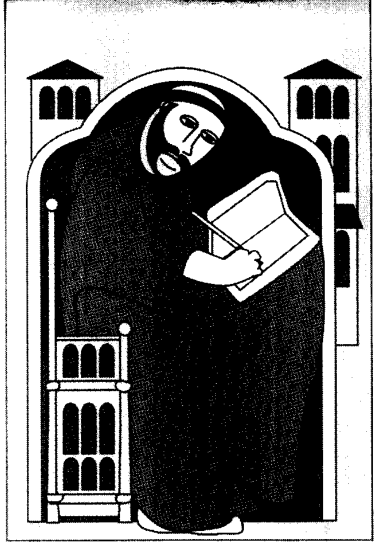
A Prototype Document Image Analysis System for Technical Journals

George Nagy
Rensselaer Polytechnic Institute, nagy@ecse.rpi.edu

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

Mahesh Viswanathan
IBM



A Prototype Document Image Analysis System for Technical Journals

George Nagy, Rensselaer Polytechnic Institute

Sharad Seth, University of Nebraska at Lincoln

Mahesh Viswanathan, IBM

Intelligent document segmentation can bring electronic browsing within the reach of most users. The authors show how this is achieved through document processing, analysis, and parsing the graphic sentence.

Let's quickly calculate the requirements of electronic data storage and access for a standard library of technical journals. A medium-sized research library subscribes to about 2,000 periodicals, each averaging about 500 pages per volume, for a total of one million pages per year. Although this article was output to film at 1,270 dpi (dots per inch) by an imagesetter, reproduction on a 300-dpi laser printer or display would be marginally acceptable to most readers (at least for the text and some of the art). At 300 dpi, each page contains about six million pixels (picture elements). At a conservative compression ratio of 10:1 (using existing facsimile methods), this yields 80 gigabytes per year for the entire collection of periodicals. While this volume is well beyond the storage capabilities of individual workstations, it is acceptable for a library file server. (Of course, unformatted text requires only about 6 kilobytes per page even without compression, but it is not an acceptable vehicle for technical material.)

A 10-page article can be transmitted over a high-speed network, and printed or displayed in *image form* in far less time than it takes to walk to a nearby library. Furthermore, while *Computer* may be available in most research libraries, you may have to wait several days for an interlibrary loan through facsimile or courier. There is, therefore, ample motivation to develop systems for the electronic distribution of digitized technical material.

However, even if the material is available in digital image form, not everyone has convenient access to a high-speed line, a laser printer, or a 2,100 × 3,000-dot display. We show how *intelligent document segmentation* can bring electronic browsing within the reach of readers equipped with only a modem and a personal computer.

Document analysis constitutes a domain of about the right degree of difficulty for current research on knowledge-based image-processing algorithms. This is important because document analysis itself is a transient application: There is no question that eventually information producers and consumers must be digitally linked. But there are also practical advantages to recognizing the location and extent of significant blocks of information on the page. This is also true for segmenting and

0018-9162/92/0700-0010$03.00 © 1992 IEEE

Computer 25:7 (1992), pp. 10-22. doi: 10.1109/2.144436

Glossary

AND-OR graph (or tree). Representation of a solution strategy in which a path from the start node to the solution node requires traversing any branch at an OR node and every branch at an AND node. In a related Min-Max search used in two-person games, a path from the start node to the solution node takes the lowest cost branch at a Min node and the highest cost branch at a Max node.

Bitmap. Digital representation of an image in which points are mapped to an array of binary pixels.

Branch-and-bound. A search technique that avoids paths certain to lead to higher cost solutions than the best solution obtained so far.

CCITT (Comite Consultatif International de Telegraphie et Telephonie). International organization which promulgates standards for facsimile coding and transmission. CCITT Group 3 compression methods are used over relatively high-error-rate channels such as the telephone network. Group 4 standards are designed for low-error-rate channels such as public data networks. The compression ratio of a code is the number of bits in the bitmapped representation of the image divided by the number of bits in the coded representation.

Compiler tools. Programs that generate lexical and syntactic analysis programs for particular applications, including assembly, interpretation, compilation, and translation of computer programs. In the Unix system, Lex is a popular tool for lexical analysis and YACC (yet another compiler compiler) is used for parsing context-free languages.

Document interchange formats. Standards for coding the organization of the layout and contents of a document to facilitate transferring documents from one computer system to another. Popular examples include the Document Content Architecture (DCA), Document Interchange Format (DIF), Document Style Semantics Specification Language (DSSSL), Office Document Architecture (ODA), and Standard Generalized Mark-up Language (SGML).

Drop cap. An oversized uppercase character spanning several printed lines, often used to begin an article or a section.

Hypertext. Data structure for text or other media that includes linkages (pointers) between conceptually related items. These linkages allow considerable latitude in selecting strategies for traversing many levels of information. The linkages are either preset or created by data users.

Kerning. Separate characters placed closer together than usual by removing some of the space between them.

Layout. Physical (for example, geometric and typographic) organization of textual and graphic matter in a document, in contrast to content.

Leading. The separation between consecutive lines of text, originally achieved in printing by inserting strips of lead between rows of type.

Lexical analysis. Each token (word) belongs to a lexical category that determines what tokens may precede or follow it. In English, examples of lexical categories are verb, noun, and pronoun. Lexical analysis determines the category of a token.

Ligature. Two or more symbols, such as æ, printed as a single pattern.

OCR (optical character recognition). The technology of converting printed or written material into computer-readable (ASCII) code. The word *optical* serves to distinguish it from magnetic ink character recognition (MICR). OCR systems usually include an optical scanner, a preprocessor to locate the characters in the appropriate order, a pattern classifier, and a contextual postprocessor.

Page icon. A reduced-scale representation of the subdivision of a printed page into nested rectangles that correspond to significant layout units. It is a graphic display of the X-Y tree of the decomposition.

Point. Printer's unit of measurement, equal to about 1/72 of an inch (US, European, and Belgian points are slightly different). One pica equals 12 points. The notation 10/12 indicates 10-point type with 2-point leading (white space).

PostScript. A programming language introduced by Adobe Systems that allows device-independent specification of typeset text and illustrations for computer printers, displays, and digital phototypesetters. Such languages are generally known as page-description languages.

Syntax. Synonymous with grammar: the formal rules that determine the permissible configuration of lexical tokens such as words. Parsing (a form of syntactic analysis) determines whether a string of words forms a legal sentence. The language accepted by a grammar is the (often infinite) set of all legal sentences. A grammar is usually expressed as a set of rewrite rules, or productions, that allow replacing the string of symbols found on the left-hand side of the production by the string found on the right-hand side to generate legal sentences. Symbols that don't occur by themselves on the left-hand side and therefore cannot be replaced are called terminal symbols; all others are called nonterminal symbols. Formally, a grammar is a quintuple consisting of a starting symbol, a delimiter, a set of terminal symbols, a set of nonterminal symbols, and a set of productions.

TIFF (tagged image-file format). A family of popular formats for color, gray-level, and black-and-white images in either compressed or uncompressed form.

VGA (video graphic adapter). A format convention for 480 × 640-pixel color screen displays introduced by IBM in 1987.

X-Y tree. A spatial data structure in which each node corresponds to a rectangular block. The children of each node represent the locations of the subdivisions of the parent block in a particular (horizontal or vertical) direction. The same organization is often represented in VLSI design by means of a polar graph. Other popular hierarchical data structures for isothetic (Manhattan) spatial subdivisions include the K-D tree and the Quad tree.

Advantages of intelligent page segmentation

Smaller blocks can be reproduced on PC displays without horizontal scrolling or loss of legibility.

Selected blocks can be transmitted more quickly than an entire page can for browsing over networks.

Layout analysis helps OCR preserve the reading order.

Key-entry of selected blocks that are intractable by OCR is less costly than rekeying entire pages.

Differentiating text from graphics and halftones leads to more efficient image compression.

Straight borders between high-contrast and halftone regions are preserved for digital reprographics.

