Handbook on Optical Character Recognition and Document Image Analysis, pp. 000–000
Eds. P. S. P. Wang and H. Bunke

CHAPTER XX

The Representation of Document Structure: a Generic Object-Process Analysis

DOV DORI
Faculty of Industrial Engineering and Management, Technion, Haifa, Israel

DAVID DOERMANN, CHRISTIAN SHIN
Document Processing Group, University of Maryland, College Park, MD, USA

ROBERT HARALICK
Intelligent Systems Laboratory, University of Washington, Seattle, WA, USA

IHSIN PHILLIPS
Seattle University, Seattle, WA, USA

MITCHELL BUCHMAN
University of Maryland, Baltimore County, MD, USA

DAVID ROSS
RAF Technologies, Inc., Readmond, WA, USA

Abstract

Document understanding has attained a level of maturity that requires migration from ad-hoc experimental systems, each of which employs its own set of assumptions and terms, into a solid, standard frame of reference, with generic definitions that are agreed upon by the document understanding community.

The logical structure of a document conveys semantic information that is beyond the document's character string contents. To capture this additional semantics, document understanding must relate the document's physical layout to its logical structure. This work provides a formal definition of the logical structure of text-intensive documents. A generic framework using a hierarchy of textons is described for the interpretation of any text-intensive document logical structure. The recursive definition of textons provides a powerful and flexible tool that is not restricted by the size or complexity of the document. Frames are analogously used as recursive constructs for the physical structure description.

To facilitate the reverse engineering process which is required to derive the logical structure, we describe DAFS— a Document Attribute Format Specification and demonstrate how our framework can serve as a conceptual framework for enhancements of DAFS.

Keywords: document understanding, document layout analysis, physical structure, logical structure, document format standards.

1. Introduction

The field of document image understanding encompasses the technology required to allow the information contained in paper documents to be accessed electronically. Although the task may seem easily definable, the general expressibility of a document suggests that document image understanding must involve more than simply recognizing a string of characters on a page and putting them into the format of a word processing system.

Historically, Optical Character Recognition (OCR) has been more intensively researched. Consequently, it has attained a considerable level of maturity. Document page layout analysis constitutes a subject of intensive research and it, too, has reached a certain level of maturity. Physical layout analysis, combined with OCR, provides for complete reproduction of documents.

Logical document structure is a hierarchy that conveys the semantics of the document. The same logical document structure can be formatted in a variety of physical layouts by changing such variables as page and font size, spacings between paragraphs and between sections, number of columns, etc. Yet, the semantics of the document remains unaltered. Logical structure analysis determines the document's semantic structure and provides data appropriate for information retrieval involving more than string matching. For example, one would like to query all the abstracts of all papers in a database which have some keyword combination in a title or section heading and were written within a certain time period. To do this, the resulting data structure(s) should be precisely defined and agreed upon by the OCR and document understanding community.

The analysis of a document image involves both the physical decomposition of the page and the derivation of the logical or semantic meaning of the set of salient fields or regions defined. For example, if we are given a newspaper page, the physical analysis involves extraction of blocks of text, graphics and half-tones as well as the identification of attributes such as font size and style. The structural analysis, on the other hand, may involve using the layout clues to identify headlines, locate bylines, group paragraphs from different columns which belong to the same article, or associate a picture with the article which references it and the photographer who took it. In general, the problem involves the extraction and use of attributes and structural relationships in the document to label document components within the contextual rules dictated by the document class or type (memo, letter, journal article, newspaper, etc.).

Document understanding is complicated by the fact that relevant information can be expressed in a non-textual medium, such as mathematics, drawings and graphics. Hence, understanding art line drawings, engineering line drawings, perspective projections, graphs, and special kinds of documents like complex mathematical formulae and music scores, are all part of the document image understanding task.

The fact that hardcopy documents convey a great deal of semantic information via the *logical structure* expressed as the two-dimensional arrangement of the text and non-text elements on the page, suggests that the structural information is an essential part of understanding. Although there are many ways to store and display the physical appearance of the same document, the logical structure should remain the same, because it reflects semantics that is part of the author's intention, but beyond the ASCII stream of characters. Determining the logical structure of a document, therefore, constitutes an important aspect of document image understanding.

In this chapter, we will address the problem of providing a standard, formal framework for the management of physical and logical descriptive information extracted from document images, and address issues of intermediate representation which arise during the analysis process. Although the techniques used in the analysis of document images differs widely from those used in the creation process, it is useful to examine the fundamental representations set forth by the more developed document processing community.

Many standards exist for representing the logical structure of a document. In publishing applications, documents are "encoded" via standards such as SGML in preparation for the actual printing process. In document understanding and page decomposition, however, we perform "reverse encoding" seeking to reverse-engineer the meaning from an image of the printed page. The greatest difference between encoding for document creation and reverse encoding of document images is that during the reverse encoding process, there may be varying levels of uncertainty in the interpretation of aspects within the document. In the document creation process, this ambiguity is not present, since the document is usually encoded by the same person who created the representation of the document. A data format for document reverse encoding must have a mechanism for representing these ambiguities.

In document image reverse encoding, it is important to move back and forth between the interpretation of the physical structure of the document and the semantic structure of the document. Although they rely heavily on one another and may share common aspects, they are still two different ways of perceiving the structure of a document. The fact that typically there is not a one-to-one mapping between the physical and logical structure complicates this requirement.

While document creation proceeds in a serial manner, document image decomposition usually traces through a document hierarchically rather than serially. This is a result of the way reverse engineering processes are usually applied to documents. An example of this is that discrimination between text and non-text regions is often performed for an entire document before any character recognition is performed.

Throughout this process, the ultimate goal is to use physical attributes to obtain a consistent and valid interpretation of the semantic attributes. For example, the text-blocks should be semantically ordered by the "reading order" (sequencing) of the text units. Further, each text unit should be assigned a semantic label. In a business letter, the sender's address, receiver's address, date, opening salutation, body, closing, and signature can generally be inferred by their relative locations on the page. Likewise, in a technical article, the title, author(s), abstract, keywords, sections, displayed equations, tables, graphs, illustrations, footnotes, page numbers, reference list, and other logical components can be deduced by their locations and/or sequencing, as well as the fonts, styles and sizes of the characters that make them up. The resulting recognized text strings are formatted such that their two-dimensional layout, deduced from the 2-D layout analysis, are recorded along with the text itself.

The resulting complex data structure, if constructed correctly, captures the entire semantics of the original document. However, it is in a much more condensed form, providing for both data compression and noise removal. This data structure enables one to retrieve information through querying and to reproduce the original page document with practically no noise.

In Section 2 of this chapter we provide an overview of the state of the art in both geometric and logical interpretation. In Section 3, we propose a working framework for the logical structure of text-intensive documents. A key definition is that of a *texton*,[a]which provides for recursion and a quantitative definition of document complexity. Being a complex system, analysis of document structure and layout requires a sound methodology. We employ the object-process analysis (OPA) methodology (Dori et al., 1995; Dori, 1995) and object-process diagrams (OPDs), which are the graphic tool of OPA, to express both the structure and behavior of a document analysis system within a coherent, unified frame of reference.

Finally, in Section 4, we describe a new and powerful document attribute format specification—DAFS, which provides mechanisms for representing and maintaining both physical and logical information during the reverse encoding process. We also show how the logical framework from Section 3 can be implemented directly using DAFS.

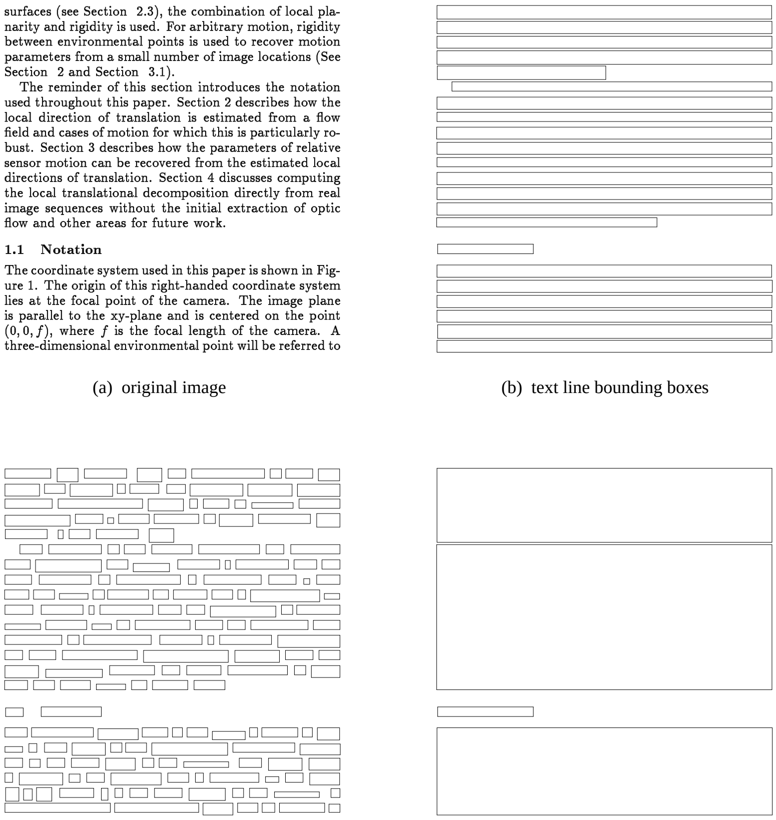
2. Literature survey

Papers covering all areas of document image analysis can be found in the First and Second International Conference on Document Analysis and Recognition [ICDAR] (1991 and 1993); the 1992, 1993, 1994 and 1995 Annual Symposia on Document Analysis and Information Retrieval, sponsored by the University of Nevada, Las Vegas; and the International Conferences on Pattern Recognition [ICPR]. The brevity of the space for this survey necessitates that many papers from these conferences are not mentioned. Surveys of document image analysis and document image understanding can be found in (Casey and Nagy, 1991 and Tang et. al. 1991).

2.1. *Geometric Layout Analysis*

A geometric page layout of a document image page is a hierarchical specification of the geometry of the different kinds of maximal homogeneous regions. A region is homogeneous if all its area is of one type, e.g., a character, a line of characters, a text paragraph, or a figure. Formally, a *geometric page layout* $\Phi = (\mathcal{R}, \mathcal{S})$, where

[a]Not to be confused with the earlier use of the term by Julesz in connection with visual texture perception (Juesz and Bergen, 1983).



surfaces (see Section 2.3), the combination of local planarity and rigidity is used. For arbitrary motion, rigidity between environmental points is used to recover motion parameters from a small number of image locations (See Section 2 and Section 3.1).

