

A General System for the Retrieval of Document Images from Digital Libraries

Simone Marinai, Emanuele Marino, Francesca Cesarini, and Giovanni Soda

Dipartimento di Sistemi e Informatica
University of Florence
Via S.Marta, 3
50139 Florence - Italy
{marinai,marino,cesarini,soda}@dsi.unifi.it

Abstract

Large collections of scanned documents (books and journals) are now available in Digital Libraries. The most common method for retrieving relevant information from these collections is image browsing, but this approach is not feasible for books with more than a few dozen pages. The recognition of printed text can be made on the images by OCR systems, and in this case a retrieval by textual content can be performed. However, the results heavily depend on the quality of original documents. More sophisticated navigation can be performed when an electronic table of contents of the book is available with links to the corresponding pages. An opposite approach relies on the reduction of the amount of symbolic information to be extracted at the storage time. This approach is taken into account by document image retrieval systems.

In this paper we describe a system that we developed in order to retrieve information from digitized books and journals belonging to Digital Libraries. The main feature of the system is the ability of combining two principal retrieval strategies in several ways. The first strategy allows an user to find pages with a layout similar to a query page. The second strategy is used in order to retrieve words in the collection matching a user-defined query, without performing OCR. The combination of these basic strategies allows users to retrieve meaningful pages with a low effort during the indexing phase. We describe the basic tools used in the system (layout analysis, layout retrieval, word retrieval) and the integration of these tools for answering complex queries. The experimental results are made on 1287 pages and show the effectiveness of the integrated retrieval.

1. Introduction

After several years of massive digitization activities, main libraries hold now large collections of digitized books and journals. Some of these collections are available in Internet, and accessible for free download. In these systems, the retrieval of relevant documents is usually based on the information provided by catalog cards (e.g. title, author, and so on). Sometimes the document textual content is converted by OCR and in this case the retrieval by (imprecise) text content is possible with techniques derived from Information Retrieval (IR) [1] (see Section 2 for an analysis of some Internet digital libraries).

Document Image Retrieval (DIR) aims at finding relevant document images from a corpus of digitized pages. DIR is a research field that lies at the borderline between classic IR and Content Based Image Retrieval (CBIR) [2]. The basic idea of document image retrieval is to find documents relying on document image features only. Relevant sub-tasks include the retrieval of documents on the basis of layout similarity [3], and the retrieval considering the textual content [4]. A recent survey [5] investigated past research and future trends in document image retrieval. Most work has been based on the processing of converted text with IR-based techniques. Fewer methods approached the retrieval by layout similarity, and related approaches have been considered for document page classification.

In [6] a general framework for document image retrieval has been proposed. The system allows users to retrieve documents on the basis of both global features of the page and features based on blocks extracted by layout analysis tools. Global features include texture orientation, gray level difference histogram, and color features. The block-based features use a weighted area overlap measure between segmented regions. More recently, the combination of global (page-level) and local features has been furtherly investigated for computing visual similarity between document

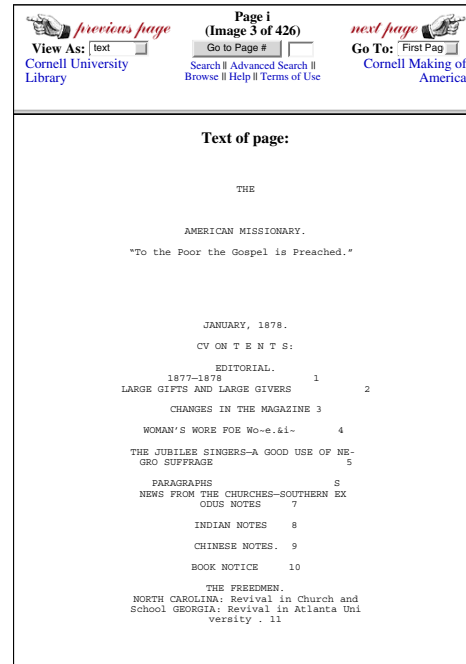
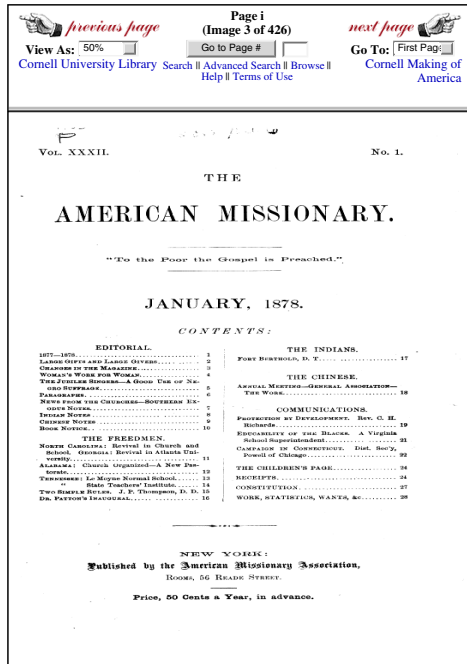


Figure 1. A page in the MOA Digital Library presented to the user with two formats: as image (left) and as text obtained by an OCR (right).

images for page classification [7]. In this approach a fixed-size feature vector is obtained by extracting some specific features in the regions defined by a grid superimposed to the page. Similar grid-based methods are described in [8] and in [9].

In order to overcome some problems related to the choice of an optimal grid size, a page retrieval method based on an MXY tree decomposition of the page has been recently proposed in [3]. This method relies on an MXY tree [10] built during image segmentation. In page indexing appropriate feature vectors describing both the global features of the page and the MXY tree structure are stored in the database. During retrieval, a query by example approach is considered. To this purpose the user first selects one sample page by browsing the collection; afterwards a comparison of the query feature vector with vectors in the database is performed with an appropriate similarity measure, and retrieved documents are shown to the user.

When dealing with the text the approaches can be clustered into two main categories on the basis of the use of OCR.

The first class of methods avoids the use of OCR and is usually based on two steps [11, 12]. In the indexing step the textual content of the document is encoded with some “ad-hoc” method, and the characters are represented with ap-

propriate features *without explicitly assigning a character class to individual objects*. In the retrieval step the relevant documents are extracted from the database by encoding the query with the same algorithm used during indexing, and matching the query representation with the encoded documents. Some queries are based on a simple word matching approach, whereas methods closer to IR identify the documents on the basis of distributions of word occurrences, and rank the fetched documents with appropriate similarity measures.

The second class of methods use OCR systems and the approaches rely on the processing of the converted text to find the information [13, 14]. In this case the systems must cope with the recognition errors that are inevitably introduced by the OCR. If errors in the retrieval task are allowed, then the uncorrected OCR output can be considered as text encoding and query matching has to deal with the previously mentioned errors. Two symmetric approaches can be considered. The first strategy relies on *query expansion*, and is based on the simulation of OCR errors during the retrieval step, by searching for distorted versions of the query term as well. For instance, when looking for the word “dog” it is possible to look also for “dgc”, expecting an OCR confusion between the “o” and the “c”. In the second approach, the query is compared with the words in the stored doc-

Making of America *Browse*

Browse Cornell University's MoA Journal Collection

- The American Missionary (1878 - 1901)
- The American Whig Review (1845 - 1852)
- The Atlantic Monthly (1857 - 1901)
- The Bay State Monthly (1884 - 1886)
- The Century (1881 - 1899)
- The Continental Monthly (1862 - 1864)
- The Galaxy (1865 - 1878)
- Harper's New Monthly Magazine (1850 - 1899)
- The International Monthly Magazine (1850 - 1852)
- The Living Age (1844 - 1900)
- Manufacturer and Builder (1869 - 1894)
- The New England Magazine (1886 - 1900)
- The New-England Magazine (1831 - 1835)
- New Englander (1843 - 1892)
- The North American Review (1815 - 1900)
- The Old Guard (1863 - 1867)
- Punchinello (1870)
- Putnam's Monthly (1853 - 1870)
- Scientific American (1846 - 1869)
- Scribner's Magazine (1887 - 1896)
- Scribner's Monthly (1870 - 1881)
- The United States Democratic Review (1837 - 1859)

Browse Cornell University's MoA Multivolume Monographs

- Official Records of the Union and Confederate Navies in the War of the Rebellion (1894 - 1922)

search *advanced* *browse* *help* *main*

© 1999 Cornell University Library. Comments and questions? [Click here.](#)

Making of America *Bibliographic Citation*

Cornell University Library

Title: The American missionary. / Volume 32, Issue 1
Publisher: American Missionary Association. **Publication Date:** Jan 1878
City: New York **Pages:** 426 page images in vol.

This journal issue: <http://cdl.library.cornell.edu/cgi-bin/moa/moa.cgi?hostid=ABK5794-0032&byte=202>

Go To: First Page of this journal issue
 Title Page
 A note on viewing the plain text of this volume

- The American missionary. / Volume 32, Issue 1, miscellaneous front pages (pp. i-ii)
- 1877 - 1878 (pp. 1-2)
- Large Gifts and Large Givers (pp. 2-3)
- Changes in the Magazine (pp. 3-4)
- Woman's Work for Woman (pp. 4-5)
- The Jubilee Singers - A Good Use of Negro Suffrage (pp. 5-6)
- Paragraphs (pp. 6-7)
- News from the Churches - Southern Exodus Notes (pp. 7-8)
- Indian Notes (pp. 8-9)
- Chinese Notes (pp. 9-10)
- Book Notice: Ethiopia, or Twenty Years of Missionary Life in Western Africa. Rev. D. K. Flickinger (pp. 10-11)
- North Carolina: Revival in Church and School. by Miss E. W. Douglass (pp. 11)
- Georgia: Revival in Atlanta University. by Rev. C. W. Francis (pp. 11-12)
- Alabama: Church Organized. by Rev. E. P. Lord (pp. 12)
- Alabama: A New Pastorate. by Rev. Charles Noble (pp. 12-13)
- Tennessee: Le Moyne Normal School. by Miss L. A. Farnes (pp. 13-14)
- Tennessee: State Teacher's Institute. by Prof. A. K. Spence (pp. 14-15)
- Two Simple Rules. by J. P. Thompson, D.D. (pp. 15-16)
- Dr. Patton's Inaugural (pp. 16-17)
- Fort Berthold, D. T. by E. H. Alden (pp. 17-18)
- Annual Meeting - General Association - The Work. by W. C. Pond (pp. 18-19)
- Protection by Development. by Rev. C. H. Richards (pp. 19-21)
- Educability of the Blacks. by R. W. P. A Virginia School Superintendent (pp. 21-22)
- Campaign in Connecticut. by Dist. Secretary of Chicago Powell (pp. 22-24)
- The Children's Page. by Miss T. N. Chase (pp. 24)
- Receipts (pp. 24-27)
- Constitution (pp. 27-28)
- Work, Statistics, Wants, &c. (pp. 28-29)
- Advertisements (pp. 29-32)

Go To: First Page of this journal issue
 Title Page
 A note on viewing the plain text of this volume

search *advanced* *browse* *help* *main*

© 1999 Cornell University Library. Comments and questions? [Click here.](#)

Figure 2. Browsing the MOA Digital Library: the index of journals (left), and the index of pages of one journal issue (on right).

ument by using a matching strategy that takes into account the “distance” (or dissimilarity) of two words. The latter approach is based on the computation of the *string edit distance* between the query and the strings in the database. This approach has been adapted by introducing “ad hoc” edit costs for most common OCR errors (e.g. [13]).

String edit distance can be used also when dealing with symbolic encoding of text in OCR-free approaches. The main problem of string edit distance based methods is the computational cost when dealing with large collections of documents. To speed-up the retrieval of similar words several approaches try to efficiently solve the *approximate string matching* problem (e.g. [15, 16]) which can be defined as follows: “find the text positions that match a pattern with up to h errors” [16]. Approximate string matching can be of low interest when the user is interested in ranking the words on the basis of their similarity with the query, since defining a priori an appropriate value for h can be difficult in general cases. In this case *ranking queries* appear more appropriate. The ranking query is a generalization of k -nearest-neighbor query with a previously unknown result set size k . The results of this query are ordered on the basis of the proximity with the query (similarly to k -nn query), but there is no need to define in advance the number of objects to be retrieved.

