Major Components of a Complete Text Reading System

SHUICHI TSUJIMOTO AND HARUO ASADA

Invited Paper

This paper describes the document image processes used in a newly developed text reading system. The system consists of three major components: document analysis, document understanding, and character segmentation/recognition.

The document analysis component extracts lines of text from a page for recognition. This procedure finds document constituents such as photographs, graphics, and text lines. To this end, the geometric structure is obtained as a hierarchy of items on the page for modeling the relationships between characters, lines, columns, and the page.

The document understanding component extracts logical relationships between the document constituents. The geometric structure of a document, obtained in the document analysis phase, can be represented by a tree. On the other hand, the logical structure is represented by another tree. A small number of generic rules are introduced to transform the geometric structure into the logical structure.

The character segmentation/recognition component extracts characters from a text line and recognizes them. Characters which touch each other may have several candidates for their break positions, and any segmented area might possibly fit several alternative characters. Therefore, an efficient resolution of ambiguities at each stage is a crucial issue in practical text reading. The authors' approach to this is based on the heuristics of character composition as well as on recognition results for omni-fonts.

Experiments on more than a hundred documents have proven that the proposed approaches to document analysis and document understanding are robust even for multicolumned and multiarticle documents containing graphics and photographs. Experiments have also shown that the proposed character segmentation/recognition method is robust enough to cope with omni-font characters which frequently touch each other.

Keywords— *Text reader, document image processing, document understanding, character segmentation, character recognition, geometric structure, logical structure, tree transformation, multiarticle documents, multicolumned documents, omni-font characters, touching characters, OCR, high speed.*

Manuscript received March 25, 1991.

The authors are with the Research and Development Center, Toshiba Corporation, 1, Komukai Toshiba-Cho, Saiwai-Ku, Kawasaki 210, Japan.

IEEE Log Number 9202593.

I. INTRODUCTION

The range and volume of publications such as newspapers, magazines, and various types of manuals are continuously increasing. At the same time, various computer-aided text processing techniques, such as desktop publishing, text data base management, and machine translation, have become possible. Automation is widely demanded in the keyboard input area, which was conventionally manual, where large amounts of documentation must be converted into a computer-readable form for data entry. A text reader meets this need. A text reader automatically analyzes each page of a document and recognizes characters on the page for input to a computer.

It is important for such a text reader system to have the ability to deal with various kinds of document layouts and omni-font characters. Three major components are essential if this capability is to be realized: document analysis, document understanding [1], and character segmentation/recognition [2]. Document analysis is a component which decomposes a document image into several consistent items which represent coherent components of the document, such as text lines, photographs, and graphics, without any knowledge of the specific format. Document understanding is a component which extracts the logical relationships between the items just described. Character segmentation/recognition is the component which then extracts characters from a text line and recognizes them. The document analysis and document understanding components need to be robust enough to cope with multicolumned and multiarticle documents including graphics and photographs. The character segmentation/recognition component needs to have the ability to read omni-font characters which might touch each other.

Up to now, several techniques for document analysis and document understanding [3]–[14] have been proposed. Wahl *et al.* [3] presented a smearing algorithm which is widely used for document analysis. Toyoda *et al.* [4] extracted Japanese newspaper articles using domain-specific knowledge. Okamoto *et al.* [5] analyzed papers containing mathematical expressions. Baird *et al.* [6] coped with the

extraction of chess games from the Chess Informant. Esposito *et al.* [7] classified document types (patents, scientific papers, etc.). Higashino *et al.* [8] proposed a flexible format understanding method using FDL, i.e., a format definition language. Tsuji *et al.* [9] represented a document structure with a tree. Nakano *et al.* [10] built a business form understanding system incorporating character recognition. Inagaki *et al.* [11] built a special-purpose machine for Japanese document understanding. Masuda *et al.* [12] presented a prototype of a Japanese text reader. Baird [13] proposed a versatile text reader whose layout analysis component was designed to be language-independent. Srihari *et al.* [14] surveyed the current status of document analysis. At the same time, many studies have been proposed for character segmentation/recognition, especially for the segmentation of characters which touch each other. Kahan *et al.* [15] discussed the segmentation of touching characters. Casey *et al.* [16] built a recursive segmentation and classification method for touching characters. Kooi *et al.* [17] analyzed the contour of an image of touching characters.

This paper presents new approaches to document analysis, document understanding, and character segmentation/recognition, which are essential components for a complete text reader system.

One of the essential issues for a text reader system is the ability to handle multiarticle documents containing figures and photographs. Generally, a document has a visual hierarchical structure in its layout. This is called geometric structure here, the hierarchy of which can be represented by a tree. Document analysis extracts this structure as a model for relationships between characters, lines, columns, and a page.

A description for configuration of articles and their components, called logical structure here, orders the reading sequence. The authors have defined document understanding as a transformation from a geometric structure to a logical structure. A small number of rules for this transformation have been proposed, based on the general assumption that a layout is designed according to human reading manner.

For practical systems, a text reader system must read a document accurately at high speed. To meet this requirement, character segmentation for touching characters is one of the critical problems. Touching characters have several candidates for their break positions, which are then confirmed by recursive segmentation and recognition[16], and finally by the linguistic context. Since any area segmented by the break positions might fit several alternative characters, this approach requires a very time consuming process to select real break positions. Therefore, a pruning, i.e., a mechanism for reducing processing time, is important in practical text reading systems. The authors' approach is based on the heuristics of character composition, e.g., an "m" is like a combination of an "r" and an "n," as well as on recognition results for omni-fonts.

Section II describes the hierarchical structure of a document in a logical and geometric way. Section III gives the design for robust document analysis for multicolumned documents. In Section IV, an algorithm for document understanding, namely the transformation rules, is introduced. Section V presents a character segmentation/recognition method for touching omni-font characters. Experimental results on a variety of documents are shown in Section VI.

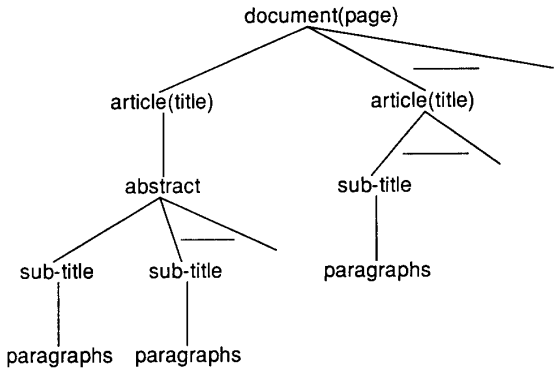


Fig. 1. Logical structure tree. Logical relationships between the items in a document can be represented by a tree.

II. Hierarchical Structure of a Document

This section describes the hierarchical structure of a document in a logical and geometric way.

A. Logical Structure

A document is normally composed of several articles, each of which consists of a title, an abstract, subtitles, and paragraphs. They are connected to each other logically in a hierarchical structure. For example, the title dominates the abstract, chapters, and sections, while subtitles dominate paragraphs. Thus, a document has a logical hierarchy. This logical structure is represented by a tree in this paper, as shown in Fig. 1.

B. Geometric Structure

A document image is composed of several blocks, each of which represents a coherent component of the document. One coherent component corresponds to a set of text lines with the same typeface and a consistent line spacing. The geometric structure, which means the geometric relationships between blocks, is also described by a tree here. Figure 2(b) shows the geometric structure tree generated from the document shown in Fig. 2(a).

In this figure, the root node represents a document page and has the value of *NULL*. Each node in the tree, except the root node, represents a set of adjacent blocks located in the same column, and has a list of blocks as a value. The list is ordered from upper to lower blocks. For example, blocks **1H**, **2B**, and **3H** in Fig. 2(a) are combined and represented as a whole by a node. The parent–daughter relationship is introduced when there is more than one block directly below another block. For example, block **4H** is the parent of blocks **1H** and **5H**. On the other hand, block **1H** is not

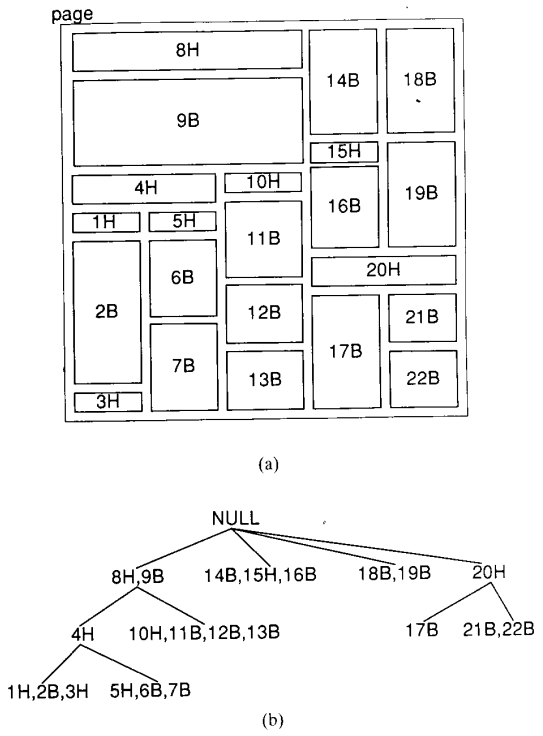


Fig. 2. Geometric structure tree. (a) Document divided into blocks. Each block represents a coherent component of a document. In this illustration, **H** indicates a *head* block, while **B** indicates a *body* block. (b) Geometric structure corresponding to (a). Geometric relationships between blocks can be represented by a tree.

the parent of block **2B** because blocks **1H** and **2B** are in the same column. Those blocks not dominated by others are directly connected to the root node as daughters. Block **20H** in Fig. 2(a) is an example of such a block because it is not dominated by either **16B** or **19B** according to the above discussion. Daughters of the root node are ordered in the sequence from left to right and top to bottom according to their location in the document. This sequence is very important in the transformation process described in Section IV.