The reminder of this section introduces the notation used throughout this paper. Section 2 describes how the local direction of translation is estimated from a flow field and cases of motion for which this is particularly robust. Section 3 describes how the parameters of relative sensor motion can be recovered from the estimated local directions of translation. Section 4 discusses computing the local translational decomposition directly from real image sequences without the initial extraction of optic flow and other areas for future work.

1.1 Notation

The coordinate system used in this paper is shown in Figure 1. The origin of this right-handed coordinate system lies at the focal point of the camera. The image plane is parallel to the xy-plane and is centered on the point $(0, 0, f)$, where f is the focal length of the camera. A three-dimensional environmental point will be referred to

(a) original image

(b) text line bounding boxes

(c) word bounding boxes

(d) text block bounding boxes

Figure 1: *Example of Document Image Decomposition: (a) a document image written in English, (b) text line bounding boxes, (c) word bounding boxes, (d) text block bounding boxes*

$\mathcal{R}$ is a set of regions and $\mathcal{S}$ is a labeled hierarchical spatial relation on the set $\mathcal{R}$ of regions. A *region* $R = (T, \theta)$, where T defines the type of the region and θ is the parameter vector of values for the region. The parameter vector may also include uncertainty for any of the parameters. Uncertainty can be a parameter standard deviation, or a tolerance interval, represented as a probability pair. For the entire parameter vector, the uncertainty can be specified by a covariance matrix. Figure 1 illustrates the character, line, and paragraph hierarchy for a small text sample.

Many of the algorithms for determining geometric layout employ the operations of mathematical morphology. Although the original algorithm developers generally did not describe their algorithms in terms of mathematical morphology and were probably not even aware that their algorithms had anything to do with mathematical morphology, our description will utilize the operations of mathematical morphology. This will allow us to be brief and precise.

Early work on page segmentation was done by Wahl et al. (1982), using a technique called the constrained run length smoothing algorithm. It essentially consists of a morphologically closing of the document image with a horizontal structuring element of specified length (they used 300) intersected with a morphological closing of the document image with a vertical structuring element of specified length (they used 500). The intersection is then morphologically closed with a horizontal structuring element of specified length (they used 30). The bounding rectangles of the connected components of the resulting image constitute the block segments. Features of the blocks include the area of the connected component of the block, the number of black pixels in the block on the original document image, the mean horizontal black run lengths of the original image within the blocks, and the height and width of the bounding rectangle of the block. Text areas are then classified into *text, horizontal solid black lines, graphic and half-tone images*, and *vertical solid black lines*. No measure of performance is given.

Nagy and Seth (1984), and Nagy et al. (1986) employ an X–Y tree as the representation of a page layout. The root node of an X–Y tree is the bounding rectangle of the full page. Each node in the tree represents a rectangle in the page. The children of a node are obtained by subdividing the rectangle of the parent node either horizontally or vertically, with horizontal and vertical cuts being alternately employed in successive levels in the tree. Hao et al. (1993) describe a variation on this technique.

Fisher et al. (1990) sample a 300dpi document image by a factor of 4 and use the run length smoothing algorithm. They then compute the connected components of the run length smoothed image. The connected components and their bounding boxes constitute the blocks of the geometric page layout. They extract connected component features such as component height, width, aspect ratio, density, perimeter, and area for classifying each block as *text* or *non-text*.

Lebourgeois et al. (1992) sample the document image by a factor of 8 vertically and 3 horizontally. Each pixel in the sampled image corresponds to an 8×3 window in the original image. If any pixel in the 8×3 window of the original image is a binary 1, then the sampled image has a binary 1 in the corresponding pixel position. The sampled image is then dilated by a horizontal structuring element to effectively smear adjacent characters into one another. Each connected component is then characterized by its bounding rectangle and the mean horizontal length of the black runs. Connected components having a vertical height within given bounds and mean horizontal run length within given bounds are then labeled as text. Lines outside the given bounds are labeled as non-text lines. Components assigned as text regions are then vertically merged into larger blocks using rules taking into account alignment. Blocks are also subdivided to separate their horizontal peninsulas. No measure of performance was given but it was indicated that the method needs improvement.

Bloomberg (1991) uses morphological operations on the document image at various resolutions to identify font style for each word. Class labels include *bold, italic*, and *normal*. The method employs a small vertical dilation followed by a close-open sequence to remove noise followed by a hit and miss transform to identify seed points of characters in the italic class or bold class. The words which are in italic or bold can then be delineated by conditionally dilating the seed with a pre-calculated word segmentation mask. No accuracy performance results are given.

Saitoh and Pavlidis (1992) proceed by sampling 8 vertical and 4 horizontal lines and then extracting the connected components. They then classify each component into *text, text or noise, diagram or table, half-tone image, horizontal separator*, or *vertical separator*, using block attributes such as block height, height to width ratio, and connectivity features of the line adjacency graph, and whether there are vertical or horizontal rulings. Page rotation skew is estimated from a least squares line fit to the the center points of blobs belonging to the same block. Blocks are subdivided based on the vertical distance between lines in a block, and the height of the lines in a block. The technique was tried on 52 Japanese documents and 21 English documents. No quantitative measure of performance was given.

Hinds et al. (1990) sample a 300dpi document image by a factor of 4 and compute from it a burst image. A burst image can be obtained from the distance transform or the erosion transform of the document image using a two-pixel vertical (horizontal) structuring element for portrait (landscape) mode images. The burst image selects only the column (row) relative maximum pixels of the erosion transform. They then compute the Hough transform of the burst image, incrementing each Hough bin by the value of the pixel in the burst image, providing its value is less than 25. The rotation skew of the image is then determined by searching the Hough parameter space for that bin having the largest accumulated value. Its angle is the rotation skew angle. They tested the technique on 13 document images, and correctly determined the rotation angle on all the images. The inter-line spacing on one document image was not correctly determined and the landscape/portrait mode was incorrectly determined on five document images.

Pavlidis and Zhou (1991) determine the geometric page layout by analyzing the white areas of a page by computing the vertical projection and looking for long white intervals from the projections. Then the column intervals are converted into column blocks, merging small blocks into larger blocks. Blocks are clustered according to their alignments and the rotation angle is estimated for each cluster. The column blocks are then outlined. Finally, each block is labeled as text or non-text using features such as the ratio of the mean length of black intervals to the mean length of white intervals, the number of black intervals over a certain length, and the total number of intervals. No performance results are given.

Baird (1992) discusses a computational geometry technique for geometric page layout by finding the maximal rectangles covering the white areas of the page. The rectangular regions not covered by the maximal sized white rectangles then constitute regions which can then be classified as text or non-text.

Amamoto et al. (1993) determine the geometric page layout by operating on the white space of the sampled document image. They open the white space of the sampled document image with a long horizontal structuring element and open it again with a long vertical structuring element. The union of these two openings constitutes the white space of the blocks. The blocks are then extracted from this white space. They decide that a block is a text block if the length of the longest black run length in the vertical and horizontal directions is smaller than a given threshold. A decision is made whether the writing is horizontal or vertical based on the number N_H of blocks whose width is greater than twice their height and the number N_V of blocks whose height is greater than twice their width. If $N_H > N_V$, then the decision is horizontal writing, else it is classified as vertical writing. Each block is then assigned a class label from the set *text, figure, image, table*, and *separation line*. No performance results are given.

O'Gorman (1992) discusses what he calls the docstrum technique for determining geometric page layout. This technique involves computing the k-nearest neighbors for each of the black connected components of the page. Each pair of nearest neighbors has an associated distance and angle. By clustering the components using the distance and angle features, the geometric regions of a page layout can be determined.

Ishitani (1993) determines the rotation skew angle as that direction in which the variance of the complexity of the white-black transitions is greatest. Specifically, a set of lines is defined for each angle. The difference between successive angles is 0.01 degree. Each line's complexity is then measured, where complexity is defined as the number of white-to-black transitions along the line. The variance of the white-to-black transition counts is then determined. The angle which maximizes this variance is the estimated rotation skew angle. It is reported that this measure does not have difficulties with document pages which have large areas of non-text. The method was tested on 40 300dpi document images taken from magazines, newspapers, manuals and scientific journals. It was reported that the rotation angle was measured to within an accuracy of 0.12 degrees.

Chen and Haralick (1994) and Chen, Haralick, and Phillips (1995) use an algorithm based on the recursively computed morphological opening and closing transforms. First the image is subsampled to a 100dpi resolution. The recursive closing transform is computed, a histogram is generated from the transform image, and a threshold is determined using an regression tree function of the histogram values, where the regression tree was previously determined off-line. The closing transform is then thresholded to produce a closing of the subsampled image that fills inter-character gaps. Then, on this closed image, a recursive opening transform is computed, and, in a similar manner, a threshold is determined from the histogram of the values of the opening transform. The opening transform is then thresholded to produce an opened image in which the character ascenders and descenders have been removed. Then the connected components of the opened image are computed and line fits to each connected component are obtained. Using the estimated orientation of each of the fitted lines and the residual fitting error, the skew angle of the document page image is obtained from a robust estimation of the fitted line orientations. To determine the performance of the algorithm they used the UW-I document image database (Phillips, Chen, and Haralick, 1993) having 1147 distinct document images each associated with a ground truth skew angle. This set of images was then rotated at eleven rotation angles: $0°, \pm 1°, \pm 2°, \pm 3°, \pm 4°$, and $\pm 5°$, to constitute a set of $1147 \times 11 = 12617$ images. The absolute difference between the estimated skew angle and the ground truth skew angle was less than $0.5°$ for more than 99% of the images and less than $1°$ for virtually all of the images.

Hirayama (1993) develops a technique for determining the geometric layout structure of a document which begins by merging character strings into text groups. Border lines of blocks are determined by linking edges of text groups. Then blocks which are over-segmented are merged and a projection profile method is applied to the resulting blocks to differentiate text areas from figure areas. Hirayama reports that on a data set of 61 pages of Japanese technical papers and magazines 93.3% of the text areas and 93.2% of the figure areas were correctly detected.

