A Document Analysis Method based on a Consistent Structural Model of Document Elements and Pages

Masaharu Ozaki

Fuji Xerox Palo Alto Lab.,
3400 Hillview ave, PAHV-403,
Palo Alto, CA 94304, U.S.A.
Masaharu_Ozaki.PAL@FujiXerox.Xerox.com

Yuusuke Ishida

Fuji Xerox System Technology Research Lab.,
9-14 Naka-cho 4-chome,
Atsugi-shi, Kanagawa, 243 Japan
yuusuke@rst.fujixerox.co.jp

Abstract

A document analysis method based on a structural model of document elements and pages is described. Layout analysis for generic document elements and logical structure analysis for specific documents are integrated in a consistent way. Each category of document element is defined as a "class". Each definition of class consists of a definition type, classes of subordinate elements with logical labels and geometric constraints among them. Classes of specific pages can be defined as well as classes of elements. Class definitions form a network whose nodes correspond to element classes and whose links correspond to element/subordinate relations. The recognition process starts from the most primitive element class, and traverses the network to find elements satisfying geometric constraints in the definition. It progresses sequentially from lower element classes to higher, until all definitions have been examined. A prototype system has been implemented. Experimental results are consistent with results obtained from procedure-oriented systems, and they show that the prototype can produce a set of logically labeled elements using classes of specific document pages.

1. Introduction

Document recognition is a complex task of reconstructing structure and contents from scanned digital images. Early works focused on image processing technology which segments document images into physical regions, such as text blocks and graphics[1][2].

A document is composed of several types of document elements. Elements, in turn, are composed of subordinate elements. Hence, a document is represented in a hierarchical structure. Generic categories of elements, such as textblocks, tables, and figures, are laid out using typographic rules. These rules represent mutual understandings between writers (compositors) and readers. Even in documents which have relatively fixed layout, like scientific journal papers, we can see a similar regularity. In such documents, subordinate elements which have logical roles, such as title, authors, abstract, and body text, are laid out in a pre-determined way. Such regularity is also a mutual understanding between publishers and readers of the technical journal. From this point of view, there is no essential difference between layout structure and logical structure. The only difference is over how wide of a range these rules are understood.

Such knowledge about the structure of documents to be recognized has been used in the recognition systems, but it is implicitly hidden in their algorithms. It is difficult to apply them to other document categories. Therefore, model-based recognition techniques should be applied in order to separate descriptions of document structure from the recognition engine. Some techniques of model-based document recognition have been described [3][4][5].

In this paper, we will propose a document analysis technique which uses a consistent structural model of document elements and pages. First, we describe a model descriptive scheme, and illustrate some examples of model descriptions. After showing a recognition control mechanism, we show some experimental results.

2. Structural Model of Document Elements and Pages

2.1 Descriptive Scheme

Document elements are separated into categories, such as textblock, table, figure, etc., which we call "classes". Specific pages, such as IEEE Transaction cover pages and Computer Software cover pages, are also individually defined as classes. Each class is allowed to have multiple structural definitions. For example, table can be composed of some solid horizontal lines and text blocks which can be seen in English publications, or composed of a lattice of solid lines and text blocks which can be seen in Japanese publications. Even though these have the same contents, the subordinate elements are different. In order to handle these cases, we allow classes to have multiple structural definitions.

Each structural definition consists of a structural type, subordinate element descriptions, and constraint descriptions among the subordinates. Syntax of class definition is shown in Fig. 2. We classify structural types according to element/subordinate relationships into four: *specialized*, *homogeneous*, *master-slave* and *heterogeneous*. These are described in detail in the next section. Each subordinate element definition is described with a label, a designation of element class, and a unitary predicate for the element's attributes. The label is used for designation of logical role of the element, like "document title" or "figure caption", or just a tag, like "graphics main body". The meaning of the label depends on the logical level of the element being described. Constraint

```
element class
    :: = (structural definition)+ ;
structural definition
    :: =      structural type
        +     (subordinate element description)+
        +     (constraints description)+ ;
subordinate element description
    :: =      label + element class + predicate ;
constraints description
    :: =      (label)+
        +     predicate ;
[( )+ means one or more occurrence.]
```

Fig. 1 syntax for class definition

descriptions are described with a predicate of geometric relationship, or a combination thereof. The predicates are not necessarily required to represent absolute positions, but relative positions among elements.

Class definitions form a network in which nodes correspond to classes, and links correspond to element/subordinate relationships. Fig. 2 shows the network which is formed by generic element classes used for experiments described in a later section. Some recursive definitions are found in the graphics and textline classes. The graphics class is basically defined as *specialized* type from a single connected-component, but it may include many kinds of elements: horizontal-line, vertical-line or textblock. Recursive definitions enable us to define complex elements like this flexibly.

