UW-ISL Document Image Analysis Toolbox : An Experimental Environment

J. Liang, R. Rogers, and R. M. Haralick

Department of Electrical Engineering
University of Washington
Seattle, WA 98195

I. T. Phillips

Department of Computer Science
Seattle University
Seattle, WA 98122

Abstract

A document image analysis toolbox, including a collection of data structures and algorithms to support a variety of applications, is described in this paper. An experimental environment is built to allow developers to develop, test and optimize their algorithms and systems. Appropriate and quantitative performance metrics for each kind of information a document analysis technique infers have been developed. The performance of each algorithm has been evaluated based on these metrics and the UW-III document image database which contains a total of 1600 English document images randomly selected from scientific and technical journals.

1 Introduction

The goal of document image analysis is to transform document images into a hierarchical representation of their structure and content. The document image analysis techniques 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. In the Intelligent Systems Laboratory (ISL) at the University of Washington, we are developing a document image analysis toolbox, including a collection of data structures and algorithms to support a variety of applications. An experimental environment has been built to allow developers to develop, evaluate and optimize their algorithms. The appropriate and quantitative performance metrics for each kind of information a document analysis technique infers have been developed. The architecture allows for convenient experimentation to evaluate the performance of different algorithms and sequences of modules. The performance of each algorithm and the whole system can be evaluated based on these metrics and significant sized test data sets. A series of document image databases have been created for this purpose. We have constructed a prototype of the system and demonstrated its flexibility and functionality on different applications.

2 Document Structure and Document Analysis Problem

The document structure consists of layout structure, logical structure, style and content.

Layout Structure

A layout structure of a document image is a specification of the geometry of the polygons, the content types of the polygons, and the spatial relations of these polygons. Formally, a layout structure is $\Phi = (\mathcal{A}, \mathcal{D})$, where $\mathcal{A}$ is a set of homogeneous polygonal areas, and $\mathcal{D}$ is a set of dividers.

Logical Structure

Logical Structure extraction involves assigning functional labels to each polygon of the page, and ordering the text polygons according to their read order. Formally, a logical structure is $\Psi = (M, R)$, where M associates polygonal areas with their types of content, and R is the reading order.

Content

The content is $O : \mathcal{A} \rightarrow \Sigma$ which associates polygonal areas $\mathcal{A}$ with their contents Σ.

Style

The page style is $\Omega = (S, T, P)$, 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.

3 Experimental Environment

It is clear that much of the early work on document analysis system provided illustrative results and hardly had their techniques tested on significant sized data sets and to measure quantitative performance [4]. The main reasons were 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 standard interchange document structure representation is necessary for developers to exchange the training and test images and share the algorithms. The experimental environment should be flexible enough to support the performance characterization of different modules and the different specific system architectures, such as top-down, bottom-up or hybrid.

Our document analysis system uses the Document Attribute Format Specification (DAFS) [9] as the standard file format. DAFS entities are conveniently defined objects within a document such as a paragraph or word. An entity can have content, which might be the text it encompasses; and properties, such as bounding box, font and point size. DAFS permits the creation of parent, child and sibling relationships between entities, providing easy representation of the hierarchical structures of a document. Through DAFS, developers will be able to exchange training and test documents and work together on shared problems.

3.1 Meta-architecture

The toolbox consists of a collection of document recognition routines. Each routine uses DAFS entities for input and output. The user can create a configuration file to specify where to look for information regarding a given entity, and which recognizer to use. The configure file lists the specific algorithm and the type of entity the algorithm works on for each step. Parameters which may be needed during recognition are passed as properties of the entity or the entity type (See Figure 3.1). There are predefined recognizers for different entity types. The user can write custom recognition routines if the ones provided do not suit the needs, then use the configuration file to associate the new recognition routines with entity types, read in parameters and set session properties. Specific document recognition architectures, such as top-down, bottom-up, hybrid, or iterative, can be defined in the configuration file. This allows for convenient experimentation to determine the best algorithm for each step and the best sequence of modules for an application, and to estimate the algorithm parameters given the training data.

3.2 Performance Evaluation of Document Analysis Algorithms

A performance evaluation needs a performance metric, ground-truth data, and an algorithm to match the output representation of document analysis algorithms with the ground-truth representation.

