IJDAR (1999) 1: 209–220

International Journal on IJDAR
Document Analysis and Recognition

Content based internet access to paper documents

Marcel Worring, Arnold W.M. Smeulders

Intelligent Sensory Information Systems, University of Amsterdam, Faculty WINS, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands; e-mail: worring@wins.uva.nl, http://carol.wins.uva.nl/~worring/

Received October 13, 1998 / Revised February 15, 1999

Abstract. When archives of paper documents are to be accessed via the Internet, the implicit hypertext structure of the original documents should be employed. In this paper we study the different hypertext structures one encounters in a document. Methods for analyzing paper documents to find these structures are presented. The structures also form the basis for the presentation of the content of the document to the user. Results are presented.

Key words: Document access – Document understanding – Hypertext structures – Hypertext presentation

1 Introduction

With the advent of the Internet, remote access to archives of digitized paper documents is now feasible. It is becoming an important tool in sharing paper-based information. As paper has been the chosen mode of communication for centuries this has an impact in many applications. For archived scientific journals it allows one to browse through older issues for reference purposes, or to follow threads for particular topics. In design archives of buildings, airplanes and ships, it enables one to study the construction, or obtain the required information for maintenance. Digitally stored patent applications aid in studying whether a patent application is worth granting, both for the applicant and the patent officer. Remote access to manuals, in principle, permits the employment of more effective, non paper-based, learning methods. In a digital museum access to precious documents, which would otherwise be hidden from the public, can be granted.

Making digitized documents available through the Internet is mostly done by showing the facsimile of the document for visual inspection. Analysis of the content and structure of the document is limited to textblocks. Optical character recognition (OCR) is then used to create a keyword or full-text index. In such systems, the user selects a document through the index and then views or prints the facsimile. Hence, the sophisticated Internet browsing and presentation tools are used for selection and presentation of facsimile of pages only.

The content of an archived document is not just text, but a highly structured collection of text fragments, figures, and tables. Document image understanding tools (e.g.,[4][16]) provide the means to analyze this structure to go beyond a simple index. Such tools concentrate on the geometric structure of the document and in addition sometimes consider the logical structure. As a consequence, the linear reading order of documents still dominates access to the archives. However, the content of documents would permit access through a non-linear hypertext structure. On the Internet, access to the hypertext structure through a web browser is natural and adds value to document archives. The main topic of this paper is to identify and use hypertext structures, in order to utilize the Internet to its full potential in accessing paper archives.

In [10][11][13][14] the automatic construction of hypertext from the content of a document is considered. The structuring of the final hypertext document is solely based on the analysis of the text. However, figures often play a major role in communicating the purpose and message of the document. They have been included by the author to aid the understanding of the most difficult and important parts. Pictures should therefore also be included in hypertext construction, both when linking to other documents, as well as within the context of one document. Using a text based system in the latter case leads, for a figure intensive document, to a continuous flipping of pages to find references from and to the figure content. We therefore focus on links between the text and the content of figures. These dominate the navigational aids for reading such documents. In addition to these, text based structures can be added.

For vector based graphics, impressive work in analyzing figure content and relating it to other pieces of information is presented in [1]. No results on the hypertext aspects of the system are presented, neither are Inter-

Correspondence to: M. Worring

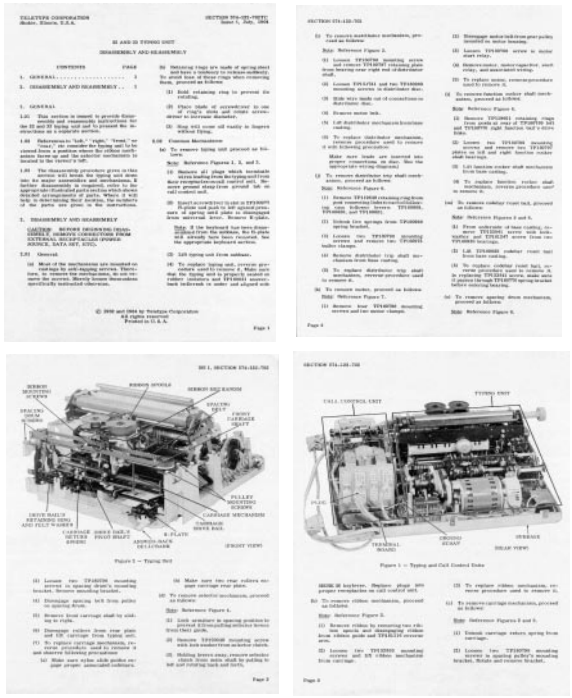


Fig. 1. Four example pages of the document we examine (they are taken from a 1962 teletypewriter manual). They fall into three classes: a title page, a pure text page, and two text/figure pages

net issues considered. The same holds for the methods reported in [15] relating the content of photographs in newspapers with their caption for indexing purposes. We focus on manuals containing text and graphical pictures or photographs with textual annotation in the picture, where the document is available as a bitmap only.

We have developed a system for hypertext construction, aspects of which have been presented at conferences [18][19] [20]. This paper extends that work. It is organized as follows. In Sect. 2 we consider document structures. From there, we present methods for deriving those document structures from a scanned paper document in Sect. 3. In Sect. 4 results of applying the proposed methods to a 1962 teletypewriter manual are presented. Example pages of this document are shown in Fig. 1. Section 5 considers Internet access and presentation.

2 Document models

2.1 Existing models

When studying document analysis, it is appropriate to consider the model for the inverse process: document creation. Models for creating paper documents are the Office Document Architecture (ODA) [3], and Standard Generalized Markup Language (SGML) [17]. They make an explicit distinction between the layout structure of the document as it follows the physical limitations of paper and printer, and the semantic or logical structure following the intent of the author. Without loss of generality, the terminology from ODA is used.

In [3], the layout or geometric structure is defined as:

geometric structure: the hierarchy of objects resulting from decomposing the document, based on the way it is presented on paper.

This tree-shaped structure captures what, where, and how things are printed. At the level of the geometric structure the document is determined by font type, font size, number of columns, inclusion of figures, etc. As concerns the document analysis, the characteristics of the geometric structure have a direct relation to geometric properties that can be measured from the image resulting from scanning.

The second structure is the:

logical structure: the hierarchy of objects resulting from decomposing the document into the parts intended by the author.