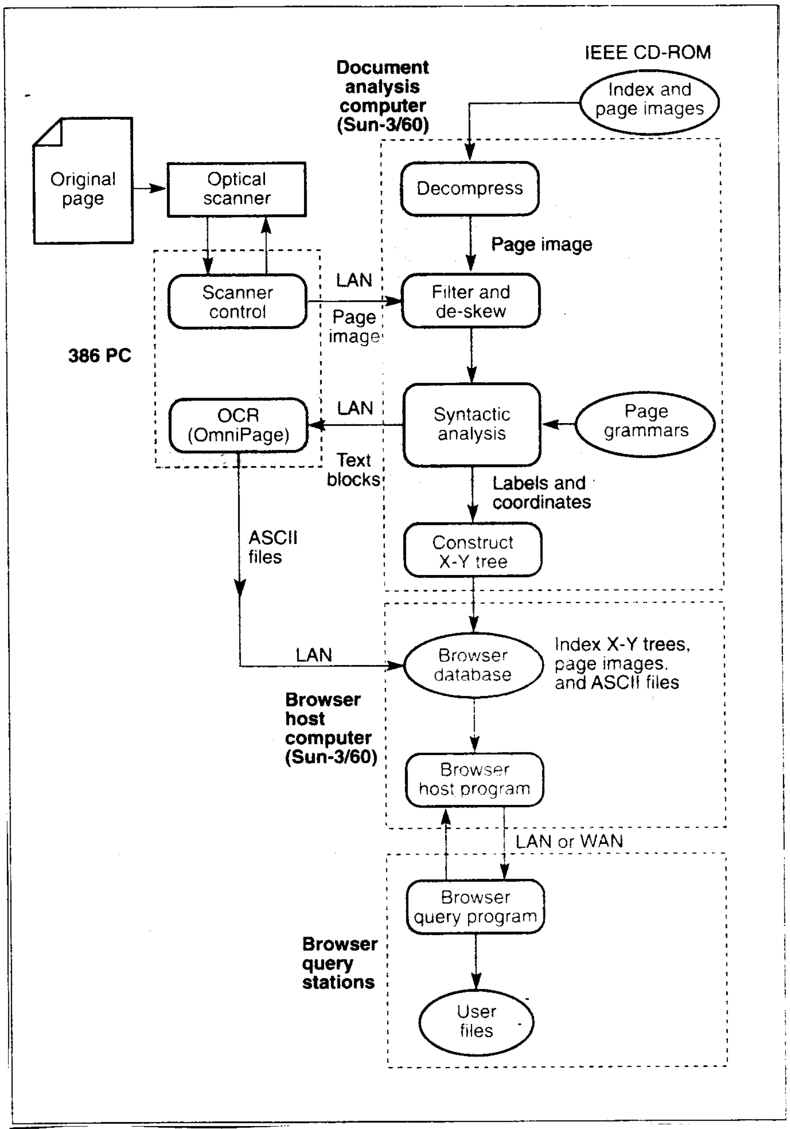


Figure 1. Gobbledoc system overview.

labeling the page image according to its logical structure, without resorting to optical character recognition (see the "Advantages of intelligent page segmentation" sidebar).

Research on OCR began at the turn of the century. Inventions for replacing telegraph operators and assisting blind readers were first demonstrated in 1914, and high-speed OCR systems have been commercially available since 1955 for specially printed forms and fixed-pitch typescript. The field of digitized page analysis started in the sixties, but only recently have the relevant technologies matured sufficiently to allow the conversion of complex typeset documents such as technical articles. Major catalysts include accurate, fast, inexpensive page scanners; high-speed computer networks; sufficient storage and processing power for digitized page images; and compression standards for digital facsimile.

Document image analysis requires the assembly of a number of software tools.[1] Gobbledoc, the system that we have developed at Rensselaer Polytechnic Institute over the last five years, integrates several existing components with a new method of image decomposition. Figure 1 shows the flow of information from the original document to the user's computer screen. A page is either scanned locally or obtained from a CD-ROM. It is segmented and labeled by using a syntactic approach, and stored in an X-Y tree. Images of text blocks are transferred to a personal computer and recognized by using OCR. The resulting ASCII output (which includes some special characters marked with escape quotes) is linked to the corresponding node of the X-Y tree. All data is then stored on the browser host computer. The browser is activated from a remote client workstation, and the user-requested information is displayed on the client terminal.

In Gobbledoc, image processing, document analysis, and OCR operations take place in batch mode when the documents are acquired. After explaining the document image acquisition process, we describe the knowledge base that must be entered into the system to process a family of page images. We show how our X-Y tree data structure converts the two-dimensional page-segmentation problem into a series of one-dimensional string-parsing problems that can be tackled by using conventional compiler tools. Next, syntactic analysis is used to divide each page into labeled rectangular blocks. Blocks labeled *text* are converted by OCR to obtain a secondary (ASCII) document representation. But such symbolic files are better suited for computerized search than for human access to the document content. Because too many visual layout clues are lost in the OCR process (including some special characters), our system preserves the original block images for human browsing. Finally, we consider storage, networking, and display issues specific to document images, and describe our prototype browser.

Partial layout specifications for *Computer* in 1991

The title lines are set in Melior 36/38 point boldface, centered.

There are one to four lines in the title: the second part of a long title is sometimes set in 24-point type.

The byline is set in Melior 12/14-point boldface, centered; affiliations may either follow the authors' names set in the same typeface or are set on the same line.

The title line precedes the byline, and the two are separated by at least 38-point leading.

Every paragraph is indented except the first and the beginning of the conclusion section, which begin with a 40/40-point drop cap.

The body type (main text) is TimesTen Roman 9/11.

The page numbers and month-year (in body type) are set flush with the margin and alternate between left and right.

Footnotes (rare) are set TimesTen Roman 7/8, separated by a hairline rule with a blank line before and after the rule.

Preprocessing

We convert each page to digital form by scanning it at 300 dpi. This rate is sufficient to preserve all significant white spaces. Since the entire X-Y tree approach is extremely sensitive to skew, we carefully align each page on the scanner bed. We recognize that this could be impossible in a production environment, where the systems could incorporate one of the excellent skew-correction methods (performed by software or hardware) already demonstrated by others.[2]

Sample pages. We also take sample pages from the CD-ROM database prepared for the IEE-IEEE by University Microfilms. The IEEE annually distributes about 130,000 pages in 300-dpi image form on CD. Other professional societies are not far behind. The IEE-IEEE database is stored in CCITT Group 4 compressed form. This is the two-dimensional Modified Modified Read (MMR) code developed as a standard for facsimile terminals operating over very-low-error-rate public data networks.[3] Currently, we decompress the page using software, but standard chips for this purpose are available. Since we have no control over the degree of skew of the CD pages, we de-skew them.

Noise. Our documents contain specks of noise caused by flawed fibers in the stock, imperfect printing, photocopying, and digitization. (High-quality journals are relatively noise-free, but we used photocopies on uncoated paper.) Such noise does not bother human readers, but it complicates automated analysis. To cope with the noise, one may construct page grammars sufficiently robust to ignore speckle. This is feasible but tedious. Instead, we filter out all specks smaller than a given size in a preliminary pass. After the analysis is completed, all specks are restored. The size threshold therefore can be quite generous. Losing a few periods or dots on the *i*'s and *j*'s does not affect the layout analysis, and they are restituted before any document component is submitted to OCR or displayed for human inspection.

Layout encoding

The first step in analyzing a specific family of documents is encoding the information that distinguishes this family from others. The required knowledge base is quite specific. It may be stated explicitly in the form of if-then rules, a form-definition language, a geometry tree of box locations, an expert system, or a formal grammar — or implicitly embedded in the processing programs. Some examples of constraints for the title page of feature articles in *Computer* as of July 1991 are shown in the accompanying sidebar. The rules for the pages from *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE-PAMI)*, used in our experiments and shown later in Figure 6, are similar.

Publication-specific systems have been successfully demonstrated on newspaper pages, business letters, articles, and tables of contents in technical journals, patent applications, resumes, typed forms with a prespecified layout, sheet music, maps, engineering drawings, circuit diagrams, and even the periodical *Chess Informant*. A complex knowledge-based system that demonstrates the usefulness of full interaction between different components has been developed for postal address location and interpretation. Recent achievements in these areas are described in the proceedings of conferences on document analysis.[4,5]

Syntactic analysis. Just as every programming language has its fixed set of rules, each publication has predetermined layout conventions. These conventions dictate the size, position, spacing, and ordering of the blocks that correspond to logical entities on a page, as shown in the sidebar for *Computer*. Accordingly, documents that satisfy the postulated layout conventions can always be parsed into different components without OCR in the same way that syntactically correct program code can be parsed into meaningful constructs without semantic understanding.

Syntactic analysis accomplishes both segmentation and component identification simultaneously. A publication-specific *document grammar* is a formal description of all legal page formats that articles in a given publication can assume. In principle, a parse of the page will reveal whether the page is acceptable under the rules of the grammar and, if so, what labels must be assigned

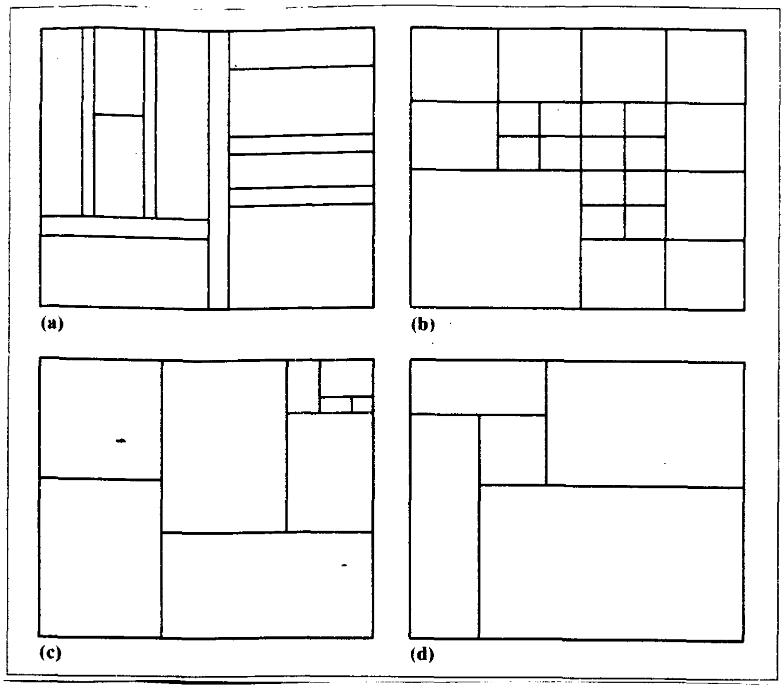


Figure 2. Hierarchical subdivision of rectangles into rectangles: X-Y tree for page segmentation (a); Quad tree for comparing or combining several images (b); K-D tree for fast search (c); example of tiling with rectangular blocks that cannot be obtained by successive horizontal and vertical subdivisions only (d).

to the various components to construe a valid graphic "sentence."