UW-III [8] is the third in a series of UW document image databases. It contains a total of 1600 English

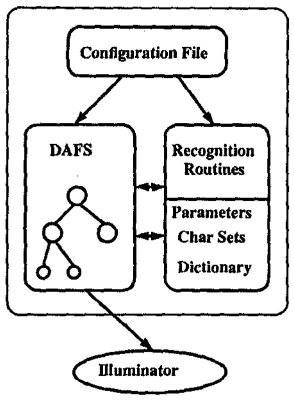

Figure 1: illustrates the environment architecture. Illuminator is the editor of DAFS files

document images randomly selected from scientific and technical journals. The documents are in DAFS format and consist of accurately ground-truthed layout and logical structure, style, and content. Each page contains a hierarchy of manually verified page, zone, text-line, and word entities with bounding box and in the correct reading order. The text ground-truth and style attributes are tagged to each zone entity. 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.

4 Algorithms in the Toolbox and the Performance Measures

The document recognition toolbox consists of the following routine libraries: layout analysis, logical structure analysis, style detection, text recognition (OCR), markup, image processing, and other utilities. In this section, we describes the document analysis algorithms in the toolbox and the corresponding performance measures.

4.1 Layout Analysis

The layout analysis discovers various objects of interest in an input document image. An object is a homogeneous region in a document image that corresponds to one type: character, word, text line, text block, text or non-text zone. Each algorithm in the layout analysis toolbox has its advantage and limitations for documents with different layout and conditions. Therefore it is necessary to characterize these algorithms by evaluating their performance on different kinds of document. The free parameters of the algorithms can be estimated for the different documents if the ground truth is provided. Currently, we are using the default parameter values that are generated by heuristics or from a small set of training samples.

4.1.1 Performance Evaluation of Layout Analysis

To evaluate the performance of the layout analysis, the detected entities are compared with the ground truth entities. Suppose we are given two sets $\mathcal{G} = \{G_1, G_2, \cdots, G_M\}$ for ground-truthed entity boxes and $\mathcal{D} = \{D_1, D_2, \cdots, D_N\}$ for detected entity boxes, comparison of $\mathcal{G}$ and $\mathcal{D}$ can be made in terms of the following two kinds of measures:

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

where $1 \leq i \leq M, 1 \leq j \leq N$, and Area($A$) represents the area of A. The measures in the above equation constitute two matrices $\Sigma = (\sigma_{ij})$ and $T = (\tau_{ij})$. The possible errors of misdetection, false alarm, splitting, merging are detected by analyzing these matrices.

The classification module classifies each extracted entity into one of the predefined categories according to its physical content type. 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 mis-classification rate can be computed from these numbers. Let $P(t, a)$ be the probability of observing a unit whose true category is t, and whose assigned category is a. The mis-classification rate is defined as,

$$P(mis - classification) = \sum_{t \in \Theta} \sum_{a \in \Theta,\; a \neq t} P(t, a),$$

where Θ is the set of content types.

4.1.2 Algorithms in the Layout Analysis Toolbox

Zone Segmentation
A zone entity is a rectangular area that consists of homogeneous data. A text zone is constrained to a single column of text and there is only one reading order for the content within the zone. A document image may be segmented using the recursive X-Y cut based on bounding boxes of connected components [3]. This method was tested on UW-III database with a total of 24,243 zones. Table 1 illustrates the percentage of correct, splitting, merging, miss, and spurious detections with respect to the ground truth.

Table 1: Performance of zone segmentation.

	Correct	Splitting	Merging	Miss	Spurious
X-Y cut	87.18%	3.62%	5.96%	1.08%	2.15%

Zone Classification
Given a homogeneous zone entity, the zone classification module classifies it according to its content. We have developed a method using feature vector generation and classification to classify each given scientific and technical document zone into one of the eight labels: text of font size 8-12pt, text of font size 13-18pt, text of font size 19-36pt, display math, table, halftone, line drawing, and ruling [6]. We have tested our method on UW-I document image data set with 979 pages and a total of 13,726 zones. The mis-classification rate of the algorithm for all zone types is 5%. The mis-classification rate for text and non-text distinction is 3%.

