Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Layout Analysis and Content Enrichment of Digitized Books

Costantino Grana · Giuseppe Serra · Marco Manfredi · Dalia Coppi · Rita Cucchiara

Received: date / Accepted: date

Abstract In this paper we describe a system for automatically analyzing old documents and creating hyper linking between different epochs, thus opening ancient documents to young people and to make them available on the web with old and current content. We propose a supervised learning approach to segment text and illustration of digitized old documents using a texture feature based on local correlation aimed at detecting the repeating patterns of text regions and differentiate them from pictorial elements. Moreover we present a solution to help the user in finding contemporary content connected to what is automatically extracted from the ancient documents.

Keywords Document Layout Analysis · Page Segmentation · Content Based Retrieval

1 Introduction

Digitized documents play an important role in the preservation of historical contents and in their diffusion to the general public. Without digital editions the huge amount of old archives and documents would not be easily accessible. Digitization allows a pervasive diffusion and also the augmentation of masterpieces with multimedia details and appealing contents. Though, the huge amount of historical archives and books make desirable that their annotation and analysis were automatic and require the minimum users intervention.

Pattern recognition and machine learning offer significant tools for automatically analyzing the content of digitized documents and improving their

C. Grana, G. Serra, M. Manfredi, D. Coppi, R. Cucchiara
Università degli Studi di Modena e Reggio Emilia
Dipartimento di Ingegneria "Enzo Ferrari"
Tel.: +390592056265
Fax: +390592056129
E-mail: {name.surname}@unimore.it

presentation. If Optical Character Recognition (OCR) methods almost yield completely reliable results, the task of identifying textual regions and separate them from other components of the page is more challenging especially in old documents. State of the art methods for Document Layout Analysis and segmentation have been proposed, and, among them, one of the most important is represented by Tesseract [1]. This OCR engine, sponsored by Google, not only offers a powerful tool for text recognition, but also provides layout analysis and image recognition modules. Despite the satisfactory results achieved on contemporary documents, the analysis of historical archives is still challenging due to the high variability of possible contents.

Moreover the large availability of pictorial material from the web can be a key element in providing updated pictorial content to old documents, in that it is possible to link how an historical site or monument was hundreds of years ago with how it is nowadays. This form of hyper linking between different times may provide an interesting way of opening ancient documents to young people and to make them available on the web with old and current content. For this to be possible, automatic or semi-automatic tools must be available given that manual labor is not feasible without advanced support.

In this paper we describe a system for accomplishing this goal. We propose a supervised learning approach to segment text and illustration of digitized old documents. We employ the autocorrelation matrix to derive a texture feature based on local correlation, improved with respect to previous approaches in two ways. Firstly we show how to use of the Fast Fourier Transform, in order to reduce the complexity of the original definition [30]; secondly, we describe the autocorrelation matrix, not only with the directional histogram, but also with the vertical and horizontal projections, which prove to be quite discriminative, in differentiating from text lines. Moreover we present a solution to help the user in finding contemporary content connected to what is automatically extracted from the ancient documents. Fig. 1 presents a schematization of the proposed work flow. Given a document image, the system automatically extracts text, images and their associated captions using a SVM classifier trained on texture features based on local correlation. To enrich the manuscripts with new related contents, extracted images and keywords contained in their captions are used to retrieve similar images from the web.

The paper is structured as follows: in Section 2 we briefly discuss the state of the art in Document Layout Analysis (DLA) and scene understanding, Section 3 explains the main steps of our proposal, the page segmentation, feature extraction and classification. Section 4 describes in detail the features encoding method we propose. Section 5 provides a novel image feature description, which does not rely on codebook training. Finally, in Section 6 we report the performance evaluation and compare it against the state of the art, followed by conclusions and future work in Section 7.

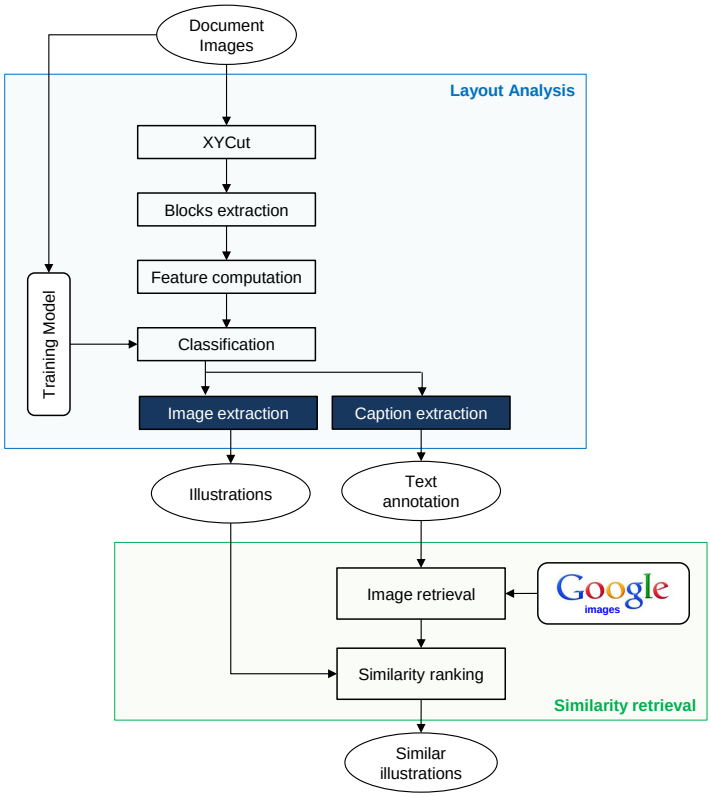


Fig. 1 A summary of the proposed approach to analyze documents and automatically enrich them with similar images searched on the web.