Although we use compiler tools developed primarily for formal languages, the syntactic analysis of document images exhibits many of the difficulties of parsing natural language. Layout conventions may be insufficient to identify *every* document component. For instance, text lines with equations buried in them may radically alter the expected line spacing. We must therefore ensure that minor deviations have only local effects. Furthermore, the grammar for a modern programming language is established from the start, while document grammars must be inferred indirectly, as later discussed. (A never-ending task: Journals frequently put on new faces.)

Block grammars. The document grammar for a specific journal consists of a set of block grammars. Each block grammar subdivides a block horizontally or vertically into a set of subblocks. The net result of applying the entire document grammar is therefore a subdivision of the page into *nested rectangular blocks*. Such a subdivision can be represented efficiently in a data structure called the X-Y tree[6] (Figure 2). The block grammars themselves are also organized in the form of a tree: The block grammar to be used to subdivide each block is determined recursively by the results of the parse at the level above.

Syntactic attributes

A (horizontal/vertical) *block profile* is a binary string that contains a zero for each horizontal or vertical scanline that contains only white pixels; otherwise it is a one.

A *black atom* is a maximal all-one substring. It is the smallest indivisible partition of the current block profile. A *white atom* is an all-zero substring.

A *black molecule* is a sequence of black and white atoms followed by a black atom. A *white molecule* is a white atom that separates two black molecules.

An *entity* is a molecule that has been assigned a *class label* (title, authors, figure caption). It may depend on an ordering relationship.

This approach effectively transforms the difficult two-dimensional segmentation into a set of manageable one-dimensional segmentation problems.

The syntactic formalism is theoretically well understood, and sophisticated software is available for lexical analysis and parsing of strings of symbols. Each block grammar is therefore implemented as a conventional string grammar that operates on a binary string called a *block profile*. The block profile is the thresholded vertical or horizontal projection of the black areas within the block. Zeros in the block profile correspond to white spaces that extend all the way across the block and are therefore good candidates for the locations of subdivisions.

Representing the structure of an entire page in terms of block grammars simplifies matters considerably. But each block grammar itself is a complex structure. It must accommodate many alternative configurations. For instance, to divide the title block from the byline block, the block grammar must provide for a varying number of title lines and bylines, and for changes in spacing caused by the ascenders and descenders of the letters. To simplify the design process, each block grammar is constructed in several stages, in terms of syntactic attributes extracted from profile features.[7]

Syntactic attributes. The first stage of a block grammar operates on the ones and zeros of the block profile. Strings of ones or zeros are called *atoms*. Atoms are divided into classes according to their length. A string of alternating black and white atoms is a *molecule*. The class of a molecule depends on the number and kind of atoms it contains. Finally, molecules are transformed into *entities* depending on the order of their appearance. The words *atom*, *molecule*, and *entity* were chosen because they are not specific to a particular publication or subdivision. (See the sidebar on syntactic attributes.)

The syntactic attributes that determine the parse are the size and number of atoms within an entity, and the number and order of permissible occurrences of entities on a page. Table 1 shows the expected variation in the horizontal profile of a page fragment that includes the title and byline. The assignment of symbols into larger units is accomplished by rewriting rules or *productions*. These

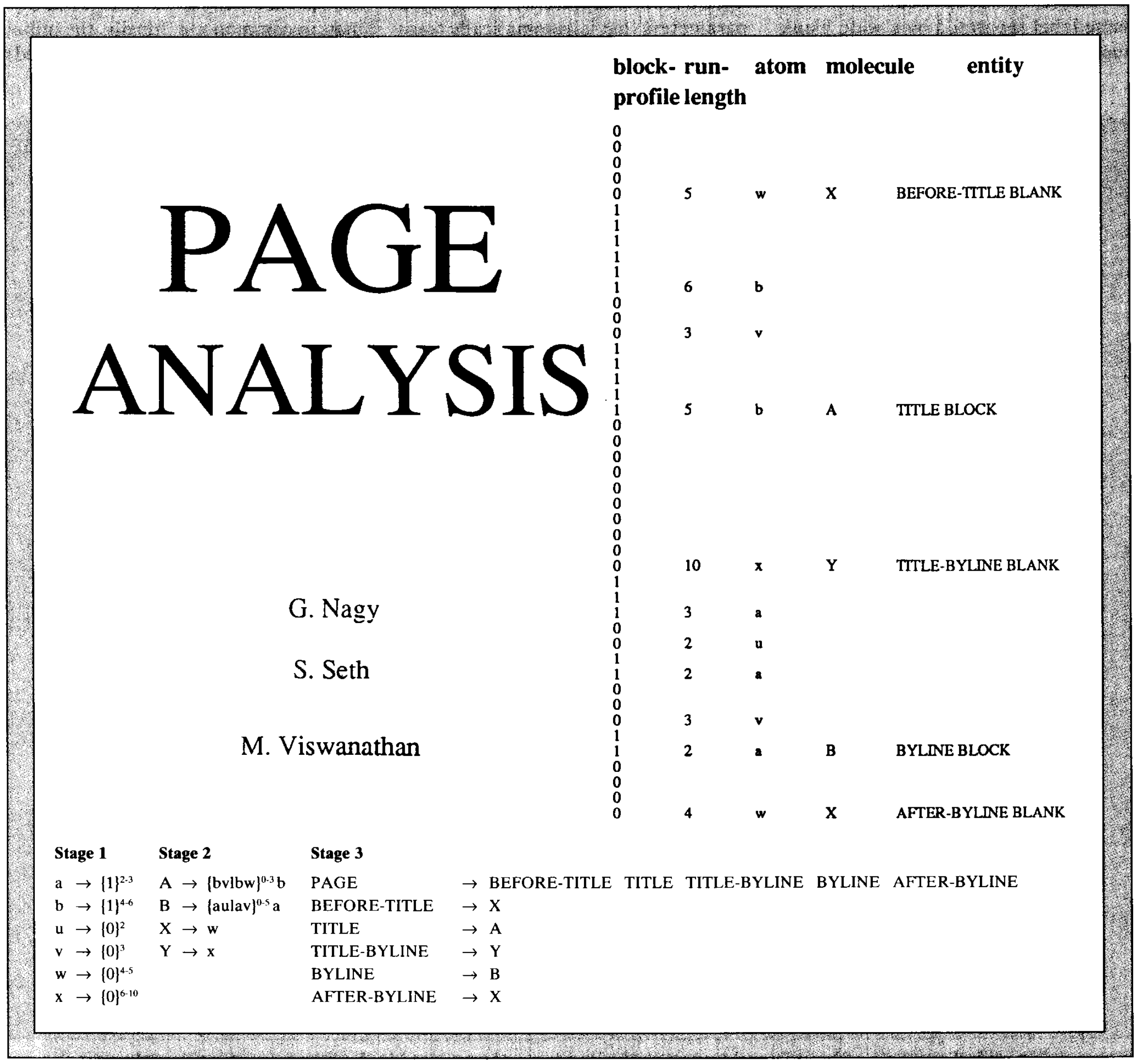
Table 1. Examples of syntactic profile features for *Computer* digitized at 300 dpi.

	Title	Title-byline-space	Byline
Black atom length	100-160	—	35-52
White atom length	4-20	10-40	2-14
No. atoms	1-4	—	1-6
No. entities	1	1	1
Precedence		Title before byline	

rules take into account the expected variability. The entries in the first row correspond to the maximum and minimum height (in pixels) of the title and byline lines in a feature article scanned at 300 dpi. The second row shows the expected profile height of the leading. There must be one to four title lines and one to six author lines, with the byline below the title. Each page has only a single title block and a single byline block.

In the greatly simplified example of Figure 3, a typical horizontal block profile is shown at the top. The atom lengths have been shortened to show the entire horizontal block profile.

