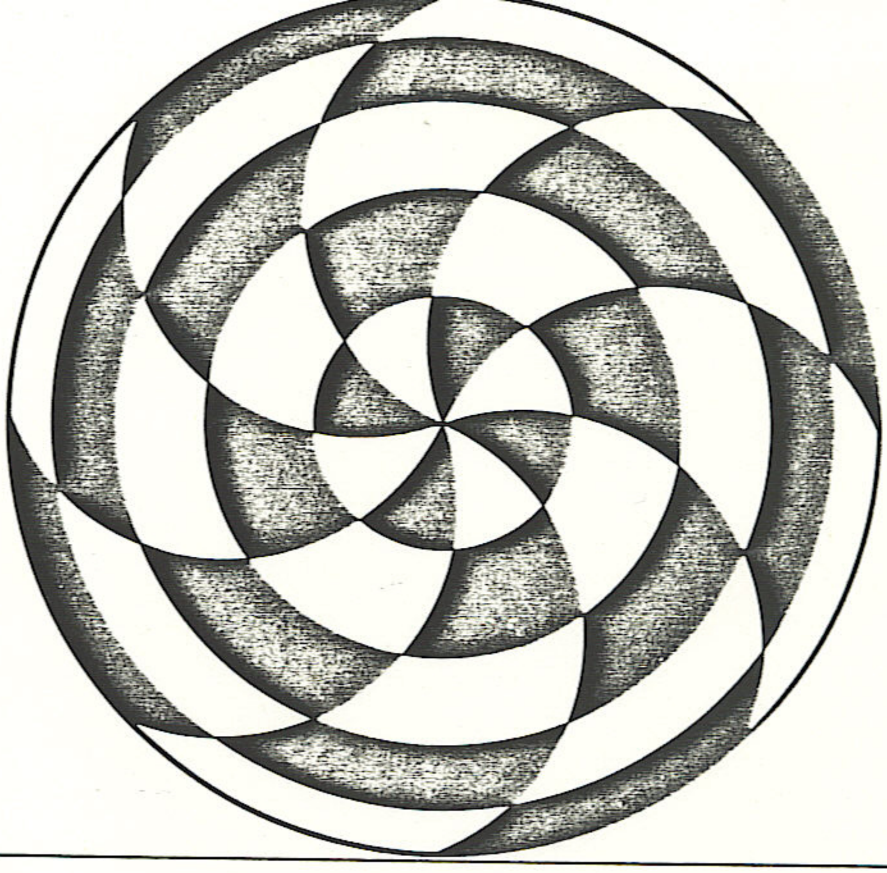
1984

HIERARCHICAL REPRESENTATION OF OPTICALLY SCANNED DOCUMENTS

George Nagy
University of Nebraska-Lincoln

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu



Seventh International Conference on Pattern Recognition

Montreal, Canada
July 30 - August 2, 1984

Proceedings
Volume 1

Organizers:

CIPPRS:
Canadian Image Processing and Pattern Recognition Society

IAPR:
International Association for Pattern Recognition

ISBN 0-8186-0545-6
IEEE Catalog No. 84CH2046-1
Library of Congress No. 84-80909
Computer Society Order No. 545

Printed in the United States of America

COMPUTER SOCIETY PRESS

HIERARCHICAL REPRESENTATION OF OPTICALLY SCANNED DOCUMENTS

George Nagy and Sharad Seth

University of Nebraska-Lincoln

INTRODUCTION

The objective of the research to be pursued is to develop a schema for representing raster-digitized (scanned) documents. The representation is to retain not only the spatial structure of a printed document, but should also facilitate automatic labeling of various components, such as text, figures, subtitles, and figure captions, and allow the extraction of important relationships (such as reading order) among them. Intended applications include (1) data compression for document transmission and archival, and (2) document entry, without rekeying, into editing, formatting, and information retrieval systems.