Unfortunately, approaching ranking queries for strings is computationally expensive, since all the words in the collection should be compared with the query and sorted. On the other hand ranking query has been efficiently solved in vectorial spaces where points are stored with multi-dimensional access methods (e.g. R-tree [17], and X-tree [18]). To use techniques specific of vectorial spaces in the domain of strings we need to formulate the string matching problem in a vectorial space. In a recent paper [4] we proposed a method for the retrieval of words encoded by describing the character-like objects of the words with classes obtained by the unsupervised learning of a Self Organizing Map (SOM [19]). The encoded words are converted into a fixed-size representation that can be compared with the query by means of the standard Euclidean distance.

In the present paper we describe the integration of the word and layout indexing and retrieval in a unique framework that can be used in Digital Library (DL) applications. We first review most common paradigms exploited by Internet DLs for document retrieval (Section 2). Section 3 is devoted to the description of the overall system, by briefly discussing the role of the various modules. In the next sections we analyze individual modules, namely the layout analysis (Section 4), the indexing and retrieval of page layout

Search Results



Figure 3. Browsing interface in the British Library web site. Thumbnails allow users to quickly identify pages.

(Section 5), and the indexing and retrieval of words (Section 6). Section 7 is dedicated to the analysis of the integration between word retrieval and layout-based page retrieval. Finally, some experimental results are discussed in Section 8, whereas the conclusions are in Section 9.

2. Retrieval in Digital Libraries

Several Digital Libraries made available in Internet parts of their collections for free browsing and download. These on-line libraries are good examples of the current technology and their properties are of interest for the Document Image Analysis research (DIA). In this section we briefly review the most important features of some relevant web sites.

The basic data that are provided to users are the catalog contents. This information is available in most web sites of libraries, and the exchange of catalog records has been standardized in the last decades through the OPAC (Online Public Access Catalog) infrastructure [20]. From the DIA point of view the most interesting libraries are those providing access to digitized pages of their holdings. The differences are related to the way users can find interesting documents, and to the file formats supported. Three main digital libraries are the subject of this analysis: Making of America (from Cornell University Library), Gallica (from National Library of France), and British Library.

The "Making of America" (MOA) digital library has been built with a collaborative effort between Cornell Uni-

versity and the University of Michigan. In the following we will refer to the Cornell Library (current URL: "http://cdl.library.cornell.edu/moa", see also [21]). MOA is a well known digital library of primary sources in American social history. The Cornell site of MOA provides access to 267 monograph volumes and over 100,000 journal articles. The MOA system allows users to view digitized pages of the 19th century texts. The website is fine for browsing the books, however images are in GIF format, the resolution is low, and each image has to be downloaded individually. Most documents in the collection have been processed with an OCR package without any check of the results. The text extracted with the OCR can be used for retrieving relevant documents with a keyword-based interface, although some OCR errors can affect the results (Figure 1). In addition it is possible also to browse the documents by looking at the index of journals, or reading (for each issue) the table of contents (Figure 2).

The "Gallica" digital library is the on-line service of the Bibliothèque Nationale de France (the National Library of France, current URL: "http://gallica.bnf.fr/"). The "Gallica" web site contains a large collection that is regularly updated adding new documents each month. Currently more than 15 millions of pages are available in digital format. Documents are not limited to French language; there are also digitized books in other languages. One interesting feature of this library is that entire books can be downloaded as a single multi-page TIF or PDF file. In addition the tables of contents of several books have been inserted in the site in order

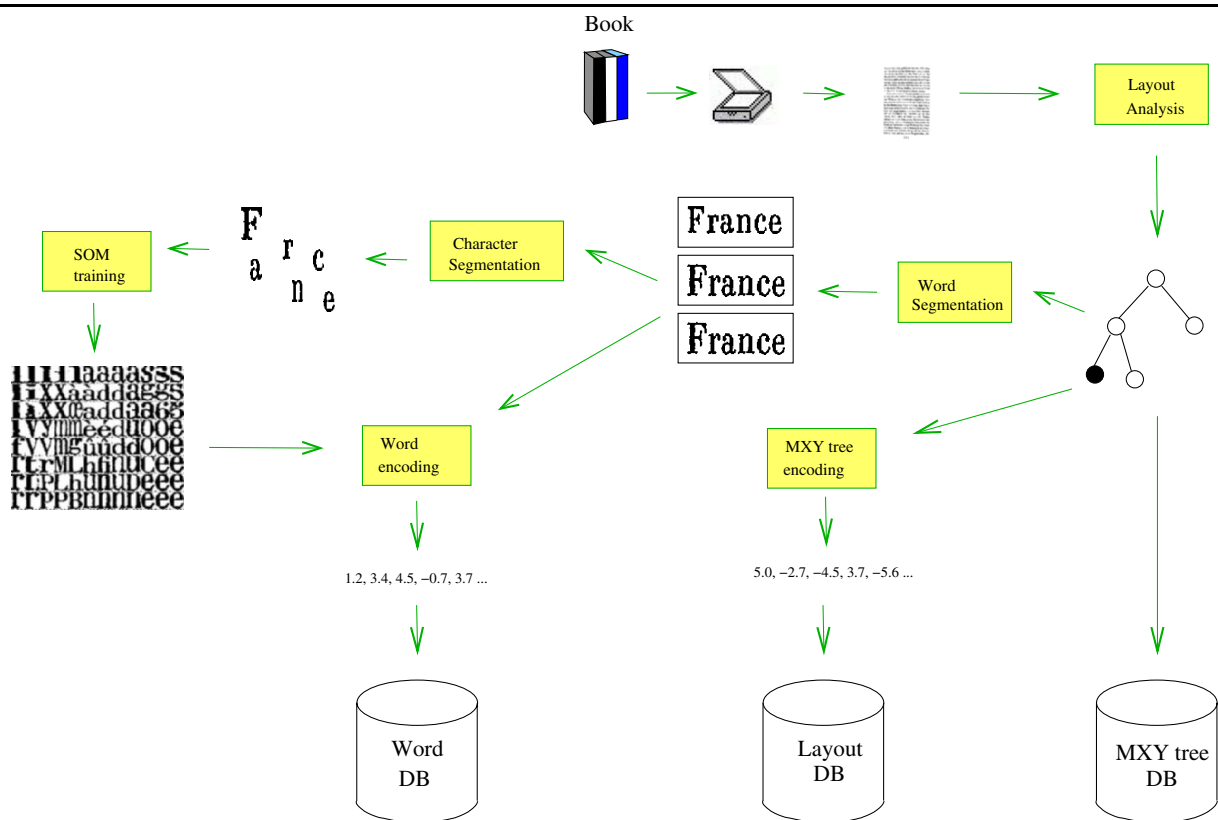


Figure 4. Tools and data structures of interest in the indexing step. The black node in the tree corresponds to a text region.

to facilitate document browsing and retrieval.

Another interesting web site (at least for historical reasons) is the site of the British Library that contains a link to two scanned books of the Gutenberg's Bible (current URL: "<http://prodigi.bl.uk/gutenberg>"). The images are high resolution, but should be downloaded individually. The pages can be retrieved by selecting the appropriate Book of the Bible, and looking for interesting pages in a thumbnails page (Figure 3).

From this brief excursus about the retrieval mechanisms exploited in current DLs it is clear that other approaches could be useful.

One direction is the implementation of libraries with high level information provided to users. An example of this approach is the "Miguel de Cervantes" digital library (current URL: "<http://cervantesvirtual.com>") that proposes large collections of manually transcribed (or checked) texts. Older documents are also annotated with linguistic contents. This approach is obviously very interesting from a user point of view, but it is very expensive.

The opposite direction is exploited by document image

retrieval systems. With this approach the information about the text and layout of scanned documents is indexed. The user can retrieve the document images with queries involving both the text and the layout.

In the rest of this paper we describe a system using this integrated retrieval approach.

3. System overview

The system described in this paper is built by combining several tools and algorithms that have been developed in the last few years by our research group. The overall system is designed in order to be able to index and retrieve in an efficient way the information contained in scanned documents. From a functional point of view two main steps must be considered: the indexing and the retrieval.

The general scheme of the indexing step is depicted in Figure 4, where the following functional blocks are interconnected:

- The *layout analysis* tool is dedicated to page segmentation, and the subsequent organization of the page

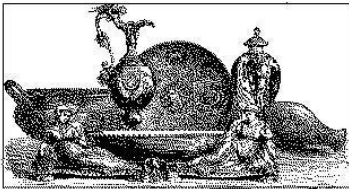
LES MERVEILLES
 DE
L'INDUSTRIE

DESCRIPTION DES PRINCIPALES INDUSTRIES MODERNES

LOUIS FIGUERE

ILLUSTRATIONS DE M. FIGUERE

PARIS — ÉDITIONS DE LA LIBRAIRIE DE LA FORTIFICATION
 1888 — 10, RUE DE LA FORTIFICATION, 10



PARIS

HOUVET ET C. ÉDITEURS
 10, RUE DE LA FORTIFICATION

MERVEILLES DE L'INDUSTRIE

Le verre Robert, qui est le plus pur, est obtenu par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires. Les éléments de ces verres sont de pureté parfaite, et les conditions de fusion et d'écoulement sont les mêmes que celles qui servent à la fabrication des verres ordinaires.

Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Le verre est une substance homogène, transparente, et qui se fond à une température élevée. Les verres ordinaires sont obtenus par des procédés qui reposent sur les mêmes principes que ceux qui servent à la fabrication des verres ordinaires.

Figure 5. Text lines and baselines in two document images.

blocks into a hierarchic data structure (the MXY tree). Moreover, this tool discriminates the text regions from those containing images. The procedures considered in this step are described in Section 4.

- Each MXY tree is encoded into a vectorial representation that is appropriate for layout-based document retrieval (Section 5). In addition each tree is entirely stored for handling integrated queries (Section 7.2).
- From each text region the words are extracted and their characters are segmented as well. The characters are used to train the SOM map required for word encoding (Section 6).

The tools employed in the indexing phase describe the scanned pages through the information stored into three main repositories (Figure 4): a collection of MXY trees (the "MXY tree DB"), an encoding of MXY Trees (the "Layout DB"), and an encoding of the words in the page (the "Word DB"). The SOM map is stored as well in order to represent the most common characters in the pages.

Given the information stored in the three databases and in the SOM map, there are several retrieval strategies that can be employed depending on user interests. These strategies are graphically depicted in Figures 10, 13, 17, 18, and will be described with more details in the rest of the paper. The basic approaches are the following:

- Retrieval of pages on the basis of the layout similarity (Section 4 and Figure 10). In this case the user chooses one page, and the system retrieves the pages with a layout similar to the user's query.
- Retrieval of pages containing a user-defined word (the position of the word in the page is shown as well). This strategy is described in Section 6 and explained in Figure 13.
- Integrated strategies that combine the retrieval by layout similarity and the word retrieval. These approaches are described in Section 7. Two examples of these strategies are:

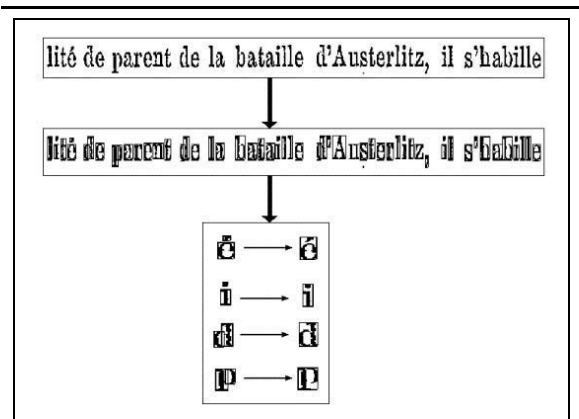


Figure 6. Connected components are first found in a text line. Character objects are computed from connected components.