Ittner and Baird (1993) determine a geometric layout by doing skew and shear angle corrections, partitioning the page into blocks of text, inferring the text line orientation within each block, partitioning each block into text lines, isolating symbols within each text line, and finally merging the symbols into words. The rotation skew angle is determined by taking the projections of the centers of the connected components of the black pixels on the page at a given angle. The angle is iteratively updated to optimize the alignment without having to compute the projection over each possible angle. After rotating the image, shear is corrected by a similar technique. They report an accuracy to within 3 minutes of arc, indicating that the method fails on perhaps one in 1000 images. Blocks are determined by the white space covering technique of Baird (1992). They report that on 100 English document image pages from 13 publishers and 22 styles, 94% of the layouts were correctly determined. The orientation of text lines in a block is determined from the minimum spanning tree of the connected components of the black pixels. The mode of the histogram of the directions of the edges in the minimum spanning tree is the orientation of the text lines in a block. Symbols in a text line are determined by taking the projection in an orthogonal direction to the text line. The projection profile is checked for a dominant frequency and the segmentation into characters is done from the projection profile with the knowledge of the dominant frequency. To determine the words in a text line, they determine a scalable word-space threshold for each text block separately. Then each text line is independently segmented to distinguish between the inter-character spacing and the inter-word spacing.

Ankindele and Belaid (1993) determine a geometric page layout that permits blocks to be polygonal as well as rectangular. They determine the elongated white spaces in the document image and then determine the intersections of these white spaces. Points of intersection are candidate vertices for a polygonal block. The polygonal blocks are then extracted from the geometry of the intersection points. No performance results are given.

2.2. *Logical Layout Analysis*

The emphasis in the works on logical layout analysis is on developing processes (algorithms) to carry out various tasks related to logical segmentation.

Tsujimoto and Asada (1990) assume that each block of the geometric page layout contains exactly one logical class. They organize the geometric page layout as a tree. Each new article in a document such as a newspaper begins with a headline which is in the head block. They find the paragraphs which belong to the head block by rules relating to the order of the geometric page layout tree and are able to assign logical structure labels of *title, abstract, sub-title, paragraph, header, footer, page number*, and *caption*. They worked on 106 document images and correctly determined the logical structure for 94 document images.

Fisher (1991), an extension of Fisher (1990), describes a rule based system to identify the geometrical and logical structure of document images. Ingold and Armangil (1991) describe a formal top-down method for determining the logical structure. Each document class has a formal description that includes composition rules and presentation rules. They have utilized the technique on legal documents.

Chenevoy and Belaid (1991) use a blackboard system for a top-down method of logical structure analysis of a document image. The system is defined in a Lisp formalism and has a hypothesis management component using probabilities.

Kreich et al. (1991) describe a knowledge-based method for determining the logical structure of a document image. To obtain the blocks they search for the largest text blocks because these are the most characteristic elements in the document layout. The search consists of grouping together the connected components which are close enough to each other. Once text blocks are determined, lines are found within each of the text blocks and words within the lines. The determination of document layout structure is based on interpreting documents and their parts as instances of hierarchically organized classes. They have defined over 300 classes for a document image and its parts. No performance results are given.

Derrien-Peden (1991) describes a frame-based system for the determination of structure in a scientific and technical document image. The basis of this system is a macro-typographical analysis. The idea is that in scientific and technical documents, changes of character size or thickness of type, white separating spaces, indentation, etc., are used to make visual searching for information easier. The technique therefore searches for such typographical indications in the document and recovers its logical organization without any interpretation of its semantic content. The first step is the determination of the geometric page layout, keeping a *part of* relationship between blocks. The logical structure determination removes running headers and footnotes and searches for the text reading order. Text blocks are then compared to logical models of classes and each text block is then assigned a class. No performance results are given.

Yamashita et al. (1991) use a model-based method. Character strings, lines, and half-tone images are extracted from the document image. Vertical and horizontal field separators (long white areas or black lines) are detected based on the extracted elements; then appropriate labels are assigned to character strings by a relaxation method. Label classes include *header, title, author, affiliation, abstract, body, page number, column, footnote, block* and *figure*. The technique was applied to 77 front pages of Japanese patent applications. They reported that the logical structure for 59 of these documents was determined perfectly.

Dengel (1993) discusses a technique for automatically determining the logical structure of business letters. He reports that on a test set of 100 letters, the recipient and the letter body could be correctly determined. Saitoh et al. (1993) determine logical layout with text block labels of *body, header, footer*, and *caption*. They tested the technique on 393 document images of mainly Japanese and some English documents. To characterize performance they measured the average number of times per document image an operator has to correct the results of the automatically produced layout. They report that on the average 2.17 times per image, areas not suitable for output have to be discarded; 0.01 times per image mis-classified areas have to be correctly labeled; and 1.09 times per image, a text area has to be reset. With respect to text ordering they report that it required moving connections 0.47 times per image, on the average, making new connections .11 times per image, and re-assigning type of text 0.36 times per image.

3. Logical and Physical Document Description

In order to describe a document's physical and logical characteristics consistently, it is advantageous to first distinguish between the document's content and its structure.

3.1. *Document Content vs. Document Structure*

The *content* of a document is the information contained in the document, which is not bound to a particular representation format. At the lowest level, this information may include a stream of characters, and these characters make up successively higher-order objects built on top of each other such as words, sentences, paragraphs, etc., up to the complete document level.

The *physical structure* of a document is how the document's content is laid out on the physical medium. The same content can be organized in a variety of ways and therefore can have many physical layouts, which stem from different values of the attributes of the physical components (point size, line spacing, page size, etc.). The medium is traditionally paper, but may be any visual host, such as a computer screen or a photographic film, which emulates the layout on the page. Bearing this extension in mind, we refer to paper documents as the classical representatives.

The *logical structure* of a document's contents are how its content is organized, prior to the enforcement of a particular physical structure. A text-intensive document, for example, typically consists of sentences, words and characters, and possibly higher-order constructs such as sections and chapters for an article, or address block, body and signature block for a correspondence.

It is essential to note that the **same** content can be viewed with respect to both the physical structure and the logical structure. A major goal of this chapter is to describe the two types of structure and how they can interact.

3.2. *Generic Document Structures*

Text-intensive documents can be classified into many types including books (text books, edited books...) technical papers, correspondence (business letters, memos, ...), newspaper articles, and conference proceedings. When attempting to describe the structural relationships between document components, it is essential to provide a level of abstraction, so as not to be caught up in the terminology associated with a specific type of document. To this end, we define terms that are generic at both the logical and physical levels. The *generic document structure* terminology allows us to define type-independent relationships among document components.

Following this rationale, we distinguish between the generic (both logical and physical) document structure from *instantiated generic document structure.* An instantiation of a generic structure allows us to begin discussing a particular class of document, and the relationships between known entities. Such a distinction is in accordance with the basic concepts of object-oriented analysis (OOA), where objects are instances of their respective classes.

Terms such as column, line and symbol in the context of the physical structure, or chapter, section, and subsection in the context of logical structure, are instances of the generic terms we define below. Some of them, such as the physical term "line", and the logical term "sentence", are applicable to a large variety of documents, while others, such as the logical term "session", are document type specific.

Finally, a *specific structure* (physical or logical) refers to the structure of a single given document, and describes the structure of that instance of the document.

3.3. *Realizing Document Structures*

The progression from generic to specific instances of a document is fairly straightforward, but the ability to reverse the process can be an important component of any analysis system. At the first level, the object class "Document" is instantiated to its type, both physical and logical, according to the document content. The structure of each instance of the generic document is thus dependent on the document's type.

By making such distinctions, systems will be able to prune the generic document tree to narrow the analysis to a given document type. For example, there is no use talking about a document's physical component called "volume" or the logical component "chapter" when the document under consideration is of type "Business Letter". Such distinctions can be made at an early point.

A second level of instantiation involves instantiating the object class of a particular document type, such as "Proceedings", to its specific structure, such as "Proceedings of Document Analysis Systems '94 Conference" (Dengel and Spitz, 1994).

We do not attempt to provide a full classification of all the text-intensive document types, let alone the particular logical or physical structure of each type. In the remainder of this section, however, we do provide a set of definitions and tools that enable this task to be done consistently and coherently, along with a few examples.

3.4. *Generic Physical Structure*

Following are the definitions related to the generic physical structure.

Frame – an area within a page of a document, which may consist of a collection of (lower level) frames and/or blocks.

Root Frame – a frame which encompasses the entire physical document.

Page – a frame which occupies a rectangular region of a page, on which the document, or part thereof, is physically recorded. A page is the basic physical document object.

Page Set – a frame which consists of multiple pages. An instance of a page set may be a single volume, or chapter, for example.

Block or **Simple Frame** – a terminal, lowest level frame, which, at the given level of granularity, need not or cannot be further decomposed.

A frame is defined to be a recursive component, as it may consist of a set of one or more frames. A block is the terminal frame, which defines a region on the page and has content. Depending on the desired level of granularity, a block's content may correspond to a column, line, word, or character, for example.

Each type of frame has an associated set of attributes. Information about a frame's location, position on the page, justification, etc. is defined by its attributes. For example, a character is characterized by such attributes as font, boldness, point-size, inclination; a line, by maximal length; etc.

Being the leaves of the tree, only blocks have content, while items higher in the hierarchy, which are compound frames, are lists of pointers to lower-level frames. The block's granularity must be at least as fine as the smallest logical component to be described. Thus, for example, if the lowest level logical component is a word, then the block cannot be a line—it must be at least a word, and it may be a character. This data structure enables the reconstruction of the structure of a frame at each level.

Consider, for example, a paragraph which is split over two pages, or a word which is split over two columns. In both cases, a single logical component (paragraph or word) is a combination of two frames, which should be concatenated to yield the entire logical component.

If, on the other hand, a single physical component is found to correspond to more than one logical object, such as the string "092596", which corresponds to a date with three logical sub-components (month, day, year), then the physical representation may be subdivided to reflect the finer granularity. This is quite rare, however, as the physical structure is normally aimed at reflecting the logical one.

An instance of a text-intensive document may have blocks and frames corresponding to a

Character – a block containing the image of a symbol in the document's language.

Word – a frame containing a group of one or more aligned characters, separated by white space.

Word is both a logical object in a document (as we define below), which conveys a certain meaning, and a physical object—the frame (page area) containing the union of its constituent characters. If word is a block, then its content is the image of the word, else it is a frame consisting of characters (each of which is a block), with each character having its image. The specializations of word are as follows:

- **Subword** – a frame containing a group of one or more aligned characters, which make up the first part or the last part of a word.
- **Preword** – a subword containing the first part of a word. A preword is located at the end of a line and ends with a hyphen.
- **Postword** – a subword containing the last part of a word. A postword is located at the beginning of a line and completes its preceding preword to a whole word.