Several recent developments render such an attempt particularly timely. The advent of public data networks has hastened the establishment of standards for digital facsimile (Castigan 78). The selection of the optimal data compression method for economical transmission depends on the accurate identification of the type of material being transmitted: algorithms that work well on text are different from those that work well on line drawings, or on half-tone photographs (Ting 80). Digital document archival using new mass-storage devices (Bell 78) is beginning to rival the cost-effectiveness of microfilm, particularly when the costs of indexing are taken into consideration.

Word-processing systems form the backbone of office automation: their effectiveness for modifying material existing only on paper is, however, severely limited by the cost of rekeying. Similarly, the cost of data entry is the primary limitation on the use of computerized information retrieval systems in, for example, full-text searches of patents and legal case histories. These applications differ from those described above in the need to actually encode the printed symbols into their ASCII or EBCIDIC representations instead of retaining only their graphic form.

An early attempt to extend standard optical character recognition (OCR) techniques to unformatted documents is described in (Nagy 68). A more comprehensive approach to document analysis, which 'assists the user in encoding printed documents for computer processing' was reported by a team from IBM (Wong 1982). Their intention is to provide an interactive tool, including an optical scanner and a high-resolution display. While we agree wholeheartedly that this is probably the most immediate and practical solution to the problem, in this project we emphasize formal techniques that are, at least in principle, amenable to complete automation.

Other research pertinent to our objectives includes those aspects of optical character recognition which deal with the classification of isolated symbols, as reviewed in (Nagy 82), and the growing literature on picture grammars (Narasimhan 64, Pfaltz 69) and scene analysis (Levine 81). We will take it for granted that suitable character recognition and image data compression algorithms are available to our system. We have, however, developed our own formalism for page representation because we believe that documents form a very special class of images for which data structures, grammars and algorithms capable of dealing with completely arbitrary pictures would be inefficient.

PROPERTIES OF DIGITIZED DOCUMENTS

By 'document' we understand a related collection of printed pages, on paper or microform, such as a technical journal or a report. Here we will consider only single-page documents.

The page is scanned by an optical digitizer operating in a raster mode (Nagy 83). The scan resolution is dictated by the number of samples required to represent the finest essential detail on the page: for a two-level (black-and-white) scanner, 400 lines per inch (62.5 micron spatial sampling interval) is considered adequate and has been adopted as a standard for high-quality facsimile. A letter-sized page is therefore represented by a 3200 x 4000 element binary array, or 13 million bits. We adopt the convention that one-bits correspond to inked portions of the page, and zero-bits to the background.

The structural elements of the page - columns, paragraphs, titles, figures, lines of text, printed symbols - are generally laid out in rectangular blocks. Furthermore, and this observation is crucial, the blocks can almost always be divided into groups in such a way that blocks that are adjacent to one another within a group have one dimension in common. Consequently, most pages can be partitioned into nested rectangular blocks by means of the simple rules described in the next section. We conjecture that this type of 'modular' layout is so common precisely because it can be specified relatively simply with both conventional and digital composition devices.

DEFINITION AND PROPERTIES OF X-Y TREES

Because of its size, one cannot afford to store the pixel-level data (the original digitized page array) more than once. In fact, even the original array may have to be stored in some compressed form, such as run-length codes. Some mechanism is therefore necessary to define and manipulate the location and extent of the meaningful constituents of a page. The data structure we use for this purpose is the X-Y tree.

Each node in the X-Y tree corresponds to a rectangle. The successors of a node correspond either to a set of rectangles obtained by horizontal ~~partitions~~ (X-cuts) of the parent rectangle, or to a set of rectangles obtained by vertical partitions (Y-cuts). Horizontal and vertical subdivisions (X-cut-sets and Y-cut-sets) alternate strictly, level by level. The first subdivision may be arbitrarily set to either horizontal or vertical. The root of the X-Y tree is the rectangle corresponding to the entire page. The leaves constitute a tesselation (tiling) of the page. The relevant properties of the X-Y tree representation are:

1. The page is guaranteed to be completely 'tiled', leaving no portion unaccounted for. Nested subdivisions appear well suited to the hierarchical structure of technical and business records.