- the retrieval of pages with a given layout, and containing an user defined word (Section 7.1 and Figure 17). This approach can be used, for instance, for identifying pages corresponding to the first page of a chapter.
- the retrieval of pages containing two (or more) user-defined words in the same text block (Section 7.2 and Figure 18). This method can be appropriate when an user is interested in locating pictures describing a given subject; in this case the keyword "Figure" will be searched together with a keyword related to the subject of interest.

In the following sections we analyze the most significant parts of the system.

4. Layout Analysis

The two main retrieval methods described in this paper are based on a symbolic representation of the page layout. Each input image is first processed by a layout analysis tool that is aimed at extracting homogeneous regions in the page. The basic items that are extracted from the image are the connected components, that are used in subsequent steps for locating words and text lines, as well as for segmenting the pages in homogeneous regions by building a tree-structure representing the page layout (the MXY tree).

4.1. Text line processing

Connected components are found with a classical region labeling algorithm, and several pieces of information are retained after this step: the coordinates of the bounding box

of the component, the number of black pixels, and the Median value of Black horizontal Run lengths in the component (MBR). The last value is a feature that, together with the character height, is strongly related to the font family and character attributes. From the computational point of view this step is the most demanding, since all the pixel in each image have to be read. All the subsequent steps work with the list of connected components obtained in this step. The original image is accessed, for word indexing, only in the parts corresponding to connected components that most likely contain characters.

Connected components are grouped together to find the text lines in the page. To this purpose adjacent components with similar height and similar MBR are grouped together. Another feature that is checked is the inter-component distance, that is related to inter-character distance in text lines. To deal with variable size text (e.g. titles), the inter-character distance is adapted to each text line, by taking into account the average distance between a horizontal neighborhood of the current character. When a text line is found, an estimated baseline for it is computed as well (see Figure 5 for an example of this processing step in two pages).

It is well known that connected components are, in Western languages, a good approximation of characters. The exceptions are due to accents (e.g. à è), and dots over 'i' and 'j'. In actual scanned documents broken and touching characters can give rise to wrong correspondences between connected components and characters. To reduce the effect of broken characters we perform a simple process of connected components merging by obtaining the Character Objects (CO). The algorithm is based on merging the connected components in the same text line with an overlap in the vertical projection profile (Figure 6).

4.2. MXY tree building

In this Section we briefly describe the segmentation that is obtained by using the MXY tree algorithm [10]. By means of this description we will analyze in Section 4.3 the improvement proposed in this paper that takes into account font changes for recursively segmenting a page.

The XY tree ([22], [23], [24]) is a well-known top-down method for page layout analysis. The basic method consists in using thresholded projection profiles to split the document into successively smaller rectangular blocks [23]. A projection profile is the histogram of the number of black pixels along parallel lines through the document. The blocks are split by alternately making horizontal and vertical "cuts" along white spaces that are found by using the thresholded projection profile. The result of such segmentation can be represented in a XY tree, where the root is for the whole page, the leaves are for blocks of the page, whereas each

level alternatively represents the results of horizontal (X-cut) or vertical (Y-cut) segmentation.

Algorithm 1 XYTree (*image*, *dir*, *vacuous*)

Input:
image: input image.
dir: direction of cutting ($H|V$).
vacuous: is a boolean that is true when *image* has been obtained with a vacuous cut.

Output:
node: node of the X-Y tree corresponding to *image*.

begin
 if IsSmall(*image*)
 then return NULL
 sub_regions \leftarrow FindRegions(*image*, *dir*)
 node \leftarrow AllocNode(*image*, *dir*)
 dir \leftarrow ChangeDir(*dir*)
 if *sub_regions* = \emptyset **then**
 if *vacuous* = true
 then return NULL;
 else begin
 node.child[0] \leftarrow XYTree(*image*, *dir*, true)
 return *node*
 end
 end
 i \leftarrow 0
 foreach *region* \in *sub_regions* **begin**
 node.child[*i*] \leftarrow XYTree(*region*, *dir*, false);
 i \leftarrow *i* + 1
 end
 return *node*
end

Algorithm 1 will be used as a basis for analyzing the variations of the XY tree segmentation method discussed here. The input to the algorithm are the region to be analyzed (*image*), the cutting direction (*dir*), and *vacuous*, a boolean that is true when *image* has been obtained with a vacuous cut. A vacuous cut [22] corresponds to a node with a unique child, that is required when a region must be cut at consecutive levels in the same direction (e.g. a column may be first cut horizontally into paragraphs, and again cut in the same direction to break paragraphs into text lines). At the first execution of XYTree, *image* corresponds to the whole image to be segmented, *dir* can be H or V (depending on the adopted strategy) whereas *vacuous* is set to false.

The function IsSmall is used to stop the splitting process when *image* is too small. Function AllocNode allocates the structure containing the node attributes (e.g. the cutting direction), and depends on the actual implementation of the algorithm. *node.child*[*i*] denotes the *i*-th child of *node*. ChangeDir changes the cutting direction, to alternate the splitting at different levels of the tree.

Algorithm 2 FindRegions(*image*, *dir*) (Separators: spaces)

Input:
image: Input image.
dir: Direction of cutting ($H|V$).

Output:
sub_regions: list of regions obtained by splitting *image*.

begin
 profile \leftarrow Projection(*image*, *dir*)
 sub_regions \leftarrow Cut(*image*, *profile*)
 return *sub_regions*
end

The main differences among various methods are in the ways split regions are obtained by function FindRegions. As shown in Algorithm 2, the extraction of regions is based on two processes: locating separators, and defining sub regions. Projection(*image*, *dir*), calculates the thresholded projection profile of *image* in direction *dir*. This procedure is the most expensive part of the algorithm, since for each execution of XYTree we need to consider all the pixels of the sub-image. One solution to this problem, that is employed also in our system, is based on the computation of projection profiles by using bounding boxes of connected components instead of single pixels [25].

Cut(*image*, *profile*) calculates the sub-regions composing *image* on the basis of *profile*. By changing the procedure Cut different segmentation strategies can be implemented.

In a recent paper [10] we proposed an extension of the X-Y tree segmentation algorithm that takes into account cuts along horizontal and vertical lines (an example of this segmentation is shown in Figure 7). In the following we briefly summarize the main algorithm (further details on the features of cutting elements can be found in [10]).

As previously remarked the XY tree segmentation requires two main operations. First, a set of potential cutting elements is found in the region to be segmented. Second, appropriate cutting elements are selected from the set previously defined.

The function FindRegions has been modified (Algorithm 3) in order to deal also with lines. At each level, the algorithm first tries to cut along lines, if no cutting lines are found, appropriate spaces are searched in the image. FindLines finds horizontal or vertical ruling lines, and CutLine calculates the regions that are obtained by splitting along lines. Similarly, FindSpaces, locates cutting spaces by using thresholded projection profiles, and CutSpace, calculates the corresponding regions.

In Algorithm 3, we consider only two kinds of separators: spaces and lines. By using these separators we obtain

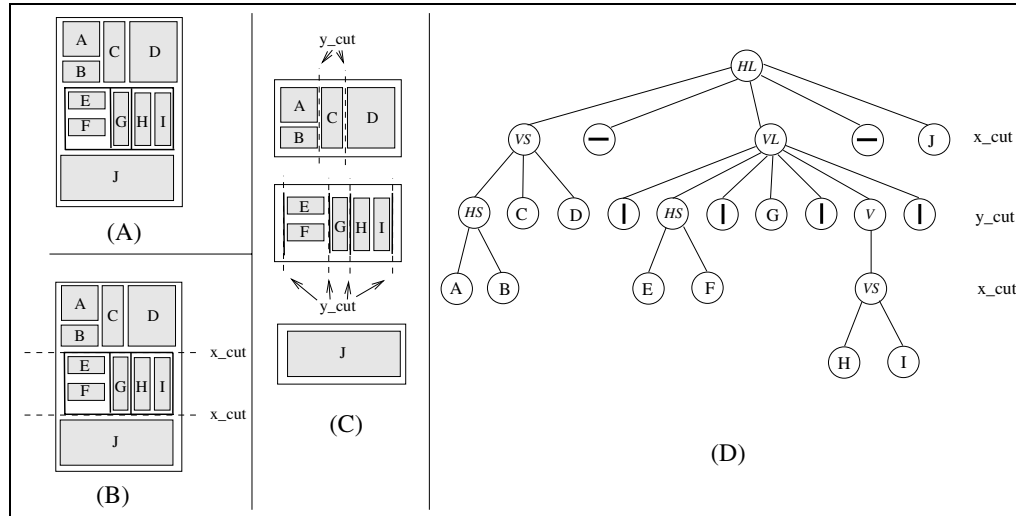


Figure 7. Example of MXY tree. (A): sample page (grey blocks correspond to homogeneous regions, thick lines are cutting lines). (B) and (C): first and second segmentation levels. The corresponding M-X-Y tree is shown in (D).

good results for a broad range of documents [10]. However, some documents contain other kinds of separators, and different items can be separated by abrupt discontinuities in the background colors, or by dashed lines. Another type of separator that is of interest when dealing with printed books is a change in font size or attributes that identifies a semantically different part (for instance the title of a Section or one or more sentences that constitute a citation). The extension to these separators is described in the next section.

4.3. Font changes as separators

The purpose of this extension of the MXY tree segmentation algorithm is the identification of separators between different zones of a text block that are obtained by changes in the font size or attributes. An example of this situation is shown in Figure 8 where we show a column text where the font features change several times.

The identification of cutting spaces is now performed by evaluating a measure of the change in font size and attributes that is computed for each horizontal white space. In this way the overall segmentation algorithm is, in fact, an integration between a bottom-up and a top-down algorithms. The bottom-up text line detection algorithm provides some features that are used by the top-down MXY tree algorithm. The obvious advantage of this algorithm (with respect to other pure bottom-up algorithms) is the use of the tree data structure for the subsequent retrieval of pages considering the layout similarity (Section 5).

In the bottom-up step we identify the text lines and compute for each of them two simple features: the Median value of Black horizontal Runs (*MBR*) and the Median value of Character Height (*MCH*). These two values can be computed directly for the features of the connected components (Section 4.1) without requiring additional accesses to the input image. Changes in *MBR* and *MCH* values suggest changes in font attributes.

This font change is more robust than variations in the size of horizontal white space between lines, or even than the distance between consecutive baselines (in fact the dis-

Algorithm 3 FindRegions (*image, dir*) (Separators: spaces and lines)

```

Input:
  image: Input image.
  dir: Direction of cutting (H|V).
Output:
  sub_regions: ordered list of regions obtained by splitting
  image.
begin
  lines ← FindLines(image, dir);
  sub_regions ← CutLine(image, lines);
  if sub_regions = ∅ then
    spaces ← FindSpaces(image, dir);
    sub_regions ← CutSpace(image, spaces);
  end
  return sub_regions
end

```

After several years of massive digitization activities, main libraries have now available large collections of digitized books and journals. Some of these collections are available in Internet, and accessible for free.

← THE SYSTEM ALLOWS USERS TO RETRIEVE DOCUMENTS ON THE BASIS OF BOTH GLOBAL FEATURES OF THE PAGE AND FEATURES BASED ON BLOCKS EXTRACTED BY LAYOUT ANALYSIS.

In these systems, the retrieval of relevant documents is usually based on the information provided by catalog cards, e.g. title, author, and so on. Sometimes the document textual content is converted by OCR and in this case the retrieval by (imprecise) text content is possible with techniques derived from Information Retrieval (IR). In a general framework for document image retrieval has been proposed.

← The system allows users to retrieve documents on the basis of both global features of the page and features based on blocks extracted by layout analysis.

Global features include texture orientation, gray level difference histogram, and color features. The block-based features use a weighted area overlap measure between segmented regions. More recently, the combination of global page-level and local features has been furtherly investigated for computing visual similarity between document images for page classification.

← The system allows users to retrieve documents on the basis of both global features of the page and features based on blocks extracted by layout analysis.

In this approach a fixed-size feature vector is obtained by extracting some specific features in the regions defined by a grid overlapped to the page.

← The system allows users to retrieve documents on the basis of both global features of the page and features based on blocks extracted by layout analysis.

