive vertices of the polygon. The vertices are given as a clockwise ordered list of points.

A polygon is *homogeneous* if all its area is of one physical type and there is a standard reading order for the content within the area. Two polygons are physical *adjacent* if each has a significant length of a side which near parallel nearby, and are separated by a divider.

A set $\mathcal{A}$ of non-overlapping homogeneous polygonal areas and the properties associated with $\mathcal{A}$ is called a *polygonal structure.*

$V : \wp(\mathcal{A}) \rightarrow \Lambda$ specifies measurement made on subset of $\mathcal{A}$, where Λ is the measurement space.

The polygonal structure is associated with the following basic sets and mappings.

Content Type

Θ is the set of physical types (text block, text line, word, table, equation, drawing, etc.).

$C : \mathcal{A} \rightarrow \Theta$ associates polygonal areas with their physical types of content.

Γ is the set of functional types (paragraph, section, word, title, heading, caption, abstract, author, footnote, page number, etc.).

$M : \mathcal{A} \rightarrow \Gamma$ associates polygonal areas with their functional types of content.

Content

Σ is the alphabet consisting of symbols. Let Σ^* be the set of all sequences of symbols from Σ. $O : \mathcal{A} \to \Sigma^*$ associates polygonal areas with their contents.

Format Attribute

We denote by $\mathcal{F}$ the set of format attributes (font type, font size, font style, justification, indentation, etc.). $S : \Gamma \to \mathcal{F}$ specifies the format attributes for each functional type of content.

Location

We denote by $\mathcal{L}$ the set of qualitative locations (top left, bottom, middle, etc.). $\mathcal{P} \subseteq \Gamma \times \mathcal{L}$ specifies the "preferred" locations of different types of content.

Spatial Relation

$\mathcal{D}$ is the set of dividers (white space, ruling, etc.). $T : \Gamma \times \Gamma \to \mathcal{D}$ specifies the divider used between different types of content (inter-line spacing, inter-column spacing, spacing between figure and caption, etc.).

Reading Order

Let $\mathcal{A} = \{A_1, \cdots, A_K\}$ be the set of polygonal areas. The reading order R of $\mathcal{A}$ is a tuple $(r_1, \cdots, r_K)$ which is a permutation of $(1, \cdots, K)$.

The document structure is a hierarchical structure. Let Φ be the set of all polygonal structures on a given document page. We define the relation $\subseteq_\Phi$ in Φ as: $x \subseteq_\Phi y$ if and only if $x \subseteq y$ and $x, y \in \Phi$. This relation is a partial ordering E of a set Φ satisfies

1. $(a, a) \in E, \forall a \in \Phi$
2. $(a, b) \in E,$ and $(b, a) \in E \Rightarrow a = b$
3. $(a, b) \in E, (b, c) \in E \Rightarrow (a, c) \in E$

We denote by $\Theta = \{\theta_1, \theta_2, \cdots, \theta_k\}$ the set of physical content types on the document page. Let $\{\Phi_1, \cdots, \Phi_k\}$ be a partition of Φ, where each Φ_i is a set of mutually disjoint polygonal structures

$$\Phi_i = \{\phi | \phi \in \Phi, C(\phi) = \theta_i\}, \tag{2.1}$$

and if there exists Φ_j and $j > i$, then $\forall\ \phi_p \in \Phi_i$, either $\phi_p \subseteq_\Phi \Phi_j$, or ϕ_p and Φ_j are disjoint. An example of the document hierarchy is shown in Figure 2.1.

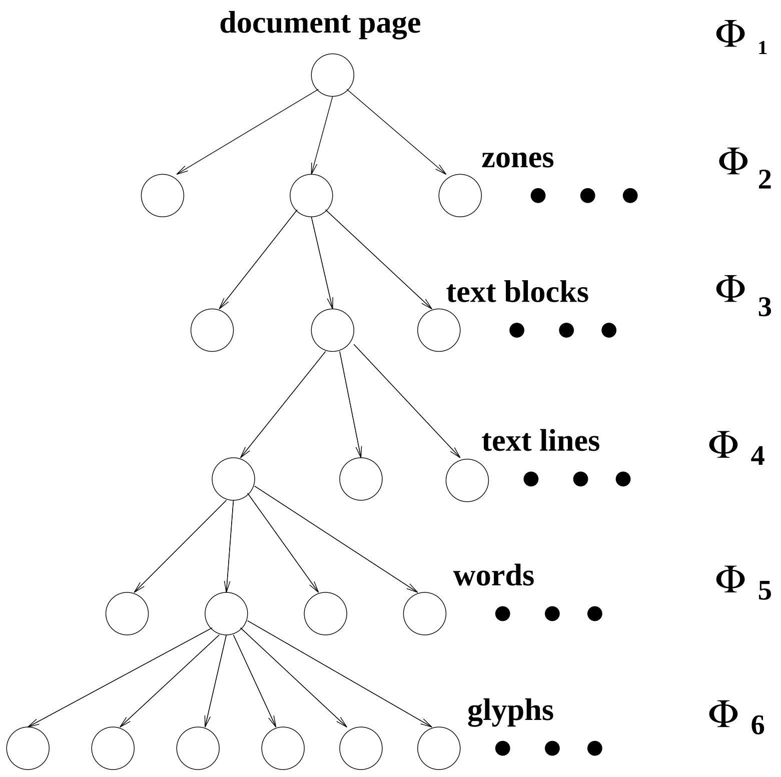


Figure 2.1: Illustrates a document hierarchy.

Document Structure Analysis Problem Statement

Given a document page Φ_1, extract a hierarchical document structure

$$\{\Phi_2, \Phi_3, \cdots \Phi_k\},$$

that maximizes the conditional probability

$$P(\Phi_2, \Phi_3, \cdots \Phi_k | \Phi_1). \tag{2.2}$$

The document structure analysis usually contains the following sub-problems:

- *Layout analysis* extracts a set $\mathcal{A}$ of homogeneous polygonal areas from a document page.
- *Logical analysis* $(M, \mathcal{R})$ involves assigning functional labels to each polygon of the page, and ordering the polygons according to their read order.
- *Content recognition* O associates polygonal areas with their contents.
- Page *style* $(S, T, \mathcal{P})$ describes the typographical properties for polygons with different content types, where S specifies the format attributes for each type of content, T specifies the divider used between different types of content, and $\mathcal{P}$ specifies the "preferred" locations of different types of content.

In this dissertation, we employ a hybrid technique for document structure extraction by analyzing the spatial configuration of the bounding boxes of different entities (connected components, words, text lines, etc.) on a given document page.

In our system, based on some assumptions on the conditional independence and ordering, the probability in Equation 2.2 is decomposed as follows,

$$\begin{aligned} & P(\Phi_2, \Phi_3, \cdots \Phi_6 | \Phi_1) \\ = \; & P(\Phi_6|\Phi_1) P(\Phi_4, \Phi_2|\Phi_6) P(\Phi_3|\Phi_4, \Phi_2) P(\Phi_5|\Phi_4), \end{aligned} \tag{2.3}$$

where $\Phi_1, \Phi_2, \cdots, \Phi_6$ represent page, zone, text block, text line, word, and glyph structures, respectively.

Figure 2.2 illustrates the processing steps of our document structure analysis system. The system reads a document page in image format or represented by a page description language, i.e. PostScript or Portable Document Format (PDF), and outputs the page's hierarchical structure.

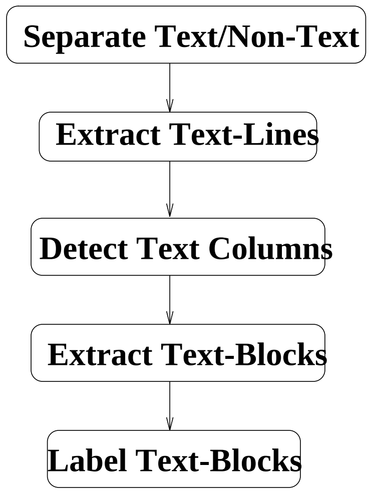


Figure 2.2: Illustrates the processes of our document structure analysis system.

Chapter 3

PERFORMANCE METRICS AND EVALUATION PROTOCOL OF DOCUMENT STRUCTURE ANALYSIS

Since document understanding techniques are moving to the consumer market, they have to perform nearly perfectly and efficiently. This means that they have to be proved out on significant sized data sets and there must be suitable performance metrics for each kind of information a document understanding technique infers [1]. However, many of the earlier techniques on document image analysis were developed on a trial-and-error basis and only gave illustrative results. There is a wide agreement upon the need of an automatic tool to benchmark the existing document analysis techniques.

This chapter introduces quantitative performance metrics for each kind of information a document analysis technique infers. A large quantity of ground-truth data, varying in quality, is required in order to give an accurate measurement of the performance of an algorithm under different conditions. In the University of Washington English Document Image Database-III [44], there are 1600 English document images that come with manually edited ground-truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text lines, and words. We do the performance evaluation by determining the correspondence between the ground-truth document structure and the automatically computed document structure, making the comparison between two structures, and reporting the performance measures.

In this chapter, a set of quantitative metrics are presented to evaluate the performance of the document analysis algorithms (Section 3.1). Section 3.2 describes content and creation of the UW-III document image database. Section 3.3 presents methods of parameter selection for rule-based algorithms and model checking for parametric document models. Finally, the performance evaluation results of a group of document layout analysis algorithms on the UW-III database are presented in Section 3.4.

3.1 Performance Assessment of Document Structure Analysis

Let g designate the ground truth polygonal structure, and d designate the detected structure, and $u(g,d)$ designate the utility of (g,d), the expected performance of an algorithm is

$$f = \sum_i \sum_f u(g,d) P(g,d), \tag{3.1}$$

where $P(g,d)$ is the probability of the algorithm output is d while the ground truth is g [57].

For each structure that we use to describe a document, there is an associated metric that measures the difference between a structure that is automatically produced and the ground-truth structure. We present the utility function and performance measure for the document analysis algorithms in this section.

Because the assessment tests only observes a finite sample, there is of necessity a difference between the observed performance on the test sample and the long-term performance on the total population. The issue of making the comparison between the specification and the observed performance is addressed by Haralick [30].

3.1.1 Matching of detected structure with the ground-truth

There are two problems in making the evaluation. The first is one of correspondence: which entities of the ground-truth set correspond to which entities of the automatically produced set. Once this correspondence is determined then a comparison of detected entities with the ground-truth entities can proceed.

Suppose we are given two sets $\mathcal{G} = \{G_1, G_2, \cdots, G_M\}$ for ground-truthed polygonal areas and $\mathcal{D} = \{D_1, D_2, \cdots, D_N\}$ for detected polygonal areas. The comparison of $\mathcal{G}$ and $\mathcal{D}$ can be made in terms of the following two kinds of measures:

$$\sigma_{ij} = \frac{\text{Area}(G_i \cap D_j)}{\text{Area}(G_i)} \; \textit{and} \; \tau_{ij} = \frac{\text{Area}(G_i \cap D_j)}{\text{Area}(D_j)} \tag{3.2}$$

where $1 \leq i \leq M$, $1 \leq j \leq N$, and Area(A) represents the area of A. The measures in the above equation constitute two matrices $\Sigma = (\sigma_{ij})$ and $T = (\tau_{ij})$. Notice that σ_{ij} indicates how much portion of G_i is occupied by D_j, and τ_{ij} indicates how much portion of D_j is occupied by G_i. Our strategy of performance evaluation is to analyze these matrices to determine the correspondence between two sets of polygonal areas:

- one-to-one match ($\sigma_{ij} \approx 1$ and $\tau_{ij} \approx 1$);
- one-to-zero match ($\sigma_{ij} \approx 0$ for all $1 \leq j \leq N$);
- zero-to-one match ($\tau_{ij} \approx 0$ for all $1 \leq i \leq M$);
- one-to-many match ($\sigma_{ij} < 1$ for all j, and $\sum_{j=1}^{N} \sigma_{ij} \approx 1$);
- many-to-one match ($\tau_{ij} < 1$ for all i, and $\sum_{i=1}^{M} \tau_{ij} \approx 1$);
- many-to-many match (others).

An example of matching between a set of ground truth entities and the detected entities is illustrated in Figure 3.1. By computing their area overlap, we construct two matrices, $\Sigma = (\sigma_{ij})$ and $T = (\tau_{ij})$, shown in Table 3.1. In this example, we find a one-to-one match (G_1 to D_1), a one-to-many match (G_2 to D_2 and D_3), a one-to-zero match (G_3 to nothing), and a many-to-many match (G_4, G_5 and G_6 to D_4 and D_5).

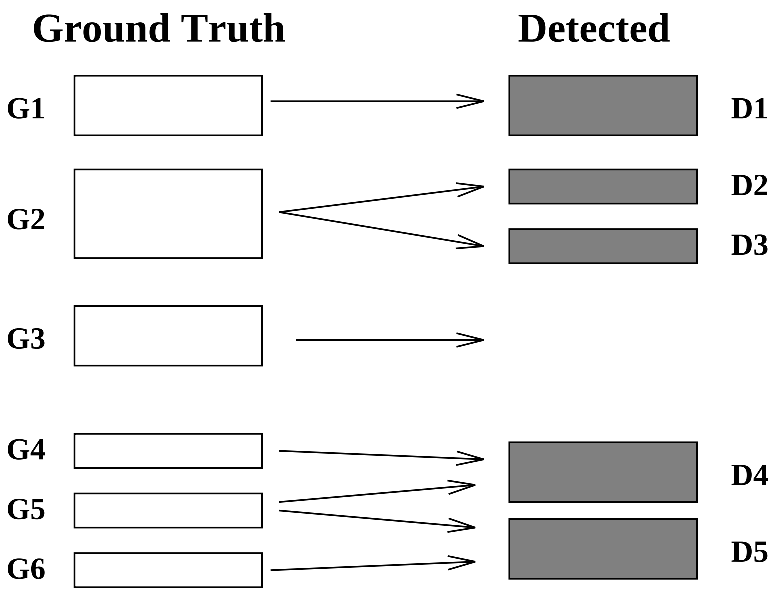


Figure 3.1: Illustrates correspondence between ground truth and detected structures.

3.1.2 Performance measure of layout analysis

Once the matching between detected structures and ground-truth structures is established, a performance measure can be computed. A one-to-one match means an object G_i is correctly identified by the segmentation process as D_j. A one-to-zero match is the case when a certain object G_i is not detected by the segmentation (misdetection), and vise versa for the zero-to-one match (false alarm). If an entity G_i matches to a number of detected entities, we call it a splitting detection. It is a merging detection when two or more objects in $\mathcal{G}$ are identified as an object D_j. The many-to-many matches are called spurious detections.

Let us denote the probability of a matching ($G^i \subset G$ is identified as $D^i \subset D$ in the sample) as

$$P^i(G, D) = \frac{\#G^i + \#D^i}{\#G + \#D}. \tag{3.3}$$

Then, the performance measure of a layout analysis process is defined as

$$f_{layout} = \sum_{i \in M} W^i P^i(G, D) \tag{3.4}$$

Table 3.1: Illustrates the area overlap matrices computed from the example in Figure 3.1.

	D_1	D_2	D_3	D_4	D_5
G_1	0.95				
G_2		0.45	0.51		
G_3					
G_4				0.7	
G_5				0.37	0.29
G_6				0.8	

$\Sigma = (\sigma_{ij})$

	D_1	D_2	D_3	D_4	D_5
G_1	0.9				
G_2		0.85	0.91		
G_3					
G_4				0.4	
G_5				0.25	0.3
G_6					0.45

$T = (\tau_{ij})$

where $M = \{correct, miss, false, merging, splitting, spurious\}$ is the set of possible matching, and W^i is the weight assigned to each type of matching.

For the example shown in Figure 3.1, we determine that G_1 is correctly detected; G_2 is split into D_2 and D_3, G_3 is missed; and G_4, G_5 and G_6 are merged and split into D_4 and D_5. In the experiment, we choose the weights as in Table 3.2. Therefore, the performance measure for the example is

$$f_{layout} = \frac{0.0 \times (1+1) + 0.5 \times (1+2) + 1.0 \times 1 + 1.0 \times (3+2)}{6+5} = 0.68.$$

Table 3.2: Weights used for computing the performance measure of layout analysis.

Correct	Merge	Split	Miss	False	Spurious
0.0	0.5	0.5	1.0	1.0	1.0

Each kind of matching can be further divided into different categories. For example, merging text across columns apparently should be given more penalty than merging two adjacent text blocks within the same column. A many-to-one matching between ground-truth zones and detected zones can be divided into the following possible detection errors:

- merge text and non-text
- merge text across columns
- merge text zones with different reading directions
- merge adjacent homogeneous text.

Equation 3.4 still works and M now is the set of categories defined by the users.

3.1.3 Performance measure of classification

The classification module classifies each extracted structure into one of the predefined categories. Labeling of the physical and functional types, detection of format attributes, and isolated character recognition are typical classification problems. The output of the classification process is compared with the labels from the ground-truth in order to evaluate the performance of the algorithm. Let $P(t,a)$ be the probability of observing a unit, in the sample, whose true category is t, and whose assigned category is a. The expected performance is

$$f_{label} = \sum_{t\in\Theta}\sum_{a\in\Theta} W(t,a)P(t,a), \tag{3.5}$$

where Θ is the set of categories, and $W(t,a)$ is the weight associated with different assignment. A contingency table is computed to indicate the number of entities of a particular class label that are identified as members of another class. The misclassification rate is defined as,

$$f_{mis-classification} = \sum_{t\in\Theta}\sum_{a\in\Theta,\ a\neq t} P(t,a). \tag{3.6}$$

An example of contingency table showing the classification results of text and non-text zones is shown in Table 3.3. We denote by $P(n,t)$ the fractions of entities in the sample that are truly non-text but are assigned text, and so on.

Table 3.3: An example of contingency table showing the classification results of text and non-text zones. Row represents the true categories and column represents the assigned categories.

	Text	Non-text
Text	P(t, t)	P(t, n)
Non-text	P(n, t)	P(n, n)

The mis-classification rate is the sum of $P(t,n)$ and $P(n,t)$,

$$f_{mis-classification} = P(n,t) + P(t,n).$$

The misdetection rate of text entity is computed as:

$$P(n|t) = \frac{P(t,n)}{P(t,t) + P(t,n)}.$$

And the false-alarm rate of text entity is computed as:

$$P(t|n) = \frac{P(n,t)}{P(n,t) + P(n,n)}.$$

3.1.4 Performance measure of reading order detection

Let $\mathcal{A} = \{A_1, \cdots, A_K\}$ be the set of structures that have been correctly identified. The detected reading order $\hat{R}$ is a tuple $(\hat{r_1}, \cdots, \hat{r_K})$ which is a permutation of true reading order $R = (r_1, \cdots, r_K)$. The problem of evaluating reading order determination algorithm can be reduced to computing the minimum number of moves required to obtain $\hat{R}$ from R. The problem can be modeled as a sorting problem where a string of K integers ordered in a random manner, must be sorted in ascending (or descending) order [35].

3.1.5 Performance measure of text recognition

Let $T = <T_1, T_2, \cdots, T_x>$ be the true symbol strings. And the output of the recognition algorithm is denoted by $O = <O_1, O_2, \cdots, O_y>$. The problem of evaluating the text recognition (OCR) algorithm can be reduced to computing the *edit distance* between T and O: the minimum number of operations (insertions, deletions, and substitutions) required to transform one string into the other. The edit distance may be computed efficiently using dynamic programming [37] [40].

The OCR Performance Evaluation (OPE) Software [26] [27] compares the groundtruth and OCR output and outputs a file containing: single character contingency table, error substring contingency table, statistics of line insertions, deletions and substitutions, and statistics of symbol insertions, deletions and substitutions.

3.1.6 Performance metrics for graphics recognition

Driven by the need to convert a large number of hard copy engineering drawings into CAD files, raster to vector conversion has been a field of intense research for the last three decades. In addition to research prototypes in several academic and industrial research centers, several commercial software products are currently available to assist users in converting raster images to vector (CAD) files. However, the process of selecting the right software for a given vectorization task is still a difficult one. Although trade magazines have published surveys of the functionality and ease of use of vectorization products [31], a scientific, well designed, comparison of the autovectorization capability of the products is not available. Responding to this need, two graphics recognition competitions were held recently[32, 33].

We developed a protocol [34] to evaluate the performance of engineering diagram recognition systems on images that contain engineering drawings of straight lines (solid or dashed), circles (solid or dashed), partial arcs of circles (solid or dashed), as well as, bounding boxes of text blocks within the images. This protocol was used in the competition of graphics recognition during the Second IAPR Workshop on Graphics Recognition [33].

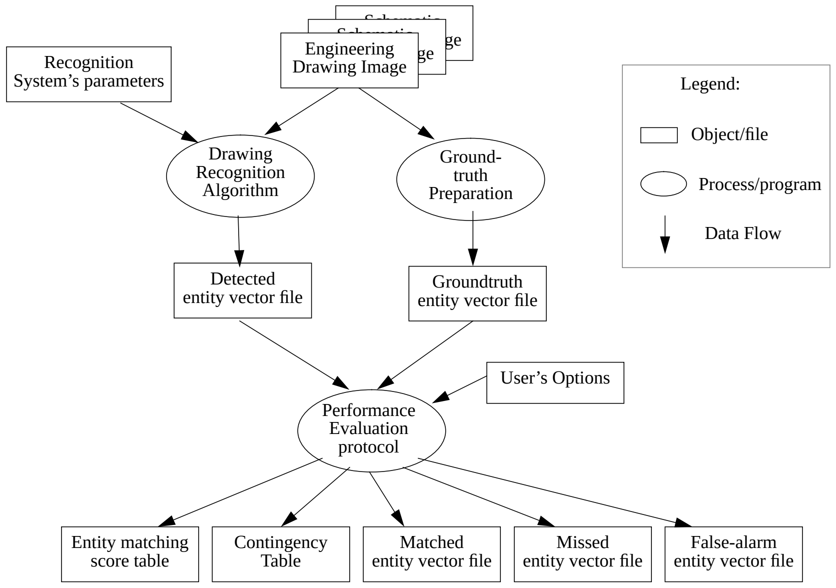

Recognition System's parameters
Engineering Drawing Image
Legend:
Object/file
Process/program
Data Flow
Drawing Recognition Algorithm
Ground-truth Preparation
Detected entity vector file
Groundtruth entity vector file
User's Options
Performance Evaluation protocol
Entity matching score table
Contingency Table
Matched entity vector file
Missed entity vector file
False-alarm entity vector file

Figure 3.2: The object-process diagram of graphics recognition evaluator.

Inputs to the evaluator are entities of the recognition algorithm's output and the corresponding ground truth. (Figure 3.2 shows an object-process diagram of our evaluator.) Since there are seven types of entities (solid and dashed lines, solid and dashed arcs, solid and dashed circles, and text areas) that are allowed, the evaluation protocol and the matching criteria are designed differently for the combinations. The matching scores for each pair is computed according to the pair's entity type combination, using the matching criteria defined for this combination. A match-score table is produced from the matching score computation.

From the computed match-score table, we search for all the one-to-one matches, resolving the problems of the one-to-many matches and the many-to-one matches, as well as, those false-alarms and mis-detections. A contingency table is produced. The contents of these tables are computed facts which when weighted by application specific weights can be summed to produce an overall score relevant to the application.

3.2 UW Document Image Databases

It is clear that much of the early work on document analysis system provided illustrative results and hardly any had their techniques tested on significant sized data sets and to measure quantitative performance [1]. The main reasons are the lack of accurate document ground truth to train and test the algorithms and the lack of appropriate and quantitative performance metrics and evaluation protocol.

A large quantity of ground-truth data, varying in quality, is required in order to give an accurate measurement of the performance of an algorithm under different conditions. Sponsored by the Advanced Research Project Agency (ARPA), we have produced the University of Washington Document Image Database [42, 43] series of ground-truth databases on CDROM for the document image analysis and recognition communities. The first two databases in the series, UW-I and UW-II, provide a variety of ground-truth information about their constituent documents, including zone and page bounding boxes, attributes, and ASCII text. In order to facilitate the training and evaluation of document layout analysis algorithms, UW-III [44] includes text line bounding box and word bounding box ground-truth, in addition to the ground-truth information provided in the previous database releases. The directory structure of the UW-III document image database is shown in Figure 3.3. Ground truth for non-text structures are also provided in the database, such as chemical formulae, mathematical formulae, and engineering line drawings. The following is the UW-III content summary:

- 25 TIFF images of journal document pages with chemical formulae and their ground truth in Xfig. Besides ground truth files, there is a page attribute file, a page condition file, a page box file, and zone box file associated with each document page.

- 25 TIFF images of journal document pages with mathematical formulae and their ground truth in Xfig and LaTeX format. Besides ground truth files, there is a page attribute file, a page condition file, a page box file, and zone box file associated with each document page.

- 40 TIFF images of journal document pages with Line-drawings and their ground truth in AutoCAD and IGES format. Besides ground truth files, there is a page attribute file, a page condition file, a page box file, and zone box file associated with each document page.

- 44 synthetic TIFF images of engineering-drawings and their ground truth in AutoCAD format.

- 33 TIFF images of technical articles generated from LaTeX and their character-level ground truth.

- 33 TIFF images of the printed and scanned article pages from the above LaTeX TIFF images and their character-level ground truth.

- 1600 skew corrected TIFF images from UW-I and UW-II, and for each of the pages, there are text-word bounding boxes, text line bounding boxes, text-zone bounding boxes.

UW-III contains a total of 1600 English document images randomly selected, copied, and scanned from scientific and technical journals. Each page contains a hierarchy of manually verified page, zone, text line, and word entities with bounding box and in the correct reading order. The text ground-truth and zone attributes are tagged to each zone entity. The zone attribute includes information about the type of the zone (i.e., text, figure, table, etc.), the dominant font size, the dominant font style, and the justification, etc. The ground-truthed information for each page is listed in Table 3.4.

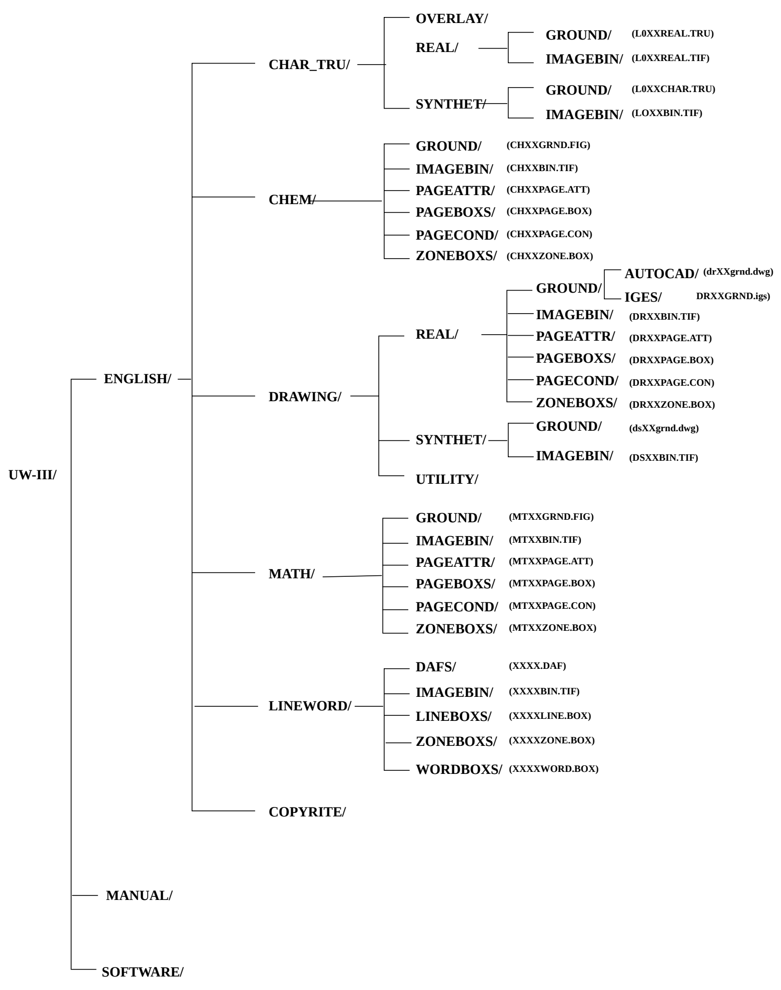


Figure 3.3: UW-III Document Image Database's directory structure

Table 3.4: Ground-truth information provided for each page in the UW-III document image database.

Structure	Ground-Truth
Layout	Hierarchy of bounding boxes enclosing page, header, footer, text and notext zones, text lines, and words for each page.
Logical	Label of content type for each zone. Reading order of text and nontext zones, text lines, and words.
Content	Text content for each text zone
Style	Page and zone format attributes

This database can be utilized by the document understanding community as a common platform to develop, test and evaluate their systems. Based on the ground-truth data, we can evaluate the performance of document analysis algorithms and build statistical models to characterize various types of document structures. For each structure that we use to describe a document, there is an associated metric that measures the difference between a structure that is automatically produced and the ground-truth structure. Figure 3.4 illustrates the process of evaluating document analysis algorithms using the UW document image database.

Document Format Attribute Specification (DAFS) [46] is used as the representation of document structure in the UW-III database. The DAFS provides a format for breaking down documents into standardized entities, defining entity boundaries and attributes, and labeling their contents (text values) and attribute values. DAFS permits the creation of parent, child and sibling relationships between entities, providing easy representation of the hierarchical structures of a document. DAFS Library [47]

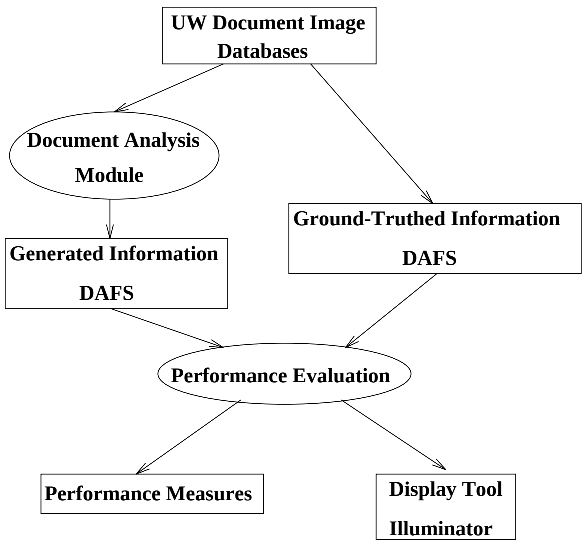


Figure 3.4: Illustrates the process of evaluating document analysis algorithms using the UW document image database.

is a set of routines for developers to work with DAFS files. A graphical user interface called Illuminator [48] enables the users to view and edit the DAFS structure. Both DAFS Lib and Illuminator are public domain software. Through DAFS, developers are able to exchange training and test documents and work together on shared problems.

The protocol used for ground-truthing layout structure and other information on the document images is illustrated in Figure 3.5. We used a double entry and triple verification procedure. First, independent ground-truth layout structures are generated either manually (text and non-text blocks) or by different algorithms (text lines and words). Then, two ground-truth files are compared and the errors are corrected until they agree with each other. This verification is done independently by two individuals. Finally, the verified pairs are compared and corrected by third group.

Several automatic tools have been developed in order to efficiently ground-truth

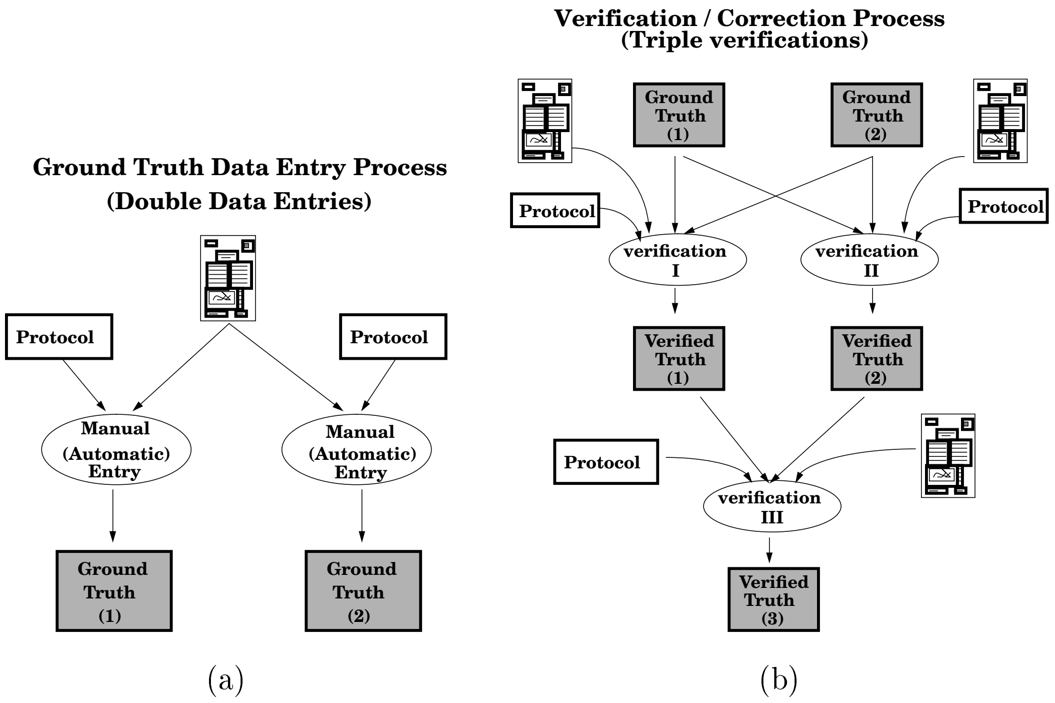


Figure 3.5: Document structure ground-truthing protocol: (a) generation of ground-truth, (b) verification and correction.

a large number of document images with very high accuracy. One compares two ground-truth files and produces common and different boxes between them. The matching criterion is described in Section 3.1.1. An erroneous word bounding box flagging algorithm makes use of the database's existing ground-truth information to detect errors in the word bounding boxes [45].

3.3 Methodology for Performance Evaluation

Performance evaluation serves two purposes. One is comparing the ground-truth and algorithm output and producing a quantitative overall measure of the algorithm. Another one is to help the developers to analyze the results and their algorithms.

In Section 3.3.1, we describe a method for determining the optimal algorithm tuning parameters given the ground-truth. Section 3.3.2 describes a method that checks the parametric models by comparing the simulated values drawn from the posterior distribution to the observed data.

3.3.1 Systematic parameter selection of rule-based algorithms

In document analysis algorithms, tuning parameters (thresholds) are often used. These parameters have been traditionally chosen by trial-and-error. In this section, we describe a systematic parameter estimation method by optimizing the algorithm performance over the training data. The parameter estimation problem can be stated as follows, *Given an algorithm with parameter* $X = (X_1, X_1, \cdots, X_N)^T$*, search the parameter vector* X *that minimizes expected performance* $f(X)$*.* This process is equivalent of fitting the algorithm's underlying model to the data. Here we describe an engineering method based on the given criterion function, instead of the traditional trial-and-error approach.

When a representative sample of a domain is available, and a quantitative performance metric is defined, we can tune the parameter values of an algorithm and select a set which produces the optimal performance on the input population. Suppose the performance measure f is the cost, then the best tuning parameter vector is X^* after

$$X^* = \arg\min_X f(X) \tag{3.7}$$

and $X = (X_1, X_2, \cdots, X_N)^T$ is the vector of parameters.

We use an iterative first-order search method to find the optimal parameters,

$$X^q = X^{q-1} + k_q^* S^q. \tag{3.8}$$

Here q is the iteration number, S^q is the search direction, and k_q^* is a scalar multiplier determining the amount of change in X for the iteration. The search direction is taken as the negative of the gradient of the function,

$$S^q = -\nabla f(X). \tag{3.9}$$

Algorithm for the first-order search method is shown in Algorithm 3.1.

Algorithm 3.1 *The gradient descent search method*

```
1. Choose $X^0$. $X \leftarrow X^0$.
2. Do
    (a) $S \leftarrow -\nabla F(X)$.
    (b) Find $\alpha^*$ to min$F(X + \alpha^* S)$.
    (c) $X \leftarrow X + \alpha^* S$.
    (d) If converged, exit.
3. Return $X$.
```

To compute the gradient of the function, we consider a first-order Taylor series expansion of $f(X)$ about X^0

$$f(X) \simeq f(X^0) + \frac{\partial f(X^0)}{\partial X} \Delta X,$$

where

$$\Delta X = X - X^0.$$

By changing only one parameter X_i by ΔX_i ,

$$\Delta X = (0, 0, \cdots, \Delta X_i, \cdots, 0, 0)^T,$$

we have,

$$f(X) \simeq f(X^0) + \frac{\partial f(X^0)}{\partial X}_i \Delta X_i.$$

This leads to

$$\frac{\partial f(X^0)}{\partial X}_i = \frac{f(X) - f(X^0)}{\Delta X_i}.$$

By tuning N parameters one by one and obtaining the corresponding performance, the gradient function $\nabla f(x)$ is,

$$\frac{\partial f(X^0)}{\partial X} = (\frac{\partial f(X^0)}{\partial X}_1, \cdots, \frac{\partial f(X^0)}{\partial X}_N).$$

After finding the search direction, we use the golden section algorithm to find the best scalar k. The golden section method for estimating the minimum or maximum of a one-variable function is a popular technique. Let F be a unimodal function of the variable X. The procedure for iteratively applying the golden section method to determine the minimum of F is given in Algorithm 3.2. We assume that the lower bound X_l and the upper bound X_u on X are known which bracket the minimum. We also assume that the function has been evaluated at each of these bounds so we know the corresponding values F_l and F_u. Let N be the number of iterations.