Line – a frame containing 1) a collection of one or more aligned words and 2) at most one preword and at most one postword.

Stack – a frame containing a collection of one or more lines stacked on top of each other and possibly separated by a non-empty white space, such that its logical content is one paragraph[b] at most.

Column – a frame containing a collection of one or more stacks on top of each other. A page may contain one or more columns. In structured documents, this number is generally fixed throughout the document.

3.5. *Generic Logical Structure*

Like the physical structure, which is represented as a hierarchy of frames, the generic logical structure is similarly viewed as a tree, in which the leaves are typically the characters, or symbols, and the root is the entire document. This structure is depicted in Figure 2.

To be able to describe generically the logical document hierarchy that is not restricted by a particular number of levels and associated level names, such as "section" and "chapter", we define the term "texton" as the logical analog of "frame".

[b]Paragraph is a logical term defined below.

Texton – a logical component of a text-intensive document, which consists of one or more (lower level) textons or simple textons.

Root Texton – a texton which is the entire document.

Examples of root textons include book, encyclopedia, concordance, dictionary, journal, newspaper, magazine, report, scientific paper, cover letter and business letter.

Simple Texton – a logical component of a text-intensive document, which is not further divided.

Instances of a simple texton are paragraph, sentence, phrase, word and character. If, for example, word is the simple texton in a particular document, then any subcomponent such as a character is a *primitive texton* document.

Compound Texton – a texton consisting of a distinct header, body, and optional trailer.

One example of a compound texton is a section of a document, which has a header—the section head, a body—the set of one or more paragraphs, and no trailer. Another, less-obvious example of a compound texton may be a signature block in a letter, which contains a header—the closing, a body—the signature, and a trailer—the printed name.

As shown in Figure 2, scanning the logical structure from the top down, the entire document (encyclopedia, book, article, business letter, etc.) is the root texton—the root of the tree. Below it is a varying number of levels of textons. The black triangle along the paths connecting a whole to its parts in Figure 2 is the aggregation symbol (Dori, 1995).

Texton is the logical analog of the physical frame. Like frame, the definition of texton is recursive, and the halting condition is that the constituent texton is a simple texton, i.e., the base logical unit, typically a character. The recursive definition of texton encompasses the entire spectrum of logical levels in any text intensive document, just as the root frame encompasses all the physical levels.

Instances of a texton, in a text-intensive document, include:

Character – a texton which is a symbol in the document's language. Normally, character is a basic texton.

Word – a texton containing a sequence of one or more characters, which has some meaning in the document's language.

As noted, both character and word have a logical, as well as a physical definition. The difference between a logical character and a physical character is that a logical character is the symbol itself, while a physical character is the image representation of the logical character. Likewise, the difference between a logical word and a physical word is that a logical word is a semantic-conveying object, while a physical word can be considered either as the image representation of the logical word or as an ordered collection of its comprising physical characters. Note that there is no logical analog to the physical term subword, whose existence stems from spatial arrangement considerations.

We continue with the definition of higher-level textons.

Phrase – a texton which is a meaningful collection of one or more words that do not necessarily form a complete grammatical sentence.

Examples of phrases include a title of a document or part thereof, a name of a person or an organization, an address, or a (possibly nested) itemized list of such entities.

Sentence – a meaningful collection of one or more phrases which correspond to a valid grammatical sentence, complete with punctuation.

Paragraphon – a texton which is a generalization of a paragraph. It consists of a group of one or more sentences and/or phrases.

Each one of the items above is an example of a paragraphon, where the title is a phrase, and it is followed by another phrase and optional sentence(s).

Unlike character and word, higher-level textons have different names than the corresponding frames, because the physical structure departs from the logical one, and the correspondence becomes more and more fuzzy as we climb up the two hierarchies. Thus, above word at the physical level is the frame called line, while the corresponding textons at the logical level are phrase and sentence. However, obviously it is unlikely that a single sentence occupies exactly one line. At the next level up, the paragraphon is analogous to stack, but again, the correspondence is only partial, because a paragraphon may stretch across more than one stack, if it starts at the end of a column and end at the first stack of the next column or even across several whole columns and pages.

At yet higher levels, the relationships between textons and frames are type-dependent. For example, the texton chapter in a textbook normally starts at a new page, as does a paper in a proceedings.

A document may contain logical elements which are referenced from multiple independent points within the document itself. They are often self contained logical units (i.e., textons) and should be treated as such. To handle such components, we define a *reference* texton.

Referenced Texton – a graphic or textual texton which is referenced from the document.

Examples of referenced components include figures, appendices, footnotes, citations, continuation text body (e.g. in newspapers) and even complete documents. The header of a texton is a label or identifier which is "referenced" by a pointer.

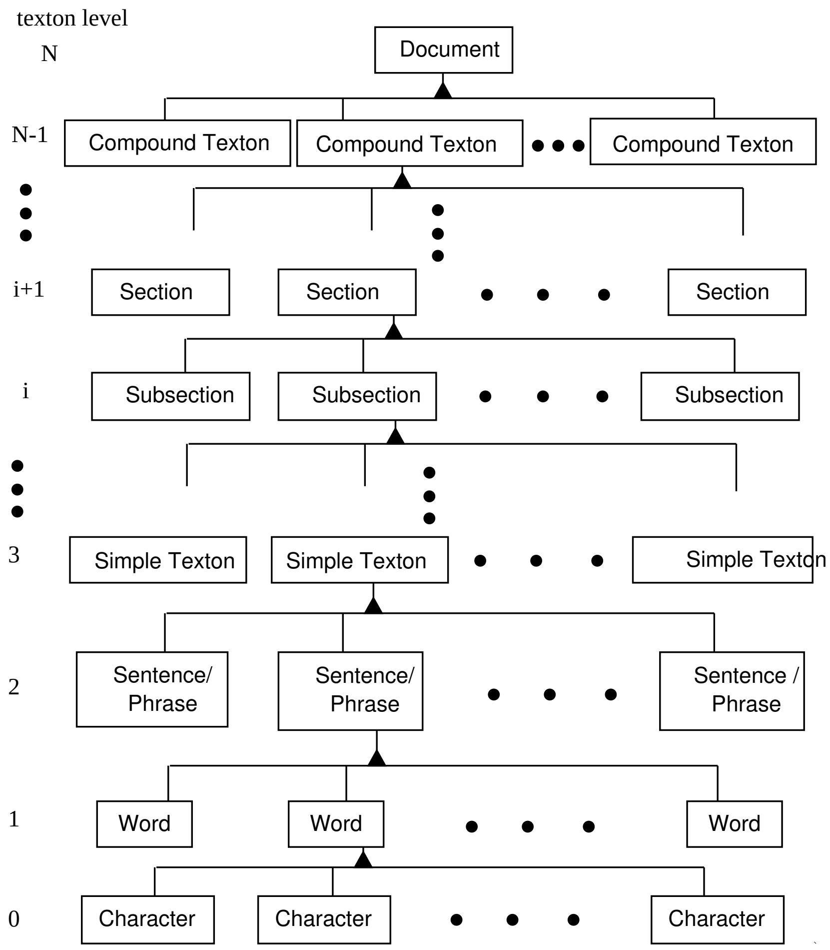


Figure 2: *The logical structure of a document described as a tree*

Pointer – a referencing texton pointing from the main text of the document to the referenced texton. This pointer is typically a phrase or a sentence, such as "continued on page 5, column 3", "see Figure X", or "(Author, 1995)".

Graphon – a referenced texton in a document whose nature is mainly non-textual, and whose function is to illustrate, explain or demonstrate the text. Examples of graphons are line drawings (engineering drawings or art-line), half-tones, photographic (black and white or color) images, maps, diagrams, charts, tables, etc.[c]

In graphons, if text exists, it supplements or enhances the graphic. A graphon has a graphic (imagery or geometric) contents and an optional *caption*, which itself is a texton, and consists of a mandatory *caption header*—the graphon identifier, and an optional caption body—the textual title or explanation of the graphon. The caption header is mandatory, because it serves as a reference pointed to by the text.

Since a graphon, like the figures in this document, normally occupies a considerable portion of the page area, the physical location of a graphon is frequently allowed to *float* in the neighborhood of where it is referred to in the main text for the first time.

Finally, in addition to hierarchal structure given by the recursive definition of the texton, the logical structure must also preserve the reading order.

Reading order - the order in which the characters or symbols in a text-intensive document must be traversed for the document to be correctly understood.

The reading order corresponds to a depth-first visit of the document's logical structure. Reading order normally makes sense only within and between the text-intensive components of a document. In the case of referenced textons, the texton appears physically only once, but may appear logically at many locations. A pointer denotes the logical appearance, so the reading order follows a round-trip "visit" to the referenced texton.

Graphons tend to "float" and can be referenced from multiple locations in the main text. Hence, like any referenced texton, the read-order is preserved by requiring a visit from the reference pointer (typically a phrase), to the graphon and back.

3.5.1. *Document Complexity*

We have seen that a text-intensive document has a hierarchy whose textons depend partially on the document's logical type and may represent chapters, sections, subsections, parts, etc. Hence, the number of texton levels in a document is a finite

[c]A table is a boundary case between text and graphon. We classify it as a graphon, because even though it contains text, the text normally does not have a definite linear reading order and it is normally enclosed within graphics—the lines that separate rows and columns.