The method discussed in, although based on different features and methods for computing the layout similarity, takes into account a partitioning of the page image into a fixed grid and uses features computed from the textual zones in the page. A grid-based approach to construct a feature vector obtained from the density of connected components was considered also in.

<i>FD</i>	<i>WS</i>	ΔMBR	ΔMCH
2	13	0	0
0	13	0	0
0	13	0	0
20	13	1	12
0	15	0	0
0	16	0	0
0	16	0	0
0	16	0	0
20	22	-1	-12
0	12	0	0
13	13	0	-8
13	13	0	8
0	13	0	0
0	13	0	0
0	12	0	0
0	13	0	0
5	7	0	3
0	5	0	0
0	5	0	0
0	5	0	0
5	10	0	-3
0	13	0	0
0	13	0	0
0	13	0	0
0	13	0	0
1	12	0	1
1	13	0	-1
2	12	2	0
0	18	0	0
0	13	0	0
2	18	-2	0
0	13	0	0
13	13	0	8
7	6	3	-4
0	5	0	0
0	5	0	0
0	3	0	0
7	12	-3	-4
0	13	0	0
1	13	0	-1
1	13	0	1
0	12	0	0
0	13	0	0
0	13	0	0

Figure 8. A text composed by several blocks with different font features. The distance between consecutive baselines is constant, however some features change. *FD* is the Feature Distance that is obtained (Eq. 1) from ΔMCH and ΔMBR . *WS* denotes the height (in pixels) of the horizontal white space.

tance between baselines in roughly constant for each pair of text lines in Figure 8). We denote with *FD* the Font Difference that is computed for each horizontal white space as follows: let ΔMCH_i be the difference between the *MCH* of the two text lines *i* and *i* + 1 (ad similarly for *MBR*). We can then compute the FD_i for space *S_i* (enclosed between text lines *i* and *i* + 1) as described in Eq. 1, where ΔMCH_i and ΔMBR_i are defined in Eq 2.

$$FD_i = \sqrt{3 \cdot \Delta MCH_i^2 + \Delta MBR_i^2} \quad (1)$$

$$\begin{aligned} \Delta MCH_i &= MCH_i - MCH_{i+1} \\ \Delta MBR_i &= MBR_i - MBR_{i+1} \end{aligned} \quad (2)$$

The multiplying factor before ΔMCH_i is considered in order to balance the contribution of ΔMCH_i and of ΔMBR_i . In Figure 8 we report a text block where the font size and attributes change several times. By comparing the values of *FD* and *WS* for each horizontal space it is possible to observe that the *FD* value measures in a more effective way font changes.

Algorithm 4 FindSpacesFD (*image*)

```

Input:
  image: Input image.
Output:
  sub_regions: ordered list of regions obtained by splitting
  image.
begin
  spaces ← FindSpaces(image, H);
  if NotUniform(spaces) then
    sub_regions ← CutSpace(image, spaces);
  else
    FD ← ComputeFD(spaces);
    if NotUniform(FD) then
      sub_regions ← CutFD(image, FD);
    else
      sub_regions ← image;
  return sub_regions
end

```

We used the *FD* value as a measure for locating and identifying cutting spaces. To describe this new segmen-

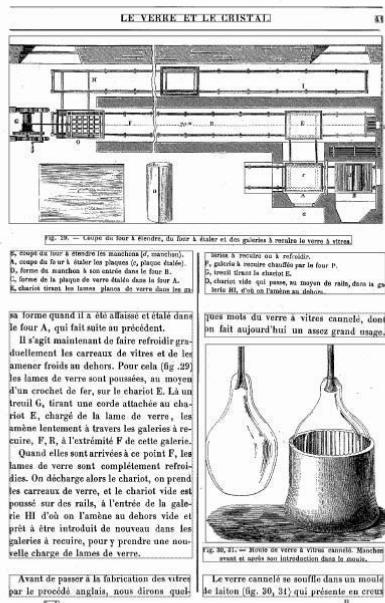


Figure 9. Examples of segmented pages obtained by the proposed method.

tation approach, we substitute the function `FindSpaces` in Algorithm 3 with the function `FindSpacesFD` described in Algorithm 4 (only when dealing with horizontal spaces). The latter algorithm implements the strategy of first looking for uniform horizontal spaces by using function `NotUniform`, that computes the standard deviation of the distribution of the widths of white spaces. When the standard deviation is low (spaces are uniformly distributed) we search potential cut position by computing the FD value for each space. The segmentation is then made along the highest values of this measure of font change (using function `CutFD`). If no way of segmentation is found, then the segmentation is stopped in this part of the tree. The corresponding region will be assigned to a leaf in the $MX Y$ tree. At the end of the segmentation each region is labeled on the basis of its content as Text (T) or Image (I). This labeling is obtained by taking into account the size of the connected components in the region, and the variation in MDR values.

The segmentation strategy just outlined has several advantages with respect to the standard $MX Y$ tree algorithm.

- There is no need to fix some thresholds that define

when the recursive segmentation needs to be stopped. In fact, this is decided by measuring the uniformity of cutting spaces (or FD values in case of uniform spacing).

- It is possible to break one of the limits of the classical XY decomposition algorithms: the segmentation of objects that are enclosed in textual regions (for instance text flowing around a picture).

Figure 9 reports two examples of segmented pages with the features just mentioned (different font sizes, and nested pictures). The left page contains a caption and two marginalia (at the bottom) that are close to upper objects but are well segmented, since the MCH and MBR values in the lines are different. The right page contains a figure that is larger than the column width. In this case the figure is well segmented (and also its caption) and the text column on the left is only partially broken. In case of use of the standard $MX Y$ tree algorithm the segmentation of this figure could be obtained only by setting the stop threshold to a value close to the interline distance. The effect on the processing of the whole page would be an over-segmentation of the page, providing a region for each text line.

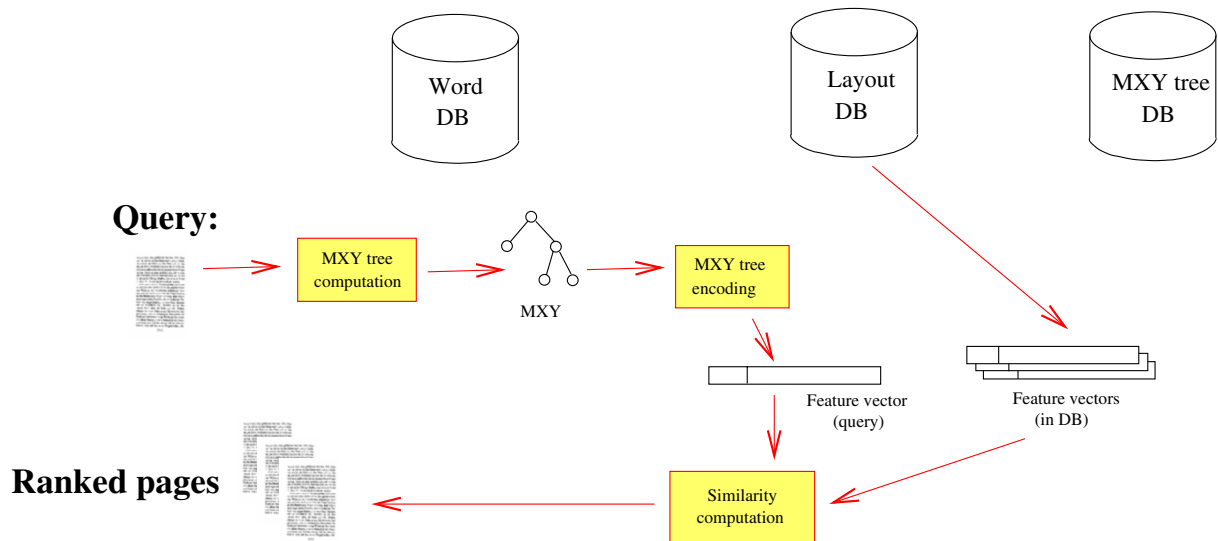


Figure 10. Main programs and data structures used in the retrieval of documents on the basis of the layout similarity.

5. Layout-based retrieval

The layout similarity is computed by considering the distance between feature vectors describing the page layout. With reference to Figures 4 and 10 there are several operations involved in layout-based document retrieval. In the indexing phase the page layout is encoded (Section 5.1) and stored in “Layout DB”. When performing the retrieval, the query page is first segmented and the corresponding layout is encoded as well. Pages in the database are subsequently ranked as described in Section 5.2.

Each feature vector contains two main groups of features. The first group is related to global features that describe the position and size of the printed part of the page with respect to the other pages contained in the same book. The second group describes the layout of the page and is obtained through an appropriate encoding of the MXY tree of the page. The page similarity is computed with a combination of two measures that operate independently for each group of features. In this section we describe the two parts of the feature vector and the similarity measure that we introduced in order to deal with this representation.

5.1. Layout encoding

The page layout is described by means of an MXY tree. This description has been demonstrated to be adequate for the classification of journal pages, where page layouts are quite complex and MXY trees are usually composed by several nodes [26]. When considering digitized books there are

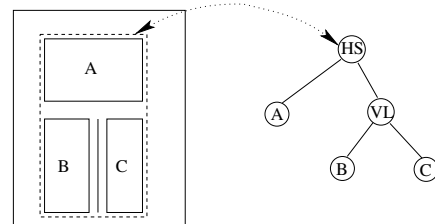


Figure 11. A simple MXY tree with the corresponding blocks in the page. The page bounding box is described in the root node.

some pages whose layout is made by a unique block. Typical examples are pages containing continuous text (the narrative part of the book). Other pages composed by a single text block contain, for instance, short dedications. These layouts can be recognized with some features describing the position and size of the printed part of the page with respect to the whole book. The printed part of a page is represented in the MXY tree root (Figure 11). Heterogeneous collections contain books with variable size. However, users are usually interested in pages with a given layout independently from the book size. The position and size of the root of the tree representing each page are normalized with respect to the bounding box of all the book pages. In so doing we obtain features that are invariant with respect to different book sizes and different book placements in the scan-

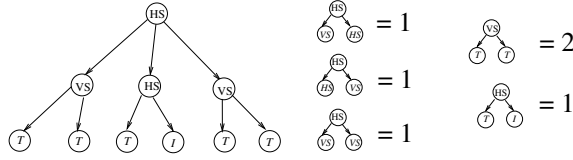


Figure 12. A simple MXY tree and the balanced tree-patterns in the tree, with the corresponding occurrences.

ner.

The features can be computed in the following way. Let (xb_i^j, yb_i^j) and (xe_i^j, ye_i^j) be the top-left and bottom-right points of the bounding box of page P_i in book B_j . The “book bounding box” can be simply computed by Equations 3:

$$\begin{aligned} XB^j &= \min_{P_i \in B_j} (xb_i^j) & YB^j &= \min_{P_i \in B_j} (yb_i^j) \\ XE^j &= \max_{P_i \in B_j} (xe_i^j) & YE^j &= \max_{P_i \in B_j} (ye_i^j) \end{aligned} \quad (3)$$

The page location can be described by computing the normalized position of the page center ($\bar{x}_i^j = \frac{xb_i^j + xe_i^j}{2}$, $\bar{y}_i^j = \frac{yb_i^j + ye_i^j}{2}$) with respect to the “Book Bounding Box” (Eq. 4):

$$x_i^j = \frac{\bar{x}_i^j - XB^j}{XE^j - XB^j} \quad y_i^j = \frac{\bar{y}_i^j - YB^j}{YE^j - YB^j} \quad (4)$$

The normalized width (w_i^j) and height (h_i^j) of page P_i in book B_j can be computed in the same fashion (Eq. 5):

$$w_i^j = \frac{xe_i^j - xb_i^j}{XE^j - XB^j} \quad h_i^j = \frac{ye_i^j - yb_i^j}{YE^j - YB^j} \quad (5)$$

The MXY tree data structure is encoded into a fixed-size feature vector for page classification by taking into account occurrences of *tree-patterns* made by three nodes [26]. This approach is motivated by the observation that similar layouts frequently contain some common sub-trees in the corresponding MXY tree.