2.2 Classification of Structural Types

(1) *Specialized*

The *specialized* structural type represents one-to-one relationships in which an element is specialized as an instance of superior class when specified constraints are satisfied. For example, a graphics can be specialized from connected-component which satisfies the specified constraints for its size. This structural type essentially does not imply an element/subordinate relationship, but it can be treated in the same manner as other structural types during recognition process.

(2) *Homogeneous*

The *homogeneous* structural type represents categories of elements which consist of an arbitrary number of subordinates that belong to an identical class. For example, a textblock consists of an arbitrary number of textlines. This relationship is often seen in basic document elements. We restrict predicates for representing geometric constraints to two major ones: "vertically-arranged" and "horizontally-arranged".

(3) *Master-Slave*

The structural type *master-slave* represents categories of elements which consist of one subordinate element (*master*), and an arbitrary number of subordinates (*slave*) which belong to an identical class. Geometric constraints are defined between the *master* subordinate and one of the *slave* ones, and defined among *slave* subordinates.

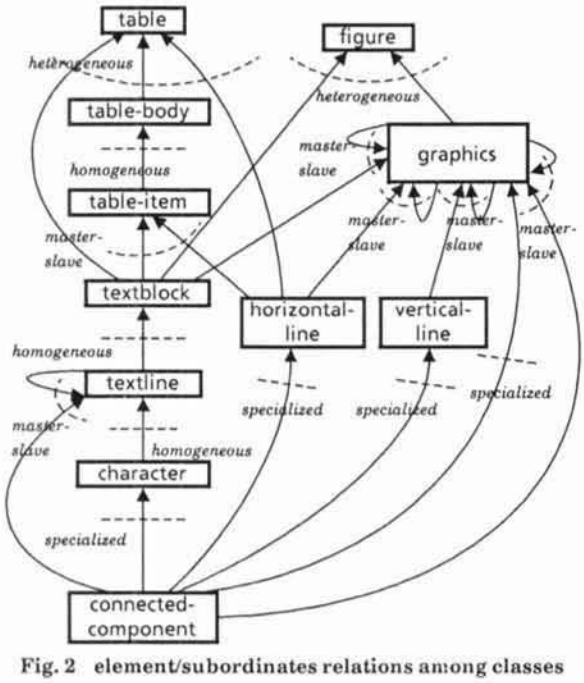

Fig. 2 element/subordinates relations among classes

(4) *Heterogeneous*

The *heterogeneous* structural type represents categories of elements which are composed of independent subordinates that belong to basically different classes. In these types of definitions, one of two appearance types, *required* or *optional*, can be designated to each subordinate description. Each geometric constraint is usually defined between two of the spatially adjacent subordinate elements. Regarding a subordinate element as a node and a constraint between two of them as a link, this definition forms a connected graph.

3. Examples of Model Descriptions

(1) **Table**

The structure of an ordinary table is illustrated in Fig. 3(a). Definitions of necessary classes are shown in Fig. 3(b). At the top class, table is defined as *heterogeneous* with subordinate a table-body, a horizontal-line, and two textblocks. Three geometric constraints are defined: (1) between the textblock labeled "caption" and the table-body, (2) the table-body and the horizontal-line labeled "rule", (3) the horizontal-line and the textblock labeled "additional-explanation".

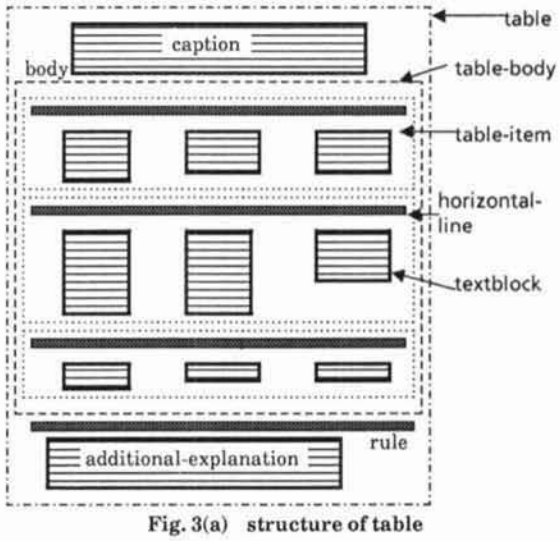

Fig. 3(a) structure of table

```
table (heterogeneous)
    :: = caption:          textblock
         body:             table-body
         rule:             horizontal-line
         additional-
         explanation:      textblock

table-body (homogeneous)
    :: = items:            table-item

table-item (master-slave)
    :: = rule:             horizontal-line
         item-
         contents:         textblock

textblock (homogeneous)
    :: = lines:            textline

textline (homogeneous)
    :: = characters:       character

character (specialized)
    :: = primitive:        connected-component

horizontal-line (specialized)
    :: = primitive:        connected-component
```