2 Related Work

Document layout analysis is an active area of research and a vast number of works have been presented in the literature. The focus of the problem is often the segmentation of text regions and the subsequent Optical Character Recognition (OCR) step of both printed and handwritten text, but approaches dealing also with pictures segmentation have been studied.

Chen *et al.* [2] give a comprehensive survey on document image classification dividing it in three main components: the problem statement, the classifier architecture and the performance evaluation, separately analyzing each component:

- The *problem statement* is related to the set and type of documents to be analyzed.
- The *classifier architecture* is the core of the problem and includes the aspects related to page segmentation, feature representation and classification.

– *Performance evaluation* is also a critical and important component of a document classifier. The challenge is often the variety of documents considered, either fixed layout documents (books or forms) and flexible layout (newspapers) that inevitably produce different sets of classes as possible outputs. To this aim, a system for ground truthing a large amount of documents and a flexible XML schema has been introduced in [3].

The page segmentation problem can be further decomposed in *Geometrical Layout Analysis* and *Logical Layout Analysis*. The former step is solely based on the geometric characteristics of the document image and aims at finding homogeneous content portions, while the purpose of the latter is to segment the page image into a hierarchy of regions based on the human- perceptible meaning of the content. Regions are therefore assigned a logical label (*e.g.* title, caption, paragraph) and a logical relation among the regions is determined (*e.g.* reading order, inclusion in the same article).

The geometrical analysis approaches can be categorized into top-down, bottom- up or mixed segmentation approaches. Top-down methods, such as XY cuts [4, 5] or methods that exploits white streams [6, 7] or projection profiles [8] are usually fast but tend to fail when dealing with complex layouts. Bottom-up methods are instead more flexible and process the image page from the pixel level and subsequently aggregate into higher level regions but with an higher computational complexity. These approaches are usually based on mathematical morphology, Connected Components (CCs), Voronoi diagrams [9–11] or run-length smearing [12].

Many other methods exist which do not fit exactly into either of these categories: the so called mixed or hybrid approaches try to combine the high speed of the top-down approaches with the robustness of the bottom-up ones. Chen *et al.* [13] proposes a method based on whitespace rectangles extraction and grouping: initially the foreground CCs are extracted and linked into chains according to their horizontal adjacency relationship; whitespace rectangles are then extracted from the gap between horizontally adjacent CCs; progressively CCs and whitespaces are grouped and filtered to form text lines and afterward text blocks. Lazzara *et al.* [14] provides a chain of steps to first recognize text regions and successively non-text elements. Foreground CCs are extracted, then delimiters (such as lines, whitespaces and tab-stop) are detected with object alignment and morphological algorithms. Since text components are usually well aligned, have a uniform size and are close to each other, the authors propose to regroup CCs by looking for their neighbors. Filters can also be applied on a group of CCs to validate the link between two CCs.

Once homogeneous regions are extracted, the other important subtask is the classification of the regions into a set of logical predefined classes (*e.g.* text blocks, tables, drawings, photos, etc.). The XY tree representation is a commonly used approach for describing the physical layout of the documents: it is used in [15] with Hidden Tree Markov Models to perform classification, and in [6] and [16] with decision trees and a KNN based classifier, respectively. Feature vectors composed by a combination of different features are

also common. Wang *et al.* [17] propose fixed length feature vectors composed by a combinations of run length encoding, correlation of text lines, spatial and area features. Meng *et al.* [18] suggest a combination of projection histograms and crossings number histograms.

The basic component of all object recognition and scene understanding systems are local descriptors [19]. The most famous and effective ones are SIFT [20], and all their color variations [21].

After describing images with unordered sets of local descriptors, we would like to directly compare them in order to get information on the images similarities. The problem could be tackled with solutions inspired by the assignment problem, but this would be infeasible as soon as we move away from tiny problems. For this reason, research has focused on finding a fixed length summary of local descriptors density distribution.

The original solution, named Bag of Words, consists in finding a set of *codewords* (obtained by the *k-means* algorithm) and assigning each local feature to a codeword. The final descriptor is given by a histogram counting the number of local features assigned to every codeword (cluster center) [22]. This last strategy was later referred to as *hard-assignment*.

A histogram is obviously a crude representation of the local features continuous density profile, it introduces quantization errors and it is sensitive to noise and outliers [23]. Thus, it would appear that by improving this density representation to more accurately represent the input feature set the classifiers performance could be improved as well [24]. For example, in [25] the hard-assignment of features is replaced with soft-assignment, which distributes an appropriate amount of probability mass to all codewords, depending on the relative similarity with each of them. Many techniques have focused on improving the local descriptors encoding, relying on training data for codewords generation.

In order to overcome the dataset dependency, some authors tried to build a codebook in a fully data-independent way. In [26] the feature space is directly discretized using a regular lattice. With four subdivisions for each dimension, the number of bins is in the order of 10^{77}, most of which are obviously empty. They thus employ a hash table and store only the non-empty bins. Constant time table lookup, i.e., independent of the size of the visual vocabulary, can then be guaranteed. In [27] it is shown that this fixed quantization method performs significantly worse then other techniques, probably due to the fact that it splits dense regions of the descriptor space arbitrarily along dimension axes, and the bins do not equally split the unit hypersphere which SIFT covers, resulting in a wildly uneven distribution of points. Moreover they further highlight on Oxford [28] and Paris [29] datasets that the performance on drop of quantization approaches when generating codewords from a dataset and using them on another. In short, referring to a configuration as dataset1/dataset2 (meaning that codewords are generated by dataset1 and used them for retrieval on dataset2), the Oxford/Oxford combination provides a mAP value of 0.673, against a Paris /Oxford mAP of 0.494.