In this structure one finds the decomposition into sections and paragraphs, the meaning of text in bold to indicate section headers or to emphasize keyphrases in the text, and the grouping of figures with their captions. This decomposition cannot be derived directly from the image, but requires external knowledge about the purpose of the document structure.

The leaves in the geometric and logical structure coincide and are single pieces of coherent data called *basic objects*. Depending on whether the geometric or logical characteristics of the objects are used, they are called geometric and logical basic objects respectively. As they form a natural unity, they are the atomic data parts. [1]

The Dexter model [7] is in use for creating *hypertext structure*. The *atomic components* and *composite components* are, like in the ODA structures, the unbreakable data items and their hierarchical composition. They form the nodes in the hypertext. *Anchors* are used to address locations within an atomic component. They are considered at this level to assure independence from the physical representation of the components. Finally, the model defines *links* as the means to capture relations. A unidirectional link l consists of specifications of its starting point $s(l)$ and its ending point $e(l)$. In the Dexter model the specification can specify nodes or anchors within the nodes. A bidirectional link is captured in two uni-directional links pointing in opposite directions. For a set of links L, $s(L)$ and $e(L)$ denote the set of all starting and ending points respectively of the links in L. Further, $d^+(n)$ and $d^-(n)$, denote the number of incoming and outgoing links of a node (or anchor) n.

2.2 A structured hyperdocument

Having defined the structure of a document and the characteristics of a hypertext it is now time to integrate the

[1] There is one case where a geometric and geometric basic object do not coincide. A logical object (say a paragraph) can start at the end of a page and be continued on the next page, while a geometric basic object is confined to one page. This has no impact on the issues raised in this paper.

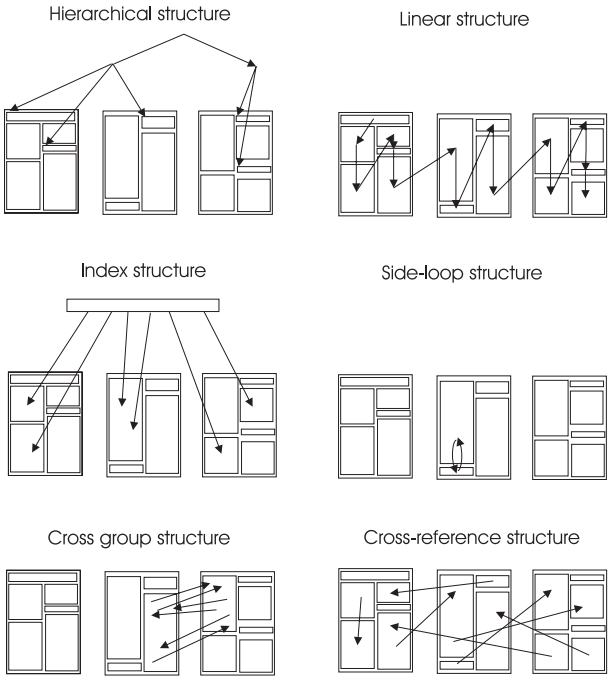


Fig. 2. Illustration of the six different hypertext structures

two definitions to arrive at a *structured hyperdocument* representation. Formally, we define a hyperdocument as the pair $H = \{N, S\}$. Here N is the set of nodes in the hyperdocument. The *nodes* are restricted to the pieces of coherent data and are equivalent to the basic components in the hypertext. Composite components are treated in a different way, to be defined later. The set S is the structure set. This is not commonly encountered in hypertext, but bears similarity to the use of collections in [6]. It can also be viewed as an extension of the hypertext structures in [2]. An element $s \in S$ can be written as $\{N_s, L_s\}$. The set L_s is a structure specific set of links providing some relational structure on the set of nodes N_s, their grouping within a structure, and/or their anchors. Each s captures a specific structure in the hyperdocument.

Now let us make the definitions more specific. Although the scanned document is one image and hence one media item, the inherent structure of the document makes it more convenient to use the classification according to geometric content. Example class labels are {*text, figure, photograph, table, horizontal line*}. As, in general, paragraphs form the smallest self-contained entities, they form the *nodes*. The *anchors* are pieces of information within the content of nodes. Useful anchors are objects or labels in a figure, objects in a photograph, table entries in a table, and keywords or key phrases in the text.

We distinguish six different types of structure in S. Each of these impose different constraints on N_s and L_s. The structures are exemplified in Fig. 2.

hierarchical structure: here N_s is a set with a tree structure. L_s is the set of links required for accessing the nodes pointing to the different children.

Clearly this corresponds to composites in a hypertext. In a hyperdocument, the prime hierarchical structure of importance is the logical structure. The geometric structure is only relevant in the document analysis phase.

linear structure: in this structure the elements of L_s lead to each of the elements in N_s, without ever reaching the same node twice.

The most important linear structure is the reading order. It corresponds to a depth-first ordering of the text blocks in the logical structure [16]. It is considered at this level and not as part of the hierarchical structure as one can define different linear orderings of the nodes in the hierarchy. Depth-first is only an example of this.

Other linear structures are the orderings by type (e.g., figures and tables).

index structure: here $s(L_s)$ contains the single element n with $d^+(n) \geq 1$ and $d^-(n) = 0$. N_s is equivalent to $e(L_s)$.

As the name suggests this is in fact the common index to a document. For example, an index based on all labels in figures, or important keywords in the text.

cross-group structure: here $s(L_s) \cap e(L_s) = \emptyset$. The set N_s is formed by the union of $s(L_s)$ and $e(L_s)$.

This class of links is important in our application as these connect labels in a figure with the text. One other cross-group structure set relates the collection of figures and the whole text. It identifies the parts in the text where certain figures are described. The sets $s(L_s)$ and $e(L_s)$ are called each other's *scopes*. So, the pieces of text related to one specific figure form the scope of that figure. This is important later.

side-loop structure: $N_s = \{n_1, n_2\}$ and $L_s = \{l_1, l_2\}$ with $s(l_1) = e(l_2) = n_1, s(l_2) = e(l_1) = n_2$,and $d^+(n_2) = d^-(n_2) = 1$.

The most prominent examples of side-loops are footnotes, explanations, asides, and appendices. They should be readable upon request, but should not influence the reading order in the document.

cross-reference structure: here L_s is the set of relations not part of any of the above structures. The set of nodes is given by $N_s = s(L_s) \cup e(L_s)$.

