A Fast Algorithm for Bottom-Up Document Layout Analysis

Anikó Simon, Jean-Christophe Pret, and A. Peter Johnson

Abstract—This paper describes a new bottom-up method for document layout analysis. The algorithm was implemented in the CLiDE (Chemical Literature Data Extraction) system (http://chem.leeds.ac.uk/ICAMS/CLiDE.html), but the method described here is suitable for a broader range of documents. It is based on Kruskal's algorithm and uses a special distance-metric between the components to construct the physical page structure. The method has all the major advantages of bottom-up systems: independence from different text spacing and independence from different block alignments. The algorithms computational complexity is reduced to linear by using heuristics and path-compression.

Index Terms—Document analysis, physical page layout, bottom-up layout analysis, Kruskal's algorithm, spanning tree, chemical documents.

1 Introduction

ALL documents have some structure, either perceived by the reader or defined by the basic publishing units such as letters, words, lines and blocks. The first type of structure is referred to as the **logical structure** of the document, while the second is termed the **physical structure** of the page. The logical structure of a document can vary for both subjective reasons (since different people can interpret the same document differently) and objective reasons (different types of document contain different logical units). The physical structure of a document is more clearly defined than the logical structure. It contains fewer elements, that are all well-defined (characters, words, lines etc.) and are independent of their contents. The series of procedures that detect both the physical and the logical structure of a document, and map the two structures onto each other is termed **document layout analysis and understanding**.

Document layout analysis finds the components of the page, i.e., the words, lines and blocks, and forms one of the first steps of all the processing. Results from this stage are important for all the other processes performed after layout analysis: the word and line information is essential for optical character recognition (OCR), the block information is essential for the extraction of the logical structure of the document, i.e., for understanding the document layout.

Chemical **Li**terature **D**ata **E**xtraction (CLiDE) is a project concerned with chemical document processing that is under development at the University of Leeds [1], [2], [3]. The goals of CLiDE are to process whole pages of scanned chemical documents (e.g., journals or books) and to extract information from both the text and the graphic.

In this paper, we discuss the layout analysis method used in CLiDE. Its originality lies in a novel definition of the distance between the segments and in the detection of the layout hierarchy as a minimal-cost spanning tree. The advantageous features of this method are: no necessity for a priori knowledge about the character size or line-spacing; detection of all the elements (words, lines, blocks, columns, stripes); efficient processing of multicolumned images involving complicated text and graphic arrangements.

2 Background

There are two basic approaches to document layout analysis: the **top-down** and the **bottom-up** methods. The top-down methods look for global information on the page, e.g., black and white stripes, and on the basis of this split the page into columns, the columns into blocks, the blocks into lines and the lines into words. The bottom-up techniques start with local information (concerning the black pixels, or the connected components[1]) and determines first the words, then merge the words into lines, and the lines into blocks. The two methods can be combined in various ways. A survey was given by Tang et al. [4] in 1994, which was a thorough summary of the published methods up to 1991.

The global information in the case of the top-down method given by Krishnamoorthy et al. [5] is the white gap between the segments, that is used in the proposed recursive X-Y cut algorithm. Pavlidis and Zhou [6] introduced a method, that even though employs the white streams and projection profile analysis, still does not require preliminary skew-correction, as the analysis is done on short-height segments. The implied assumption for this method is that the page does not have severe skew. A robust, multi-lingual top-down method was proposed by Ittner and Baird [7]. After correction of the skew of the image, this method detects the blocks on the basis of the white streams. It generates the minimal spanning tree for detection of the text line orientation, and finally uses the projection profiles of the blocks to find the text lines. All of these methods, as is typical for the top-down technique, work only for Manhattan layouts, i.e., for pages with clear horizontal and vertical white gaps between and within the blocks. Even with this constraint, it is acknowledged that the top-down techniques are important, as significant proportion of the documents have Manhattan layouts. In addition top-down analysis methods are generally fast, as finding the white gaps is usually a linear algorithm.