Algorithm 3.2 *Golden section algorithm for the unconstrained minimum.*

```
Given $X_l$, $F_l$, $X_u$, $F_u$, $N$, $K = 1$, and $\tau = 0.381966$.

  1. Update $X_1 = (1 - \tau)X_l + \tau X_u$. Evaluate $F_1 = F(X_1)$.

  2. Update $X_2 = \tau X_l + (1 - \tau)X_u$. Evaluate $F_2 = F(X_2)$.

  3. While $K < N$, Do

       • If $F_1 > F_2$, Then

           (a) $X_l = X_1$ and $F_l = F_1$

           (b) $X_1 = X_2$ and $F_1 = F_2$

           (c) Update $X_2 = \tau X_l + (1 - \tau)X_u$. Evaluate $F_2 = F(X_2)$

       • Else, Then
```

(a) $X_u = X_2$ and $F_u = F_2$

(b) $X_2 = X_1$ and $F_2 = F_1$

(c) Update $X_1 = (1-\tau)X_l + \tau X_u$. Evaluate $F_1 = F(X_1)$

$K = K + 1$

4. Exit

In searching for the value of X for which F is a minimum, it is necessary to begin by finding a lower bound X_l and an upper bound X_u which bracket the solution. We assume that the solution lies in the positive X domain and that the function is unimodal. Beginning with an initial lower bound X_l and a proposed upper bound X_u, we evaluate the functions F_l and F_u, respectively. If $F_u \geq F_l$, then X_u is obviously an upper bound, assuming the function has a negative slope at X_l. Because the solution is assumed to lie in the positive X domain, it follows that X_l is the lower bound and the solution is complete. If F_u is less than F_l, we must iteratively applying an updating formula to achieve the desired bounds,

$$\begin{aligned} X_u^{new} &= (1+a)X_u^{old} - aX_l^{old} \\ X_l^{new} &= x_u^{old}, \end{aligned} \tag{3.10}$$

with the constant $a = (1+\sqrt{5})/2$ [41].

3.3.2 Parametric model checking by comparing data to the posterior predictive distribution

A good statistical analysis should include at least some check of the adequacy of the fit of the model to the data and the plausibility of the model for the purpose for which the model will be used. One basic technique for checking the fit of a model to data is to draw simulated values from the posterior distribution of replicated data and compare these samples to the observed data [51]. Any systematic differences between the simulations and the data indicate potential failings of the model.

Let y be the observed data and θ be the vector of parameters. Let y^{rep} be the replicated data that could have been observed. The posterior predictive distribution is,

$$p(y^{rep}|y) = \int p(y^{rep}|\theta)p(\theta|y)d\theta. \tag{3.11}$$

We measure the discrepancy between model and data by defining test quantities, the aspects of the data we wish to check. A test quantity, or discrepancy measure, $T(y,\theta)$, is a scalar summary of parameters and data that is used as a standard when comparing data to predictive simulations.

Lack of fit of the data with respect to the posterior predictive distribution can be measured by the tail-area probability, or $p-$value, of the test quantity, and computed using posterior simulations of (θ, y^{rep}). The $p-$value is defined as the probability that the replicated data could be more extreme than the observed data, as measured by the test quantity:

$$p - value = Pr(T(y^{rep},\theta) \geq T(y,\theta)|y). \tag{3.12}$$

In practice, we usually compute the posterior predictive distribution using the simulation. If we already have L simulations from the posterior density of θ, we just draw one y^{rep} from the predictive distribution for each simulated θ; we now have L draws from the joint posterior distribution, $p(y^{rep},\theta|y)$. The posterior predictive check is the comparison between the realized test quantities, $T(y,\theta^l)$, and the predictive test quantities, $T(y^{repl},\theta^l)$. The estimated $p-$value is just the proportion of these L simulations for which the test quantity equals or exceeds its realized value; that is, for which $T(y^{repl},\theta^l) \leq T(y,\theta^l), l = 1,\cdots,L$. A model is suspect if the tail-area probability for some meaningful test quantity is close to 0 or 1.

3.4 Experimental Results

3.4.1 Performance evaluation of rule-based document layout analysis algorithms

In this section, we briefly present a rule-based algorithm that extracts document layout structure using the bounding boxes of different entities [15]. Then, we report the performance of each module on the images from the UW-III Document Image Database.

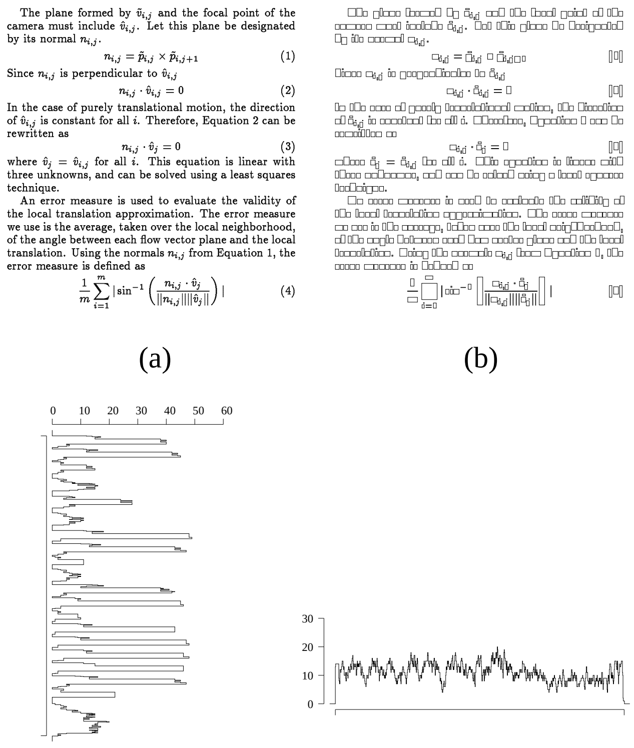
Algorithm description

The input of the algorithm is a set of bounding boxes that enclose the connected components on a binary document image. Figure 3.4.1(a) shows a segment of a document image and Figure 3.4.1(b) shows the bounding boxes produced in this step.

The document page segmentation roughly divides the image into a list of zones by analyzing the spatial configuration of bounding boxes. The analysis can be done by projecting bounding boxes onto vertical and horizontal lines [11]. A projection profile is a frequency distribution of the bounding boxes on the projection line (Figure 3.4.1). Figures 3.4.1(c) and 3.4.1(d) show the horizontal and vertical projection profiles of the bounding boxes in Figure 3.4.1(b).

A document image can be segmented using a recursive X–Y cut [38] procedure based on the bounding boxes. At each step, the horizontal and vertical projection profiles are calculated. Then a zone division is performed at the most prominent valley in either projection profile. The process is repeated recursively until no sufficiently wide valley is left. The free parameters used in the page segmentation algorithm are as follows:

- the minimum connected component width and height: w_1 and h_1.
- the maximum connected component width and height: w_2 and h_2.



The plane formed by $\tilde{v}_{i,j}$ and the focal point of the camera must include $\hat{v}_{i,j}$. Let this plane be designated by its normal $n_{i,j}$.

$$n_{i,j} = \tilde{p}_{i,j} \times \tilde{p}_{i,j+1} \tag{1}$$

Since $n_{i,j}$ is perpendicular to $\hat{v}_{i,j}$

$$n_{i,j} \cdot \hat{v}_{i,j} = 0 \tag{2}$$

In the case of purely translational motion, the direction of $\hat{v}_{i,j}$ is constant for all i. Therefore, Equation 2 can be rewritten as

$$n_{i,j} \cdot \hat{v}_j = 0 \tag{3}$$

where $\hat{v}_j = \hat{v}_{i,j}$ for all i. This equation is linear with three unknowns, and can be solved using a least squares technique.

An error measure is used to evaluate the validity of the local translation approximation. The error measure we use is the average, taken over the local neighborhood, of the angle between each flow vector plane and the local translation. Using the normals $n_{i,j}$ from Equation 1, the error measure is defined as

$$\frac{1}{m}\sum_{i=1}^{m} |\sin^{-1}\left(\frac{n_{i,j} \cdot \hat{v}_j}{||n_{i,j}||||\hat{v}_j||}\right)| \tag{4}$$

Figure 3.6: Illustrates (a) an English document, (b) bounding boxes of connected components of black pixels, (c) horizontal projection profile, (d) vertical projection profile.

- the maximum height/width aspect ratio of connected component: r_1.

  The above thresholds decide if a connected component will be considered for the projection.

- the minimum width of the valley in the vertical projection profile, w_3, and in the horizontal profile, w_4.

  If w_3 or w_4 is smaller than a threshold, no cut will be applied.

- the maximum height/width aspect ratio of the current region: r_2.

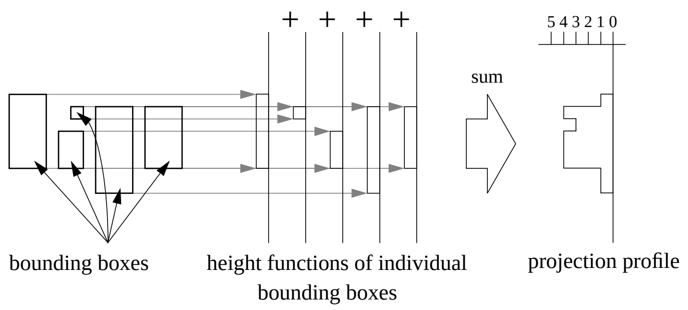


Figure 3.7: Illustrates a horizontal projection profile, which is obtained by accumulating the bounding boxes onto the vertical line. The projection profile is a histogram which shows frequencies of projected bounding boxes.

If r_2 is larger than a threshold, no further cut will be applied.

Inside each zone generated from the page segmentation process, we analyze the spatial configuration of bounding boxes to extract text lines. From Figure 3.4.1(c) and 3.4.1(d), it is clear that the text lines can be extracted by finding the distinct high peaks and deep valleys at somewhat regular intervals in the horizontal projection profile. The free parameters used in this text line extraction algorithm are lised as follows:

- the minimum height of text line: h_3.
- the maximum height of text line: h_4.
- the threshold for detecting rule: r_3.

  If a detected text line contains only one component, and the text line's aspect ratio is larger than r_3, the text line is considered as a rule.
- the minimum size of connected components accepted for projection: w_5 and h_5.

- the maximum size of connected components accepted for projection: w_6 and h_6.
- the maximum aspect ratio of connected components accepted for projection: r_4.

For all textual zones, we merge text lines into text blocks. The beginning of a text block, such as a paragraph, math zone, section heading, etc., are usually marked either by changing the justification of the current text line, or by putting extra space between two text lines, or by changing font size or font style. So when a significant change in inter-text line spacings or justification occurs, we say that a new text block begins. The free parameters used in this algorithm are:

- Justification threshold: t_1.
- Inter-line spacing threshold: t_2.

Parameter tuning

The parameters of each algorithm are decided using the optimization process described in Section 3.3.1. For the segmentation process, the weights we use for computing the performance is given in Table 3.2. Therefore, our performance criterion is the cost for converting the detected layout structure to the ground-truth.

For each algorithm, we start with their default parameters obtained by observing a small number of images, then we tune the parameters by minimizing the algorithm's cost on the images in the UW-III database, until the improvement of performance is less than a threshold. The algorithms' cost after the parameter tuning, compared to their performance using the default parameters is shown in Table 3.5. The average improvement is 17.6% with respect to the original performance.

The numbers and percentages of miss, false, correct, splitting, merging and spurious detections of each algorithm after the optimization process are presented in the following sections. The parameter values before and after the optimization process are also given.

Table 3.5: Illustrates the performance of algorithms before and after the optimization process.

	Page	Line	Block
original	0.143	0.022	0.141
optimized	0.104	0.017	0.137
improved by	27.3%	22.7%	2.8%

Performance of page segmentation

Given a set of connected components computed from a document image, page segmentation partitions the components into a set of zones, such that each zone is homogeneous (text zones with the same read order, and non-text zones). The parameters before and after the optimization are given in Table 3.6. Table 3.7 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground-truth zones as well as the algorithm output. Since the page segmentation finds the coarse homogeneous zones, we do not consider the merging of adjacent text within the same column as error.

Table 3.6: The default and optimized parameters for the page segmentation.

	w_1	h_1	w_2	h_2	r_1	r_2	w_3	w_4
default	3	3	1800	2200	50	5	30	30
optimized	3.4	3.3	2615	2187	52.6	3.1	39.1	34.4

This algorithm is restricted to bi-directional X-Y cuttable layouts. It is also sensitive to severe page skew. So deskew has to be done before applying the projection. To decide if applying a cut on a projection profile valley, a threshold on the width

Table 3.7: Performance of the X-Y cut page segmentation with respect to the ground-truth and the algorithm output.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	24216	21019 (86.80%)	462 (1.91%)	2186 (9.03%)	245 (1.01%)	304 (1.25%)
Detected	14848	11346 (76.41%)	1883 (12.68%)	710 (4.78%)	592 (3.99%)	317 (2.14%)

of valley is used. Instead of having a global value, the threshold might be adaptively determined by considering the width and depth of the projection profile valley, the size of nearby connected components, and the aspect ratio of current zone, etc.

Performance of text line segmentation

Given a set of homogeneous text zones, and the connected components enclosed by each zone, the text line segmentation partitions the components into a set of text lines. In this experiment, the ground truth text zones are used as the input. The parameters before and after the optimization are given in Table 3.8. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections of the text line extraction algorithm are shown in Table 3.9.

Table 3.8: The default and optimized parameters for the text line segmentation.

	h_3	h_4	r_3	w_5	h_5	r_4
default	10	500	5	3	3	50
optimized	8.4	1000	7.4	41	20	44.3

Table 3.9: Performance of the text line extraction algorithm

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	105439	100471 (95.39%)	124 (0.12%)	4543 (4.31%)	157 (0.15%)	33 (0.03%)
Detected	102494	100471 (98.03%)	383 (0.37%)	1504 (1.47%)	4 (0.00%)	132 (0.13%)

The advantages of this algorithm are that it is very simple and fast, and it produces very low misdetection rate. Table 3.9 shows that most of detection errors are merging text lines. This algorithm is sensitive to skew (global or local), smear, warping, and noise. If the inter-text line spacing is very small and the superscript, subscript, or the ascender and descender of adjacent lines overlap or touch with each other, the text lines are usually merged.

Performance of text block extraction

Given a set of text lines within the same column, text block extraction partitions the text lines into a set of text blocks. In this experiment, the ground truth text lines are used as the input and the column information is assumed to be known. The default and optimized parameters are given in Table 3.10. The performance of text block extraction algorithm is shown in Table 3.11. Its total cost is 13.76%.

Table 3.10: The default and optimized parameters for the text block extraction.

	t_1	t_2
default	20	30
optimized	17.8	26.2

Table 3.11: Performance of text block extraction with respect to the ground-truth and the algorithm output.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	21738	16680 (76.73%)	1670 (7.68%)	3014 (13.87%)	2 (0.01%)	372 (1.71%)
Detected	23302	16680 (71.58%)	5191 (22.28%)	1094 (4.69%)	0 (0.00%)	337 (1.45%)

This algorithm utilizes the paragraph formatting attributes as the cues for text block segmentation. Instead of the global threshold for the inter-line spacing, and the justification, a local threshold can be adaptively determined based on the text font size within a homogeneous region.

3.4.2 Model checking of a parametric document layout model

The model checking technique (3.3.2) is used to check the fit of a parametric document layout model. The model and the corresponding test quantities are described in this section. The experimental results are also presented.

Probabilistic linear displacement model

The text lines and text blocks on a document image are usually laid out in a very structured manner - having preferred spatial structures. A spatial model called the probabilistic linear displacement model (PLDM) was developed by Chen *et. al.* [53].

A text line is viewed as a string of equally spaced collinear words, denoted by $\mathcal{B} = \{B_1, B_2, \cdots, B_M\}$. Let L denote a baseline of these words, which is represented by the equation:

$$x \sin\varphi + y \cos\varphi - \rho = 0 \tag{3.13}$$

where φ denote the orientation of the baseline and ρ be the distance of the baseline to the origin of the coordinate system. Let $\epsilon_i = \epsilon(B_i, L)$ define a distance function of B_i to the baseline L and $i = 1, 2, \cdots, M$. Let $\delta_i = \delta(B_i, B_{i+1})$ define a distance function between B_i and B_{i+1} along the baseline L (also defined as the displacement), where $i = 1, 2, \cdots, M-1$. Figure 3.8 shows an example of the model. ϵ_i $(i = 1, 2, \cdots, M)$

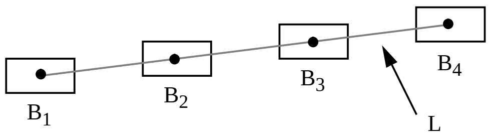


Figure 3.8: illustrates an example of the linear displacement model. The geometric centroids of B_1, B_2, B_3, B_4 are collinear and equally spaced along the baseline L.

and δ_i $(i = 1, 2, \cdots, M-1)$ are assumed to be independent random variables given that they share the same baseline (ρ, φ), where ϵ_i and δ_i are assumed to follow a Normal distribution:

$$\epsilon_i \sim Normal(0, \sigma_\epsilon) \quad \delta_i \sim Normal(\mu_\delta, \sigma_\delta)$$

We assume the following non-informative prior specification for the hyperparameters

$$\sigma_\epsilon \sim Gamma(1.0E-3, 1.0E-3),$$
$$\sigma_\delta \sim Gamma(1.0E-3, 1.0E-3).$$

And μ_δ is assumed to come from a Gaussian distribution,

$$\mu_\delta \sim Normal(\mu, \sigma)$$

where μ and σ are known constants.

A text block is characterized as a sequence of equally spaced text lines. Figure 3.9 illustrates four homogeneous types of text block alignment. The left, center, or right vertical edges of the text line bounding boxes can be simply modeled by the PLDM.

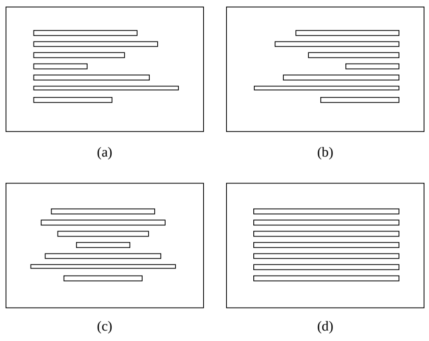


Figure 3.9: Illustrates four homogeneous types of text block alignment: (a) left justified, (b) right justified, (c) center justified, and (d) justified.

Let $< L_1, L_2, \cdots, L_n >$ be a sequence of n consecutive text lines that come from a text block P. In Figure 3.10, let h_i and w_i denote the height and width of the text line L_i respectively, where $i = 1, 2, \cdots, n + 1$. Let s_i denote the line spacing

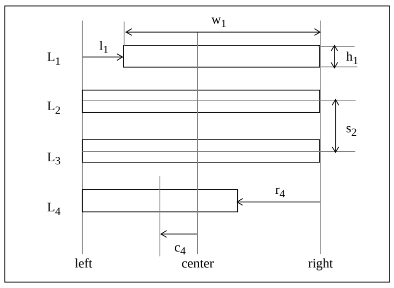


Figure 3.10: Illustrates text line features within a text block.

between two consecutive text lines L_i and L_{i+1}, which is the gap distance between the center horizontal edges of the two lines, where $i = 1, 2, \cdots, n$. Let l_i, c_i and r_i denote correspondingly the distances of the left, center and right vertical edges of L_i to the left, center and right vertical edges of the text block P (denoted by $\mathcal{E}_l$, $\mathcal{E}_c$ and $\mathcal{E}_r$ respectively), where $i = 1, 2, \cdots, n + 1$.

It is assumed that the text line height and the text line spacing are identically and independently distributed Normal

$$h_i \sim Normal(\mu_h, \sigma_h), \quad s_i \sim Normal(\mu_s, \sigma_s),$$

where μ_h and μ_s have a normal prior probability distribution.

The displacement of text line edge from the text block edge is also assumed to follow a Normal distribution:

$$Leftjustified : l_i \sim Normal(0, \sigma_\epsilon)$$

$$Rightjustified : r_i \sim Normal(0, \sigma_\epsilon)$$

$$Centerjustified : c_i \sim Normal(0, \sigma_\epsilon)$$

$$Bothjustified : c_i \sim Normal(0, \sigma_\epsilon).$$

In addition, the text line width of both right and left justified text is assumed to be normal distributed, while the the text line width of left, right and center justified text is assumed to follow a uniform distribution.

We consider two different statistics for $T(y)$ which may be sensitive to outlying observations in a Normal model. These are the coefficients of skewness and kurtosis, which are estimated by the average value of the third and fourth moments about the mean divided by the third and fourth powers of the standard deviation respectively [52]. For samples from a Normal distribution, the skewness coefficient has expectation zero, while the kurtosis coefficients has expectation 3. Positive skewness indicates an extended right tail, while negative skewness implies an extended left tail. A kurtosis coefficient > 3 indicates an excess of values near the mean and far from it, with a corresponding depletion elsewhere: this is the manner in which the $t-$distribution departs from the Normal. Values < 3 result from distributions with a flatter top than Normal.

We perform a posterior test on the independence between two variables x and y using the correlation coefficient as the testing quantity,

$$T = \frac{Covariance(X, Y)}{\sigma_x \sigma_y}. \tag{3.14}$$

Results of checking the document layout structure model

The model checking criteria described in Section 3.4.2 are implemented in BUGS [52] (Bayesian inference Using Gibbs Sampling). BUGS is a program that carries out Bayesian inference on statistical problems using a simulation technique known as Gibbs sampling.

Several dozen images are selected from the UW-III database as the testing set for the experiment. The document images in the UW-III data set have been deskewed and the bounding boxes of text lines are manually ground truthed and verified. Therefore, the coordinates ϕ and ρ are given for each text line.

The results for each of the model checks based on a 1000 iteration BUGS run are given in Table 3.12, 3.13 and 3.14, respectively.

The experiment result for checking the word space and displacement model suggests that the normal distribution assumption on word vertical displacement is inadequate. The discrepancy on independence between observed and replicated data cannot easily explained by chance.

The assumption of normal distribution on the text line height and space cannot be rejected.

As expected, there is no evidence against the normal model on the distance between text line edge and text block edge.

Performance of text line extraction

The numbers and percentages of miss, false, correct, splitting, merging and spurious detections of the PLDM-based text line extraction algorithm are shown in Table 3.15. In this experiment, the ground truth text words provided by the UW-III database are used as the input. This algorithm works well when the models are close approximations of the actual situations. A large number of model parameters also need to be determined given the training data. The space between words is assumed as equal

Table 3.12: Bayesian p-values $p(T < T^{rep})$ of (a) word horizontal displacement; (b) word vertical displacement; and (c) independence of horizontal and vertical displacement.

T = Skewness coefficient:		
skew.obs	-0.474	95% interval: (-0.538, -0.413)
skew.rep	1.482	95% interval: (-11.83, 7.35)
p.skew	0.741	
T = Kurtosis coefficient:		
kurtosis.obs	2.032	95% interval: (1.896, 2.182)
kurtosis.rep	729	95% interval: (11.39, 1734)
p.kurtosis	1.000	

(a)

T = Skewness coefficient:		
skew.obs	0.695	95% interval: (0.653, 0.743)
skew.rep	-0.1156	95% interval: (-0.7956, 0.7662)
p.skew	0.033	
T = Kurtosis coefficient:		
kurtosis.obs	3.071	95% interval: (2.872, 3.287)
kurtosis.rep	27.48	95% interval: (8.567, 40.74)
p.kurtosis	1.000	

(b)

T = Correlation coefficient:		
correlate	3.45E-2	
correlate.rep	8.342E-5	95% interval: (-3.126E-2, 3.227E-2)
p.correlate	0.017	

(c)

Table 3.13: Bayesian p-values $p(T < T^{rep})$ of (a) line space; (b) line height; and (c) independence of line space and height.

T = Skewness coefficient:		
skew.obs	-1.048	95% interval: (-1.208, -0.8955)
skew.rep	1.718	95% interval: (-75.53, 28.48)
p.skew	0.745	
T = Kurtosis coefficient:		
kurtosis.obs	1.450	95% interval: (1.170, 1.790)
kurtosis.rep	6662	95% interval: (8.222, 7678)
p.kurtosis	1.000	

(a)

T = Skewness coefficient:		
skew.obs	-1.091	95% interval: (-1.271, -0.921)
skew.rep	0.169	95% interval: (-3.264, 5.079)
p.skew	0.875	
T = Kurtosis coefficient:		
kurtosis.obs	2.202	95% interval: (1.760, 2.730)
kurtosis.rep	34.44	95% interval: (6.638, 140.2)
p.kurtosis	1.000	

(b)

T = Correlation coefficient:		
correlate	1.74E-3	
correlate.rep	2.474E-5	95% interval: (-4.293E-3, 4.070E-3)
p.correlate	0.184	

(c)

Table 3.14: Bayesian p-values $p(T < T^{rep})$ of text line edges within a paragraph.

T = Skewness coefficient:		
skew.obs	8.413E-2	95% interval: (-1.975E-2, 1.874E-1)
skew.rep	-7.421E-2	95% interval: (-3.072, 3.667)
p.skew	0.432	
T = Kurtosis coefficient:		
kurtosis.obs	3.687	95% interval: (3.235, 4.187)
kurtosis.rep	53.86	95% interval: (8.112, 249.1)
p.kurtosis	1.000	

with a Gaussian variation. Therefore, when the inter-word spacing is much larger than the regular spacing, due to the large font size, forced justification, heading, etc., the text line is split. This algorithm assumes the words within a text line form a straight line. Therefore, it may split a text line when the words are not in a straight line due to some distortion, such as warping. The algorithm is unsuitable for documents that do not fit its model. It assumes the documents have homogeneous layout, i.e., similar font size and spacing between text entities. Therefore, the algorithm produces poor results on pages with various layout attributes.

Performance of text block extraction

In this experiment, we apply the PLDM-based text block extraction algorithm to the ground truth text lines provided by the UW-III database. Table 3.16 illustrates the numbers and percentages of miss, false, correct, splitting, merging and spurious detections with respect to the ground truth text blocks as well as the algorithm output. The performance of this algorithm depends on if the documents fit its model and on the training data by which the model parameters are determined.

Table 3.15: Performance of line segmentation (PLDM) with respect to the ground truth and the algorithm output.

Ground Truth	Correct	Splitting	Merging	Miss	Spurious
105439	101736 (96.49%)	1486 (1.41%)	2136 (2.03%)	11 (0.01%)	70 (0.07%)
Detected	Correct	Splitting	Merging	False	Spurious
106120	101736 (95.87%)	3235 (0.30%)	988 (0.93%)	0 (0.00%)	161 (0.15%)

Table 3.16: Performance of text block segmentation (APLDM) with respect to the ground truth and the algorithm output.

Ground Truth	Correct	Splitting	Merging	Miss	Spurious
21738	15861 (72.96%)	1414 (6.50%)	3312 (15.24%)	8 (0.04%)	1143 (5.26%)
Detected	Correct	Splitting	Merging	False	Spurious
22370	15861 (70.90%)	3933 (17.58%)	1456 (6.51%)	0 (0.00%)	1120 (5.01%)

3.5 Summary

This chapter introduces quantitative performance metrics for each kind of information a document analysis technique infers. A large quantity of ground-truth data, varying in quality, is required in order to give an accurate measurement of the performance of an algorithm under different conditions. In the University of Washington English Document Image Database-III [44], there are 1600 English document images that come with manually edited ground-truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text lines, and words. We do the performance evaluation by determining the correspondence between the ground-truth document structure and the automatically computed document structure, making the comparison between two structures, and reporting the performance measures.

Chapter 4

A UNIFIED APPROACH TO DOCUMENT STRUCTURE ANALYSIS

In this chapter, we formulate the document structure analysis as a partitioning problem. The goal of the problem is to find an optimal solution to partition the set of glyphs on a given document to a hierarchical tree structure where entities within the hierarchy are associated with their physical properties and semantic labels. This chapter describes a document structure extraction algorithm that is probability based, where the probabilities are estimated from an extensive training set of various kinds of measurements of distances between the terminal and non-terminal entities with which the algorithm works. The off-line probabilities estimated in the training then drive all decisions in the on-line segmentation module. An iterative, relaxation like method is used to find the partitioning solution that maximizes the joint probability.

We have implemented the text line and text block extraction algorithm using this framework. The algorithm was evaluated on the UW-III database of some 1600 scanned document image pages. The text line extraction algorithm identifies and segments 99.76% of text lines correctly, while the preliminary result of the text block extraction shows 91% accuracy.

This chapter is organized as follows. In Section 4.1, the consistent partition and labeling problem is formulated and a generic algorithm is presented. The application of our method to extraction of the text lines and text blocks are described in Section 4.2 and 4.3, respectively. In Section 4.4, we discuss the experimental protocol. Finally, the experimental results are presented in Section 4.5.

4.1 Consistent Partition and Labeling Problem and Algorithm

Given a set of polygonal areas at a certain level of document hierarchy, document structure extraction is to generate another level of hierarchy, which contains a set of polygonal structures of interest. Let $\mathcal{A}$ be the set of entities at the source level. Let $\mathit{\Pi}$ be a *partition* of $\mathcal{A}$ and each element of the partition is an entity on the target level. A system $\mathit{\Pi}$ of nonempty sets is called a partition of $\mathcal{A}$ if

1. $\mathit{\Pi}$ is a system of mutually disjoint sets, i.e., if $C \in \mathit{\Pi}$, $D \in \mathit{\Pi}$, and $C \neq D$, then $C \bigcap D = \emptyset$;

2. the union of $\mathit{\Pi}$ is the whole set $\mathcal{A}$, i.e., $\bigcup \mathit{\Pi} = \mathcal{A}$.

Let L be a set of labels, such as content, content type, format attributes, etc., that can be assigned to elements of the partition. The possible labels depend on the specific application. For the text recognition, L may contain the alphabet of all symbols to be recognized. For the segmentation problem, one can use the style attribute of each structure as the label.

Function $f : \mathit{\Pi} \to L$ associates each element of $\mathit{\Pi}$ with a label. $V : \wp(\mathcal{A}) \to \mathit{\Lambda}$ specifies measurement made on subset of $\mathcal{A}$, where $\mathit{\Lambda}$ is the measurement space.

The consistent partition and labeling problem can be formulated as follows: *Given initial set $\mathcal{A}$, find a partition $\mathit{\Pi}$ of $\mathcal{A}$, and a labeling function $f : \mathit{\Pi} \to L$, that maximizes the probability:*

$$\begin{aligned} P(V(\tau) : \tau \in \mathit{\Pi}, f, \mathit{\Pi}|\mathcal{A}) &= P(V(\tau) : \tau \in \mathit{\Pi}|\mathcal{A}, \mathit{\Pi}, f)P(\mathit{\Pi}, f|\mathcal{A}) \\ &= P(V(\tau) : \tau \in \mathit{\Pi}|\mathcal{A}, \mathit{\Pi}, f)P(f|\mathit{\Pi}, \mathcal{A})P(\mathit{\Pi}|\mathcal{A}). \end{aligned} \tag{4.1}$$

By making the assumption of conditional independence, that when the label $f(\tau)$ is known then no knowledge of other labels will alter the probability of $V(\tau)$, we can decompose the probability (4.1) into

$$P(V(\tau) : \tau \in \mathit{\Pi}, f, \mathit{\Pi}|\mathcal{A}) = \prod_{\tau \in \mathit{\Pi}} P(V(\tau)|f(\tau))P(f|\mathit{\Pi}, \mathcal{A})P(\mathit{\Pi}|\mathcal{A}) \tag{4.2}$$

The above proposed formulation can be uniformly apply to construction of the document hierarchy at any level, e.g., text-word, text line, and text block extraction, just to name a few. For example,

1. *text block extraction and labeling*

   Given a set of text lines, group text lines into a set of text blocks, each block having homogeneous formatting attributes, e.g. leading, justification, and the attributes between neighboring blocks being similar.

2. *character segmentation and recognition*

   Given a set of components (over-segmented primitives), and a set of characters within an alphabet, group components into a set of glyphs, each glyph being associated with a character, and subsequences of the characters satisfying context-constraint relationships.

The brute-force method will search through all possible partitions with all possible labels, and select the configuration which produces the highest probability. The number of partitions of an $n-$set into k blocks is called a Stirling number of the second kind:

$$S(n,k) = kS(n-1,k) + S(n-1,k-1), \tag{4.3}$$

where $S(1,1) = 1$. The number of partitions of an $n-$set is called a Bell number:

$$b_{n+1} = \sum_{k=0}^{n} b_k \left(\begin{array}{c} n \\ k \end{array} \right), \tag{4.4}$$

where $b_0 = b_1 = 1$. Table 4.1 shows the number of possible partitions for the set with different size.

Fortunately, the polygonal areas on a document page are usually aligned with certain read order. For example, the read order of the English document is top-down and left-to-right. Therefore, we can make the elements of $\mathcal{A}$ sequential according to

Table 4.1: Illustrates the number of possible partitions (Bell number) for the sets with different size.

Size of set	1	2	5	7	10	14
Number of partitions	1	2	52	877	115975	190,899,322

the read order, then check whether or not each pair of adjacent elements should be grouped,

$$Group(A_i, A_{i+1}) = \begin{cases} Yes \\ No \end{cases} \tag{4.5}$$

Total number of ordered partitions of an $n-$set is 2^{n-1}.

Let $A = (A_1, A_2, \cdots, A_N)$ be a linearly ordered subset of polygonal areas (chain in $\mathcal{A}$). Let $\mathcal{G} = \{Y, N\}$ be the set of grouping labels. Let A^p be a set of pairs of adjacent polygonal areas, such that $A^p \subset A \times A$ and $A^p = \{(A_i, A_j)|A_i, A_j \in A \text{ and } j = i+1\}$. Grouping function, $g : A^p \to \mathcal{G}$, associates each pair of adjacent elements of A with a grouping label, where $g(i) = g(A_i, A_{i+1})$. Then, the partition probability $P(\mathit{\Pi}|A)$ can be computed as follows,

$$\begin{aligned} P(\mathit{\Pi}|A) &= P(g|A) \\ &= P(g(1), \cdots, g(N-1)|A_1, \cdots, A_N) \\ &= P(g(1)|A_1, A_2) \times \cdots \times P(g(N-1)|A_{N-1}, A_N) \\ &= \prod_{i=1}^{N-1} P(g(i)|A_i, A_{i+1}). \end{aligned} \tag{4.6}$$

Therefore, the joint probability (4.1) is further decomposed as

$$\begin{aligned} &P(V(\tau) : \tau \in \mathit{\Pi}, f, \mathit{\Pi}|A) \\ &= \prod_{\tau \in \mathit{\Pi}} P(V(\tau)|f(\tau))P(f|\mathit{\Pi}, A) \prod_{i=1}^{N-1} P(g(i)|A_i, A_{i+1}). \end{aligned} \tag{4.7}$$

An iterative search method (see Figure 4.1) is developed to find the consistent partition and labeling that maximizes the joint probability (4.7). First, the grouping probability $P(g(i)|A_i, A_{i+1})$ between each pair of adjacent input entities is computed, by observing the spatial relationship between the two. An initial partition is determined based on the grouping probabilities. Then, we start to adjust the partition and assign labels to the members of the partition, by maximizing the labeling and the label context probability $\prod_{\tau \in \Pi} P(V(\tau)|f(\tau))P(f|\Pi, A)$. At each iteration, the adjustment that produces the maximum improvement of the joint probability (4.7) is selected. The iteration stops when there is no improvement on the joint probability. Given an $n-$set, the size of the search space is therefore decreased to $t \times (n-1)$, where t is the number of iterations.

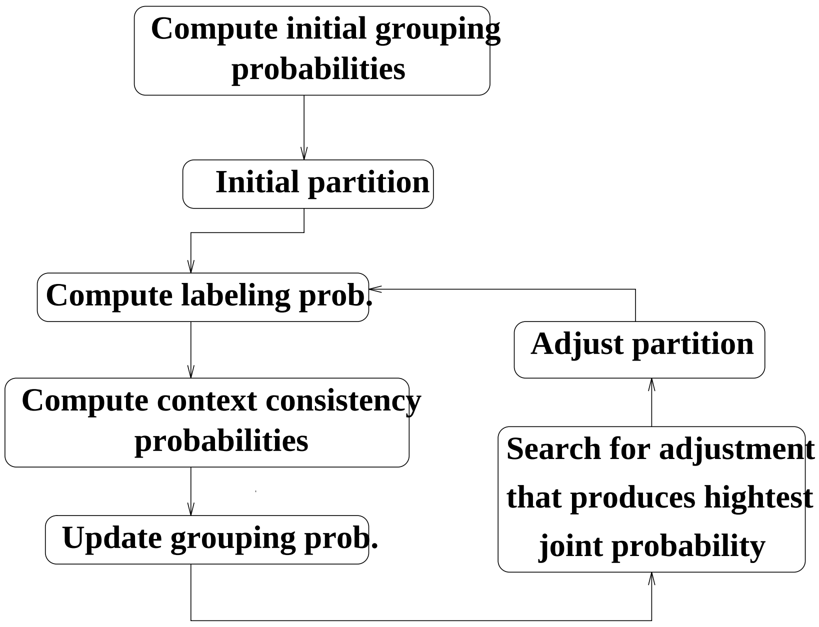

Compute initial grouping probabilities
Initial partition
Compute labeling prob.
Compute context consistency probabilities
Update grouping prob.
Search for adjustment that produces hightest joint probability
Adjust partition

Figure 4.1: Illustrates the iterative search method.