In stage 1, runs of ones or zeros are condensed into atoms according to their length. For example, black runs of 2 to 3 are called a. In stage 2, atoms are grouped into molecules. (A run of zero to three black-white sequences of two atoms of types b,v or b,w, followed by a black atom of type b, is labelled A.) In stage 3, molecules are interpreted as *document entities.*



block-profile	run-length	atom	molecule	entity
0				
0				
0				
0				
0	5	w	X	BEFORE-TITLE BLANK
1				
1				
1				
1				
1				
1	6	b		
0				
0				
0	3	v		
1				
1				
1				
1				
1	5	b	A	TITLE BLOCK
0				
0				
0				
0				
0				
0				
0				
0				
0				
0	10	x	Y	TITLE-BYLINE BLANK
1				
1				
1	3	a		
0				
0	2	u		
1				
1	2	a		
0				
0				
0	3	v		
1				
1	2	a	B	BYLINE BLOCK
0				
0				
0				
0	4	w	X	AFTER-BYLINE BLANK

Stage 1	Stage 2	Stage 3	
$a \rightarrow \{1\}^{2\text{-}3}$	$A \rightarrow \{bv \mid bw\}^{0\text{-}3} b$	PAGE	$\rightarrow$ BEFORE-TITLE TITLE TITLE-BYLINE BYLINE AFTER-BYLINE
$b \rightarrow \{1\}^{4\text{-}6}$	$B \rightarrow \{au \mid av\}^{0\text{-}5} a$	BEFORE-TITLE	$\rightarrow$ X
$u \rightarrow \{0\}^{2}$	$X \rightarrow w$	TITLE	$\rightarrow$ A
$v \rightarrow \{0\}^{3}$	$Y \rightarrow x$	TITLE-BYLINE	$\rightarrow$ Y
$w \rightarrow \{0\}^{4\text{-}5}$		BYLINE	$\rightarrow$ B
$x \rightarrow \{0\}^{6\text{-}10}$		AFTER-BYLINE	$\rightarrow$ X

Figure 3. Illustration of a simplified block grammar for the first vertical subdivision of a title page similar to that of *Computer*.

The block grammar at the bottom specifies that the title block (molecule A) may have one to four lines of print, separated by blank lines. Each line of print (atom b) may be four to six pixels high, and each blank line may be three (v), four, or five (w) pixels high. (The height of the blank lines is divided into two ranges because of the overlap in the range of the heights of the blank lines in the title and author blocks.) Stage 3 of this block grammar ensures that the title block precedes the byline block.

Note that molecule X is sometimes interpreted as a before-title blank and sometimes as an after-byline blank, depending on its location with respect to other molecules. Stage 4 (not illustrated) merges consecutive entities with the same label separated by a wide blank, such as text paragraphs.

Block grammars based on such attributes typically have hundreds of productions for a single publication such as *IEEE-PAMI*. Defining each block grammar directly on the binary profile strings, while theoretically equivalent, would be too cumbersome in practice.

Grammar specification. Constructing the block grammars by hand is still very time-consuming and is the greatest weakness of our current approach. Available sources of information are computer-aided measurements on samples of scanned copy, publishers' style manuals, and macro definitions customized for a given publication in page formatters. We have also developed a tabular representation of layout parameters that is automatically translated into block grammars.

Appealing alternatives to our tabular representation of document layout include the Office Document Architecture (ODA), Standard Generalized Mark-up Language (SGML), and Document Style Semantics Specification Language (DSSSL) document interchange standards.[8] However, it is not easy to extract from these specifications the criteria for alternating horizontal and vertical subdivisions required by our decomposition.

Low-level block grammars tend to be much more generic (exhibit less publication-dependent variability) than high-level grammars. One can therefore often reuse existing block grammars for a new family of publications. There are, for instance, only a few paragraph formats in common use. Ideally, the initial parameters would be adjusted (learned)

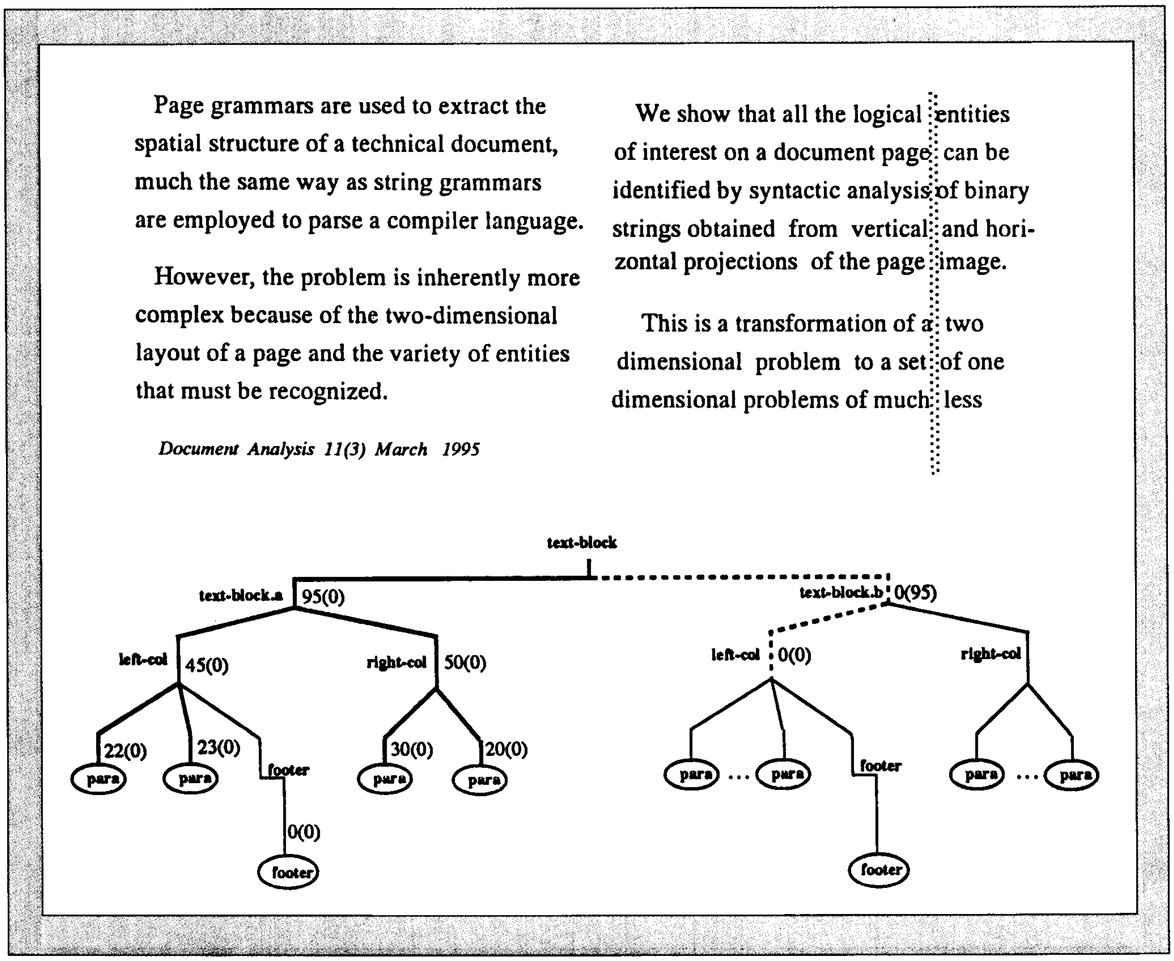

Page grammars are used to extract the spatial structure of a technical document, much the same way as string grammars are employed to parse a compiler language.
However, the problem is inherently more complex because of the two-dimensional layout of a page and the variety of entities that must be recognized.
Document Analysis 11(3) March 1995
We show that all the logical entities of interest on a document page can be identified by syntactic analysis of binary strings obtained from vertical and horizontal projections of the page image.
This is a transformation of a two dimensional problem to a set of one dimensional problems of much less
text-block
text-block.a
95(0)
text-block.b
0(95)
left-col
45(0)
right-col
50(0)
left-col
0(0)
right-col
22(0)
23(0)
30(0)
20(0)
para
footer
0(0)

Figure 4. Simplified example of branch-and-bound analysis: a two-column text block with a *river* (accidental vertical alignment of blank spaces) (a), and a block grammar *text-block.a* dividing the block at the central gutter (b).

during actual operations; this is a subject of continuing research.

Page decomposition. We now show how the block-grammar concept can be extended to segment and label a whole page. A block grammar is intended to interpret each subblock of a given block as a particular (labeled) entity. At the subblock level, block grammars assign labels to abstract, title-block, byline-block, reference-entry, and figure-caption entities. Each of these lower level entities may have alternative grammars corresponding to different formats. During the parsing of a block, a unique label is associated with each subblock. Only blocks that are assigned labels for which no block grammars are provided are assumed to be correctly segmented and labeled. Every other *nonterminal* symbol assigned by a block grammar is equated with the start symbol of a block grammar at the level below. Unlike the four stages of each block grammar, the set of block grammars cannot be combined into a single master grammar, even in principle, because at each level a new set of block profiles must be extracted, and the location of each block boundary depends on the parse at the level above.

Several dozen different classes of logical components are isolated and identified by parsing the recursively generated block profiles. The recursive algorithm that subdivides the page verifies the correctness of label assignments in a depth-first traversal. It returns with success only if parsing is successful for the given block and all of its nested subblocks. The algorithm may try each of several alternative block grammars in turn. For example, if it is not known what part of an article a given page represents, it may be parsed with two block grammars: one designed for a title page and one for a nontitle page. The parse may therefore be considered an AND-OR tree: At each level, *every* subblock must be identified by means of one of a set of alternative grammars.