In contrast to the above, bottom-up techniques use local information to build up higher information units. O'Gorman [8] has proposed the "docstrum" method, where the relationship of the objects are expressed with polar coordinates (distance and angle). The image is segmented by finding the k nearest-neighbor pairs between the components. Text orientation and the spacing parameters are estimated. With these features the algorithm works for most layouts. Tsujimoto and Asada [9] have proposed a segmentation method, that performs the calculations on the run-length representation of the image, and is more efficient than a bit-map representation. Connected components are extracted, and segments iteratively merged and classified. Saitoh et al. [10] have proposed a similar method, but the reduction of the image is done in both, x and y, directions, i.e., the pixels in the reduced image correspond to a square in the original image. A method for detection of the text line orientation according to geometrical property-heuristics has been given as well. Application of a classical method, namely, the run-length-smearing algorithm (RLSA) (introduced by Wong et al. [11]), was used by Fan et al. [12]. After the RLSA has been performed, the close text lines are merged and the resulting blocks are classified according to a feature-based classification scheme. Bottom-up methods are usually widely applicable to various layouts, but they are generally at least quadratic in time and space, as they calculate "distances" for all unit

• *The authors are with the Institute for Computer Applications in Molecular Sciences, School of Chemistry, University of Leeds, Leeds, England. E-mail: {aniko, jcp, johnson}@mi.leeds.ac.uk.*

Manuscript received Oct. 3, 1994; revised Dec. 13, 1996. Recommended for acceptance by R. Kasturi.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P97011.

1. Connected component of a bit map is a set of black pixels, where a route exists between any two pixels of the set through adjacent black pixels.

(connected component) pairs in the page, although reduction of the size of the image reduces the processing time significantly. A combination of the top-down and bottom-up techniques was applied to document segmentation, by Taylor et al. [13], where documents are first segmented with the recursive X-Y cut algorithm and then results are analysed using the RLSA.

Since the Manhattan constraint is not acceptable for chemical documents, it was felt that a fast bottom-up algorithm, that can process wide variety of complicated text and graphic layouts is needed.

3 The Bottom-Up Method Implemented in CLiDE

Units of the physical page structure can be viewed in a hierarchical structure and thus can be represented with an n-ary tree (Fig. 1). With this representation the reading order can be deduced from the level of depth of the layer. Dashed arrows on the figure mark the reading order (either from top to bottom or from left to the right). For example, strips are read from top to bottom on the page, while columns are read from left to the right within the strips. If the nodes are ordered according to the arrows at each level, then the preorder traversal [14] of this tree gives the correct reading order of a page.

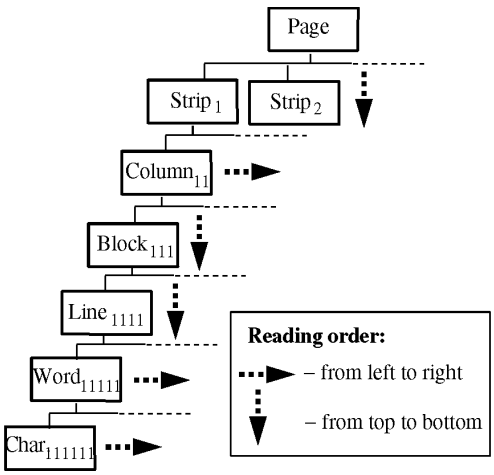


Fig. 1. Physical layout structure of a page.

The document layout analysis method implemented in CLiDE builds up the tree structure given in Fig. 1 in a bottom-up manner, i.e., it starts by processing the connected components of the image, and results in a list of words, text lines, text and graphic blocks, columns and strips. After the image has been loaded, the connected components of the page are found and the noise-like connected components are removed. Document layout analysis starts with calculation of the *distances* between the pairs of the connected components. If one thinks about the connected components as being the vertices of a graph, and the *distances* between them as the weighted edges of the same, then words, lines, blocks, etc., can be derived from the minimal-cost spanning tree, built with Kruskal's algorithm [14]. The word distance is emphasised, because of its novel definition in the implemented algorithm.

The employed method does not need any preliminary separation of the components and it does not assume that there are only characters on the page. However, it does assume that the text lines of the page are horizontal and that the page is scanned with very little skew, or that skew correction has been performed on the whole image. The method is not particularly sensitive to skew of the page and can correctly segment images with a tolerance of $\pm 5^{\circ}$ of skew.

3.1 Definition of the Distance

The distance between any two components (either word, line, block or connected components) of a page is expressed with the distance between their enclosing boxes as shown in Fig. 2. Let us define this as the maximum distance. The formal definition of the maximum distance is given below.