Algorithm 4.1 presents a detailed description of the algorithm. Given an initial set $\mathcal{A}$ of polygonal areas, we first construct the read order of elements of $\mathcal{A}$. Let $A = (A_1, A_2, \cdots, A_M)$ be a linearly ordered subset of polygonal areas.

Algorithm 4.1 *Consistent Partition and Labeling*

1. Determine initial partition

   Let $t = 0$, $\Pi^t = \{\{A_m\}\}_{m=1}^{M}$.

   (a) Compute $P_i^0(Y) = P(g(i) = Y|A_i, A_{i+1})$ and $P_i^0(N) = P(g(i) = N|A_i, A_{i+1})$ where $1 \leq i \leq M-1$.

   (b) Let $R \subseteq A \times A$ and $R = \{(A_i, A_{i+1})|P_i^0(Y) > P_i^0(N)\}$. Update partition

$$\Pi^{t+1} = \{\tau|\tau = \{A_i, \cdots, A_j\}, \text{ where } (A_k, A_{k+1}) \in R, k = i, \cdots, j-1\}. \tag{4.8}$$

2. Search for optimal partition adjustment

   Repeat

   - For $i = 1$ to $M-1$ Do
     - If $A_i \in U$, $A_{i+1} \in W$, and $U \neq W$ Then,

       (a) Let $T = U \bigcup W$, and $\hat{\Pi} = T \bigcup (\Pi^t - U - W)$.

       (b) Find labeling f by maximizing

$$P_{label} = \prod_{\tau \in \hat{\Pi}} P(V(\tau)|f(\tau))P(f|A, \hat{\Pi})$$

       (c) Update $P_i^{\hat{t}}(Y) \propto P_i^0(Y) \times P_{label}$, and $P_i^{\hat{t}}(N) = P_i^{t-1}(N)$.

     - If $A_i \in W$ and $A_{i+1} \in W$, where $W = \{A_k, \cdots, A_i, A_{i+1}, \cdots, A_j\}$, Then

       (a) Let $S = \{A_k, \cdots, A_i\}$, $T = \{A_{i+1}, \cdots, A_j\}$
       and $\hat{\Pi} = (\Pi^t - W) \bigcup S \bigcup T$

       (b) Find labeling f by maximizing

$$P_{label} = \prod_{\tau \in \hat{\Pi}} P(V(\tau)|f(\tau))P(f|A, \hat{\Pi})$$

```
        (c) Update $P_i^t\hat{(N)} \propto P_i^0(N) \times P_{label}$, and $P_i^t\hat{(Y)} = P_i^{t-1}(Y)$
    End
• Select $k$ such that,
        $k = \arg\max_i(\max\{\hat{P_i^t}(Y), \hat{P_i^t}(N)\})$
• If $P_k^t\hat{(Y)} > P_k^t\hat{(N)}$, Then
    – $T = U \bigcup W$ where $A_k \in U, A_{k+1} \in W$
    – $\Pi^{t+1} = (\Pi^t - U - W) \bigcup T$
  Else, $W = \{A_i, \cdots, A_k, A_{k+1}, \cdots, A_j\}$,
    – Let $S = \{A_i, \cdots, A_k\}$ and $T = \{A_{k+1}, \cdots, A_j\}$
    – $\Pi^{t+1} = (\Pi^t - W) \bigcup S \bigcup T$
• If $P(V, f, \Pi^{t+1}|A) \leq P(V, f, \Pi^t|A)$, end and return $\Pi^t$.
  Else, let $t = t + 1$ and continue.
```

In the following sections, we describe the examples of applying our algorithm to extraction of the text lines and the text blocks.

4.2 Text Line Extraction Algorithm

Given a set of glyphs, the goal of the text line extraction is to partition glyphs into a set of text lines, each text line having homogeneous properties, and the text lines' properties within the same region being similar. The text lines' properties include, deviation of glyphs from the baseline, direction of the baseline, text line's height, and text lines' width, etc.

4.2.1 Algorithm description

Figure 4.2 gives an overview of the text line segmentation algorithm. Without loss

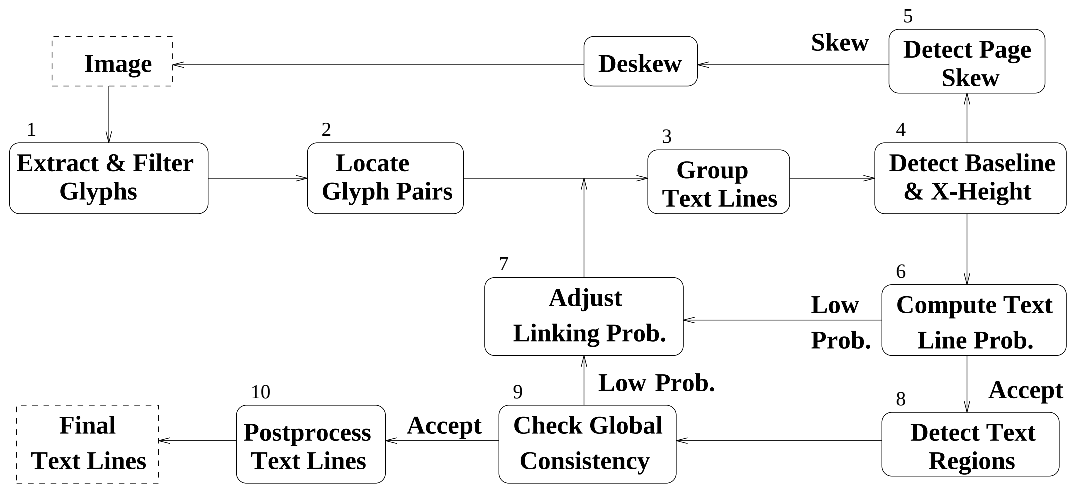


Figure 4.2: Illustrates the processing steps of the text line segmentation algorithm.

of generality, we assume that the reading direction of the text lines on the input page is left-to-right. The text line segmentation algorithm starts with the set of the connected component bounding boxes of a given binary image of a textual document.

Algorithm 4.2 *Extract text lines from glyphs*

1. Extract and filter glyphs:

   We apply the standard connected component algorithm to obtain the glyph set, $C = \{c_1, c_2, \cdots, c_M\}$. Those components that are smaller than the $threshold_{small}$ or larger than the $threshold_{large}$ are removed from C and put into the reserve set.

2. Locate glyph pairs:

   For each $c_i \in C$, we search for its right adjacent neighbor. At the end of this step a set of glyph sequences is established, $T_{chain} = \{t_1, t_2, \cdots, t_{K_1}\}$, where each t_k is a set of linearly ordered glyphs. For each linked pair, c_i and c_j, we compute the grouping probability, $P(sameline(i,j)|c_i, c_j)$. This is the estimated probability that two components with their sizes and spatial relationships lie on the same text line.

3. Group text lines:

   For each pair (c_i, c_j), we estimate the linking probability $P(link(i,j))$ between c_i and c_j. If $P(link(i,j) = Y) > P(link(i,j) = N)$, c_i and c_j are grouped into one text line; otherwise, we disconnect their link.

   During the initial partition, $P(link(i,j)) = P(sameline(i,j)|c_i, c_j)$, and this step yields our initial text line set, $T_{initial} = \{t_1, t_2, \cdots, t_{K_2}\}$.

4. Detect base-line and x-height:

   For each $t_k \in T$, we apply a robust line fitting algorithm to the right-bottom corners of all glyphs in t_k to obtain the base-line and the x-height of t_k. The computation of base-line and x-height are given in Section 4.2.3.

5. Detect page skew:

   The median of all the computed base-lines' direction for the entire set T is taken as the page skew angle, $angle_{skew}$. If the $angle_{skew} > threshold_{skew}$, we rotate the image by $-angle_{skew}$ using the technique given in [59], and the process repeats from step 1. Otherwise, proceed to the next step.

6. Compute text line probability:

   For each text line $t_k \in T$, We compute its probability of having the homogeneous text line properties, $P_{label} = P(V(t_k)|\text{textline}(t_k))$, where the measurement $V(t_k)$ is the deviation of connected components of t_k from its base-line.

7. Adjust pairs linking probability:

We update the linking probability between c_i and c_j by combining their grouping probability with the text line labeling and context consistency probability,

$$P(link(i,j)) \propto P(sameline(i,j)|c_i, c_j)P_{label}P_{context},$$

where P_{label} and $P_{context}$ are computed in Step 6 and 9 respectively. Then, the process repeats from step 3.

8. Detect text regions:

   The extracted text lines are grouped into a set of regions, such that majority of the text lines within a region share the same left or right boundary, or center edge. A complete description of this step is given in section 4.2.4.

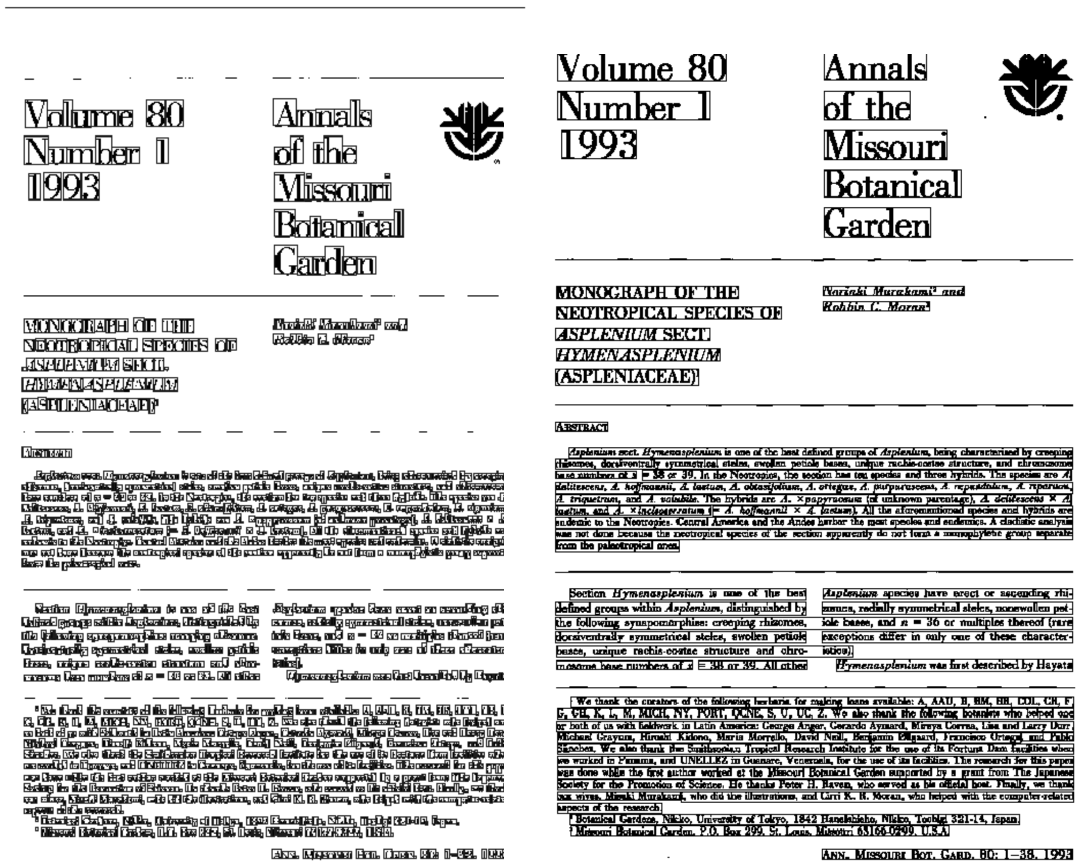
9. Verify context consistency:

   Let $R = \{R_1, R_2, \cdots, R_n\}$ be the set of the detected text regions from the last step. Let $R_i \in R$ and $R_i = \{t_1, t_2, \cdots, t_k\}$. We examine the probability, $P_{context}(\omega(t_j), \omega(R_i))$, that t_j's attributes $\omega(t_j)$ being consistent with its neighboring text lines within R_i. Then, we update the linking probability for each pair associated with t_j, and the process repeats from step 3. The complete description of the global consistent verification, the splitting and the merging procedures are given in detail in section 4.2.5.

10. Postprocess text lines:

    Finally, all components which were initially put into the reserved set and those text lines which were not included during the text-zone formation, or as the results of splitting, are now be individually examined to determine whether it could be included in any of the segmented text lines.

Figure 4.3 and 4.4 illustrate the text line detection process. Figure 4.3(a) shows a set of connected component bounding boxes. The extracted initial text line segments by merging pairs of connected components are illustrated in Figure 4.3(b). We notice some text lines are split horizontally while some are merged across different columns. Figure 4.4(c) plots the extracted text regions by grouping the edges of text line segments. Finally, the corrected text lines given the observations on text regions are shown in Figure 4.4(d).



Volume 80
Number 1
1993

Annals
of the
Missouri
Botanical
Garden

MONOGRAPH OF THE NEOTROPICAL SPECIES OF *ASPLENIUM* SECT. *HYMENASPLENIUM* (ASPLENIACEAE)[1]

Noriaki Murakami[2] *and Robbin C. Moran*[3]

ABSTRACT

Asplenium sect. *Hymenasplenium* is one of the best defined groups of *Asplenium*, being characterized by creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure, and chromosome base numbers of $x = 38$ or 39. In the Neotropics, the section has ten species and three hybrids. The species are *A. delitescens*, *A. hoffmannii*, *A. laetum*, *A. obtusifolium*, *A. ortegae*, *A. purpurascens*, *A. repandulum*, *A. riparium*, *A. triquetrum*, and *A. volubile*. The hybrids are *A.* ×*papyraceum* (of unknown parentage), *A. delitescens* × *A. laetum*, and *A.* ×*inclosserratum* (= *A. hoffmannii* × *A. laetum*). All the aforementioned species and hybrids are endemic to the Neotropics. Central America and the Andes harbor the most species and endemics. A cladistic analysis was not done because the neotropical species of the section apparently do not form a monophyletic group separate from the paleotropical ones.

Section *Hymenasplenium* is one of the best defined groups within *Asplenium*, distinguished by the following synapomorphies: creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure and chromosome base numbers of $x = 38$ or 39. All other *Asplenium* species have erect or ascending rhizomes, radially symmetrical steles, nonswollen petiole bases, and $n = 36$ or multiples thereof (rare exceptions differ in only one of these characteristics).

Hymenasplenium was first described by Hayata

[1] We thank the curators of the following herbaria for making loans available: A, AAU, B, BM, BR, COL, CR, F, G, GH, K, L, M, MICH, NY, PORT, QCNE, S, U, UC, Z. We also thank the following botanists who helped one or both of us with fieldwork in Latin America: George Argent, Gerardo Aymard, Mireya Correa, Lisa and Larry Dorr, Michael Grayum, Hiroshi Kidono, Mario Morello, David Neill, Benjamin Øllgaard, Francisco Ortega, and Pablo Sánchez. We also thank the Smithsonian Tropical Research Institute for the use of its Fortuna Dam facilities when we worked in Panama, and UNELLEZ in Guanare, Venezuela, for the use of its facilities. The research for this paper was done while the first author worked at the Missouri Botanical Garden supported by a grant from The Japanese Society for the Promotion of Science. He thanks Peter H. Raven, who served as his official host. Finally, we thank our wives, Misaki Murakami, who did the illustrations, and Cirri K. R. Moran, who helped with the computer-related aspects of the research.

[2] Botanical Gardens, Nikko, University of Tokyo, 1842 Hanaishicho, Nikko, Tochigi 321-14, Japan.

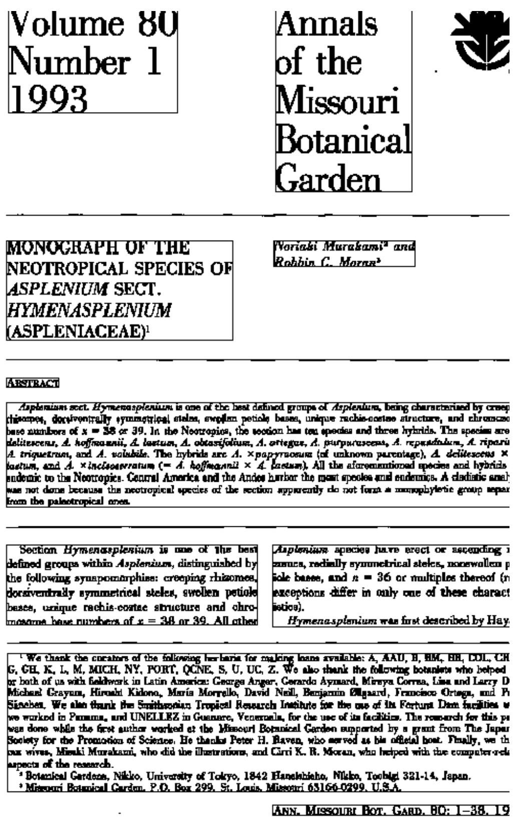
[3] Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166-0299, U.S.A.

ANN. MISSOURI BOT. GARD. 80: 1–38. 1993

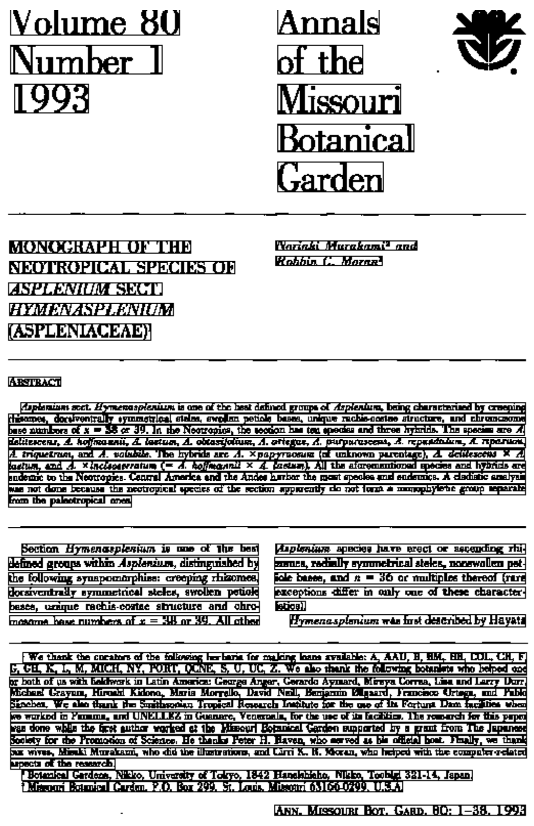
(a) (b)

Figure 4.3: Illustrates a real document image overlaid with the extracted bounding boxes of (a) the connected components; and (b) the initial text line segments.

A few cases that the algorithm failed are shown in Figure 4.5. A vertical merging error was shown in Figure 4.5(a). Figure 4.5(b) and (c) illustrate horizontal and



(c)



(d)

Volume 80
Number 1
1993

Annals
of the
Missouri
Botanical
Garden

MONOGRAPH OF THE NEOTROPICAL SPECIES OF *ASPLENIUM* SECT. *HYMENASPLENIUM* (ASPLENIACEAE)[1]

Noriaki Murakami[2] *and Robbin C. Moran*[3]

ABSTRACT

Asplenium sect. *Hymenasplenium* is one of the best defined groups of *Asplenium*, being characterized by creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure, and chromosome base numbers of $x = 38$ or 39. In the Neotropics, the section has ten species and three hybrids. The species are *A. delitescens*, *A. hoffmannii*, *A. laetum*, *A. obtusifolium*, *A. ortegae*, *A. purpurascens*, *A. repandulum*, *A. riparium*, *A. triquetrum*, and *A. volubile*. The hybrids are *A.* ×*papyraceum* (of unknown parentage), *A. delitescens* × *A. laetum*, and *A.* ×*incisoserratum* (= *A. hoffmannii* × *A. laetum*). All the aforementioned species and hybrids are endemic to the Neotropics. Central America and the Andes harbor the most species and endemics. A cladistic analysis was not done because the neotropical species of the section apparently do not form a monophyletic group separate from the paleotropical ones.

Section *Hymenasplenium* is one of the best defined groups within *Asplenium*, distinguished by the following synapomorphies: creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure and chromosome base numbers of $x = 38$ or 39. All other *Asplenium* species have erect or ascending rhizomes, radially symmetrical steles, nonswollen petiole bases, and $n = 36$ or multiples thereof (rare exceptions differ in only one of these characteristics).

Hymenasplenium was first described by Hayata

[1] We thank the curators of the following herbaria for making loans available: A, AAU, B, BM, BH, COL, CR, F, G, GH, K, L, M, MICH, NY, PORT, QCNE, S, U, UC, Z. We also thank the following botanists who helped one or both of us with fieldwork in Latin America: George Arger, Gerardo Aymard, Mireya Correa, Lisa and Larry Dorr, Michael Grayum, Hiroshi Kidono, Maria Morgello, David Neill, Benjamin Øllgaard, Francisco Ortega, and Pablo Sánchez. We also thank the Smithsonian Tropical Research Institute for the use of its Fortuna Dam facilities when we worked in Panama, and UNELLEZ in Guanare, Venezuela, for the use of its facilities. The research for this paper was done while the first author worked at the Missouri Botanical Garden supported by a grant from The Japanese Society for the Promotion of Science. He thanks Peter H. Raven, who served as his official host. Finally, we thank our wives, Misaki Murakami, who did the illustrations, and Cirri K. R. Moran, who helped with the computer-related aspects of the research.

[2] Botanical Gardens, Nikko, University of Tokyo, 1842 Hanaishicho, Nikko, Tochigi 321-14, Japan.

[3] Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166-0299, U.S.A.

ANN. MISSOURI BOT. GARD. 80: 1–38. 1993

Figure 4.4: Illustrates a real document image overlaid with the extracted bounding boxes of (c) the text regions; and (d) the corrected text lines.

vertical splitting errors due to the large spacing. A spurious error caused by warping is shown in Figure 4.5(d).

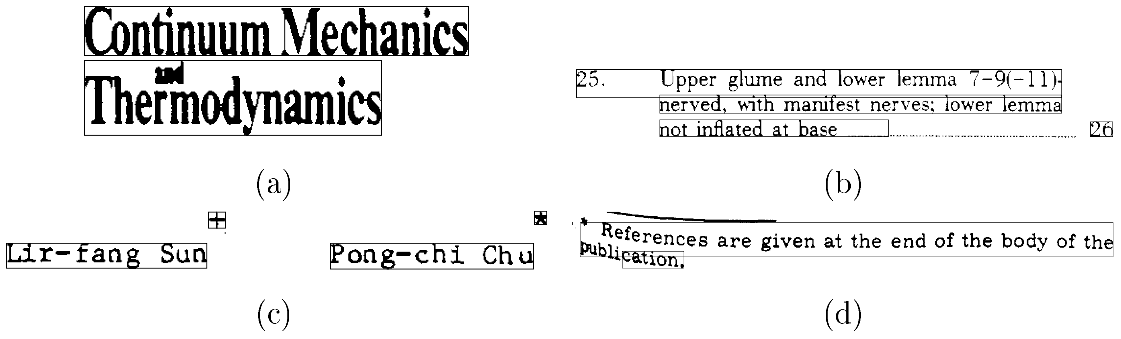

Continuum Mechanics
and
Thermodynamics
(a)
25. Upper glume and lower lemma 7–9(–11)-nerved, with manifest nerves; lower lemma not inflated at base 26
(b)
Lir-fang Sun
Pong-chi Chu
(c)
References are given at the end of the body of the publication.
(d)

Figure 4.5: Illustrates examples that the text line detection algorithm failed.

4.2.2 Group text lines

Let $C = \{c_1, c_2, \cdots, c_M\}$ be the set of glyphs, the connected-component set after the too small and too large components are removed. Each glyph $c_i \in C$ is represented by a bounding box (x, y, w, h), where x, y is the coordinate of top-left corner, and w and h are the width and height of the bounding box respectively. The spatial relations between two adjacent boxes are shown in Figure 4.6.

For a pair of entities c_i and c_j, the horizontal distance $d_h(i,j)$ and vertical distance $d_v(i,j)$ between them are defined as

$$d_h(i,j) = \begin{cases} x_j - x_i - w_i & \text{if } x_j > x_i + w_i \\ x_i - x_j - wj & \text{if } x_i > x_j + w_j \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

$$d_v(i,j) = \begin{cases} y_j - y_i - h_i & \text{if } y_j > y_i + h_i \\ y_i - y_j - h_j & \text{if } y_i > y_j + h_j \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

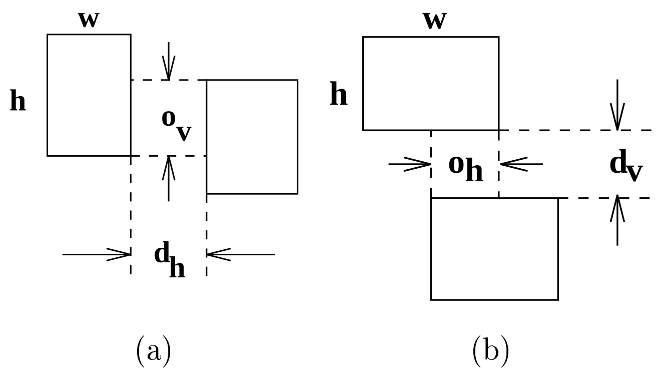


Figure 4.6: Illustrates the spatial relations between two bounding boxes that are (a) horizontally adjacent; and (b) vertically adjacent.

The horizontal overlap $o_h(i,j)$ and vertical overlap $o_v(i,j)$ between c_i and c_j are defined as

$$o_h(i,j) = \begin{cases} x_i + w_i - x_j & \text{if } x_j > x_i \text{ and } x_j < x_i + w_i \\ x_j + w_j - x_i & \text{if } x_i > x_j \text{ and } x_i < x_j + w_j \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

$$o_v(i,j) = \begin{cases} y_i + h_i - y_j & \text{if } y_j > y_i \text{ and } y_j < y_i + h_i \\ y_j + h_j - y_i & \text{if } y_i > y_j \text{ and } y_i < y_j + h_j \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

An entity $c_j = (x_j, y_j, w_j, h_j)$ is horizontally adjacent to entity $c_i = (x_i, y_i, w_i, h_i)$, when

$$c_j = \arg\min_{c_k \in C}(d_h(i,k) | k \neq i, x_k > x_i, \text{and } o_v(i,k) > 0). \tag{4.13}$$

Similarly, an entity c_j is vertically adjacent to a entity c_i, when

$$c_j = \arg\min_{c_k \in C}(d_h(i,k) | k \neq i, y_k > y_i, \text{and } o_h(i,k) > 0). \tag{4.14}$$

For each pair of adjacent glyph c_i and c_j, given the above observations h_i, w_i, h_j, w_j, $d(i,j)$, and $o(i,j)$, we compute the probability that c_i and c_j belong to the same text line:

$$P(sameline(i,j)|c_i,c_j) = P(sameline(i,j)|h_i,w_i,h_j,w_j,d(i,j),o(i,j)).$$

4.2.3 Detect text line properties

Let $t = (c_i, c_{i+1}, \cdots, c_k)$ be an extracted group of glyphs. Each glyph c_j is represented by a rectangular box $(x_{1j}, y_{1j}, x_{2j}, y_{2j})$, where (x_{1j}, y_{1j}) and (x_{2j}, y_{2j}) are the coordinates of the box's top-left and bottom-right corners, respectively. Function f associates t with a label of homogeneous text line, $P(V(t)|f(t))$, where $V(t)$ is the measurement made the t.

Figure 4.11 illustrates a text line and its baseline. The x-height is the height of the lower case characters within the text line.

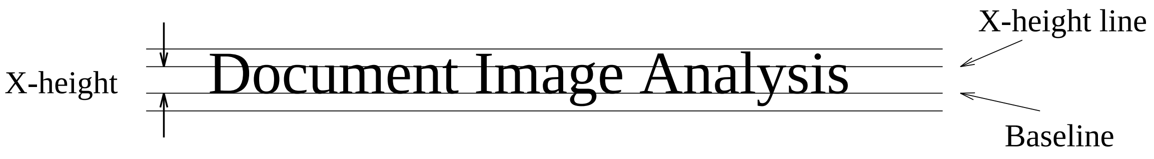


Figure 4.7: Illustrates the baseline and x-height of a text line.

The baseline coordinate of a text line is estimated using a robust estimator. The robust estimation means it is insensitive to small departures from the idealized assumptions for which the estimator is optimized. We want to fit a straight line

$$y(x; a, b) = a + bx$$

through a set of data points $\{(x_{2j}, y_{2j})\}$, which are the bottom-right corner of glyph boxes, since ascenders are used more often in English texts than descenders. The merit function to be minimized is

$$\sum_{j=i}^{k} |y_{2j} - a - bx_{2j}| \tag{4.15}$$

The median d_M of a set of numbers d_j is also the value which minimizes the sum of the absolute deviations $\sum_j |d_j - d_M|$. It follows that, for fixed b, the value of a that minimizes the merit function (4.15) is

$$a = median\{y_{2j} - bx_{2j}\} \tag{4.16}$$

Equation (4.16) for the parameter b is

$$0 = \sum_{j=i}^{k} sgn(y_{2j} - a - bx_{2j}) \tag{4.17}$$

This equation can be solved by bracketing and bisection [60].

Given a set of baseline angles $\{\theta_1, \theta_2, \cdots, \theta_P\}$, the skew angle of page is estimated as

$$\theta_{page} = \text{median}\{\theta_1, \theta_2, \cdots, \theta_P\}.$$

If skew angle θ_{page} is larger than a certain threshold, the page will be rotated by $-\theta_{page}$.

For each given text line $t = (c_i, \cdots, c_k)$ and the estimated baseline (a, b), we compute the absolute deviation of the distance from the glyphs' bottom-right corners (x_{2j}, y_{2j}) to the estimated baseline

$$dev(t, a, b) = \sum_{j=i}^{k} |y_{2j} - a - bx_{2j}|.$$

The x-height of a text line is estimated by taking the median of the distance from the top-left corner (x_{1j}, y_{1j}) of each glyph box to the baseline

$$h_x(t) = \text{median}\{d(x_{1j}, y_{1j}, a, b) | i \leq j \leq k\}$$

Given the observations on text line t, we can compute the likelihood that t has the property of a text line

$$P(h_x(t), dev(t, a, b) | \text{textline}(t)). \tag{4.18}$$

Thus, the linking probability between c_j and c_{j+1} can be updated as follows:

$$\begin{aligned} P(\text{link}(j, j+1) = Y) &\propto P(\text{sameline}(j, j+1) = Y | c_j, c_{j+1}) \\ &\times P(h_x(t), dev(t, a, b) | \text{textline}(t)), \end{aligned} \tag{4.19}$$

and

$$\begin{aligned} P(\text{link}(j, j+1) = N) &\propto P(\text{sameline}(j, j+1) = N | c_j, c_{j+1}) \\ &\times P(h_x(t_1), dev(t_1, a, b) | \text{textline}(t_1)) P(h_x(t_2), dev(t_2, a, b) | \text{textline}(t_2)), \end{aligned} \tag{4.20}$$

where $t_1 = (c_i, \cdots, c_j)$ and $t_2 = (c_{j+1}, \cdots, c_k)$. The probability (4.19) and (4.20) sum to one,

$$P(\text{link}(j, j+1) = Y) + P(\text{link}(j, j+1) = N) = 1.$$

4.2.4 Detect text regions

Given a set of text line bounding boxes $T = \{t_1, t_2, \cdots, t_n\}$, our goal is to group text lines into a set of regions $R = \{R_1, R_2, \cdots, R_q\}$, while the text lines in each region $R_p = (x_p, y_p, w_p, h_p)$ are in the single column bounded by left edge x_p and right edge $x_p + w_p$.

Given an entity box (x, y, w, h), its horizontal projection (see Figure 4.8) is defined as

$$\text{horz-profile}[j] = \text{horz-profile}[j] + 1, x \leq j < x + w.$$

And its vertical projection is

$$\text{vert-profile}[j] = \text{vert-profile}[j] + 1, y \leq j < y + h.$$

Let (x_m, y_m, w_m, h_m) represent the bounding box of a text line $t_m \in T$. We assign the left edge of t_m to be $e_{lm} = x_m$, the right edge of t_m to be $e_{rm} = x_m + w_m$, and

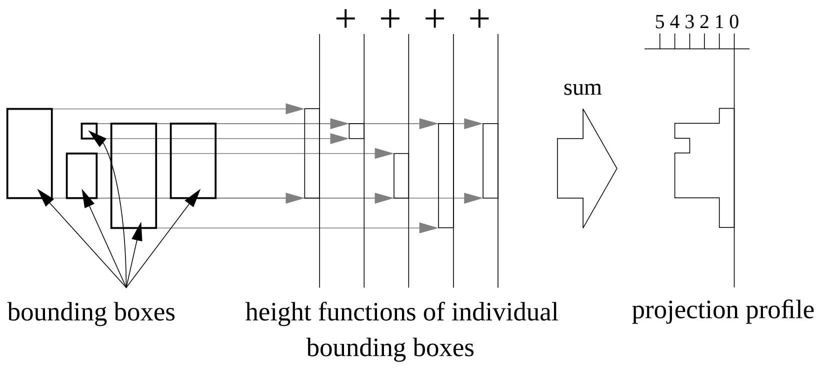


Figure 4.8: Illustrates the horizontal projection of bounding boxes.

the center of t_m to be $e_{cm} = x_m + w_m/2$. The vertical edge count on the three edges of the text line bounding boxes $t_m \in T$ is defined as:

$$\begin{aligned} C_l[j] &= C_l[j] + 1, j = e_{lm}, \text{for } 1 \le m \le n \\ C_c[j] &= C_c[j] + 1, j = e_{cm}, \text{for } 1 \le m \le n \\ C_r[j] &= C_r[j] + 1, j = e_{rm}, \text{for } 1 \le m \le n. \end{aligned}$$

Algorithm 4.3 *Column edge detection*

1. Compute the horizontal projection profile of all text line boxes.

2. Segment the page into a set of large regions, by making cut at the gaps of horizontal projection profile, where the width of gap is larger than a certain threshold. The threshold is determined by the median height of detected text lines.

3. For each region,

   Let $E_l = (e_{l1}, \cdots, c_{ln})$ be the set of left edges. Let $E_c = (e_{c1}, \cdots, c_{cn})$ be the set of center edges. Let $E_r = (e_{r1}, \cdots, c_{rn})$ be the set of right edges.

   Repeat

(a) Compute the vertical projection count of the left edges C_l, right edges C_r, and center edges C_c of text line boxes.

(b) Find a place that has the highest total count within its neighborhood of width w,

$$\arg\max_{i,j}(\sum_k C_i[k] | i \in \{l, r, c\}, j - \frac{1}{2}w \leq k < j + \frac{1}{2}w), \tag{4.21}$$

where w is determined by the dominant text line height within the region.

(c) Let $E_{ij} = \{e_{ik} | e_{ik} \in E_i, j - \frac{1}{2}w \leq e_{ik} < j + \frac{1}{2}w\}$.

(d) Construct a column as a set of text lines whose i edges are within E_{ij}

$$R_p = \{t_m = (e_{lm}, e_{cm}, e_{rm}) | e_{im} \in E_{ij}, i \in \{l, c, r\}\}.$$

(e) Determine the column's left edge as median$\{e_{lm} | t_m \in R\}$, and the column's right edge as median$\{e_{lm} | t_m \in R\}$.

(f) Remove the text line boxes enclosed by the detect column R from T, and remove their edges from E_i.

(g) If $T = \emptyset$, Exit.

(h) Else, continue.

End

If the inter-column spacing between two adjacent columns are very small, it may cause the majority of text lines from those two columns to merge. On the other hand, a list-item structure usually has large gaps and this causes splitting errors. In order to detect these two cases, we compute the vertical projection profile of the glyph bounding boxes enclosed by each column.

If there is a zero-height valley in the profile, we compute the probability that the region should be split into two columns:

$$P(\text{twocolumn}(R) | w_{gap}, n, h_m, h_l, h_r, w_l, w_r), \tag{4.22}$$

where w_{gap} is width of the profile gap, n is the total number of text lines within the current region R, h_m is the median of text line height within R, h_l and w_l (h_r and w_r) are the height and width of the region on the left (right) side of gap. If the probability is larger than a certain threshold, we split the region at the detected gap.

Given a pair of adjacent columns, the probability that they are part of the list-item structure is:

$$P(\text{list-item}(R_l, R_r)|w_{gap}, h_l, h_r, w_l, w_r, n_l, n_r) \tag{4.23}$$

where n_l and n_r are the number of text line within the left and right columns respectively.