Another reason for alternative grammars is that a block profile may have two legal parses. This happens when the label of a block cannot be determined from its own profile, but only at a level below. In that case, the block must be parsed under several hypotheses, using alternative grammars.

Error control. Some portion of the page may be in an unexpected format for which no alternative grammar has been provided. Since this will yield an unrecognizable subblock at some point, the entire parse will fail. The AND-OR approach is therefore modified to avoid catastrophic failure and determine efficiently the *best possible labeling*. "Best" is defined as the *maximum cumulative area* of the labeled blocks. A *branch-and-bound* strategy avoids parsing a subblock if the maximum labeled area of the page cannot be increased over the current lower bound even with complete segmentation and labeling of the subblock (Figure 4). Subblock grammar left-col correctly divides the left column into two paragraphs and a footer, but the footer grammar fails because it expects even smaller type. Grammar right-col correctly processes the right column, so 95 percent of the area (all but the footer block) is now labeled.

In an attempt to improve on this, the second alternative top-level grammar, text-block.b (designed for narrower gutters), is applied to the entire block and divides it at the river. Now, however, left-col fails, because the lines of text do not line up across the central gutter, so the block cannot be segmented into horizontal strips at the prescribed line-spacing. The search is abandoned without attempting the rightmost branches of the tree, because even if the region to the right of the river were correctly labeled, the overall result would be inferior to the earlier parse. The numbers on the branches of the search tree show the percentage of area identified and, in parentheses, the applicable lower bound.

Typographic conventions. Our method uses a publication-specific knowledge base in the form of a document grammar. We have also studied segmentation techniques based on typesetting conventions[9] rather than on publication-specific knowledge. Previous methods generally imbed the typographic constraints in the processing routines. It is therefore difficult to take an existing program and form a clear conception of its capabilities without extensive experimentation. Further progress may depend on the development of more consistent and comprehensive knowledge bases through explicit codification of such information. Some examples of generic typesetting knowledge for technical journals set in derivatives of the Latin alphabet are shown in the sidebar called "Generic typesetting rules." The commercial software that we use for OCR incorporates such typesetting rules to locate and isolate characters in a block of text.

Generic typesetting rules

Printed lines are roughly horizontal.

The baselines of characters are aligned.

Each line of text is set in a single point size.

Ascenders, descenders, and capitals have consistent heights. In roman fonts, serifs are aligned.

Typefaces (including variants such as italic or bold) do not change within a word.

Within a line of text, word and character spaces are uniform, and word spaces are larger than character spaces.

Lines of text in a paragraph are spaced uniformly.

Each paragraph is left-justified or right-justified (or both), with special provisions for the first and last line of a paragraph.

Paragraphs are separated either by wider spaces than lines within a paragraph or by indentation.

Illustrations are confined to rectangular frames.

In multicolumn formats, the columns are spaced uniformly.

OCR

Rather than develop our own OCR, we considered commercially available products.[10-12] High-end systems consist of a combination of hardware and software. Some require only an additional processor board. Low-end systems are software only and typically run on PCs with extended internal storage. Some OCR systems can be trained for specific typefaces.

We selected the OmniPage system from Caere Corp. because of its superior performance (without user training) on typefaces and sizes used in technical journals. Nevertheless, the ASCII version produced by the OCR system is not as accurate as a keyed-in version. Unusual typefaces may baffle the system, but most of the classification errors are due to missegmented characters (particularly in small point sizes, and kerned, bold, underlined, and italicized typefaces). Formulas and equations are sometimes mangled. Lines with drop caps (such as the first letter of this article), large mathematical symbols, or superscripts may be missed altogether. Some mistakes seem irrational and may be repeated many times.

In other applications, where accuracy is essential, the text would be corrected using a spelling checker and human postediting. However, even without expensive postprocessing, the ASCII version obtained by OCR is useful not only for automated search but also because it can be converted into a standard document-interchange format[8] or copied directly into a user's word processor or desktop publishing system. (We leave aside the difficult questions of pricing, copyright law, and plagiarism.)

Text-image compression. We are also attempting to use the output of the OCR program to provide a highly compressed version of the text image. Only the first graphic instance of each character pattern is stored. When a subsequent pattern is identified by the OCR with the same alphabetic label, the bitmaps of the original pattern and the new pattern are compared, and if they match, only a pointer to the original bitmap is stored. The degree of match required governs the fidelity of reproduction. Earlier symbol-matching schemes carried out bitmap comparisons with all previous prototype patterns. We use, instead, an efficient OCR system for pattern identification. On dense all-text printed pages scanned at 300 dpi, preliminary results indicate compression ratios twice as high as CCITT Group 4.

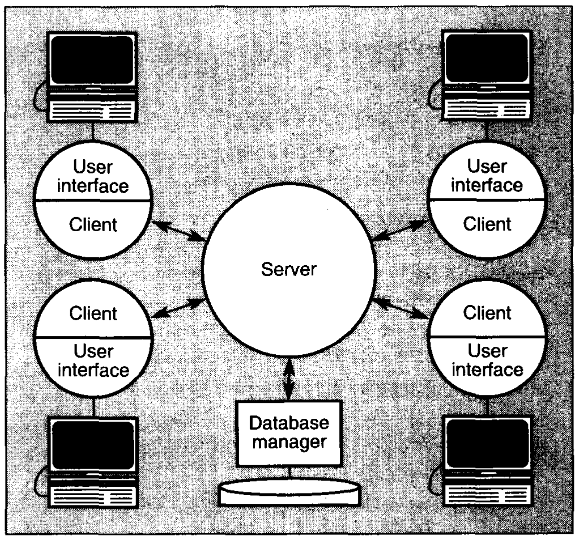


Figure 5. The remote browsing system consists of a central server connected to several query stations by either a local or a wide area network.

Document browser

Remote image browsing systems often use some form of progressive encoding and transmission. Initially, only a low-resolution image is displayed. Greater detail then emerges as additional data is transmitted. Such a scheme is almost useless for printed documents, since a low-resolution version is nearly illegible. Instead, we display parts of the page on demand, using the previously obtained subdivision into meaningful blocks.

The browser system consists of two programs on separate computers that communicate. One is a host program that handles requests from remote users and manages the database of digitized documents, labeled X-Y trees, and ASCII renditions of the textual leaf nodes of the X-Y trees. The other is a (query-station) client program that generates the user requests. The portions of the labeled image requested by the user display at the client station. For portability, we have also implemented an X-Windows version of the browser (see Figure 5).

The host program consists of the server and a database manager. The server handles the communication between the host and the query station. The query station consists of a client and a user interface. The interface transfers a user's request to the client. The client establishes a connection to the host and submits the request. The database manager parses the request received by the server at a specified port and sends the necessary data to the query station. The data received from the server is then transmitted to the user.

The user selects the desired page from a pop-up menu. A geometric representation of the page (a scaled-down labeled X-Y tree with only leaf nodes) displays. The desired block is then retrieved by "mousing" the block in the geometric representation. Using the ASCII button, the user can display either an ASCII version of the text block or a 300-dpi image.

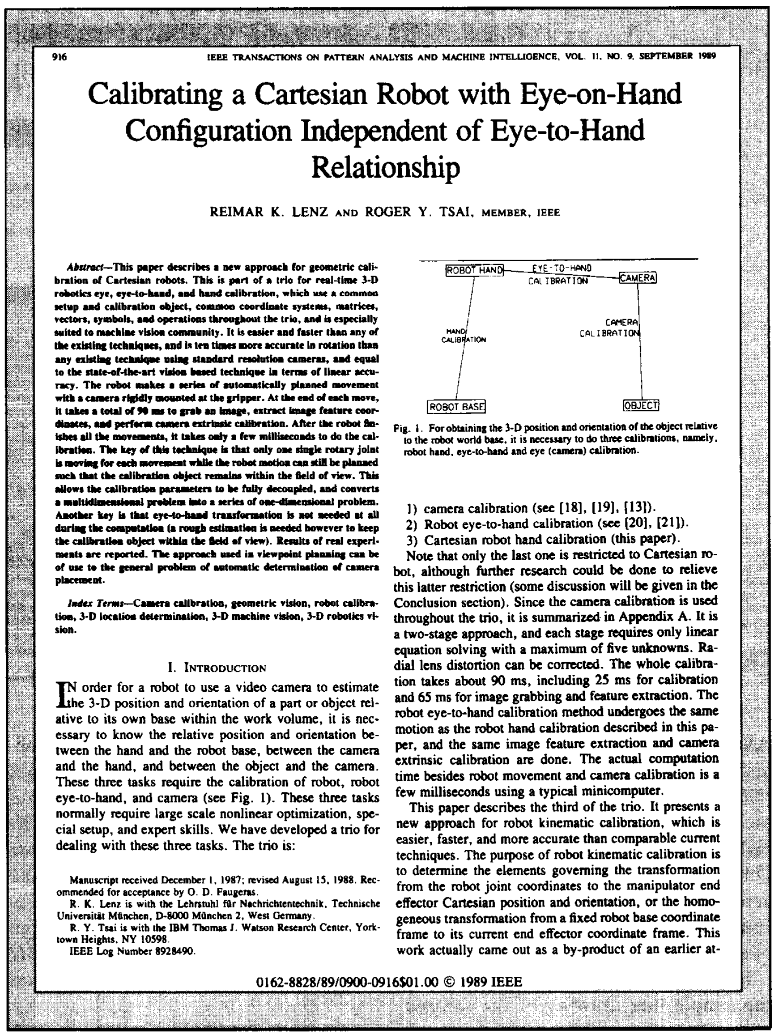
Line wrapping. Most technical printed material is laid out in columns narrow enough for display at 300 dpi on a 480 x 640-pixel VGA screen. However, some type — often the title and abstract — spreads across the full page and cannot be displayed in image form without unwieldy horizontal scrolling. When a text block is wider than a preset width, we continue the segmentation process to the line level. We then locate interword blanks which, in each line of text, are wider than the widest intercharacter space. Each text line is divided at one or more word boundaries and consecutive segments are displayed under one another. This is a form of line-wrapping for text in *image*, rather than symbolic, form.