Trees composed by three nodes can have two basic structures. The first pattern has the root and two children (and is denoted as *balanced tree pattern*). The second pattern is made by the root, one child, and a child of the second node (*chain tree pattern*). MXY tree nodes contain symbolic attributes describing the purpose of the node. Internal nodes represent the cut strategy considered: we can have cuts along either spaces or lines in the horizontal direction (*HS*, *HL*), or in the vertical direction (*VS*, *VL*). Leaves correspond to homogeneous blocks in the page: text (*T*), image (*I*), horizontal line (*hL*), or vertical line (*vL*). Since allowed node labels are in a fixed number, the number of

possible *tree-patterns* (denoted with TP) is fixed as well. In Figure 12 we show an example tree and the corresponding balanced tree-patterns.

Under these hypotheses, similar pages have some *tree-patterns* in common, and sometimes similar pages contain the same number of occurrences of a given *tree-pattern* (for instance table pages usually contain a large number of *tree-patterns* with lines). Unfortunately, there are some patterns that appear roughly in every document, and in this case these patterns are not very useful for measuring page similarities.

These peculiarities are very similar to the use of index terms in classic Information Retrieval. We extended the vector model approach, used in IR for dealing with textual documents, to our representation based on *tree-patterns*. The vector model of IR (see [1], Chapter 2) is based on a vectorial description of the document textual contents. The vector items are related to the occurrences of index terms, that usually correspond to words, in the document. Actual vector values are weighted in order to provide more importance to most discriminant terms. One common approach relies on the well known *tf-idf* weighting scheme. Basically, index terms that are present in many documents of the collection have a lower weight since their presence is not discriminant. In our approach the vector model is used to describe the page layout. To this purpose we use the MXY tree representation, and occurrences of *tree-patterns* are considered instead of word-based index terms. The extension of *tf-idf* weighting to this case is straightforward; the weight assigned to the k -th *tree-pattern* in page P_i is computed by the following equation:

$$w_{i,k} = f_{i,k} \cdot \log\left(\frac{N}{n_k}\right) \quad (6)$$

where $f_{i,k}$ is the frequency of the k -th *tree-pattern* in the tree corresponding to page P_i normalized with respect to the maximum *tree-pattern* frequency in the tree associated to page P_i , N is the total number of pages, and n_k is the number of pages containing the k -th *tree-pattern*.

5.2. Similarity computation

The similarity between two pages is computed by taking into account the corresponding feature vectors that are composed by two parts. The feature vector describing page P_i in book B_j can be represented as depicted in Eq. 7.

$$\vec{P}_i = [T, L, I, x_i^j, y_i^j, w_i^j, h_i^j, \dots, w_{i,1}, \dots, w_{i,k}, \dots, w_{i,TP}] \quad (7)$$

The first seven values correspond to global features. (T, L, I) are binary values describing the tree root. ($x_i^j, y_i^j, w_i^j, h_i^j$) describe the position and size of the tree

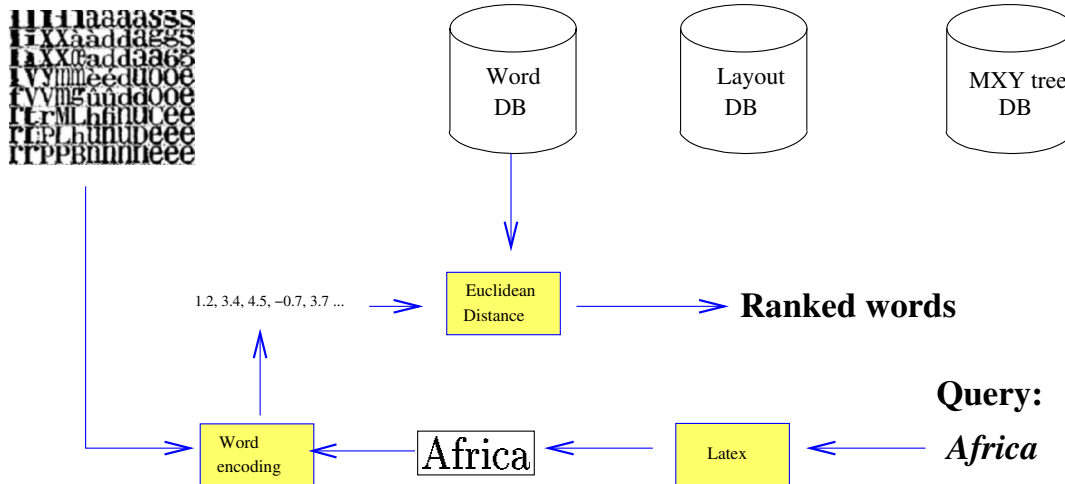


Figure 13. Main programs and data structures used for word retrieval. In this example the input is the query word “Africa”.

root (Eq. 4 and 5). The remainder of the vector contains an encoding of the MXY tree associated to the page: $w_{i,k}$ $k = 0, \dots, TP$ are the weights associated to the occurrences of *tree-patterns* (Eq. 6).

The MXY tree of a page with a single block has a unique node corresponding to a text block, a line, or an image. These three cases are described with a mutual exclusion in T, L, I values (for instance a text block is described by $T = 1, L = 0, I = 0$). When the page contains more blocks, then the root does not correspond to a single block, and in this case the three values are all set to zero.

The similarity between a query page q and a generic page p in the database is computed by combining two similarity measures for the two components of the feature vector. Let \mathcal{F} be the feature vector space, and $\vec{V} \in \mathcal{F}$ be a generic vector in \mathcal{F} . We indicate with \vec{V}_{GL} and \vec{V}_{XY} the global and the MXY-tree specific sub-vectors, respectively. Let $\vec{Q} \in \mathcal{F}$ and $\vec{P} \in \mathcal{F}$ be the feature vectors corresponding to the query page q and to the page p , respectively. The similarity between q and p can be computed by Eq. 8

$$Sim(\vec{P}, \vec{Q}) = \alpha \cdot SimEuc(\vec{P}_{GL}, \vec{Q}_{GL}) + \beta \cdot SimCos(\vec{P}_{XY}, \vec{Q}_{XY}) \quad (8)$$

The similarity between \vec{P}_{GL} and \vec{Q}_{GL} is computed by using the Euclidean distance between the two sub-vectors (Eq. 9). The distance is divided by the maximum value that can be reached ($\sqrt{6}$) in order to bound the maximum value to 1, and this value is subtracted from 1 in order to obtain values close to 1 when the pages are similar and the two sub-vectors are the same.

$$SimEuc(\vec{P}_{GL}, \vec{Q}_{GL}) = 1 - \frac{\sqrt{\sum_{i=1}^7 (P_{GL}[i] - Q_{GL}[i])^2}}{\sqrt{6}} \quad (9)$$

The similarity between the two sub-parts of the vector describing the MXY tree is computed by taking into account the *cosine of the angle* between the two vectors (Eq. 10)

$$SimCos(\vec{P}_{XY}, \vec{Q}_{XY}) = \frac{\vec{P}_{XY} \cdot \vec{Q}_{XY}}{|\vec{P}_{XY}| \cdot |\vec{Q}_{XY}|} = \frac{\sum_{i=1}^{TP} (P_{XY}[i] \cdot Q_{XY}[i])}{\sqrt{\sum_{i=1}^{TP} P_{XY}[i]^2} \cdot \sqrt{\sum_{i=1}^{TP} Q_{XY}[i]^2}} \quad (10)$$

The two parameters α and β are used in order to weight the contribution of the two parts to the overall similarity measure. Several tests have been made by varying the values of α and β as discussed in [3]. The main conclusion that can be drawn from these experiments is the observation that the retrieval algorithm is robust with respect to different values of α and β , and that good results can be obtained with a wide range of values for α and β . In the integrated system described here we set $\alpha = 0.5$ and $\beta = 0.5$.

6. Word indexing and retrieval

Similarly to layout retrieval, word retrieval is composed by an indexing phase (Figure 4) and by a retrieval one (Figure 13). In the indexing phase the textual parts of a page are first segmented into words and then *character objects* (CO) are extracted by locating connected components and grouping together overlapping components (see Section 4). This

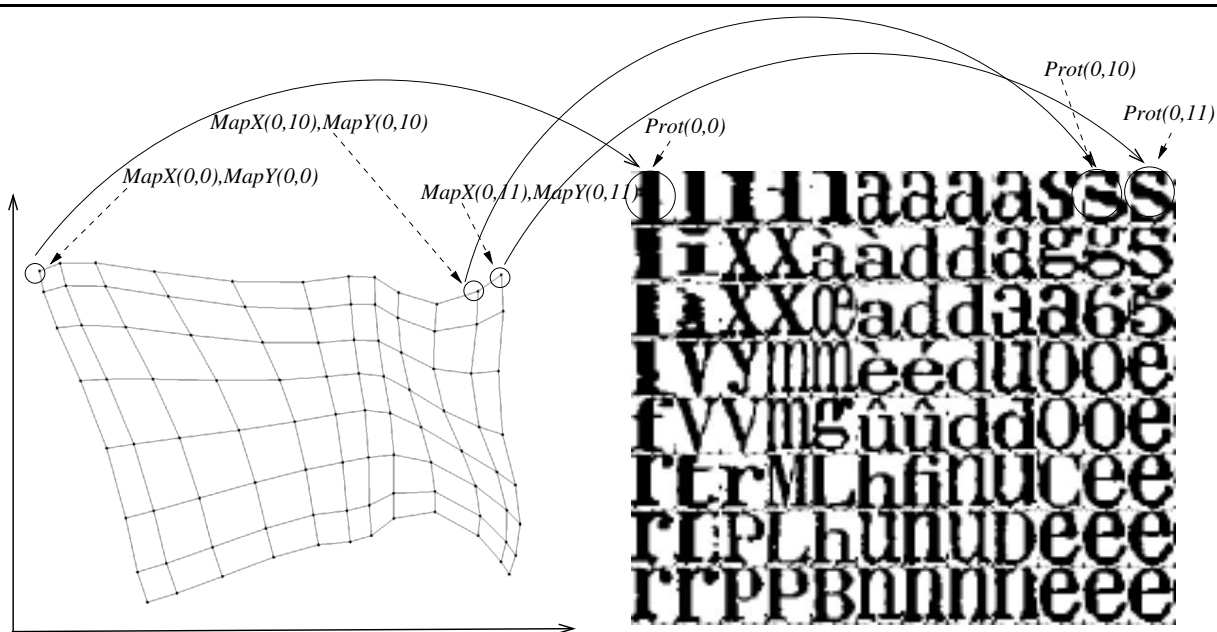


Figure 14. Two graphical representations of the trained SOM map. Left: 2D coordinates of the neurons, showing the proximity of similar nodes. Right: some COs that are most similar to prototypes corresponding to the output neurons.

process is not error-free and broken or touching characters can be considered as *CO*. The *COs* of each word are labeled according to a clustering algorithm, so as to obtain a fixed-size representation. In the retrieval phase a query word is typed by the user and the system (using the \LaTeX package) generates a query image that is encoded with the same algorithm used in the indexing. Most similar words are found by computing a simple Euclidean distance between the query word and the words stored in the database.

6.1. Character clustering

Character-like coding is a well known approach for performing text retrieval without OCR. The indexing algorithm is composed by two main parts. In the first step character objects are extracted from the document image. In the second step each *CO* is described with a symbolic (a code) or sub-symbolic (a feature vector) representation. Character description is based on the clustering of similar characters. The main distinction between this step and an OCR is that here the system does not try to assign the “right” class to characters. In its simplest form the encoding can be based on *character shape coding* that has been successfully applied for Information Retrieval without OCR [11]. Character shape codes capture the main features of individual characters without the computational cost of OCR. With a

more expensive approach a feature vector can be computed for each *CO*, as described in a recent application to text retrieval [12].

