# Two Complementary Techniques For Digitized Document Analysis

George Nagy, Junichi Kanai, Mukkai Krishnamoorthy
Mathews Thomas, and Mahesh Viswanathan
*Rensselaer Polytechnic Institute*
*Troy, NY, U.S.A.*

## Abstract

Two complementary methods are proposed for characterizing the spatial structure of digitized technical documents and labelling various logical components without using optical character recognition. The top-down method segments and labels the page image simultaneously using publication-specific information in the form of a page-grammar. The bottom-up method naively segments the document into rectangles that contain individual connected components, combines blocks using knowledge about generic layout objects, and identifies logical objects using publication-specific knowledge. Both methods are based on the X-Y tree representation of a page image. The procedures are demonstrated on scanned and synthesized bit-maps of the title pages of technical articles.

## Introduction

We are building a system to partition and label the different logical objects in a document image *without* using OCR. Our equipment consists of a Microtek scanner and a Sun workstation. Intended applications include selective interactive retrieval of portions of digitized document images, component-specific data compression for document transmission and archival, and preprocessing complex pages to facilitate partial or complete optical character recognition.

Approaches for segmenting and labeling document image components may be classified as *top-down* or *bottom-up* [18]. Top-down approaches first divide the document into major regions which are further divided into sub-regions by smearing [11,22], Fourier transform

detection [6] or templates [5]. Bottom-up approaches first extract individual connected components. These components are progressively refined by layered grouping operations based region-border line-coding [3], field-segmentation [9], rule-based analysis [14], and segmentation by 8-connectivity [20]. Most recently, Ingold, Bonvin and Coray have offered a number of suggestions for extracting a logical description of a document from a physical description [7].

## The X-Y tree data structure

Our basic representation scheme for printed documents is the *X-Y tree*, which is a nested decomposition of rectangles into rectangles [13, 17]. At each level of decomposition, all divisions are performed in the same direction, either horizontally or vertically. Horizontal and vertical subdivisions may, but need not, alternate level by level. The number of subdivisions is variable. The root is the binary page-image. Most leaf-nodes represent characters, character-fragments, or parts of an illustration. The X-Y tree can represent hierarchically both the physical structure and the logical structure of a document. We contend that its unified data structure is more appropriate for document *analysis* than data structures devised primarily for document *preparation* [4,8].

A simple way to generate an X-Y tree for a digitized page is recursive segmention of the horizontal or vertical *profile* of each block [1,7]. The vertical profile of a block consists of the horizontal projection (on the vertical axis) of the array, i.e., of the sequence of pixel counts in each row of the array. Conversely, the horizontal profile is the vertical projection of the block. Given either profile, one may divide a block into *grey* and *white* sub-blocks by segmenting it between the rows (columns) where the profile exhibits a zero to non-zero transition or vice versa. Beginning at the page level and recursively alternating this process on gray blocks yields the *physical transition X-Y tree* of the document. An example is shown in Figure 1.

The process stops when neither the horizontal nor the vertical profile of a block contains any transitions. Such

# RESEARCH IN INFORMATION SCIENCE AND TECHNOLOGY

## I. INTRODUCTION

PROGRAM GOALS

## II. PROGRAMS

INFORMATION SCIENCE PROGRAM

INFORMATION TECHNOLOGY PROGRAM

Figure 1. The X-Y tree is superimposed on a sample of text scanned at 300 ppi.

| File Name  | NFS100   | NFS300   | PAMI300  |
|------------|----------|----------|----------|
| Resolution | 100 ppi  | 300 ppi  | 300 ppi  |
| Level 1    | 75       | 73       | 57       |
| Level 2    | 4373     | 4108     | 4020     |
| Level 3    | 7471     | 6802     | 6296     |
| Level 4    | 2586     | 3420     | 6989     |
| Level 5    | 1766     | 2695     | 9756     |
| Level 6    | 101      | 206      | 976      |
| Level 7    | 0        | 0        | 151      |
| Level 8    | 0        | 0        | 21       |
| Total      | 16736    | 17304    | 28266    |
| Leaves     | 13135    | 13531    | 22362    |

Table 1. The number of nodes at various levels of the X-Y tree is reported for the page shown in Figure 1 (scanned at both 100 and 300 ppi) and for a page from a technical journal.

an indivisible block is called *black*. Depending on whether one begins with a horizontal or vertical profile, two different trees are obtained for the same document. These two trees have identical black leaves. The node distribution for several document images is shown in Table 1. As expected, the data structure exploits the rectilinear layout of technical documents, which typically results in wide, shallow trees. Once the tree is constructed, *all* further processing is carried out on the tree rather than the much more voluminous bit-map.