Linkages. The dual text-image representation provides opportunities to enhance document access. For instance, the OCR system can identify all instances of the word "Figure" in the text and in figure captions. Since we know the location of each figure and figure caption, we can derive the figure number of each illustration. Then, when the user encounters a mention of a figure in the text, depressing the mouse button brings the figure to the screen in a separate window. Alternatively, all figures mentioned in any text paragraph on the screen can be simultaneously displayed with reduced resolution.

Similar concepts apply to cited references. Again, the ASCII version is necessary to set the linkages automatically, but the references can be invoked from either image or ASCII text display. While we have experimented only with linkages within the same document, the concepts can be extended to multiple documents. Image processing plays a role here only in the identification of the coordinates of a recognized keyword in the image and in segmenting the page into blocks. Further enhancement of the information falls in the realm of hypertext.

Implementation. The decompression, de-skewing, noise removal, and syntactic analysis programs run on a Sun-3/60. The uncompressed bitmap of the entire page is stored only once. The profile extraction routines access the required blocks through the *x* and *y* coordinates stored after each parse in the X-Y tree. The parsers at each stage are C-language programs generated by the Unix compiler utilities Lex and YACC. A Unix shell program controls the recursion between levels. Generation and storage of the ASCII files is an off-line process. The MicroTek scanning program EyeStar, the OmniPage OCR system, and the CD-ROM access programs run on an Intel 80386 PC with 4 megabytes of memory.

The host and client programs of the remote browser also run on Sun workstations. All browser interactions are on-line processes. The prototype system has been tested with the host located at the Rensselaer Polytechnic Institute and the query station at the University of Nebraska, and vice versa. A single host can serve several query stations simultaneously. However, our browser does not have the elaborate information retrieval and bibliographic navigation facilities necessary for an operational system. At present, our browser contains only a few dozen pages, which can be selected from a simple menu. Furthermore, its response time is far too slow for anyone browsing in earnest. It does, however, demonstrate several functions related to document image analysis that would be desirable for full access to a technical library through a workstation.



916 IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 11, NO. 9, SEPTEMBER 1989

Calibrating a Cartesian Robot with Eye-on-Hand Configuration Independent of Eye-to-Hand Relationship

REIMAR K. LENZ AND ROGER Y. TSAI, MEMBER, IEEE

Abstract—This paper describes a new approach for geometric calibration of Cartesian robots. This is part of a trio for real-time 3-D robotics eye, eye-to-hand, and hand calibration, which use a common setup and calibration object, common coordinate systems, matrices, vectors, symbols, and operations throughout the trio, and is especially suited to machine vision community. It is easier and faster than any of the existing techniques, and is ten times more accurate in rotation than any existing technique using standard resolution cameras, and equal to the state-of-the-art vision based technique in terms of linear accuracy. The robot makes a series of automatically planned movement with a camera rigidly mounted at the gripper. At the end of each move, it takes a total of 90 ms to grab an image, extract image feature coordinates, and perform camera extrinsic calibration. After the robot finishes all the movements, it takes only a few milliseconds to do the calibration. The key of this technique is that only one single rotary joint is moving for each movement while the robot motion can still be planned such that the calibration object remains within the field of view. This allows the calibration parameters to be fully decoupled, and converts a multidimensional problem into a series of one-dimensional problem. Another key is that eye-to-hand transformation is not needed at all during the computation (a rough estimation is needed however to keep the calibration object within the field of view). Results of real experiments are reported. The approach used in viewpoint planning can be of use to the general problem of automatic determination of camera placement.

Index Terms—Camera calibration, geometric vision, robot calibration, 3-D location determination, 3-D machine vision, 3-D robotics vision.

I. INTRODUCTION

IN order for a robot to use a video camera to estimate the 3-D position and orientation of a part or object relative to its own base within the work volume, it is necessary to know the relative position and orientation between the hand and the robot base, between the camera and the hand, and between the object and the camera. These three tasks require the calibration of robot, robot eye-to-hand, and camera (see Fig. 1). These three tasks normally require large scale nonlinear optimization, special setup, and expert skills. We have developed a trio for dealing with these three tasks. The trio is:

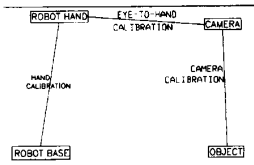


Fig. 1. For obtaining the 3-D position and orientation of the object relative to the robot world base, it is necessary to do three calibrations, namely, robot hand, eye-to-hand and eye (camera) calibration.

1) camera calibration (see [18], [19], [13]).
2) Robot eye-to-hand calibration (see [20], [21]).
3) Cartesian robot hand calibration (this paper).

Note that only the last one is restricted to Cartesian robot, although further research could be done to relieve this latter restriction (some discussion will be given in the Conclusion section). Since the camera calibration is used throughout the trio, it is summarized in Appendix A. It is a two-stage approach, and each stage requires only linear equation solving with a maximum of five unknowns. Radial lens distortion can be corrected. The whole calibration takes about 90 ms, including 25 ms for calibration and 65 ms for image grabbing and feature extraction. The robot eye-to-hand calibration method undergoes the same motion as the robot hand calibration described in this paper, and the same image feature extraction and camera extrinsic calibration are done. The actual computation time besides robot movement and camera calibration is a few milliseconds using a typical minicomputer.

This paper describes the third of the trio. It presents a new approach for robot kinematic calibration, which is easier, faster, and more accurate than comparable current techniques. The purpose of robot kinematic calibration is to determine the elements governing the transformation from the robot joint coordinates to the manipulator end effector Cartesian position and orientation, or the homogeneous transformation from a fixed robot base coordinate frame to its current end effector coordinate frame. This work actually came out as a by-product of an earlier at-

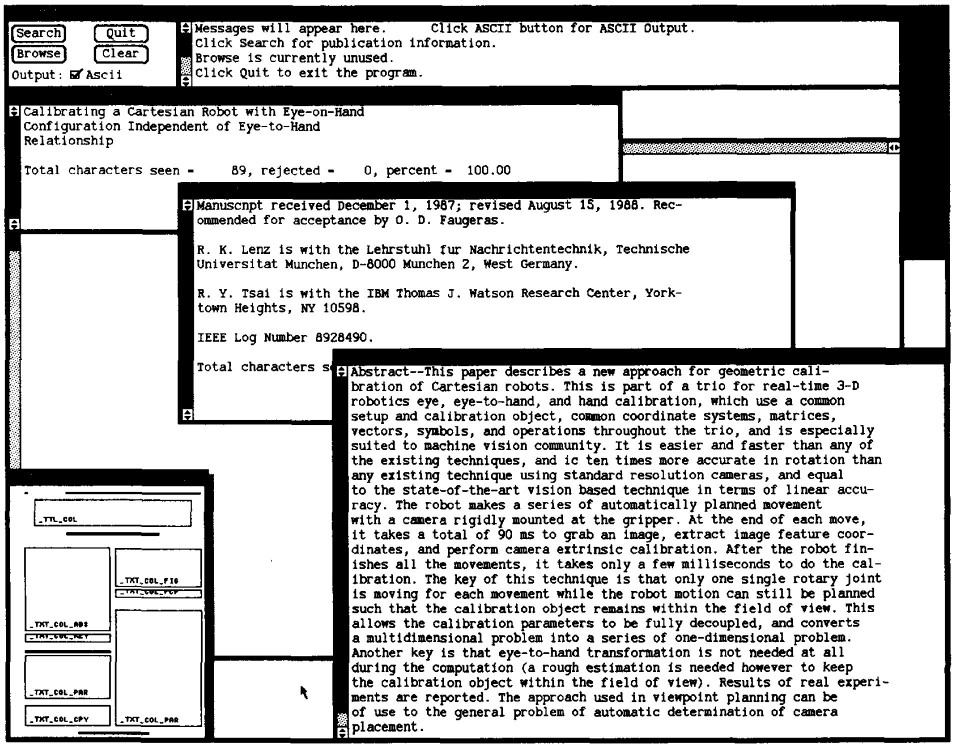
Manuscript received December 1, 1987; revised August 15, 1988. Recommended for acceptance by O. D. Faugeras.
R. K. Lenz is with the Lehrstuhl für Nachrichtentechnik, Technische Universität München, D-8000 München 2, West Germany.
R. Y. Tsai is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.
IEEE Log Number 8928490.