A different strategy was proposed in [24], in order to avoid codeword generation completely and in this way intrinsically remove any dataset dependency. The idea is to first model each set of vectors by a probability density function (pdf) and then compare them with a kernel defined over the pdfs. The advantage of modeling each image's set of descriptors independently are that each image model is tailored to the specific descriptor set and hence should be more accurate. This solution received little attention, because of the need of using specific kernels for image comparison, again posing scalability issues.

We propose to follow this latter way of modeling local features distributions, by taking advantage of the properties of the Multivariate Gaussian Distribution. By employing a suitable projection, detailed in Section 5, we can transform the distribution to a vectorial representation which allows to use the dot product to closely approximate a distance between distributions. In this way we are able to provide a comprehensive summary of an unordered set of features, considering also the correlation between the different local descriptors dimensions and without introducing any dataset dependency.

3 Page Layout Segmentation

We approach the page segmentation problem starting from the idea that textual and pictorial regions in a document are characterized by different local patterns: lines of text exhibit a regular structure which can be exploited to successfully differentiate them from illustrations in a classification framework. The method we propose can be decomposed in the steps depicted in Fig. 2. The geometric layout analysis is performed by extracting the main regions from the page using the XY cut segmentation, then each region is divided in small squared blocks of size n, and local correlation features are computed on each block and classified using a Support Vector Machine.

The XY cut is a well known recursive algorithm for top-down page segmentation. The method works by firstly projecting the pixels' values on the vertical and horizontal axis of the image and subsequently by finding a low density region of the projection histograms, *i.e.* by finding the white spaces of the page. In this way the page is recursively segmented in rectangular regions.

We used the recursive XY cut with a preprocessing phase of binarization and morphological closure on the page in order to filter out the white interline spaces. The closure is performed with a squared structuring element of size 41×41 pixels. At each iteration the white borders surrounding the regions are removed before calling the next recursive step to find the internal cuts. Algorithm 1 provides the pseudo code of our approach to recursive XY cuts. By exploiting this algorithm, we obtain a segmentation of the page, usually corresponding to the two columns of text or parts of them, and, if existing, full page images.

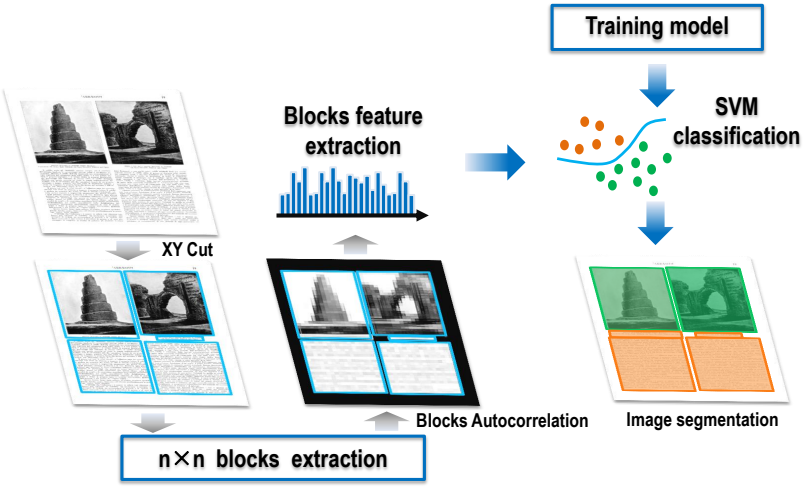


Fig. 2 System overview of the proposed Page Layout Segmentation algorithm.

4 Local Correlation Features

In order to distinguish between textual and pictorial regions we used a texture analysis method similar to the one proposed in [30] and [31]. The approach exploits the autocorrelation matrix, an effective feature for finding repeating patterns which is particularly suited in this case since textual textures have a pronounced orientation that heavily differs from that of illustrations. The autocorrelation function is defined as the cross correlation of a signal with itself, and represents a measure of similarity between two signals. Once applied to a grayscale image, it produces a central symmetric matrix, that gives an idea of the degree of regularity of the texture. The methods consists in the subdivision of the original image into square blocks of size n. The formal definition of the autocorrelation of a block is:

$$C(k,l) = \sum_{y=\max(0,l)}^{n-1+\min(0,l)} \sum_{x=\max(0,k)}^{n-1+\min(0,k)} I(x,y) \cdot I(x+k, y+l) \tag{1}$$

where l and k are defined in $[-n/2, n/2]$. It is important that the size n of the blocks is chosen such that the repeating pattern of the textual blocks is highlighted.