Each block is classified as one of two categories: *head* and *body*, in order to distinguish titles from texts. This classification is carried out by examining the physical properties of a block. *Head* is the name for blocks in which there are only a few text lines; text is biased to the left or is centered. Larger type fonts are used in *head* blocks in many cases. This kind of block corresponds to titles or subtitles when a geometric structure is transformed into a logical structure. Headers, footers, page numbers, and captions also belong in this category. *Body* corresponds to blocks consisting of text lines only. It normally has a considerable number of text lines and smaller type fonts. An indentation is often found in the first text line of a *body* block. Abstracts as well as paragraphs belong to this category. In Fig. 2, **H** and **B** indicate *head* and *body* blocks, respectively.

III. Document Analysis

This section describes a document analysis which breaks down a document image into several blocks and constructs a geometric structure tree whose nodes represent a set of blocks.

The authors' approach to document analysis is described as bottom-up. This approach first extracts words from a document image, which are then merged into text lines. Text lines are then combined into blocks which usually correspond to paragraphs.

A geometric structure is generated according to the parent-daughter relationships between blocks. These relationships are established by examining the column a block belongs to, and its vertical position.

A. Run-Length Image Representation

A run-length image representation is used in the proposed system. The run-length representation is more efficient than a bit-map representation, especially in image processing by software. Accessing pixels in a bit-map image is usually very time consuming because a general-purpose computer is not designed to deal with bit-map images stored in a byte representation. Therefore, all the image processes in our system are designed to directly access run-length coded images instead of accessing bit-map images. Using the run-length representation for document image processing yields high-speed processing.

B. Text Line Extraction

The text line extraction procedure is described in this subsection. Here, the authors define a segment as a rectangular area which circumscribes a text line or a part of it.

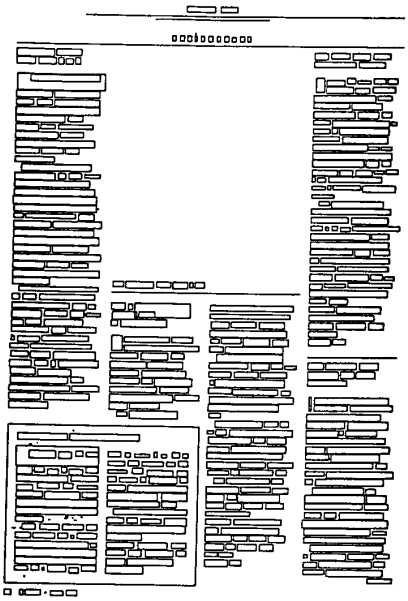
The text line extraction process is divided into four subprocesses. The first is to extract adjacent connected components as a segment. The second classifies the segments into text lines, figures, graphics, and so on. The third is a merging process for adjacent segments which are classified into text lines. The last is another merging process for the segments in the same column defined by the column boundaries. Words are usually extracted by both the first and second subprocesses, and text lines are obtained in the third subprocess. The fourth process is added to cope with cases where a long blank between words prevents the words from being merged into a text line.

These subprocesses are detailed as follows.

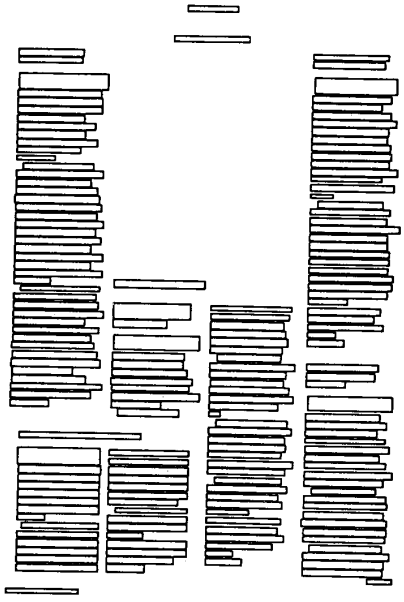
1) Segment Extraction: The first subprocess in the extraction of a text line, extracts adjacent connected components as a segment by connecting two black runs whose spacing (white run) is shorter than a certain threshold (smearing algorithm, see [3]). The threshold is made very small (say 1 mm) so that words located in different columns are not merged with each other. Figure 3(a) shows a sample document image which is an example of a multicolumned, multiarticle document. The result of segment extraction for this sample image is shown in Fig. 3(b).

2) Classification of Segments: Segments are classified into text lines, figures (photographs), tables, frames, horizontal

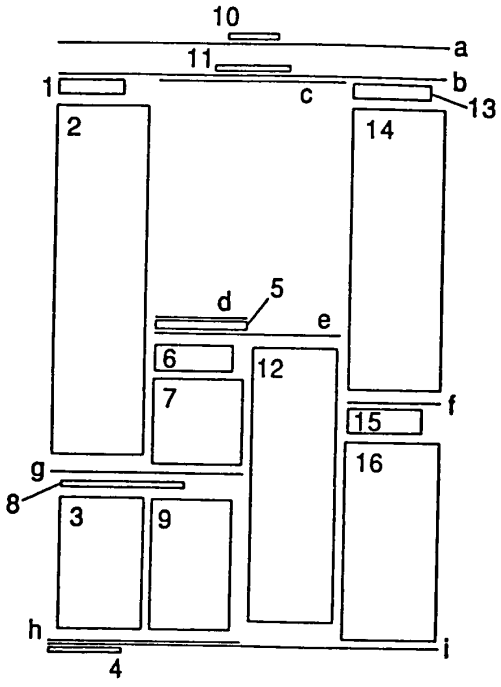
(a)



(b)



(c)



(d)

Fig. 3. Block extraction. A domument image is analyzed using a bottom-up approach. (a) Original document image. A document image is input by a scanner at a resolution of 300 dpi. (b) Segment extraction results. Segments represent the rectangular area circumscribing a text line or a word. (c) Text line extraction result. Neighboring segments are merged into a text line after column definition. (d) Block extraction result. Vertically adjacent text lines are combined into a block.

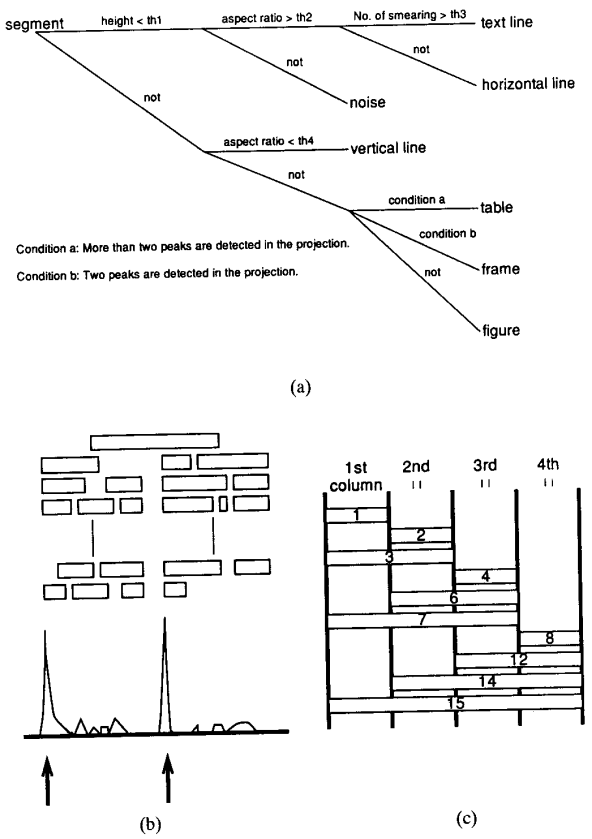


Fig. 4. Document analysis. (a) Classification of a segment. Each segment is temporarily classified by means of the physical properties of the segment. Final classification will be done later. (b) How to determine the column position. The left edges of segments contribute to the location of columns. (c) *Group number*, depending on column number. *Group number* will be introduced to describe how each segment is related to the column positions.

lines, vertical lines, and noise according to the physical properties of the segments. These physical properties of a segment are the size (width and height), aspect ratio (width/height), horizontal and vertical projection, and the number of smeared white runs [3]. Of these properties, the last one is a metric which represents the complexity of the segment.

Figure 4(a) roughly sketches the method of classifying segments, where **th1**, **th2**, **th3**, and **th4** are the given thresholds. Thresholds **th1**, **th2**, and **th4** are fixed, while threshold **th3** is defined by the segment size.

The number of smeared white runs is a metric of the horizontal white–black transition. If this metric is smaller than **th3** for a long horizontal segment, then the segment is classified as a *skewed* horizontal line; otherwise it is classified as a text line.

Frames are classified further into *text frames* and *figure frames* by examining their contents. If there are lots of text lines inside a frame, this frame is defined as a *text frame* representing a box. When figures are found inside the frame, it is a *figure frame*.

This classification is temporary, and a final classification is fixed after the merging processes. For example, noise segments are temporarily considered as text lines because they might be isolated periods, commas, or dots on the letters "i" and "j."

These classified segments play differing roles in the document analysis and understanding process. A horizontal line is used as a field separator in document understanding, and a vertical line is useful for defining column settings. A text line located below a figure (photograph), a figure frame, or a table is possibly a caption. A text frame emphasizes the independence of the text lines located within it.

3) Merging Process: The blank length between words is usually proportional to the height of the words on a document. The third subprocess makes use of this idea, merging neighboring segments which are defined as text lines if the blank space between them is smaller than a threshold which is made proportional to the words' height.

4) Text Line Extraction: In the last subprocess, columns are extracted by detecting their left edges as follows. The left coordinates of columns are defined by local maxima in a vertical projection profile, which is made by vertically adding up the left edges of those segments classified as text lines (see Fig. 4(b)). Naturally, the right edges of segments may also contribute to defining column positions, if text lines are adjusted properly at the right.

The skew of the page is corrected before the above vertical projection is obtained for the entire page.

After defining the columns, neighboring segments in the same column are merged into one text line. The result of text line extraction from Fig. 3(a) is shown in Fig. 3(c).