This last class mostly contains semantic connections. Note, that we use the same name for the class covering links not covered by other structures as does [2]. However, in their paper they only consider the hierarchical and linear structure. The index, cross-group, and side-loop structure would all fall in their cross-reference structure. Hence, the discriminatory power of our set of classes is higher.

2.3 Document presentation models

If a hypertext system conforms to the Dexter model [7], each of the components has *presentation specifications*

stating in an abstract way how the component should be presented to the user. Among other information it should state the logical channel [8], an abstraction of a physical channel capable of presenting a certain type of media. In [8] an additional important concept is the *link context*. The link context defines which part of the currently displayed document is replaced or otherwise affected by following the link. Any visible components which are not affected remain on the screen unaltered.

For the presentation of the structures derived in the previous section, high level presentation specifications are needed. In fact the type of a structure in S can be considered such a high level specification. Each of these should be compiled into a set of low level presentation specifications and link contexts on the items constituting the structure.

Any of the hypertext structures needs at least two channels. One is for displaying the content of the nodes. The other presents the navigation controls to the user. These navigation controls provide access to the nodes and links in the structure, regardless of their content. For paper manuals two content channels are used, one for text and one for figures. In fact the channel for a figure is composed of two subchannels for displaying the figure itself and its caption.

For *linear structure,* the navigation channels should provide means to go forward and backward in the structure, relative to the current position. The only node one can jump to directly is the first node in the structure.

When a *hierarchical structure* is placed in the navigation channel some visual representation for the different levels is required. Each node in the visual representation has a direct correspondence to a node in the structure. Hence they can be used as the source of a link which one can follow to the related node. For text, the hierarchical structure is directly related to the chapter, section, subsection structure of the text. Hence, a good visualization of a node is to use the text of the corresponding header, augmented with color or indentation to indicate different levels.

For an *index structure,* the visualization of the node should also have a direct connection to the content of the node. For an index on labels in the figures, one could either show the piece of the figure within the bounding box of the label, or the ASCII text corresponding to it. In the above, content and navigation are clearly separated and link context defines that the currently displayed node in the content channel should only be replaced when it is not the component the activated link points to.

For the *side-loop* structure, link context also plays an important role. When a side-loop is activated the content of the displayed node should be replaced by the content of the endpoint of the link. Navigation control should be adjusted such that the only navigational possibility is to go back to the place where the node was activated. A pop-up window is a typical example implementation of this behavior.

The link context for a *cross-group structure* defines that the displayed component should only be replaced when it is not the component the activated link is pointing to. When separate channels for the two components forming the cross-group are used, subsequent activation of links in either direction assures that both components are displayed and remain displayed as long as links within the cross-group are used. The navigation channels should display the different ways of leaving the cross-group structure.

Cross-reference structure can be so varied that we do not make any particular remarks about their presentation.

The abstract presentation specification allows one to specify the presentation independent of the presentation platform, in Sect. 5 this will be described for Internet access to the hyperdocument.

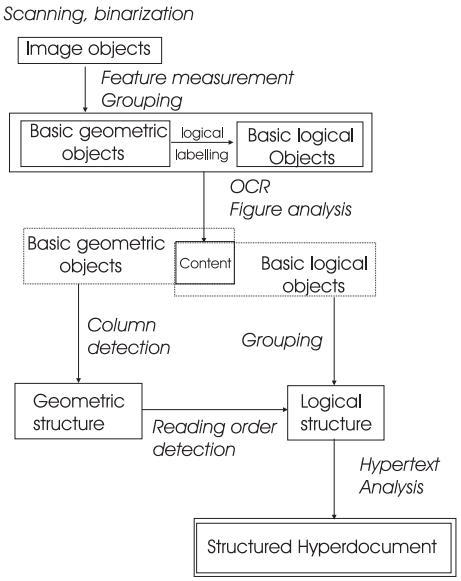


Fig. 3. Overview of the steps involved in converting a paper document into a structured hyperdocument

3 From paper to structured document

The document model we have defined forms the basis for algorithms to convert paper documents into structured hyperdocuments. These algorithms require processing phases addressing various aspects of the document structures and content. A number of processing steps are distinguished based on the different representation levels. They are exemplified in Fig. 3. We describe the methods we have developed. They are inspired by the case study we have performed, but are general enough to be tailored easily for use in other applications.

3.1 From paper to image objects

Pages are scanned in grey values and then segmented using the Isodata thresholding technique [5]. The binary images are used in the analysis. Grey valued originals are used for display purposes. The image resolution is the base resolution. For speeding up some of the processing, the original image can be reduced to other resolutions. Our system allows for the use of different resolutions in

different parts of the image. These are all mapped to the common document reference coordinate system.

Our low level object representation is the:

image object: a binary 8-connected object with its geometric features.

We believe that using these as the basis representation, rather than run-length coded images as often used in the literature (e.g.,[16]), offers more flexibility in defining features. For efficiency reasons, most of the processing is performed on the bounding boxes of the connected components. This is a convenient representation, although it does require that the document exhibits no skew. If skew is present, the document should be deskewed in a preprocessing stage.

For each image object, a set of geometric features is defined. In our case width, height, and aspect ratio are sufficient.

3.2 From image objects to basic geometric objects

The first step in finding the basic geometric objects is classifying the image object into a set of geometric object classes. The result of which is called a segment. Here a decision tree method is used, similar to [16]. In our case the class labels are {*text, figure, horizontal line, vertical line, noise*}. As the classified image objects in general do not correspond to the basic objects as defined in ODA, the notion of segment is in fact more general:

segment: a set of image objects with the same geometric label.

Features of a segment are either segment specific or the minimum, maximum, average, or modal value of the features of the image objects in the group. The values $x^i_{min}, x^i_{max}, y^i_{min}, y^i_{max}$ define the bounding box for segment i. Characteristics of segments are considered at three levels: features of the individual segments, relations between pairs of segments, and characteristics based on the whole set of segments. Each of these is now considered.

The individual characteristics we use are the width and height, as well as features based on the position of a segment on the page. Consider a segment s which is intersected by the middle line which vertically divides the page into two equal sides. Now, let d_l denote the distance from the left side of the segment to middle line and similarly d_r for the right side. Then the centrality c of the segment is defined as:

$$c(s) = \frac{d_r - d_l}{d_r + d_l}$$

The measure is 0.0 when the segment is perfectly centered and ± 1.0 when it is not centered at all, occurring when d_r or d_l is equal to zero respectively.