Implementing the autocorrelation following this definition gives an algorithm with a high computational complexity, roughly proportional to n^4. According to the cross-correlation theorem the cross-correlation of two signals is equal to the product of the Fourier Transform of each one, where one of them has been complex conjugated.

$$f \star g \Leftrightarrow F' \cdot G \tag{2}$$

Algorithm 1 Recursive XY Cut (RXYC)

Input: *image*
Output: *list* ▷ list of regions on the page

```
Binarize(image)
MorphologicalClosure(image)
RXYC_Step(image)

procedure RXYC_Step(ROI)
    Remove white borders
    vProj ← vertical projection (sum of rows values)
    hProj ← horizontal projection (sum of columns values)
    hCut ← first y such as vProj(y) < Thresh
    if hCut then                                          ▷ Horizontal cut found
        ROI ← ROI
        ROI.h ← hCut
        RXYC_Step(ROI)                                    ▷ XYCut on the first sub-image
        ROI.y ← ROI.y + hCut
        ROI.h ← ROI.h − hCut
        RXYC_Step(ROI)                                    ▷ XYCut on the second sub-image
    else
        vCut ← first x such as hProj(x) < Thresh
        if vCut then                                      ▷ Vertical cut found
            ROI ← ROI
            ROI.w ← vCut
            RXYC_Step(ROI)                                ▷ XYCut on the first sub-image
            ROI.x ← ROI.x + vCut
            ROI.w ← ROI.h − vCut
            RXYC_Step(ROI)                                ▷ XYCut on the second sub-image
        else
            list ← list ∪ ROI
        end if
    end if
end procedure
```

where F and G denote the transformed signals and F' is the complex conjugate of F. Since the autocorrelation is a special case of the cross-correlation, Eq. 2 becomes:

$$f \star f \Leftrightarrow F' \cdot F = |F|^2 \tag{3}$$

and for the Wiener-Khinchin theorem, the Fourier Transform of an autocorrelation function is the power spectrum, or equivalently, the autocorrelation is the inverse Fourier transform of the power spectrum. Following these properties, we efficiently compute the autocorrelation of the blocks only using two steps of the Fast Fourier Transform (FFT) with to a reduction of the complexity from $\mathcal{O}(N^4)$ of the naive approach to $\mathcal{O}(N \log N)$.

The result of the autocorrelation can be employed to extract an estimate of the relevant directions within the texture. Usually, the autocorrelation matrix is encoded with a *directional histogram*, a polar representation in which each direction is determined by an angle $[0°, 360°)$ and the bin value is given by the

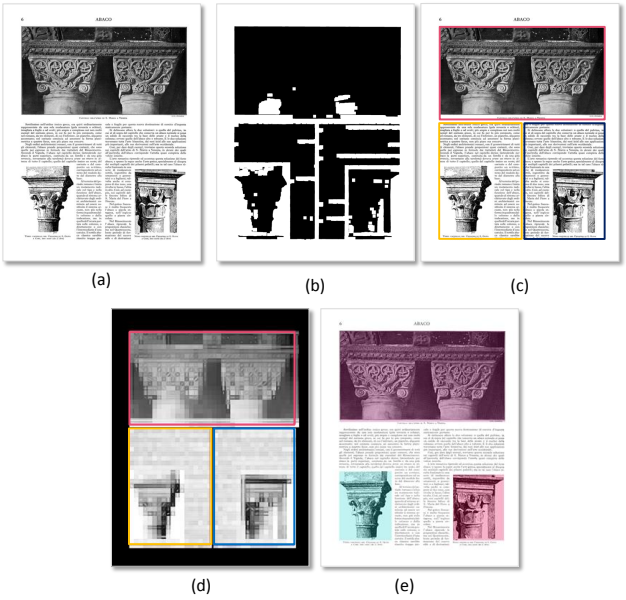
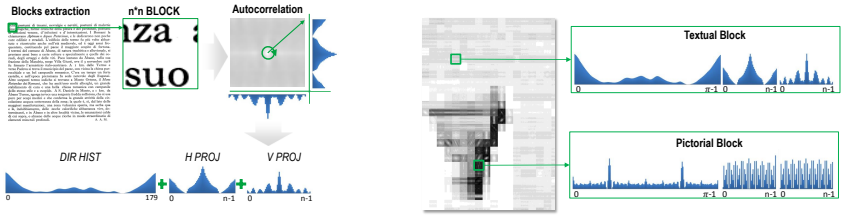


Fig. 3 Sequence of step: (a) Input image, (b) Closure, (c) XY-Cut, (d) Local correlation features, (e) Final illustration segmentation.



(a) Descriptors computation. (b) Feature vectors.

Fig. 4 (a) Feature vectors computation from an image block. The descriptor is the concatenation of the directional histogram (DIR HIST) with the projections of the autocorrelation matrix along the horizontal (H PROJ) and vertical directions (V PROJ). (b) Image showing the autocorrelation on every block of a page and example feature vectors obtained from a text area and from an illustration.

sum of the pixels along that direction.

$$w(\theta) = \sum_{r \in (0, n/2]} C(r\cos\theta, r\sin\theta) \tag{4}$$

Since the autocorrelation matrix has a central symmetry by definition, we consider only the first half of the direction histogram in the range $[0°, 180°)$. ω and r are quantized: the step of ω is set to $1°$ and the step of r is set to 1 pixel. Using this encoding a text block is characterized by peaks around $0°$

and 180° because of the horizontal dominant direction, conversely an image block is described by a generic multi-modal distribution.

We finally enrich the descriptor concatenating the directional histogram with the projections of the autocorrelation matrix along the vertical and horizontal directions in order to enhance the repeating pattern of the text lines. Fig. 4 provides a visual summary of the feature extraction process and few example results.