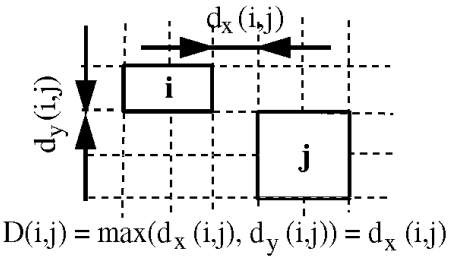


Fig. 2. The maximum distance for any i and j boxes.

Let us suppose that $\forall i \in [1, n]$ CC_i are the connected components of the page. The distance between any two of these $\forall i$, $\forall j$: $i, j \in [1, n]$ is defined as:

$$D(i,j) = \max\left(d_x(i,j), d_y(i,j)\right) \tag{1}$$

where

$$d_x(i,j) = \begin{cases} 0 & \text{if } L(i,j) < R(i,j) \\ L(i,j) - R(i,j) & \text{otherwise} \end{cases} \tag{2}$$

$$L(i,j) = \max\left(CC_i.left_x(\ ), CC_j.left_x(\ )\right) \tag{3}$$

and

$$R(i,j) = \min\left(CC_i.right_x(\ ), CC_j.right_x(\ )\right) \tag{4}$$

The definition of $d_y(i, j)$ is similar to $d_x(i, j)$, but taking the top and bottom instead of the leftmost and rightmost points of the enclosing box.

3.2 The Applied Kruskal's Algorithm

Let us consider the connected components of an image to be the vertices of an undirected graph, and the distances between any two of them as the edges. A complete graph is therefore $G = (V, E)$, with $|V| = n$ and $|E| = \binom{n}{2}$, where n is the number of the connected components. Let us call this graph the **page graph**.

Each edge (i, j) (that is, between i and j vertices) in this graph has a cost attached to it, that is, the distance of the i and j connected components. A **spanning tree** [14] of a connected graph is a tree that contains all the vertices of the graph and those vertices are all connected. A **minimal-cost spanning tree** of a graph is the one spanning tree, for which the sum of the edges is minimal compared to the other spanning trees of the same graph. There are several well known methods for generating a minimal cost spanning tree of a connected graph [14]. One of these is the Kruskal algorithm.

A minimal spanning tree of the page graph, generated with the Kruskal algorithm [14] maps exactly the closest connections between the physical units of the page. In addition, there is a one-to-one correspondence between the physical units of the page and the subtrees of a Kruskal minimal spanning tree of the page graph. The reason for this is the method used by Kruskal to build the minimal-cost spanning tree.

The minimal-cost spanning tree is built by always inserting the smallest of the remaining unused distances. Hence in each step of

the algorithm, the actual state contains some number of components, that have the smallest internal distance at the current level (initially all the vertices are in different components). Therefore these components have the largest cohesion at the current level. At the state when the next smallest distance doubles or triples the size of the last internal distance, there is a change in the layer. For example, the component had only contained characters of a word, while with the new distance, that is much more than the maximum of the previous ones was, it contains another word, building together a line (see Fig. 3—valid internal word distances are labelled in the order of insertion into the graph: d_1, d_2, etc.; invalid distances are marked with a dashed cross. Distance $d_{12}^{1,2,3}$ are invalid in all three cases, because $d_{12}^{1,2,3} >> d_{11}$). Applying this reasoning generally to all the layers of the tree, with some additional heuristics about coefficient of the layers, components of the physical layout can be identified.

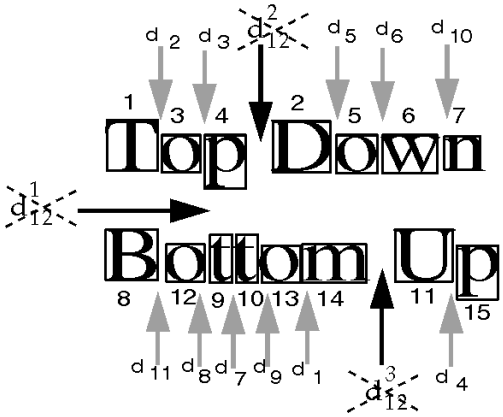


Fig. 3. Generation of word components of the image.