4.2.5 Verify context consistency

Given the observations on a text line $t = (c_i, c_{i+1}, \cdots, c_j)$ and its neighbors $N(t) = R - t$ within the same region R, we compute the probability that t is vertically consistent, merged, or split:

$$P(\text{v-consistent}(t, N(t))|h(t), h(N(t)), h_c(t), h_c(N(t))) \tag{4.24}$$

where $h(t)$ is the x-height of the text line t, $h(N(t))$ is the median of text line x-height in $N(t)$, $h_c(t)$ is the median height of glyphs in t, and $h_c(N(t))$ is the median height of glyphs in region $N(t)$. Then, we can update the linking probability between a pair of adjacent glyphs c_i and c_j:

$$P(link(i,j)) \propto P(\text{sameline}(i,j)|c_i, c_j)P(\text{v-consistent}(t, N(t))), \tag{4.25}$$

where $c_i \in t$, and $c_j \in R$.

Given a pair of adjacent text lines t_m and t_n within the same region, we can update the linking probability between a pair of glyphs $c_i \in t_m$ and $c_j \in t_n$:

$$\begin{aligned} P(link(i,j)) &\propto P(\text{sameline}(i,j)|c_i, c_j, \text{sameregion}(i,j)) \\ &\propto P(c_i, c_j|\text{sameline}(i,j))P(\text{sameregion}(i,j)|\text{sameline}(i,j)). \end{aligned} \tag{4.26}$$

Similarly, if a text line is across two or more regions, we can update the linking probability for each pair of adjacent glyphs that belong to different zones

$$\begin{aligned} P(link(i,j)) &\propto P(\text{sameline}(i,j)|c_i, c_j, \text{diffregion}(i,j)) \\ &\propto P(c_i, c_j|\text{sameline}(i,j))P(\text{diffregion}(i,j)|\text{sameline}(i,j)). \end{aligned} \tag{4.27}$$

4.3 Text Block Extraction Algorithm

Given a set of text lines, the text block extraction algorithm groups text lines into a set of text blocks, each block having homogeneous formatting attributes, e.g. leading, justification, and the attributes between neighboring blocks being similar.

4.3.1 Algorithm description

Figure 4.9 gives an overview of the text block extraction algorithm. Without loss of generality, we assume that the reading direction of the text blocks on the input page is top-down. The text block extraction algorithm starts with the set of the text line bounding boxes.

Algorithm 4.4 *Extract text blocks from text lines*

1. Compute local grouping probabilities:

   We first construct the read order of input text lines. For each pair of adjacent text lines, t_i and t_j, we compute the probability that they are within the same text block:

   $$P(\text{SameBlock}(t_i, t_j)|t_i, t_j).$$

2. Group text blocks:

   Given the linking probability $P(link(i,j))$ between a pair of adjacent text lines t_i and t_j, if $P(link(i,j) = Y) > P(link(i,j) = N)$, we group the pair into a text block, otherwise, t_i and t_j are assigned to separate blocks.

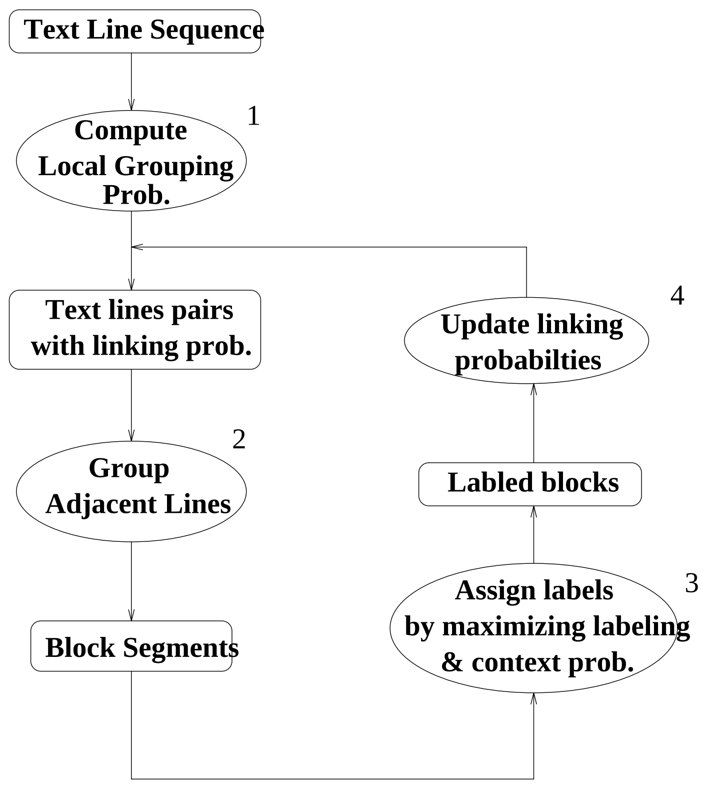

Text Line Sequence
Compute Local Grouping Prob.
1
Text lines pairs with linking prob.
Update linking probabilties
4
Group Adjacent Lines
2
Labled blocks
Assign labels by maximizing labeling & context prob.
3
Block Segments

Figure 4.9: Illustrates the process of text block grouping and labeling.

During the initial partition, $P(link(i,j)) = P(sameblock(i,j))$, and this step yields our initial text line set, $B_{initial} = \{B_1, B_2, \cdots, B_K\}$.

3. Label text blocks

Compute the probability of an extracted block B_k has the homogeneous properties of a text block:

$$P(V(B_k)|f(B_k)),$$

where the labeling $f(B_k)$ includes homogeneous leading and the text alignment type. The text blocks within the same neighbor usually have similar alignment type. The context constraint $P(f|B,T)$ is modeled by a Markov Chain model.

4. Update linking probability and adjust partition

   Given the computed labeling probabilities, we update the linking probability between each pair of adjacent text lines. During each iteration, the adjustment which produces the maximum improvement of the linking probability is selected. Then, we continue to Step 2 and adjust the partition according to the updated linking probabilities. If there is no improvement on the linking probability, we stop the iteration and return the extracted text blocks.

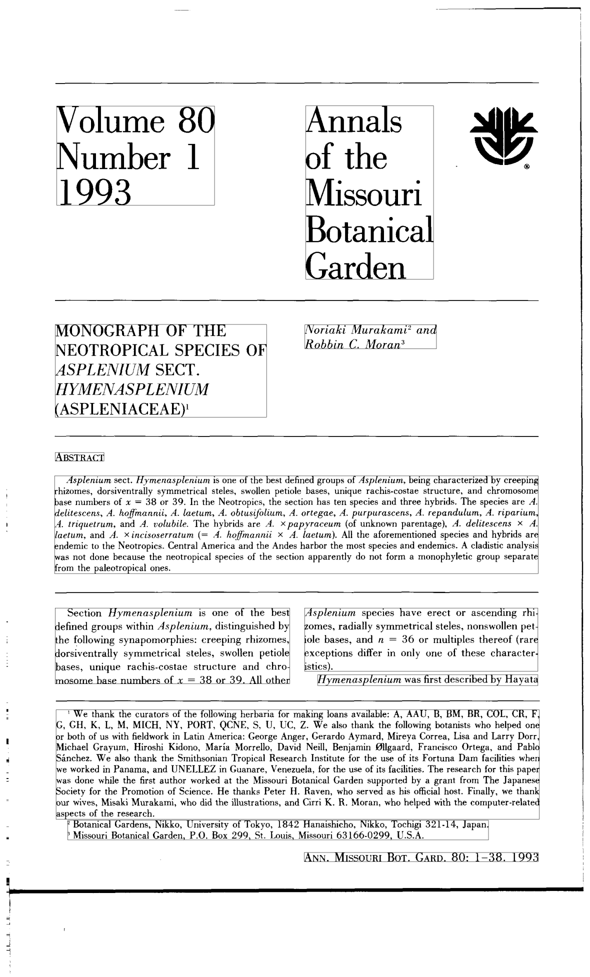
Figure 4.10 illustrates the extracted text blocks given the observed text lines.

4.3.2 Group text blocks

Figure 4.11 illustrates a pair of adjacent text lines. Leading is the distance between baseline of the top text line and the x-height line of the bottom text line.

For each pair of vertically adjacent text lines t_i and t_{i+1}, where the text line is represented by a bounding box (x, y_b, w, h_x) (y_b is the coordinate of the baseline and h_x is the x-height), we make the following observations:

- x-height: h_i and h_{i+1}
- inter-line spacing: $d(i, i+1)$
- horizontal overlap: $o(i, i+1)$
- left edge offset: $e_l(i, i+1) = x_i - x_{i+1}$
- center edge offset: $e_c(i, i+1) = x_i - x_{i+1} + (w_i - w_{i+1})/2$
- right edge offset: $e_r(i, i+1) = x_i - x_{i+1} + w_i - w_{i+1}$



Volume 80
Number 1
1993

Annals
of the
Missouri
Botanical
Garden

MONOGRAPH OF THE NEOTROPICAL SPECIES OF *ASPLENIUM* SECT. *HYMENASPLENIUM* (ASPLENIACEAE)[1]

Noriaki Murakami[2] *and Robbin C. Moran*[3]

ABSTRACT

Asplenium sect. *Hymenasplenium* is one of the best defined groups of *Asplenium*, being characterized by creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure, and chromosome base numbers of $x = 38$ or 39. In the Neotropics, the section has ten species and three hybrids. The species are *A. delitescens*, *A. hoffmannii*, *A. laetum*, *A. obtusifolium*, *A. ortegae*, *A. purpurascens*, *A. repandulum*, *A. riparium*, *A. triquetrum*, and *A. volubile*. The hybrids are *A.* ×*papyraceum* (of unknown parentage), *A. delitescens* × *A. laetum*, and *A.* ×*incisoserratum* (= *A. hoffmannii* × *A. laetum*). All the aforementioned species and hybrids are endemic to the Neotropics. Central America and the Andes harbor the most species and endemics. A cladistic analysis was not done because the neotropical species of the section apparently do not form a monophyletic group separate from the paleotropical ones.

Section *Hymenasplenium* is one of the best defined groups within *Asplenium*, distinguished by the following synapomorphies: creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure and chromosome base numbers of $x = 38$ or 39. All other *Asplenium* species have erect or ascending rhizomes, radially symmetrical steles, nonswollen petiole bases, and $n = 36$ or multiples thereof (rare exceptions differ in only one of these characteristics).

Hymenasplenium was first described by Hayata

[1] We thank the curators of the following herbaria for making loans available: A, AAU, B, BM, BR, COL, CR, F, G, GH, K, L, M, MICH, NY, PORT, QCNE, S, U, UC, Z. We also thank the following botanists who helped one or both of us with fieldwork in Latin America: George Anger, Gerardo Aymard, Mireya Correa, Lisa and Larry Dorr, Michael Grayum, Hiroshi Kidono, María Morrello, David Neill, Benjamin Øllgaard, Francisco Ortega, and Pablo Sánchez. We also thank the Smithsonian Tropical Research Institute for the use of its Fortuna Dam facilities when we worked in Panama, and UNELLEZ in Guanare, Venezuela, for the use of its facilities. The research for this paper was done while the first author worked at the Missouri Botanical Garden supported by a grant from The Japanese Society for the Promotion of Science. He thanks Peter H. Raven, who served as his official host. Finally, we thank our wives, Misaki Murakami, who did the illustrations, and Cirri K. R. Moran, who helped with the computer-related aspects of the research.
[2] Botanical Gardens, Nikko, University of Tokyo, 1842 Hanaishicho, Nikko, Tochigi 321-14, Japan.
[3] Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166-0299, U.S.A.

ANN. MISSOURI BOT. GARD. 80: 1–38. 1993

Figure 4.10: Illustrates a real document image overlaid with the bounding boxes of the extracted text blocks.

The interline spacing $d(i)$ is normalized by the text lines' x-height,

$$d_i = \frac{d(i,i+1)}{\frac{1}{2}(h_i + h_{i+1})}.$$

We use ratio of two text lines' x-height to measure the difference between their font size,

$$x_i = \frac{\min(h_i, h_j)}{\max(h_i, h_j)}.$$

The horizontal overlap between t_i and t_{i+1} is normalized by h_i and h_{i+1} respectively,

$$o_i = \frac{o(i,i+1)}{h_i} \text{ and } o_{i+1} = \frac{o(i,i+1)}{h_{i+1}}.$$

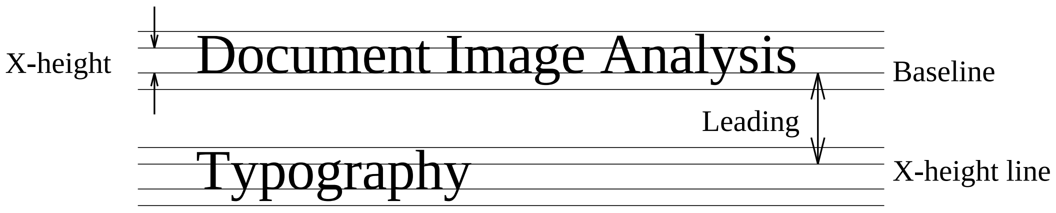


Figure 4.11: Illustrates a pair of adjacent text lines and their spatial relationships.

The relative location between the left edges of t_i and t_{i+1} is defined as,

$$rl_i = \begin{cases} 0 & \text{if } e_l(i,i+1) \ < \frac{1}{2}(h_i + h_{i+1}) \\ 1 & \text{if } e_l(i,i+1) \geq \frac{1}{2}(h_i + h_{i+1}) \\ -1 & \text{if } e_l(i,i+1) \leq \frac{1}{2}(h_i + h_{i+1}) \end{cases}$$

The relative location between the center edges of t_i and t_{i+1} and the relative location between the right edges of t_i and t_{i+1} are defined in the similar fashion.

Given the above measurements, we compute the probability that t_i and t_{i+1} belong to the same block,

$$P(\text{SameBlock}(i,i+1)|x_i, o_i, d_i, rl_i, rc_i, rr_i). \tag{4.28}$$

4.3.3 Homogeneous text block properties

A text block usually has homogeneous inter-line spacing, and certain alignment type (justification, indentation, and hanging). Given a detected text block B, we compute the probability that B has homogeneous leading, and certain type of text alignment,

$$P(V(B)|\text{TextBlock}(B)) = P(V(B)|\text{Leading}(B), \text{Alignment}(B)).$$

In this section, we describe the estimation of homogeneous leading. The text alignment detection method is given in next section.

Let $B = (t_1, \cdots, t_n)$ be an extracted text block. $D_B = (d(1), d(2), \cdots, d(n-1))$ is a sequence of inter-line spaces, where $d(j)$ is the space between t_j and t_{j+1}. We

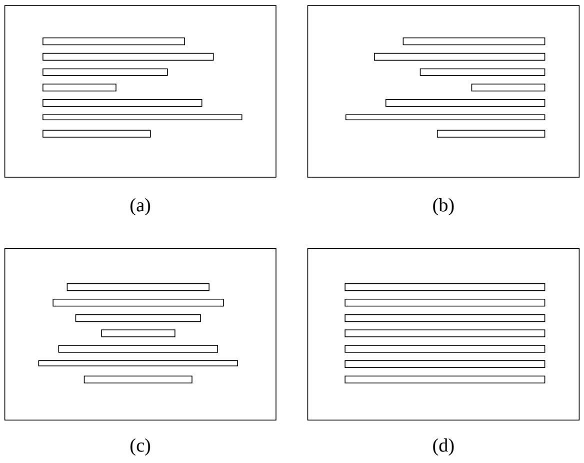


Figure 4.12: Illustrates four types of text justification. (a) left justified, (b) right justified, (c) center justified, and (d) justified.

compute the median and the maximum value of the elements of D_B. The probability of a text block B having homogeneous leading is estimated as:

$$P(\text{median}(D_B), \max(D_B)|\text{Leading}(B)). \tag{4.29}$$

4.3.4 Text alignment detection

Figure 4.12 illustrates four homogeneous types of text block alignment. The various types of indentation are shown in Figure 4.13. Other types of alignment, such as the justified-hanging and left-hanging, can also appear in some document images (See Figure 4.14).

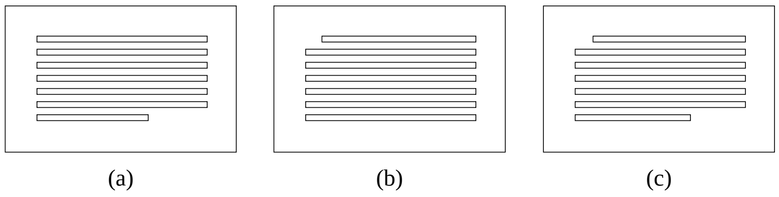


Figure 4.13: Illustrates the various types of indentation. (a) indentation at the first line, (b) indentation at the last line, and (c) indentation at the first and the last lines.

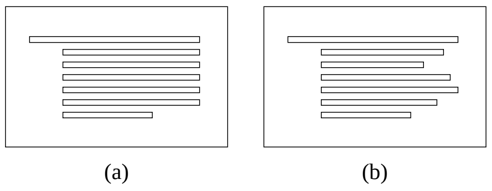


Figure 4.14: Illustrates the hanging type alignment. (a) justified-hanging; (b) left-hanging.

A justification and indentation label is assigned to each possible block using the model shown in Figure 4.15. Given a text block B that consists of a group of text lines

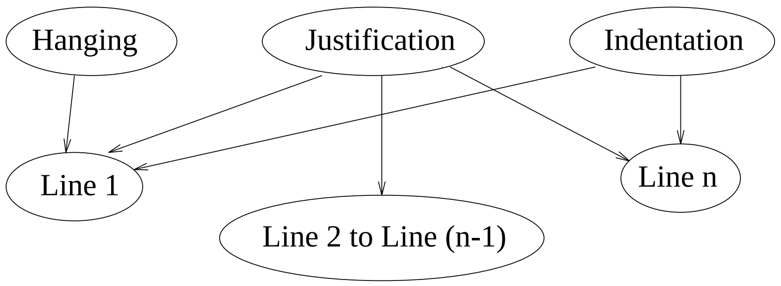


Figure 4.15: Illustrates the process that determines the alignment type of a text block.

$B = (t_1, t_2, \cdots, t_n)$, we determine the text alignment of B by observing the alignment of the text line edges. Let e_{li} be the left edge of the text line t_i and let e_{li} and e_{ri} be the center and right edges of the line box respectively. Let E_l be the left edges of text line 2 to n, such that $E_l = \{e_{li} | 2 \leq i \leq n\}$. E_c is the center edges of text line 2 to $n-1$, and E_r is the right edges of text line 1 to $n-1$. We first estimate the median of E_l, then compute the absolute deviation D_l of the elements of E_l from its median,

$$D_l = \{d_i | d_i = e_{li} - \text{median}(E_l), 2 \leq i \leq n\}.$$

Similarly, we estimate the absolute deviation of center edges and right edges: D_c and D_r. Then, we compute the probability of B being left, center, right, or both justified by observing the mean absolute deviation of left, center and right edges,

$$P(\text{mean}(D_l), \text{mean}(D_c), \text{mean}(D_r)|J(B)), \tag{4.30}$$

where $J(B)$ is the justification type assigned to block B.

For each detected text block, we verified the consistency of text alignment, by checking the maximum deviation of text line edges from the corresponding edge of the text block.

$$\begin{aligned} &P(\max(D_l)|J(B) = left) \\ &P(\max(D_c)|J(B) = center) \\ &P(\max(D_r)|J(B) = right) \\ &P(\max(D_l, D_r)|J(B) = both) \end{aligned} \tag{4.31}$$

Finally, we determine if a left- or both-justified text block has left hanging on the first text line, Let D_{l1} be the distance from e_{l1}, the left edge of the first line, to the median of E_l, the hanging probability is

$$P(D_{l1}|H(B)), \tag{4.32}$$

where $H(B)$ is the hanging label assigned to block B.

4.3.5 Context consistency

The context constraint $P(f|B,T)$ of alignment types of the neighboring text blocks is modeled as a Markov chain,

$$P(f(B_t)|f(B_{t-1}), \cdots, f(B_1)) = P(f(B_t)|f(B_{t-1}))$$

where $B_t \in B$. Therefore, the labeling and context consistency probability

$$\prod_{B_t \in B} P(V(B_t)|f(B_t))P(f|B,T)$$

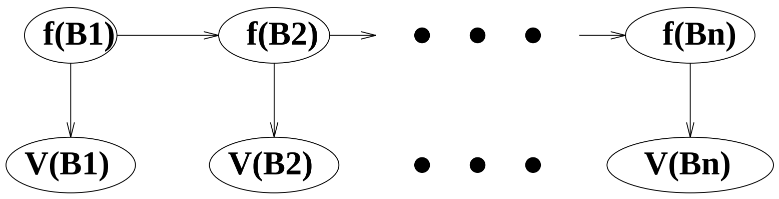


Figure 4.16: Illustrates a model of the text alignment consistency.

is actually a hidden Markov model, shown in Figure 4.3.5.

Given a partition, sequence of labels which maximizes the joint probability can be computed using the Viterbi algorithm. To find the single best state sequence, $Q = \{q_1 q_2 \cdots q_T\}$, for the given observation sequence $O = \{O_1 O_2 \cdots O_T\}$, we need to define the quantity

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t | \lambda] \tag{4.33}$$

i.e. $\delta_t(i)$ is the highest probability along a single path, at time t. By induction we have

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1}). \tag{4.34}$$

We use array $\psi_t(j)$ to keep track of the arguments which maximized (4.34). The complete procedure for finding the best state sequence is stated as follows [61]:

Algorithm 4.5 *Viterbi Algorithm*

1. Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), 1 \le i \le N \tag{4.35}$$

$$\psi_1(i) = 0. \tag{4.36}$$

2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), 2 \le t \le T, 1 \le j \le N \tag{4.37}$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], 2 \le t \le T, 1 \le j \le N. \tag{4.38}$$

3. Termination:

$$\begin{aligned} p^* &= \max_{1\leq i\leq N}[\delta_T(i)] \\ q_T^* &= \arg\max_{1\leq i\leq N}[\delta_T(i)]. \end{aligned} \tag{4.39}$$

4. Path (label sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \cdots, 1. \tag{4.40}$$

4.4 Experimental Protocol

Controlled experiments are an important component of computer vision, for the controlled experiment demonstrates that the algorithm, designed by the computer-vision researcher, recognizes, locates, and measures what it is designed to do from image data [57]. A properly designed scientific experiment provides evidence to accept or reject the hypothesis that the algorithm performs on a specified accuracy level.

The discrete contingency tables are used to represent the joint and conditional probabilities (see Section 4.4.1). In Section 4.4.2, we describe the experimental protocol for estimating probabilities used in the document analysis algorithms, from a significant sized training set. The method we use for estimating the algorithm performance is presented in Section 4.4.3.

4.4.1 Probability representation

The discrete contingency tables are used to represent the joint and conditional probabilities used in the algorithm. Each variable of the table has a finite number of mutually exclusive states. If A is a variable with states $a_1, \cdots, a_n$, then $P(A)$ is a probability distribution over these states:

$$P(A) = (x_1, \cdots, x_n),\ x_i \geq 0,\ \sum_{i=1}^{n} x_i = 1,$$

where x_i is the probability of A being in state a_i [62]. If the variable B has states $b_1, \cdots, b_m$, then $P(A|B)$ is an $n \times m$ table containing numbers $P(a_i|b_j)$ (See Table 4.2). $P(A, B)$, the joint probability for the variables A and B, is also an $n \times m$ table. It

Table 4.2: An example of $P(A|B)$. Note that the columns sum to one.

	b_1	b_2	b_3
a_1	0.33	0.67	0.75
a_2	0.67	0.33	0.25

consists of a probability for each configuration (a_i, b_j) (See Table 4.3).

Table 4.3: An example of $P(A, B)$. Note that the sum of all entries is one.

	b_1	b_2	b_3
a_1	0.1	0.2	0.3
a_2	0.2	0.1	0.1

When the fundamental rule

$$P(A|B)P(B) = P(A, B)$$

is used on variables A and B, then the procedure is to apply the rule to the $n \times m$ configurations (a_i, b_j):

$$P(a_i|b_j)P(b_j) = P(a_i, b_j).$$

This means that in the table $P(A|B)$, for each j the column for b_j is multiplied by $P(b_j)$ to obtain the table $P(A, B)$.

From a table $P(A, B)$ the probability distribution $P(A)$ can be calculated. Let a_i be a state of A. There are exactly m different events for which A is in state a_i, namely the mutually exclusive events $(a_i, b_1), \cdots, (a_i, b_m)$. Therefore,

$$P(a_i) = \sum_{j=1}^{m} P(a_i, b_j).$$

This calculation is called marginalization and we say that the variable B is marginalized out of $P(A, B)$. By marginalizing B out of Table 4.3 we get $P(A) = (0.6, 0.4)$.

The division in Bayes' rule

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

is treated in the same way as the multiplication in the fundamental rule (see Table 4.4).

Table 4.4: $P(B|A)$ as an example of applying Bayes' rule to Table 4.2 and $P(B) = (0.3, 0.3, 0.4)$.

	a_1	a_2
b_1	0.17	0.50
b_2	0.33	0.25
b_3	0.50	0.25

Rather than entering the value of each variable for each individual in the sample, cell count records, for each possible combination of values of the measured variables, how many members of the sample have exactly that combinations of values. A cell count is simply the number of units in the sample that have a given fixed set of values for the variables. The joint probability table can be computed directly from the cell count. An example of the cell count table is shown in Table 4.5. It has 7 variables and 2 states for each variable. Therefore, the table has total of 32 cells.

4.4.2 Estimation of probability distribution and parameters from training set

The steps for estimating the conditional and joint probability tables are as follows:

Table 4.5: Illustrates an example of the cell count table.

		B	No		Yes	
		A	No	Yes	No	Yes
E	D	C				
No	No	No	52	123	24	14
		Yes	12	56	11	56
	Yes	No	12	78	56	23
		Yes	45	67	87	23
Yes	No	No	23	32	12	123
		Yes	45	65	23	89
	Yes	No	90	12	234	128
		Yes	9	12	1	3

1. Determine the variables to observe.

2. Collect and record the data observations.

3. Study graphics and summaries of the collected data to reveal low-dimensional relationships between variables.

4. Choose a model describing the important relationships seen or hypothesized in the data.

5. Quantize the value of each variable into a finite number of mutually exclusive states.

6. Compute the cell count table from the data.

A tree structure quantization is used to partition the value of each variable into bins. At each node of the tree, we search through all possible threshold candidates on each variable, and select the one which gives minimum value of entropy. In growing a tree, the binary partitioning algorithm recursively splits the data in each node until either the node is homogeneous or the node contains too few observations. In order to construct the quantized table from the tree, one follows the path from the root to the leafs within certain level and record the splits made on each variable. The bins on each variable form the cells in the space. The total number of cells, is predetermined based on the memory limitation and the number of samples in the training set. Given this number, one can determine how many levels of nodes will be used to form the cells. For example, suppose a domain has two variables A and B. First, we collect and record the data observations. Then, a classification tree training process is applied to the observed data and the constructed tree in shown in Figure 4.17. If we want to limit the total number of cells under 20, the nodes with depth up to three are used to construct the table. In our example, four thresholds are determined on variable A and the variable B is split three times. Therefore, A and B are quantized into 5 bins and 4 bins respectively. The constructed table with 20 cells is shown in Table 4.6.

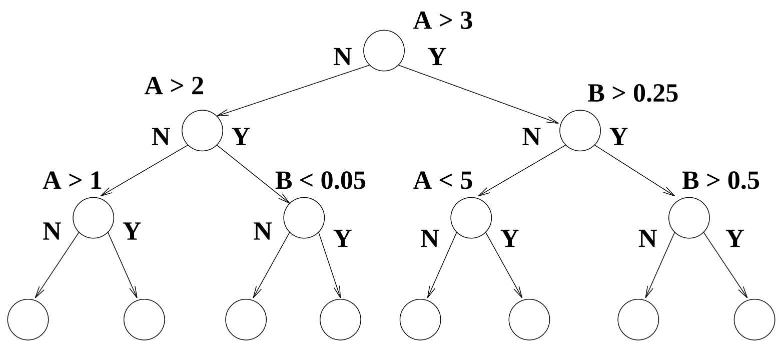


Figure 4.17: Display a constructed tree-based model used for quantization.

Table 4.6: Illustrates a contingence table, where the quantization of variables are determined using the tree structure shown in Figure 4.17.

	$A < 1$	$1 \leq A < 2$	$2 \leq A < 3$	$3 \leq A < 5$	$A \geq 5$
$B < 0.05$					
$0.05 \leq B < 0.25$					
$0.25 \leq B < 0.5$					
$B \geq 0.5$					

In the rest of this section, we present the detailed description of probability distribution estimation for the text block extraction. We conduct a series of experiments to empirically determine the probability distributions that we used to extract text blocks.

1. Text block grouping

   Figure 4.18 illustrates the histogram of measurements made on adjacent text lines that belong to the same block and different blocks. Figure 4.19 illustrates the histogram of the relative location of two adjacent text lines' left, enter, and right edges. A classification tree training program is used to recursively partition the measurement space into cells, by find the threshold that minimizes the entropy at each iteration. The total number of text lines within the data set is around $10,5020$ and this number determines the upper bound of the number of cells. In this experiment, the number of states for the variables are $6, 5, 5, 5, 3, 3, 3, 3$, respectively. Therefore, the probability (4.28) is represented by a table with $20,250$ cells.

2. Text block leading

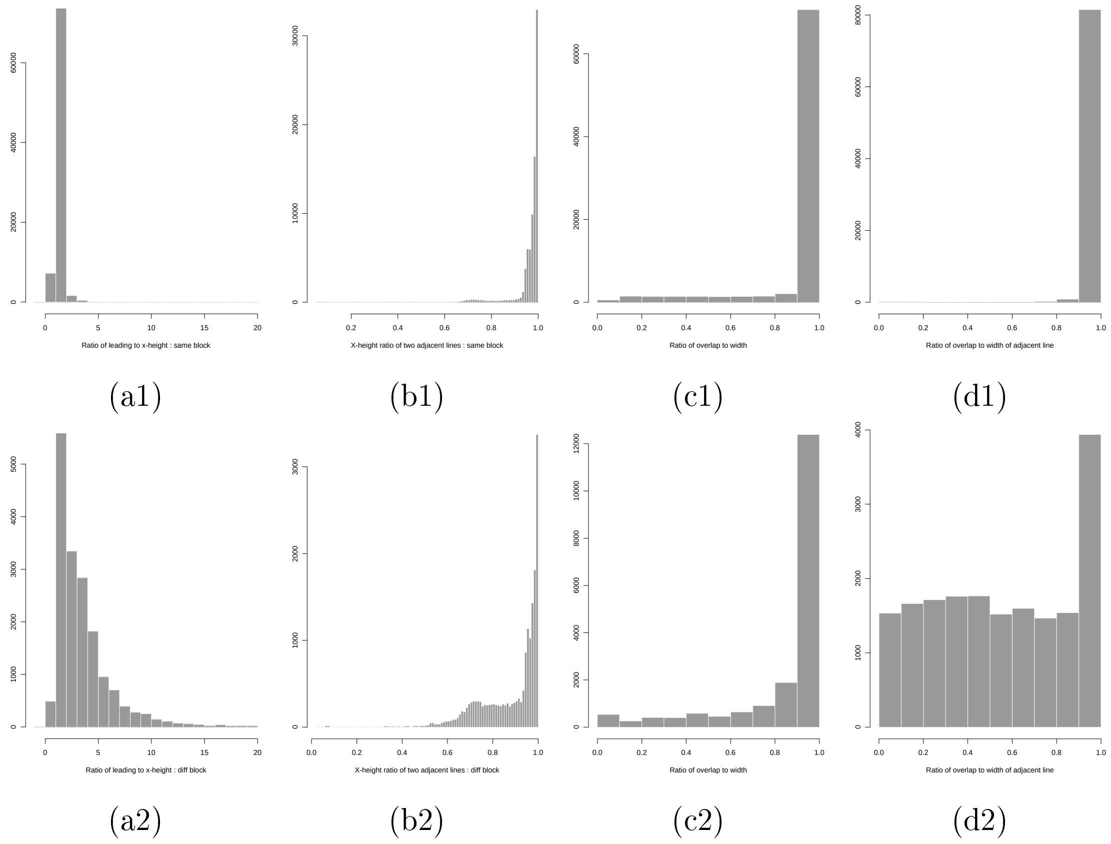


Figure 4.18: Illustrates the histograms of (a) leading to x-height ratio, (b) x-height ratio, (c) the ratio of horizontal overlap to the width of current text line, (d) the ratio of horizontal overlay to the width of adjacent text line, for adjacent text lines that belong to (1) the same text block (2) different text blocks.

Figure 4.20 illustrates the plot of inter-line spacing versus dominant leading within a paragraph and the histogram of the inter-line spacing divided by the dominant leading within a paragraph. The leading variable is quantized to 16 bins. Therefore, the probability (4.29) is represented by a cell count table with 256 cells.

3. Text block alignment

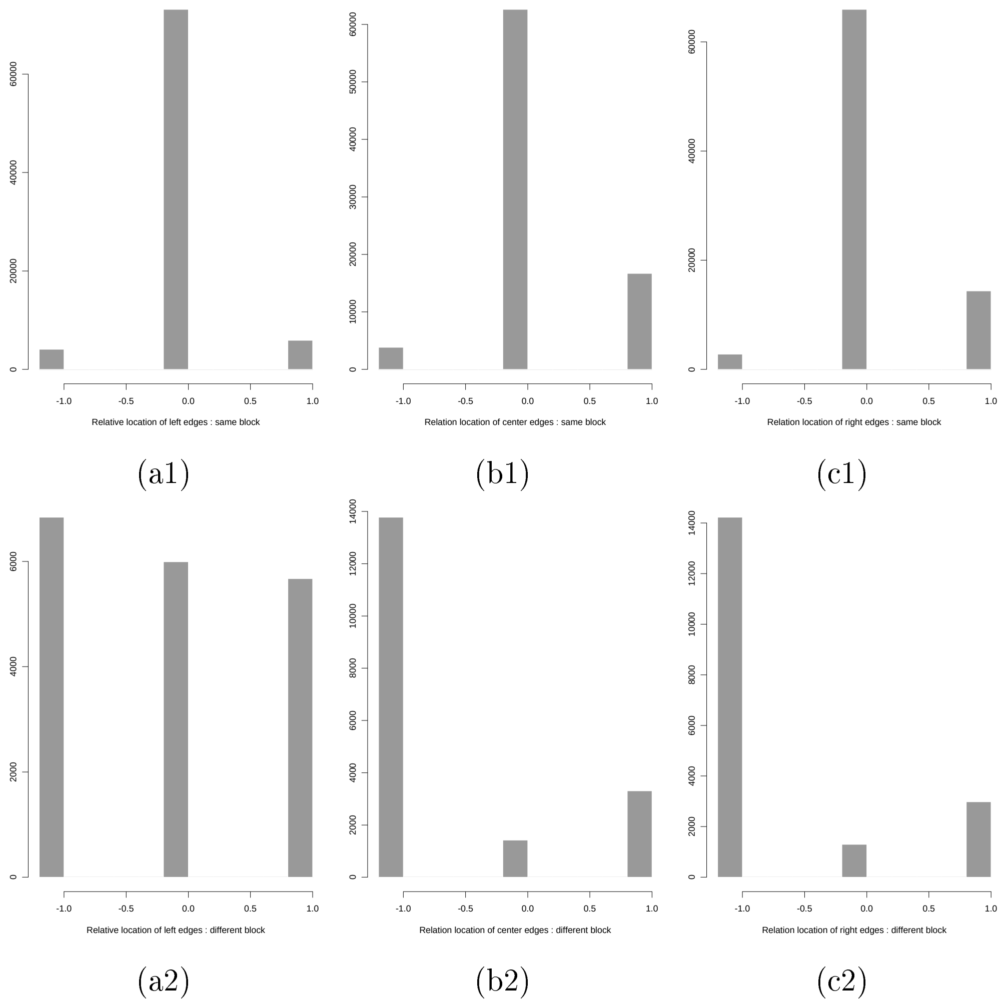


Figure 4.19: Illustrate the relative location between (a) the left edges, (b) the center edges, and (c) the right edges of two adjacent text lines that belong to (1) same text block (2) different text block.