5 Image description with Gaussians of Local Descriptors

Every image extracted from the documents should be described with a feature vector. We follow the common trend of summarizing it with an unordered set of local features, in our case SIFT descriptors. In order to provide a tractable description of the inherently unknown pdf of an unordered set of feature vectors, we employ the most classical parametric distribution, that is the multivariate Gaussian distribution. Let $F = \{\mathbf{f}_1 \dots \mathbf{f_N}\}$ be the set of d-dimensional local features and suppose that they are independent and identically distributed samples from a multivariate Gaussian distribution, defined as

$$\mathcal{N}(\mathbf{f}; \mathbf{m}, \mathbf{C}) = \frac{1}{|2\pi\mathbf{C}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{f}-\mathbf{m})^T \mathbf{C}^{-1}(\mathbf{f}-\mathbf{m})}, \tag{5}$$

where $|\cdot|$ is the determinant, $\mathbf{m}$ is the mean vector and $\mathbf{C}$ is the covariance matrix; $\mathbf{f}, \mathbf{m} \in \mathbb{R}^d$ and $\mathbf{C} \in \mathbb{S}_{++}^{d\times d}$, and $\mathbb{S}_{++}^{d\times d}$ is the space of real symmetric positive semi-definite matrices. The mean and covariance parameters are estimated from F as follows:

$$\mathbf{m} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}_i, \tag{6}$$

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N} (\mathbf{f}_i - \mathbf{m})(\mathbf{f}_i - \mathbf{m})^T. \tag{7}$$

The estimated covariance matrix encodes information about the variance of the features and their correlation, and, together with the mean, provides a good insight on the set of features F. The space of covariance matrices can be formulated as a differentiable manifold, but not as a vector space (e.g. the covariance space is not closed under multiplication with a negative scalar). Unfortunately, many efficient machine learning algorithms assume that the data points form a vector space where dot product is defined, therefore they cannot readily work with covariance matrices.

It is important to consider that a manifold is a topological space that is locally similar to a Euclidean space. In particular a Riemannian manifold is a differentiable manifold in which each tangent space has an inner product, which varies smoothly from point to point [32].

Recently, it has been shown by Pennec *et al.* [33] that it is possible to endow the space of covariance matrices with an affine-invariant Riemannian

metric (thus defining a Riemannian manifold), which allows to map covariance matrices to points in the Euclidean space.

The first step is the projection of the covariance matrices on an Euclidean space tangent to the Riemannian manifold, at a specific tangency matrix $\mathbf{P}$. The second step is the extraction of the orthonormal coordinates of the projected vector. In the following, matrices (points in the Riemannian manifold) will be denoted by bold uppercase letters, while vectors (points in the Euclidean space) by bold lowercase ones.

More formally, the projected vector of a covariance matrix $\mathbf{C}$ is given by:

$$\mathbf{t_C} = \log_{\mathbf{P}}(\mathbf{C}) \triangleq \mathbf{P}^{\frac{1}{2}} \log\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{C}\mathbf{P}^{-\frac{1}{2}}\right)\mathbf{P}^{\frac{1}{2}} \tag{8}$$

where log is the matrix logarithm operator and $\log_{\mathbf{P}}$ is the manifold specific logarithm operator, dependent on the point $\mathbf{P}$ to which the projection hyperplane is tangent. The matrix logarithm operators of a matrix C can be computed by eigenvalue decomposition ($\mathbf{C} = \mathbf{U}\mathbf{D}\mathbf{U}^T$); it is given by:

$$log(\mathbf{C}) = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k}(\mathbf{C} - \mathbf{I})^k = \mathbf{U}log(D)\mathbf{U}^T. \tag{9}$$

The orthonormal coordinates of the projected vector $\mathbf{t_C}$ in the tangent space at point $\mathbf{P}$ are then given by the vector operator:

$$\mathrm{vec}_{\mathbf{P}}(\mathbf{t_C}) = \mathrm{vec}_{\mathbf{I}}\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{t_C}\mathbf{P}^{-\frac{1}{2}}\right) \tag{10}$$

where $\mathbf{I}$ is the identity matrix, while the vector operator on the tangent space at identity of a symmetric matrix $\mathbf{Y}$ is defined as:

$$\mathrm{vec}_{\mathbf{I}}(\mathbf{Y}) = \left[y_{1,1}\ \sqrt{2}y_{1,2}\ \sqrt{2}y_{1,3} \ldots y_{2,2}\ \sqrt{2}y_{2,3} \ldots y_{d,d}\right]. \tag{11}$$

Substituting $\mathbf{t_C}$ from Eq. 8 in Eq. 10, the projection of $\mathbf{C}$ on the hyperplane tangent to $\mathbf{P}$ becomes

$$\mathbf{c} = \mathrm{vec}_{\mathbf{I}}\left(\log\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{C}\mathbf{P}^{-\frac{1}{2}}\right)\right). \tag{12}$$

Thus, after selecting an appropriate projection origin, every covariance matrix is projected to an Euclidean space. Since $\mathbf{c}$ is a symmetric matrix of size $d \times d$ a $(d^2 + d)/2$-dimensional feature vector is obtained.

As observed in [34], by computing the sectional curvature of the Riemmanian manifold [35], i.e., the natural generalization of the classical Gaussian curvature for surfaces, it is possible to show that this space is almost flat. This means that the neighborhood relation between the points on the manifold remain unchanged, wherever the projection point $\mathbf{P}$ is located. Therefore, from a computational point of view, the best choice for $\mathbf{P}$ is the identity matrix, which simply translates the mapping into applying the $\mathrm{vec}_{\mathbf{I}}$ operator to the standard matrix logarithm. This also frees us from the problem of optimizing

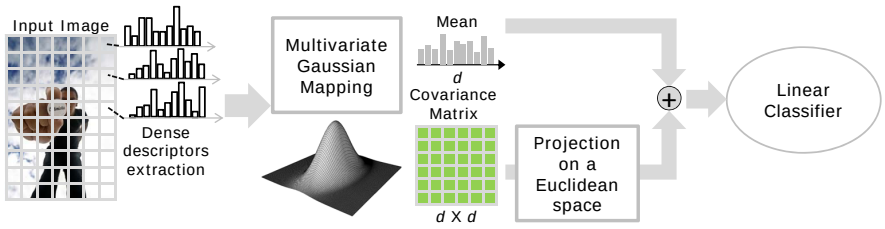


Fig. 5 An image is represented as a Weighted Pyramid of Gaussians of local descriptors. The covariance matrix is projected on the tangent space and concatenated to the mean to obtain the final region descriptor.

the projection point for the specific data under consideration, leading to a generally applicable descriptor.