0162-8828/89/0900-0916$01.00 © 1989 IEEE

Figure 6. *IEEE-PAMI* title page extracted from the IEEE CD-ROM database. A de-skewed version is used for OCR, and the image is also filtered for syntactic analysis.

Example. A page from the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, extracted from the CD-ROM and printed on an Apple-Writer from its PostScript representation, is reproduced in Figure 6. The image is de-skewed and filtered before generating a labeled X-Y tree for the page. The X-Y tree contains the coordinates of all segments in the page along with the assigned labels. The 12 subimages in the example that correspond to text are converted into a TIFF (tagged image-file format) and transferred to the PC using a network file-transfer protocol. In the example, these include page number, header, title, byline, abstract, keywords, section titles, two paragraphs of text, figure caption, footnote, and footer. OmniPage is run separately on each subimage and the corresponding ASCII output files are returned to the Sun system.

When the browser is activated, it gives the user successive choices of publication name, volume, issue, and page number. The browser control buttons are shown at the top left corner of Figure 7 on the next page. The final selection prompts the display of a labeled geometric representation of the X-Y tree of the selected page, as shown in the bottom left corner of the screen. Now, the user can choose between ASCII and image renditions of the processed blocks. Multiple requests are entertained, as is evident from Figure 7, which shows the



```
Messages will appear here.      Click ASCII button for ASCII Output.
Click Search for publication information.
Browse is currently unused.
Click Quit to exit the program.
```

```
Calibrating a Cartesian Robot with Eye-on-Hand
Configuration Independent of Eye-to-Hand
Relationship

Total characters seen =     89, rejected =     0, percent =  100.00
```

```
Manuscnpt received December 1, 1987; revised August 15, 1988. Rec-
ommended for acceptance by O. D. Faugeras.

R. K. Lenz is with the Lehrstuhl fur Nachrichtentechnik, Technische
Universitat Munchen, D-8000 Munchen 2, West Germany.

R. Y. Tsai is with the IBM Thomas J. Watson Research Center, York-
town Heights, NY 10598.

IEEE Log Number 8928490.

Total characters s
```

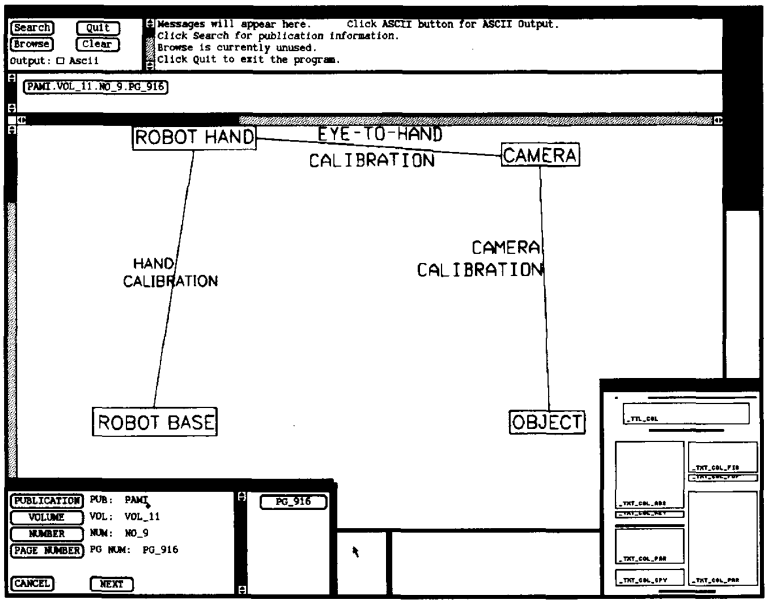
```
Abstract--This paper describes a new approach for geometric cali-
bration of Cartesian robots. This is part of a trio for real-time 3-D
robotics eye, eye-to-hand, and hand calibration, which use a common
setup and calibration object, common coordinate systems, matrices,
vectors, symbols, and operations throughout the trio, and is especially
suited to machine vision community. It is easier and faster than any of
the existing techniques, and ic ten times more accurate in rotation than
any existing technique using standard resolution cameras, and equal
to the state-of-the-art vision based technique in terms of linear accu-
racy. The robot makes a series of automatically planned movement
with a camera rigidly mounted at the gripper. At the end of each move,
it takes a total of 90 ms to grab an image, extract image feature coor-
dinates, and perform camera extrinsic calibration. After the robot fin-
ishes all the movements, it takes only a few milliseconds to do the cal-
ibration. The key of this technique is that only one single rotary joint
is moving for each movement while the robot motion can still be planned
such that the calibration object remains within the field of view. This
allows the calibration parameters to be fully decoupled, and converts
a multidimensional problem into a series of one-dimensional problem.
Another key is that eye-to-hand transformation is not needed at all
during the computation (a rough estimation is needed however to keep
the calibration object within the field of view). Results of real experi-
ments are reported. The approach used in viewpoint planning can be
of use to the general problem of automatic determination of camera
placement.
```

Figure 7. ASCII display of three *IEEE-PAMI* text blocks processed by OmniPage.

ASCII renditions of the title, footnote, and abstract blocks. Note that the word "Manuscript" in the footnote block is incorrectly recognized, as is the word "is" in the seventh line of the abstract. OmniPage records the number of characters and the number of rejects but cannot, of course, calculate the number of substitution errors.

Figure 8 shows the 300-dpi version of the simple line drawing of the sample page. Since the page image is currently available on the IEEE CD-ROM only in binary form, photographs would also be displayed with extreme contrast. The image can be either scrolled or zoomed if its size exceeds the screen size. Figure 9 shows both the image and ASCII displays of a text block — the figure caption.

Experimental results. We have successfully processed 21 photocopied pages from the *IBM Journal of Research and Development* and 20 pages from *IEEE-PAMI.* Since these pages were used for development and improvement of the page grammars, we then randomly selected 12 pages from each journal for an independent test. Nine of the *PAMI* pages were segmented and labeled perfectly. Three had minor mistakes. Of the 12 *IBM Journal* pages, seven were segmented perfectly, three had minor mistakes, and two missed about one quarter of the page. All mistakes could be corrected by simple modifications of the block grammars, but it is clear that several design-and-test cycles would be required for acceptable performance. An interactive step, similar



```
Messages will appear here.      Click ASCII button for ASCII Output.
Click Search for publication information.
Browse is currently unused.
Click Quit to exit the program.
```

```
PUB:     PAMI
VOL:     VOL_11
NUM:     NO_9
PG NUM:  PG_916
```

Figure 8. Image display of a figure block. This block was selected by clicking on the topmost block in the right-hand column of the geometric representation (bottom right). If the figure is small enough, it is shown at full 300-dpi resolution on the screen; otherwise it is reduced or scrolled.

to postediting in OCR, could also be invoked when the algorithm fails. Since failures can be flagged by the system itself, the overall throughput would not be greatly affected.

The bulk of the two to three minutes required to process each new page is consumed in recursive profile extraction. The algorithms were coded with little regard to efficiency; for instance, we used shell scripts whenever possible. We have recently implemented a profile extraction algorithm on a 32 × 32-processor DAP (distributed array of processors) computer. Initial comparisons show that the time required to extract the horizontal and vertical profiles of a 2,000 × 3,000-pixel image is reduced to one tenth.

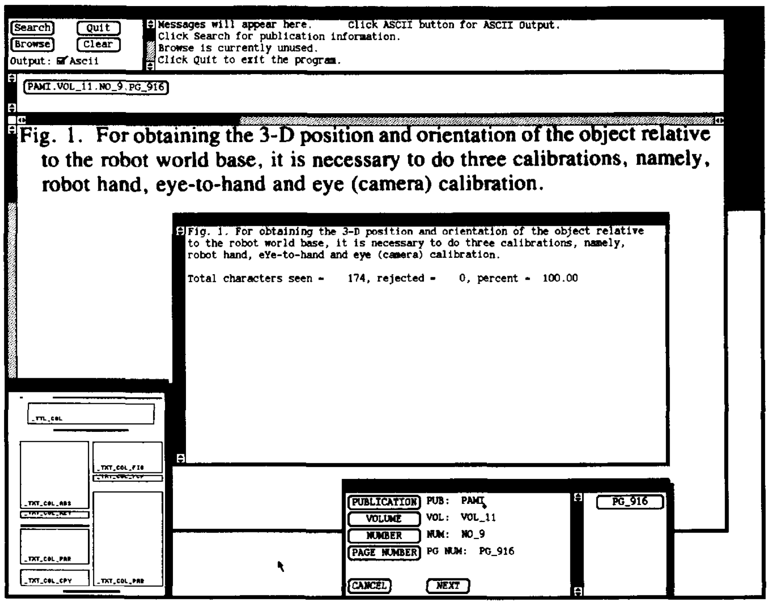
Although an ASCII representation of technical documents is adequate for many purposes, a faithful rendition of the original layout is highly desirable for human access. Not only is this rendition necessary for graphics, equations, and tables whose computer representation is not yet standardized, but also preservation of the original layout and typography enhances legibility compared with OCR output, which preserves only some of this information.