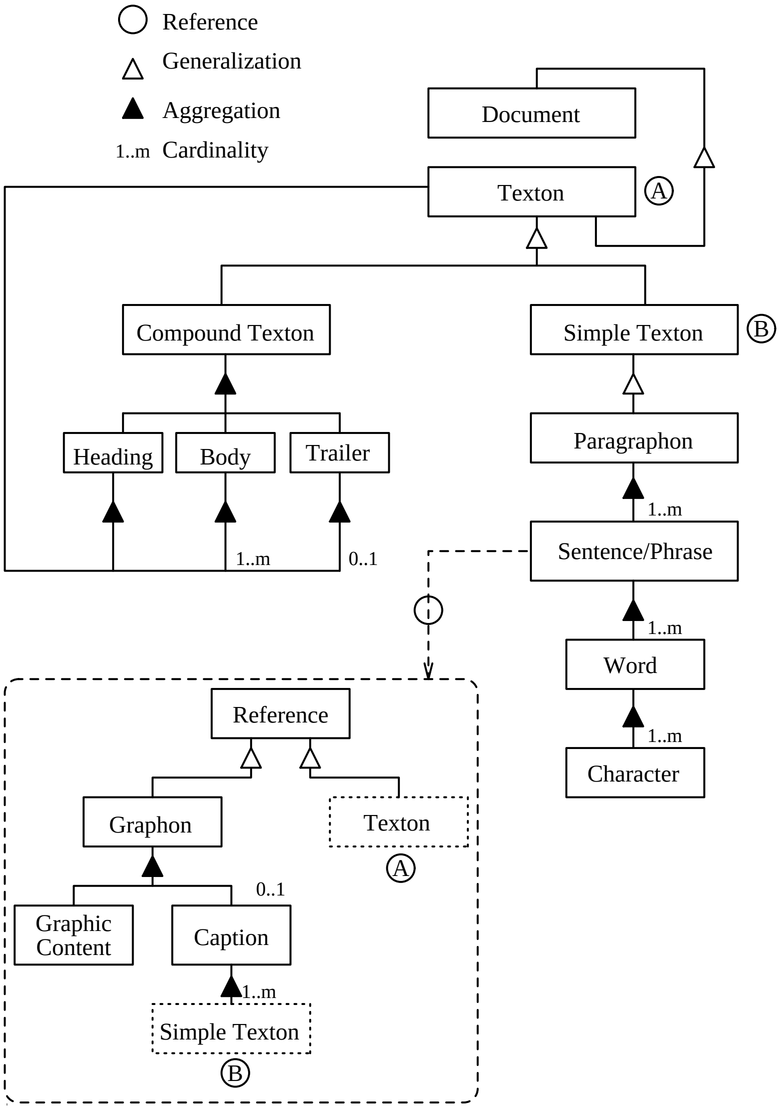


Figure 3: *The object-process diagram of the generic logical structure of a text intensive document*

number, normally not greater than 10. This number depends on the nature of the document and indicates its structural complexity. The numbering of the levels is bottom up, with zero assigned to the character level.

The *logical complexity*of a document is the level number of the document's root texton.

Consider, for example, a journal paper, whose body consists of sections. The body of each section is a paragraphon. Assigning the level numbers 0, 1, and 2 to the character, word, and sentence levels, respectively, a paragraphon is a level 3 texton, and the entire document is a level 4 texton. Hence the complexity of this document is 4. If at least one of the sections is divided into subsections, and no subsection is divided into sub-subsection, then the document complexity is 5.

Although usually there is a relation between the document's size and complexity, these two terms should not be confused. The size can be measured by the number of pages, words or characters. A dictionary, for example, may be a very large document, but its complexity is not necessarily high. Similarly, an outline may be relatively small, but have much higher logical complexity.

3.5.2. *Simple and Compound Textons*

Having defined textons and their roles in the document logical structure, we turn to a more abstract and comprehensive description of the logical document layout than the one given in Figure 2. Figure 3 is an object-process diagram, or shortly, OPD (Dori et al. 1995; Dori, 1995), which describes the structure of a document.

The object Document is a specialization of a Texton, which is the root of the structure. This is denoted by the generalization symbol—the blank triangle going from Texton to Document. A Texton is a generalization of Compound Texton and Simple Texton. This is denoted by the blank triangle from Texton to both Compound Texton and Simple Texton in Figure 3. A *simple texton* is a generalization of a paragraphon.

A character is defined to be a *level 0 texton. A word* is a level 1 texton, as it consists of one or more characters, and a sentence is a level 2 texton. A simple texton in the main text of the document is therefore a level 3 texton. Below it in the main text reside the sentence or phrase (level 2 texton), the word (level 1 texton), and the character (level 0 texton). As we show below, these level numbers may vary for side text, such as the table of contents in a book.

Although in the simplest form, one may conceive of a primitive document consisting of a single character, perhaps conveying a coded message, a single word document, a single sentence/phrase document or single paragraph document, we refer consider the simplest document as a document which is a compound texton. Therefore, the minimal complexity of any document is 4. A simple document, such as a standard business letter, is an example of a level 4 document. It has a header—sender and recipient identification and subject, a body—one or more paragraphs (level 3 textons), and a trailer—salutation, signature, etc.

The black triangle between Compound Texton on one hand and Heading, Body, and Trailer on the other hand, is an aggregation (whole-part) relation, expressing the fact that a compound texton consists of these three parts. The default cardinality (participation constraint) of the aggregation symbol is (1..1):(1..1), i.e., exactly one (minimum 1 and maximum 1) part for exactly one whole.

Consider a texton of level n. The cardinality of the header of this texton is 1, i.e., there is exactly one texton of level $n-1$ which is the header of the level n texton. The cardinality of the texton's body is $1..m$, meaning that there is a number between 1 and many textons of level $n-1$ in the body of the level n texton. Finally, the cardinality of the (optional) texton's trailer is 0..1, i.e., there is at most one texton of level $n-1$ functioning as the trailer of the level n texton. The "0..1" next to Trailer denote that Trailer is optional. In other words, a texton has either two or three parts and must have exactly one Heading, one Body and at most one Trailer. In summary, Heading has exactly one texton, Body has a number of between 1 and many (denoted "$1..m$" in Figure 3) textons, and Trailer, if it exists, has one texton.

For example, a section in a paper is a compound texton. It has a header—the section title; a body, consisting of one or more paragraphons; and no trailer. As another example, a textbook is a compound texton, whose heading is everything from the beginning of the book to the beginning of the first chapter. The body of the book consists of a number of chapters and its trailer is everything from the end of the last chapter to the end of the book (appendices, glossary, index, etc.).

3.5.3. *The Recursion in Texton Definition; the Body Path*

Since Compound Texton is Texton, and Compound Texton has Heading, Body and Trailer, each having at least one Texton, we get a recursive definition. As in any recursion, to avoid infinite looping, a halting condition must exist. The halting condition, as expressed in the object-process diagram of Figure 3, occurs when the textons of Heading, Body and Trailer of the Compound Texton are all Simple Textons. When a texton is simple, the recursion stops, because from this level downward, we descend through the phrase level and the word level down to the character level. In the case of a referenced texton, the pointer—a Simple Texton in the main text—links it to preserve the reading order, while the referenced texton itself is a Compound Texton, whose header is the identifier the pointer points to.

Body Path is the path in the tree structure going from the root node—the entire document—through successively decreasing levels of compound textons, all the way down to the simple texton (the paragraphon level), such that the path always visits the body of each texton. Since by definition any compound texton has a body, such a path is guaranteed to exist, and it is unique.

The level of a character, which is the last node — the leaf — along the body path, is defined to be zero. This implies that along the body path the level number of a word is one, the sentence/phrase level is two, and the level of the paragraphon—the simple texton—is three. Note that these numbers are not necessarily the same for characters, words, sentences and paragraphs which are not nodes along the body path. As we show in the example below, the level numbers may be higher or lower than the ones along the body path, depending on whether the path from the top texton (the document level) is longer or shorter than the length of the body path, respectively. The fact that of the three compound texton parts only two are mandatory, gives rise to a 2-3 tree structure, as we demonstrate in the example in the next section.

3.5.4. *DAS94 Proceedings—a case in point*

To demonstrate the use of the concepts and terms presented above, and to show how document complexity is defined, consider the document *Proceedings of DAS94* (Dengel and Spitz, 1994). The structure of the document is described in Figure 4. The structure is detailed down to the Simple Texton level. Since a compound texton may have either three parts (Header, Body, and Trailer) or two parts (Header and Body), the resulting structure in Figure 4 is a 2-3 tree.

As indicated in the legend of Figure 4, the body path is marked by thick line segments. The level numbers are written in parentheses next to the corresponding textons along the path. The body path visits the nodes "Proceedings of DAS94", "Session", "Paper", "Section", "Subsection", and "Paragraph", in this order. Assigning the number 3 to the paragraph level and counting up we get that "Subsection" is at level 4, "Section" is at level 5, "Paper" is at level 6, "Session" is at level 7, and the entire document "Proceedings of DAS94" is at level 8. Hence the complexity of this document is 8.

As a compound texton, "Proceedings of DAS94" has a header, a body and a trailer. The header is a level 7 texton, which, in turn, consists of three level 6 texton: a header—"Front Page" and "Copyright note", a body—"Chairmen's Message", and a trailer—"Table of Contents". Chairmen's Message consists of a level 5 header—the title "Chairmen's Message," a level 5 body, consisting of seven level 4 paragraphons, and a level 5 trailer, containing two level 4 paragraphons. The first phrase is "Kasierslautern, October 1994," and the second—the names of the two document editors. As we see here, both the paragraphs and the phrases, which are basic textons, are at level 4 rather than 3. The reason is that the path traversed here is not the body path. As noted, a basic texton is guaranteed to be at level 3 exactly only when it is on the body path. In other paths it may be more (as here) or less than 3. The path that ends with "Author", "Affiliation", and "Address", for example, is the longest one. It is longer by two edges than the body path. Therefore "Address", which is a simple texton, is a level 1 texton in this case, as shown at the bottom of Figure 4. Table of Contents is a level 6 texton, consisting of a header—the title "Table of Contents," a body, and no trailer. The body of the Table of Contents consists of eight items. Each item is a level 5 texton called Session Contents. It has a header—session number and name, a body—a phrase (itemized list) of three level 4 items, each called Paper Details, and no trailer. Each Paper Details item is a level 3 texton. It consists of three paragraphons, each containing