C. Block Extraction

In order to describe how each segment is related to the column positions, a *group number* is introduced for each segment, as shown in Fig. 4(c).

The *group number* (*g.n.*) is given by the following equation:

$$g.n. = \sum_{i=1}^{N} cp[i]^{*}2^{(i-1)}, \tag{1}$$

where

$cp[i] = 1$ (if the segment is in the ith column),
$cp[i] = 0$ (otherwise),
N is the number of columns.

Vertically adjacent text lines with the same group number are combined into a block. Here, vertical adjacency is determined by a threshold defined by the line interval.

A *Head* or *body* label is attached to each block according to the conditions described in subsection II-B. Numerals in Fig. 3(d) indicate the blocks generated from Fig. 3(a).

D. Generating a Geometric Structure Tree

Lastly, a geometric structure tree is generated, as described in subsection II-B. Each node of the tree represents a set of adjacent blocks with the same *group number*. Parent-daughter relationships between nodes are established by examining *group numbers* and their vertical locations.

Nodes not dominated by others are directly connected to the root node. These nodes are ordered by examining their *group numbers* and vertical locations so that a block to the left and on the top precedes the others.

Generation of a geometric structure for virtual field separators, as introduced in subsection IV-C, will be described later.

IV. Document Understanding

This section defines document understanding as a transformation from a geometric structure to a logical structure. A small number of the transformation rules are introduced here. Virtual field separator techniques are also proposed for universal transformations.

A. What Is Document Understanding?

It is true that the reading order, i.e., the sequence in which a document is read, is not completely fixed until the document's contents have been examined. However, it is also generally true that the actual reading order intended by authors often influences the layout, because the author wants the reading order to be defined easily before reading starts. For example, one can easily pick out the titles in a document since they are usually emphasized in one of several ways.

The reading order is inferred by means of the titles and other region separators. This observation leads to the following approach for document understanding. The reading order is derived from the logical structure. The authors define the transformation from a geometric structure to a logical structure as document understanding. On the other hand, the reverse transformation is not unique because a logical structure could correspond to a variety of geometric structures.

B. Basic Algorithm for Tree Transformation

This subsection presents an algorithm for the geometric to logical structure transformation. The algorithm is composed of four transformation rules that define the conditions under which an element in a node list is moved. These rules are illustrated in parts (a) through (d) of Fig. 5, where **H** indicates a *head* block, **B** indicates a *body* block, and **S** indicates that a block can be either *body* or *head*. In the tree, each node is sequentially numbered in the depth-first order. This is called depth-first indexing.

Four transformation rules are described below. Through this transformation processes, a node which becomes *NULL* is deleted.

Rule (a):

If

a node (say **A**) is a terminal node, and
the first element of node **A** is a *body*, and
the preceding node (say **B**) in the *depth-first indexing* is a terminal node,

then

remove the first element from node **A**, and
append it to the last element of node **B**.

Figure 5(a) illustrates the transformation process of this rule. The rule is based on the observation that a title has a single set of paragraphs as a daughter in the logical structure. Therefore, if the parent of a terminal node containing *bodies* has several daughters, then only one of them can be the true daughter of the parent. It is reasonable that the eldest daughter represents the text dominated by the parent and that the others should be merged to her.

Rule (b):

If

a node (say **A**) is a terminal node that is not connected to the root node, and
the preceding node (say **B**) in the *depth-first indexing* is a terminal node, and
the first element of node **A** is not *NULL*, and
last element of node **B** is a *head*,

then

remove the first element from node **A**, and
append it to the last element of node **B**.

Figure 5(b) illustrates this rule. Here, element **a1** is removed and appended to node **B**. This transformation is the same as that for rule (a). The difference is that the first element of node **A** does not need to be a *body* if the last element of node **B** is a *head*.

Rule (c):

If

a node (say **A**) contains a *head* block, and
it is not the first element of the node,

then

generate a younger sister node (say **D**), and
remove the *head-body* sequence that begins with that *head* block and ends with the last element of node **A**, with daughters of node **A**, if any, and
attach them to the younger sister node **D**.

Figure 5(c) illustrates the conversion process of this rule. When a node includes more than one *head-body* sequence, a new sister node is generated for each *head-body* sequence by applying this rule recursively. This rule is mainly for extracting chapters and sections headed by a subtitle.

Rule (d):

If

there is a *head* block sequence in a node, and
it is the first part of the node,

then

generate a daughter node, and
move the *body* sequence that follows the *head* sequence to the daughter node.

Figure 5(d) shows a case in which node **A** has a single *head* sequence and a single *body* sequence. In this case, the *body* sequence is separated from node **A** and moved to a new node **C**, which is a newly generated daughter of node **A**. By this rule, each node comes to have either *head* or *body* sequence. This rule is applied after rules (a), (b), and (c) have been completed.

The next step is an interpretation process, where a label is attached to each node. Here, the labels include *title*,

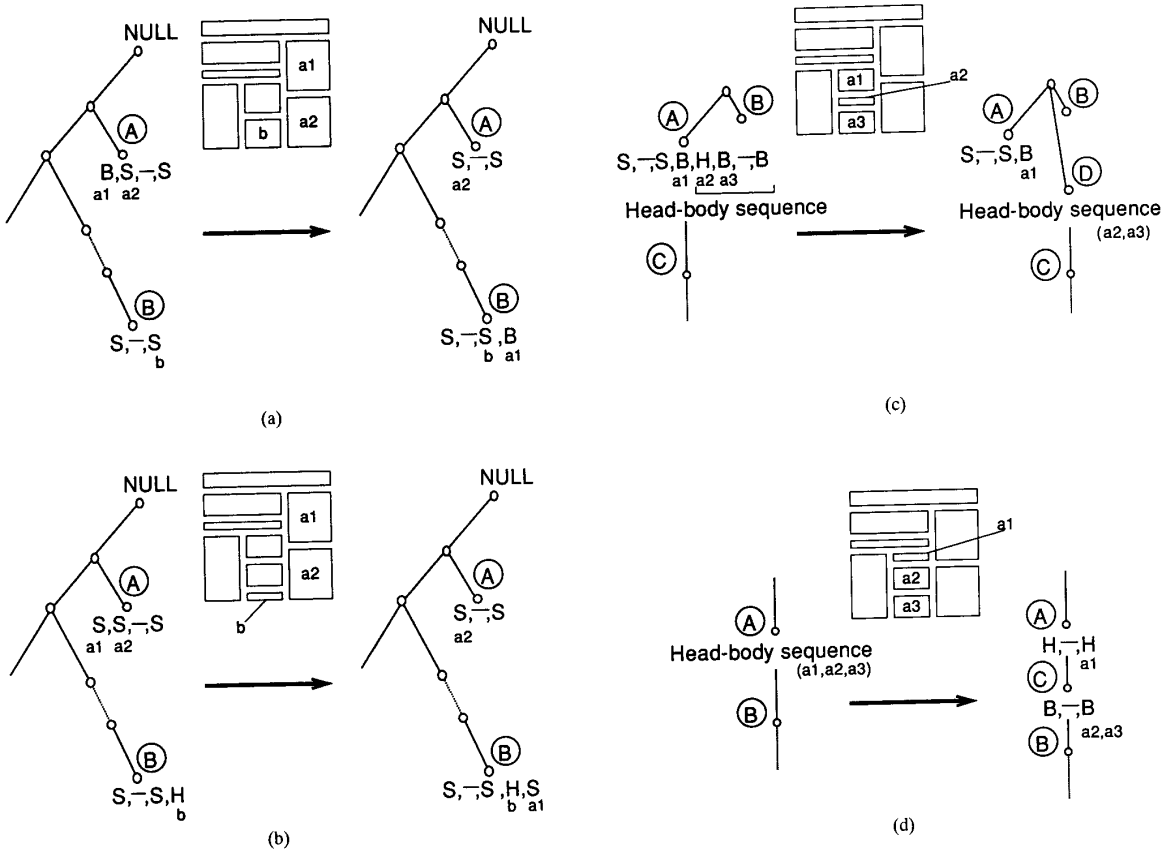


Fig. 5. Transforming a geometric structure into a logical structure using four rules. Transformation rules define the conditions for the movement of an element in a node list of a tree. In this illustration, **H** indicates a *head* block, **B** indicates a *body* block, and **S** indicates either. (a) Rule (a): This rule is based on the observation that each title has a single set of paragraphs. (b) Rule (b)—similar to rule (a). (c) Rule (c)—extracts chapters of sections for a subtitle. (d) Rule (d)—attaches a unique class (*head*/*body*) to each node.

abstract, *subtitle*, *paragraph*, *header*, *footer*, *page number*, and *caption*. If a daughter of the root node has children and she is a *head* sequence, she represents a *title*. If she has no children and she is a *head* sequence, one of the labels *header*, *footer*, *page number*, or *caption* is attached to her according to her location on the page. For example, a block which is centered and located at the bottom of a page is a *page number*. Any *head* blocks other than daughters of the root node are *subtitles*. *Body* blocks in terminal nodes are normally *paragraphs*. A *body* block which is the eldest and whose next sister is a *subtitle* represents an *abstract*. A *body* block with daughters also represents an *abstract*.

Figure 6 shows an example of the transformation process which generates a logical structure from the geometric structure of the document shown in Fig. 2.

C. Virtual Field Separators for a Universal Transformation

Field separators and rectangular text frames are good identification tokens for understanding document images. A field separator signals a break in the text lines, and explicitly distinguishes text lines located below it from those above it. A frame signals the independence of text lines within it.

The authors employ a virtual field separator technique which avoids the additional rules of special transformations for the field separators and frames to effectively introduce the information carried by them. This also helps the realization of a universal transformation.

In the virtual field separator technique, a *head* block with a *NULL* list is substituted for a field separator (see Fig. 7(a)). In Fig. 7, the numerals indicate that the block is a real one, while the letters indicate that the block is a virtual one; namely, it is a field separator. A field separator with a *head* label behaves as if it were a title block and it emphasizes a break in the text lines.