Finally, the unordered set of feature vectors F can be described by a Gaussian of local descriptors (GOLD), that is the concatenation of the mean and the orthonormal projection of the covariance matrix. A summary of the proposed approach is presented in Fig. 5.

In image classification systems, feature normalization techniques have the potential to greatly decrease the error rate of the classification, and thus increase the overall performance. When dealing with classifiers relying on dot-product (such as linear SVMs) there is some recent convergence on the combined use of power normalization and unit length normalization using a L_2 metric [36, 37].

Power normalization consists in applying, to each dimension of the descriptor, the function:

$$f(x) = sign(x)|x|^{\alpha} \quad \text{with } 0 < \alpha < 1. \tag{13}$$

Perronnin *et al.* [36] justify the use of power normalization with the empirical observation that it has the ability of "unsparsifying" the representation, making it suitable for dot-product similarity. A different interpretation is provided in [38] where it is shown that applying the square root (a special case of the power normalization with $\alpha = 0.5$) is equivalent to employ the Hellinger's kernel (Bhattacharyya's coefficient). Moreover Safadi and Quénot [39] tested different normalization approaches and distance measures on several image descriptors, and observed that power normalization consistently leads to better performance. Moreover they optimized the α parameter for every descriptor and distance combination, and concluded that the optimal value when using dot product is approximately 0.5.

Motivated by these results, we apply power normalization to the GOLD vector, with $\alpha = 0.5$. While α optimization could slightly improve the performance, it would lead to a dataset-dependent tuning, again in contrast with our purposes.

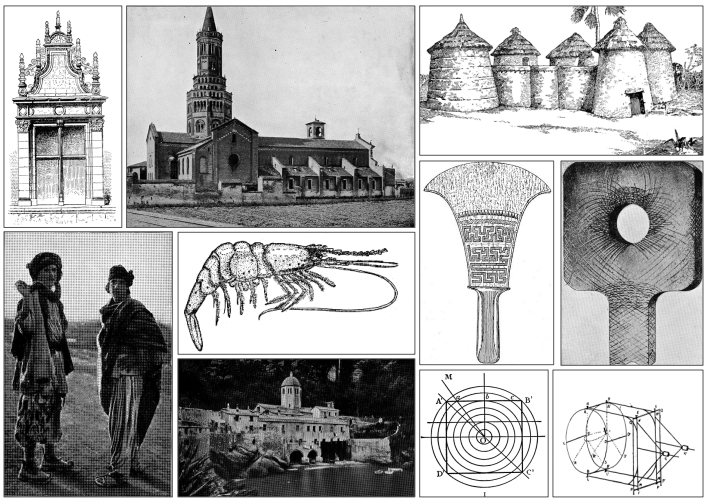


Fig. 6 Sample images from the Treccani dataset

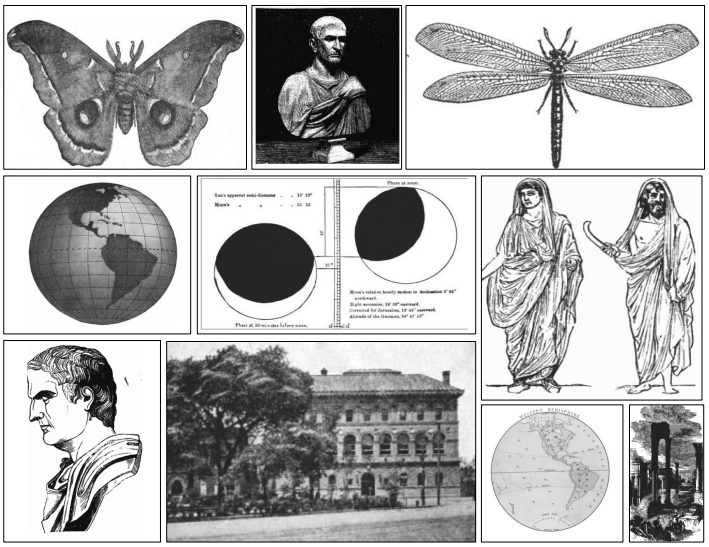


Fig. 7 Sample images from the Gutenberg13 dataset

6 Experiments

In this section we report the results obtained on the digitized pages of the "Enciclopedia Treccani"[1]. The encyclopedia consists of a set of volumes firstly published between 1925 and 1936. In our evaluation we only considered the first volume which is composed of 1183 pages. The original size of each digitized

[1] `http://www.treccani.it`

Table 1 Blocks classification results on the Treccani dataset, obtained with different features extracted from the autocorrelation matrix. The values are the percentage of image blocks classified as image (TP), as text (FN) or text blocks classified as image (FP).