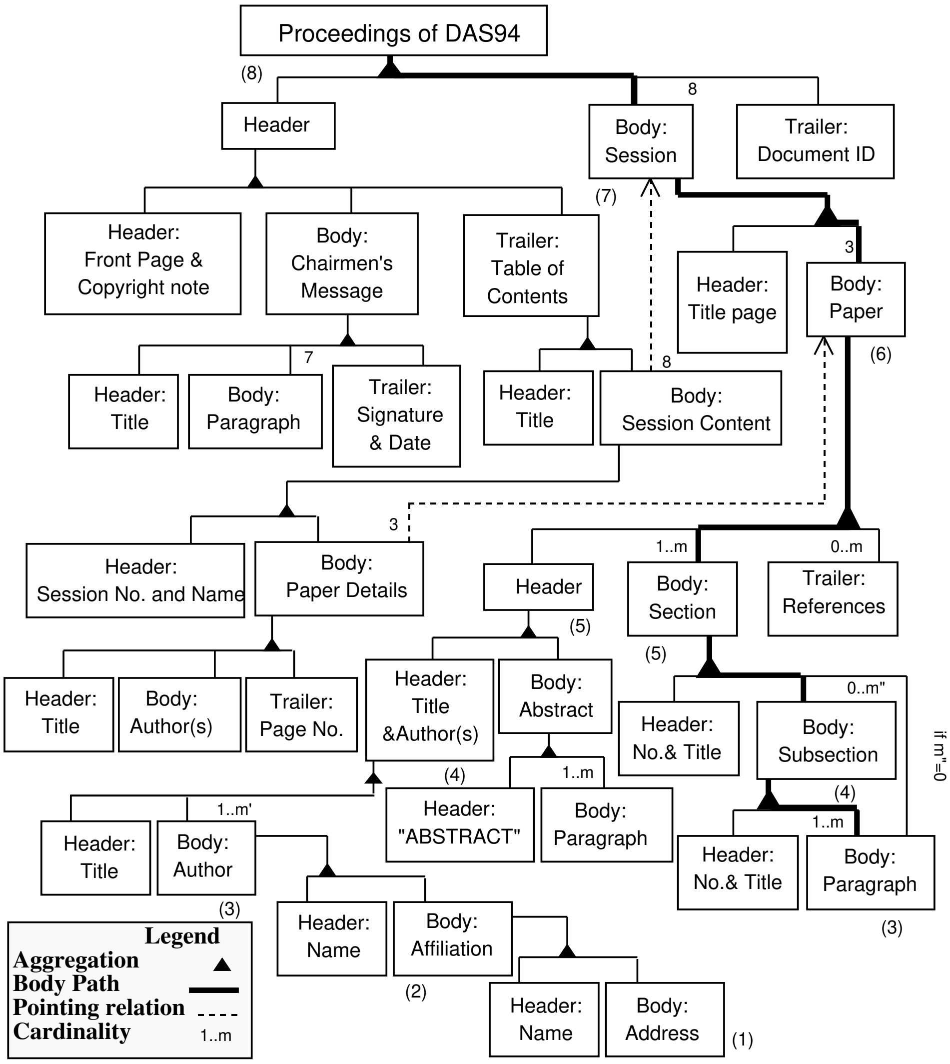


Figure 4: *An OPD describing an instance of a generic logical structure — the Proceedings of DAS'94 (Dengel and Spitz, 1994)*

a single phrase. The first phrase is the paper name, the second phrase is the author name, and the third phrase is the page number.

In most of the papers, Section consists directly of paragraphs, but several papers have subsections (see for example page 139 in the document). To accommodate this variability, we add the condition "if $m'' = 0$" along the aggregation link from Section to Paragraph in Figure 4, where m'' is the number of subsections in a section. This means that if there are no subsections in the section, then Section consists directly of paragraphs.

3.6. *Relating Physical and Logical Structure*

Having defined generic terminology for both logical and physical structure, it is straightforward to relate the two at the content level, in most cases. Figure 5 shows the structure of a simple document, a multiple page chapter. The chapter is a compound texton, with a header (title) and two body components (one abstract and one section). The abstract is a simple texton and the section is a compound texton, consisting of two simple textons (paragraphons).

The physical structure subdivides the document into rectangular blocks. The content is shared in both structures. Note that the structure allows logical components (i.e., the abstract) to be split over two pages. For most documents, the logical complexity will be higher.

4. The Document Attribute Format Specification — DAFS

Integrating the physical structure with the logical one is a difficult problem. We now return to the physical layout as described by DAFS – Document Attribute Format Specification.

While many formats exist for composing a document from electronic storage onto paper, no satisfactory standard exists for the reverse process. DAFS is a file format specification for documents with a variety of uses. It is developed under the Document Image Understanding (DIMUND) project funded by ARPA and is meant to be the file format for all documents used whose content has been examined either manually or automatically and which form parts of DIMUND databases. In addition, DAFS-formatted documents are used in the Illuminator project, where they are employed for training and testing document image understanding tools.

DAFS is intended to be a standard for the representation of document images and their partial interpretations during document decomposition. It is hoped that DAFS will prove to be general enough to enjoy widespread use, particularly in applications such as OCR and document image understanding. Several standards have been developed which address the creation or composition of documents, but none of these standards is well suited to the problem of document decomposition. There are many applications which would require some form of document decomposition, including character recognition and document image understanding. DAFS is a new format, designed explicitly for representing reverse encoding—the encoding of decomposed documents. As such, DAFS is designed to allow representation of both the physical and semantic information contained within a document image, but it is desired that this format have many applications beyond these specific ones. With this as a goal, the principle was established that the format should be extensible to meet the needs of a broad base of users, and that the format should not impose unnecessary assumptions on the potential users.

4.1. *DAFS Design*

This section describes the philosophy which was used in the development of DAFS.

4.1.1. *Object-Oriented Design*

In the process of reverse encoding a document image, it becomes clear that it would be convenient to be able to handle portions of the image as discrete objects. An object might be any part of a document that may be defined in a stable form. Under DAFS, an object is called an "entity", and is essentially one or more rectangular pieces of image from the document. Note that a DAFS "entity" is analogous to the SGML element and has nothing to do with the SGML concept of entity. Examples of useful entities are "paragraph", "character", and "document". Each may be part of a document, but needs to have a sufficiently stable form to be unambiguously defined. By implementing this specification in an object-oriented fashion, each entity may have any number of properties associated with it, allowing information about the entity to be entered. (See "DAFS Entities" for more information.) Another advantage of an object-oriented format is that an object may also contain other objects, which is the key to a hierarchical structure, and spreading attributes to specializing objects.

4.1.2. *Hierarchical Design*

Often the objects we wish to define in a document fit into a hierarchical relationship. For example, a character may be contained within a word. That word may then be contained within a line of text. This in turn may be within a paragraph, within a page. All of these may be part of a document. DAFS provides the ability to create "parent", "child" and "sibling" relationships between entities, forming the basis for specifying any hierarchy desired.

The use of a hierarchical structure for describing objects within a document can make the description of the document more compact. Users can leave out the layers or details that they do not need. For example, an OCR application might use the structure document-paragraph-line-word-character in processing a document image. A pitch and phase detector, working with the same image, might use the structure document-line and have no need to word and character information.

4.1.3. *Extensibility*

As with any large software system, we do not expect that any primary design will be general enough to accommodate all potential applications of DAFS. Hence, an important design goal of DAFS is to give it the greatest potential for extension. In addressing such future needs, the current specification is designed to provide a practical degree of extensibility, and to permit asynchronous revisions of data and applications to coexist.

The use of discrete objects to describe document structure permits easy extension to the set of objects available for this purpose. In another venue, the availability of an unlimited number of properties for each object permits user extensions to document descriptions. The user may classify, modify, or record information about a pre-existing document's content by creating and using new properties. (See "Properties" for additional information.)

In order to preserve the extensibility of DAFS and to maintain compliance with this specification, a process that conforms to DAFS must either interpret code values as specified, or pass through these values and not interpret them at all. A process must not change a code that it cannot interpret.

4.2. *Primary DAFS Requirements*

There are a number of requirements which DAFS must meet to fulfill its purpose.

In general DAFS must be powerful enough to serve as the format for database and tools document interchange. In was developed with the hope that at some point, this specification will develop into a standard for data interchange in the document understanding community.

Of secondary interest, it should offer expandability, allow alternatives/possibilities (e.g., allow a supposed character's content to be labeled 'm' or 'rn' or 'iii'), allow confidence values for each alternative, so that a measured choice can be made among them, support many human languages, help tools process images rapidly, and be human readable (to allow editing on non-DAFS tools), among others.

4.3. *DAFS and existing standards*

The definition of DAFS has been influenced by a number of current standards. Some of these are from private companies, while others are international standards. Among the private formats are IBM's RFT:DCA, the Rich Text Format (RTF) format from Microsoft, and the CDA from Digital Equipment Corporation. International standards have been implemented by the International Organization for Standardization (ISO) and the Comité Consultatif International Télégraphique et Téléphonique (CCITT) to assist in the open exchange of documents. These standards include the Open Document Architecture (ODA), ISO 8613, and the Standard Generalized Markup Language (SGML), ISO 8879. The three of these which most influenced DAFS are SGML for overall structure and text handling, CCITT group IV for images, and ISO 10646 for Unicode.

These formats have been made available to the public and are implemented by numerous vendors. Each was created to provide a common interchange format for moving documents among heterogeneous document formatting and text publishing systems. Since they were originally conceived for encoding documents for publishing applications, none of these formats adequately address the problems of document decomposition and reverse encoding.

4.3.1. *DAFS and SGML*

The advantages of SGML are so strong it was decided to use it as a basis for DAFS. Some of these advantages include:

1. SGML is well-designed. Much thought has gone into what makes up a document; how to handle natural hierarchies and nesting of hierarchies; how to link document content and markup, yet be able to distinguish between the two.
2. SGML is designed to be general and to allow modification. For example, the character set can readily be changed, enabling SGML to handle documents in many foreign language scripts.
3. While by no means perfect in handling non-English text, SGML offers several reasonable alternatives. These are discussed in "DAFS-U Storage Format".
4. SGML was designed to be easy to parse. It is possible to start anywhere in an SGML document and be able to discover where you are without scanning from the beginning. Elements not recognized by the current application are easily ignored.
5. The user does not need to know everything about a document to begin using SGML. It is not necessary to know the total number of paragraphs, for example.
6. SGML is a well-known and widely used standard.

DAFS is being implemented as a special SGML application with a set Document Type Definition (DTD), but with some built-in features providing the extensibility and flexibility to cover a wide range of applications. The three main disadvantages of SGML and their solutions are discussed below.

1. SGML discourages encoding of physical characteristics. It was decided to overlook SGML inhibitions about encoding physical attributes. A DTD was created which encodes essential attributes, including physical ones like "bold" and "point size", and which allows users to add their own.
2. SGML does not handle images easily. Document decomposition applications will often require both image and text (for example, the image of a page and the corresponding OCR'd text). SGML was not designed with this need in

mind. The difficulty arises from the need to distinguish intentional SGML tags from chance sequences of image bytes which, by chance, read the same. To overcome this, DAFS could pursue one of the following options:

- Escape the images. This method of handling the problem makes reading and writing tedious.
- Write the image after the final tag of the SGML data. Unfortunately, the end of an SGML file is not well defined, and not all SGML parsers would necessarily interpret this the same way.
- Write the image in an associated external file which is referenced by the text file. Maintaining multiple files for one document can be inconvenient and is not aesthetic, but there is no question regarding which bytes are image and which are text.