Fig. 3(b) class definitions for describing tables

Though the subordinate element labeled "additional explanation" is designated as *optional*, others are designated as *required*. Class table-body is defined as *homogeneous* with an arbitrary number of subordinates table-items. Table-item is defined as *master-slave* with one horizontal-line and an arbitrary number of textblocks below the line. The other classes can be defined in similar ways.

(2) Cover Page of Computer Software

An example of class definition of a specific page, a cover page of "Computer Software" is illustrated in Fig. 4. An instance of the cover page is shown in Fig. 6. This class is defined with *heterogeneous* type. Elements which compose the cover page are illustrated as nodes, and geometric constraints between elements are illustrated as links in the figure. Each element is defined with textline, textblock, and horizontal-line, and corresponding labels "title", "author" "abstract", "footnote rule", etc. Geometric constraints are defined with simple predicates for relative relations between bounding rectangles: above-below (vertically overlapped), left-right (horizontally overlapped) and left-aligned

4. Control of Recognition Process

The recognition process is bottom-up. It starts from the most primitive element class, connected-component, and progresses towards complex element classes. It goes through all links in the network made by element/subordinates relationships defined in the classes until evaluation of all structural definitions is completed. It traverses the network of document element classes shown in Fig. 2. Evaluation of each structural definition is performed by the following steps: (1) collect all instances of designated subordinate element classes (2) find sets of instances which satisfy constraints described in the definition. (3) make new element data with sets of subordinates thus found.

Evaluation order of structural definitions is basically determined by the following steps:

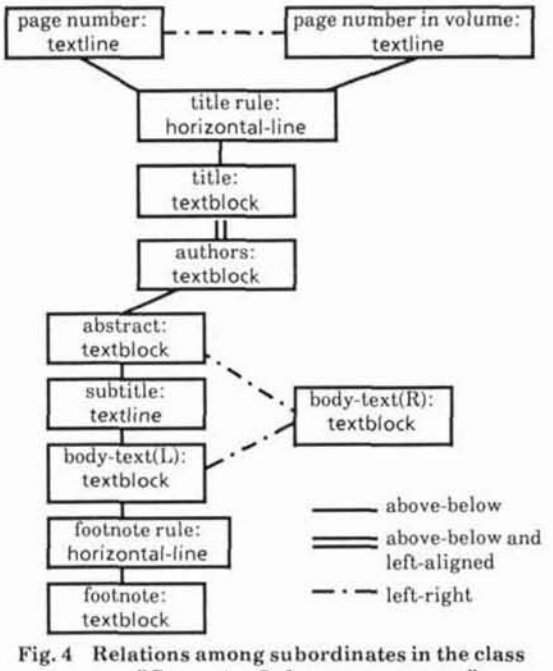

Fig. 4 Relations among subordinates in the class "Computer Software cover page"

(1) When there is an element class whose structural definitions have been completely evaluated, we call it "finished". Before the process starts, only the class connected-component is set as "finished".

(2) All structural definitions whose subordinate element classes' status are all "finished", or recursive definitions whose subordinate element classes' except recursive one's status are "finished" can be evaluated on the next step.

(3) When all structural definitions are evaluated, the process is over.

5. Experimental Result

We implemented a prototype in CommonLisp. It receives rectangular areas of connected components extracted from a scanned binary image at 300dpi, and outputs recognized element and page structures. In addition to element classes shown in Fig. 2, we also defined five classes for cover pages of scientific journal papers (including both English and Japanese ones).

(1) Layout Analysis

To see how well the system works on the layout structure analysis, we performed an experiment with the document element definitions shown in Fig. 2. The input document image and final result are shown in Fig. 5. This test data includes some textblocks, two figures, and one table. The halftone area located at the bottom left has been removed in the preprocess. The system produces the same result as traditional procedure-oriented document image segmentation methods.

(2) Specific Page Labeling

Adding five page classes on experiment (1), we performed an experiment for recognizing specific page structure. The input document image and final result are shown in Fig. 6. The result shows that the system determines that the input document is a cover page of Computer Software, and that each element is properly labeled. The other four page classes failed to match the input data.

6. Conclusion and Discussions

We have proposed a document analysis method using a consistent structural model of document elements and pages in a hierarchical manner. It can be easily adapted to the various kinds of documents by modifying or adding classes to the system, but some enhancements of this technique are required.