The above explanation requires that the hierarchically ordered physical units have ordered internal cohesion. This condition is not satisfied in all types of layouts. For example, in some documents there exist lines with zero distance between them, while there is always a non-zero distance between the words of a line (see Fig. 4), i.e., in these cases the internal distance of a block is smaller than the internal distance of a line. This clearly shows that the spatial (i.e., the distance) criterion alone is not enough to map the physical layout of the page. The case shown in Fig. 4 does not cause any problem to the human reader, because of the adopted perception that words belong to the same line only if they are written *so*. Therefore one of our heuristics for word detection requires that words which belong to the same line have a vertical intersection[2] which is at least 70% of their heights. Another failure mode is if in the sequence, the distances of large-font within a word come in between the distances of small-font characters within and between words, the steepness will be lost. If this case is typical for the processed document types appropriate heuristics should be applied.

3.3 Using the Path-Compression Algorithm and Some Heuristics for Reducing Processing Time

The total time required for the Kruskal algorithm depends upon the method by which the insertion of the next minimal edge is performed. A common approach is to use a path-compression algorithm [14]. In following a path from some node to the root, this method makes each node encountered along the path a direct child of the root [14], therefore speeding up the finding of a root for a node in following searches. This speeds up the Kruskal algorithm to $O(e\alpha(e))$, where $e = |E|$, i.e., e is the number of the edges of the graph, and $\alpha(x)$ is the inverse of Ackermann's function [14].

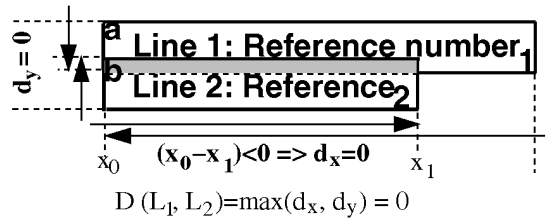


Fig. 4. Overlapping lines are not merged together.

The next problem is to reduce the number of the edges in the graph of the page, as calculation of all the edges still requires $O(n^2)$, where $n = |V|$, i.e., n is the number of the connected components (i.e., vertices). We propose to calculate only the edges that are *probably necessary* instead of all of them. Calculating all of the distances is unnecessary, because it is known that components that do not have vertical intersection are not in the same word (with the exception: 'i,' 'j' etc., but these can be merged with the bulk of the word in an additional loop). Also components that are placed horizontally at a larger distance from each other than certain proportion of their height, are not adjacent letters of a word, or are not in the same word at all. If the connected components are sorted by their y-coordinates initially, the calculation of the necessary distances can be done with a linear algorithm. After a layer has been extracted (e.g., the words have been found), the graph may be regenerated on the new elements, whose cardinality is much smaller than that of the previous layer. Hence for the detection of words the vertices of the graph are the connected components, for the detection of lines the vertices are the words, for detection of blocks the vertices are the lines etc.

It is especially important to compute only the *probably necessary* edges in the case of the first two levels (words and lines), where the number of the vertices of the graph is much larger than in the upper levels. Assuming that a document has only horizontal text lines, unnecessary edges for detection of the words (lines) are:

- edges between characters (words) that do not have vertical intersection, or the value of their vertical intersection is smaller than some threshold (30% (50%) of their height for the words (lines));
- edges between characters (words) that are horizontally placed at larger distance from each other than some threshold (20% (150%) of their height for the words (lines));
- edges between characters (words) that are separated with a graphic line (vertical or horizontal page separator).

If the components are sorted initially by the $y_{\max}$ value of their enclosing boxes and the above conditions, with the strict thresholds, are applied, then the distance calculation—i.e., calculation of the edges of the graph—can be performed in $O(n)$ time and space. Thus, both phases, the detection of the words and the detection of the lines, are linear with respect to the number of the connected components for the detection of the words, and with respect to the number of the words for the detection of the lines.

In contrast to the case of the first two layers, the computational complexity of the other phases of layout detection (the detection of the blocks, detections of the columns and detections of the strips) does not greatly influence the total processing time, as the number of vertices of the graph has been drastically reduced. The distance between every pair of components is calculated, and the pure

2. Vertical intersection of boxes i, j is defined as the intersection of the two y-intervals, that are determined by $y_{\min}$ and $y_{\max}$ coordinates of the boxes, i.e., $\left[y^i_{\min}, y^i_{\max}\right]$ and $\left[y^j_{\min}, y^j_{\max}\right]$.

minimal spanning tree generation using Kruskal's algorithm is carried out. The only condition for each layer is that originating from the adopted way of representing the unit, i.e., elements of blocks and columns have to have horizontal intersection, while elements of strips have to have vertical intersection. As the computational complexity of Kruskal's algorithm with path-compression is $O(e\alpha(e))$, where e is the number of edges of the graph and is quadratic with respect to the number of vertices, this is a good upper limit for the complexity in this situation (as we have restrictive conditions).