For vertical position the binary predicates

above_middle and *below_middle* are used. These are true when the whole segment is above or below the vertical middle of the page respectively.

The general form of a relation between two ordered segments s_1, s_2 is given by $predicate_{param}(s_1, s_2)$.

Relations between pairs of segments can either be symmetric or asymmetric. The asymmetric relation

is_contained_within is true when the bounding box of s_1 encloses s_2. A more complex parameterized predicate is

$in\_margin_{sx,sy}$. This constructs a virtual bounding box by scaling $s_1$with a factor sx in the horizontal and sy in the vertical direction. It then checks whether there is an intersection with the bounding box of s_2. The final asymmetric predicate *total_vertical_overlap* is true when the y-interval of the bounding box of $s_1$encloses the y-interval of s_2.

The simplest symmetric relation is *intersects*, checking the intersection of the bounding boxes of s_1 and s_2. More complicated are the predicates

$horizontal\_group_{sh,ov}(s_1, s_2)$ and

$vertical\_group_{sw,oh}(s_1, s_2)$. These are used to check whether segments belong to the same horizontal or vertical group respectively. The parameters sh and ov are bounds on the following two scale independent measures: relative vertical overlap O_v and relative horizontal spacing S_h of segments s_i and s_j. They are exemplified in Fig. 4.

$$S_h(s_i, s_j) = \frac{max(x^i_{min}, x^j_{min}) - min(x^i_{max}, x^j_{max})}{max(y^i_{max}, y^j_{max}) - min(y^i_{min}, y^j_{min})}$$

$$O_v(s_i, s_j) = 1 - \frac{|y^i_{min} - y^j_{min}| + |y^i_{max} - y^j_{max}|}{max(y^i_{max}, y^j_{max}) - min(y^i_{min}, y^j_{min})}$$

In similar way sw and oh are bounds on:

$$S_w(s_i, s_j) = \frac{max(y^i_{min}, y^j_{min}) - min(y^i_{max}, y^j_{max})}{\frac{1}{2}\,(\overline{h_i} + \overline{h_j})}$$

$$O_h(s_i, s_j) = 1 - \frac{|x^i_{min} - x^j_{min}| + |x^i_{max} - x^j_{max}|}{max(x^i_{max}, x^j_{max}) - min(x^i_{min}, x^j_{min})}$$

A difference between horizontal and vertical groups is that for the latter the average height $\overline{h}$ of the objects in the segments is used. This is particularly appropriate for grouping text fragments, assuring that different font sizes are not mixed.

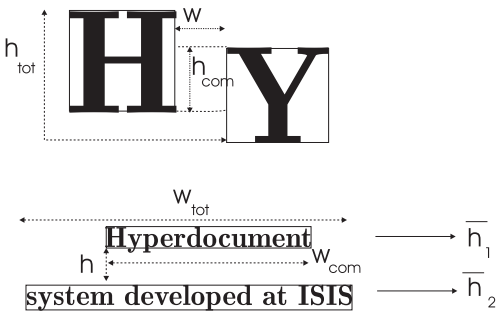


Fig. 4. Graphical illustration of the measures $S_h = w/h_{tot}$, $O_v = h_{com}/h_{tot}$, $S_v = h/\frac{1}{2}(\overline{h_1} + \overline{h_2})$, $O_h = w_{com}/w_{tot}$, where h_1 and h_2 denote the average height in segment 1 and 2 respectively

The final class of predicates is testing global properties of a segment. These predicates are defined with respect to all other segments. We use

top_most, and *bottom_most* with definitions as expected.

In the following the notation *predicate(type)* denotes all segments of the given geometric type satisfying the predicate. Similarly *predicate(type1,type2)* denotes all pairs (s_1, s_2) satisfying the predicate. These are used for selecting segments. On selected segments the operations *delete,* and *change_type(new_type)* can be applied. For pairs these can also be applied, with the difference that they can be applied either to the first element of the pair, the second element of the pair, or to both. Furthermore, the operation *join* combines the two elements of a pair into one new segment. When two segments are joined the new segment receives the objects of both segments. The bounding box is adjusted accordingly.

The simple, yet powerful, method of selection and action forms the basis for further document analysis. These methods are used in deriving the basic geometric objects.

The image objects are usually the smallest basic items in the image which can be given an interpretation in document terms, like characters and parts of figures. In general they do not correspond to the basic components required in the geometric structure which are the single paragraphs and complete figures. The text fragments are grouped into text lines by iteratively using a *horizontal_group(text,text)* selection, followed by a *join* operation. The same holds for going from lines to paragraphs using *vertical_group(text,text).* As the figures do not have boxes, the elements of any pair satisfying *intersects(figure,any_type)* are iteratively *joined.* This yields the proper bounding box for the figure.

For our documents the above operations generate all the basic geometric objects.

3.3 From geometric objects to geometric structure

For multi-column documents, the geometric structure is mostly concerned with column structure. For two column documents, segments are classified into {*centered, left_column, right_column*}.

The column of a segment s is computed by considering whether it is intersected by the middle line. If not, the column is obvious, otherwise the following is used:

$$column_\xi(s) = \begin{cases} left\_column & c(s) < -\xi \\ centered & -\xi \leq c(s) \leq \xi \\ right\_column & c(s) > \xi \end{cases}$$

where ξ is the parameter for deciding when an element is considered to be centered and c is centrality.

A more elaborate approach for column detection for newspaper-like documents is considered in [16]. This method is not suited for centered segments in the document, as it depends on the alignment of the bounding boxes in the vertical direction.

3.4 From geometric to logical objects

The basic objects now have a geometric label. The next step is the logical labeling of the text items. This requires making explicit the knowledge one has about the document class. We encode the knowledge into a set of predicates and actions as defined in Sect. 3.2. The following knowledge about the class of documents is used: there are one or two headers on the top of the page, page numbers are at the bottom of the page, title pages have both a title and footer above and below the textbody respectively. For our document class this leads to the classification strategy exemplified in the following table.

predicate	**new type**
top_most(text)	*header*
vertical_overlap(text,header)	*header*
bottom_most(text)	*page_number*
in_margin(figure,text)	*caption*
segment_centered(text)∧ *above_middle(text)*	*title*
segment_centered(text)∧ *below_middle(text)*	*footer*

Segment_centered(text) is the same as deciding whether a column is centered (see Sect. 3.3). The algorithm is suited for the title page, the pure textual pages, and the combined text/figure pages present in our documents.