The upper and lower lines of a text frame are treated as virtual field separators. Nodes for these lines are connected to the root node so that text lines in the frame are independent of the others, as shown in Fig. 7(b). It should be noted that a node for the lower line does not have any blocks other than itself. A block located below the lower line and a block located above the upper line are combined in the same node. For example, **5B** and **2B** in Fig. 7(b) are combined in the same node.

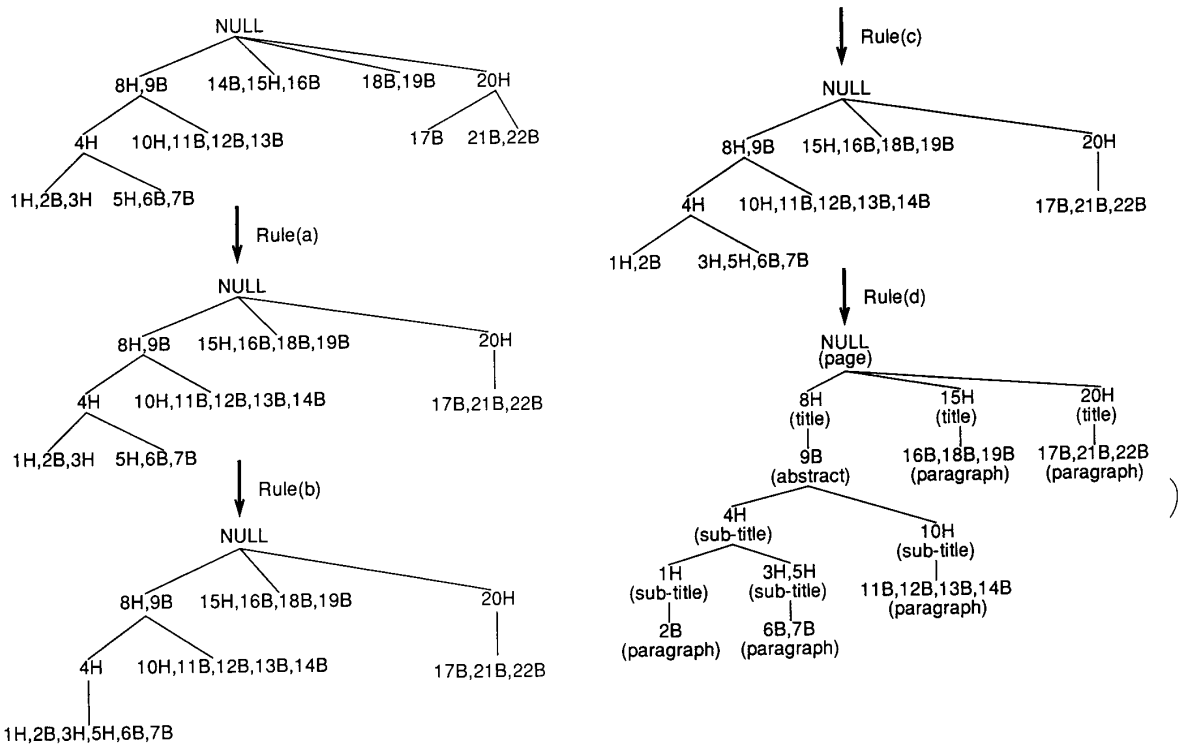


Fig. 6. Example of transformation process. The geometric structure tree of the document shown in Fig. 2 is transformed into a logical structure tree using four rules.

The virtual field separator technique is useful in treating photographs and figures, as well as their captions. In this case, a virtual frame circumscribing the figure or photograph and its caption is generated.

The virtual field separator is also defined in this virtual frame in order to distinguish text in the figure from the caption. Figure 7(c) shows the use of virtual field separators in this case. In this figure, blocks **a** and **c** are the upper and lower lines of the generated virtual frame, respectively. Block **b** is a virtual field separator for the caption.

The virtual field separator for footers is generated above them (see Fig. 7(d)), while the virtual field separator for headers is generated below them.

Virtual field separators are also used for the understanding of what we call a manual format, where title blocks are located to the left of text blocks, as shown in Fig. 7(e). In this format, each title and text pair is treated as being relevant. A virtual field separator is generated above this pair if two horizontally adjacent blocks are located in the same vertical position. Using such separators, the titles and corresponding texts are connected to each other in the logical structure tree.

In the last stage, the redundant field separators, if any, are deleted. A redundant field separator often appears when a real one happens to exist where a virtual one is generated.

This introduction of the virtual field separators does not require an increase in the number of transformation rules.

D. Geometric Structure of Virtual Field Separators

This subsection describes how a geometric structure of virtual field separators is generated. First, blocks representing headers, footers, and captions are detected. A *head* block which is located at the bottom of a page is regarded as a footer, and a *head* block at the top of a page is assumed to be a header. The existence of long horizontal lines will help the footer and header to be detected. A *head* block located below a figure is treated as a caption. Virtual field separators are generated in the way described in subsection IV-C.

At this stage, it is determined whether the page is laid out in the *manual format*. The condition for *manual format* is that a page consist of two columns and that each *head* block in the left-hand column correspond to a *body* block in the right-hand column in terms of vertical position. Virtual field separators are also generated for this *manual format*.

Virtual blocks are substituted for real and virtual field separators. Letters in Fig. 3(d) indicate several virtual blocks. Block **f** is a real field separator obtained from the original image, blocks **g** and **h** are virtual ones derived from a text frame, and blocks **c**, **d**, and **e** are also virtual ones generated for a figure and its caption. Blocks **a** and **b** are virtual ones prepared for headers, and block **i** is for a footer. Real field separators, which are located below

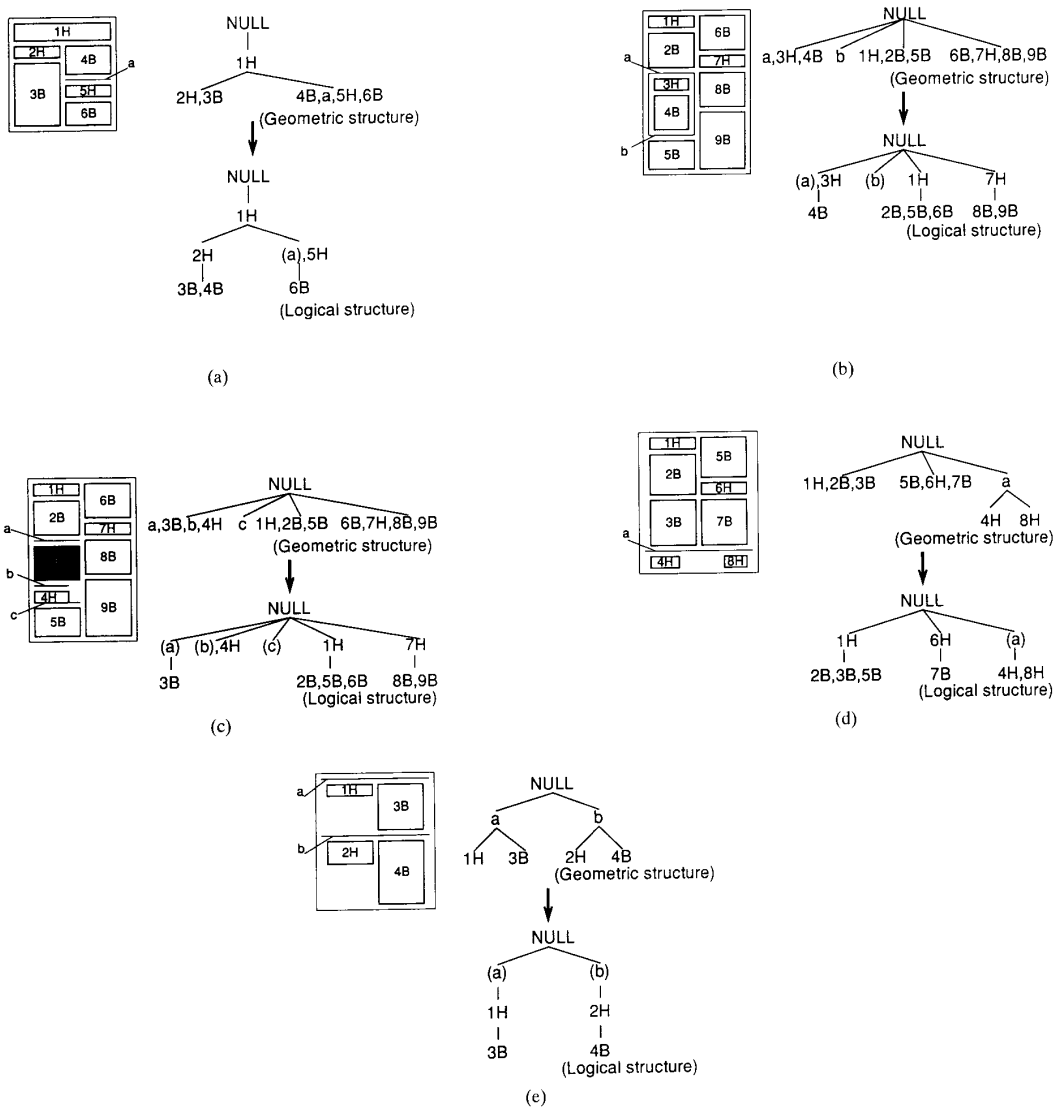


Fig. 7. Virtual field separator techniques for a universal transformation. The virtual field separator technique avoids the need for additional rules of special transformations for field separators and figures. This helps the realization of a universal transformation. Numerals in this illustration indicate real blocks, while letters indicate virtual blocks. (a) Field separator treated as a virtual block. A field separator signals a break in text lines. (b) Frame treated as two virtual field separators. A frame signals the independence of the text lines within it. (c) Virtual field separators generated for a figure and the caption. Text lines below a figure may be its caption. (d) Virtual field separator for footer. (e) Virtual field separators for *manual format*. Title blocks are located to the left of text blocks in a document laid out in a *manual format*.

blocks **10**, **11**, and **5** in the original image, are deleted because virtual ones have been generated where the real ones already exist.