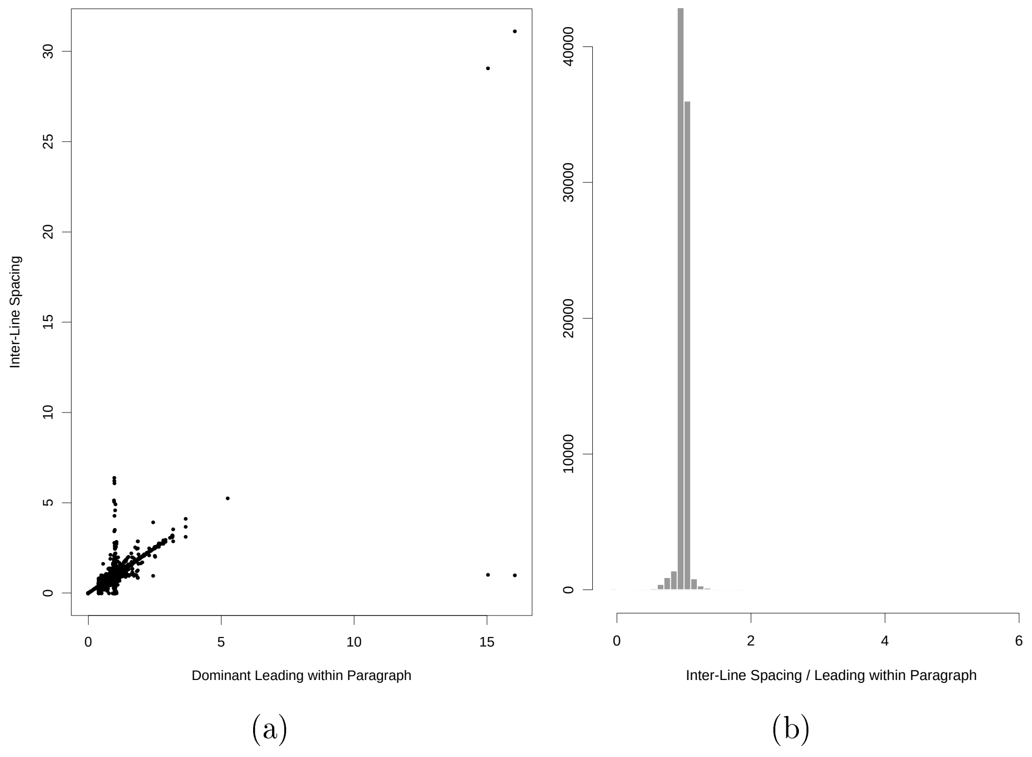
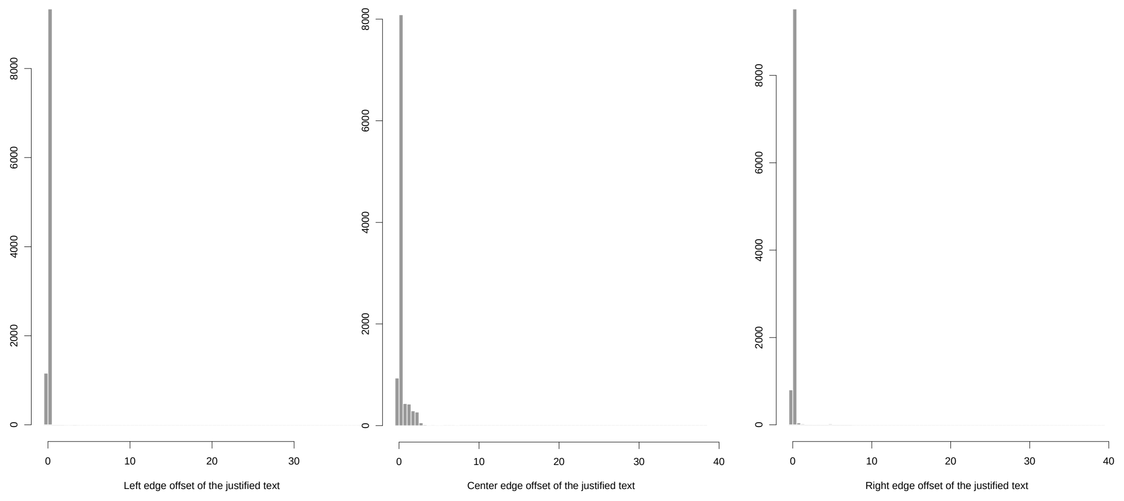


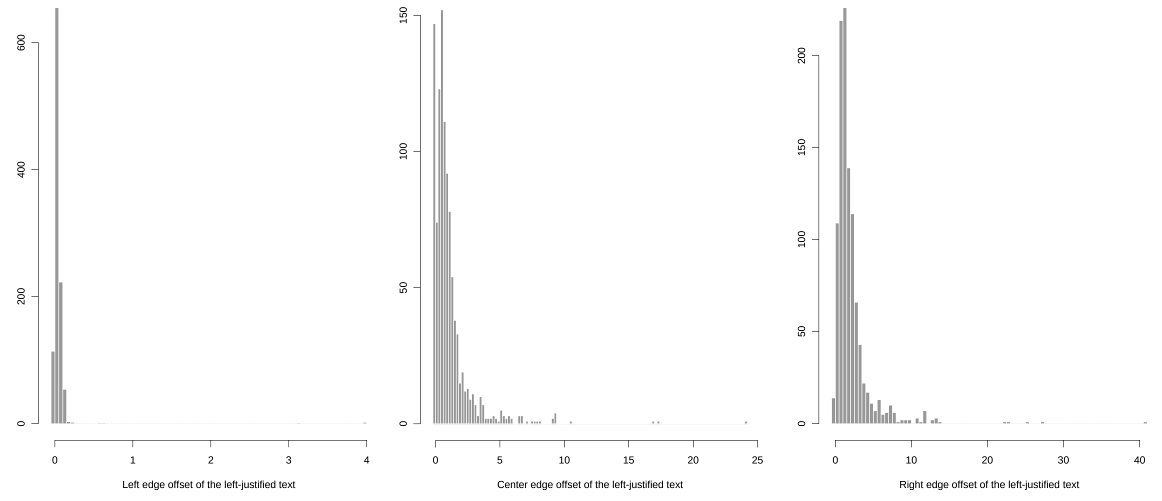
Figure 4.20: Plots (a) inter-line spacing vs. dominant leading within a paragraph; (b) histogram of inter-line spacing / dominant leading within a paragraph.

Based on the assumption of conditional independence, the probability of text block alignment type is decomposed into (4.30), (4.31), and (4.32). Three cell count tables are computed to represent those independent probabilities. Figure 4.21 and 4.22 illustrate the histogram of the mean absolute deviation of the left, center and right edges from the edges of the justified, left-justified, center-justified, and right-justified text, respectively. The values are normalized by the dominant text size within the paragraph. The numbers of states for the mean absolute deviation of the left, center, and right edges are 4, 5, and 7 respectively.



(a) (b) (c)

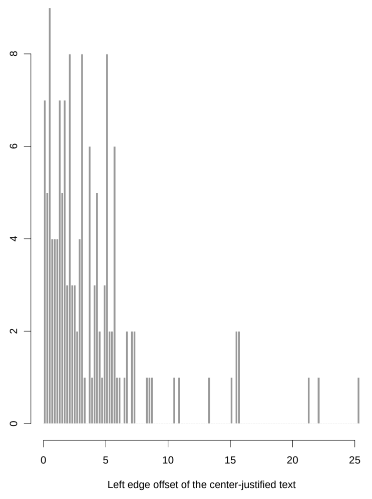
Justified



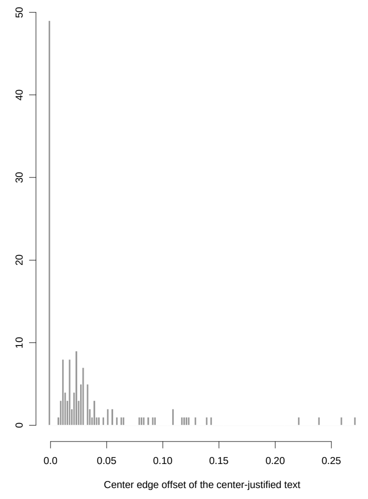
(a) (b) (c)

Left-Justified

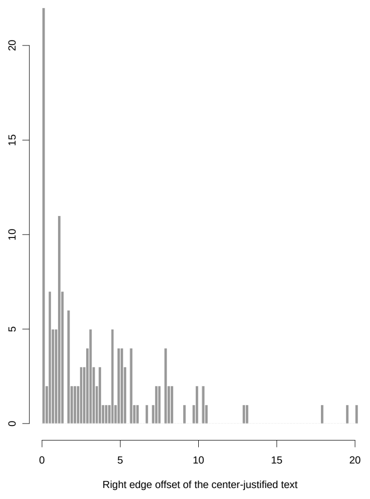
Figure 4.21: Illustrates the histogram of the mean absolute deviation of (a) left edge offset, (b) center edge offset, (c) right edge offset of text lines from the edges of the paragraph, for the justified and left-justified text.



(a)

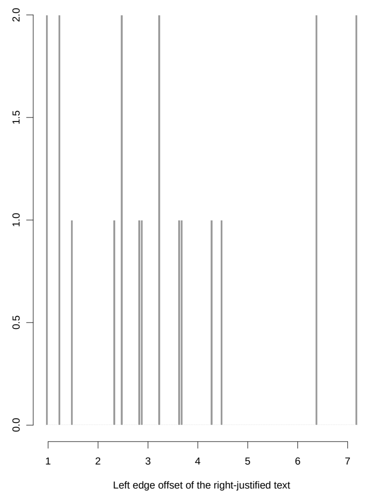


(b)

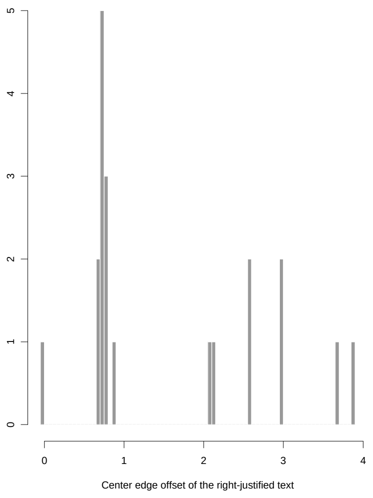


(c)

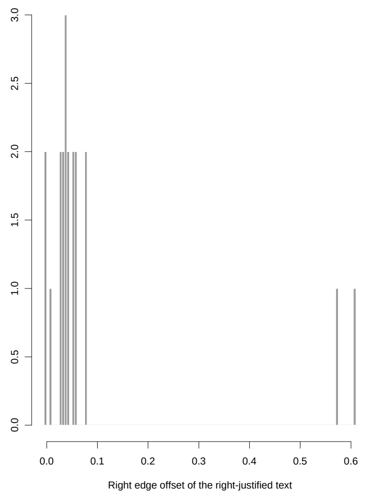
Center-Justified



(a)



(b)



(c)

Right-Justified

Figure 4.22: Illustrates the histogram of the mean absolute deviation of (a) left edge offset, (b) center edge offset, (c) right edge offset of text lines from the edges of the paragraph, for the center-justified and right-justified text.

Figure 4.23 illustrates the histogram of the left edge offset of the first text line within each paragraph, with or without hanging. The values are normalized by the dominant text size with the paragraph. The first text lines' left edge offset is quantized into 6 states.

Figure 4.24 illustrates the histogram of the maximum left and right edge offset of the text lines within the full-justified text. The values are normalized by the dominant text size within the paragraph. The maximum edge offset is quantized to 25 states.

Figure 4.25 illustrates the histogram of the maximum left, center, and right edge offset of the text lines within the left, center and right justified text, respectively. The values are normalized by the dominant text size within the paragraph.

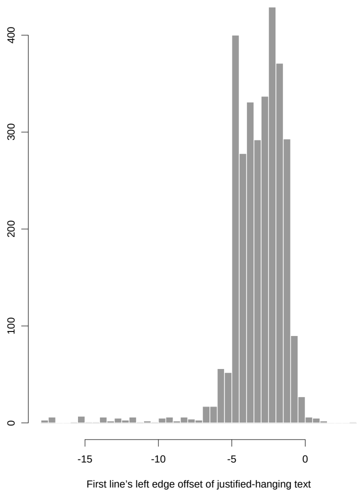
4. Markov chain of text alignment

   Given the alignment type of each text block within the same neighbor, a Markov chain model can be easily updated. Table 4.7 illustrates a computed state transition table $A = \{a_{ij}\}$, where a cell a_{ij} records the number of samples in the training set that the current state is i given the predecessor state is j.

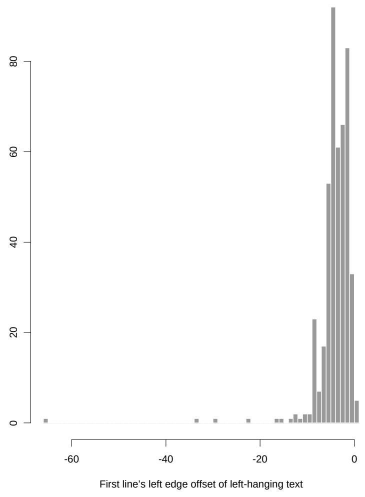
4.4.3 Performance estimation

The quantitative performance metrics of the document structure analysis algorithms are described in Chapter 3. In the UW III document image database, there are total of 1600 document pages. Each page contains manually verified ground truth of text block, text line and word bounding boxes. We use this data set as the training and testing bed of our algorithms.

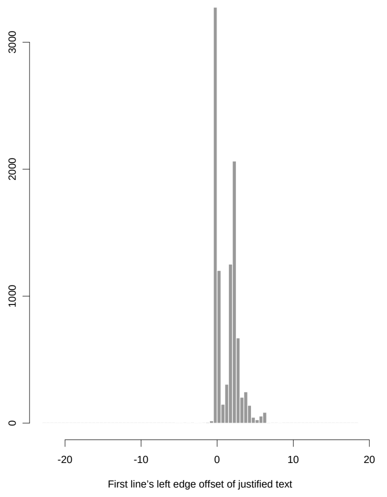
We use the cross-validation method to estimate the performance of document structure analysis algorithms. The data set is divided into N parts. We train on the first $N-1$ parts, and then test on the Nth part. Then train on the $N-1$



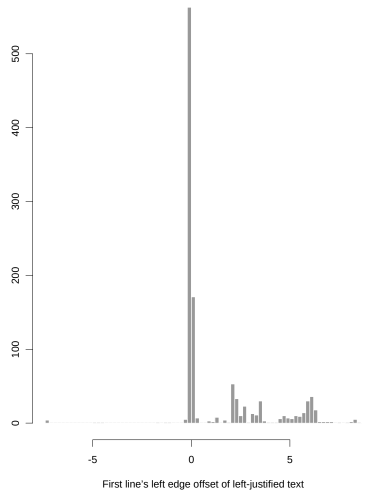
(a1) Justified-Hanging



(b1) Left-Hanging

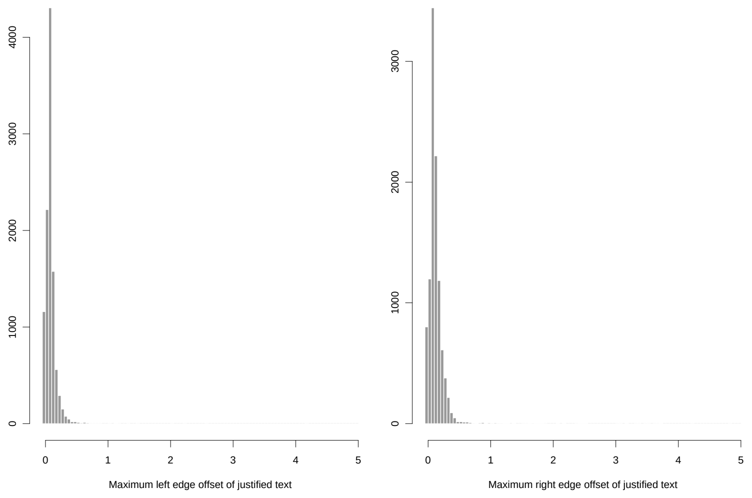


(a2) Justified



(b2) Left-Justified

Figure 4.23: Illustrates the histogram of the left edge offset for the first text line of the paragraph which is (a) justified or left-hanging, compared to (b) justified or left-justified without hanging.



(a) (b)

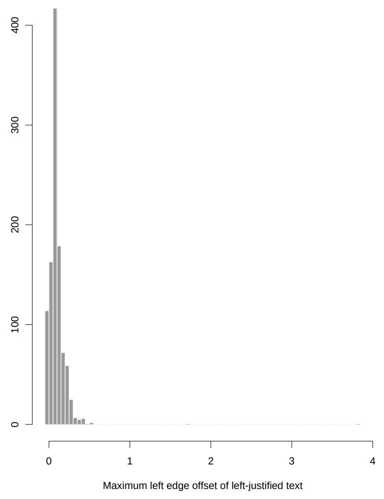
Figure 4.24: Illustrates the histogram of (a) the maximum left edge offset, and (b) the maximum right edge offset, of text lines from the edges of the paragraph, for the full-justified text.

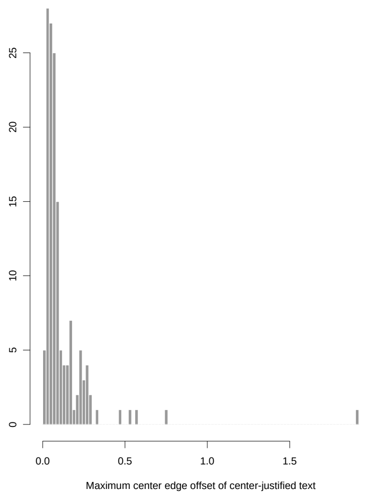
parts, omitting the $N-1st$ part, and test on the $N-1st$ part. We continue the training and testing, each time omitting one part from the probability distribution estimation procedure and then testing on the omitted part. Then we combine the results of the N tests together to establish an estimate of the algorithm performance. The performance evaluation results of text line and text block extraction using the cross-validation method are presented in Section 4.5.

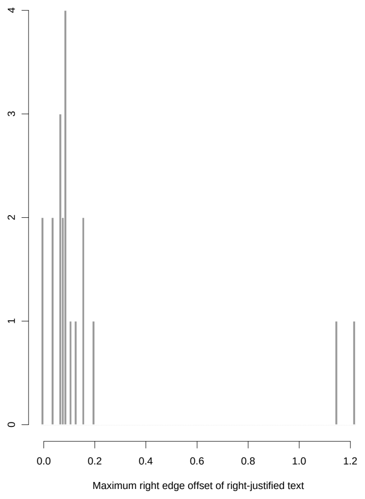
4.5 Experimental Results

4.5.1 Performance of text line and text block extraction from scanned document images

The text structure extraction algorithms described in the previous sections are applied to the total of 1600 images from the UW-III Document Image Database. A







(a) (b) (c)

Figure 4.25: Illustrates the histogram of (a) the maximum left edge offset of text lines within the left-justified text, (b) the maximum center edge offset of text lines within the center-justified text, and (c) the maximum right edge offset of text lines within the right-justified text.

3-fold cross-validation method is used to estimate the algorithms' performance. We partition the data set into three parts, use two parts to do the training and use another part to test and evaluate the performance. The training and testing procedure is repeated three times and a different part is used for testing at each time. Finally, the performance measures from 3 parts are combined as the overall performance of the data set.

Text structure extraction from the connected components within the text ares

In the UW-III database, a page is partition into a set of zones and each zone is assigned a physical content type, such as text, math, table, drawing, figure, ruling, etc. In this experiment, we assume the text areas on a page are given. First, the connected components are extracted from the image. Then, we classify the components into two

Table 4.7: Illustrates a computed state transition table of the text alignment type. Rows are the predecessor states and columns are the current states.

	Justified	Left	Center	Right	Just-hanging	Left-hanging
Justified	4526	186	63	1	229	48
Left	148	397	0	1	229	48
Center	93	2	23	0	5	2
Right	2	1	0	0	0	0
Just-hanging	175	1	1	0	1153	19
Left-hanging	37	15	1	0	26	194

sets: one includes all components that are enclosed by the text areas; another consists of components belong to other regions, such as table, math, line drawing, etc.

The text line extraction algorithm is applied to the set of connected components that are within the text areas. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections are shown in Table 4.8. Of the 105,020 ground truth text lines, 99.76% of them are correctly detected, and 0.08% and 0.07% of lines are split or merged, respectively. Most of the missing errors are due to the rotated text.

Table 4.8: Performance of the text line extraction algorithm.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	105020	104773 (99.76%)	80 (0.08%)	78 (0.07%)	79 (0.08%)	10 (0.01%)
Detected	105019	104773 (99.77%)	172 (0.16%)	37 (0.04%)	25 (0.02%)	12 (0.01%)

In the UW-III database, there are total of 21,779 ground truth text blocks. And each text block is associated with a text alignment label.

First, we apply the text alignment detection algorithm to the ground-truth blocks, and estimate its performance. The classification contingency table is shown in Table 4.9. Note that the classification algorithm rejects a text block when there are less than 3 text lines within the block. The total number of tested text blocks is 11,753. 11,348 of them are correctedly classified and the number of misclassification is 405. Therefore, the text alignment detection algorithm has a misclassification rate of 3.45%.

Table 4.9: Illustrates the text alignment classification results on the ground truth text blocks from the UW-III database. Each text block has at least 3 text lines. The misclassifcation rate is 3.45%.

	Justified	Left	Center	Right	Justified-hanging	Left-hanging
Justified	8254	129	7	2	5	0
Left	19	747	0	0	1	5
Center	20	2	122	1	1	0
right	1	0	1	18	0	0
Justified-hanging	22	1	0	0	1983	128
Left-hanging	4	5	0	0	51	224

Then, we apply the text block extraction algorithm on the ground truth text lines in the UW-III database. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections are shown in Table 4.10. Of the 21,788 ground truth text blocks, 91% of them are correctly detected, and 2.57% and 5.74% of blocks are split or merged, respectively.

Finally, the text block extraction algorithm is applied to the text lines generated

Table 4.10: Performance of the text block grouping algorithm applied on the ground truth text lines.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	21788	19828 (91.00%)	560 (2.57%)	1250 (5.74%)	1 (0.01%)	149 (0.68%)
Detected	21709	19828 (91.34%)	1219 (5.62%)	501 (2.31%)	0 (0.00%)	161 (0.74%)

by our text line extraction algorithm. Table 4.11 shows that 88.91% of text blocks are correctly identified and segmented.

Table 4.11: Performance of text block grouping algorithm applied on the detected text lines.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	21788	19373 (88.91%)	826 (3.80%)	1298 (5.96%)	9 (0.04%)	282 (1.30%)
Detected	22180	19373 (87.34%)	1999 (9.02%)	521 (2.35%)	1 (0.00%)	286 (1.29%)

Text structure extraction from connected components classified as characters

In the previous section, we reported the performance of the text line and text block extraction given the connected components within the text areas. However, the text/nontext information is not always given. In this experiment, we apply the text structure algorithms on whole page. First, the connected component analysis is applied to the input binary image. Then, a few simple measurements are made on each extracted component, i.e., height, width, aspect ratio, black pixel density within the component, etc.. Based on these observations, we determine if a component has the attributes of a character. The components with extreme physical properties are removed from the set. And we run the text structure extraction algorithm on all remained character-like components (glyphs).

For the extracted text lines within the text areas, we compare their bounding boxes with the bounding boxes of the ground truth text lines. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections are shown in Table 4.12. Figure 4.26 illustrates the bounding boxes of the text lines extracted from areas of text, table and drawing.

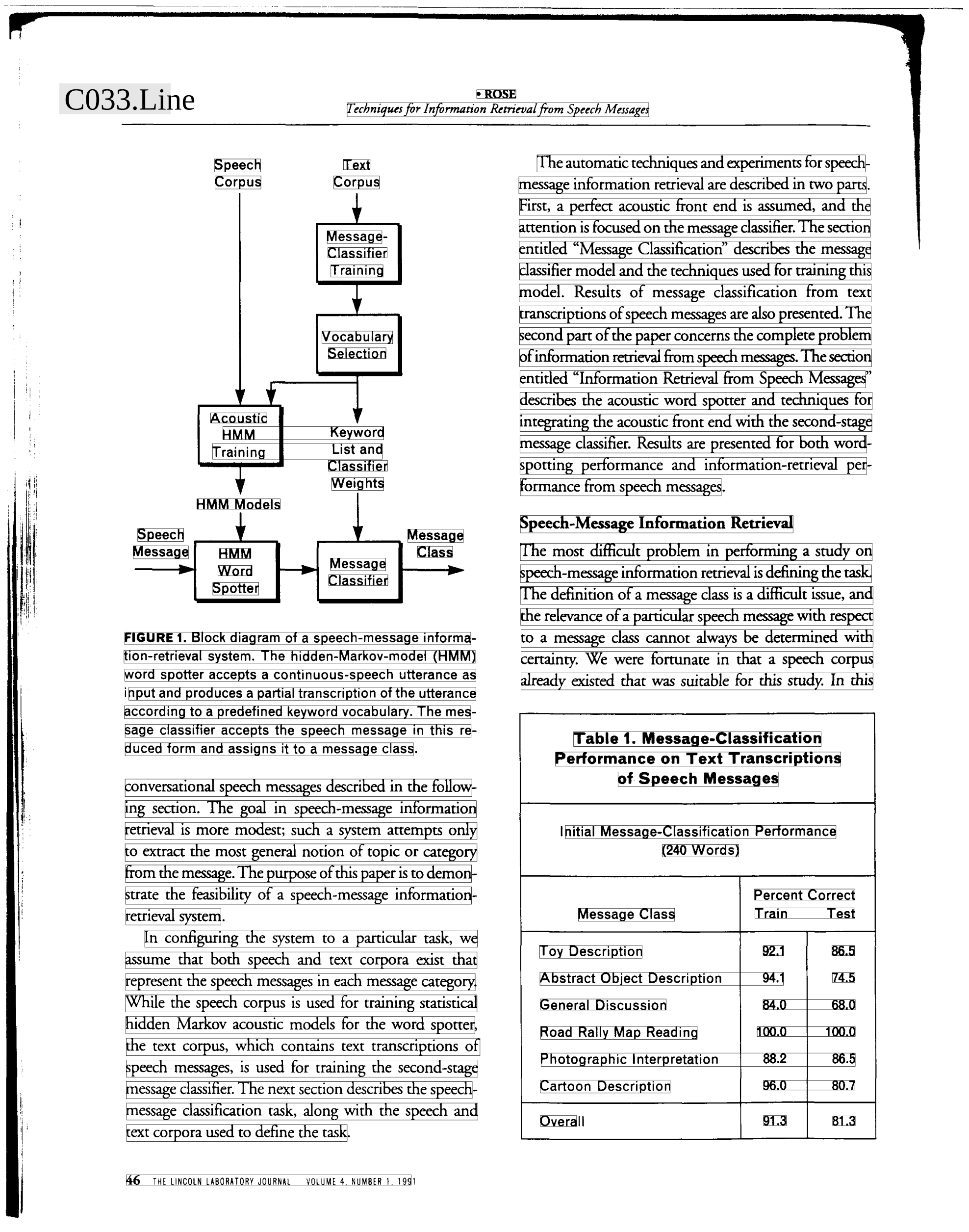
Table 4.12: Performance of text line extraction algorithm on normal sized connected components.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	105020	103829 (98.87%)	638 (0.61%)	422 (0.40%)	66 (0.06%)	65 (0.06%)
Detected	106146	103829 (97.81%)	1653 (1.56%)	210 (0.20%)	36 (0.03%)	418 (0.39%)

The text block extraction algorithm is applied to the bounding boxes of the text lines generated from the previous step. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections are shown in Table 4.13.

4.5.2 Text line and block extraction from glyphs produced by an OCR system

In this experiment, a commercial OCR system is used to identify the character glyphs from the document images. Given the bounding boxes of detected glyphs, we apply our algorithm to reconstruct the structure of each document. We use Caere's M/Text



C033.Line

• ROSE
Techniques for Information Retrieval from Speech Messages

Speech
Corpus
Text
Corpus
Message-
Classifier
Training
Vocabulary
Selection
Acoustic
HMM
Training
Keyword
List and
Classifier
Weights
HMM Models
Speech
Message
HMM
Word
Spotter
Message
Classifier
Message
Class

FIGURE 1. Block diagram of a speech-message information-retrieval system. The hidden-Markov-model (HMM) word spotter accepts a continuous-speech utterance as input and produces a partial transcription of the utterance according to a predefined keyword vocabulary. The message classifier accepts the speech message in this reduced form and assigns it to a message class.

conversational speech messages described in the following section. The goal in speech-message information retrieval is more modest; such a system attempts only to extract the most general notion of topic or category from the message. The purpose of this paper is to demonstrate the feasibility of a speech-message information-retrieval system.

In configuring the system to a particular task, we assume that both speech and text corpora exist that represent the speech messages in each message category. While the speech corpus is used for training statistical hidden Markov acoustic models for the word spotter, the text corpus, which contains text transcriptions of speech messages, is used for training the second-stage message classifier. The next section describes the speech-message classification task, along with the speech and text corpora used to define the task.

The automatic techniques and experiments for speech-message information retrieval are described in two parts. First, a perfect acoustic front end is assumed, and the attention is focused on the message classifier. The section entitled "Message Classification" describes the message classifier model and the techniques used for training this model. Results of message classification from text transcriptions of speech messages are also presented. The second part of the paper concerns the complete problem of information retrieval from speech messages. The section entitled "Information Retrieval from Speech Messages" describes the acoustic word spotter and techniques for integrating the acoustic front end with the second-stage message classifier. Results are presented for both word-spotting performance and information-retrieval performance from speech messages.

Speech-Message Information Retrieval

The most difficult problem in performing a study on speech-message information retrieval is defining the task. The definition of a message class is a difficult issue, and the relevance of a particular speech message with respect to a message class cannot always be determined with certainty. We were fortunate in that a speech corpus already existed that was suitable for this study. In this

Table 1. Message-Classification Performance on Text Transcriptions of Speech Messages

Initial Message-Classification Performance (240 Words)

Message Class	Percent Correct Train	Percent Correct Test
Toy Description	92.1	86.5
Abstract Object Description	94.1	74.5
General Discussion	84.0	68.0
Road Rally Map Reading	100.0	100.0
Photographic Interpretation	88.2	86.5
Cartoon Description	96.0	80.7
Overall	91.3	81.3

46 THE LINCOLN LABORATORY JOURNAL VOLUME 4, NUMBER 1, 1991

Figure 4.26: Illustrates a real document image overlaid with the bounding boxes of the text lines extracted from areas of text, table, and drawing.

Table 4.13: Performance of text block extraction on normal-sized characters.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	21788	17580 (80.67%)	1928 (8.86%)	1735 (7.97%)	22 (0.10%)	523 (2.40%)
Detected	21872	19000 (86.86%)	1895 (8.67%)	860 (3.94%)	0 (0.00%)	117 (0.53%)

OCR engine and the Professional Developer's Kit Version 6.1.2. Given a document image, the system produces recognized characters and words, and their bounding boxes. Text columns, blocks, and lines are not identified. Therefore, the input of this experiment is a set of glyph bounding boxes, and the goal is to construct the document hierarchy from the glyphs.

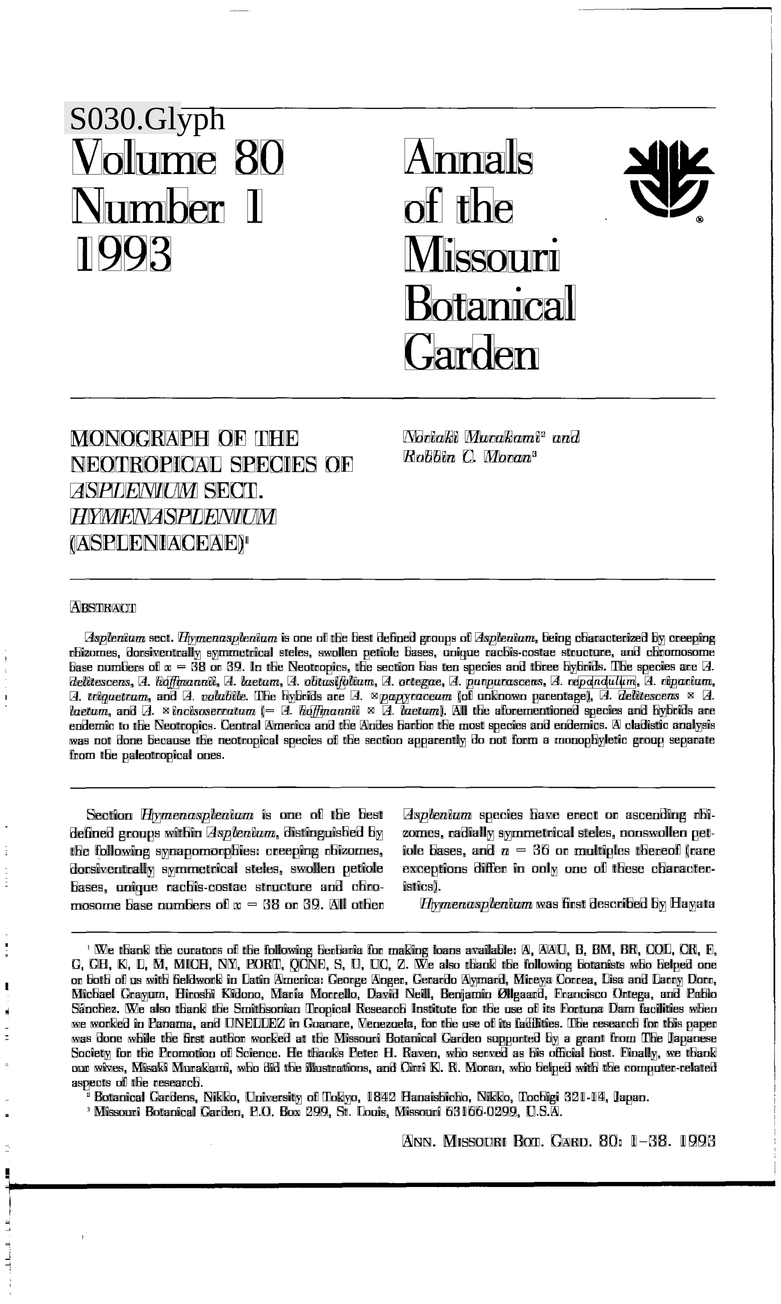
Figure 4.27 shows a set of glyph bounding boxes produced by Caere's OCR engine. The extracted text lines are illustrated in Figure 4.28(a). Figure 4.28(b) illustrates result of the text block extraction.

We test the algorithm on 1600 pages from the UW-III database, and use the cross-validation method to estimate the algorithm's performance. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections of text lines are shown in Table 4.14.

The text block extraction algorithm is applied to the bounding boxes of the text lines generated from the previous step. The numbers and percentages of miss, false, correct, splitting, merging and spurious detections are shown in Table 4.15.

4.5.3 Text line and block extraction from characters within a PDL-based file

In this section, we report performance of the text line and text block extraction on document pages in the format of the Page Description Language (PDL), i.e. PostScript,



S030.Glyph

Volume 80
Number 1
1993

Annals
of the
Missouri
Botanical
Garden

MONOGRAPH OF THE NEOTROPICAL SPECIES OF *ASPLENIUM* SECT. *HYMENASPLENIUM* (ASPLENIACEAE)[1]

Noriaki Murakami[2] *and Robbin C. Moran*[3]

ABSTRACT

Asplenium sect. *Hymenasplenium* is one of the best defined groups of *Asplenium*, being characterized by creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure, and chromosome base numbers of $x = 38$ or 39. In the Neotropics, the section has ten species and three hybrids. The species are *A. delitescens*, *A. hoffmannii*, *A. laetum*, *A. obtusifolium*, *A. ortegae*, *A. purpurascens*, *A. repandulum*, *A. riparium*, *A. triquetrum*, and *A. volubile*. The hybrids are *A.* ×*papyraceum* (of unknown parentage), *A. delitescens* × *A. laetum*, and *A.* ×*incisoserratum* (= *A. hoffmannii* × *A. laetum*). All the aforementioned species and hybrids are endemic to the Neotropics. Central America and the Andes harbor the most species and endemics. A cladistic analysis was not done because the neotropical species of the section apparently do not form a monophyletic group separate from the paleotropical ones.

Section *Hymenasplenium* is one of the best defined groups within *Asplenium*, distinguished by the following synapomorphies: creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure and chromosome base numbers of $x = 38$ or 39. All other *Asplenium* species have erect or ascending rhizomes, radially symmetrical steles, nonswollen petiole bases, and $n = 36$ or multiples thereof (rare exceptions differ in only one of these characteristics).