The decision was to use references to external image files, arguing that the negative aspects of storing a single document across more than one file are offset by the certain knowledge of which bytes are text and which are image. Including image with text via external files is not without precedent. The CALS (Computer-aided Acquisition and Logistics Support) standards by the US Department of Defense call for just such handling of mixed text and image. Under CALS, text is to be converted to SGML and image to another format, and the two information types are stored in separate but linked files.

3. SGML applications are not necessarily portable from one DTD to another, yet different applications will require different DTD's. We are alleviating this problem by carefully constructing the DAFS DTD to be as generally applicable as possible, and by building in a limited expandability through DAFS "properties". These properties provide a way for users to create new categories of information about a document's entities, and are discussed further under "Properties".

4.4. *DAFS Tags*

The following suggestions regarding tags have guided the creation of the DAFS tagset and DTD:

1. Because DAFS aims to support many languages and scripts, DAFS will incorporate the Unicode character set. SGML (and by extension, DAFS) allows any character to be used in a tag, and the idea of specifying a defined set of tags violates the complete freedom of SGML. Nevertheless, it is highly recommended that all tag characters be the Unicode equivalent of ASCII. Limiting tag characters to ASCII helps maintain their human-readability, and eases the interconversion of DAFS-U and DAFS-A—two DAFS file types, discussed in "DAFS Storage Formats".

2. It is anticipated that new tags will be developed by users, but in the interest of portability of the resulting documents, the DAFS-defined tags should be used as much as possible.

3. High-frequency entity names should be kept short.

4.5. *DAFS Storage Formats*

DAFS has three storage formats, each designed for a different purpose, but all three are wholly interconvertible. These are the compact binary format DAFS-B (BINARY), and the "human-readable" DAFS-A (ASCII) and DAFS-U (Unicode). In DAFS-B, image and text (if any) are stored in a binary format in one file. DAFS-A is a direct application of SGML. DAFS-U is similar to DAFS-A, but modified to allow Unicode characters as content. In DAFS-A and DAFS-U, images (if any) are included in external linked files. Software libraries which enable reading and writing of all three versions have been developed and made publicly available.

4.6. *DAFS Entities*

DAFS entities are conveniently defined objects within a document such as a paragraph or word. In essence, an entity is one area of an image which is usually defined by a bounding box (though it need not be). An entity can have content, which might be the text it encompasses, properties, such as bounding box, font and point size, and hierarchical relationships with other entities, allowing specification of read orders and page layouts.

Prospective users have requested that a list of document elements and characteristics be definable under DAFS and DAFS has been structured to make each of the following DAFS entity types available.

Doc	The document as a whole.
Page	A given page.
Column	A column of text.
Paragraph	A delimited block of text comprising a paragraph.
Line	A line of text.
Newline	A newline, or carriage return.
Word	A word in the text.
Glyph	A single character in the text. "Glyph" rather than "character" is used because we are referring to an area of image which is meant to be a character, but which may not actually be correctly segmented.
Space	A Glyph whose textual content is the space character.

4.6.1. *Hierarchical Relationships*

DAFS permits the creation of parent, child and sibling relationships between entities, providing easy representation of the hierarchical structures of a document.

As an example, consider a paragraph entity made up of words which are in turn composed of glyphs or characters. The component glyphs are the child entities of each word, while the paragraph is the parent of each word. The other words in the paragraph are a given word's siblings.

4.6.2. *Properties*

An entity may have various attributes or "properties" associated with it. A few predefined properties are listed below.

bounding box	Rectangular box delimiting the entity or portions of it.
font class	Information on the character font (e.g. Courier or Helvetica).
point size	Size of the printed characters.
bold	Characters printed with thicker lines for emphasis.
italic	Characters printed with slanting lines for emphasis.

DAFS permits the creation of an unlimited number of user defined properties. A property is used to describe or classify an entity and its contents, and exists only in association with the entity to which it refers. In SGML applications, such attributes are generally predefined in the DTD. DAFS introduces user-defined properties as a way for users to create their own entity categories and descriptions, without the need to alter the underlying DAFS DTD. It provides flexibility for handling a large variety of applications, yet protects the ability to share tools and data.

4.6.3. *Confidences*

Since DAFS is meant for use with document decomposition, and since there is always some uncertainty or ambiguity associated with determining exactly what the content of an entity is, DAFS must be able to assign confidence values to all its entities.

conf	Contains the confidence in the value of an entity's contents or its properties. It defaults to a single unsigned byte 0-255.
alternative set	This is a list of alternatives suggested or allowed as the content of an entity. Alternatives within an alternative set are meant to be read as either the first one or the second, and so on.

4.6.4. *Alternative Sets and Property Ranges*

A related idea is the allowed range of values for properties. We anticipate the existence of tools using DAFS which test the effectiveness of automatic character recognizers, page decomposers, and other image understanding tools. For example, an automatic page decomposition tool might put a bounding box around a paragraph, different from the one the human creating the test set had assigned. The testing tool must determine whether the machine-set bounding box is close enough to the human-created 'ideal'. Since exact matches are not required for this type of application, the exact values of some properties may be uncertain. DAFS accommodates this need with entities which set the allowed range of properties. The property "bold" is another example. It may have an allowed range of 100-200 for font class 1. If "boldness" of a glyph is measured as 132, this is within the range and it will be concluded that the glyph is in fact bold. For consistency, all other bold glyphs like it from font class 1 should also have a boldness of 132.

4.6.5. *Alternative Contents: Or and Borrow*

DAFS must be able to handle alternative values for entity content. Any attempt to decompose a document will engender areas of uncertainty. A classic OCR uncertainty, for example, involves distinguishing 'I' (capital I), '1' and 'l' (lower case L). If just one of the three is selected as "most likely", the fact that the other two were very nearly as likely is lost. DAFS provides easy means of preserving and presenting sets of such alternatives. The use of alternatives is available not only for sets of characters, but for any other kind of entity as well.

The key to DAFS alternatives involves the concepts of child type and entity borrowing. An entity's children may be of "And" type, such as the component glyphs of a word, which are all meant to be presented together. "Or" type children, on the other hand, are alternatives of one another; only one of the set can be present at one time. A glyph may have "Or" children 'I', '1' and 'l', allowing a variety of techniques to be tried in selecting the best of the possibilities.

Entity borrowing is another useful device which permits easy data sharing among entities. As an example, consider a document that has a read order different from the physical order of the entities on the page. The Document could be an "Or" type entity with two child entities. The first child would arrange the Paragraphs, Words, etc. to represent the read order. The second would arrange them to represent page layout, borrowing the same images used by the read-order child, but arranging them differently. The Borrow concept allows the data to appear only once, but to be arranged and used in multiple ways. Through Borrowing, DAFS files can be more compact than would otherwise be possible.

4.6.6. *Alternative Read Orders and Page Layouts*

Documents can have multiple allowed read orders and page layouts, and it will be desirable to encode them into the document itself for the applications which use them. Automatic testers might use this information when evaluating page decomposition systems. Alternative read orders and page layouts rely on the entity borrowing capability discussed above, so that the same entities from the image of the document can be linked together in different orders.

5. Representing Textons and Specific Structure in DAFS

During the document understanding process, we must analyze and infer the *specific instances* of the logical and physical structures of the document. DAFS provides mechanisms which allow us to represent both the logical and physical

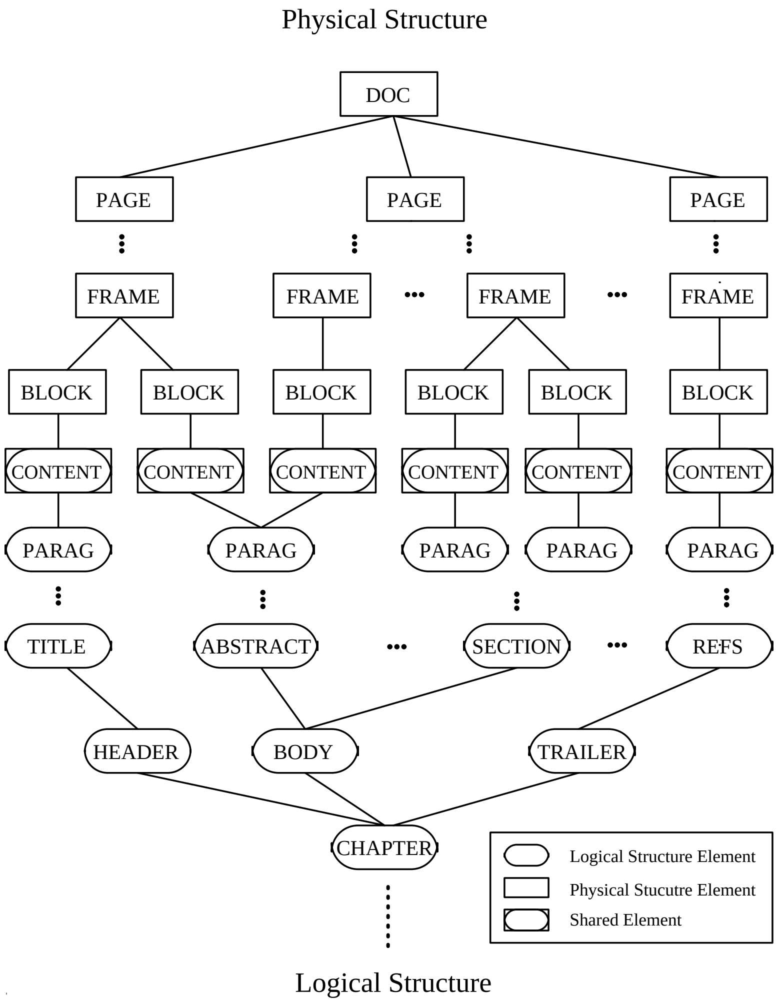


Figure 5: *A sparse schematic representation of the physical and logical structure of the content of a document "chapter"*

structures, as well as alternative structures and associated confidences. Using "or" children, multiple hypotheses can be carried along in the reverse encoding process.

Recall that DAFS provides a basic entity structure, which may have a property list and data, as well as child, image and borrowed entities. The property list has both system-level and user-defined properties for information such as the entity name, relationships among children, and bounding boxes for physical entities.

The encoding of physical information is straightforward and was highlighted previously in the description of DAFS (Section). Each physical "object" in the document is represented by a single DAFS entity. From the generic description, each physical entity will correspond to a component of the generic physical description — Page set, Page, Frame or Block. Frames will have content which corresponds to a list of frames and/or blocks and blocks will have content which corresponds to a physical region on the page.