## Top-down segmentation and labeling

Just as a programming language has a fixed set of rules, each publication has predetermined format conventions that are observed. These conventions dictate the *absolute* and *relative ordering* ("author-block" below "title-block"); *absolute position* (page-number one inch from bottom-of-page," "two-inch left margin"); *relative spacing* ("no more than 40% variation in between-word spaces in a text-line," "uniform spacing between-word spaces in a text-line,"; *size* ("footnotes and page-number in 8-point type," "4-point leading between author-lines,"); and *quantity* ("at most 45 printed lines per column of text," "at most 40 characters per columnar text-line," "only one title-block per page"). These attributes have different values for blocks at various levels in the tree.

Documents can be parsed into different components without OCR in the same way that program code can be parsed into meaningful constructs without (semantic) understanding. A key difference, however, is that valid programs are, by definition, never ambiguous, while layout, even if scrupulously observed, may be insufficient to uniquely determine document components. Furthermore, the grammar for modern programming languages is established from the start, while the imprecise specification of page layouts renders it necessary to infer page grammars by indirect means.

A *publication-specific page grammar* is a formal description of all legal page formats that a given publication can assume. In principle, a parse of the page will reveal whether a given page is acceptable under the rules of the page grammar and, if so, what non-terminal

symbols must be assigned to the various components to construe a valid "sentence" in the corresponding language.

The advantage of using the syntactic formalism is that it is theoretically well understood, and there are many tools that allow prediction of the results using different classes of grammars. Furthermore, sophisticated software is available for lexical analysis and parsing of strings of symbols. In order to remain within the domain of strings, we analyze only the horizontal and vertical threshold profiles of each block. The analysis allows subdivision and labeling of the components of a block along one direction, as constrained by the X-Y tree. Lower-level blocks can then be subdivided and labeled in the same way. To construct a formal page grammar for publication-specific page layout in terms of profiles of the X-Y tree, we need the following definitions of syntactic features:

A (horizontal/vertical) *segmentable block profile* is a binary string that contains a "1" for each horizontal or vertical scanline that contains more pixels than a given threshold, and a zero elsewhere. The binary values could also be assigned on some other basis.

A *black atom* is an all-one substring of the segmentable block profile. It is smallest indivisible partition of a segmentable block profile. (Note that an atom may be split following segmentation at the current level and orthogonal segmentation at the next level, because its block profile will now be different.) Examples of black atoms at the top level are lines of print, while at a lower level they might be character blocks. A *white atom* is an all-zero substring between two black atoms, for example an inter-line, inter-paragraph, or inter-word space. *Atom length* is the number of symbols in an atomic substring.

A *black entity* is a contiguous sequence of black and white atom pairs followed by a black atom (i.e., black entities begin and end with black atoms). Examples at the top level are Author and Title blocks, while at a lower level they might be words. A *white entity* is a white atom separating two black entities. At a given level, not all white atoms are white entities.

*Entity class* is a logical label assigned to an entity. There is a one-to-one mapping from entities to entity-class, and a one-to-many mapping from entity-class to entities. Examples of "grey" entity classes are: Header, Title, Authors, Main-text, Word. White entity classes derive their label from those of adjacent grey entities (e.g., "Title/Authors").

*Entity attributes* are quantities computable on the basis of the sequence of atomic lengths within a given entity. The publication-specific page grammar provides constraints on the permissible values of entity attributes. The following attributes are only instances of many that could be defined. They were selected on the basis of their apparent relevance to the segmentation and labeling process.

*Entity atomic length range* (*black* or *white*) is the range of minimum and maximum acceptable lengths of the black (white) atoms in a specified entity class. For example, the title-line height in a given publication may be restricted to 36-42 pixels.

*Entity valence* is the number of black atoms in an entity. For example, we may specify no more than three black atoms (lines-of-text) in an entity labeled Title, or no more than 20 black atoms (character-slugs) per Word.

*Entity divergence_(black* or *white)* is the maximum difference in length between two black (white) atoms within a *single* entity of the specified class.