Clearly the set of predicates used above does not generalize directly to other document classes and should be redefined when needed. For documents with a simple layout finding the right set of predicates is fairly easy.

An alternative to using a predicate based system is to use a hierarchical set of templates where each level in the hierarchy is a refinement of the higher level [4].

Our system does not include backtracking, as does for example [4], and hence expects that geometric structure has been identified correctly. If desired, our system allows for interactive correction of the geometric or logical objects. In fact, users can select blocks and these selected blocks are equivalent to blocks for which a predicate yields a positive outcome. Hence, any of the mentioned operations can be applied to them.

3.5 From basic objects to content

Up to this point we only considered the objects in the document and their structure, we have not looked at their content. A distinction is made between the content of textual objects like paragraphs and captions, and the content of figures.

Textual parts are processed using a commercial OCR-package (OmniPage) to retrieve the ASCII text. The ASCII text of the textual objects is in raw format. For further processing, the text is tokenized where we recognize the token types {*word, numeral, punctuation, white-space, start_of_line, end-of-line*}.

To extract the content of figures in a hypertext context, our focus is on labels in the figure. One can make a distinction between four geometric classes of labels [20] exemplified in Fig. 5:

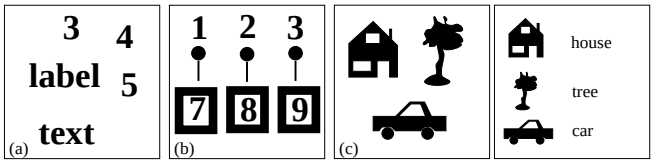


Fig. 5a–d. Overview of the different types of labels we distinguish. **a** plain alphanumeric labels, **b** alphanumeric template labels, **c** icon labels, **d** legend labels

- *plain alphanumeric labels* : facsimile of their corresponding ASCII string.
- *alphanumeric template labels* : text strings derived from a template where the variable part is a plain alphanumeric label and the fixed part is some visual shape.
- *icon labels* : non-alphanumeric labels distinguished by their shape alone.
- *legend labels* : icon labels with an associated textual definition.

In this paper we consider textual labels only; methods for deriving the other classes are based on template matching and are described in [20].

The content of figures is analyzed at full resolution to avoid losing important details. The observation is made that in general characters in the image have approximately the same height with the exception of dashes, parentheses, and punctuation. In many cases one can assume that the labels have a minimum number of characters n_c. When labels are short descriptions rather than plain numbers or single character identifiers, one can safely assume that any label that seems to be composed of two or less characters ($n_c = 3$) is in fact not a label.

To find candidate characters, the modal height h is calculated. Elements in the range $[(1-\alpha)h,(1+\alpha)h]$, are kept as candidate characters for the labels. To account for noise and taking the characteristics of characters into account a value of $\alpha = 0.15$ is used. The remaining candidates are joined using the predicate *horizontal_group*. At this point we expect to have found all characters except for the special characters '-', '(' and ')'. To add them we join all pairs of segments for which the predicate *total_vertical_overlap* is true. After this step, small labels (width $< n_c h$) are discarded. Note, that the selection of small labels is with respect to height h, rather than width. This is because subsequent lines in a label are well separated, but subsequent characters are often connected, hence yielding unreliable estimates of average character width. Remaining label lines are then joined using *vertical_group* to find complete multi-line labels.

The resulting segments are sent to the OCR package. Again the raw text is tokenized for use in further analysis.

The output of the OCR system is used to verify whether the label is genuine. To be specific, labels are discarded if the result of OCR contains more than 50% rejects, when less than n_c alphanumeric characters are present in the label, or when less than 50% of the characters are alphanumeric. The latter is based on the observation that OCR systems tend to classify figure fragments, which in our case are mostly parts of the photographs into non-alphanumeric characters.

For line graphics a more sophisticated approach is required. Here, labels which are non-genuine are likely to contain pieces of vertical and horizontal lines which will often be recognized by the OCR system as valid characters (among others 'I', '-', 'l'). Then verification is only possible by lexical or semantic methods.

Line graphics have one further consequence in our system. The use of modal height for classification of candidate characters fails when recurring patterns are present in the image like a set of dashed lines. These should be detected beforehand, for example with the methods presented in [9].

3.6 From layout and content to logical structure

The final step before deriving the hypertext structure is to find the logical structure, hierarchically grouping segments on the basis of their logical labels. We first consider the pieces of text which are not part of the main body. In our case the caption is grouped with the corresponding figure. All other logical segments have a meaning of their own and no direct relation to other objects. Only finding the logical structure of the text body remains. This can either be done by starting with the layout information [16] and then applying rules capturing knowledge of layout conventions, or on the basis of the content of the text. Here the latter approach is followed.

We start off by linearly ordering the segments. As the text is in two-column format, this is just page by page and column by column. Within each column, ordering is defined by vertical position. On top of this linear ordering comes the hierarchical structure of chapters and sections. This is found by searching the text for section indicators. For our documents the following regular expressions are used, where * means zero or more occurrences and $^+$means at least one occurrence.

- *chapter*: <start-of-line> <numeral> <.> <word>$^+$ <end-of-line>
- *section*: <start-of-line> <numeral><.><numeral> <word>$^+$<end-of-line>

Intervening tokens with classification white-space are accepted. Clearly the subsequent tokens of class word found in these expressions denote the headings of the different chapters and sections. As the search is based on the output of the OCR system, errors might occur. A verification step is performed, checking whether the sequence defined by the subsequent chapters and sections is increasing in unit steps. If this is not the case, either a section indicator is missed or the OCR of the section number is not correct.

Whereas text labels in a figure all have the geometric classification text, they can in fact also have a logical classification indicating their meaning. As in the logical labeling of basic objects this is domain specific and requires knowledge about the content of the figures.

In our case three logical classes can be distinguished. First, figures can have a label of class *title*. Second, there are labels of class *note* which provide contextual information about the figure. All other labels in our application are of class *name*, each of them naming a (part of) an object in the figure.