In the DAFS representation, the physical entity's *type* will correspond to a physical document component, such as column, paragraph, line, word or character. Which physical components are valid, however, is clearly dependent on the document's type.

Logical entities will be organized similarly in a hierarchical manner. Entities will be used to represent logical components and are, once again, document class dependent. For a journal article the types may correspond to a page, section, paragraph, etc., whereas for a business letter, the entity types may include the address block, greeting and salutation, for example. Each logical entity is related to other entities with parent, child, and sibling relationships, inferred from the hierarchy. The concept of a referenced texton is implemented in DAFS by a user defined *reference* property which provides the ID of a logical (or physical) entity which it references.

There are also several special purpose entities which we define to aid in organizing the physical and logical entity trees. The *Document Root* represents the root of the document and has two children - the *physical root*, pointing to the physical hierarchy, and the *logical root*, pointing to the logical hierarchy. The children of the physical root are simply the largest physical components of the document, e.g. pages in a journal article, page sets in a multi-volume document, etc. The children of the logical root are the maximal logical components. One component is likely the main body, and the remaining components are often referenced textons (or sub-documents) such as figures. Distinguishing between logical components at such a high level is necessary because of the fact that although figures are embedded in the physical document, they are disjoint (and referenced) from the main body of the document.

The terminal components for both hierarchies are characters and graphic blocks. Both hierarchies descend independently toward the leaf nodes, where they share the character and graphic components via the DAFS borrow construct. From this representation, given a logical component, one can obtain relevant physical information and vice versa.

6. Summary

The structure of a document conveys semantic information that is beyond its character string contents. To capture this additional semantics, document understanding must perform a reverse encoding of the document and relate the physical layout to the logical structure. This work has proposed a formal generic framework for the definition and interpretation of any text-intensive document physical and logical structure, that is not restricted by the size or complexity of the document. The physical document is described by a hierarchy of frames and the logical structure of text-intensive documents is described as a hierarchy of textons. The definition of textons provides a powerful and flexible tool for document logical structure analysis. We also propose a method for determining quantitatively in an objective, reproducible, and unbiased way the complexity of such documents.

We have also presented a description of a new and powerful document attribute format specification—DAFS, which provides mechanisms for representing and maintaining both physical and logical information during the reverse encoding process, and have shown how it can be used to relate logical and physical structure at the content level.

Acknowledgments

The partial support of this research by the Technion VPR Fund and by the Advanced Research Projects Agency is gratefully acknowledged, as is the help of Azriel Rosenfeld in editing this chapter.

The current versions of the DAFS document, DAFS Library and Illuminator Software are available at http://documents.cfar.umd.edu.

References

[1] O.T. Akindele and A. Belaid, "Page Segmentation by Segment Tracing," *2nd ICDAR*, Tsukuba, 1993, p341-344.

[2] N. Amamoto, S. Torigoe, and Y. Hirogaki, "Block Segmentation and Text Area Extraction of Vertically/Horizontally Written Document," *2nd ICDAR*, Tsukuba, 1993, p739-742.

[3] H.S. Baird, "Background Structure in Document Images," *Advances In Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, p253-269.

[4] H.S. Baird, "Document Image Defect Models and Their Uses," *2nd ICDAR*, Tsukuba, 1993, p62-67.

[5] D.S. Bloomberg, "Multiresolution Morphological Approach to Document Image Analysis," *1st ICDAR*, Saint-Malo, 1991, p963-971.

[6] R.G. Casey and G. Nagy, "Document Analysis – A Broader View," *1st ICDAR*, Saint-Malo, 1991, p839-849.

[7] S. Chen and R. Haralick, "An Automatic Algorithm for Text Skew Estimation in Document Images Using Recursive Morphological Transforms," *Proceedings, ICIP-94: 1994 IEEE International Conference on Image Processing*, Austin, TX, November 13–16, 1994, Vol. I, p. 139–143.

[8] Y. Chenevoy and A. Belaid, “Hypothesis Management for Structured Document Recognition,” *1st ICDAR*, Saint-Malo, 1991, p121-129.
[9] A. Dengel, “Initial Learning of Document Structure,” *2nd ICDAR*, Tsukuba, 1993, p86-90.
[10] A. Dengel, L. Spitz, (Eds.) *Proceedings of DAS94—Document Analysis Systems*, Kasierslautern, Germany, October 18-20, 1994. DFKI Document D-94-06.
[11] D. Derrien-Peden, “Frame-based System for Macro-typographical Structure Analysis in Scientific Papers,” *1st ICDAR*, Saint-Malo, 1991, p311-319.
[12] D. Dori, “Object-Process Analysis: Maintaining the Balance Between System Structure and Behavior,” *Journal of Logic and Computation*, 5, 2, April 1995, p1-23 (to appear).
[13] D. Dori, I. Phillips and R.M. Haralick, Incorporating Documentation and Inspection into Computer Integrated Manufacturing: an Object-Process Approach. In *Applications of Object-Oriented Technology in Manufacturing*, S. Adiga (Ed.), Chapman & Hall, London, 1995 (to appear).
[14] F. Esposito, D. Malerba, G. Semeraro, “An Experimental Page Layout Recognition System for Office Document Automatic Classification: An Integrated Approach for Inductive Generalization,” *10th ICPR*, Atlantic City, 1990, p557-562.
[15] J.L. Fisher, “Logical Structure Descriptions of Segmented Document Images,” *1st ICDAR*, Saint-Malo, 1991, p302-310.
[16] J.L. Fisher, S.C. Hinds, and D.P. D’Amato, “A Rule-Based System for Document Image-Segmentation,” *10th ICPR*, Atlantic City, 1990, p567-572.
[17] H. Fujisawa and Y. Nakano, “A Top-Down Approach for the Analysis Of Documents,” *Proceedings of the IAPR Workshop on Syntactic and Structural Pattern Recognition*, Murray Hill, New Jersey, 1990, p113-122.
[18] X. Hao, J.T.L. Wang, and P.A. Ng, “Nested Segmentation: An Approach for Layout Analysis in Document Classification,” *2nd ICDAR*, Tsukuba, 1993, p319-322.
[19] J. Higashino, H. Fujisawa, Y. Nakano, and M. Ejiri, “A Knowledge Based Segmentation Method for Document Understanding,” *8th ICPR*, Paris, 1986, p745-748.
[20] S.C. Hinds, J.L. Fisher and D.P. D’Amato, “A Document Skew Detection Method Using Run-Length Encoding and the Hough Transform,” *10th ICPR*, Atlantic City, 1990, p464-468.
[21] Y. Hirayama, “A Block Segmentation Method for Document Images with Complicated Column Structures,” *2nd ICDAR*, Tsukuba, 1993, p91-94.
[22] R. Ingold and D. Armangil, “A Top-Down Document Analysis Method for Logical Structure Recognition,” *1st ICDAR*, Saint-Malo, 1991, p41-49.
[23] Yasuto Ishitani, “Document Skew Detection Based on Local Region Complexity,” *2nd ICDAR*, Tsukuba, 1993, p49-52.
[24] D.J. Ittner and H.S. Baird, “Language-Free Layout Analysis,” *2nd ICDAR*, Tsukuba, 1993, p336-340.
[25] B. Julesz and J.R. Bergen, “Textons, the Fundamental Elements in Preattentive Vision and Perception of Textures”, *Bell System Technical Journal*, 1983, p1619-1645.
[26] J. Kanai, T.A. Naratker, S.V. Rice, and G. Nagy, “Performance Metrics for Document Understanding Systems,” *2nd ICDAR*, Tsukuba, 1993, p424-427.
[27] T. Kanungo, R.M. Haralick and I.T. Phillips, “Global and Local Document Degradation Models,” *2nd ICDAR*, Tsukuba, 1993, p730-734.
[28] J. Kreich, A. Luhn, and G. Maderlechner, “An Experimental Environment for Model Based Document Analysis,” *1st ICDAR*, Saint-Malo, 1991, p50-58.

[29] F. Lebourgeois, Z. Bublinski, and H. Emptoz, "A Fast and Efficient Method For Extracting Text Paragraphs and Graphics from Unconstrained Documents," *11th ICPR*, The Hague, 1992, p272-276.
[30] G. Nagy and S.C. Seth, "Hierarchical Representation of Optically Scanned Documents," *7th ICPR*, Montreal, 1984, p347-349.
[31] G. Nagy, S.C. Seth, S.D. Stoddard, "Document Analysis with an Expert System," *Pattern Recognition in Practice II*, E.S. Gelsema and L.N. Kanal, editors, North Holland, Amsterdam, 1986, p149-159.
[32] L. O'Gorman, "The Document Spectrum for Bottom-Up Page Layout Analysis," *Advances In Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, p270-279.
[33] T. Pavlidis and J. Zhou, "Page Segmentation by White Streams," *1st ICDAR*, Saint-Malo, 1991, p945-953.
[34] I.T. Phillips, S. Chen, and R.M. Haralick, "CD-ROM Document Database Standard," *2nd ICDAR*, Tsukuba, 1993a, p478-483.
[35] I.T. Phillips, J. Ha, R.M. Haralick, and D. Dori, "The Implementation Methodology for a CD-ROM English Document Database," *2nd ICDAR*, Tsukuba, 1993b, p484-487.
[36] T. Saitoh and T. Pavlidis, " Page Segmentation without Rectangle Assumption," *11th ICPR*, The Hague, 1992, p 277-280.
[37] T. Saitoh, M. Tachikawa, and T. Yamaai, "Document Image Segmentation and Text Area Ordering," *2nd ICDAR*, Tsukuba, 1993, p323-329.
[38] Y.Y. Tang, C.Y.Suen, C.D. Yan, and M. Cheriet, "Document Analysis and Understanding: A Brief Survey," *1st ICDAR*, Saint-Malo, 1991, p17-31.
[39] S. Tsujimoto and H. Asada, "Understanding Multi-articled Documents," *10th ICPR*, Atlantic City, 1990, p551-556.
[40] F.M. Wahl, K.Y. Wong, and R.G. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," *Computer Graphics and Image Processing*, Vol 20, 1982, p375-390.
[41] A. Yamashita, T. Amasno, H. Takahashi, and K. Toyokawa, "A Model Based Layout Understanding Method for the Document Recognition System," *1st ICDAR*, Saint-Malo, 1991, p130-138.
[42] C.L. Yu, Y.Y. Tand and C.Y. Suen, "Document Architecture Language (DAL) Approach to Document Processing," *2nd ICDAR*, Tsukuba, 1993, p103-106.