*Cardinality* is the number of entities in an entity-class (i.e., in a profile string) with a given entity class label. A cardinality range of [0-1] for Acknowledgment would indicate that this entity is optional, and there is at most one per page.

*Class precedence* determines the partial order induced by the linear order of the classes of entities in a segmentable block profile. The page grammar may impose restrictions on the permissible pairwise precedences (e.g., Authors before Title).

A good example of a rather complete source of the necessary publication-specific values of these attributes is the *Author's Kit of Instructions* for these *Proceedings*, but such detailed layout descriptions are seldom obtainable for technical journals. We therefore recorded measurements on two dozen objects found in ten samples each of three technical journals. A publication-specific *page grammar* can now be defined more strictly as *a set of rules that specify allowable segmentations of the block profile into labeled entities* ("interpretations") *according to attributes based on the lengths of the atoms*. We have also developed formal expressions for the above attributes to ensure that they can be expressed as string properties for manipulation by the Unix utilities *Lex* and *Yacc*.

Lex is a lexical analyzer that divides a symbol string into a sequence of tokens described as a *regular expression* [15]. Each token is specified as a pattern of alternative subsequences, with repetitions allowed. We use several cascaded stages of Lex processing to identify entities according to our page grammar. The first Lex stage extracts atoms from the 0/1 profile string. Each atom is given a temporary identifier. The second stage accepts these atom identifiers and produces tentative entity labels. After stage two, the block may be considered segmented into entities but not fully labeled.

Yacc is a parser with single-token look-ahead for context-free languages [10]. Yacc takes into account the cardinality and precedence constraints of the page grammar to determine which of several possible interpretations an ambiguous tentative label can have. It accepts the string of tentative entity labels from the second Lex stage and either produces a string of valid entity labels or an "invalid string" message, but it cannot alter the demarcation boundary within an entity. Once Yacc has produced an acceptable segmentation and labeling, we can repeat the entire process on the profiles of each of the sub-blocks. Of course, the Lex and Yacc grammars for the sub-blocks will depend on the label assigned above, just as the top-level grammar depends on the publication type. It is also possible to alter the grammar and reprocess the higher level depending on what is found below.

An example of syntactic analysis was demonstrated on several synthetically (using *troff)* generated *IBM*

*Journal of Research and Development* pages [21]. The results of each stage of Lex and the Yacc stage are all shown in Figure 2. Not shown are code and results for partial second and third level segmentation, which was also accomplished. The child-entity codes are generally simpler, but different for each parent entity. At the second level, we locate the columns. At the third level, we identify subheadings and individual paragraphs.

### Bottom up layout tree generation

To complement the approach discussed above, we are also attempting to recognize text objects by working our way up from the bottom (leaf nodes) of the transition-cut X-Y tree. This approach takes two steps to labeling logical objects in the document.

In the first step, leaf nodes of the physical tree are classified and manipulated to form character-blocks. Character-blocks are assembled to form word blocks, word blocks into text-line blocks, text-line blocks into text-blocks, text-blocks and graphic objects into column blocks, and so on. The resulting tree, which segments the page into layout objects rather than logical objects, is called a *layout tree*. In the second step, high-level layout objects in a layout tree are labeled as types of logical objects.

Generating the layout tree actually involves more than merely assembling blocks of the physical tree. Not all blocks can be merged while retaining the X-Y tree data structure. In fact, relatively complex operations are necessary to preserve this structure. A pervasive constraint is, however, that *black* leaf blocks are never subdivided. *White* leaf blocks may, however, be divided, as is the case in the *reorientation* operation discussed below. In general, however, in the bottom-up approach labeling progresses from smaller blocks to larger blocks.

Since the transition-cut scheme naively segments a document into small pieces, segmented blocks are often fragments of a layout object. Physical objects are associated with these fragments. The bottom-up approach must therefore manipulate three types of document objects: 1) logical objects, 2) layout objects, and 3) physical objects. We deal with physical objects and layout objects to generate layout trees, and with layout objects and logical objects to generate logical trees. Examples of logical objects, layout objects, and physical objects are:

Logical : Title, Paragraph, Running header
Layout : Text-line-block, Word-block, Character-block
Physical: Character-fragment, Text-block-fragment

Knowledge about such objects is organized into three-levels based on a generalization-specialization hierarchy. The levels are *generic knowledge, publication-class specific knowledge,* and *publication-specific knowledge.* However, further refinement may be necessary: for example, publication-specific knowledge may need a sub-class called *title-page.* The following matrix shows the organization of the entire knowledge base and gives a single example of specific knowledge items for each category.