Text-line Segmentation
The layout analysis toolbox includes three different methods to extract text-line entities: the extracted word entities are grouped into lines based on a Probability Linear Displacement Model [1]; the text zones are segmented into lines by cutting the projection profile of connected component bounding boxes [3]; the connected components are grouped into lines by merging and splitting the connected component bounding boxes [5]. The algorithms were tested on UW-III database with a total of 105,439 text-lines (See Table 2).

Table 2: Performance of text line segmentation .

	Correct	Splitting	Merging	Miss	Spurious
PLDM	96.49%	1.41%	2.03%	0.01%	0.07%
Projection	94.78%	0.12%	4.78%	0.28%	0.04%
Group c.c.	97.56%	0.53%	1.26%	0.60%	0.06%

Word Segmentation
Two different word segmentation methods are provided in our tool box: the extracted text-lines are segmented into words based on the vertical projection profile of connected component bounding boxes within the text-line [3]; the black pixels are merged into word using recursive morphological closing transform [1]. These methods were tested on UW-III database with a total of 828,201 words (See Table 3).

Table 3: Performance of word segmentation .

	Correct	Splitting	Merging	Miss	Spurious
Projection	97.83%	0.63%	1.36%	0.16%	0.03%
Morphology	79.22%	2.30%	12.18%	5.68%	0.63%

Text-block Extraction

A text-block is a text entity that can be assigned a functional label (paragraphs, section heading, captions, etc.). The current algorithms to extract text-blocks are: grouping text-lines and making text-blocks by analyzing the alignment of neighboring text-line bounding boxes [3]; characterizing the text-block structure based on the augmented Probabilistic Linear Displacement Model [1]. These methods were tested on UW-III database with a total of 21,738 text blocks (See Table 4).

Table 4: Performance of text block segmentation .

	Correct	Splitting	Merging	Miss	Spurious
Alignment	75.64%	7.42%	15.10%	0.00%	1.83%
APLDM	72.96%	6.50%	15.24%	0.04%	5.26%

4.2 Style Detection

The formatting properties for each zone entity in the document are ground-truthed as attributes in UW-III database. We compare each detected format attribute with the ground-truth attribute and provide the corresponding contingency table. However, the dominant font information for each zone may not be accurate enough for the evaluation of the font attribute detection algorithms which usually work on the character or word-level. In UW-III database, a software package for the automatic generation of character-level ground-truth for scanned documents is provided [10]. We can use this method to generate the real character or word-level font attribute ground-truth. In our system, a set of global features are extracted from text entities (text-block, text-lines, or words) to identify the font attributes: font type, style, size, and spacing mode.

4.3 Logical Structure Analysis

The logical structure analysis consists of the following sub-problems: text entity labeling (paragraph, title, section heading, caption, etc.); and reading order determination; The logical labeling module assigns each entity a functional label. For each correctly extracted entity, we compare the automatically assigned label with the ground-truth label. 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 mis-classification rate is used as the performance measure.

Let $\mathcal{A} = \{A_1, \cdots, A_K\}$ be the set of entities which have been correctly identified. The detected reading order $\hat{R}$ is a tuple $(r_1, \cdots, r_K)$ which is a permutation of true reading order $R = (A_1, \cdots, A_K)$. The problem of evaluating reading order determination algorithm can be reduced to computing the "edit distance" between true reading order R and the output of the algorithm $\hat{R}$. The edit distance is the minimum number of editing operations required to change $\hat{R}$ to R.

4.4 Text Recognition

The extracted text entities are sent to OCR engine to recognize the text content. The problem of evaluating the OCR algorithm can be reduced to the string matching problem between true symbol strings $<T_1, T_2, \cdots, T_x>$ and the output of the OCR algorithm $<O_1, O_2, \cdots, O_y>$. The OCR Performance Evaluation (OPE) Software [2] is provided in UW-I document image database. The program compares the ground-truth and OCR output and output 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.

4.5 Automatic Markup

Given a document structure and content, and a specification of formatting attributes (style), markup module converts the document structure into a formatted document in a desired file format. The file format depends on the application, i.e., SGML for document interchange, RTF for editing and document reconstruction, HTML for hyperlinking, and PDF for document archiving.