The logical classification is performed by considering the tokenized text of each label identified in the figure. Titles in our application correspond to the different views one can have and these can be identified by looking for the string "view" enclosed in parentheses. The notes are explicitly identified by the word "note". All other labels are classified as name.

For any label of class *name,* a remark in parentheses in a label is detected as such and removed from the textual content of the label. They provide side-information only and not information directly related to what the label is pointing at.

More complex classifications of figure content and structure requires elaborate domain knowledge [1].

3.7 From logical structure to hypertext

Having found the logical structure and the content of the basic objects in the document, it is time to find the hyperdocument structures identified in Sect. 2.2.

The logical structure of the document provides the *hierarchical structure* of the hyperdocument. It also provides information to derive the *linear structures* required for the reading order and accessing the figures in the document. Computing a standard *index structure* based on the labels in the figure is also trivial. An index of important keywords in the text can be found automatically based on the statistics of occurrence in the text [12]. Note that it might be necessary to consult a list of synonyms here before performing the statistics.

The *cross-group structure* between the set of figures and the text can only be found when there is some explicit way of reference to figures. In our application these references can be found by searching for the patterns:

- "Note"<:> "Reference Figure"<numeral>
- "Note"<:>"Reference Figures"<<numeral>,>$^{+}$ "and" <numeral>

Other common ways of referring to figures are, "see figure <numeral>", "as shown in figure <numeral>", "fig. <numeral> illustrates", "(fig.<numeral>)", etc.

In all cases the numerals are restricted to fall within the range of number of figures detected. When references to more than one figure at the time are made, the sequence of numerals should further be properly increasing, another verification of the OCR-process. The values of the numerals are used to derive the links of the cross-group structure that relates the text with the set of figures. This also provides us with the scope of each figure in the text as defined in Sect.2.2

To find the *cross-group structure* for a specific figure and its scope in the text, the tokenized text of each label in the figure is searched for in the corresponding text. The scope serves to limit the keyword search to the proper part of the text. This not only limits the number of superfluous links, but it also yields the proper links when the same label is used in different figures.

A number of characteristics of the labels have to be taken into account. First, the labels in the figure consist of multiple words. Second, the text in both the label and the associated text is the result of OCR and hence contains errors. As a consequence, keywords may not match exactly. Finally, the text of the labels does not necessarily appear in the same order and with the exact words in the text.

The solution chosen is to first match each individual token of class word in the label to each token of class word in the text within the given scope. Matching of the two word tokens is done by finding the largest common substring. The match value, indicating the quality of the match, is equal to the number of characters in the substring, divided by the number of characters in the larger word. The result for each label is a list of matching pairs for which the matching value is sufficiently large. To arrive at matches of the complete multi-word labels, the individual matches are combined whenever they are found close to each other in the body of the text. In our case two matches are considered close, when the number of intervening tokens of type *word* is smaller than or equal to 1. This also finds labels in which the words are not in exactly the same order.

As verification of the picture label linking process one could use the observation that each label is expected to have at least one relation with an item in the text. So, if no such connection is found, the matching procedure should be adjusted. This requires either consulting a list of synonyms or a thesaurus with semantic relations.

No *side-loop structures* are present in our document set. Footnotes, in general, are relatively easy to detect when the OCR recognizes superscripts in the text correctly. This is the most frequently encountered convention for indicating footnotes. We then need to identify whether the superscript is part of a textual part of a document or a formula. Naturally, footnotes also have to be incorporated in the classification of logical basic objects.

As no semantic linking is considered there is no remaining *cross-reference structure*.

4 Results

The above-defined techniques have been applied in a small scale case study. It consists of an 11 page section of a teletypewriter manual. There are three different types of pages in the document. One title page, and subsequently either text pages or pages containing one figure. In total 8 figures are present. See Fig. 1 for examples. In total they contain 105 textual labels, eight of which are *titles* and one is a *note*. The text consists of 2 sections and 7 subsections.

The pages were scanned at 300 dpi. For the layout and logical analysis, one page of each class is used for optimizing the parameters that were not fixed beforehand like sh, ov, sw, and oh used in grouping of segments and

ξ used in defining columns. All other parameters were fixed at the values indicated earlier.

A reduction factor of 4 in both dimensions was used, effectively analyzing at 75 dpi. As we have made a clear decomposition of the different steps involved in the analysis, each of the steps can be optimized individually. Parameters were therefore easily tuned.

The layout and logical analysis of the 11 pages is almost perfect. One error is made; a section indicator somewhat set apart from the corresponding column is not joined properly and missed in the column detection. Due to our scale independent measures, solving this would lead to the merging of the two columns.

For optimizing the parameters used in detection of text labels in the figure, the selected figure page is used. The figure selected contains both parentheses and dashes in the textual labels. In total 116 labels are found. There is one case where two individual labels are joined and one case where one label is split into two. There are no false-negatives and 8 false positives, 6 of which are rejected after OCR.

Finding the hyperstructures is performed on the complete 11-page section. Parameters are as indicated in Sect. 3.7.

The first structure we consider is the hierarchical structure. All but one entry of the content page is found correctly. The section indicator 2.03 came out as t.03 after OCR. The error is detected in the verification step, so backtracking could have been performed. Furthermore, this indicator is in fact the one that was wrongly classified in the layout phase.

The *cross-group structure* between the set of figures and the text i.e., all references to figures, consists of 32 links. These are all found correctly.

In the logical classification of the content of figures identifying the titles and notes, no errors are made by the system.

The remaining 99 labels are used for finding the *cross-group structures* between textual labels in a figure and the text. A total of 221 links is found. None of the remaining non-genuine labels are linked. There are 14 genuine labels not linked to the text for various reasons. Most important are OCR errors, especially introduced spaces where none are present. This is probably a consequence of the particular font used in this 1962 document. Finally, there are labels which as such do not appear in the text and can hence only be found by a semantic analysis of the text, or by use of a thesaurus.

5 Internet access

When the document is accessed via the Internet, the user should be given access to the content and the structure of the document in conformance with the presentation model of Sect. 2.3. The low level presentation specifications have to be realized using Internet specific techniques. For this purpose, HTML is used as the basic hyperdocument description.