Nodes for frames are connected to the root node so that they do not disturb the other node sequences. In fact, nodes for text frames are connected as eldest daughters of the root node, while nodes for figure frames are connected as youngest daughters.

Figure 8(a) shows the geometric structure tree constructed from Fig. 3(d).

E. Example of Tree Transformation

Figure 8(b) is the logical structure obtained from the geometric structure shown in Fig. 8(a) through the transformation process. Five articles are found in Fig. 8(b). The first article is in a text frame; **8** indicates the *title*, and **3** and **9** indicate *paragraphs*. Other articles are **1** (*title*), **2** (*paragraph*) and **6** (*title*), **7**, **12** (*paragraphs*) and **13** (*title*), **14** (*paragraph*) and **15** (*title*), **16** (*paragraph*). Two *headers* (**10**, **11**), a *footer* (**4**), and a *caption* (**5**) are also found.

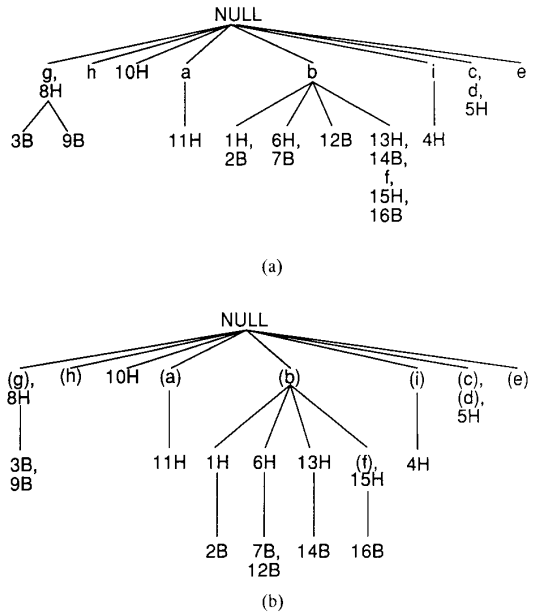


Fig. 8. Geometric and logical structure trees for the document in Fig. 3(a). Numerals in this illustration indicate real blocks, while letters indicate virtual blocks. (a) Geometric structure. (b) Logical structure. Logical structure tree is obtained from a geometric structure tree through the transformation process.

V. Character Segmentation/Recognition

Text lines are obtained in the document analysis and document understanding phase. The character segmentation phase extracts characters from these text lines. This procedure consists in extracting and recognizing characters.

The existence of touching characters makes it more difficult to design an effective character segmentation procedure because separating touching characters involves a number of ambiguities. Here, reduction of the number of such ambiguities is a crucial problem. This section presents a character segmentation/recognition method which is designed to be robust against touching characters.

A. Hierarchical Structure of a Text Line

A text line has a hierarchical structure; the text line consists of words, which in turn consist of characters.

Figure 9 shows an overview of the character segmentation hierarchy in the proposed system. First, a connected area in a text line image is defined as a component. Next, components above and below one another are combined. For example, in Fig. 9, the "i" is formed by combining two components. Components which are too small to be characters are regarded as noise, and are removed. Words are detected by examining the spaces between components.

B. Discriminating Touching Characters from a Single Character

Each component might be a single character or a pair of touching characters. Previous work has employed the concept of an aspect ratio to distinguish a component representing touching characters from a component representing a single character. However, the aspect ratio alone is not sufficient to separate proportional touching characters. For example, in Fig. 9, the single character "A" has a greater aspect ratio than "lt," which comprises two characters.

In the authors' approach, a component representing touching characters is found from the results of character recognition. First, a component is recognized as a single character, and then any component whose similarity obtained in character recognition is less than a fixed level (described in subsection V-F-4) in detail) is assumed to be a touching pair of characters. Character segmentation for touching characters will now be described.

C. The Authors' Approach to Touching Character Segmentation

One approach often adopted in the pattern recognition field is to divide the process into a number of phases and execute them sequentially by proposing several candidates. Ambiguities remain unresolved, however, in a phase where only one candidate is unsuccessfully sought as a solution.

There are generally two different approaches to solving this problem. One is an approach in which the ambiguities in each phase remain until the final phase. The other is an approach in which ambiguities in each phase are positively resolved in that phase. The former is used for problems where the final phase needs more emphasis than the other phases. The latter is suitable in cases where the phases are individually and sequentially managed. The authors' approach for touching character segmentation belongs to the former class rather than the latter.

D. Ambiguities in Touching Character Segmentation

The procedure for touching character segmentation mainly consists of three phases. First, candidates for break positions of touching characters are nominated by analyzing the touching character image. Second, the candidates are reduced by adopting recursive segmentation and recognition [16]. Last, a linguistic context confirms the break positions.

This procedure leads to several ambiguities. For example, each component image has several candidates nominated as break positions, and each segmented area may fit several alternative characters. (In Fig. 9, the "l" segment of the word "filter" fits an "l," a "t," an "I," an "i," and so on.). Another ambiguity is that an individual component may fit several possible touching characters. (In Fig. 9, the "lt" portion of the word "filter" fits the combination of an "l" and a "t," a single character "k.")

E. Resolving Ambiguities

Resolving ambiguities at each phase is very important in the segmentation of touching characters for two reasons. One is efficiency, meaning that a system which suppresses the number of possibilities to be checked can then segment touching characters at high speed. The other is power, meaning that a system which reduces ambiguities suffers fewer character recognition errors. For example, a Roman

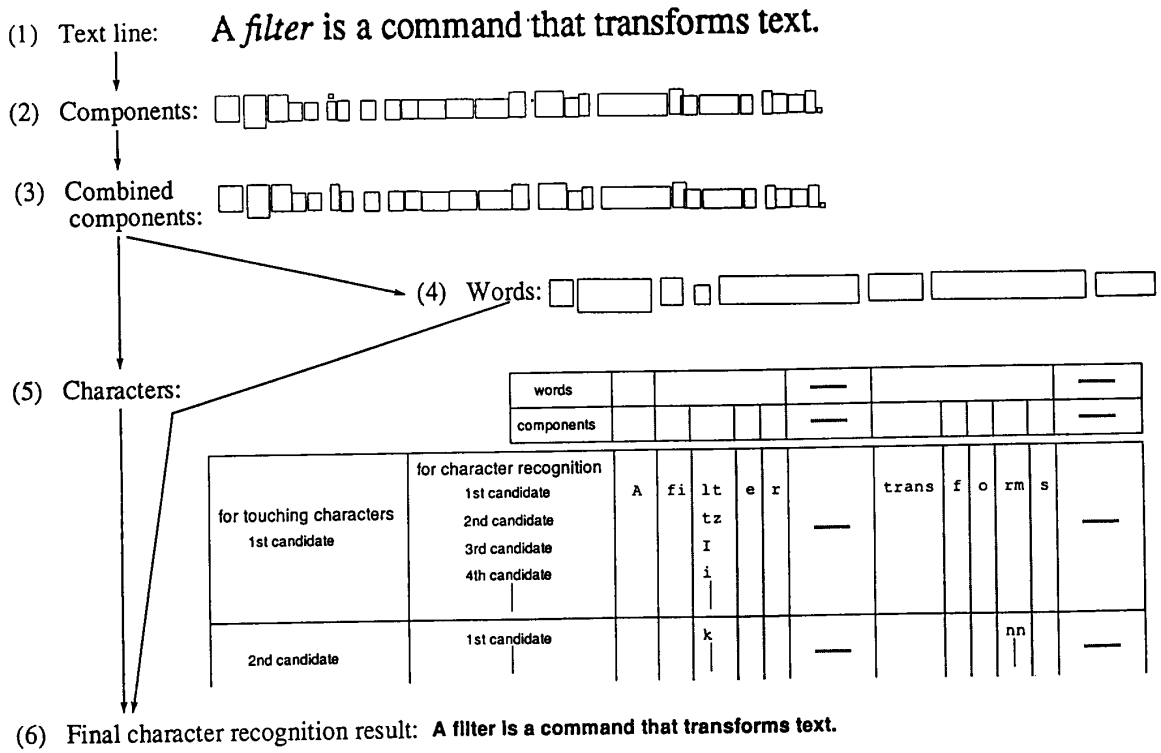


Fig. 9. Character segmentation overview. The character segmentation procedure extracts characters from a text line. Candidates for character segmentation and character recognition are deferred for a linguistic context procedure to confirm the final recognition result.

numeral "III" might be mistakenly changed into the English word "ill" if the candidate "i" remains a candidate for the first letter and the whole word is accepted by linguistic confirmation. As another example, numerals "50" and "200" might be mistakenly changed into the English words "SO" and "ZOO," respectively.

The authors employ heuristics of character composition as well as recognition results for omni-fonts to resolve ambiguities. The heuristics of character composition (e.g., a "W" looks like a combination of two "V"'s) avoids the need for extra recursive segmentation and recognition. Use of recognition results for omni-fonts reduces the number of candidates used in the linguistic confirmation process.

F. Procedure for Touching Character Segmentation

The authors propose a *break cost* as a new metric to nominate break positions in touching character images in the proposed procedure for touching character segmentation:

1) The *break cost* nominates candidates for break positions of touching characters.
2) It is examined whether each candidate is selected as a real break position or not; this is done by searching for an optimal path in a *search tree* where the recursive-segmentation-and-recognition approach [16] is adopted. Heuristics of character composition avoid the necessity for implementing a complete search.
3) Knowledge of the linguistic context is used to confirm the candidates. The number of linguistic context confirmations is reduced by employing recognition results for omni-fonts.

1) New Metric for Segmenting Touching Characters: Previous work has employed a vertical projection profile of the touching character image to find positions for separating touching characters. This profile is a function which maps the number of black pixels in each vertical column to the column's horizontal position. The authors introduce a new metric, the *break cost*, to evaluate the degree of contact.