In our system we use Self Organizing Maps (SOM [19, 27]) for character object clustering. The Self Organizing Map is a special kind of artificial neural network that is based on competitive learning algorithms, where the output neurons of the network compete among themselves. The SOM neurons are arranged in a two dimensional lattice (feature map). Each neuron receives inputs from the input layer and from the other neurons in the map. During learning the network performs clustering by means of a competitive learning mechanism [27]. Each neuron has attached one feature vector that can be thought as a prototype for the patterns associated to the corresponding cluster. Moreover, each neuron is placed in a 2D space. At the beginning of the learning neurons are placed in random positions, during learning the neurons are moved so as to reflect cluster similarity by means of distance in the map. Further details on the learning process can be found in [27].

In our approach each *CO* is first scaled to fit a 8 by 10 grid. The 80 values that are computed from the pixel density in each item of the grid are used for *CO* representation. Similarly, the SOM prototypes are vectors with 80 elements. With reference to Figure 14, we can see the information associated to each neuron (i, j) : the neuron position

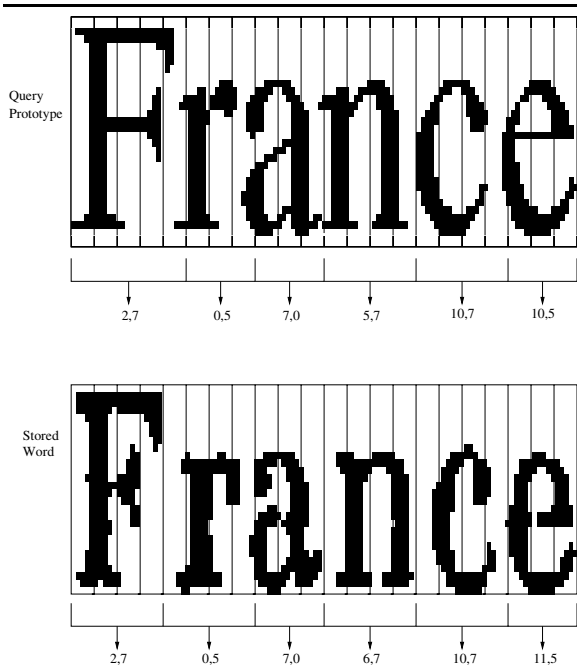


Figure 15. Top: the slices superimposed to a query word obtained with \LaTeX . Bottom: a word in the database with the corresponding slices. Below each word we report the coordinates of the SOM neurons corresponding to each slice.

is given by $MapX(i, j)$ and $MapY(i, j)$ whereas its prototype ($Prot(i, j)$) is graphically represented by the CO in the training set that is closer to the 80-dimensional prototype vector.

One advantage of the use of SOM for CO clustering is the spatial organization of the feature map that is achieved after the learning process. Basically, more similar clusters are closer than more different ones. Consequently the distance among prototypes in the output layer of the SOM can be considered as a measure of similarity between characters in the clusters (see Figure 14 for an example).

6.2. Word encoding

When digitized words contain broken or touching characters, two instances of a given word are not constrained to have the same number of COs. This problem is solved in OCR systems by checking the recognition results with appropriate dictionaries. In our system, each CO is assigned to one SOM neuron, and identified with the coordinates of the neuron in the map. A word can therefore be represented

with a variable size vector corresponding to COs output neurons.

This variation in string length is an obstacle to fast retrieval of words, and a string edit distance algorithm should be considered for word retrieval. However, the use of string edit distance for comparing word encodings is somehow inappropriate, since we lose one of the main features of original words: regardless of touching or broken characters the overall word size is nearly fixed in a given document. This feature is considered in holistic keyword spotting, where entire words are described with features that are afterwards compared with a word representation during retrieval (e.g. [28]).

One simple approach in holistic word recognition is based on zoning of the word (e.g. [29]). Zoning consists in overlapping the word with a fixed-size grid, and computing some features (e.g. the density of black pixels) in each grid region. One advantage of word zoning is the property that individual characters are located “roughly” in the same position in the grid regardless of touching or broken characters in the word.

We extended this approach to convert a variable length word encoding (obtained by CO extraction and SOM clustering) into a fixed-size symbolic description (the *expanded string*). The basic algorithm is the following one:

1. The COs in the word are located.
2. Each CO is labeled with the output neuron of the trained SOM.
3. The word image is partitioned into a fixed number of vertical slices. The number of slices (NS) is computed on the basis of the word aspect ratio (N) and ranges from $NS = 25$ when $N < 0.15$ to $NS = 6$ when $N > 0.55$.
4. Each slice gets the coordinates of the largest CO overlapping it.

After this step a fixed-length vector is assigned to each word, though the length of the vector is larger than the original word. This process is summarized in Figure 15.

We can formalize the construction of the encoding vector for a word W_i with a SOM map of size $N \times M$ as follows. We first define the “winning neuron” of the SOM for a given CO, as the neuron whose prototype has the lowest distance with respect to the CO under consideration (Eq. 11), where i, j are the indexes used to identify a neuron in the output layer, and $Prot(i, j)$ is the vector that corresponds to the prototype assigned to the neuron (graphically indicated as a character in Figure 14).

$$(i_k, j_k) = \underset{i=0, \dots, N-1}{\underset{j=0, \dots, M-1}{\operatorname{argmin} \operatorname{Dist}(CO(k), Prot(i, j))}} \quad (11)$$

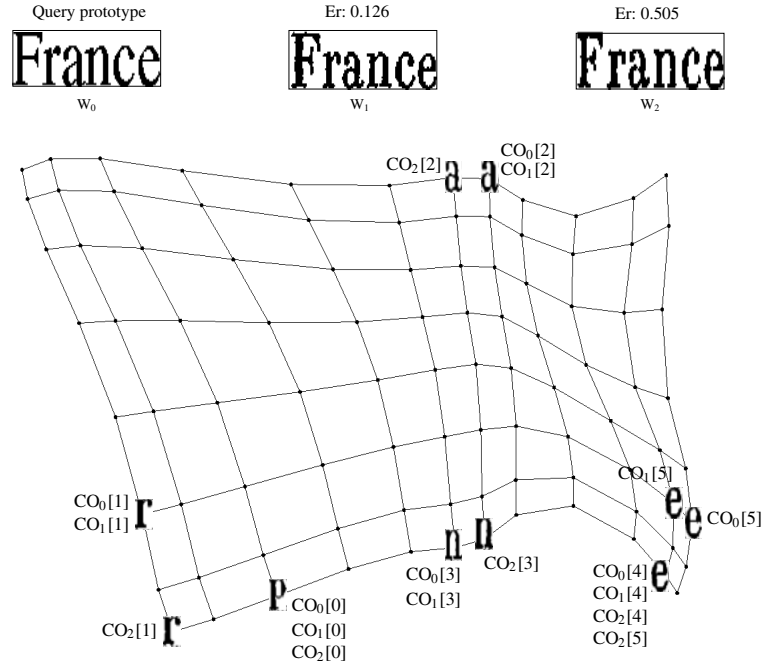


Figure 16. SOM neurons assigned to each CO of three words. The first word is a query image. The other words are instances in the database.

Let $NSlices(i, k)$ be the number of slices assigned to the k -th CO of word W_i , and let $MapX(i, j)$ and $MapY(i, j)$ be the position of the output neuron (i, j) . The word W_i (composed by n COs) can be represented by the vector reported in Eq. 12).

$$\vec{W}_i = [MapX(i_0, j_0), MapY(i_0, j_0), \dots, MapX(i_0, j_0), MapY(i_0, j_0), MapX(i_1, j_1), MapY(i_1, j_1), \dots, MapX(i_1, j_1), MapY(i_1, j_1), \dots, MapX(i_{n-1}, j_{n-1}), MapY(i_{n-1}, j_{n-1}), \dots, MapX(i_{n-1}, j_{n-1}), MapY(i_{n-1}, j_{n-1})] \quad (12)$$

Where $MapX(i_k, j_k), MapY(i_k, j_k)$ is repeated $NSlices(i, k)$ times.

The vectors are compared with the query in the retrieval step, as described in the next Section.

6.3. Computing word similarity

Two main operations are performed for retrieving the words matching a given query (see Figure 13). First, a textual query is translated into one or more word images that are encoded with the method described in Section 6.2. Sec-

ond, each image is compared with the word descriptions that are stored in the database.

From a user point of view the queries are made to the system with a simple text-based interface. Starting from the ASCII word one “clean” word image is obtained with \LaTeX software. The clean image is corrupted with synthetic noise (we use programs based on the Baird’s noise model) in order to simulate actual distortions occurring in real world. Note that this step is the unique point in the overall system that needs to be customized when changing the set of documents considered, since the predominant font in the text must be indicated. It is important to observe that the SOM clustering is robust with respect to font changes. For instance (Figure 16) the neurons corresponding to characters in the query word (W_0) and in W_2 are identical, though the ‘r’ and the ‘a’ are quite different in the two words. In all the experiments involving roman alphabet we use the standard \LaTeX font, whereas with documents containing Gothic text we changed the \LaTeX font accordingly.

Each word image is represented into a fixed-size vector (as described in Section 6.2) and compared with vectors in the database having the same length. Words with different aspect ratios are unlikely to correspond to the query, and consequently are not considered for the comparison. In the top part of Figure 15 we can see one example of a proto-

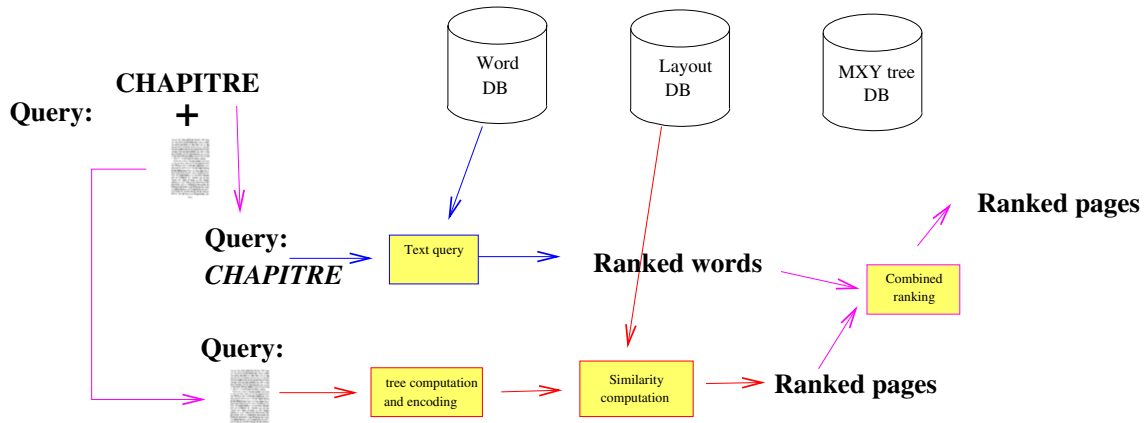


Figure 17. Integrated retrieval obtained by combining the score of layout similarity with the score of word retrieval. The query is composed by a combination of a query word (in this case the word *CHAPITRE*) and a page layout.

type word generated with \LaTeX .

Words in the database are ranked on the basis of the similarity with the query word by comparing the corresponding vectors with the Euclidean distance. In Figures 15 and 16 we can observe that the Euclidean distance between expanded strings approximates the similarity between the corresponding words for two reasons. First, also in presence of broken or touching characters the strings have the same length and codes corresponding to well segmented characters are roughly in the same position in the string. Second, the topology preservation feature of SOM maps allows us to evaluate also the similarity between close clusters in the output layer of the SOM, since closer nodes have lower distances. This is simplified in Figure 16 where the SOM neurons associated with each *CO* of three words (W_0 and W_1 are shown in Figure 15) are identified. The first word corresponds to the query generated by \LaTeX , whereas the other are actual words. It is interesting to see that some characters (for instance the 'a's) are quite different in the font used for query generation and in the actual font. However, the two characters are mapped quite close.

The main weakness of this similarity evaluation is related to the processing of very noisy words. One of the more interesting features of this method is the ability to adapt to different fonts. For instance in [4] we show the results that can be achieved when dealing with Gothic fonts.