The content channels are implemented as frames. For the content of the text channel a choice has to be made. One can show the original picture of the text and show the anchors in the overlay. The advantage of this is that the original text layout is preserved, it does however have some disadvantages. Most importantly, one needs an OCR system capable of indicating the exact position of each word detected as otherwise no anchors can be overlayed. Only recently have OCR developer's kits been released which indeed have this functionality. It is, furthermore, difficult to remove irrelevant layout information like page numbers and section numbers, as they have to be cut out of the picture. Ideally the text is the (error-free) output of the OCR system with formatting preserved and irrelevant information removed. As we, at the time of writing, did not have access to a proper OCR developer's kit we show the raw unformatted output of the OCR system.

The separate navigation channels for figures and text are also implemented using frames. The access to the linear ordering of figures is through the simple previous and next links. Access to the linear structure of the text is via a scrollbar as this is the most familiar way for most users. Note that a scrollbar allows arbitrary access points in the text, which should not be the case for linear structure. However, a hyperlinked index or content page makes the use of arbitrary access obsolete. In our presentation, the content page is in the true navigation channel, the hierarchical structure is visualized through indentation. An overview of the presentation is shown in Fig. 6.

To create the HTML representation, our system takes the different hypertext structures encountered in the document and converts them to a HTML document description. It is not possible in HTML to describe a bidirectional relation between the anchors in the figures and the anchors in the text, as needed for the cross-group structure. We do encode the information on anchors in the figure and their links to anchors in the text in the HTML document, but this cannot be handled by the WWW server directly. Therefore, our hyperdocument server has a cross-group server in addition. The cross-group server filters the link activations the user made. When the user activates a link in the current cross-group structure, only a presentation command is given to emphasize the label in the figure. If on the other hand, the link points out of the current cross-group, communication with the WWW server takes places to bring up the new figure. Navigation controls are automatically adjusted. Clicking a label in the figure induces a scrolling action on the associated text to the first occurrence of this label in the text. As more than one occurrence can be found, a Java applet provides access to the linear ordering of the anchors at the end of the link.

As concerns the speed of access, note that the largest amount of time is spent in initially retrieving the document and when a link is followed outside of the current cross-group structure. In both cases an image and possibly large amounts of text have to be transferred and communication with the WWW server is needed. Interacting with the anchors in a cross-group structure is cheap as the only information transferred is information on the links which are activated and presentation commands,

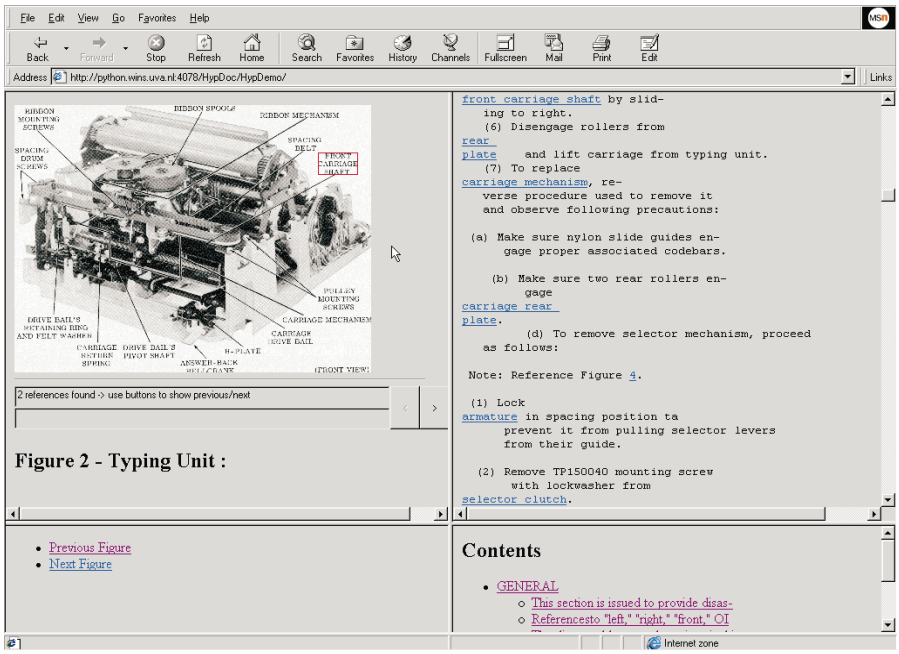


Fig. 6. Overview of the presentation of the document to the user

content being already present. The more links present in a cross-group structure, the more economical the system becomes. Time in the cross-group will in general increase. The use of structured hyperdocuments is also more efficient than showing the facsimile of the pages in the document. Only the figures in the document have to be transferred as bitmaps, text can be transferred in its ASCII form. The overhead resulting from having to add structural information about the document is negligible.

The architecture of the system is shown in Fig. 7. The core of the architecture is the hyperdocument server consisting of the cross-group server and the WWW-server. The server controls the files containing the generated HTML-based document description and carries out communication with the web browser, being the user interface.

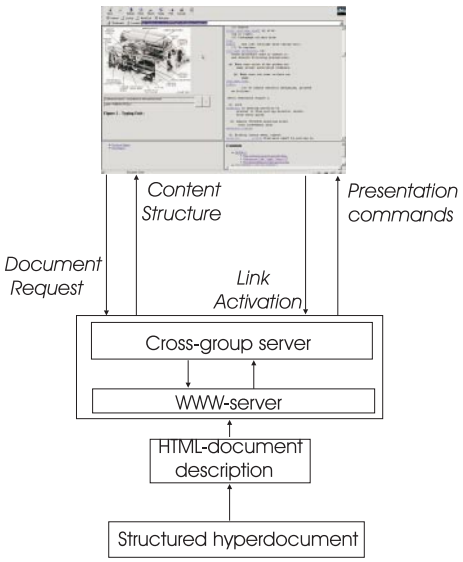


Fig. 7. Overview of the architecture of the hyperdocument server. It shows how a structured hyperdocument is translated into an Internet accessible presentation. Also the special importance of the cross-group server is apparent

6 Conclusion

Internet access to paper documents should be done on the basis of a proper document representation. The *structured hyperdocument* introduced in this paper integrates definitions from creation models for documents (ODA) and hypertext (Dexter/AHM). It consists of six types of structure descriptions considering different aspects of the document. For document image analysis additional representation levels are the image objects and segments used as an intermediate between paper and the basic geometric objects in the document structures.