The *break cost* is defined between each pair of neighboring columns. It is calculated by accumulating the number of black pixels vertically in the image obtained after an AND operation between neighboring columns. Figure 10 shows both the traditional vertical projection and the new projection for an input image. In this illustration, the new metric shows the ability to detect a prominent break between the left-hand and right-hand areas of the input image, while the vertical projection fails to do so.

This *break cost* consumes little processing time when the input image is represented by vertical run length.

If a typeface is *italic*, its *break cost* is calculated after the image is straightened, as shown in Fig. 11.

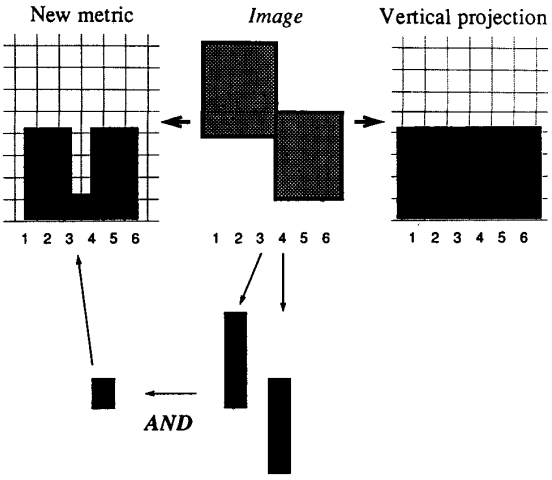


Fig. 10. *Break cost* for segmenting touching characters. *Break cost* evaluates the degree of contact. In this illustration, the new metric succeeds in exhibiting a prominent break between the left-hand and right-hand areas of the image, while traditional vertical projection fails to do so.

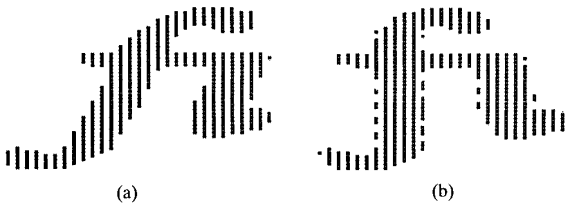


Fig. 11. Italic characters. (a) Italic character image. (b) Straightened italic characters. Italics are straightened before the touching character segmentation procedure is applied. The image will be returned to oblique when the character recognition procedure is applied.

2) Candidates for Break Positions: The authors assume that some variations in the *break cost* function can be found when two characters touch each other. The break position candidates are obtained by finding local minima in a smoothed *break cost* function. Several break position candidates occur as smooth local minima, while candidates are also found at both the start and end positions of touching character image. Figure 12(a) shows an example of touching characters. The *break cost* function for Fig. 12(a) is shown in Fig. 12(b), and the smoothed *break cost* function is shown in Fig. 12(c). The arrows indicate the break position candidates.

There are two kinds of candidates: one is more likely to be a real break position, and is called a preferred candidate, while the other is less likely. If the *break cost* distribution around a candidate shows a dominant sharp peak, then preference is given to this specific candidate. Preference is also given to candidates at the start and end positions of touching character image. In Fig. 12(c), thick arrows show the *preferred* candidates.

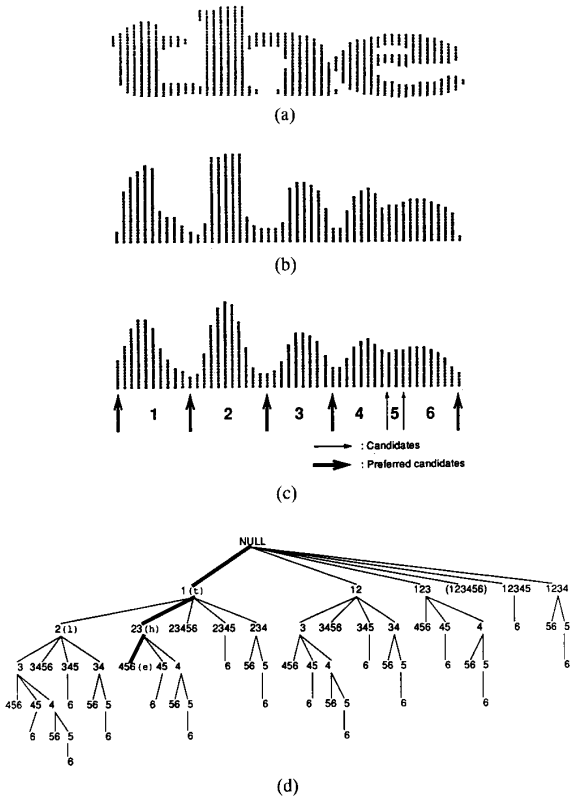


Fig. 12. Segmentation of touching characters. (a) Touching character image. (b) *Break cost*. Some changes are assumed in the *break cost* function when characters touch each other. (c) Smoothed *break cost*. The arrows indicate the candidates for break positions. The numbers indicate the areas segmented by the candidates. (d) Search tree. An optimal path representing a character segmentation result is searched for in depth-first order in the search tree through the nodes.

3) Search Tree: A search tree is introduced to represent a search order for the break position candidates. Figure 12(d) shows the search tree obtained from Fig. 12(c). In Fig. 12(c), the numbers indicate the areas of a touching character image segmented by the break position candidates.

This search tree is specified as follows:

1) Each node, except for the root node which has a *NULL* value, represents a combination of segmented areas which may correspond to a single character. For example, in Fig. 12(c), the subset {**2**, **3**} corresponds to a node which represents an "h."
2) Each node has a list of segmented areas, the first element of which is geometrically connected to the last element of the preceding node in a depth-first order. In other words, the daughters of a node represent succeeding characters. For example, the subset {**1**} has five daughters: {**2**}, {**2**, **3**}, {**2**, **3**, **4**, **5**, **6**}, {**2**, **3**, **4**, **5**}, and {**2**, **3**, **4**}.
3) The daughters of each node are sequentially ordered in the following ways:
   a) Among segmented areas ending at *preferred*

candidates, a smaller segmented area precedes the others. For example, the subset {**1**} precedes the subset {**1**, **2**} among the daughters of the root node.

b) Among adjacent segmented areas ending at *nonpreferred* candidates, a larger segmented area precedes the others. For example, the subset {**1**, **2**, **3**, **4**, **5**} precedes the subset {**1**, **2**, **3**, **4**,} among the daughters of the root node.

c) A segmented area ending at a *nonpreferred* candidate succeeds the smallest segmented area among the larger segmented areas ending at *preferred* candidates. For example, the subset {**1**, **2**, **3**, **4**, **5**} succeeds the subset {**1**, **2**, **3**, **4**, **5**, **6**} among the daughters of the root node.

This sequence is determined so that *preferred* candidates are sought in advance of *nonpreferred* candidates.

A path representing a character segmentation result is searched for in depth-first order in a search tree through the nodes. If a node is accepted as a segmented area through character recognition results, then its daughters are examined; otherwise, its sisters are checked. The criterion for character recognition acceptance is described in subsection V-F-4. The search terminates if the path arrives at a leaf node.

An example for this search is explained using Fig. 12. The subset {**1**} is recognized first, and character recognition accepts it as a "t." Consequently, its daughters are examined as follows. Subset {**2**} is accepted as an "l," but its daughter subsets {**3**}, {**3**, **4**, **5**, **6**}, {**3**, **4**, **5**}, and {**3**, **4**} are rejected. So, in this case, the path for subset {**2**} is ignored. As the next step, the sister of subset {**2**} is examined. Finally, subsets {**1**} for a "t," {**2**, **3**} for an "h," and {**4**, **5**, **6**} for an "e" are segmented from the touching characters.

4) Employing Recognition Results for Omni-fonts: A character pattern is recognized by the *multiple similarity method* [18], [19], which is designed to be insensitive to the varieties of omni-fonts. The *multiple similarity method* is a sophisticated pattern matching method which can be summarized as follows:

Given an input pattern $\boldsymbol{x} \in R^n$, the similarity value s_i for a certain category C_i is defined by

$$s_i = \sum_{j=1}^{p} (\lambda_{ij}/\lambda_{i1})(\boldsymbol{x}, \varphi_{ij})^2 / \|\boldsymbol{x}\|^2 \|\varphi_{ij}\|^2, \qquad (2)$$

where λ_{ij} and φ_{ij} are the jth eigenvalue and the jth eigenvector, respectively, obtained from a correlation matrix which is calculated from the training data belonging to C_i.

The *multiple similarity method* outputs several recognition candidates. The condition used to determine whether to reject characters is as follows:

$$\begin{aligned} \text{if } (s1 < \boldsymbol{th_1}) \quad & rejected \\ \text{else if } (s1 - s2 > \boldsymbol{th_2}) \quad & accepted \\ \text{else} \quad & conflict \end{aligned}$$

where similarities for the first and second candidates are denoted as $s1$ and $s2$, respectively, and both $\boldsymbol{th_1}$ and $\boldsymbol{th_2}$ are thresholds determined by statistical analysis for each character category.

Conflict suggests that both the first and second candidates are ambiguous. For example, an "l," an "I," a "1," and an "i" belong to this *conflict* category. However, despite the above rule, if the first candidate is an "l" but the second candidate is an "m," then this recognition result should be rejected.

For *conflict* characters, the images are further examined. For example, in the case of an "l," an "I," a "1," or an "i," the existence of a dot should be examined by measures such as the number of combined components described in subsection V-A. For the pair "c" and "e," the presence of a hole in the character should be examined.

Both *accepted* and *conflict* characters are regarded as being accepted. The difference between *accepted* and *conflict* is that, in employing a linguistic context, an *accepted* character fits the first candidate in recognition, while a *conflict* character fits several alternative candidates. This classification (*accepted/conflict/rejected*) enables the system to save time. Also, this classification can prevent the recognition results from being mistakenly changed by implementing a linguistic context, as described in subsection V-E. This approach makes the character segmentation method very powerful.