We have demonstrated a prototype version of a system, based on syntactic document analysis and OCR, that can provide useful remote access to stored technical documents. The two aspects that differentiate our system from others are (1) X-Y tree data structures that are particularly suitable for printed matter, and (2) syntactic analysis of image blocks at increasing levels of refinement.

We are now interfacing our prototype browser with the Rensselaer Polytechnic Institute library information system, Infotrax. The system already includes all IEEE indexing data, including abstracts, since 1988. Infotrax yields, for each article of potential interest, the publication title, volume, issue, and page number. Our sample documents are already indexed with the corresponding information.

In addition to its use for information retrieval, we intend to adapt our system to provide input for automated or interactive indexing. Once an article is processed, only ASCII versions of such relevant fields as title, author, and abstract would be forwarded to the indexing station. The body of the text would remain available for subsequent full-text searches, as opposed to searches on selected index terms only.

Longer-term research objectives include developing improved methods for the acquisition of publication-specific knowledge bases, possibly including some form of learning. We are, however, also investigating to what extent documents can be analyzed by using generic typesetting knowledge only. ■



```
Messages will appear here.      Click ASCII button for ASCII Output.
Click Search for publication information.
Browse is currently unused.
Click Quit to exit the program.
```

PAMI.VOL_11.NO_9.PG_916

Fig. 1. For obtaining the 3-D position and orientation of the object relative to the robot world base, it is necessary to do three calibrations, namely, robot hand, eye-to-hand and eye (camera) calibration.

```
Fig. 1. For obtaining the 3-D position and orientation of the object relative
to the robot world base, it is necessary to do three calibrations, namely,
robot hand, eYe-to-hand and eye (camera) calibration.

Total characters seen -    174, rejected -    0, percent -  100.00
```

PUBLICATION PUB: PAMI
VOLUME VOL: VOL_11
NUMBER NUM: NO_9
PAGE NUMBER PG NUM: PG_916
PG_916

Figure 9. Image and ASCII versions of the figure-caption block. The lowercase "y" in "eye-to-hand" in the last line was incorrectly identified as uppercase.

Acknowledgments

Former and current students who have contributed extensively to the project are J. Kanai, E. McCaughrin, R. Sathyamurthy, N. Shirali, and J. Waclawik. Our collaborators at Rensselaer include M. Krishnamoorthy and T. Spencer, and Associate Director of Rensselaer Libraries P. Molholt. We are pleased to thank H.S. Baird (AT&T Bell Laboratories) for document-compression software and E. Annese (Olivetti Corp., Italy) for de-skewing software. The work was supported in part by US West Advanced Technologies and the US Department of Education College Library Technology and Cooperation Grants Program.

References

1. R.G. Casey and K.Y. Wong, "Document-Analysis Systems and Techniques," *Image Analysis Applications*, R. Kasturi and M.M. Trivedi, eds., Marcel Dekker, New York, 1990, pp. 1-36.

2. H.S. Baird, "The Skew Angle of Printed Documents," *Proc. SPSE 40th Conf. Symp. Hybrid Imaging Systems*, SPSE, Springfield, Va., 1987, pp. 14-21.

3. K.R. McConnell, D. Bodson, and R. Schaphorst, *Fax: Digital Facsimile Technology and Applications*, Artech House, Norwood, Mass., 1989, p. 96.

4. *Structured Document Image Analysis*, H.S. Baird, H. Bunke, and K. Yamamoto, eds., Springer-Verlag, 1992, in press.

5. *ICDAR, Proc. First Int'l Conf. Document Analysis and Recognition*, AFCET, Paris, 1991.

6. G. Nagy and S. Seth, "Hierarchical Representation of Optically Scanned Documents," *Proc. Seventh Int'l Conf. Pattern Recognition*, Order No. M545 (microfiche), IEEE CS Press, Los Alamitos, Calif., 1984, pp. 347-349.

7. M. Viswanathan, "Analysis of Scanned Documents — A Syntactic Approach,"

in *Structured Document Image Analysis*, H. Baird, H. Bunke, and K. Yamamoto, eds., Springer-Verlag, 1992, in press.

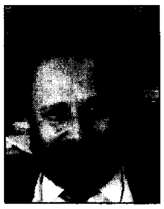
8. J.-M. De La Beaujardiere, "Well-Established Document Interchange Formats," in *Document Manipulation and Typography*, J.C. van Vliet, ed., Cambridge University Press, 1988, pp. 83-94.

9. R. McLean, *The Thames and Hudson Manual of Typography*, Thames and Hudson, London, 1980.

10. R. Bradford and T. Nartker, "Error Correlation in Contemporary OCR Systems," *Proc. First Int'l Conf. Document Analysis and Recognition*, AFCET, Paris, 1991, pp. 516-523.

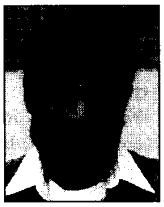
11. V. Garza et al., "OCR Product Comparison," *Infoworld*, Oct. 22, 1990, pp. 73-90.

12. L. Grunin, "OCR Software Moves into the Mainstream," *PC Magazine*, Oct. 30, 1990, pp. 299-350.



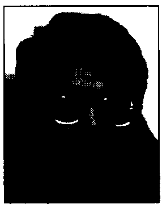
George Nagy has been professor of computer engineering at Rensselaer Polytechnic Institute since 1985. Before that he was professor of computer science at the University of Nebraska at Lincoln, where he worked on remote sensing applications, geographic information systems, computational geometry, and human-computer interfaces. He also conducted research on various aspects of pattern recognition and OCR at the IBM T.J. Watson Research Center for 10 years. In addition to digitized document analysis and character recognition, his interests include solid modeling, finite-precision spatial computation, and computer vision.

Nagy received the BEng degree in engineering physics and the MEng degree in electrical engineering from McGill University and the PhD degree in electrical engineering (on neural networks) from Cornell University in 1962. He is a senior member of the IEEE, a member of the Computer Society, and a member of ACM.



Sharad Seth is a professor in the Computer Science and Engineering Department at the University of Nebraska at Lincoln. Besides document analysis, his research has been in testing and testable design of microelectronic circuits.

Seth received the BEng degree in 1964 from the University of Jabalpur, India, the MTech degree in 1966 from the Indian Institute of Technology, Kanpur, and the PhD degree in electrical engineering in 1970 from the University of Illinois at Urbana-Champaign. He serves on the editorial boards of *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, and the *Journal of Electronic Testing: Theory and Applications* (*JETTA*). Seth is a senior member of the IEEE, a member of the Computer Society, and a member of ACM.



Mahesh Viswanathan is a development staff member with IBM Storage Systems Products Division, San Jose, California. His research interests include printing and document analysis.

Viswanathan received the BSc degree in physics in 1980 from Loyola College, Madras, India, and the BE degree in electrical engineering in 1984 from the Indian Institute of Science, Bangalore. He received the MS degree in electrical engineering from San Diego State University in 1986 and the PhD degree in computer and systems engineering in 1990 from Rensselaer Polytechnic Institute.

Readers can contact George Nagy at Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180; (518) 276-6078; fax (518) 276-6261; e-mail gnagy@mts.rpi.edu.

VICTORIA UNIVERSITY OF TECHNOLOGY

FOOTSCRAY CAMPUS

PO Box 14428, MMC, Melbourne, Vic. Australia 3000
Phone (613) 688 4315. Fax (613) 688 4965

Chair of Computer Science (Level E)

The Department of Computer and Mathematical Sciences wishes to appoint a Senior Academic to the Chair of Computer Science. The Department is a constituent of the Faculty of Science and offers undergraduate and postgraduate courses and masters and doctoral degrees by research. The Department's course and research interests are directed towards the application of mathematical and computer based techniques to industry and business. The major research interests in computer science are in the areas of software engineering, object oriented methodology, artificial intelligence, networks, parallel processing and image processing.

You will be responsible for providing academic, research and professional leadership in Computer Science but the major responsibility of the position will be the development of a distinctive research profile in Computer Science within the Department. Some teaching duties in your area of expertise will also be expected.

A PhD with relevant professional experience in Computer Science is essential. You will also be expected to have a proven record in research with experience in developing and administering research activities.

The appointment will be made on a tenured basis. Salary will be A$77,900 per annum (Level E). Superannuation, travel and removal allowance and temporary housing assistance are available.

For further information contact Associate Professor Ian Murray, Head of Department on (613) 688 4690.

Contact Footscray Campus Human Resources for a position description and forward your written application to the above address (preferably two copies) quoting Reference Number F/CM02 and stating the names and telephone number(s) of three referees by 20th July, 1992. Quote Campus on envelope.

AK18865

ICPR '92

CALL FOR ATTENDANCE

11th International Conference on Pattern Recognition

August 30-September 3, 1992
The Hague, The Netherlands

For further information contact:

ICPR Secretariat
Delft University of Technology
Electrical Engineering Department
PO Box 5031
NL-2600 GA Delft
The Netherlands
Telephone: 31 (15) 786-052
Facsimile: 31 (15) 622-000
e-mail: icpr@et.tudelft.nl