	% of image blocks		
Feature	**TP**	**FN**	**FP**
Dir. Hist.	96.38	3.62	5.27
Dir. Hist. + H. Proj.	98.25	1.75	4.79
Dir. Hist. + V. Proj.	98.41	1.59	4.92
Dir. Hist. + H. Proj. + V. Proj.	99.42	0.58	4.68

page is 22110×28819 pixels at 1 bpp, and we used a downscaled version with a factor of 0.125 along each dimension and converted to gray -scale so that the new images are 2763×3602 pixels at 8 bpp. The pages are two-column text with a number of illustration per page that may vary from 0 to 10. The main difficulty of this dataset, compared to other document analysis datasets, is the variety of graphical elements that is not limited to photos, but also includes drawings and charts. The overall number of manually annotated illustrations is 1157.

For a more comprehensive analysis we also built the Gutenberg13 dataset [2]. This dataset was created using a set of publicly available e-books from Project Gutenberg[3]. The e-books have been converted to grayscale with a resolution of 300 pixel/inch printing 4 pages for sheet, resulting in a two column text layout with a font size similar to the Treccani dataset. The total number of pages is 268 and each page is 2481×3509 pixels. Some example images of the two datasets are provided in Fig. 6 and in Fig. 7.

6.1 Layout Segmentation

In our experiments we set the block size to 64×64 pixels, enough to consider a line of text. Recalling that our descriptor is the concatenation of the vertical and horizontal projection of the autocorrelation matrix and its directional histogram, we obtained a 308 dimensional feature vector for each block. We used an SVM with RBF kernel whose C and γ parameters have been estimated using cross validation (the values $C = 4096$ and $\gamma = 0.5$ have been used). The training set consists of 4000 blocks randomly sampled half from text regions and half from image regions.

Table 1 reports the results on the Treccani dataset, including the directional histogram, the two projections and their combination, and shows how both of the projections contain indeed useful information, which can be leveraged by the SVM classifier.

[2] The dataset is available online at: `http://imagelab.ing.unimore.it/imagelab/CHIRetrieval/Gutenberg13.zip`

[3] `http://www.gutenberg.org`

Table 2 Comparison between our method based on local correlation, the same method with cross testing and Tesseract on the Treccani and Gutenberg13 datasets. Results are reported in terms of number of TP, FN, FP. The values are the number of images of the three classes and the percentage of image pixels.

		Number of images			% image pixels		
Dataset	**Method**	**TP**	**FN**	**FP**	**TP**	**FN**	**FP**
Treccani	**Our Method**	361	5	132	99.57	0.43	4.53
	Tesseract	200	166	16	52.28	47.72	0.39
Gutenberg13	**Our Method**	524	11	62	99.23	0.77	2.39
	Our Method XT	461	20	64	91.30	8.70	2.66
	Tesseract	486	40	9	83.13	16.87	1.02

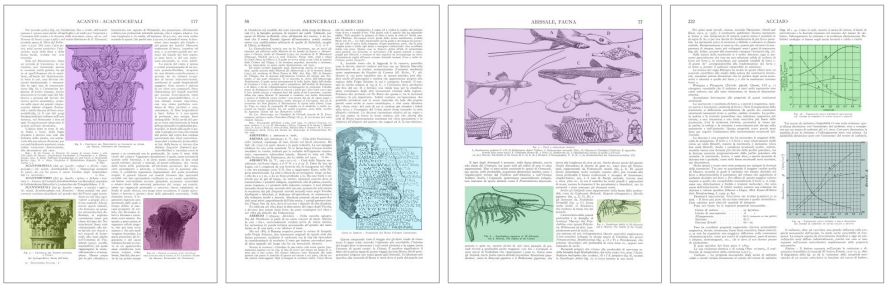
We compared our proposal with the results given by the layout analysis module of Tesseract [4], the Optical Character Recognition engine sponsored by Google. The Tesseract image detection algorithm, follows a classical approach used by most commercial OCR engines, that is a sequence of morphological binary operations. In particular the algorithm starts with a Halftone Region Extraction. This consists in a $8\times$ downsample, setting pixels only if the number of set pixels in the block is large; then an opening with 5×5 square structuring element is performed, followed by a reconstruction of the image by a $8\times$ upsample. In this way, small elements (most notably all characters) are removed and a set of seeds for possible images is found. This is then followed by a closing step (with a 4×4 structuring element) and a 4-connected binary seed fill operation. A final 8-connected binary seed fill is employed to fix the border pixels. This approach alone would be sensible to elongated lines and structures in the printed page, so another set of aggressive downsample and upsample operations takes care of removing the elongated lines. Since the image-find function of Tesseract is essentially based on morphological operations, in order to maximize Tesseract performance we threshold the pages to the maximum level of 255, meaning that everything which is not white is considered as black.

In order to evaluate the performance we solved the matching problem between the ground truth and the results obtained by the two methods exploiting the *Hungarian Method*. The nodes of the bipartite graph correspond to the GT bounding boxes and to the automatically extracted bounding boxes, while the edges are weighted using a measure of the overlap between bounding boxes. Given two bounding boxes l and r, the weight is computed as:

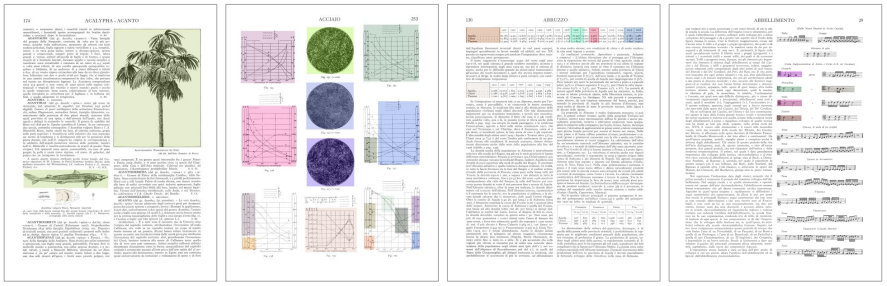
$$w_{lr} = \frac{l \cap r}{l \cup r} \tag{14}$$

Results are reported in terms of the number of True Positives (TP), False Negatives (FN) and False Positives (FP). Table 2 shows the performance comparison of our method and Tesseract on the Treccani and Gutenberg13 datasets,

[4] `https://code.google.com/p/tesseract-ocr`



(a) Correct segmentation results.