5) Heuristics of Character Composition: Figure 13 shows another example of touching characters where the character segmentation procedure fails. Two subsets, {**1**, **2**} for an "n" and {**3**, **4**, **5**, **6**} for an "n," are selected. If all possible combinations of subsets were to be examined, subsets {**1**} for an "r" and {**2**, **3**, **4**, **5**, **6**} for an "m" would also be selected.

Since examination of all the subset combinations would require exhaustive processing time, the authors employ heuristics of character composition to solve this problem.

As an example of such heuristics,

"m"	⟨−⟩	"r," "n"
"q"	⟨−⟩	"c," "j"
"k"	⟨−⟩	"l," "c"
"B"	⟨−⟩	"l," "3"
"H"	⟨−⟩	"I," "-," "I".
"mm"	⟨−⟩	"n," "u." "n"
"ck"	⟨−⟩	"d," "c"

etc.

With these kinds of heuristics, another combination of subsets can be generated. In Fig. 13, the combination "nn" builds a hypothesis of another possible combination, representing an "r" and an "m." The authors have prepared more than 30 examples of such combinations. This idea makes the character segmentation method efficient.

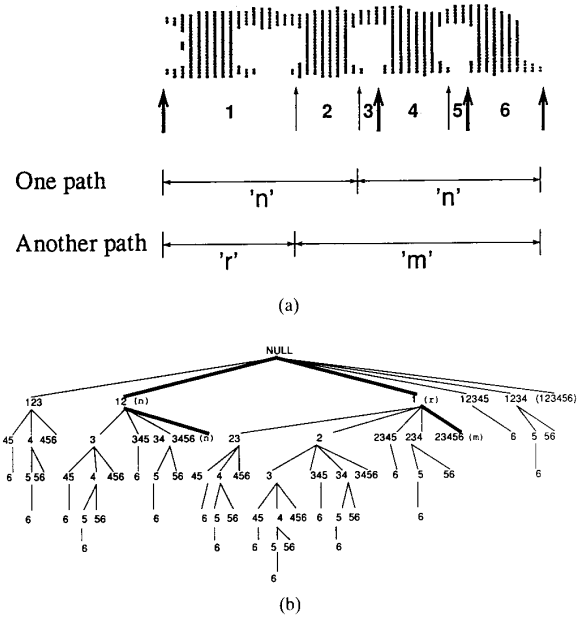


Fig. 13. Touching "r" and "m." Touching "n" and "n" may be identified from the touching characters where touching "r" and "m" should be identified. (a) Candidates for touching characters. (b) Search tree. Depth-first search stops once one path representing character segmentation result is found. Heuristics of character compositon generate another possible path. In this example, the combination "nn" builds a hypothesis of another combination, "rm."

6) Linguistic Context: All candidate combinations of touching characters are deferred for a linguistic context procedure to confirm the character segmentation result. In Fig. 9, the word "transforms" confirms that it is a combination of an "r" and "m" rather than an "n" and "n."

VI. Experimental Results

The methods proposed in this paper were implemented on a recognition board consisting of a RISC processor and memory. A scanner with a resolution of 300 dpi was directly connected to this recognition board. All procedures were realized in software alone [20].

Experiments on document analysis and document understanding were carried out on 106 documents taken from magazines, journals, newspapers, books, manuals, letters, and scientific papers. Some of the test documents with their various layouts are displayed in Fig. 14. Experiments on character segmentation were carried out on 32 of the 106 documents. These documents contained various fonts, and had many touching characters.

In experiments on document analysis and document understanding, there were 12 documents whose layouts were not correctly interpreted. This was attributed to three reasons. One was that the geometric structure was not correctly constructed because of errors in *segment* and/or *block* extraction in the document analysis process. Seven documents out of the 106 tested were not correctly interpreted for this reason. Another was because the proposed transformation rules did not cover all actual layouts. Four documents fell into this category. A document whose title or abstract was located in the middle of the text blocks, as in Fig. 15(a), came into this category. The last reason was that documents did not have geometrically and logically defined hierarchical structures. Figure 15(b) is a kind of table and does not have a hierarchical structure in the geometric and logical sense. These kinds of documents, however, are usually in the minority. One of the test documents was in this category.

Table 1 Experimental Results of Character Segmentation

Document No.	No. Characters	No. Touching Characters	No. Errors**	No. Errors in Character Segmentation
1	6742	4756	19	8
2	6334	4773	56	43
3	4846	3452	17	7
4	3972	2828	18	4
5	3000	2181	12	3
	100%	72%*	0.5%***	
		100%		0.4%****

*Seventy-two percent of characters touched each other in the test documents.

**Recognition errors were due to both character recognition errors and character segmentation errors.

***Recogniton accuracy was 99.5%.

****Character segmentation accuracy was 99.6%.

Table 1 shows the experimental results of character segmentation for five documents where 72% of characters touched each other on average. The average recognition error rate after a linguistic context confirmation was 0.5% for these five documents. Recognition errors were due to errors in both character recognition and character segmentation.

Character segmentation failed to separate individual characters from touching characters with an error rate of 0.4%. There were three reasons for this. One was that break position candidates were not correctly nominated because of complicated contact. Figure 16(a) shows an example of tangled touching, where a two-dimensional distribution analysis of the image is required. Errors in the second document in Table 1 were caused mainly by this problem. Another reason was that insufficient examples of heuristics of character composition had been prepared. Figure 16(b) shows an example where one combination of a "T," a "1," a "J," and an "R" was accepted before the real combination of a "T," a "U," and an "R" was examined. In this example, good heuristics of character composition, e.g., a "U" could be a "1" and a "J," would generate the real combination. The last reason was that a character was rejected as a consequence of errors in character recognition though the character was

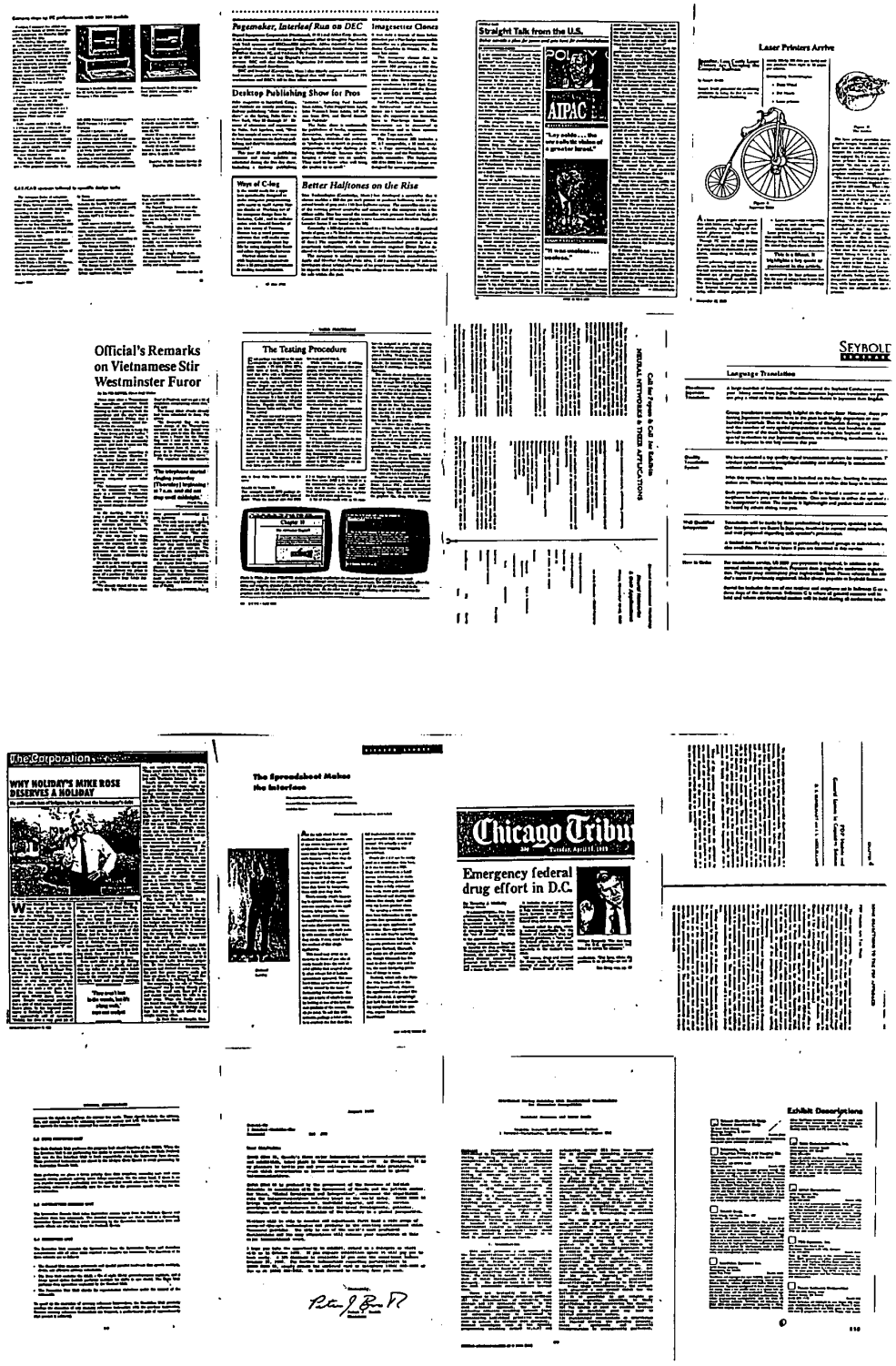
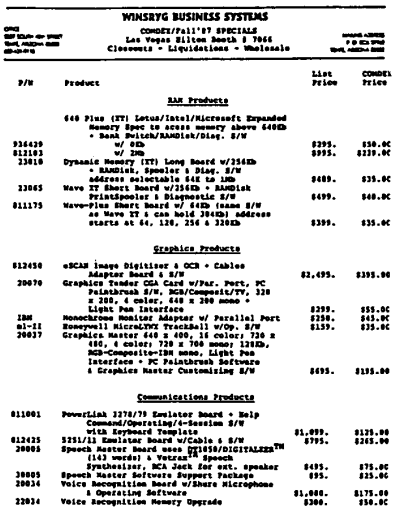


Fig. 14. Test samples. Experiments were carried out on documents with various layouts.

correctly segmented. Figure 16(c) shows an example where the character identifying an "a" was not accepted.