2. Only rectangles are generated, allowing identical processing steps at every level.

3. At each level, only a linear (i.e., either horizontal or vertical) subdivision must be considered, which allows a well-ordered sequential examination of the blocks.

4. Unlike quadtrees (Samet 80) the method does not depend on the choice of origin and is more context sensitive.

5. X-Y trees tend to be shallower than K-D trees (Bentley 75), where only binary partitions are allowed. Furthermore, binary partitions do not usually correspond to meaningful partitions of a printed page.

6. X-Y trees can be readily extended to three or more dimensions; this may be advantageous in dealing with multi-page documents, and possibly in other applications.

AUTOMATIC EXTRACTION OF X-Y TREES USING PAGE GRAMMARS

To obtain the X-Y tree decomposition of a page as a set of nested rectangles, at each step of the recursive process of page decomposition we try to subdivide a rectangle into smaller rectangles by making cuts along a predetermined (horizontal or vertical) direction. The decision about where the cuts need to be placed is made by examining a certain number of pixel vectors (beyond the current one) that run across the width (or height) of the rectangle being considered for further division. The process is analogous to determining token boundaries in the lexical scanning phase of compiling (Aho and Ullman 1977).

At each step the process must decide whether or not a cut needs to be made, say, immediately before the current pixel vector. As in token boundary determination it may be necessary to examine several other pixel vectors in the vicinity of the current pixel vector and to undo an earlier decision by backtracking. As a consequence of the two-dimensional nature of a document, however, the final boundaries of blocks defining 'tokens' for the parsing and higher phases are not determined until the end of the process.

It should be noted that the 'rules' for making cuts depend on the context of the rectangle being subdivided. For example, paragraphs will be separated by a much wider white space than successive lines within a paragraph. As another example, consider word and character segmentation, for which elaborate rules have been developed for conventional OCR (Hoffman 71).

The final set of rectangular blocks obtained by the cutting process described above form the terminal symbols of a page grammar. Every derivation tree in the grammar defines a particular decomposition of a page. A page grammar is a generic tool for directing search for cuts using a 'knowledge base' which stores specific information about the rules for cutting at different nodes of the derivation tree. As we show below for an example grammar, with a syntax-directed cutting process the derivation tree for a given page is developed concurrently with the definition of block boundaries. The final product of the application of the lexical scanning and recursive decomposition process is an X-Y tree. Each node of the tree defines implicitly the sequence of rules that were applied to obtain the block represented by the node under consideration. The exact location and size of the block is preserved in an 'extent' attribute which contains the minimum and maximum x and y coordinates of the block. The leaves of the X-Y tree are identical to those of the derivation tree, and may similarly carry designations such as 'character' or 'photograph'.

Within the X-Y tree, the nodes are labeled by a block-index β. The block index consists of a sequence of H's and V's separated by integers. Each H denotes a horizontal cut, and each V denotes a vertical cut. The integers give the number of cuts in each set, with '0' corresponding to a vacuous cut. The block index, which is built during the construction of the derivation tree, also specifies the rule to be used at the next cut. Each rule is numbered in the binary notation: we obtain the number of the next cutting rule to be used by replacing each non-zero integer with '1' in the block index and omitting the H's and V's. For example, the number of the rule to be tried on the block with index 1H3V2H4V0H2 is 111101. Two trailing zeros might be used as a cue to a leaf node.

Attribute Grammar to Determine Block Indices

We present below a simplified example of an attribute grammar to clarify the ideas discussed in the above paragraphs. The only attribute used is the block index which is inherited by a node from its parent in the derivation tree. Our attribute grammar, which can be implemented using a simple stack algorithm, and the rules for propagating the block index, are shown below. The left column shows grammar rules with the attribute of a symbol shown immediately following the symbol within parentheses. The right column shows assignment rules for the attributes. The underlying page grammar is best understood by the following association of meaning to the grammar symbols:

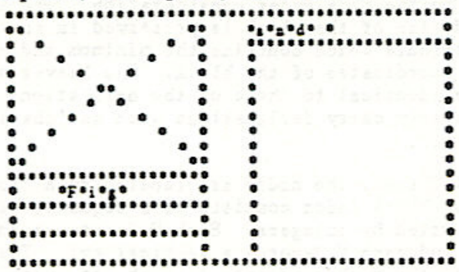
P:	'page'
x:	'block generated by a vertical cut'
y:	'block generated by a vertical cut'
X:	'procedure for generating horizontal cuts'
Y:	'procedure for generating vertical cuts'

It is easily verified that the page grammar conforms to our rule of alternating cuts in the horizontal and vertical directions and allows for the possibility of

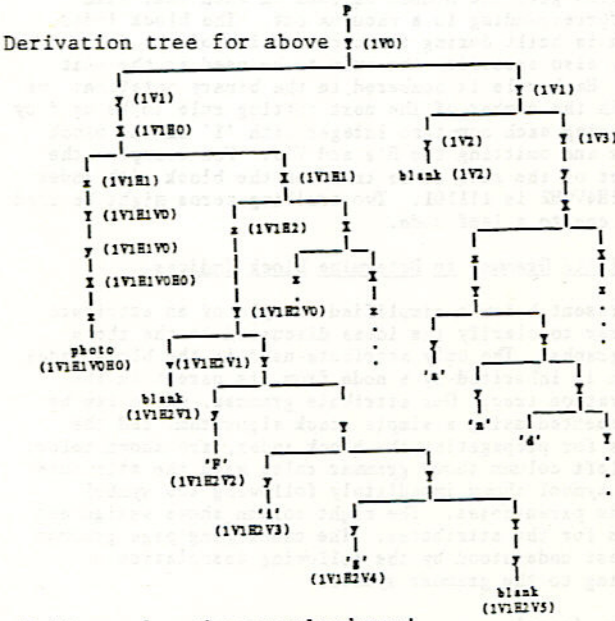
```
P ::= Y(β')             β' := 1V0
Y(β) ::= y(β')Y(β')     β' := if β = <anything>0 then
                              <anything>V1 else (last digit
                              of β incremented by 1)
Y(β) ::= y(β')          β' := if β = <anything>0 then β
                              else (β with last digit
                              incremented by 1)
y(β) ::= t(β')          β' := β
y(β) ::= X(β')          β' := βH0
X(β) ::= x(β')X(β')     β' := (β with last digit incre-
                              mented by 1)
X(β) ::= x(β')          β' := if β = <anything>0 then β
                              else (β with last digit
                              incremented by 1)
x(β) ::= t(β')          β' := β
x(β) ::= Y(β')          β' := βV0
```

vacuous cuts in a particular direction so that, for example, a column may be first cut horizontally into paragraphs using one set of rules and again cut in the same direction to break paragraphs into lines using another set of rules.

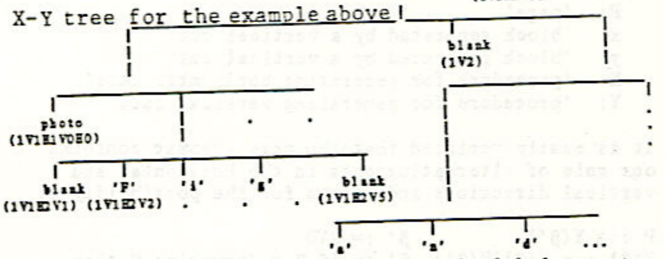
An example of a caricature of a printed page, its partial derivation tree, and the corresponding partial X-Y tree, are shown below. This page has two columns, a photograph in the top left corner, a figure caption, and equally spaced lines of text.



Derivation tree for above



X-Y tree for the example above



The remaining task is to associate meaningful document component labels with each block in the X-Y tree, as required by the application at hand. In general, we expect that these component labels will be synthesized attributes, assigned in a bottom-up traversal of the X-Y tree. Hence not only the labels of lower-level nodes, but also all of the cutting rules used to define a block and, of course, the extent of the block under examination, are available to the component-labeling algorithm. For instance, we might have synthesis rules of the following type:

line:	sequence of adjacent character blocks of same height, separated by character-segmenting rule;
paragraph:	sequence of blocks of line_of_character of same length, separated by line_finding rule;
column:	sequence of paragraph blocks of same width, separated by paragraph-cutting rule;
photo:	large block with approximately equal frequency of '1' and '0' pixels;
drawing:	large block with low frequency of '1' pixels.