|  | LEX-1 | LEX-2 | YACC |
|---|---|---|---|
|  | Q | W |  |

J. H. Greiner | B g
C. J. Kircher | B g
S. P. Klepner | B g
S. K. Lahiri | B f
A. J. Warnecke | C
S. Basavaiah | B g
E. T. Yen | C f
John M. Baker | C
P. R. Brosious | B g
H.-C. W. Huang | C f
M. Murakami | C
I. Ames | B g
| C f

G    AUTHOR

|  | P | V |  |

# Fabrication Process for Josephson Integr

| | h | | H  TITLE |

|  | O | S |  |

*A process for fabrication experimental Josephson integrated circuits is described that , vacuum-deposited Pb-alloy and SiO films patterned by photoresist stencil lift-off. The pre ously reported, with changes having occurred in junction electrodes, tunnel barrier fo. geometry, and minimum linewidths. Films of Pb-In(12 wt%)-Au(4 wt%) alloy (200-80C junction base electrodes, interferometer controls, and interconnection lines. Tunnel barr trode films by thermal oxidation and subsequent sputter-etching in an rf-oxygen plasma formed 400-nm-thick Pb-Bi(29 wt%) alloy films. Ground planes are formed from 300 subtractive etching and insulated in part by a NbO layer formed by liquid anodization pound AuIn (30-43 nm thick) are used for forming terminating, load, and damping re for interlayer insulation, for defining junction areas in interferometers, and as protect achieved mainly by means of photoresist lift-off stencils. By utilizing this process, experi containing ~100 interferometers with lines as small as 2.5 microns in width have been st*

C    ABSTRACT

## Introduction

Josephson tunneling devices exhibit fast switching and low power dissipation [1], characteristics that make them attractive for future computer applications [2]. Work is in progress to explore the potential of circuits containing such devices. A process was developed that allowed several initial types of logic and memory circuits to be successfully fabricated [3]. Since that time, improvements have been made in the process, and other investigators have successfully made devices and circuits using adaptations of the process [4].

This paper describe to prepare recent Josep multilayer, integrated ci: Si substrates primaril: deposited thin films. generally Pb alloys and layers were patterned u methods and optical linewidths of 2.5 micron tion process is presented

E    SUBTITLE

C    MAIN TEXT

N

K    FOOTNOTES

F    FOOTER

|  | I | O |  |

Figure 2. Syntactic analysis is used to identify major components of a synthesized page of the *IBM Journal of Research and Development*.

| Knowledge | Logical | Layout |
|---|---|---|
| Generic | No subtitle at the bottom a pageword | Type base lines in a word are collinear |
| Class-specific (technical) | In references, no leading within an item | No text-line lateral to a graphical object |
| Pub-specific (IBM J.of R&D) | Indentation for first-level heading = 1" | Max type size is 22 points |

Table 2. Both logical and layout knowledge are organized into a three-level generalization-specialization hierarchy.

The knowledge base for bottom-up processing is necessarily different from that used for top-down processing: it is much less document-specific. So far we have defined about thirty rules that relate typesetting conventions to X-Y trees. Most of these are at the level of common-sense (generic) knowledge but we prefer to incorporate them explicitly in the knowledge base rather than implicitly into the processing programs (as they usually are in standard OCR software). Some examples:

- A character (symbol) is contained in a rectangular block called a character block.
- A character block is at most as large as the corresponding type body.
- The base line of a type is parallel to the head (foot) of a page.
- The base lines of types are collinear in a word.
- Type faces and point size do not change within a word.
- The base lines of words are collinear in a text line.
- The white space between text-line blocks is equal or greater than the leading within the blocks.

The next step is to incorporate layout knowledge into the bottom-up tree-processing algorithms. We emphasize that these algorithms have no access whatsoever to the digitized page itself, and will label the document objects only on the basis of the X-Y tree. A *definition language* relates the domain knowledge to structures of X-Y trees. The language expresses a fact by applying a *predicate* to a list of *arguments*. An argument is either a constant or a variable. Facts can be related by universal quantifiers, existential quantifiers, AND's, OR's, negations , and conditions. A simple sample definition:

Rule-"rule": A block is a *horizontal rule* if the block is a black (leaf) node, and its height is less than 3 points, and its width is greater than 20 points.