Hymenasplenium was first described by Hayata

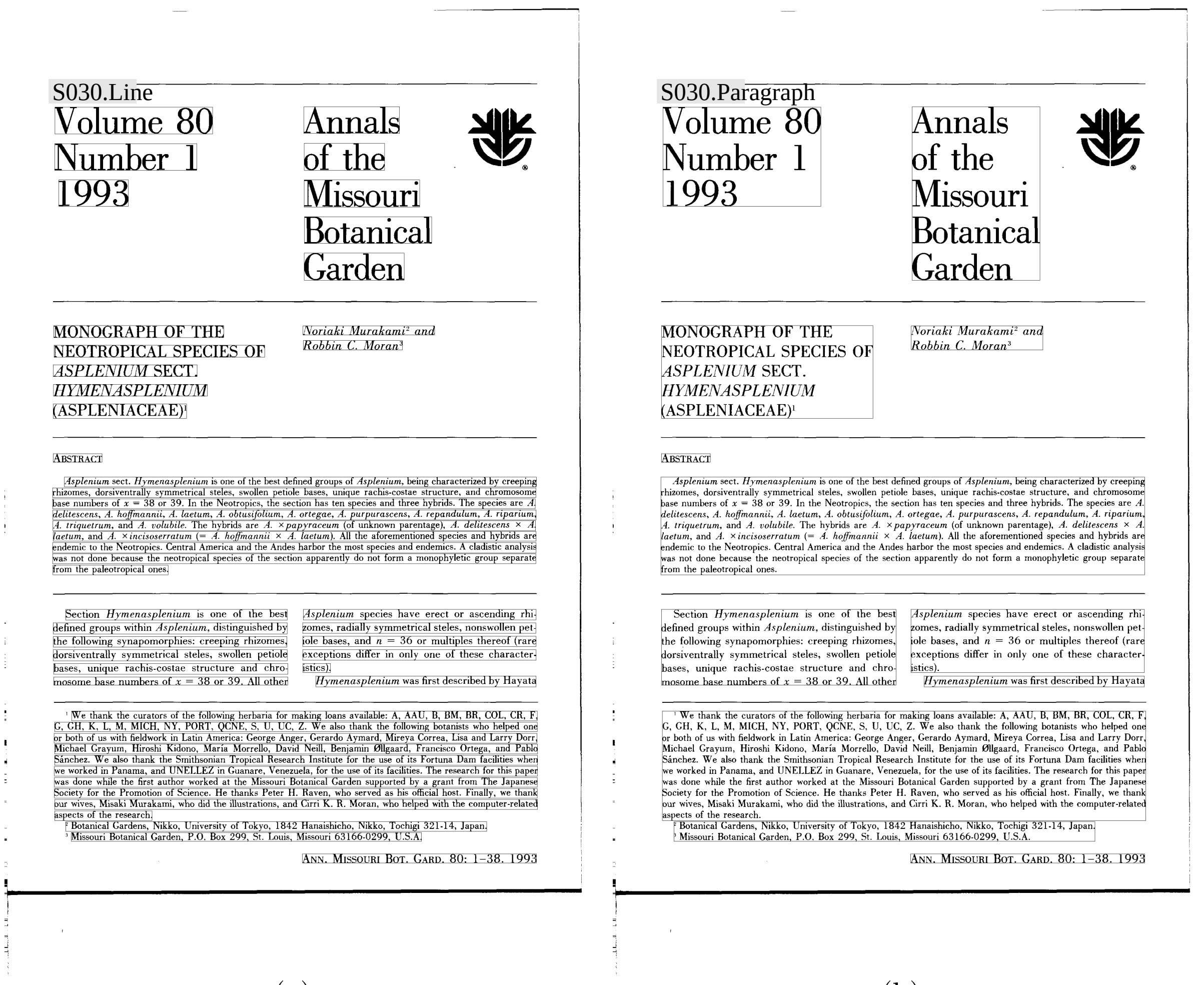
[1] We thank the curators of the following herbaria for making loans available: A, AAU, B, BM, BR, COL, CR, F, G, GH, K, L, M, MICH, NY, PORT, QCNE, S, U, UC, Z. We also thank the following botanists who helped one or both of us with fieldwork in Latin America: George Anger, Gerardo Aymard, Mireya Correa, Lisa and Larry Dorr, Michael Grayum, Hiroshi Kidono, Maria Morrello, David Neill, Benjamin Øllgaard, Francisco Ortega, and Pablo Sánchez. We also thank the Smithsonian Tropical Research Institute for the use of its Fortuna Dam facilities when we worked in Panama, and UNELLEZ in Guanare, Venezuela, for the use of its facilities. The research for this paper was done while the first author worked at the Missouri Botanical Garden supported by a grant from The Japanese Society for the Promotion of Science. He thanks Peter H. Raven, who served as his official host. Finally, we thank our wives, Misaki Murakami, who did the illustrations, and Cirri K. R. Moran, who helped with the computer-related aspects of the research.

[2] Botanical Gardens, Nikko, University of Tokyo, 1842 Hanaishicho, Nikko, Tochigi 321-14, Japan.

[3] Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166-0299, U.S.A.

ANN. MISSOURI BOT. GARD. 80: 1–38. 1993

Figure 4.27: Illustrates a real document image overlaid with the bounding boxes of the glyphs identified by an OCR system.



S030.Line

Volume 80
Number 1
1993

Annals of the Missouri Botanical Garden

MONOGRAPH OF THE NEOTROPICAL SPECIES OF *ASPLENIUM* SECT. *HYMENASPLENIUM* (ASPLENIACEAE)[1]

Noriaki Murakami[2] *and Robbin C. Moran*[3]

ABSTRACT

Asplenium sect. *Hymenasplenium* is one of the best defined groups of *Asplenium*, being characterized by creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure, and chromosome base numbers of $x = 38$ or 39. In the Neotropics, the section has ten species and three hybrids. The species are *A. delitescens*, *A. hoffmannii*, *A. laetum*, *A. obtusifolium*, *A. ortegae*, *A. purpurascens*, *A. repandulum*, *A. riparium*, *A. triquetrum*, and *A. volubile*. The hybrids are *A.* ×*papyraceum* (of unknown parentage), *A. delitescens* × *A. laetum*, and *A.* ×*incisoserratum* (= *A. hoffmannii* × *A. laetum*). All the aforementioned species and hybrids are endemic to the Neotropics. Central America and the Andes harbor the most species and endemics. A cladistic analysis was not done because the neotropical species of the section apparently do not form a monophyletic group separate from the paleotropical ones.

Section *Hymenasplenium* is one of the best defined groups within *Asplenium*, distinguished by the following synapomorphies: creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure and chromosome base numbers of $x = 38$ or 39. All other *Asplenium* species have erect or ascending rhizomes, radially symmetrical steles, nonswollen petiole bases, and $n = 36$ or multiples thereof (rare exceptions differ in only one of these characteristics).

Hymenasplenium was first described by Hayata

[1] We thank the curators of the following herbaria for making loans available: A, AAU, B, BM, BR, COL, CR, F, G, GH, K, L, M, MICH, NY, PORT, QCNE, S, U, UC, Z. We also thank the following botanists who helped one or both of us with fieldwork in Latin America: George Anger, Gerardo Aymard, Mireya Correa, Lisa and Larry Dorr, Michael Grayum, Hiroshi Kidono, María Morrello, David Neill, Benjamin Øllgaard, Francisco Ortega, and Pablo Sánchez. We also thank the Smithsonian Tropical Research Institute for the use of its Fortuna Dam facilities when we worked in Panama, and UNELLEZ in Guanare, Venezuela, for the use of its facilities. The research for this paper was done while the first author worked at the Missouri Botanical Garden supported by a grant from The Japanese Society for the Promotion of Science. He thanks Peter H. Raven, who served as his official host. Finally, we thank our wives, Misaki Murakami, who did the illustrations, and Cirri K. R. Moran, who helped with the computer-related aspects of the research.

[2] Botanical Gardens, Nikko, University of Tokyo, 1842 Hanaishicho, Nikko, Tochigi 321-14, Japan.

[3] Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166-0299, U.S.A.

ANN. MISSOURI BOT. GARD. 80: 1–38. 1993

(a)

S030.Paragraph

Volume 80
Number 1
1993

Annals of the Missouri Botanical Garden

MONOGRAPH OF THE NEOTROPICAL SPECIES OF *ASPLENIUM* SECT. *HYMENASPLENIUM* (ASPLENIACEAE)[1]

Noriaki Murakami[2] *and Robbin C. Moran*[3]

ABSTRACT

Asplenium sect. *Hymenasplenium* is one of the best defined groups of *Asplenium*, being characterized by creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure, and chromosome base numbers of $x = 38$ or 39. In the Neotropics, the section has ten species and three hybrids. The species are *A. delitescens*, *A. hoffmannii*, *A. laetum*, *A. obtusifolium*, *A. ortegae*, *A. purpurascens*, *A. repandulum*, *A. riparium*, *A. triquetrum*, and *A. volubile*. The hybrids are *A.* ×*papyraceum* (of unknown parentage), *A. delitescens* × *A. laetum*, and *A.* ×*incisoserratum* (= *A. hoffmannii* × *A. laetum*). All the aforementioned species and hybrids are endemic to the Neotropics. Central America and the Andes harbor the most species and endemics. A cladistic analysis was not done because the neotropical species of the section apparently do not form a monophyletic group separate from the paleotropical ones.

Section *Hymenasplenium* is one of the best defined groups within *Asplenium*, distinguished by the following synapomorphies: creeping rhizomes, dorsiventrally symmetrical steles, swollen petiole bases, unique rachis-costae structure and chromosome base numbers of $x = 38$ or 39. All other *Asplenium* species have erect or ascending rhizomes, radially symmetrical steles, nonswollen petiole bases, and $n = 36$ or multiples thereof (rare exceptions differ in only one of these characteristics).

Hymenasplenium was first described by Hayata

[1] We thank the curators of the following herbaria for making loans available: A, AAU, B, BM, BR, COL, CR, F, G, GH, K, L, M, MICH, NY, PORT, QCNE, S, U, UC, Z. We also thank the following botanists who helped one or both of us with fieldwork in Latin America: George Anger, Gerardo Aymard, Mireya Correa, Lisa and Larry Dorr, Michael Grayum, Hiroshi Kidono, María Morrello, David Neill, Benjamin Øllgaard, Francisco Ortega, and Pablo Sánchez. We also thank the Smithsonian Tropical Research Institute for the use of its Fortuna Dam facilities when we worked in Panama, and UNELLEZ in Guanare, Venezuela, for the use of its facilities. The research for this paper was done while the first author worked at the Missouri Botanical Garden supported by a grant from The Japanese Society for the Promotion of Science. He thanks Peter H. Raven, who served as his official host. Finally, we thank our wives, Misaki Murakami, who did the illustrations, and Cirri K. R. Moran, who helped with the computer-related aspects of the research.

[2] Botanical Gardens, Nikko, University of Tokyo, 1842 Hanaishicho, Nikko, Tochigi 321-14, Japan.

[3] Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166-0299, U.S.A.

ANN. MISSOURI BOT. GARD. 80: 1–38. 1993

(b)

Figure 4.28: Illustrates the results of text line and text block extraction, given the glyph bounding boxes identified by an OCR system.

Table 4.14: Performance of text line extraction algorithm on glyphs detected by a OCR system.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	105020	103989 (99.02%)	261 (0.25%)	517 (0.49%)	219 (0.21%)	34 (0.03%)
Detected	104975	103989 (99.06%)	571 (0.54%)	258 (0.25%)	9 (0.01%)	148 (0.14%)

Table 4.15: Performance of text block extraction on OCR produced characters.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	21788	18010 (82.66%)	1545 (7.09%)	1576 (7.23%)	60 (0.28%)	597 (2.74%)
Detected	23054	18010 (78.12%)	3659 (15.87%)	686 (2.98%)	0 (0.00%)	699 (3.03%)

Portable Document Format (PDF). Although PDF/PS is already a “text rich” electronic format, the encoding of distinct functional page components is not explicit in PDF/PS and their reliable identification requires similar approaches used in document structure analysis as are applied to scanned documents. PDF/PS provides bounding box information for components in a fashion which is compatible with the connected components to a great degree.

Our motivations for beginning with PDF/PS is based on the fact that there exists a wealth of documentation which is either already stored in PDF/PS or easily converted to it. Being able to migrate this material into a more workable, normalized, and semantically rich form such as SGML/XML is highly desirable. A great benefit of beginning with PDF/PS, in contrast with scanned images, is that with PDF/PS all text characters are explicitly and unambiguously defined, thus "accuracy" for actual textual content is 100%. It is the identification and delimitation of the structural and typographic characteristics which is the challenge.

In the UW-I document image database, there are 168 synthetic document images generated from the LaTeX files. The LaTeX software compiles an unformatted input file and produces formatted document pages in the DVI format, which is also a page description language. In a DVI file, the physical location and appearance of each glyph has been determined. We developed a tool to extract the bounding box of each glyph and its font information from the DVI files.

Given the bounding boxes of the glyphs on a document page, we apply the text line extraction algorithm to group glyphs into text lines, then our text block extraction algorithm is used to identify the text blocks. The bounding boxes of extracted text blocks are compared to the ground truth boxes available in the UW database. (The ground truth information of text line bounding boxes is not available). The performance is given in Table 4.16.

Table 4.16: Performance of text block extraction algorithm applied on the glyphs produced from the LaTeX files.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	1705	1526	93	67	0	19
		(89.50%)	(5.46%)	(3.93%)	(0.00%)	(1.11%)
Detected	1801	1526	222	28	0	25
		(84.73%)	(12.33%)	(1.56%)	(0.00%)	(1.39%)

4.6 Summary

In this chapter, we formulate the document segmentation as a partitioning problem. The goal of the problem is to find an optimal solution to partition the set of glyphs on a given document to a hierarchical tree structure where entities within the hierarchy are associated with their physical properties and semantic labels. A unified approach is proposed. A Bayesian framework is used to assign and update the probabilities during the segmentation. An iterative, relaxation like method is used to find the partitioning solution that maximizes the joint probability.

A text line extraction algorithm has been implemented to demonstrate the usage of this framework. This algorithm consists of two major components – off-line statistical training and on-line text line extraction. The probabilities used within this algorithm are estimated from an extensive training set of various kinds of measurements of distances between the terminal and non-terminal entities with which the algorithm works. The off-line probabilities estimated in the training then drive all decisions in the on-line segmentation module. The on-line segmentation module first extracts and filters the set of connected components of the input image to obtain a set of glyphs. Each glyph is linked to its adjacent neighbor to form glyph pairs. Associated with each link is the pair's linking probability. The entire text line extraction process can be viewed as an iterative re-adjustment of the pairs' linking probabilities on the glyph set. The segmentation algorithm terminates when the decision can be made in favor for each link within the final set of text line segments.

The algorithm was tested on the 1600 pages of technical documents within the UW-III database. A total of 105,020 text lines within these pages, the algorithm exhibits a 99.8% accuracy rate. Currently, we are implementing a text block extraction algorithm, also using the proposed framework. This new algorithm is currently at the testing phrase and the preliminary result looks promising.

Chapter 5

RECOGNIZING SPECIAL SYMBOLS FROM THE DOCUMENT IMAGES AND OCR RESULTS

5.1 Introduction

Optical character recognition (OCR) is one of the most popular and successful applications of automatic pattern recognition [67]. Today, reasonably good OCR packages can be bought for as little as $100. However, the current OCR systems can only recognize characters within their fixed alphabets. There is no training tool available to allow users to add new symbols to the alphabet and update the classifier. In many documents, especially the scientific and technical documents, many special symbols (e.g. Greek letters, mathematical symbols, etc.) other than the regular ASCII characters present and need to be manually verified.

This chapter describes algorithms for automated segmentation and recognition of special symbols not handled by OCR systems. Given a document image and the OCR recognized text, the character segmentation refines the character coordinates. Then, the normal characters are distinguished from special symbols to be processed by the symbol recognition module. Finally, features are extracted from the special symbol subimages and a supervised classifier is used to assign the subimages to one of predefined categories.

The characters' positions within a text line are used as the features for matching recognized characters and the glyphs (connected components) on a document image. The glyphs' positions are determined by a classifier adaptively trained within the same text zone. For each character string (with length up to three), we compute its probability being a special symbol. The observations are the character string, the sequence of confidence levels, and the context information. A list of special symbol candidates are also produced at this stage. A probability map for each special symbol is computed from the training samples. Given an input symbol image, we compute the distance from the image to each of the trained probability maps. Using the probabilities provided by the detection stage as the a prior probability, we use a Bayesian framework to update the probability of each candidate.

This chapter is organized as follows. The problem statement and the system overview are given in Section 5.2. Section 5.3 describes a character segmentation method that refines the character segmentation output provided by the OCR engines. A method to distinguish normal characters from special symbols is presented in Section 5.4. The feature extraction and special symbol classification method is described in Section 5.5. The experimental results are given in Section 5.6.

5.2 Problem Statement

This section formalizes the special symbol recognition problem as finding the special symbols that maximizes the Bayesian probability of the special symbols by observing the OCR generated output and the features from the glyph images.

Let $\hat{X}$ denote the recognized information by the OCR systems from a document image I. Let G be a set of glyphs on image I. Let S be the set of special symbols on I. The problem of special symbol recognition can be formulated as follows: *Given a set of glyphs G from a document image, and the recognized characters $\hat{X}$ produced by an OCR system, detect the special symbols S within the alphabet Λ, that maximizes the conditional probability $P(S|\hat{X}, G)$.*

The conditional probability $P(S|\hat{X}, G)$ can be evaluated as:

$$P(S|\hat{X}, G) = \frac{P(\hat{X}, G|S)P(S)}{P(\hat{X}, G)}$$

$$= \frac{P(\hat{X}|S)P(G|S)P(S)}{P(\hat{X})P(G)}. \quad (5.1)$$

The system is divided into three major modules: character segmentation, special symbol detection, and special symbol classification. The system process diagram is shown in Figure 5.1.

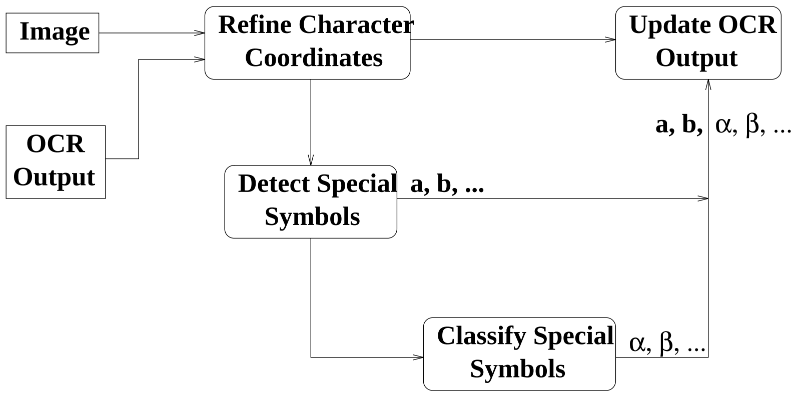


Figure 5.1: Illustrates the processing steps of the special symbol recognition system.

First, the connected components and their bounding boxes are extracted from a document image. Suppose that the coordinates of the text lines and words are given, we match the connected component bounding boxes with the text lines and words. Now, the problem becomes finding the correspondence between a group of component bounding boxes enclosed by a word box, with a group of characters belonging to the same word. The relative position of components/characters in the text lines are used as clue to find the best match. We estimate the baseline and x-height for each text line, and compute the relative location of each component with respect to the baseline and x-height line. Each component is assigned a position class according to its position features. The position class of a character is known once the character is given. Then, we can match the characters with components by finding the correspondence between their position class strings. Finally, we merge and split the connected components and assign their bounding boxes to the corresponding characters.

A special symbol is usually recognized as one or more certain characters by an OCR system. The recognized characters are usually associated with low recognition confidence levels. The special symbol detection module detects the possible special symbols from the OCR generated output. Given a text string, the recognition confidence level associated with each character, and the string's left and right neighbors, this module computes the posterior probability of this text string being a special symbol instead, and outputs a list of possible special symbol candidates for the further classification.

For each possible special symbol candidate produced by the identification step, the classification module computes features from the glyph(s), and assign the glyph(s) into one of predefined categories.

5.3 Character Segmentation Refinement

The objective of the character segmentation module is to refine the coordinates of all symbols provided by an OCR system. Specifically, the module inputs the coordinate data and any other necessary information from the OCR system, and outputs the refined coordinates of all symbols.

Problem Statement: Given a sequence $A = (a_1, \cdots, a_N)$ of characters and a sequence $B = (b_1, \cdots, b_M)$ of connected components computed from the image, merge and split the elements of B into a sequence $G = (g_1, \cdots, g_N)$ of glyphs and each element of G being associated with a character

$$[(g_1, a_1), (g_2, a_2), \cdots, (g_N, a_N)],$$

that minimizes the criterion

$$D(F(S), F(G)),$$

where F represents the features of the symbols and glyphs, and D is the distance measure between two sequences of features.

We first extract connected components from the input image. A vertical grouping process is performed in order to merge the characters that are vertically separated, such as the dots of “i” and “j”. Then, we compute the rectangular bounding box for each component.

The next step aims to find the match between the extracted components and their corresponding text segments, e.g. text blocks, text lines and words. The coordinates of these text entities are usually provided by the OCR systems. Otherwise, a document page segmentation is needed to group the connected components into text lines, words, and text blocks [58]. The area overlap between the connected component bounding box and the text entity box is used as the metric for matching.

Algorithm 5.1 *Match words with connected components*

```
For each word $W = (a_1, \cdots, a_N)$ in the document

  1. Find the connected components that have the maximum area overlap with the
     word box. Let $B = (b_1, \cdots, b_M)$ be the sequence of components enclosed by the
     word box.

  2. If $N = M$, Then

     (a) Assign the bounding box of each component $b_i$ to its corresponding symbol
         $a_i$.

     (b) Collect the statistics from the matched components and symbols, and up-
         date a classifier.

  3. Else, While $N \neq M$, Do

     (a) compute a feature sequence $F(B) = (F(b_1), \cdots, F(b_M))$ from the compo-
         nents.
```

(b) Compute a feature sequence $F(W) = (F(a_1), \cdots, F(a_N))$ from the symbols.

(c) Compute the distance $D(F(B), F(W))$ between $F(B)$ and $F(W)$.

(d) Adjust the component sequence B by minimizing the distance $D(F(B), F(W))$.

End

Assign each connected component bounding box to its matched character.

5.3.1 Extraction of position features

Within each word box, a character might correspond to one connected component (one-to-one match), more than one component (one-to-many match), or part of a component (many-to-one match). We use the relative position of characters from the baseline and the x-height line as the clue for matching characters with components. The baseline and x-height line of a text line is shown in Figure 5.2.

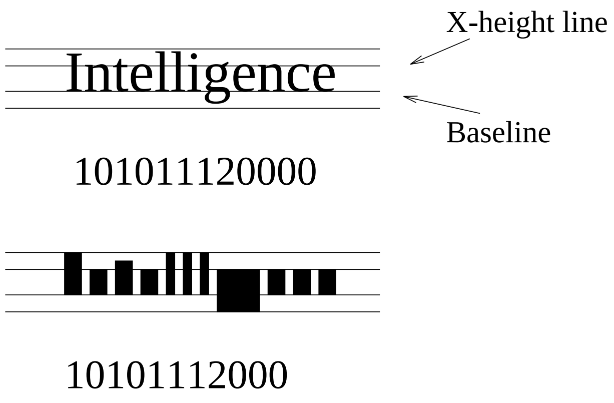


Figure 5.2: Illustrates the position features of characters and glyphs within a word.

A connected component is represented by its bounding box (x_1, y_1, x_2, y_2), where (x_1, y_1) is the coordinate of the top-left corner and (x_2, y_2) is the bottom-right corner. Given the bottom-right corners of the components within a text line, we use a robust line-fitting algorithm [58] to estimate the text line's baseline coordinates. Then, the x-height is estimated from the distance of the components' top-left corners to the baseline. Next, we compute a feature string for each word and a feature string for the connected components enclosed by that word, using the detected baseline and the x-height. These feature strings are then matched with each other to decide which connected components correspond to which characters.

We use a feature string to represent each word. A character is assigned a class label according to its position in the text line. The character position class is defined as follows:

- Class 0: character within baseline and x-height, e.g. a, c, e, o.
- Class 1: character that extends above x-height (ascender), e.g. b, A, C, P.
- Class 2: character that extends below baseline (descender), e.g., p, g, q
- Class 3: character that extends above x-height and below baseline, e.g. {, }.
- Class 4: superscript or higher punctuation mark.
- Class 5: subscript or lower punctuation mark.

Therefore, for the above example of the word "Intelligence", the word feature string is 101011120000.

For each component $b_i = (x_{1i}, y_{1i}, x_{2i}, y_{2i})$ within a text line, we first compute its relative distance from the text-line's baseline and x-height. Let d_{xi} denote the distance from (x_{1i}, y_{1i}) to the x-height line. Let d_{bi} be the distance from (x_{2i}, y_{2i}) tothe baseline. Then, we assign a position class label to the component b_i according to the computed feature vector (d_{xi}, d_{bi}). The feature string for the connected components in Figure 5.2 will be 10101120000. Note that the character g and e are merged as one component.

After computing the relative location of each connected component from the baseline and x-height, we need to decide the position class of the component. Instead of using global thresholds, we build an adaptive classifier for each text zone. First, for all words that we have found the matching components, we determine the position class for each character and collect the position feature values for each class. A binary tree classifier is adaptively trained given the computed position features and their known position classes. At each node of the classification tree, we search for the best threshold values of the decision rule by minimizing the number of misclassification errors.

Within each text zone, besides the parameters for determining the position of a glyph within a text line, we also record the width of each character. The width information will be used as the constraint when splitting the components.

5.3.2 Matching connected components with characters

In this step, we determine the match between characters within a word and connected components enclosed by the word's bounding box. If a perfect match can be found, the bounding box of each connected component is assigned to its corresponding character. Otherwise, the matching could be one-to-many or many-to-one. An iterative search method is performed to find the correspondence. At each iteration, if the number of characters is larger than the number of components, we merge the characters pair-wise and assign the combined position class to the merged character pair. For example, a class 0 character merged with a class 1 character will produce class 1. If the number of characters is less than the number of components, we merge the components pair-wise and re-compute the position feature and assign position class to the newly generated component. We search through all possible pair-wise combinations and select the one that maximizes the length of the *longest common subsequence* between two position strings. We continue the iteration until the number of characters and components match. After the iterative matching, we can determine the correspondence between characters and connected components within each word. In our example, characters ‘g” and “e” are found to be merged into one connected component.

Given two sequences X and Y, we say that a sequence Z is a common sequence of X and Y if Z is a subsequence of both X and Y. In the longest common subsequence problem, we are given two sequences $X = (x_1, x_2, \cdots, x_m)$ and $Y = (y_1, y_2, \cdots, y_n)$ and wish to find a maximum-length common subsequences of X and Y. The LCS problem can solved efficiently using dynamic programming [63].

Algorithm 5.2 *Iterative character matching by maximizing the length of LCS*

```
1. Given a sequence $A = (a_1, \cdots, a_N)$ of $N$ characters and a sequence $B = (b_1, \cdots, b_M)$ of $M$ components,

2. While $N \neq M$, Do

   (a) If $N > M$, For $i = 1$ to $N - 1$ Do,

         i. Let $\hat{A}_i = (A - a_i - a_{i+1}) \bigcup \hat{a}_i)$, where $\hat{a}_i = a_i \bigcup a_{i+1}$.
        ii. Compute the position string $F(\hat{A}_i)$
       iii. Compute $LCS(F(\hat{A}_i), F(G))$

       End

       Search for the adjustment that maximizes $LCS(F(\hat{A}_i), F(B))$

           $k = \arg\max_i(LCS(F(\hat{A}_i), F(B)))$

       Let $A = \hat{A}_k$ and $N = N - 1$.

   (b) Else If $N < M$, For $i = 1$ to $M - 1$ Do,

         i. Let $\hat{B}_i = (B - b_i - b_{i+1}) \bigcup \hat{b}_i)$, where $\hat{b}_i = b_i \bigcup b_{i+1}$.
        ii. Compute the position string $F(\hat{B}_i)$
       iii. Compute $LCS(F(A), F(\hat{B}_i))$
```

```
        End
        Search for the adjustment that maximizes $LCS(F(A), F(\hat{B}_i))$
                $k = \arg\max_i (LCS(F(A), F(\hat{B}_i)))$
        Let $B = \hat{B}_k$ and $M = M - 1$.
    End
```

The above algorithm determines the correspondence between the sequence of characters and the sequence of components: one-to-one match, one-to-many match, and many-to-one match. If a character is one-to-many matched with a group of components, the components are merged into one glyph. If a group of characters are many-to-one matched to one component, we split the component into glyphs, such that each glyph is associated with a character. Merging of two or more connected components into one character is straightforward. To split a connected component that corresponds to two or more characters, we first compute the projection profile of the foreground pixels enclosed by the bounding box of the component. The cutting position should be one of the local minimums of profile. To determine which minimum to choose, we again use the context information. Based on the assumption that the character font size and style do not vary much within a text zone, we use the width statistics of the characters, which have been perfectly matched with a component, as the constraint when finding the best cutting position. The computed statistics are the minimum, mean, and maximum of the width for each character. Then, we use the information to determine the approximate range of width for the current character and select a local minimum within this range.

Figure 5.3(a) illustrates characters and their bounding boxes produced by an OCR system. The computed connected component bounding boxes are shown in Figure 5.3(b). Note that the characters “r” and “y” are merged as one component.

So do the characters “K” and “i”. The character bounding boxes after the segmentation refinment are shown in Figure 5.3(c).

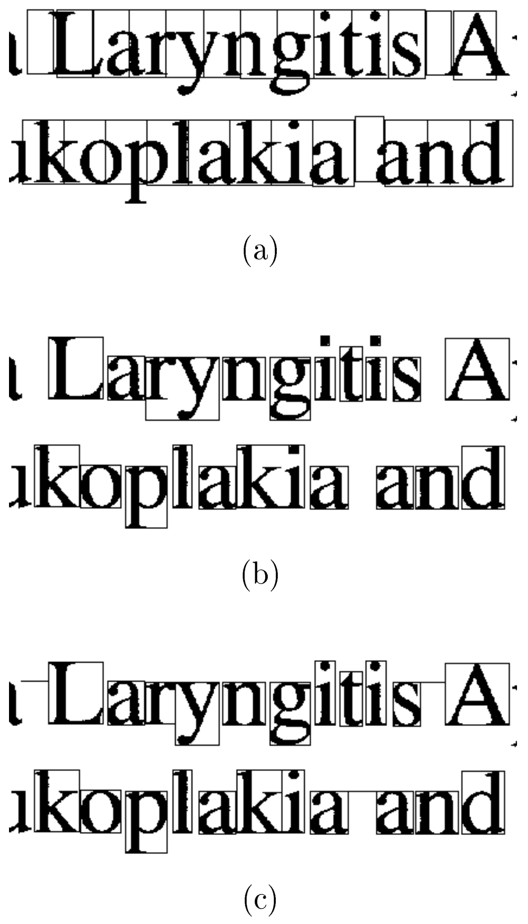


Figure 5.3: Illustrates the character bounding boxes (a) before segmentation; (b) from connected components; (c) after segmentation refinement.

5.4 Special Symbol Detection

Given a text string (with maximum length 3), the recognition confidence level associated with each character, and the string’s left and right neighbors (with length 1), the special symbol detection computes the posterior probability of this text string being a special symbol instead, and outputs a list of possible special symbol candidates for the further classification.

5.4.1 Problem statement

Let the observed character be $x = (a, c)$, where a is the letter and c is the OCR confidence level. Let the observed character string be $X = x_1 x_2 \cdots x_m$, where $m \leq 3$. The probability that a special symbol $s \in S$ has caused X can be expressed by the use of Bayes rule

$$P(s|X) = \frac{P(X|s)P(s)}{P(X)}, \tag{5.2}$$

where $P(X|s)$ is the probability of observing X under the condition that s is the right symbol. $P(s)$ is the a priori probability of s and $P(X)$ is the probability of the character sequence X.

In the training step, for each given sequence of characters and their confidence levels, the probability that this sequence of characters is indeed a certain special symbol is calculated. A probability look-up table is constructed for all special symbols $s \in S$.

Due to the insufficient size of training data (only a couple of special symbols per page), the computed probability table is fairly sparse and causes high misdetection rate. The purpose of the following two steps is to improve the generalization of the detector.

5.4.2 Weighted combination of character and confidence

This step is based on the assumption that most of the special symbols are recognized as normal characters with low confidence levels. Therefore, we still want to assign a small probability of being special symbols to a sequence of characters with low confidence; even if this character sequence does not appear in the probability table computed from the training data.

Let the observed confidence sequence be $C = c_1 c_2 \cdots c_m$. The probability that a special symbol $s \in S$ has caused C can be expressed by the use of Bayes rule

$$P(s|C) = \frac{P(C|s)P(s)}{P(C)}, \tag{5.3}$$

where $P(C|s)$ is the probability of observing C under the condition that s is the right symbol. $P(s)$ is the a priori probability of s and $P(C)$ is the probability of the confidence sequence C.

We combine the probability $P(s|X)$ and $P(s|C)$ using a linear weighted combination:

$$P(s|X,C) = \frac{cm}{cm+1} P(s|X) + \frac{1}{cm+1} P(s|C), \tag{5.4}$$

where cm is the average confidence level of the given sequence of confidence levels.

5.4.3 Data smoothing

Compared to polynomial regression, data smoothing relies on the data to specify the form of the model. Data smoothing is to fit a curve to the data points locally, so that at any point the curve at that point depends only on the observations at that point and some specified neighboring points. Because such a fit produces an estimate of the response that is less variable than the original observed response, the result is called a smooth, and procedures for producing such fits are called scatterplot smoothers.

We use the kernel smoother to smooth the data in order to improve the generalization of the special symbol detector. A kernel-type smoother is a type of local average smoother that, for each target point x_i in predicator space, calculates a weighted average $\hat{y}_i$ of the observations in a neighborhood of the target point:

$$\hat{y}_i = \sum_{j=1}^{n} w_{ij} y_j \tag{5.5}$$

where

$$w_{ij} = \tilde{K}(\frac{x_i - x_j}{b}) = \frac{K(\frac{x_i - x_j}{b})}{\sum_{k=1}^{n} K(\frac{x_i - x_k}{b})} \tag{5.6}$$

are weights which sum to one [65].

The function K used to calculate the weights is called a kernel function, which typically has the following properties:

- $K(t) \geq 0$ for all t
- $\int_{-\infty}^{\infty} K(t)dt = 1$
- $K(-t) = K(t)$ for all t.

The parameter b is the bandwidth parameter, which determines how large a neighborhood of the target point is used to calculate the local average. The following is a list of possible kernels:

- Box:
$$K_{box}(t) = \begin{cases} 1, |t| \leq 0.5 \\ 0, |t| > 0.5 \end{cases} \tag{5.7}$$
- Triangle:
$$K_{tri}(t) = \begin{cases} 1 - \frac{|t|}{C} & |t| \leq \frac{1}{C} \\ 0 & |t| > \frac{1}{C} \end{cases} \tag{5.8}$$
- Normal:
$$K_{nor}(t) = \frac{1}{\sqrt{2\pi}k} \exp\left(-t^2/2k^2\right). \tag{5.9}$$

The constants shown in the forms above are used to scale the resulting kernel so that the upper and lower quartiles occur at $\pm.25$. We use "triangle" because it is the simplest and fastest to calculate.

5.4.4 Context information

The context information is also very useful to determine if a sequence of characters is actually a special symbol. For example, the special symbol "γ" is often recognized as

"y" with relatively high confidence. It causes posterior probability $P(s = \gamma | a = y, c)$ to be very low and the symbol "γ" is missed. If we can observe from the data that the probability of "γ" followed by "-" is larger than the probability of character "y" followed by "-", this context information can be used to update the posterior probability and still catch the symbol "γ".

Let the observed character string before X be $X^- = x_1^- x_2^- \cdots x_n^-$. Let the observed character string after X be $X^+ = x_1^+ x_2^+ \cdots x_p^+$. We use $n = p = 1$. The probability that a special symbol $s \in S$ has caused X can again be expressed by the use of Bayes rule as follows:

$$\begin{aligned} & P(X^-, s, X^+ | X^-, X, X^+) \\ = & \frac{P(X^-, X, X^+ | X^-, s, X^+) P(X^-, s, X^+)}{P(X^-, X, X^+)}. \end{aligned} \tag{5.10}$$

Based on the assumption of conditional independence among X^-, X, and X^+,

$$\begin{aligned} & P(X^-, X, X^+ | X^-, s, X^+) \\ = & P(X^- | X^-) P(X | s) P(X^+ | X^+) \\ = & C \times P(X | s). \end{aligned} \tag{5.11}$$

where C is a constant. Therefore, the probability (5.10) is approximated as

$$\begin{aligned} & P(X^-, s, X^+ | X^-, X, X^+) \\ \propto & \frac{P(X|s) P(X^-, X^+ | s) P(s)}{P(X^-, X^+ | X) P(X)} \\ = & P(s|X) \frac{P(X^-, X^+ | s)}{P(X^-, X^+ | X)} \end{aligned} \tag{5.12}$$

where $P(X^-, X^+ | s)$ is the probability that, given a glyph is a special symbol s, its left neighbor is X^- and its right neighbor is X^+. And $P(X^-, X^+ | X)$ is the probability that, given an observed character sequence X, its left neighbor is X^- and its right neighbor is X^+.

5.5 Special Symbol Classification

This section describes the special symbol feature extraction and classification. The special symbol classification extracts descriptive features, which are independent of the sizes, font types and font styles of the symbols. Furthermore, these features will be sufficiently characteristic of the symbols to yield a very high degree of recognition accuracy. Then, a supervised classifier is used to assign an input symbol image into one of predefined categories, based on extracted feature vectors.

Problem Statement: Given a glyph g, compute features $F(g)$ from the sub-image of g, classify g into one of predefined categories $s \in S$, that maximizes the conditional probability $P(s|F(g))$.

The probability $P(s|F(g))$ can be evaluated as

$$P(s|F(g)) = \frac{P(F(g)|s)P(s)}{P(F(g)} \tag{5.13}$$

where $P(s)$ is the prior probability of special symbol s. We use the probabilities provided by the detection stage as the a prior.