7. Integrated Document Retrieval

We analyzed in the previous sections two basic tools that can be used for retrieving relevant pages either on the basis of their layout or by looking for the words contained in the indexed pages. In this section we describe the inte-

gration of these retrieval algorithms to provide additional retrieval mechanisms to users. The methods proposed are based on two main principles: first, it is possible to combine the ranking of pages computed by the two previously described methods, to obtain an overall score; second, it is possible to use the layout information (represented by the MXY tree) to establish a relationship between keywords found in a page. The experimental results supporting this integration will be described in Section 8.

Let us first summarize the sorting of pages and words in the database on the basis of their similarity with respect to a query page pq and a query word wq .

The layout retrieval associates a similarity measure to each page in the database. During the indexing a feature vector (\vec{P}_i) is associated to each page p_i in the database as described in Section 5.2. This feature vector is computed also for the query page pq obtaining the vector \vec{P}_Q . The similarity of the query with each page in the database is computed by $Sim(\vec{P}_i, \vec{P}_Q)$ (Eq. 8). The pages in the database can thus be sorted on the basis of this similarity value (higher values correspond to most similar pages). This information is represented in vector $LayPos[i]$ that associates to each page p_i in the database its position. In particular the page p_k such that $LayPos[k] = 0$ is the page most similar to the query pq (usually, if pq is stored in the database, then $p_k \equiv pq$).

Concerning words, the retrieval mechanism described in Section 6.3 allows us to associate a distance between the query word wq and each word w_j in the database. A feature vector (\vec{W}_j) is computed for each stored word and for the query word (\vec{W}_Q). By computing the Euclidean distance between \vec{W}_Q and \vec{W}_j it is possible to sort the words in the database. In this case the lowest distance values cor-

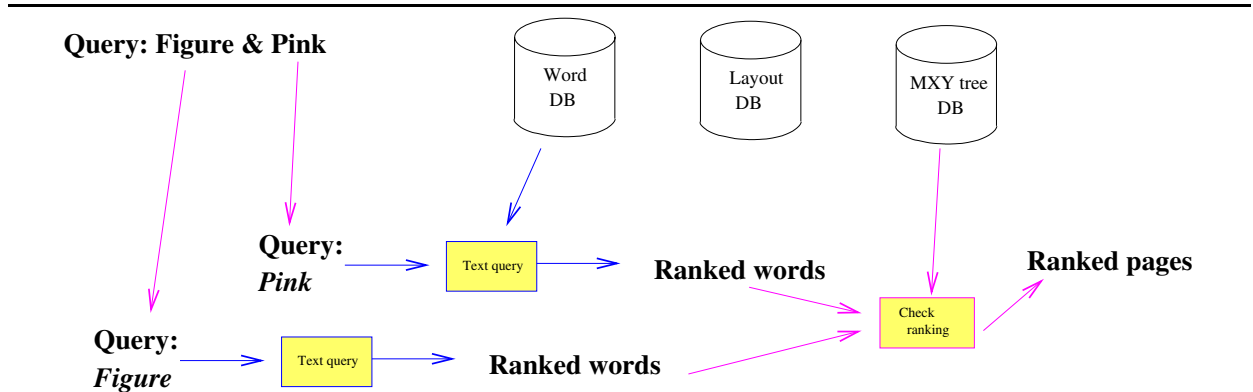


Figure 18. Integrated retrieval obtained by selecting pages where two user-defined keywords are in the same region.

respond to most similar words, and it is possible to sort all the pages as described by vector $WordPos[j]$ (the word W_h such that $WordPos[h] = 0$ is the word most similar to the query). Each word w_j in the database has some additional information associated to it: the page containing it (PW_j) and its coordinates in the page (XW_j , and YW_j).

7.1. Combining layout and words

The first integration strategy is based on a simple combination of the scores obtained by the two methods previously described (Figure 17). The user formulates the query by providing to the system two items: one page pq , that is used for the layout retrieval, and one word wq , that is used for word retrieval. The system provides as response to this query a sorting of the pages on the basis of their similarity to the combined query. This similarity is computed as follows.

Given pq and wq , the system first computes the two similarity vectors $LayPos[]$ and $WordPos[]$ with the independent methods previously described. To each page it is subsequently associated a new position in the integrated sorting ($CombPos[]$). Let Fw_i be the first word in page p_i in vector $WordPos$ ($Fw_i = \text{argmin} WordPos[j] \mid PW_j = P_i$), we can then compute $CombPos$ by Eq. 13.

$$CombPos[i] = LayPos[i] + WordPos[Fw_i] \quad (13)$$

By sorting the pages on the basis of increasing values of $CombPos$ it is possible to show to the user the pages in increasing order of similarity with respect to the combined query. This approach to combine the retrieval mechanisms can be used for improving the layout retrieval by some keywords related to the page. One example is the retrieval of the pages corresponding to the start of a section. We will discuss this example in Section 8.

7.2. Evaluating mutual position of retrieved words

An useful approach for locating relevant information is the use of queries based on the retrieval of several keywords (Figure 18). In Information Retrieval this task corresponds to the search for documents containing all the specified keywords (or a combination of these, depending on the query formulation). In this case the retrieval is affected by the definition of “document”, that can be a book, a journal, an article, a page or even a sentence. In the case of document image retrieval the most obvious resolution is the page, and frequently this is the object that can be retrieved by multiple keyword searches. However, when the layout information is available, as in our system, other approaches can be pursued.

As an example let us consider an user interested in the retrieval of figures related to a given subject. In this case a simple approach could be based on searching for pages containing both the keyword “Fig.”, and one or more words describing the subject of interest. Obviously, one limit of this approach is that in this way some false positives could be retrieved, since also pages with figures not related to the subject, but with the keywords in the text will be retrieved. To solve this problem in our system there are some types of combined queries. Some examples of queries (related to the experiments reported in the next section) are the following:

- Find the pages containing the specified keywords.
- Find the pages where one text block contains all the keywords (in this case it is possible also to display the text block).
- Find the pages where one text block contains the specified keywords, and the block is in some relationship with respect to another region (e.g. the text block is “below” an image region).

In order to solve all these queries the system must compute a measure of the likelihood that a layout object (a page or a text block) contains more user-defined words. In case of word indexing by OCR an exact matching strategy could answer to these queries with a boolean approach, by retrieving the objects where all the query words match some object. However, when searching for a word our system does not provide a boolean answer, but a ranked list of words. When searching for more keywords we can run the algorithm several times (one for each word) and then integrate the lists provided for each word in a way similar to the approach described in Section 7.1.

8. Experimental Results

In this section we discuss the experiments that have been carried out for evaluating the performance of the integrated retrieval described in Section 7. Experimental results of the individual methods used for layout-based retrieval and for word retrieval have been already reported in [3] and [4], respectively.

The experiments reported below are representative of typical queries that can be of interest for users of DLs. The tests have been made in order to show the utility of integrating more retrieval mechanisms into a single one. The results are reported in terms of Precision-Recall plots (the vertical axis corresponds to the Precision, and the horizontal one to the Recall).

8.1. Dataset features

The experiments reported in this section have been made on a dataset composed by books belonging to an existing Digital Library. In particular we worked with two books downloaded from the Gallica DL¹. The books contain a total of 1287 pages that can be divided into a few classes on the basis of their layout. In Figure 19 we show the features (and names) of some of the most significant classes.

8.2. Combining layout and words

The first experiments are aimed at evaluating the effectiveness of the integration method described in Section 7.1. From a user point of view an interesting feature is the retrieval of pages in a book that correspond to the beginning of chapters. There are two ways to identify these pages: by finding pages with a given layout (the page class is named *SecM2* in table 19), or by looking for the keyword “CHAPITRE” (chapter in French). Figure 20 compares the

¹ The pages correspond to the two books N0024670, and N0024671 that can be downloaded also starting from the database page of DAS 2004 web page: www.dsi.unifi.it/DAS04.

results that can be achieved by using the layout-based approach (continuous line), and an integrated one (dashed line). Since there are several pages with layout *SecM2* we used in turn each of them as a query page, and then we averaged the Precision-Recall plots of each experiment to obtain Figure 20.

The low performance of the layout-based method are improved by using also the appropriate keyword. We must notice that the results obtained using only word retrieval (searching for pages containing the word “CHAPITRE”) are better than both methods.

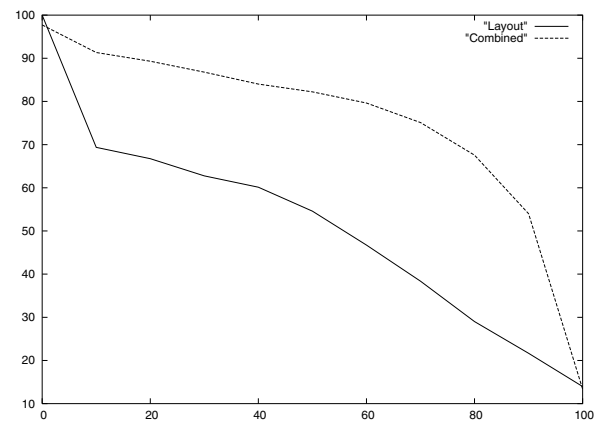


Figure 20. Precision-recall plots related to the combination of layout similarity and word searching (Keyword: *CHAPITRE*, layout: *SecM2*).

The retrieval mechanism described in Section 7.1 can be used also for retrieving pages containing a figure. In this case the query can be made by using the keyword “Fig.” and one page containing a figure. In Figure 21 we show the Precision-Recall plots that are obtained when using the keyword “Fig.” and pages of classes “ImageText2” and “Text2Image” respectively. Also in this case the use of the integrated method allows us to get better results with respect to the use of the layout retrieval alone. The use of the word based query (where we just look for pages containing the word “Fig.”) provides in this case worst results with respect to the integrated approach. This is due to two reasons: first, there are some words “Fig.” also in the body of the text (for instance due to links to the figures in the text); second, the keyword based retrieval locates also pages with the other types of layout containing figures (see Table 19) that are considered as errors.



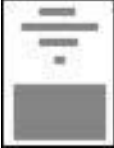





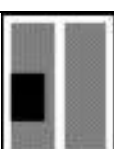
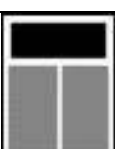
Name	Description	Example	Name	Description	Example
<i>Title</i>	Title page		<i>Sect0</i>	Section type 0	
<i>Issue1</i>	Issue page (1 column)		<i>SecS2</i>	Section start page (2 columns)	
<i>SecM2</i>	Section mark page (2 columns)		<i>Text1</i>	Text on one column.	
<i>Text2</i>	Text in two columns		<i>Image</i>	Pages containing images only	
<i>Text2Image</i>	Text in two columns and images in the text		<i>ImageText2</i>	Text in two columns and at least one image	

Figure 19. Some classes in the database (black rectangles correspond to images, gray ones correspond to text regions). Classes containing images are *Image*, *ImageText2*, and *Text2Image*.

The reason why the layout retrieval is not very effective with these experiments is due to the fact that there are some sub-classes corresponding to the *SecM2* class (for instance the chapter title can be either on the left or in the right text column), and to *Text2Image* and *ImageText2*. One solution to this problem could be the use of tree-grammars for query expansion (by producing additional query pages for each page proposed by the user). To this purpose, it is possible to use the approach that we proposed for the problem of page classification [30].