To summarize, using the the above heuristics and path compression method, the layout analysis can be performed in linear time with respect to the number of the connected components.

3.4 Classification of the Segments

In the first three phases of the layout analysis, i.e., the generation of the words, the lines and the blocks, the segments are classified into two groups: text and graphic, where graphic refers to line drawing (the classification could be extended to half-tone pictures, but these are not typical in chemical literature). This classification is based of the classification features proposed by Tsujimoto and Asada [9], e.g., the height of the component, aspect ratio of the enclosing box and the percentage of black pixels per unit.

The class of a segment is inherited from the previous, lower layers, i.e., if a word has been classified as graphic, then by expanding it into a line it will automatically become a graphic line, thereafter a graphic block. It is important to classify the segments already at the words level, as the *necessary edges* heuristics are different for text and graphics. Text words must have a significant vertical intersection to be considered as belonging to the same line, while graphic words are merged with any other word which is close enough to their enclosing boxes.

4 Results

The described document layout analysis method has been implemented in C++ and compiled on Sun 4/25 workstation (21 MHz) and on SG Indy (R4600 PC, 100 MHz). It has been tested on 98 document pages, covering books (31 pages), reports (25 pages) and journals (37 pages, from eight different chemistry journals). The considered pages included chemical reaction drawings with text insets; title pages with blocks of variable font sizes; double column pages with full-page-width figures; etc. The algorithm has performed with the average speed of 2,000 connected components/sec of CPU time on the Sun (8,000 CC/sec of CPU time on the SGI), which means a processing time of about two seconds for a dense journal page image with 4,000 connected components on the Sun (approx. 0.5 sec on the SGI). Detailed numerical results of the layout analysis, cardinality of the found segments and the required processing time, on the Sun are given for some of the test images in Table 1.

The accuracy of the method has proved to be good. The 98 test images were analysed with an error rate of 1% at the block level (there were approx. 2,000 blocks in total, with only 20 segmentation and/or classification errors being made), however these errors occurred in seven different images, meaning that 7.14% of the images required manual correction. The segmentation errors were mainly due to accidental proximity of different types of objects (e.g., a graphic region being too close to a label region), while the classification errors were always in cases of small graphics, containing small dashes and letters only. Fig. 5 illustrates the results of the block detection phase for one of the test images (JOC_3 in Table 1).

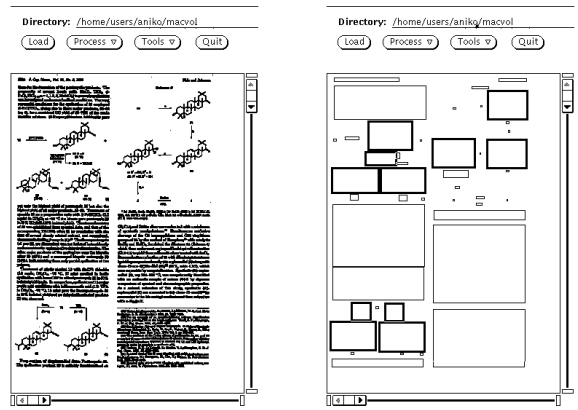
5 Summary

A new bottom-up document layout analysis algorithm has been presented. It detects the layout hierarchy as a minimal-cost spanning tree. A new distance definition, the maximum distance of the components, corresponds to the cost of an edge of the spanning tree. With additional heuristics the computational complexity of the algorithm is less than that of previous bottom-up techniques, as it is linear with respect to the number of the connected components. It enables both Manhattan and a wide variety of other documents to be processed as quickly as with the top-down methods. These two aspects, generality and speed, are the major advantages of the CLiDE document layout analysis method. Experimental results amply demonstrate the accuracy and effectiveness of the method.