We developed three sets of features: geometric moment invariants, Zernike moments, and probability maps. Each feature set is evaluated on the special symbol classification and the probability maps produce the best results.

5.5.1 Geometric moment invariants

Given a gray-scale subimage Z containing a symbol candidate, the regular moments of order $(p+q)$ are defined as:

$$m_{pq} = \sum_{i=1}^{N} Z(x_i, y_i)(x_i)^p(y_i)^q, \tag{5.14}$$

where the sum is taken over all the M pixels in the subimage. A binary image can be considered a special case of a gray-level image with $Z(x,y) = 1$ for foreground pixels and $Z(x,y) = 0$ for background pixels. The translation-invariant central moments of order $(p+q)$ are obtained by placing origin at the center of gravity:

$$\mu_{pq} = \sum_{i}^{M} Z(x_i, y_i)(x_i - \bar{x})^p (y_i - \bar{y})^q, \tag{5.15}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}. \tag{5.16}$$

Moment invariants are invariant to size and rotation, and some moment invariants are also invariant to skew and mirror images. For general linear transformation, Hu [70] and Reiss [68] gave the relative invariants that are functions of the second- to fourth-order central moments. We use these 9 moments as the features for symbol classification.

5.5.2 Zernike moments

The zernike moments are projections of the input image onto the space spanned by a set of orthogonal functions:

$$V_{nm} = R_{nm}(x, y) e^{jm \tan^{-1}(y/x)}, \tag{5.17}$$

where $j = \sqrt{-1}$, $n \geq 0$, $|m| \leq n$, $n - |m|$ is even, and

$$R_{nm}(x, y) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (x^2 + y^2)^{(n/2)-s} (n-s)!}{s! (\frac{n+|m|}{2} - s)! (\frac{n-|m|}{2} - s)!} \tag{5.18}$$

For a digital image, the Zernike moment of order n and repetition m is given by:

$$A_{nm} = \frac{n+1}{\pi} \sum_{x} \sum_{y} f(x, y) [V_{nm}(x, y)]^{\star}. \tag{5.19}$$

where $x^2 + y^2 \leq 1$ and the symbol $\star$ denotes the complex conjugate operator. To compute the Zernike moments of a given image, the center of the image is taken as the origin and the pixel coordinates are mapped to the range of unit circle, i.e., $x^2+y^2 \leq 1$. Those pixels falling outside the unit circle are not used in the computation.

The magnitudes $|A_{nm}|$ are rotation invariant. The original image can be reconstructed using the Zernike moments. Translation- and scale-invariance can be obtained by shifting and scaling the image prior to the computation of the Zernike moments. The first-order regular moments can be used to find the image center and zeroth order central moment gives a size estimate.

In the experiment, we use the Zernike moments with order 12. The total number of moments is 7: $A_{12,0}, A_{12,2}, A_{12,4}, A_{12,6}, A_{12,8}, A_{12,10}, A_{12,12}$.

5.5.3 Probability maps

Probability maps are the distribution of the symbol foreground at each point of a $n \times m$ grid. The probability maps are robust to variability in font types and font styles of the symbols. Furthermore, these maps are sufficiently characteristic of the symbols to yield a very high degree of recognition accuracy.

The computation of probability maps is as follows.

Preprocessing

To achieve scale and translation uniformity, the regular moments (i.e., m_{pq}) of each image are utilized. Translation invariancy is achieved by transforming the image into a new one whose first order moments, m_{01} and m_{10}, are both equal to zero. This is done by transforming the original image $f(x, y)$ into another one which is $f(x + \bar{x}, y + \bar{y})$, where $\bar{x}$ and $\bar{y}$ are the centroid location of the original image computed from:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}.$$

Scale invariancy is accomplished by enlarging or reducing each image using its zeroth order moment. m_{00} is set equal to a predetermined value a. The original image function $f(x, y)$ is transformed into a new function $f(x/a, y/a)$, with $a = \sqrt{\beta/m_{00}}$. In summary, an image function $f(x, y)$ can be normalized with respect to scale and translation by transforming it into $g(x, y)$, where

$$g(x, y) = f(\frac{x}{a} + \bar{x}, \frac{y}{a} + \bar{y}). \tag{5.20}$$

The normalized image is re-sampled to a $n \times m$ grid. In the experiment, we choose both n and m as 17. We select a as 255 multiplied by half of the total number of pixels.

Compute probability maps

Given a set of normalized training samples, we compute the probabilities that a symbol produces foreground value (255) at each pixel, and generate a probability map for each symbol. In other words, each probability map is the projection profile of a special symbol's normalized images within the training set. Given the image I of a special symbol S_k, its probability map T_k is computed as

$$T_k(i, j) = T_k(i, j) + I(i, j) \tag{5.21}$$

The values of the probability map is then normalized to certain range, i.e. 0-255. The computed probability maps for a set of special symbols are shown in Figure 5.4.

Classification

Given a preprocessed image as input, with foreground as 255 and background as 0, we first normalize the image to $n \times m$ grid. Then, we compute the distance between the normalized input image from each of the trained probability maps, and assign the map with the smallest distance to the input image. The distance d between an input image I and a probability map T is defined as the sum of absolute difference:

$$d(I, T) = \sum_{i}^{N} \sum_{j}^{M} |I(i, j) - T(i, j)|. \tag{5.22}$$

Let $D = \{d_1, d_2, \cdots, d_i, \cdots, d_N\}$ denote the distance between an input image I and the trained probability maps $\{T_1, T_2, \cdots, T_N\}$. The likelihood that an input glyph I

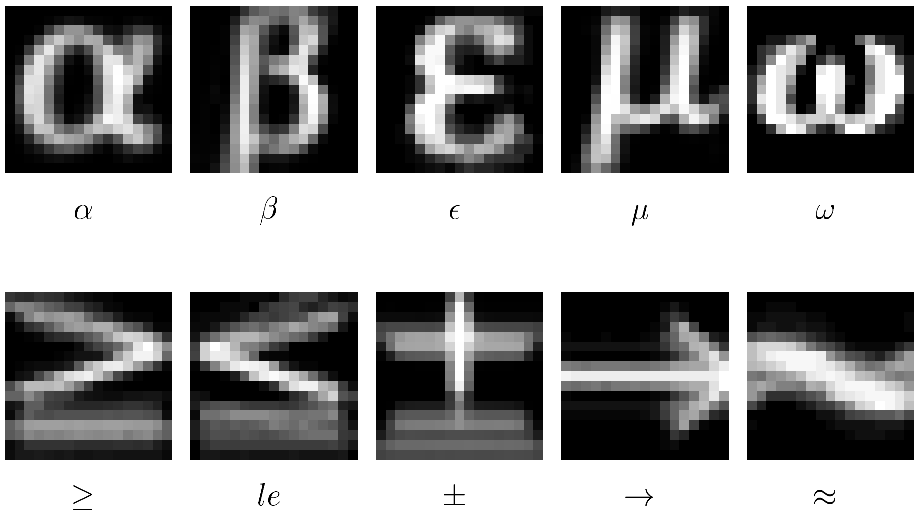


Figure 5.4: Illustrates the probability maps of some special symbols.

is indeed a special symbol s_i can be computed as

$$P(I|s_i) = \frac{1/d_i}{\sum_{k=1}^{N}(1/d_k)} \tag{5.23}$$

Using the probabilities provided by the detection stage as the a prior probability, we update the probability of each candidate by observing the image features. Give a sequence of OCR produced character-confidence pairs $X = (X_1, X_2, \cdots, X_N)$, where $X_i = (a_i, c_i)$, we search for the special symbols using the following algorithm.

Algorithm 5.3 *Special symbol recognition*

```
For $i = 1$ to $N$, Do
    1. For $j = 0$ to 2, Do
        (a) If $i + j > N$, Stop.
        (b) Let $\hat{X} = \bigcup_{k=i}^{i+j} X_k$.
        (c) Determine the left neighbor $X^- = X_{i-1}$ and the right neighbor $X^+ = X_{i+j+1}$.
```

(d) Compute the probability that $\hat{X}$ is a special symbol $s \in S$, as the multiplication of the probabilities in Equation 5.12 and 5.23,

$$P(X^-, s, X^+|X^-, \hat{X}, X^+) \propto P(I|s)P(s|\hat{X})\frac{P(X^-, X^+|s)}{P(X^-, X^+|\hat{X})}, \tag{5.24}$$

where I is the image associated with $\hat{X}$.

End

2. Select $\hat{X}$ that produces the maximum probability. If the probability is larger than a predetermined threshold, replace $\hat{X}$ by the special symbol s.

End

5.6 Experimental Results

At the National Library of Medicine, a system called the Medical Article Record System (MARS) is being developed and used for bibliographic indexing of medicine related literature. An OCR system is applied to each scanned document page and produces recognized text. Then, the OCR output is manually verified and corrected. The special symbols are inserted into the text. We use a total of 5516 pages from the NLM as the training and testing set to evaluate the special symbol recognition system. Each page in the data set consists of a pair of files, one is the OCR generated output, and the other contains the manually verified text. Therefore, we use the scanned image and the first file as the input of our system. The updated file produce by the system is compared against the manually edited file, which serves as the ground truth.

5.6.1 Segmentation

To evaluate the performance of the segmentation module, the detected bounding boxes must be compared with the ground truth bounding boxes. The correspondence between two sets of boxes: which boxes of the ground truth set correspond to which boxes of the automatically produced set, is determined based on an area overlap measure. Then the possible errors of misdetection, false alarm, splitting, and merging are detected for each box.

We manually ground-truthed 32 images, with total of 36394 character boxes, and used those images to evaluate the performance of the character segmentation module. The evaluation results are shown in the following table.

Table 5.1: Performance of the character segmentation algorithm.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	36394	36250 (99.61%)	43 (0.12%)	88 (0.24%)	1 (0.00%)	12 (0.03%)
Detected	36396	36250 (99.60%)	90 (0.25%)	42 (0.11%)	1 (0.00%)	13 (0.04%)

The table shows that 99.6% (36250) of ground truth character boxes (36394) have been correctly segmented. 43 boxes are split into total of 90 boxes, while 88 boxes are merged into total of 42 detected boxes. We found most of merging and splitting errors are due to the inaccurate classification of the character box position within the text line. We assume the information that whether a character is superscript or subscript is provided by the OCR output. Unfortunately, this information is not reliable. In order to improve the accuracy of character position classification, we will develop an algorithm to automatically determine whether a character is superscript or subscript.

5.6.2 Symbol detection

The data set with total of 5516 pages is divided into 5 subsets, and the cross validation method is used to estimate the performance of the symbol detection module. The total number of special symbols in the data set is 4046. The experimental results of special symbol detection are shown in Figure 5.5. The number of misdetections and false alarms are plotted corresponding to different threshold values. We started the experiments by only using the character information. Then the weighted combination is added. We smoothed the probability densities on the confidence levels to improve the generalization. Finally, we included the context information.

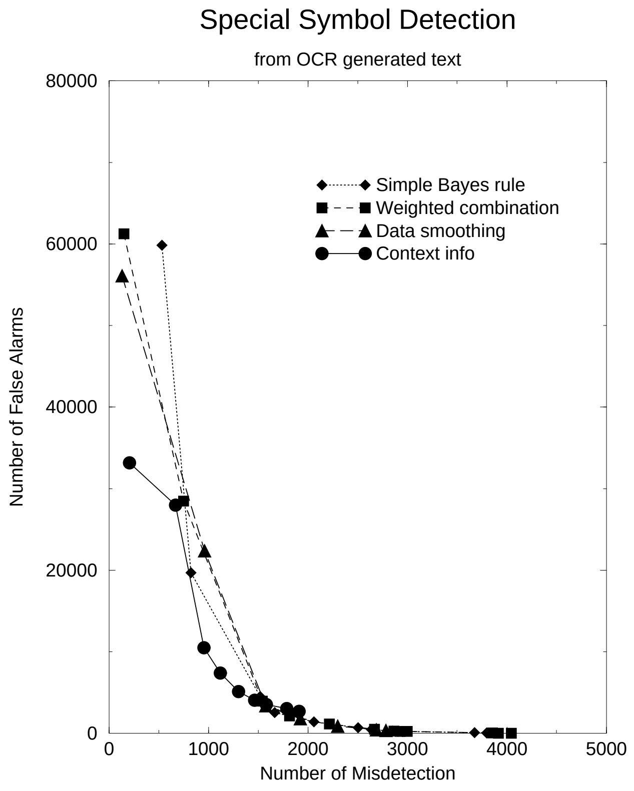


Figure 5.5: Plots the number of false alarms vs. the number of misdetections for detecting special symbols from OCR output.

A simple trigram table is used as the representation for the context information. We will experiment on estimating trigram probabilities by combining several group n-gram probabilities, rather than by direct lookup in a trigram table [66]. The purpose is to reduce the memory needed and to improve the generalization of the detector.

Features used	Number of correct detection
Moment invariants	3122 (82.3%)
Zernike moments	3297 (86.9%)
Probability maps	3630 (95.7%)

Table 5.2: Performance of the special symbol classification using different features.

For example, the string 7 ± 2 may have never occurred in the training samples even though there were many occurrences of strings of the form $< digit > \pm < digit >$.

5.6.3 Symbol classification

We selected 13 special symbols with relatively large number of samples in the data set. The selected symbols are: α, β, $\circ$ (Degree), δ, ϵ, γ, $\geq$, κ, $\leq$, μ, $\pm$, $\rightarrow$, $\approx$ (or $\sim$, $\simeq$, $\cong$). The total number of samples is 3794.

We use the decision tree classifier from S-PLUS [65] for the moment features and the nearest neighbor classifier for the probability maps. A 5-fold cross validation method is used to estimate the accuracy of classification. The experimental results for three sets of features are shown in Table 5.6.3.

Finally, we test the performance of the combination of two modules: locating special symbols and classification. The input is document images and the information generated by an OCR system. The character coordinates are supposed to have been refined by the segmentation module. The possible special symbols are detected by checking each character string from the text generated by OCR, then the symbol classifier is applied to the candidates. Classification module takes detection probabilities as a prior and updates them by observing the image features. If the confidence of the top choice is higher than a predetermined threshold, the character string will be replaced by the name of the special symbol.

The first step of evaluation is to find the correspondence between special symbols in the program generated output and the special symbols in the ground truth file. We use an area overlap measure to determine which boxes of the ground truth set correspond to which boxes of the automatically produced set. Then the possible errors of misdetection, false alarm, splitting and merging, and correct detection are detected for each box, by finding the one-to-zero, zero-to-one, one-to-many, many-to-one, and the one-to-one matches.

For the special symbols which have been correctly detected (one-to-one match), the second step of evaluation is to determine the rate that they are correctly classified, by comparing the labeled category with the ground truth symbol name. The correct classification rate is used as the performance measure.

The misdetection/false alarm rate using different threshold values is plotted in Figure 5.6(a). The correct classification rate versus the threshold values are shown in Figure 5.6(b).

5.7 Summary

This chapter describes algorithms for automated segmentation and recognition of special symbols not handled by OCR systems. Given a document image and the OCR recognized text, a character segmentation method is developed to refine the character segmentation output provided by the OCR engines. Then, the normal characters are distinguished from special symbols to be processed by the symbol recognition module. Finally, features are extracted from the special symbol subimages and a supervised classifier is used to assign the subimages to one of predefined categories.

The characters' positions within a text line are used as features for matching recognized characters and the glyphs (connected components) on the document image. The glyphs' positions are determined by a classifier adaptively trained within the same text zone. For each character string (with length up to three), we compute

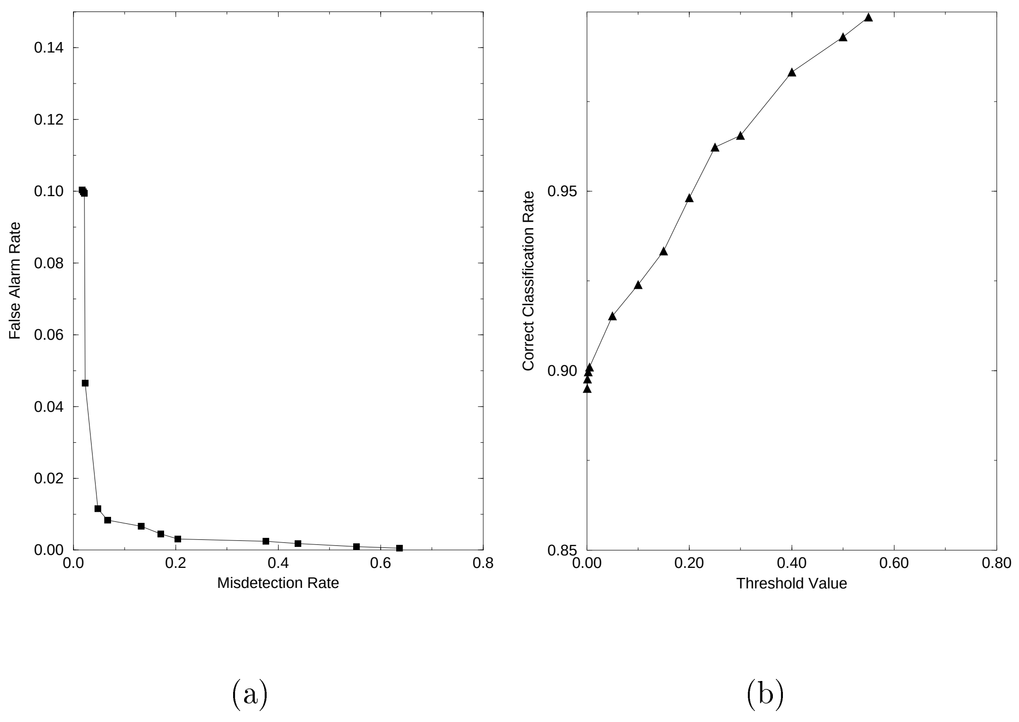


Figure 5.6: Plots (a) false alarm rate vs. misdetection rate; (b) correct classification rate; using different threshold values.

its probability of being a special symbol. The observations are the character string, the sequence of OCR confidence levels, and the context information. The probability map for each special symbol is computed from the training data. Given an input symbol image, we compute the distance from the input image to each of the trained probability maps. Using the probabilities provided by the detection stage as the a prior probability, we use a Bayesian framework to update the probability of each candidate.

The special symbol segmentation and recognition system has been integrated with the Medical Article Record System (MARS) for bibliographic indexing at the National Library of Medicine.

Chapter 6

A PERFORMANCE EVALUATION PROTOCOL FOR GRAPHICS RECOGNITION SYSTEMS

6.1 Introduction

Systems which convert existing paper-based drawings into electronic format are in demand and a few have been developed. However, the performance of the prototypes and commercial systems is either unknown, or only reported in a limited way by the system developers. An evaluation for these systems, or their subsystems, would contribute to the advancement of the field. Responding to this need, a dashed-line detection competition for developers of dashed-line detection algorithms was proposed and took place during the first International Workshop on Graphics Recognition at Penn State University, in 1995. A benchmark[71] was developed and used in that competition. That benchmark includes a performance evaluator and a software tool that automatically generates dashed-line test images and the corresponding ground-truth.

In this chapter, we extend that protocol to evaluate the performance of graphics recognition systems on images that contain straight lines (solid or dashed), circles (solid or dashed), partial arcs of circles (solid or dashed), and text blocks. Engineering drawings primarily use a combination of these geometric elements. Therefore, despite being restricted to these simple entity types, the evaluation protocol is applicable to a wide variety of drawings. Upgrading the evaluator to handle other types of entities is straightforward. To do this, one needs to provide the evaluator the parameters of the entity or entities and the performance evaluation criteria. To evaluate a given recognition system, the system is tested on a set of pre-selected test images. The results of the recognition system are matched, using the criteria defined in this protocol, with the corresponding ground-truth of the test images. The matching results are the numbers of one-to-one matches, one-to-many matches, many-to-one matches, as well as the numbers of false-alarms and misses. Performance measurements for the recognition system can be formulated using a linear combination of some or all of the matching results.

Our evaluator is designed to be used by recognition system researchers and developers for testing and enhancing their recognition algorithms. The evaluator allows the users to select 'text-only', 'graphics-only', or 'all' option for their systems performance evaluations. The 'text-only' evaluation option is designed for recognition systems that detect only the text blocks in the input image. The 'graphics-only' evaluation option is designed for recognition systems that detect only the graphical entities. The 'all' evaluation option is designed for systems that can detect both graphics and text blocks.

This chapter is organized as follows: In Section 6.2, we give a brief review of some related work. In Section 6.3 we specify the parameters for the entities. The protocols for performance evaluation and entity matching are given in Section 6.4. In Section 6.5, we present the matching criteria for each pair of valid combinations. Finally, the linear combination of the match scores is described in Section 6.6.

6.2 Previous Work

Performance evaluation and benchmarking have been gaining acceptance in all areas of computer vision. An overview of this area is available at [72]. Performance evaluation of graphics recognition is still a very young field; objective and quantitative methods for evaluation of graphics recognition have been proposed very recently [71, 73, 74, 75]. Kong *et al.* [71] propose a quantitative method for evaluating the recognition of dashed lines. Hori and Doermann [73] propose a quantitative performance measurement methodology for task-specific raster to vector conversion. Wenyin and Dori [74] present a protocol for evaluating the recognition of straight and circular lines. All of these methods are limited in their applicability.

Kong *et al.* [71] use angle, distance, relative overlap, and offset between line segments for evaluating line matches and for detecting line styles. They use several arbitrary and rigid thresholds. They do not allow for fragmentation of detected lines.

Hori and Doermann [73] instantiate and extend Haralick's framework for performance characterization in image analysis [76], in an application-dependent manner, for measuring the performance of raster to vector conversion algorithms. The "applications" addressed in the work are thinning, medial line finding, and line fitting – all low level techniques that do not completely constitute vectorization. It is hard to extend the work to evaluate a complete vectorization system. Hori and Doermann's protocol does not distinguish between detection rate and false alarm rate. It does not include an overall evaluation metric. It does not allow for fragmentation of detected lines.

Wenyin and Dori [74] propose performance evaluation indices for straight and circular line detection. Detection and false alarm rates are defined at both the pixel level and the vector level. Pixel level performance indices (measures of shape preservation) are not appropriate when dealing with real images that contain severe distortion introduced by warping and other defects in the hard copy drawing and by the scanning/imaging system. Attempts to obtain a high pixel recovery index would unnecessarily require the detected vectors to be true to the distorted shape of the imaged lines, thereby making the detected lines fragmented. Wenyin and Dori weight all true positives and false positives by their respective lengths. This is inappropriate if the goal of the evaluation is to measure the cost of post-processing operations that are necessary to correct the mistakes of vectorization. The time for manual post processing does not depend significantly on the length of a true positive, a missed entity, or a false positive. Time taken for adding or deleting a line in a CAD tool does not depend significantly on the length of the line.

Neither of the above methods addresses the extraction or separation of text from graphics. It is not possible to evaluate graphics recognition systems on realistic drawings without accounting for text in the drawings. Wenyin and Dori [75] propose a protocol for evaluating text-graphics separation. In this protocol, the quality of the recognized text boxes is measured using Q_b, the basic quality, and Q_{fr}, the fragmentation quality. The protocol does not explicitly penalize overlapping text boxes; they are penalized in an indirect way. Q_{fr} implicitly penalizes overlap among recognized text boxes. For N recognized text boxes that are identical and have a 100% overlap with a text box in the ground-truth, Q_{fr} would be $1/\sqrt{N}$. The theoretical basis for penalizing overlapping text box recognition by $1/\sqrt{N}$ is not stated. Moreover, the protocol of [75] does not allow one to accept one of the N identical text boxes as a good match and to label others as false alarms. Therefore, this penalty term cannot be used to measure post-processing/editing cost.

6.3 Entity Definition

6.3.1 Entity specification

Currently our evaluator handles seven types of entities: solid and dashed lines, solid and dashed arcs, solid and dashed circles, and text areas. The specification of the parameters for these seven types are given below:

- Solid or dashed line type: For a solid or dashed line segment, the parameters are: the entity type indicator (a solid line or a dashed line), the x- and y-coordinates of the two end points (no special ordering for the two points) and the orientation of the line. The orientation we use here is in degrees, clock-wise with respect to x-axis (See Figure 6.1).

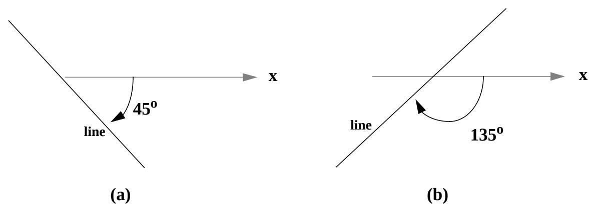


Figure 6.1: The orientation of the line in (a) is 45 degrees, in (b) is 135 degrees.

- Solid or dashed circle type: For a solid or dashed circle, the parameters are: the entity type indicator (a solid or a dashed circle), the x- and y-coordinates of the center, and the radius.

- Solid or dashed arc (partial circle) type: For a solid or dashed arc, the parameters are: the entity type indicator (a solid or a dashed arc), the x- and y-coordinates of the center, the radius, the beginning and the ending angles (in a clock-wise order, in degrees) of the arc. The beginning and ending angles for arcs are also in degrees, clock-wise with respect to x-axis. In some cases, the beginning angles can be larger than the ending angles. (See Figure 6.2.)

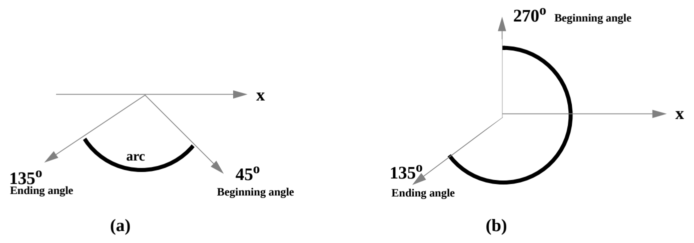


Figure 6.2: The beginning and ending angles of two arcs. In (b), the beginning angle of the arc is larger than the ending angle of the arc.

- Text area type: A text area is represented by a rectangular box and its orientation. The parameters are: the entity type indicator (a text area), the x- and y-coordinates of any two opposite corners of the rectangle, and the orientation of the longest side of the rectangle.

6.3.2 Valid entity type combinations

To speed up the matching score computation, we compute only those pairs having potential matches (e.g., line with line, etc.). The matching score for all incomparable combinations are set to zero. The following is the list of the valid entity type combinations. Others combinations are considered as incomparable.

- Solid-line with solid-line
- Solid-circle with solid-circle
- Solid-arc with solid-arc
- Dashed-line with dashed-line
- Dashed-circle with dashed-circle
- Dashed-arc with dashed-arc
- Solid-line with solid-arc
- Dashed-line with dashed-arc
- Solid-arc with solid-circle
- Dashed-arc with dashed-circle
- Text-area with text-area

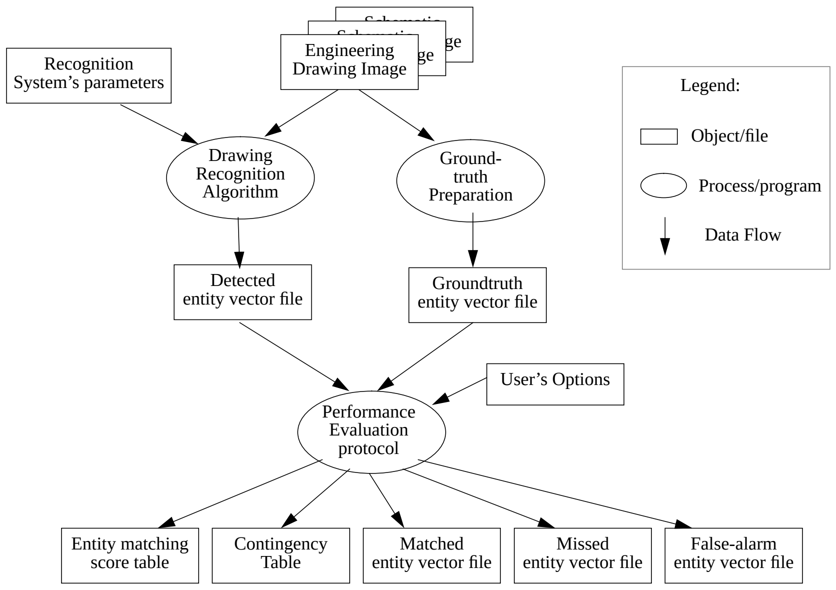

Schematic Image
Schematic Image
Engineering Drawing Image
Recognition System's parameters
Drawing Recognition Algorithm
Ground-truth Preparation
Detected entity vector file
Groundtruth entity vector file
User's Options
Performance Evaluation protocol
Entity matching score table
Contingency Table
Matched entity vector file
Missed entity vector file
False-alarm entity vector file
Legend:
Object/file
Process/program
Data Flow

Figure 6.3: The object-process diagram of our evaluator

6.4 Performance Evaluation Protocol

6.4.1 Evaluation overview

Inputs to the evaluator are entities of the recognition algorithm's output and the corresponding ground-truth. Figure 6.3 shows an object-process diagram of our evaluator. Since there are seven types of entities (solid and dashed lines, solid and dashed arcs, solid and dashed circles, and text areas) that are allowed, the evaluation protocol and the matching criteria are designed differently for each of the combinations. The matching scores for each pair is computed according to the pair's entity type combination, using the matching criteria defined for this combination. These criteria are defined in Section 6.5. A match-score table is produced from the matching score computation.

From the computed match score table, we search for all the one-to-one matches, resolving the problems of the one-to-many matches and the many-to-one matches, as well as, those false-alarms and misses. The matching results are the numbers of one-to-one matches, one-to-many matches, many-to-one matches, as well as the numbers of false-alarms and misses. Performance measurements for the recognition system can be formulated, using a linear combination of some or all of the matching results, which when weighted by application specific weights can be summed to produce an overall score relevant to the application.

6.4.2 Evaluation protocol

The performance of a detection algorithm can be measured by counting the number of matches between the entities detected by the algorithm and the entities in the ground-truth, and the numbers of misses and false alarms. We consider a perfect result of a detection algorithm, if each and every one of the entities in the detected list matches one and only one entity of the same type in the ground-truth list and vice versa. The following is the protocol of this computation:

1. Obtain the detected entities and the entities' parameters and form a detected entities list (D-list $= (d_1, d_2, \cdots, d_M)$) for the entities and their parameters.

2. Obtain the ground-truth entities and the entities' parameters and form a ground-truth entities list (G-list $= (g_1, g_2, \cdots, g_N)$) for the entities and their parameters.

3. Compute the matching score table (See Section 6.4.3).

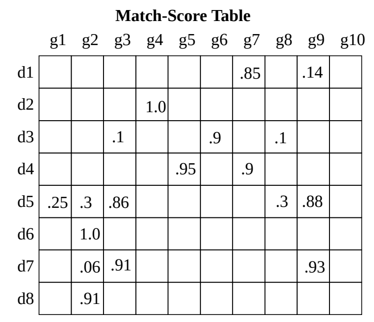
4. Compute the one-to-one matches, resolving the problems of several matches, either from the detections or from the ground-truth (See Section 6.4.4).

5. Compute the one-to-many and many-to-one partial matches.

6. Compute the false-alarms and the misses (See Section 6.4.6).

6.4.3 Matching score table computation

The matching score table is computed as follows. We compute the matching scores (ranging from 0 to 1, 1 being a perfect match) for each pair (with a valid combination) of entities, one from D-list (detected entities) and one from G-list (ground-truth entities), using the matching criteria defined (in Section 6.5) for the pair's combination. The matching score for all invalid combinations are set to zero. A two-dimensional data structure, the *match-score table*, is used to store the results of this computation. A higher matching score indicates a higher degree of match between the corresponding pair of entities. Figure 6.4 illustrates such a table. Blanks are read as zeros.



Match-Score Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							.85		.14	
d2				1.0						
d3			.1			.9		.1		
d4					.95		.9			
d5	.25	.3	.86					.3	.88	
d6		1.0								
d7		.06	.91						.93	
d8		.91								

Figure 6.4: An example of the match-score table

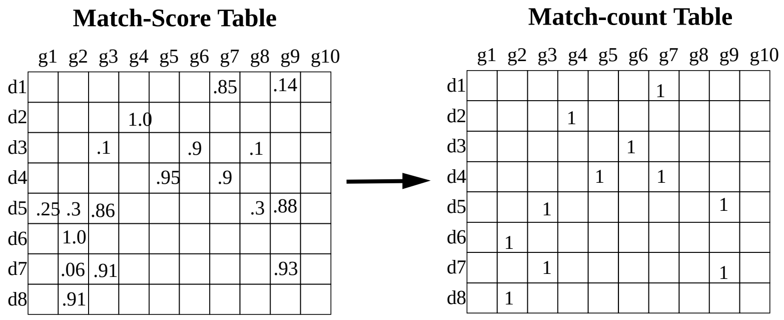
Note that entries in row i of the match-score table represent the matching results from the i-th entity in the D-list to all entities in the G-list. Within a given row, a single entry having a high score value indicates a potential good match of the pair corresponding to that entry.

6.4.4 Computing one-to-one matches

The protocol for computing the entity one-to-one matches is as follows:

1. Compute a two-dimensional match-count table from the computed match-score table. The entry match-count(i, j) is set to 1 if the match-score(i, j) is greater

or equal to *upper-threshold*, otherwise, it is set to zero. Currently, the *upper-threshold* is set to .85. However, we allow users to set their own threshold. Figure 6.5 illustrates the match-count table (on the right), which is computed from the match-score table on the left.



Match-Score Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							.85		.14	
d2				1.0						
d3			.1			.9		.1		
d4					.95		.9			
d5	.25	.3	.86					.3	.88	
d6		1.0								
d7		.06	.91						.93	
d8		.91								

Match-count Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							1			
d2				1						
d3						1				
d4					1		1			
d5			1						1	
d6		1								
d7			1						1	
d8		1								

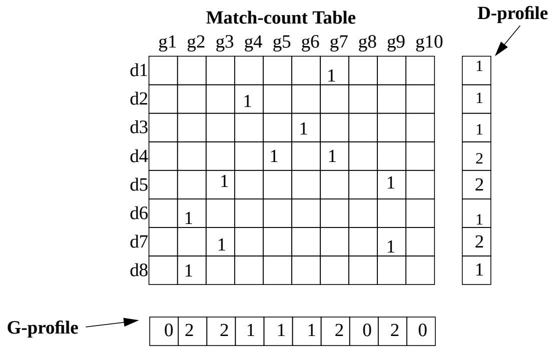
Figure 6.5: An example of the match-count table (on the right) computed from the match-score table on the left.

2. Two projection profiles (D-profile and G-profile) are computed from the match-count table, the result of Step 1.

   The entry $D(i)$ is computed as the sum of the matches in the i-th row of the match-count table. Likewise, the entry $G(j)$ is computed as the sum of the matches in the j-th column of the match-count table. Figure 6.6 illustrates the D-profile and the G-profile computed from the match-count table. The interpretation of these two profiles is as follows:

   - One-to-one matches: A detected entity d_i is one-to-one matched with a ground-truth entity g_j, if $D(i) = 1$, match-count$(i, j) = 1$, and $G(j) = 1$. If the detection algorithm produces a perfect result, all entries in the D-profile and the G-profile will be one.

   - Many-to-one conflicts: An entry in the G-profile, say $G(j)$, is greater than one. That is, there are multiple D entities matching with the entity d_j in



Match-count Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	D-profile
d1							1				1
d2				1							1
d3						1					1
d4					1		1				2
d5			1						1		2
d6		1									1
d7			1						1		2
d8		1									1
G-profile	0	2	2	1	1	1	2	0	2	0	

Figure 6.6: The two projection profiles for the match-count table of Fig. 6.5.

the G-list.

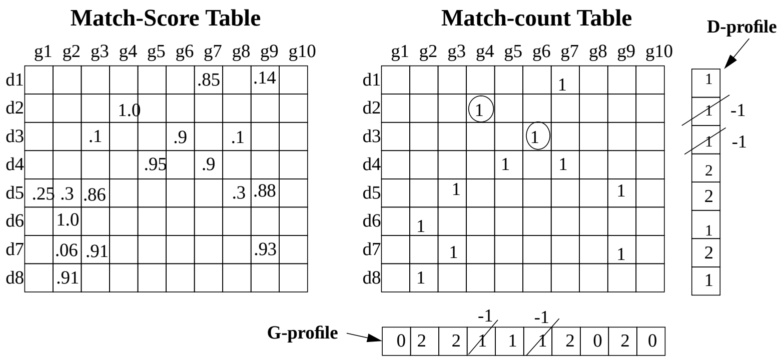
- One-to-many conflicts: An entry in the D-profile, say $D(i)$, is greater than one. That is, the entity d_i matches two or more entities in the G-list.
- False-alarms: A zero entry in the D-profile indicates that no strong match is found from this D entity to any of the entities in the G-list.
- Misses: A zero entry in the G-profile indicates that no strong match is found from this G entity to any of the entities in the D-list.

3. Compute the one-to-one match list.

   For each $D(i) = 1$, we attempt to locate the pair (i, j) such that both $G(j) = 1$ and match-count$(i, j) = 1$, a one-to-one match (d_2 and d_3 in Figure 6.7). We put the pair, (i, j), in the one-to-one match list, and set $D(i)$ and $G(j)$ to -1 (block the two entities from further consideration).