In the current implementation, all classes are treated equally. But the existence of some classes depends on the existence of superior classes. For example, table-items become meaningful when they are recognized as subordinates of table-body. To avoid that situation, we have to introduce a descriptive scheme for local class definition within higher level classes, and the recognition mechanism that requires that if recognition of a higher class is not possible, subordinate element recognition should be abandoned.

The current implementation works well if the input document is laid out under regular typographic rules. But these rules are often broken. These exceptions of rules should be described in each specific page structure. Moreover, the implemented bottom-up process is time-consuming, because the system tries to verify all possibilities of elements or pages described in the system. In order to make this system effective and robust to the variety of layout, it should be combined with a top-down analysis. The effectiveness of top-down ana-

lysis in document recognition has been described in [6][7]. We believe that we can introduce a new recognition process using a model described in the same manner as in the current prototype.

Acknowledgement

We would like to acknowledge Dr. Larry Spitz of Xerox PARC for reading this paper and giving us significant suggestions.

References

[1] Wong, K., et al., "Document Analysis System", *IBM Journal of Research and Development*, 26(6), 1982

[2] Fletcher, A.L. and Kasturi,R., "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images", *IEEE Trans on PAMI*, Vol.10, No.6, 1988

[3] Higashino, J. et al., "A Knowledge-based Segmentation Method for Document Understanding", *in Proc. of 8th ICPR*, Paris. France, 1986

[4] Kise, K. et al., "Model-based Understanding of Document Images", *in Proc. of MVA'90*, Tokyo, Japan, 1990

[5] Yamashita, A. et al., "A Model Based Layout Understanding Method for the Document Recognition System", *in Proc. of ICDAR '91*, St. Malo, France, 1991

[6] Ingold, I. and Armangil, D., "A Top-down Document Analysis Method for Logical Structure Recognition", *in Proc. of ICDAR '91*, St. Malo, France, 1991

[7] Spitz, L., "Style Directed Document Recognition", *in Proc. of ICDAR '91*, St. Malo, France, 1991

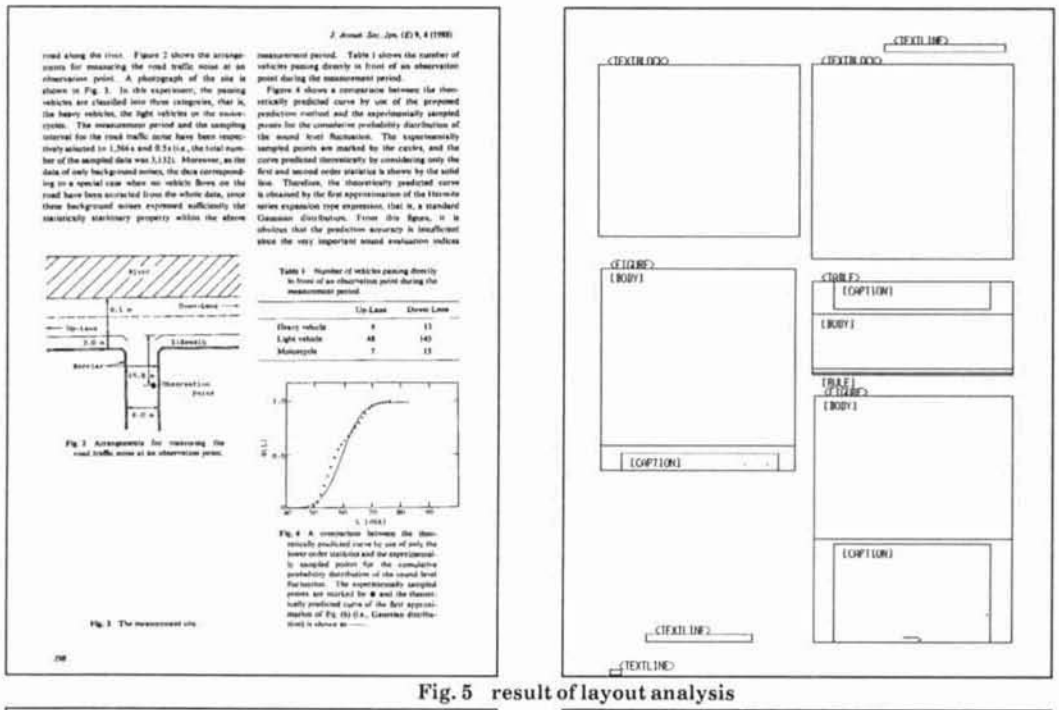

Fig. 5 result of layout analysis

Fig. 6 result of page labeling