TABLE 1
SOME STATISTICAL RESULTS OF PAGE LAYOUT ANALYSIS

image	#(CC's, calc.dist.pairs, words, lines, blk., col., strips)	proc time	speed cc/ sec.
	only text on page:		
$JACS_1$	(5364, 4104, 1075, 101, 20, 9, 4)	2.77	1936.79
$JACS_2$	(5678, 4448, 1080, 111, 30, 9, 4)	2.92	1944.52
	graphic and text on page:		
$JACS_3$	(4592, 3681, 787, 96, 25, 9, 4)	2.35	1954.04
$JACS_4$	(2995, 2413, 562, 195, 18, 7, 5)	1.53	1953.34
JOC_1	(3623, 2887, 603, 114, 33, 8, 5)	1.73	2094.22
JOC_2	(3976, 3157, 706, 117, 35, 5, 2)	2.15	1849.30
JOC_3	(3901, 3188, 649, 105, 37, 2, 1)	1.92	2031.77
	mainly graphic, but some text is present on page:		
JMC_1	(1139, 1037, 171, 59, 16, 3, 3)	0.54	2109.26

Processing time is given on Sun 4/25 platform.



(a) (b)

Fig. 5. The original image (JOC, vol.59, no.9, page 2,328) on the left, the found text and graphic (bold frames) blocks are shown on the right.

Appendix

Abbreviations of the journal names used in Fig. 5 and Table 1:

JACS Journal of the American Chemical Society
JOC Journal of Organic Chemistry
JMC Journal of Medicinal Chemistry.

Acknowledgments

We are thankful to Valerie Gillet and Zsolt Zsoldos for helping us to structure this article and to Chemical Abstract Services for providing us with the test images.

REFERENCES

[1] P. Ibison, F. Kam, R.W. Simpson, C. Tonnelier, T. Venczel, and A.P. Johnson, "Chemical Structure Recognition and Generic Text Interpretation in the CLiDE Project," *Proc. Online Information 92*, London, England, 1992.

[2] F. Kam, R.W. Simpson, C. Tonnelier, T. Venczel, and A.P. Johnson, "Chemical Literature Data Extraction: Bond Crossing in Single and Multiple Structures," *Proc. Int'l Chemical Information Conf.*, Annecy, France, 1992.

[3] P. Ibison, M. Jacquot, F. Kam, A.G. Neville, R.W. Simpson, C. Tonnelier, T. Venczel, and A.P. Johnson, "Chemical Literature Data Extraction: The CLiDE Project," *J. Chem. Inf. Comput. Sci.*, vol. 33, no. 3, pp. 338-344, 1993.

[4] Y.Y. Tang, C.D. Yan, and C.Y. Suen, "Document Processing for Automatic Knowledge Acquisition," *IEEE Trans. Knowledge and Data Engineering*, vol. 6, no. 1, pp. 3-21, 1994.

[5] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic Segmentation and Labeling of Digitized Pages from Technical Journals," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 7, pp. 737-747, 1993.

[6] T. Pavlidis and J. Zhou, "Page Segmentation and Classification," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 6, pp. 484-496, 1992.

[7] D.J. Ittner and H.S. Baird, "Language-Free Layout Analysis," *Proc. Second Int'l Conf. Doc. Anal. and Recogn. (ICDAR'93)*, pp. 336-340, Japan, Oct. 1993.

[8] L. O'Gorman, "The Document Spectrum for Page Layout Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1,162-1,173, 1993.

[9] S. Tsujimoto and H. Asada, "Major Components of a Complete Text Reading System," *Proc. IEEE*, vol. 80, no. 7, pp. 1,133-1,149, 1992.

[10] T. Saitoh, T. Yamaai, and M. Tachikawa, "Document Image Segmentation and Layout Analysis," *IEICE Trans. Information and Systems*, vol. 77, no. 7, pp. 778-784, 1994.

[11] K.Y. Wong, R.G. Casey, and F.M. Wahl, "Document Analysis System," *IBJ J. Research and Development*, vol. 26, no. 6, pp. 647-656, 1982.

[12] K.-C. Fan, C.-H. Liu, and Y.-K. Wang, "Segmentation and Classification of Mixed Text/Graphics/*i*mage Documents," *Pattern Recognition Letters*, vol. 15, no. 12, pp. 1,201-1,209, 1994.

[13] S.L. Taylor, D.A. Dahl, M. Lipshutz, C. Weir, L.M. Norton, R.W. Nilson and M.C. Linebarger, "Integrating Natural Language Understanding with Document Structure Analysis," *Artificial Intelligence Rev.*, vol. 8, no. 2, pp. 255-276, 1994.

[14] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *Data Structures and Algorithms*. Reading, Mass.: Addison-Wesley Publishing Company, 1983.