4. Resolving the many-to-one conflicts.

   For each $D(i) = 1$ (but did not produce a one-to-one match in Step 3), we locate the pair (i, j) such that match-count$(i, j) = 1$ and $G(j) > 1$. There are three such pairs in Figure 6.7): $(d_1, g_7), (d_6, g_2)$, and (d_8, g_2).



Match-Score Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							.85		.14	
d2				1.0						
d3			.1			.9		.1		
d4					.95		.9			
d5	.25	.3	.86					.3	.88	
d6		1.0								
d7		.06	.91						.93	
d8		.91								

Match-count Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	D-profile
d1							1				1
d2				$\textcircled{1}$							~~1~~ -1
d3						$\textcircled{1}$					~~1~~ -1
d4					1		1				2
d5			1						1		2
d6		1									1
d7			1						1		2
d8		1									1
G-profile	0	2	2	~~1~~ -1	1	~~1~~ -1	2	0	2	0	

Figure 6.7: One-to-one matching (the resulting one-to-one matches are circled).

Without lost of generality, let $D(i) = 1$ and $D(k) = 1$, and let $G(j) = 2$ (meaning that there are two D entities, d_i and d_k, matching with the entity g_j in the G-list, such as (d_6, g_2) and (d_8, g_2) in Figure 6.7). Suppose match-score$(i, j) \geq$ match-score(k, j),

- If $D(i) = 1$, we select the pair (i, j). Then, we put the pair (i, j) in the one-to-one match list, set $D(i)$ and $G(j)$ to -1, and decrease $D(k)$ by one. For example, if $i = 6$, we would select the pair (d_6, g_2) over (d_8, g_2) in Figure 6.7.
- If $D(i) > 1$ and there is a pair (i, t) such that match-score$(i, t) >$ match-score(i, j), i.e. d_1 and d_4 with g_7 in Fig. 6.7, we would not select the pair (i, j). In this case, we would select the pair (k, j) instead.

  For the example in Figure 6.7, d_1 and d_4 match g_7, which is a many-to-one conflict. The pair (d_1, g_7) is selected instead of (d_4, g_7) since the match-score$(4, 5) >$ match-score$(4, 7)$.

A similar treatment is done if $G(j)$ is greater than two. This step is repeated until no more $D(i)$ is equal to one.

5. Resolving the one-to-many conflicts.

For each $D(i) = 2$, let g_j and g_k be the two entities in G-list that match with the entity d_i in the D-list. If match-score$(i, j) \geq$ match-score(i, k), we put the pair (i, j) in the one-to-one match list and set $D(i)$ and $G(j)$ to -1, and decrease $G(k)$ by one. Otherwise, we put the pair (i, k) in the one-to-one match list and set $D(i)$ and $G(k)$ to -1, and decrease $G(j)$ by one. A similar treatment is done if $G(j)$ is greater than two. This step is repeated until no more $D(i)$ is two or greater.

6.4.5 Computing partial matches: one-to-many and many-to-one

At the end of the one-to-one entity matching (Section 6.4.4), the value of each $D(i)$ indicates whether the i-th entity has a one-to-one match. In particular, if $D(i) = -1$, it indicates that the i-th entity in D-file has a one-to-one match, otherwise, it indicates that it does not have a one-to-one match (the same idea for the entities in G-file). One could, for example, consider each $D(i) \geq 0$ a false alarm and each $G(j) \geq 0$ a miss detection.

However, it may be the case that a detected entity which does not have a one-to-one match may in fact match with a group of two or more ground-truth entities. For example, a detection algorithm may have located a text bounding box on the input image that includes several text lines, while those text lines are given one text bounding box each in the ground-truth file. Therefore, to give partial credits to the detection algorithms for finding one-to-many partial matches, we do as follows. For each $D(i) \geq 0$ in the D-profile, we collect a set $\{g_j\}$ of entities in G-profile, such that $G(j) \geq 0$ (also did not have a match by any D entity) and the match-score(i, j) is greater than the *lower-threshold* (so that we would not include any noise). Currently, the *lower-threshold* is set to .05. However, we allow users to set their own threshold. If the sum of the scores for the entities in the collected list is greater than the *upper-threshold*, we consider the i-th D entity having a one-to-many partial match to elements of the set $\{g_j\}$. And we set $D(i)$ and all those $G(j)$ to -1.

A similar protocol is applied to find the many-to-one matches (many detected entities matching with one ground-truth entity).

6.4.6 False-alarms and misdetections

Finally, the false-alarms are those entities d_i having $D(i) \geq 0$, and the miss-detections are those entities g_j having $G(j) \geq 0$.

6.5 Matching Criteria

6.5.1 Line-line matching protocol and criteria

This protocol is for both solid lines and dashed lines. Let d_i be a detected line entity in the D-list and g_j be a ground-truth line entity in the G-list. Let match-score(i, j) be the corresponding entry of d_i and g_j. To compute the entry (i, j), we do the follows:

1. If d_i and g_j have the same endpoints (a perfect match), we set match-score(i, j) to 1 and skip the following steps.

2. We compute the angle between d_i and g_j as the included angle between these two line segments (see Fig. 6.8). If $angle(d_i, g_j) \leq 5$, we continue to the next step, otherwise match-score(i, j) is set to zero.

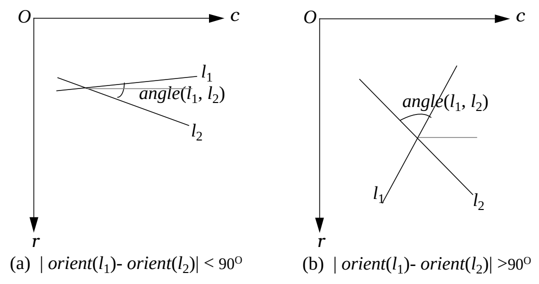

O
c
l_1
$angle(l_1, l_2)$
l_2
r
(a) $|orient(l_1) - orient(l_2)| < 90^o$
O
c
$angle(l_1, l_2)$
l_1
l_2
r
(b) $|orient(l_1) - orient(l_2)| > 90^o$

Figure 6.8: Angles between two line segments.

3. We compute the distance, $llDist(d_i, g_j)$, between d_i and g_j. The distance is computed as the average of the orthogonal distance from the midpoint of d_i to g_j and the orthogonal distance from the midpoint of g_j to d_i (see Fig. 6.9). If $llDist(d_i, g_j) \leq \theta_{ll}$, where θ_{ll} is a predetermined threshold, we continue to the next step. Otherwise, match-score(i, j) is set to zero.

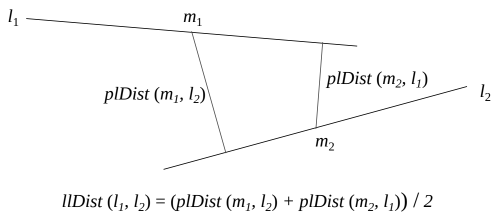

l_1
m_1
$plDist\ (m_2, l_1)$
l_2
$plDist\ (m_1, l_2)$
m_2
$llDist\ (l_1, l_2) = (plDist\ (m_1, l_2) + plDist\ (m_2, l_1))\ /\ 2$

Figure 6.9: Line-line distance between two line segments.

4. Compute the relative overlap between d_i and g_j with respect to the orientation of g_j,

$$RelativeOverlap(d_i, g_j, orient(g_j).$$

The α-projection of a line $l = (c_1, r_1, c_2, r_2)$ is the projection of l onto the given orientation $\alpha \in (-90°, 90°]$. The α-projection of l, $proj(l, \alpha)$, is also a line segment, where its two endpoints (c'_1, r'_1) and (c'_2, r'_2) are the projections of (c_1, r_1) and (c_2, r_2) onto the orientation α, respectively.

$$\begin{aligned} c'_1 &= \cos\alpha(c_1\cos\alpha + r_1\sin\alpha) \\ r'_1 &= \sin\alpha(c_1\cos\alpha + r_1\sin\alpha) \\ c'_2 &= \cos\alpha(c_2\cos\alpha + r_2\sin\alpha) \\ r'_2 &= \sin\alpha(c_2\cos\alpha + r_2\sin\alpha) \end{aligned} \tag{6.1}$$

If $|\alpha - orient(l)| \leq 90°$, $proj(l, \alpha)$ is given by (c'_1, r'_1, c'_2, r'_2); otherwise, it is given by (c'_2, r'_2, c'_1, r'_1).

The α-overlap of two line segment l_1 and l_2, $overlap(l_1, l_2, \alpha)$, is a relationship function of l_1 and l_2 with respect to a given orientation α:

$$overlap(l_1, l_2, \alpha) = length(proj(l_1, \alpha) \cap proj(l_2, \alpha)). \tag{6.2}$$

The relative overlap of two line segments l_1 and l_2 is defined as the ratio between the overlap function $overlap(l_1, l_2, \alpha)$ and the length of the longer segment:

$$RelativeOverlap(l_1, l_2, \alpha) = \frac{overlap(l_1, l_2, \alpha)}{\max(length(l_1), length(l_2))}. \tag{6.3}$$

5. Compute match-score(i, j) as

$$RelativeOverlap(d_i, g_j, orient(g_j)) - \frac{angle(d_i, g_j)}{180} - \frac{llDist(d_i, g_j)}{\theta_{ll}}. \tag{6.4}$$

The threshold values used here were determined heuristically based on the dimension of the entities. However, we allow these threshold values to be set by the user.

6.5.2 Arc-arc matching protocol and criteria

This protocol is for both solid arcs and dashed arcs. Let A_1 be a detected arc entity in the D-list and and A_2 be a ground-truth arc entity in the G-list. Let C_1 and C_2 be the centers of A_1 and A_2, and let R_1 and R_2 be the two radii (see Figure 6.10).

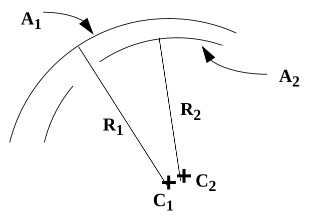


Figure 6.10: An example of arc-arc entity pair

Let match-score(i, j) be the entry that stores the matching result for this pair. The protocol for computing match-score(i, j) is as follows:

1. If the two arcs, A_1 and A_2, are identical (with the same centers, same radii, same beginning and end angles), we set match-score(i, j) to 1 and skip the following steps.

2. Compute the point-to-point distance, $ppDist(C_1, C_2)$, between the two centers C_1 and C_2. If this distance is greater than θ_{CC}, we set match-score(i, j) to zero, and skip the following steps.

3. Compute the absolute difference between the two radii,
$$RadiusDist(R_1, R_2) = |R_1 - R_2|.$$
If this distance is greater than θ_{RR}, we set match-score(i, j) to zero, and skip the following steps.

4. Compute the ratio of these two radii,
$$RadiusRatio(R_1, R_2) = \frac{min(R_1, R_2)}{max(R1, R_2)}.$$
If this ratio is smaller than 85%, we set match-score(i, j) to zero, and skip the following steps.

5. We project A_1 onto A_2, take the portion of A_2 that is within this projection, and call it A_3. The projection is defined as follows. We construct two lines, l_1 and l_2, from the center of A_1 to the two endpoints of A_1. The wedge between these two lines (also within the beginning and the ending angles of A_1) is the projection of A_1 (See Figure 6.11).

   Similarly, we project A_2 onto A_1 and take the portion of A_1 that is within this projection, call it A_4 (see part (d) in Figure 6.12).

6. If either A_3 or A_4 does not exist, we set the match-score(i, j) to zero and skip the rest. In the case that both A_3 and A_4 exist, we define a line segment L_1 from the two ends of A_3 and another line segment L_2 from the two ends of A_4.

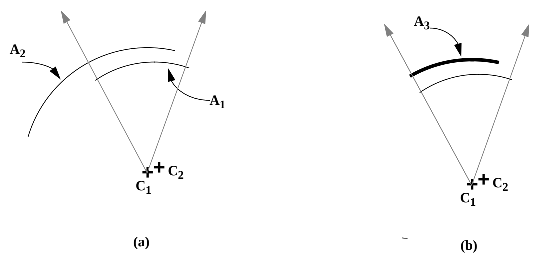


Figure 6.11: Illustrates (a) The projection of A_1 onto A_2, (b) A_3, the portion of A2 within the projection.

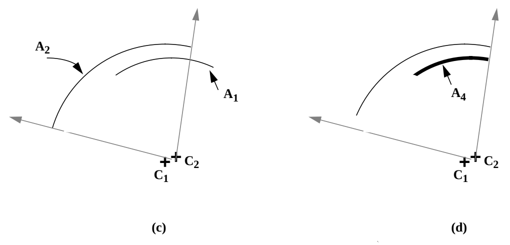


Figure 6.12: Illustrates (c) The projection of A_2 onto A_1, (d) A_4, the portion of A_1 within the projection.

7. Taking L_1 and L_2, we apply the line-line matching criteria to this pair. The match-score(i, j) for this pair is the result of the line-line matching result times F, an adjustment. Without lost of generality, let L_1 be shorter than L_2 and the distance between the two ends of A_2 be greater than that of A_1. Then F is computed as the ratio of the length of L_1 and the distance between the two ends of A_2.

6.5.3 Arc-line matching protocol and criteria

This protocol is for solid-arc with solid-line pairs and dashed-arc with dashed-line pairs. Let A_1 be a detected arc entity in the D-list, and let C_1 be the center and R_1 be the radius of A_1. Let L_1 be a ground-truth line entity in the G-list (see Figure 6.13).

Let match-score(i, j) be the entry that stores the matching result for this pair.

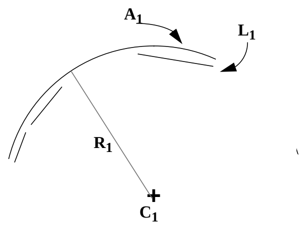


Figure 6.13: An arc-line entity pair

The protocol for computing match-score(i, j) is as follows:

1. We construct two lines, l_1 and l_2, from the center of A_1, to the two endpoints of L_1. We also construct a new line segment, R_2, from the center of A_1 to the midpoint of L_1. In the sense, we are constructing an artificial arc, A_2, taking C_1 as its center, R_2 as its radius, the orientation of l_1 as its beginning angle, and the orientation of l_2 as its ending angles (see Figure 6.14).

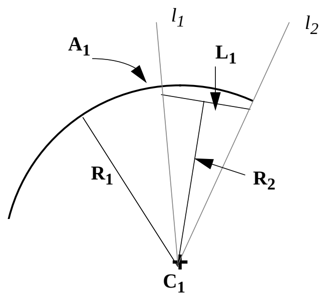


Figure 6.14: Construction of the new triangle and the new line segment.

2. Compute the absolute difference between R_1 and R_2, $|R_1 - R_2|$. If this distance is greater than θ_{RR}, we set match-score(i, j) to zero, and skip the following steps.

3. We project the artificial arc, A_2, onto A_1, take the portion of A_1 between l_1 and l_2, and call it A_3. If A_3 does not exist, we set match-score(i, j) to zero and skip

the rest. In the case that A_3 exists, we construct a new line segment, L_1, from the new arc A_3 (see Figure 6.15).

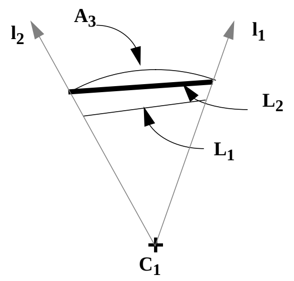


Figure 6.15: The new line segment, L_2, which is constructed from A_3.

4. Taking L_1 and L_2 , we apply the line-line matching criteria to this pair. The match-score is set to the result of the line-line matching result times F, an adjustment. Without lost of generality, let L_1 be shorter than L_2 and the distance between the two ends of A_1 be greater than that of A_2. Then F is computed as the ratio of the length of L_1 and the distance between the two ends of A_1.

6.5.4 Arc-circle matching protocol and criteria

This protocol is for solid-circle with solid-arc pairs and dashed-circle with dashed-arc pairs. Let A_1 be an arc entity in the D-list and and Cir be a circle entity in the G-list. Let C_1 and C_2 be the centers of A_1 and Cir_2, and let R_1 and R_2 be the two radii.

Let match-score(i, j) be the entry that stores the matching result for this pair of entities. The protocol for computing match-score(i, j) is as follows:

1. We project A_1 onto Cir_2, take the portion of the Cir_2 that is within this projection, and call it A_2. In the sense, we are constructing an artificial arc, A_2,

taking C_2 as its center, R_2 as its radius, and the orientations of the two sides of the projection as its beginning and ending angles.

2. We construct a line segment, L_1, from the two ends of A_1 and another line segment, L_2, from the two ends of A_2. Taking L_1 and L_2, we apply the line-line matching criteria to this pair. The match-score is set to the result of the line-line matching result times F, an adjustment. F is computed as the inside angle of A_1 divided by 360.

6.5.5 Circle-circle matching protocol and criteria

This protocol is for both solid and dashed circles. Let Cir_1 be a circle entity in the D-list and and Cir_2 be a circle entity in the G-list. Let C_1 and C_2 be the two centers and let R_1 and R_2 be the two radii (see Figure 6.16).

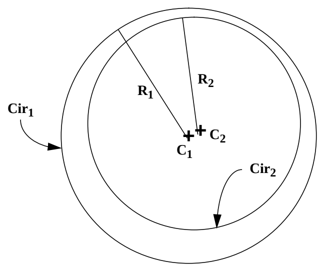


Figure 6.16: A circle-circle entity pair.

Let match-score(i, j) be the entry that stores the matching result for this pair of entities. The protocol for computing match-score(i, j) is as follows:

1. If the two circles, Cir_1 and Cir_2, are identical (with the same centers and the same radii), we set match-score(i, j) to 1 and skip the following steps.

2. Compute the point-to-point distance, $ppDist(C_1, C_2)$, between the two centers C_1 and C_2. If this distance is greater than θ_{CC}, we mark the entry as a non-match and skip the following steps.

3. Compute the absolute difference between the two radii, $RadiusDist(R_1, R_2) = |R_1 - R_2|$. If this distance is greater than θ_{RR}, we mark the entry as a non-match and skip the following steps.

4. Compute the ratio of these two radii,

$$RadiusRatio(R_1, R_2) = \frac{\min(R_1, R_2)}{\max(R1, R_2)}.$$

   If this ratio is smaller than the preset threshold, we mark the entry as a non-match; otherwise, the entry is marked as a match.

5. The matching score for this pair is

$$RadiusRatio(R_1, R_2) - \frac{ppDist(C_1, C_2)}{\min(R1, R_2)} - \frac{RadiusDist(R_1, R_2)}{\min(R1, R_2)} \tag{6.5}$$

6.5.6 Text-text matching protocol and criteria

A text area is represented by a rectangular box and its orientation. The rectangular box is described by the x- and y-coordinates of any two opposite corners of the rectangle, and the orientation of the longest side of the rectangle.

Let t_1 be a detected text entity in the D-list and let t_2 be a ground-truth text entity in the G-list. Let P_1 and P_2 be the two opposite corners of t_1 and let Q_1 and Q_2 be the two opposite corners of t_2.

Let match-score(i, j) be the corresponding entry of t_1 and t_2. The computational protocol for the match-score(i, j) is as follows:

1. If t_1 and t_2 are identical (a perfect match), we set match-score(i, j) to 1 and skip the following steps.

2. Let P_{mid} be the midpoint of the diagonal line of t_1 and let Q_{mid} be the midpoint of the diagonal line of t_2. Without lost of generality, let $ppDist(P_1, P_2) > ppDist(Q_1, Q_2)$. That is, t_1 has longer diagonal than that of t_2. We construct

a circle centered at P_{mid} with radius equal to the $ppDist(P_{mid}, P_1)$. If both the corner points of t_2, Q_1 and Q_2, are outside of this circle (this means that there is no overlap between t_1 and t_2), we set match-score(i, j) to zero, and skip the following steps. This step is designed to limit the search space.

3. Compute the other two opposite corners of t_1 and t_2, so that each of the text boxes is now represented as a rectangular box. Let P and Q be the two rectangles.

4. Compute the intersection I of P and Q. If I is empty, we set match-score(i, j) to zero. Otherwise, we compute the area of P, Q, and I. And we set

$$\text{match-score}(i, j) = \frac{area(I)}{\max(area(P), area(Q))}.$$

6.6 Performance Measurements

Performance measurements for a recognition system can be formulated, using a linear combination of some or all of the matching results: the counts of the matches, the false-alarms, and the misses. Let *one2one* be the count of the one-to-one matches, *one2many* be the count of the one-to-many matches, *many2one* be the count of the many-to-one matches, $false\_alarm$ be the count of the false-alarms, *misdetection* be the count of the misses, N be the count of the ground-truth entities, and M be the count of the detected entities. We define the following system performance measurements:

- The Detection Rate:

$$DetectionRate = w_1 \cdot \frac{one2one}{N} + w_2 \cdot \frac{one2many}{N} + w_3 \cdot \frac{many2one}{N}. \tag{6.6}$$

DetectionRate is, roughly, the percentage of the ground-truth entities being detected. Here, w_1 should weight more than that of w_2 and w_3 since one should favor a one-to-one match over the other two types of matches. For the benchmark used in the contest that was held in [33], w_1 and w_2 were set to 1.

- The Misdetection Rate:

$$MisdetectionRate = \frac{misdetection}{N}. \tag{6.7}$$

MisdetectionRate is the percentage of the ground-truth entities which were not detected by the recognition system. Note that *DetectionRate* and *MisdetectionRate* may not necessary add up to one, because of the factors involve in the computation of *DetectionRate*.

- The False-alarm Rate:

$$FalseAlarmRate = \frac{false\_alarm}{M}. \tag{6.8}$$

FalseAlarmRate is the percentage of the detected entities produced by the system but do not have their correspondences in the ground-truth.

- The Recognition Accuracy Rate:

$$AccuracyRate = w_4 \cdot \frac{one2one}{M} + w_5 \cdot \frac{one2many}{M} + w_6 \cdot \frac{many2one}{M}. \tag{6.9}$$

AccuracyRate indicates, roughly, the percentage of the detected entities within the result file have their matches in the ground-truth entities. Thus, one can consider *AccuracyRate* as a measurement of the overall accuracy rate of a recognition system. Again, one should have more weight on w_4 than that of w_5 and w_6 to favor the one-to-one matches.

- The Post-editing Cost:

$$EditingCost = w_7 \cdot false\_alarm + w_8 \cdot misdetection + w_9 \cdot one2many + w_{10} \cdot many2one. \tag{6.10}$$

EditingCost is an estimated cost for a human post-editing effort to clean-up the recognition result. It should be clear that a higher *EditingCost* requires a higher post-editing effort. Entities missing from the result file need to be added and those false-alarms need to be removed. Moreover, for each one-to-many match, one need to remove the many (those partial matches) from the result file and add the real one to it. And for each many-to-one match, it requires one removal and many additions. Note that, w_7 is the factor for one deletion effort and w_8 is the factor for one insertion effort, the two factors should be weighted according to the post-editing tool one uses for a deletion and an insertion during the post cleaning. The weights assigned to w_9 and w_{10} are more complex; they are dependent on the method used in the counting of these two types of matches. For the benchmark used in the contest of [33], we set w_7 and w_8 to one, and assigned zero to both w_9 and w_{10}.

Chapter 7

CONCLUSION AND FUTURE WORK

The goal of the document structure analysis problem is to find an optimal solution to partition the set of glyphs on a given document to a hierarchical tree structure where entities within the hierarchy are associated with their physical properties and semantic labels. We developed a document structure extraction algorithm that is probability based, where the probabilities are estimated from an extensive training set of various kinds of measurements of distances between the terminal and non-terminal entities with which the algorithm works. The off-line probabilities estimated in the training then drive all decisions in the on-line segmentation module. An iterative, relaxation like method is used to find the partitioning solution that maximizes the joint probability. We have implemented the text line and text block extraction algorithms using this framework.

Another example of the above framework is the development of a system that detects and recognizes the special symbols (Greek letters, mathematical symbols, etc.) on technical document pages, that are not handled by the current OCR systems.

We presented the quantitative performance metrics for each kind of information a document analysis technique infers. A large quantity of ground-truth data, varying in quality, is required in order to give an accurate measurement of the performance of an algorithm under different conditions. We constructed the University of Washington English Document Image Database-III, which contains 1600 English document images that come with manually edited ground-truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text lines, and words. This database can be utilized by the OCR and document understanding community as a common platform to develop, test and evaluate their systems. Based on the ground-truth data, we can evaluate the performance of document analysis algorithms and build statistical models to characterize various types of document structures.

We developed a protocol to evaluate the performance of graphics recognition systems on images that contain straight lines (solid or dashed), circles (solid or dashed), partial arcs of circles (solid or dashed), and text blocks. Our evaluator is designed to be used by recognition system researchers and developers for testing and enhancing their recognition algorithms.

The future extension of the current work includes:

- Functional labeling of text entities;
- Detection and decomposition of tabular structure;
- Mathematical equation/formula detection and analysis.

BIBLIOGRAPHY

[1] R.M. Haralick. Document image understanding: geometric and logical layout, *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* pp. 385-90, 21-23 June 1994.

[2] IBM Almaden. Document Analysis and Recognition web site. http://www.almaden.ibm.com/cs/dare.html.

[3] T.A. Bayer. Understanding structured text documents by a model based document analysis system. *Proc. 2nd Int. Conf. on Document Analysis and Recognition,* pp 448–453, Japan, 1993.

[4] A. Belaid, J. Anigbogu and Y. Chenevoy. Qualitative analysis of low-level logical structure. *Proceedings of Electronic Publishing,* pp 435-446, Darmstadt, Germany, 1994.

[5] A. Conway. Page grammars and page parsing: A syntactic approach to document layout recognition. *Proc. 2nd Int. Conf. on Document Analysis and Recognition,* pp 761-764, Japan, 1993.

[6] A. Dengel. About the logical partitioning of document images. *3rd Symposium on Document Analysis and Information Retrieval,* pp 209-218, Las Vegas, 1994.

[7] A. Dengel and F. Dubiel. Clustering and classification of document structure: A Machine Learning Approach. *Proc. 3rd Int. Conf. on Document Analysis and Recognition,* pp 587-591, Montreal, 1995.

[8] F. Esposito, D. Malerba and G. Semeraro. Automated acquisition of rules for document understanding. *Proc. 2rd Int. Conf. on Document Analysis and Recognition,* pp 650-654, Tsukuba, Japan, 1993.

[9] P. Fankhauser and Y. Xu. MarkItUp! an incremental approach to document structure recognition. *Proceedings of Electronic Publishing,* pp 447-456, Darmstadt, Germany, 1994.

[10] G.S.D. Farrow, C.S. Xydeas, and J.P. Oakley. Conversion of scanned documents to the open document architecture. *1994 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP-94,* pp 109-112, Australia, 1994.

[11] J. Ha, R.M. Haralick, and I.T. Phillips. Document page decomposition using bounding boxes of connected components of Black Pixels, in: L.M. Vincent and H.S. Baird, eds., *Proceedings Document Recognition II,* pp 140-151, San Jose, CA, 1995.

[12] T. Hu and R. Ingold. A mixed approach toward an efficient logical structure recognition from document images. *Proceedings of Electronic Publishing '94,* pp 457-468, Darmstadt, Germany, 1994.

[13] J. Kreich, A. Luhn, and G. Maderlechner. An experimental environment for model based document analysis. *Proc. 1st Int. Conf. on Document Analysis and Recognition,* pp 50–58, France, 1991.

[14] J. Kreich. Robust recognition of documents. *Proc. 2nd Int. Conf. on Document Analysis and Recognition,* pp 444-447, Tsukuba, Japan, 1993.

[15] J. Liang, J. Ha, R.M. Haralick, and I.T. Phillips. Document layout structure extraction using bounding boxes of different entities. *Proceedings Third IEEE Workshop on Applications of Computer Vision,* pp 278–283, Sarasota, FL, 1996.

[16] D. Niyogi and S.N. Srihari. Knowledge-based derivation of document logical structure. *Proc. 3rd Int. Conf. on Document Analysis and Recognition,* pp 472–475, Montreal, 1995.

[17] G. B. Porter III and E. V. Rainero. Document reconstruction: a system for recovering document structure from layout. *Proceedings of Electronic Publishing,* pp 127-141, Lausanne, Switzerland, 1992.

[18] P. W. Palumbo, S. N. Srihari, J.Soh, R. Sridhar, and V. demjanenko. Postal address block location in real time. *IEEE Computer*, pp 34-42, July 1992.

[19] K. Summers. Near-wordless document structure classification. *Proc. 3rd Int. Conf. on Document Analysis and Recognition,* pp 462-465, Montreal, 1995.

[20] K. Summers. Toward a taxonomy of logical document structures. *Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies (DAGS '95),* pp 124-133, Boston, 1995.

[21] S. Tsujimoto and H. Asada. Major components of a complete text reading system. *Proceedings of the IEEE,* pp 1133–1149, Vol. 80, 1992.

[22] F.M. Wahl, K.Y. Wong, and R.G. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing* pp 375–390, Vol. 20, 1982.

[23] S.Y. Wang and T. Yagasaki. Block selection: a method for segmenting page image of various editing styles. *Proceedings Third International Conference on Document Analysis and Recognition,* pp 128–135, Montreal, Canada, 1995.

[24] A. Yamashita, T. Amasno, H. Takahashi, and K. Toyokawa. A model based layout understanding method for the document recognition System. *Proc. 1st Int. Conf. on Document Analysis and Recognition,* pp 130-138, Saint-Malo, France, 1991.

[25] T. Watanabe, Q. Luo, and N. Sugie. Structure recognition methods for various types of documents. *Machine Vision and Applications,* pp 163-176, Vol. 6, 1993.

[26] S. Chen. *OCR Performance Evaluation Software User's Manual,* UW English Document Image Database - (I) Manual, 1993.

[27] S. Chen, S. Subramaniam, R.M. Haralick, and I.T. Phillips. Performance evaluation of two OCR systems. *Proc. SDAIR*, Las Vegas, 1995.

[28] J. Kanai, S V. Rice, T.A. Nartker, and G. Nagy. Automated evaluation of OCR zoning. *IEEE Trans. on PAMI,* pp 86–90, Vol. 17, 1995.

[29] M.D. Garris. Evaluating spatial correspondence of zones in document recognition systems. *Proc. Int. Conf. on Image Processing,* pp 304-307, vol. 3. Washington, DC, Oct. 1995.

[30] R.M. Haralick. Performance assessment of near-perfect machines. *Machines Vision and Applications,* pp 1-16, Vol. 2, no. 1, 1989.

[31] David Byrnes, Raster-to-vector comes of age with AutoCAD release 14. *CADALYST*, pp 48–70, December, 1997.

[32] *Proceedings of the First IAPR Workshop on Graphics Recognition*, University Park, PA, August, 1995.

[33] *Proceedings of the Second IAPR Workshop on Graphics Recognition*, Nancy, France, August, 1997.

[34] I.T. Phillips, J. Liang, A.K. Chhabra, and R.M. Haralick. A performance evaluation protocol for graphics recognition systems. *Graphics Recognition*, pp 372–389, Springer, 1998.

[35] S. Latifi. How can permutation be used in the evaluation of zoning algorithms. *International Journal of Pattern Recognition and Artificial Intelligence,* pp 223-237, Vol. 10, No. 3, 1996.

[36] J. Liang, I.T. Phillips, and R.M. Haralick. Performance evaluation of document layout analysis on the UW data set. *Proceedings Document Recognition IV* pp 149–160, San Jose, CA, 1997.

[37] W. Masek and M. Patterson. A faster algorithm computing string edit distance. *J. Comput. System Sci.,* pp 18–31, Vol. 20, 1980.

[38] G. Nagy and S, Seth. Hierarchical representation of optically scanned documents. *Proceedings Seventh ICPR,* pp 347–349, Montreal, Canada, 1984.

[39] S. Randriamasy and L. Vincent. Benchmarking page segmentation algorithms. *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* pp 411–416, Seattle, WA, 1994.

[40] R. Wagner and N. Fischer. The string to string correction problem. *Journal of the Association of Computing Machinery,* pp 168–173, Vol. 21, 1974.

[41] G.N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design,* McGraw-Hill, 1984.

[42] I.T. Phillips, S. Chen and R.M. Haralick. English document database standard. *Proc. of the Second International Conference on Document Analysis and Recognition* , pp 478-483, Japan, October 20-22, 1993.

[43] I.T. Phillips, S. Chen, J. Ha and R.M. Haralick. English document database design and implementation methodology. *Proc. of the second Annual Symposium on Document Analysis and Information Retrieval*, pp 65-104, April 26-28, 1993.

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

[45] R. Rogers, I.T. Phillips, and R.M. Haralick. Semiautomatic production of highly accurate word bounding box ground-truth. *Proceedings ICPR Workshop on Document Analysis Systems* pp 375–387, Malvern, PA, 1996.

[46] RAF Technology, Inc. *DAFS Document Attribute Format Specification*, 1994.

[47] RAF Technology, Inc. *Programmer's Guide to the DAFS Library*, 1994.

[48] RAF Technology, Inc. *Illuminator User's Manual,* 1994.

[49] A. Gelman, X. Meng, and H. Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica,* pp. 733-807, Vol. 6, 1996.

[50] A. Gelman, Y. Goegebeur, F. Tuerlinckx, and I.V. Mechelen. Diagnostic checks for discrete-data regression models using the posterior predictive simulations. 1997.

[51] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis.* Chapman and Hall.

[52] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. *BUGS Bayesian Inference Using Gibbs Sampling Manual.* 1996.

[53] S. Chen. *Document Layout Analysis Using Recursive Morphological Transforms.* Ph.D. Thesis, University of Washington, 1995.

[54] T. Kanungo and R.M. Haralick and I.T. Phillips. Global and local document degradation models. *Proc., the Second International Conference on Document Analysis and Recognition*, Japan, 1993.

[55] T. Kanungo, H. S. Baird and R.M. Haralick. Validation and estimation of document degradation models. *Proc. of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, 1995.

[56] T. Kanungo. *Document Degradation Models and a Methodology for Degradation Models Validation.* Ph.D. Thesis, University of Washington, 1996.

[57] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision.* Volume I, Addison-Wesley, 1992.

[58] J. Liang. *A Unified Approach for Extracting Document Structure,* ISL Technical Report, Intelligent Systems Laboratory, University of Washington, 1999.

[59] S. Chen, R.M. Haralick and I. Phillips. Automatic text skew estimation in document images. *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR'95)*, pp. 1153-1156, August 1995, Montreal, Canada.

[60] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press.

[61] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, pp 257–285, Vol. 77, No. 2, February, 1989.

[62] F.V. Jensen. *An Introduction to Bayesian Networks.* Springer, 1996.

[63] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*, MIT Press, 1990.

[64] A. Dengel, R. Hoch, F. Hones, T. Jager, M. Malburg, A. Weigel. Techniques for improving OCR results. *Handbook of Character Recognition and Document Image Analysis*, pp. 227-258, Word Scientific Publishing Company, 1997.

[65] MathSoft. *S-PLUS Guide to Statistics,* 1997.

[66] M. Bokser. Omnidocument technologies. *Proceedings of the IEEE*, pp 1066-1078, Vol 80, No. 7, July, 1992.

[67] O.D. Trier, A.K. Jain and T. Taxt. Feature extraction methods for character recognition - a survey. *Pattern Recognition*, pp 641-662, Vol. 29, No. 4, 1996.

[68] T. H. Reiss. *Recognizing planar objects using invariant image features*, Lecture notes in computer science 676, Springer-Verlag, 1991.

[69] A. Khotanzad and Y.H. Hong. Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Recognition and Machines Intelligence*, Vol. 12, No. 5, May 1990.

[70] M.K. Hu. Visual pattern recognition by moment invariants, *IRE Trans. Inf. Theory*, pp 179-187, Vol. 8, February 1962.

[71] B. Kong, I. Phillips, R. Haralick, A. Prasad, and R. Kasturi. A benchmark: performance evaluation of dashed line detection algorithms. *Graphics Recognition: Methods and Applications, First International Workshop, University Park, PA, USA, August 1995, Selected Papers*, Vol. 1072 of *Lecture Notes in Computer Science*, pp 270–285. Springer, Berlin, 1996.

[72] ECVNet. Benchmarking and Performance Evaluation web site. http://pandora.imag.fr/ECVNet/benchmarking.html.

[73] O. Hori and S. Doermann. Quantitative measurement of the performance of raster-to-vector conversion algorithms. *Graphics Recognition: Methods and Applications, First International Workshop, University Park, PA, USA, August 1995, Selected Papers*, Vol. 1072 of *Lecture Notes in Computer Science*, pp 57–68. Springer, Berlin, 1996.

[74] L. Wenyin and D. Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications: Special Issue on Performance Characteristics of Vision Algorithms*, pp 240–250, Vol. 9, 1997.

[75] L. Wenyin and D. Dori. A protocol for performance evaluation of algorithms for text segmentation from graphics-rich documents. *Proceedings of Second IAPR Workshop on Graphics Recognition*, pp 317–324, Nancy, France, August 1997.

[76] R. Haralick. Performance characterization in image analysis: Thinning, a case in point. *Pattern Recognition Letters*, pp 5–12, Vol. 13, 1992.