5 Example Applications

For different applications, specific document recognition architectures can be constructed by specifying the algorithm to use on the hierarchy at each step. The following are two specific architectures which demonstrate the flexibility of the document analysis tool box.

Document Reconstruction

This architecture (Figure 5) converts scientific journal pages to Rich Text Format (RTF) files, which can be edited by word processors. The layout analysis produces a set of text and non-text entities. The logical structure analysis assigns each text entity a functional label and detects the reading order. An OCR engine is called to recognize the content of text entities. The DAFS to RTF conversion converts the generated DAFS file to an RTF file. The user can define the page style by editing the configuration parameter file or the generated RTF file. We can use Microsoft^{TM} Word^{TM} to reconstruct or edit the document from the generated RTF file.

Document Archiving

This architecture (Figure 5) converts the document pages to Portable Document Format (PDF) files. It is very similar to the function of Adobe^{TM} Acrobat^{TM}

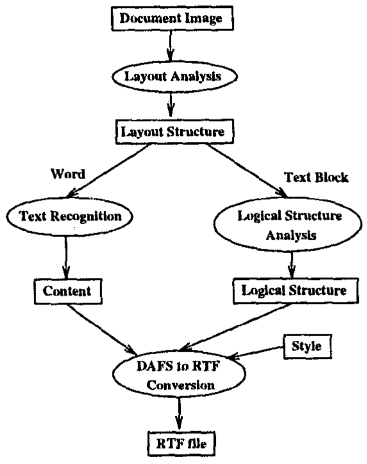

Figure 2: illustrates the architecture for journal article reconstruction.

Capture^{TM} product. A word segmentation routine is called to extract word entities from the input document image. Then an OCR engine is used to recognize the words, and the font style of each word is estimated by a style detection routine. DAFS to PDF conversion module converts the generated DAFS file into a PDF file. The PDF objects retain the exact location of the DAFS entities.

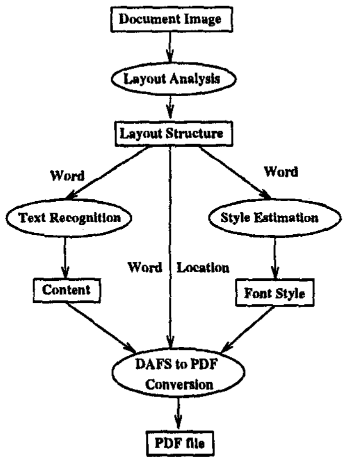

Figure 3: illustrates the architecture for document archiving.

6 Summary

The objective of our research is to develop an experimental environment to support the design and evaluation of document analysis algorithms and systems. A document analysis tool box is being developed to provide a collection of algorithms to support scanned document recognition. We have implemented the basic functionality of each module described above. A system prototype has been constructed and it demonstrates the flexibility and functionality of the environment. We are currently developing new analysis, understanding, and control algorithms to improve the accuracy, efficiency, and robustness of the system on a broad range of document images.

References

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

[2] S. Chen. *OCR Performance Evaluation Software User's Manual.* ISL Report, E.E. Dept., U. of Washington.

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

[4] R.M. Haralick. *Document Image Understanding: Geometric and Logical Layout.* Proceedings of CVPR'94, pp. 385-90, 21-23 June 1994.

[5] J. Liang, J. Ha, R. Rogers, B. Chanda, I.T. Phillips, and R.M. Haralick. *The Prototype of A Complete Document Understanding System.* Proc. IAPR Workshop on Document Analysis Systems, pp 130-154, 1996.

[6] J. Liang, I.T. Phillips, J. Ha, R.M. Haralick. *Document Zone Classification Using the Sizes of Connected Components.* Proceedings of the SPIE, Vol 2660, Document Recognition III, pp 150–157, San Jose, 1996.

[7] J. Liang, R.M. Haralick, and I.T. Phillips. *Performance Evaluation of Algorithms in ISL Document Layout Analysis Toolbox.* ISL Technical Report, University of Washington, 1996.

[8] 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.

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

[10] T. Kanungo, R.M. Haralick. *Automatic Generation of Character Groundtruth for Scanned Documents: A Closed-Loop Approach.* Proceedings of ICPR'96, pp 669-675, 1996.