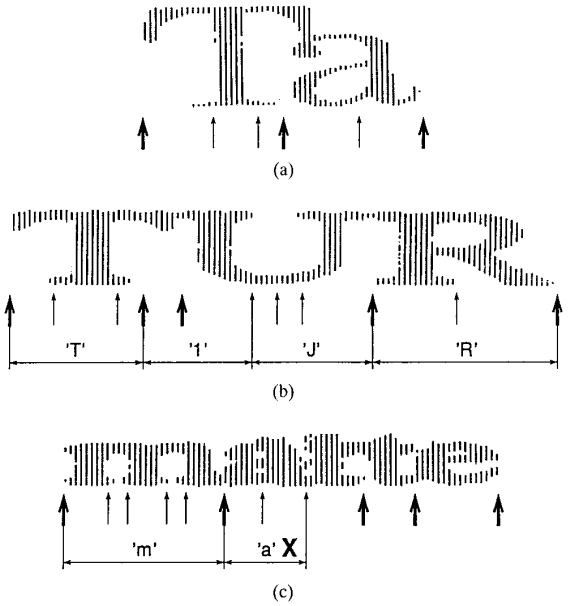
The new metric, the *break cost* for touching character segmentation, was also evaluated and compared with the traditional vertical projection. In the first document in Table 1, for example, the number of character segmentation errors

(a)



(b)

Fig. 15. Examples of incorrectly interpreted documents. Twelve documents were found which were not correctly interpreted out of 106 in total. (a) Document with an abstract located in the middle of the text. The proposed rules for document understanding did not cover this layout. (b) Document not having a hierarchical structure in a geometric sense.



(a)

(b)

(c)

Fig. 16. Problems in segmenting touching characters. (a) Tangled touching. Candidates for break positions were not correctly nominated because of the complicated contact. (b) Insufficient heuristics of character composition. Additional heuristics of character composition (that is, a "U" can be an "l" and a "J") would be required. (c) Rejected character. Correctly segmented character "a" was rejected in character recognition.

would be 23 if traditional vertical projection were employed to evaluate the degree of contact, while the number of errors was only 8 when the new metric was employed.

Lastly, overall recognition accuracy and processing speed are described. Recognition accuracy was on the average 99.7% for photocopies of original good-quality documents after linguistic context adaptation. The authors employed a lexicon with more than 30 000 entries for the linguistic context.

The processing speed was 60 to 120 characters per second depending on the quality of the input documents. Actual processing time for the sample documents shown in Fig. 3(a) was 70 seconds, including 20 seconds for scanning, 4 seconds for run-length coding, 5 seconds for generating the geometric structure tree, i.e., document analysis, 1 second for the structure transformation, i.e., document understanding, and 40 seconds for character segmentation/recognition on a total of about 4400 characters.

VII. Conclusion

This paper describes three major components of a complete text reading system. New approaches to document analysis, document understanding, and character segmentation/recognition have been introduced. For document analysis, this paper presents a method capable of robust behavior even for multicolumned documents with graphics and photographs. This approach gives the items on the page a hierarchy which models the relationships between characters, lines, columns, and the page. For document understanding, this paper reports an attempt to build a method of understanding document layouts without the assistance of the character recognition results, i.e., the meaning of the contents. It is shown that documents have an obvious hierarchical structure in their geometry which

can be represented by a tree. A small number of rules are introduced to transform the geometric structure into a logical structure which represents the semantics carried by the documents. The virtual field separator technique is employed to utilize the information carried by special constituents of documents such as field separators and frames, keeping the number of transformation rules small. For character segmentation/recognition, an efficient and powerful character segmentation method for touching characters has been presented. This approach employs heuristics of character composition as well as recognition results for omni-fonts to resolve ambiguities in segmenting touching characters. A new metric to evaluate the degree of touching is also introduced to identify each character in a group of touching characters.

Experimental results on a variety of documents have shown that the proposed methods are applicable to most of the document types commonly encountered in daily use, although there is still room for further refinement in the transformation rules for document understanding and resolution of ambiguities in segmenting touching characters.

REFERENCES

[1] S. Tsujimoto and H. Asada, "Understanding multi-articled documents," in *Proc. 10th Int. Conf. Pattern Recognition* (Atlantic City, NJ), 1990, pp. 551–556.

[2] S. Tsujimoto and H. Asada, "Resolving ambiguity in segmenting touching characters," in *Proc. 1st Int. Conf. Document Anal. and Recognition* (Saint Malo, France), 1991, pp. 701–709.

[3] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Comput. Graphics, and Image Processing*, vol. 20, pp. 375–390, 1982.

[4] J. Toyoda, Y. Noguchi, and Y. Nishimura, "Study of extracting japanese newspaper article," in *Proc. 6th Int. Conf. Pattern Recognition* (Munich, Germany), 1982, pp. 1113–1115.

[5] M. Okamoto and A. Miyazawa,"An experimental implementation of document recognition system for papers containing mathematical expressions," in *Pre-Proc. 1990 Syntactic & Structural Pattern Recognition* (Murray Hill, NJ), 1990, pp. 335–351.

[6] H. S. Baird and K. Thompson, "Reading chess," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 6, pp. 552–559, 1990.

[7] F. Esposito, D. Malerba,and G. Semeraro, "An experimental page layout recognition system for office document automatic classification: an integrated approach for inductive generation," in *Proc. 10th Int. Conf. Pattern Recognition* (Atlantic City, NJ), 1990, pp. 557–562.

[8] J. Higashino, H. Fujisawa, Y. Nakano, and M. Ejiri, "A knowledge-based segmentation method for document understanding," in *Proc. 8th Int. Conf. Pattern Recognition* (Paris, France), 1986, pp. 745–748.

[9] Y. Tsuji *et al.*, "Document recognition system with layout structure generator," in *Proc. IAPR Workshop on Machine Vision Applications* (Tokyo, Japan), 1990, pp. 479–482.

[10] Y. Nakano, H. Fujisawa, and O. Kunisaki, "A document understanding system incorporating character recognition," in *Proc. 8th Int. Conf. Pattern Recognition* (Paris, France), 1986, pp. 801–803.

[11] K. Inagaki, T. Kato, T. Hiroshima, and T. Sakai, "MACSYM: A hierarchical parallel image processing system for event-driven pattern understanding of documents," *Pattern Recognition*, vol. 17, no. 1, pp. 85–108, 1984.

[12] I. Masuda, N. Hagita, and T. Akiyama, "Approach to smart document reader system," in *Proc. 1985 Comput. Vision Pattern Recognition* (San Francisco, CA), 1985, pp. 550–557.

[13] H. S. Baird, "Anatomy of a page reader," in *Proc. IAPR Workshop on Machine Vision Applications* (Tokyo, Japan), 1990, pp. 483–486.

[14] S. N. Srihari and G. W. Zack, "Document image analysis," in *Proc. 8th Int. Conf. Pattern Recognition* (Paris, France), 1986, pp. 434–436.

[15] S. Kahan, T. Pavlidis, and H. S. Baird, "On the recognition of printed characters of any font and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 2, pp. 274–288, 1987.

[16] R. G. Casey and G. Nagy, "Recursive segmentation and classification of composite character patterns," in *Proc. 6th Int. Conf. Pattern Recognition* (Munich, Germany), 1982, pp. 1023–1026.

[17] R. Kooi and W. C. Lin, "An on-line minicomputer-based system for reading printed text aloud," *IEEE Trans. Syst., Man, Cybern.*, vol. 8, no. 1, pp. 57–62, 1978.

[18] T. Iijima, H. Genchi, and K. Mori, "A theory of character recognition by pattern matching method," in *Proc. 1st Int. Joint Conf. Pattern Recognition* (Washington, DC), 1973, pp. 50–57.

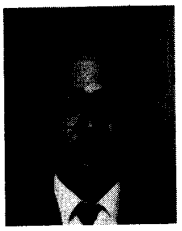
[19] E. Oja, *Subspace Methods of Pattern Recognition*. Research Studies Press, 1983.

[20] S. Tsujimoto and H. Asada, "Document image processing for accurate and high speed text reading," in *Pre-Proc. 1990 Syntactic & Structural Pattern Recognition* (Murray Hill, NJ), 1990, p. 501.

Shuichi Tsujimoto was born in Osaka, Japan, on February 3, 1961. He received the B.E. and M.Sc. degrees form Osaka University, Osaka, Japan, in 1984 and 1986, respectively, for his work on data compression by means of the spline approximation.

He joined the Toshiba Corporation, Japan, in 1986 and since then has been at the Information Systems Laboratory, Research and Development Center, Kawasaki, Japan. His current research interests include document image processing, layout understanding, text processing, character segmentation, and character recogniton for a state-of-the-art text reader.

Mr. Tsujimoto is a member of the Institute of Electronics, Information and Communication Engineers of Japan.



Haruo Asada was born in Osaka, Japan, in 1948. He received a B.E. degree in 1970 and an M.E. degree in 1972, both in mathematical engineering and instrumentation physics from the University of Tokyo, Tokyo, Japan.

Since 1972 he has been the the Toshiba Corporation, Japan, where he has worked on character recognition, speech recognition, pattern recognition theory, shape analysis, robot vision, and image processing hardware. During the academic year 1983–1984 he was a visiting scientist at the MIT Artificial Intelligence Laboratory, Cambridge, MA, where he worked with the robotiscs group on 2-D shape description. He is now a senior research scientist at the Information Systems Laboratory, Toshiba Research and Development Center. His current research interests focus on document image processing, pattern recognition theory, and shape recognition.