8.3. Evaluating mutual position of retrieved words

In order to evaluate the second integration strategy we made a test that is aimed at retrieving figures related to a

given subject. To this purpose we performed a combined search by looking for the join retrieval of pages containing both the subject related keyword and the word "Fig.". We compared four methods where the conditions checked for answering the query have increasing complexity:

1. We return the pages containing both the keyword and "Fig.".
2. We return the pages where one region contains both the keyword and "Fig.".
3. We return the pages containing an image region just over one region containing both the keyword and "Fig.".
4. We return the pages satisfying condition 3 and such that a constraint is imposed on the region (that has to

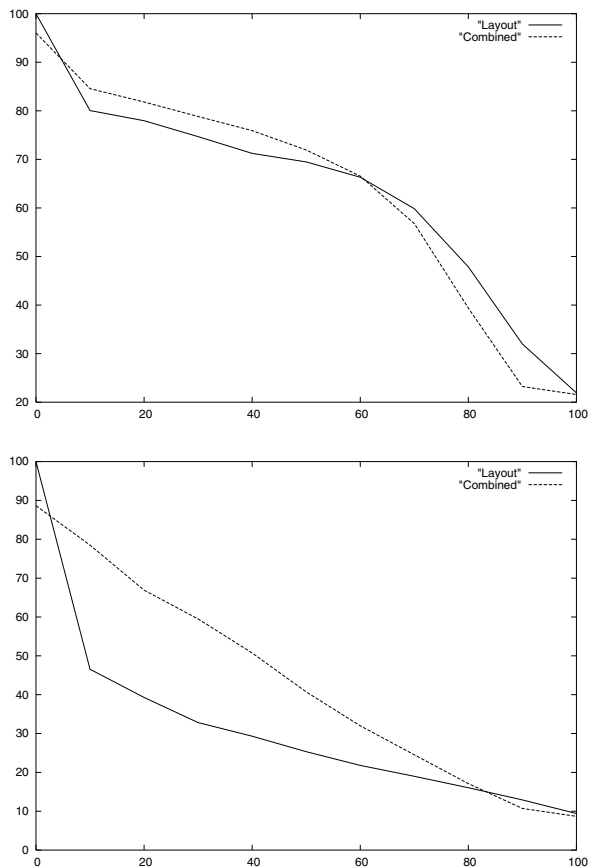


Figure 21. Precision-recall plots related to the identification of pages containing a figure. Top: class “ImageText2”. Bottom: class “Text2Image”.

be larger than higher) and to the mutual position of the two words (“Fig.” has to be on the left of the other keyword).

We performed a retrieval test with each of the four approaches just described and we looked for 16 keywords (with a total of 26 figures related to them). At the end we averaged the Precision-Recall plots that have been obtained for each of the 16 queries obtaining the plot in Figure 22. The continuous line corresponds to the most complex conditions (number 4 in the previous list), and it is not surprising that it provides better results with respect to the other approaches. The intermediate plot corresponds to conditions number 3, whereas the last plot corresponds to the simplest approach (we just look for words in the same page). The

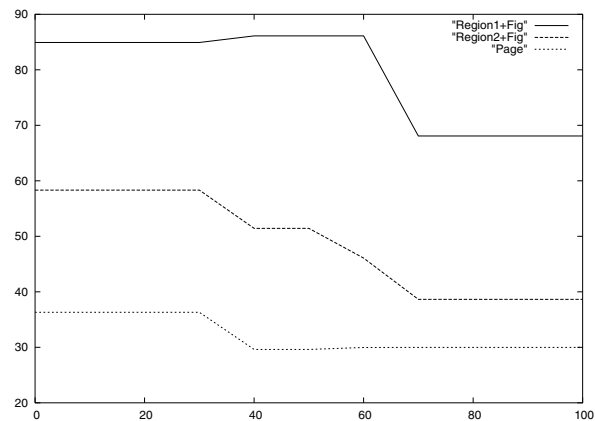


Figure 22. Precision-recall plot for retrieving figures related to a subject (captions containing a keyword).

plot related to condition number 2 (both words in a region) is similar to the page plot, but it is always over it (the difference is only 1% so it is not possible to see both in the Figure).

8.4. Execution time

A document retrieval system can be effectively employed by a user only if the response times are not too high. The performance of the proposed system cannot be simplified in a few values, since there are several factors that affect the overall execution time. In this section we briefly analyze the main steps involved in the system and report, for each of them, the user time. By *user time* we mean the time elapsed from the submission of a command to its completion. The machine considered in our experiments is a Personal Computer with a CPU AMD Athlon with a CPU frequency of 1600 MHz, and equipped with 512 MBytes of RAM. The operating system is a *Debian Linux*, and the experiments have been made without other heavy processes running.

As explained in Section 3 the two main steps are the indexing and the retrieval of documents. Table 1 summarizes the *user time* for the principal tools in the indexing phase. The most expensive tools are those directly involving the image. In particular the location of connected components and the location of horizontal and vertical ruling lines is the most expensive step. The other heavy task is word location and encoding that is based on the segmentation of individual characters and their encoding by means of the trained SOM. It is worth to remember that the SOM train-

Main tool	Execution time	Notes	Reference
Location of connected components and horizontal / vertical lines	10 sec/page		Section 4.1
MXY Tree building	2.33 sec/page		Sections 4.2, 4.3
Layout encoding	12 sec	Overall time for 1287 MXY trees	Section 5.1
SOM Training	98 sec	training with 10 pages	Section 6.1
Word location and encoding	6.28 sec/page		Section 6.2

Table 1. Execution time for the main tools during the indexing. Overall we processed 1287 pages containing 706,768 words.

ing is made once for each collection, and it involves a few pages. For instance for the experiments reported in Section 8 we trained the SOM with the words contained in 10 pages only.

From a user point of view the most critical task is the retrieval one, whereas the indexing time is less important. The *user time* for several retrieval strategies described in the paper is reported in Table 2. During layout retrieval the query page is compared with the MXY trees corresponding to the whole dataset of 1287 by means of Eq. 8. The word retrieval is composed by two main steps: prototype generation by \LaTeX and word matching. The latter step allows also to retrieve the word position in the page as well. It is useful to point out that the first time is fixed and does not depend on the number of stored words. The straightforward combination of layout and word retrieval only requires 7 seconds, that are the sum of the individual processing times. Not surprisingly, the most expensive task is the retrieval that takes into account the combination of some word searches and the retrieval of MXY trees from the "MXY tree DB" (as described in Section 7.2).

9. Conclusions

In this paper we described a general system for the retrieval of document images belonging to digital libraries by looking to image features only. There are three main contributions of the paper.

First, we described a new segmentation algorithm that is appropriate for dealing with digitized books and journals. The method is an extension of the classic MXY tree algorithm and allows to cut regions by looking for changes in font attributes.

Second, we proposed two general approaches for integrating two algorithms previously presented for the retrieval of document images based on the layout and on the presence of user-defined keywords. These integrated approaches have been tested with several experiments dealing with scanned pages of two books.

Third, we described the overall system by showing the relationships among different parts, and the flexibility of the overall architecture to future expansions.

Although a large effort has been made in the last years for building this system, the work is not yet concluded. Concerning the layout retrieval some improvements are expected to be achieved by using tree-grammars for query expansion in order to take into account layout variations. Concerning the word retrieval there are some problems to be solved in the case of heavily broken and touching characters.

The development of this system has been possible for the help provided by several people. We would like to acknowledge the contributions of M. Ardinghi, L. Buccheri, D. Cristofani, M. Lastri, S. Matucci, P. Tanganelli, for their programming contributions. We are grateful also to the partners of the EU-funded METAe project for the fruitful project discussions that were the seed for some of the ideas behind this system.

References

- [1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, 1999.
- [2] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," in *IEEE Trans. PAMI*, vol. 22, pp. 1349–1380, December 2000.
- [3] F. Cesarini, S. Marinai, and G. Soda, "Retrieval by layout similarity of documents represented with MXY," in *Document Analysis Systems V* (S. V.-L. 2423, ed.), pp. 353–364, 2002.
- [4] S. Marinai, E. Marino, and G. Soda, "Indexing and retrieval of words in old documents," in *Proc. 7th ICDAR*, pp. 223–227, 2003.
- [5] D. Doermann, "The indexing and retrieval of document images: A survey," *Computer Vision and Image Understanding*, vol. 70, pp. 287–298, June 1998.
- [6] D. Doermann, J. Sauvola, H. Kauniskangas, C. Shin, M. Pietikainen, and A. Rosenfeld, "The development of a

Main tool	Execution time	Notes	Reference
Layout retrieval	1 sec/query		Section 5.2
Word retrieval	6 sec/query	Made by two steps:	Section 6.3
Prototype generation	3 sec	Constant time	
Word sorting	3 sec	Time for 706,768 words	
Combining layout and words	7 sec/query		Section 7.1
Mutual position of words	48 sec/query		Section 7.2

Table 2. Execution time for the main retrieval strategies. The time reported is the time needed to answer to one query.

- general framework for intelligent document image retrieval,” in *Document Analysis Systems*, pp. 605–632, 1996.
- [7] C. Shin and D. Doermann, “Classification of document page images based on visual similarity of layout structures,” in *SPIE 2000*, pp. 182–190, 2000.
- [8] J. Hu, R. Kashi, and G. Wilfong, “Comparison and classification of documents based on layout similarity,” *Information Retrieval*, vol. 2, pp. 227–243, May 2000.
- [9] J. F. Cullen, J. J. Hull, and P. E. Hart, “Document image database retrieval and browsing using texture analysis,” in *Proc. 4th ICDAR*, pp. 718–721, 1997.
- [10] F. Cesarini, M. Gori, S. Marinai, and G. Soda, “Structured document segmentation and representation by the modified X-Y tree,” in *Proc. 5th ICDAR*, pp. 563–566, 1999.
- [11] A. F. Smeaton and A. L. Spitz, “Using character shape coding for information retrieval,” in *Proc. 4th ICDAR*, pp. 974–978, 1997.
- [12] C. L. Tan, W. Huang, Z. Yu, and Y. Xu, “Imaged document text retrieval without OCR,” in *IEEE Trans. PAMI*, vol. 24, pp. 838–844, June 2002.
- [13] D. P. Lopresti, “Robust retrieval of noisy text,” in *Proc. of ADL’96*, pp. 76–85, 1996.
- [14] K. Taghva, J. Borsack, and A. Condit, “Evaluation of model-based retrieval effectiveness with OCR text,” *ACM TOIS*, vol. 14, pp. 64–93, January 1996.
- [15] A. Takasu, “Document filtering for fast approximate string matching of erroneous text,” in *Proc. 6th ICDAR*, pp. 916–920, 2001.
- [16] G. Navarro, “A guided tour to approximate string matching,” *ACM computing surveys*, vol. 33, no. 1, pp. 31–88, 2001.
- [17] A. Guttman, “R-tree: a dynamic index structure for spatial searching,” in *Proc. Proc. ACM SIGMOD*, pp. 47–57, 1984.
- [18] S. Berchtold, D. A. Keim, and H.-P. Kriegel, “The X-tree: an index structure for high-dimensional data,” in *Proc. Proc. 22nd VLDB*, pp. 28–39, 1996.
- [19] T. Kohonen, *Self-organizing maps*. Springer Series in Information Sciences, 2001.
- [20] W. Y. Arms, *Digital Libraries*. MIT Press, 2000.
- [21] A. R. Kenney and O. Y. Rieger, *Moving Theory into Practice - Digital Imaging for Libraries and Archives*. Research libraries group, 2000.
- [22] G. Nagy and S. Seth, “Hierarchical representation of optically scanned documents,” in *Proc. 7th ICPR*, pp. 347–349, 1984.
- [23] G. Nagy and M. Viswanathan, “Dual representation of segmented technical documents,” in *Proc. 1st ICDAR*, pp. 141–151, 1991.
- [24] M. Viswanathan, “Analysis of scanned documents - a syntactic approach,” in *Structured Document Image Analysis*, pp. 115–136, Springer-Verlag, 1992.
- [25] J. Ha, R. Haralick, and I. Phillips, “Recursive x-y cut using bounding boxes of connected components,” in *Proc. 3th ICDAR*, pp. 952–955, 1995.
- [26] F. Cesarini, M. Lastris, S. Marinai, and G. Soda, “Page classification for meta-data extraction from digital collections,” in *Proc. DEXA 2001*, pp. 82–91, 2001.
- [27] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [28] J. Trenkle and R. Vogt, “Word recognition for information retrieval in the image domain,” in *SDAIR*, pp. 105–122, 1993.
- [29] F. Cesarini, M. Gori, S. Marinai, and G. Soda, “INFORMys: A flexible invoice-like form reader system,” *IEEE Transactions on PAMI*, vol. 20, pp. 730–745, July 1998.
- [30] S. Baldi, S. Marinai, and G. Soda, “Using tree grammars for training set expansion in page classification,” in *Proc. 7th ICDAR*, pp. 829–833, 2003.