An alternative to be explored here is relaxation labeling (Zucker 78).

VALIDATION

Once a hierarchical representation of a digitized printed page has been obtained with the above method (or, indeed, any other method), it is necessary to determine the faithfulness or accuracy of this representation. We propose to validate a given representation by comparing it to the specification according to which the page was originally generated. A formal, machine-readable specification is, however, conveniently available only when the page is generated using a computerized document-preparation system, which usually consists of a text editor and a formatter. Among well-known document preparation systems are (Ossana 76, Knuth 79, IBM 79).

Once two hierarchical representations of the same page have been obtained, one from the scanned document and one from the document-preparation system, the level at which the comparison must be performed depends on the intended application. For compression purposes, for example, it is sufficient to determine whether the locations of the black and white areas are identical, to within a prescribed threshold, in the two representations. If the system were to be used for the entry of printed documents into a word-processor, then the sequence of the alphanumeric characters would be most important. If, however, an information retrieval application were considered, then items such as title, authors, page numbers, and dates would have to be correctly labeled.

REFERENCES: Aho, A.V. and J.D. Ullman Principles of Compiler Design Addison Wesley, Reading, MA, 1977 | Bell, C.G. et.al. Computer Engineering Digital Press, Bradford, MA, 1978 | Bentley, J.L. 'Multidimensional binary search trees used for associative searching' Comm. ACM 18, 8, pp. 509-517, September, 1975 | Castigan, D.M. Electronic delivery of documents and graphics, Van Nostrand, New York, 1978 | Hoffman, R.L. and J.W. McCullough 'Segmentation methods for recognition of machine-printed characters' IBM J. Res. and Dev. 15 pp. 101-184, 1971 | IBM Document Composition Facility and Document Library Facility, General Information - Program Product GH20-9158-2, File No. S370-34, Second Edition, 1979 | Knuth, D.E. TEX and METAFONT - New directions in typesetting, Digital Press, Bedford, MA, 1979 | Levine, M.D. and S.I. Shaheen, 'A modular computer vision system for picture segmentation and interpretation' IEEE-PAMI 2, 5, pp. 540-556, September 1981 | Nagy, G. 'A preliminary investigation of techniques for the automated reading of unformatted text' Comm. ACM 11, 7, pp. 480-487, July 1968 | Nagy, G. 'Optical character recognition - theory and practice' Handbook of Statistics II, (L. Kanal and P.R. Krisnaiah, eds.) North Holland, Amsterdam, pp. 621-649, 1982 | Nagy, G. 'Optical scanning digitizers' IEEE Computer 16, 5, pp. 13-25, May 1983 | Narasimhan, R. 'Labeling schemata and syntactic description of pictures' Information and Control 7, pp. 151-179, June 1964 | Ossanna, J.F. NROFF/TROFF User's Manual Computing Science Technical Report #54, Bell Laboratories, Cherry Hill, NJ, 1976 | Pfaltz, J.L. and A. Rosenfeld 'Web Grammars' Proc. 1st Int. Conf. Artificial Intelligence Washington D.C., pp. 609-619, 1969 | Samet, H. 'Region representation: quadtrees from boundary codes' Comm. ACM 23, 3, pp. 163-170, March 1980 | Ting, D. and B. Prasada 'Digital processing techniques for encoding of graphics' Proc. IEEE 68, 7, pp. 757-769, July 1980 | Wong, K.Y., R.G. Casey and F.M. Wahl 'Document analysis system' IBM J.Res. and Dev. 26, 6, pp. 647-656, November 1982 | Zucker, S.W. et.al., 'Relaxation processes for scene labeling: convergence, speed and stability' IEEE-SMC 8 pp. 41-48, 1978.