(Forall X)(h_rule(X) <- node(X) &
                       color(X,black) &
                       height(X,H) & lc(H,3) &
                       width(X,W) & gc(W,20))

To illustrate the relationship between the physical tree and the layout tree, consider as a simple conceptual example the block (a):

```
age     a|g|e    a|g|e    a|g|e
bad     b|a|d    -|-|-    -----
                 b|a|d    b|a|d

(a)     (b)      (c)      (d)
```

Because the ascenders and descenders overlap, there is no clear space between them, so the naive segmentation results in a vertical segmentation, as shown in (b). Now each of the three blocks can be horizontally segmented (c). What we would like, however, is the grouping shown in (d), which has the same leaf nodes as (c). Similar situations occur when we attempt to segment horizontally a two-column page if the lines of text in the left column and the right column are not aligned. We are developing algorithms that perform the necessary transformation of a physical subtree into a layout subtree, using primitive tree-manipulation operations such as *neighbor-finding*, *neighbor-merging*, and *reorientation*. The most interesting operation is the last one, which transforms a subtree initiated by a horizontal cut into a subtree initiated by a vertical cut *with identical black leaves* [12].

### Block labeling using KEE

Given a correct layout tree, it can can be readily labeled with an expert system using appropriate information about the layout of the page. We experimented with EXSYS and M1 on a PC/AT [23], but used KEE on tests on the journal *Artificial Intelligence* (which has a single-column, small-format layout). About thirty KEE frames were prepared on the basis of visual observations: this may be regarded as a publication-specific knowledge base [19].

The title pages of ten articles were scanned at 150 ppi and segmented as described above. The tree was then processed using KEE, and nodes corresponding to selected entities (such as *header, title, author, affiliation, recommender, abstract, section header, text-paragraph, footer*) were labeled. Although the specific pages that were classified were not part of the knowledge collection phase, 60% to 100% (depending on the page) of the entities were labeled correctly. The main factor preventing a higher rate of identification was scan skew, which can be corrected either by a micrometer adjustment on the scanner (as we have since done), or in software [2].

In order to validate our results, we modified the troff *ms* macros to generate the various AI journal entities described above directly. We then used these macros to reproduce a page of the journal as faithfully as possible, printed it at 300 dpi, scanned it at 150 ppi, and processed it using KEE as above. We derived from the troff code the identity of each entry using the *tm* (*print string on terminal*) request, and also its position on the page using certain user-accessible registers. A comparison of the KEE output with the troff output is shown in Table 3.

### Conclusion

The syntactic top-down approach works well at the top levels of document analysis, but may be too cumbersome to deal with characters or character fragments. The bottom-up approach needs little if any publication or class-specific knowledge, but may not lead to correct identification of high-level logical entities in documents with a complex format. We are now attempting to combine the two methods. To build a complete

| KEE | | | TROFF | |
|---|---|---|---|---|
| Label | Range | Label | Range | |
| Header | 0, | 12 | Header | 0, 25 |
| Title | 127, | 160 | Title | 117, 159 |
| Author | 204, | 222 | Author | 185, 210 |
| Author Affil. | 250, | 270 | Author Affil. | 229, 254 |
| Recommender | 327, | 346 | Recommender | 306, 331 |
| Abstract | 398, | 409 | Abstract | 384, 405 |
| Section Header | 606, | 620 | Section Header | 593, 618 |
| 1st Paragraph | 636, | 656 | 1st Paragraph | 625, 650 |
| Paragraph | 843, | 864 | Paragraph | 832, 857 |
| Paragraph | 974, | 993 | Paragraph | 965, 980 |
| Paragraph | 1075, | 1091 | Footnote | 1065, 1090 |
| Footer | 1138, | 1154 | Footer | 1127, 1152 |

(1 Unit = 1/150 inch)

**Table 3. The macro calls and vertical coordinates uses by TROFF to synthesize a page of the journal Artificial Intelligence are compared with the labels and block-coordinates obtained by KEE on the transition-segmented TROFF bit-map output.**

system, we must also be able to classify leaf blocks as parts of text or illustrations, and to manipulate compressed versions of the page images. Work on these aspects is underway in our laboratory.