The different representation levels provide a solid base on which to develop the required document image analysis tools as well as the hypertext analysis tools. The resulting decomposition of the complex problem allows for optimization of the individual steps. Methods for each analysis step have been proposed. A pure bottom-up approach is followed. The document image analysis goes through different processing phases resulting in the content and logical structure of the document. This forms the starting point for identification of the hypertext structures.

The proposed methods have been used in a case study. The performance is sufficient to identify most of the structures automatically. Of importance is the large number of links found between text and figures. They are not complete as semantic links are not considered. However, they relieve the document creator from the tedious task of identifying all links. He/she can now concentrate on the small set of links which are difficult to identify automatically.

The structures in the document are also employed in the presentation of the document to the user. The resulting access paths and visual representation of the document now guides the user to quickly find the information searched for. It also leads to improved performance of the document server as only relevant information is transferred. The structuring of both the text and the pictures in the document also provides a base for further research into indexing a collection of documents and providing links between them based on its full content rather than limiting it to text as in [10] [11][13][14].

In the current work a modular bottom-up approach is followed. Although verification steps are considered, no backtracking is performed. The user is put in charge of dealing with detected inconsistencies. To improve on this, a connection to a reasoning system is urged for. It requires an extensive knowledge base, making explicit which document structures are present in the document collection, and what their characteristics are. The available knowledge should then be distributed over the different modules.

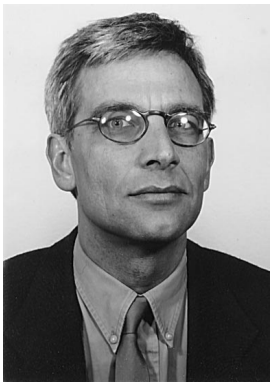
Our future research will built upon the modules presented here to analyze large collections of manuals designed for training and maintenance. A knowledge base for this purpose will be created. As this research will be done in conjunction with educational experts it also allows evaluation of the effectiveness of the interface classes presented here, in comparison to traditional methods of presentation.

Acknowledgements. We greatly appreciate the work of Edo Poll and Frank Seinstra who developed most of the hyperdocument server. Furthermore, we would like to thank the anonymous reviewers for their detailed comments.

References

1. L.S. Baum, J.H. Boose, and R.J. Kelley. Graphics recognition for a large-scale airplane information system. In Proceedings of second IAPR workshop on Graphics Recognition, pages 62–69, 1997
2. R. A. Botafogo, E. Rivlin, and B. Shneiderman. Structural analysis of hypertext: Identifying hierarchies and useful metrics. ACM Transactions on Information Systems, 10(2):142–180, 1992
3. I.R. Campbell-Grant. Introducing ODA. Computer Standards & Interfaces, 11:149–157, 1991
4. A. Dengel. From paper to office document standard representation. IEEE Computer, 25(7):63–67, 1992
5. R.O. Duda and P.E. Hart. Pattern classification and scene analysis. Wiley, 1973
6. F. Garzotto, L. Mainetti, and P. Paolini. Adding multimedia collections to the Dexter model. In Proceedings of the European Conference on HyperText, pages 70–80, 1994
7. F. Halasz and M. Schwartz. The Dexter hypertext reference model. Communications of the ACM, 37(2):30–39, 1994
8. L. Hardman, D.C.A. Bulterman, and G. v. Rossum. The Amsterdam hypermedia model: Adding time and context to the Dexter model. Communications of the ACM, 37(2):50–62, 1994
9. R. Kasturi, S.T. Bow, W. El-Masri, J. Shah, J.R. Gattiker, and U.B. Mokate. A system for interpretation of line drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10):978–992, 1990
10. G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. IEEE computer, 25(7):10–22, 1992
11. C. Pearce and E. Miller. The TELLTALE dynamic hypertext environment: approaches to scalability. In Intelligent hypertext: advanced techniques for the world wide web, volume 1326 of Lecture Notes in Computer Science. Berlin, Heidelberg, New York: Springer 1997
12. G. Salton. Another look at automatic text-retrieval systems. Communications of the ACM, 29(7):648–656, 1986
13. G. Salton, J. Allan, and C. Buckley. Automatic structuring and retrieval of large text files. Communications of the ACM, 37(2), 1994
14. S. Satoh, A. Takasu, and E. Katsura. A collaborative approach supporting method between document processing and hypertext construction. In Second International Conference on Document Analysis and Recognition, Tsukuba Science City, Japan, pages 533–536, 1993
15. R.K. Srihari. Automatic indexing and content-based retrieval of captioned images. IEEE Computer, 28(9):49–56, 1995
16. S. Tsujimoto and H. Asada. Major components of a complete text reading system. Proceedings of the IEEE, 80(7):1133–1149, 1992
17. B.C. Watson and R.J. Davis. ODA and SGML: an assessment of co-existence possibilities. Computer Standards & Interfaces, 11:169–176, 1991
18. M. Worring and A.W.M. Smeulders. Content based hypertext creation in text/figure databases. In A.W.M. Smeulders and R. Jain, editors, Proceedings of the First international workshop on image databases and multimedia search, pages 59–66, 1996
19. M. Worring and A.W.M. Smeulders. From linear to nonlinear reading: a case study to provide internet access to paper documents. In Fourth International Conference on Document Analysis and Recognition, Ulm, pages 273–277. IEEE computer society, 1997
20. M. Worring, R. van den Boomgaard, and A.W.M. Smeulders. Structured hyperdocument generation using OCR and icon detection. In Third International Conference on Document Analysis and Recognition, Montreal, pages 1180–1183, 1995

Marcel Worring (M.Sc. honors 1988, PhD. 1993) is an assistant professor of computer science at the University of Amsterdam. In the fall of 98, he was a visiting scholar at the Visual Computing Lab, University of California, San Diego. Main topic of research is access to paper and video documents and its relation to databases. The goal is to add structure to collections of data to allow for access, exploration, and presentation. Projects are conducted in close relation with industry.



Arnold W.M. Smeulders is professor on Computer Science on Multi Media Information Processing. His interest is in image databases and intelligent interactive image analysis, as well as system engineering aspects of image processing. He is director of the Computer Science Institute and of the Intelligent System Lab of the University of Amsterdam. The ISIS-group conducts research on the theory of computer vision, industrial vision and multimedia information.