## Acknowledgement

## References

[1] R. Ascher, G. Koppelman, M. Miller, G. Nagy, and G. Shelton, Jr., "An Interactive System For Reading Unformatted Printed Text," *IEEE Transactions on Computers*, Vol. C-20, No. 12, pp. 1527-1543, December 1971

[2] H. Baird, "The Skew Angle of Printed Documents," *Advanced Printing of Symposium Summaries, SPSE's 40th Annual Conference and Symposium on Hybrid Imaging Systems*, Rochester, NY, pp. 21-24, May 1987

[3] N. Bartneck, "Image Analysis Based on Image Description Graphs with Contour Coded Object," *Proceedings of IEEE 7th International Conference on Pattern Recognition*, Montreal, Canada, 1984, pp. 1108-1110

[4] P. Chen and M. Harrison, "Multiple Representation Document Development," *IEEE Computer*, Vol. 21, No. 1, 1988, pp. 15-31

[5] A. Dengel and G. Barth, "Document Description and Analysis by Cuts," *Proceedings of R.I.A.O.*, MIT, March 1988

[6] M. Hase and Y. Hoshino, "Segmentation Method of Document Images by Two-Dimensional Fourier Transformation," *System and Computers in Japan*, Vol. 16, No. 3, 1985, pp. 38-47

[7] R. Ingold, R. Bonvin, and G. Coray, "Structure Recognition of Printed Documents," *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography*, Nice, Italy, April 1988

[8] ISO, "Part 2: Office Document Architecture," *Information Processing - Text Preparation and Interchange - Text Structures*, ISO/DIS 8613/2, June 1986

[9] S. Ito and S. Sakatani, "Field Segmentation and Classification in Document Image," *Proceedings of IEEE 6th International Joint Conference on Pattern Recognition*, Munich, 1982, pp. 492-495

[10] S. C. Johnson, "Yacc: Yet Another Compiler-Compiler," *Comp. Sci. Tech. Rep. No. 32*, Bell Laboratories, Murray Hill, New Jersey, 1975

[11] E. Johnston, "Short Note: Printed Text Discrimination," *Computer Graphics and Image Processing*, 1974, pp. 83-89

[12] J. Kanai, M. S, Krishnamoorthy, and T. Spencer, "Algorithms for Manipulating Nested Block Represented Images," *Advance Printing of Paper Summaries, SPSE's 26th Fall Symposium*, Arlington, Virginia, Oct 1986, pp. 190-193

[13] D. Klarner and S. Magliveras, "The Number of Tilings of a Block with Blocks," *Europ. J. Combinatorics 9*, 1988, pp. 317-330

[14] K. Kubota, O. Iwaki, and H. Arakawa, "Document Understanding System," *Proceedings of IEEE 7th International Conference on Pattern Recognition*, Montreal, Canada, 1984, pp. 612-614

[15] M.E. Lesk, "Lex - A Lexical Analyzer Generator," *Comp. Sci. Tech. Rep. No. 39*, Bell Laboratories, Murray Hill, New Jersey, 1975

[16] G. Nagy and S. Seth, "Hierarchical Representation of Optically Scanned Documents," *Proceedings of IEEE 7th International Conference on Pattern Recognition*, Montreal, Canada, 1984 pp. 347-349

[17] G. Nagy, S. Seth, and S. D. Stoddard, "Document Analysis with An Expert System," *Pattern Recognition in Practice II*, North-Holland, 1986

[18] S.N. Srihari, "Document Image Analysis: An Overview of Algorithms," *Advanced Printing of Symposium Summaries, SPSE's 40th Annual Conference and Symposium on Hybrid Imaging Systems*, Rochester, NY, May 1987, pp. 28-31

[19] M. Thomas, "Knowledge Representation Schemes for Document Analysis System," Master's Thesis, Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, March 1988

[20] J. Toyoda, Y. Noguchi, and Y. Nishimura, "Study of Extracting Japanese Newspaper Article," *Proceedings of 6th International Joint Conference*

*on Pattern Recognition*, Munich, 1983, pp. 1113-1115

[21] M. Viswanathan and M.S. Krishnamoorthy, "A Syntactic Approach to Document Segmentation," to appear: I.A.P.R. Syntactical and Structual Pattern Recognition Workshop, Pont-a-Mousson, Sep. 12-14, 1988

[22] K.Y. Wong, R.G. Casey, and F.M. Wahl, "Document Analysis System," *IBM J. Res. Develop.*, Vol. 26, No. 6, 1982, pp.647-656

[23] J. Yu, "Document Analysis Using X-Y Tree and Rule-Based System," Master's Thesis, Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, December 1986