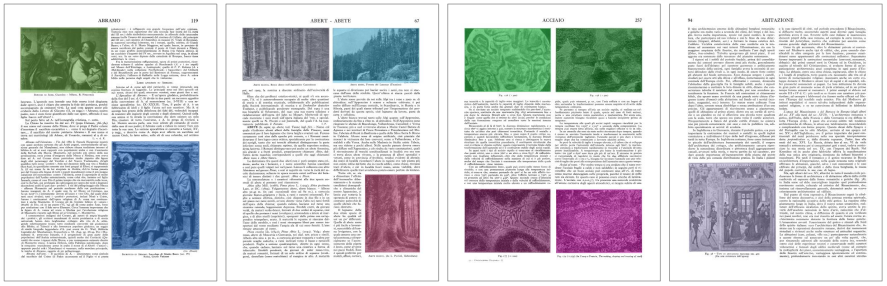


(b) Wrong segmentation results.

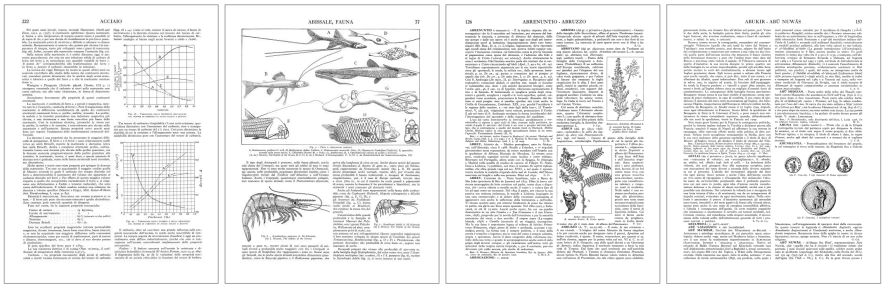
Fig. 8 Illustration segmentation obtained with our method. (a) Photographs, draws and charts correctly segmented. (b) Examples of oversegmented regions and wrong detections.

reporting, in the first block of columns, the number of images of the three classes (TP, FN, FP). The second block of columns, instead, focuses on the performance in terms of percentage of pixels with respect to the total number of pixels annotated as illustration.

As Table 2 shows, our method outperforms Tesseract on the Treccani dataset. Tesseract has indeed the main drawback of not being able to recognize most of the drawings although it makes a good job in finding photographs. Our proposal, instead, exploiting a supervised classification approach, has the capability of learning the correct classification also of drawings and charts. The experiment also demonstrates how our method produces a higher number of false positives, but we would like to highlight that despite the high number of images, false positive results correspond to small areas as shown in terms of percentage of pixels. These errors usually correspond to portions of music sheets, tables or drawings as displayed in Fig. 8. Some examples of illustration segmentations obtained with Tesseract, that highlight the difficulty in extracting drawings and charts, are reported in Fig. 9. In order to further evaluate the reliability of our proposal and the robustness of the model learnt by the Support Vector Machine, we performed a cross-testing. Specifically we used the model learnt on the Treccani dataset to test the segmentation on the Gutenberg dataset. The results, reported in Table 2, show how our proposal is generic and applies to different types of documents provided the similar



(a) Good segmentation.



(b) Bad segmentation.

Fig. 9 Illustration segmentation obtained with Tesseract Layout Analysis module. (a) Examples of photographs correctly segmented. (b) Charts and drawings not detected as illustrations.

structure, and that does not over fit on the specific training data. Despite the better performance achieved when training and testing on the same dataset (more than 99% of true positive images segmented), Table 2 reports how, even without a re-training phase, more than the 90% of the pictorial elements are correctly identified.

6.2 Similar Images Web Search

Once a picture and its relative caption has been extracted from a document, similar images can be retrieved from the web, simply using the caption as the query string. Web search engines like Google Images produce very noisy results and, usually, a lot of unintended content is retrieved. Filtering out visually unrelated images is therefore crucial.

To provide a quantitative evaluation of our descriptor w.r.t. other popular image retrieval solutions, we created a dataset suitable for image retrieval evaluation with focus on Cultural Heritage Imaging [5]. The dataset is composed of images downloaded from Google Images and representing objects/scenes

[5] The dataset is available online at: `http://imagelab.ing.unimore.it/imagelab/CHIRetrieval/CHIdataset.zip`

Table 3 Comparison of image retrieval results using our approach and other state of the art approaches. Mean Average Precision is reported.

Category	$GOLD$	$SIFT_{512}$	$SIFT_{1024}$	$SIFT_{4096}$	RGB_{hist}
church	**0.502**	0.194	0.229	0.249	0.216
statue	0.120	0.071	0.110	**0.121**	0.115
manuscript	**0.604**	0.340	0.491	0.520	0.158
capitel	**0.427**	0.259	0.153	0.192	0.120
rose window	**0.626**	0.041	0.294	0.341	0.326
mosaic	**0.337**	0.043	0.194	0.210	0.110
altar	**0.364**	0.350	0.205	0.220	0.245
inscription	**0.256**	0.056	0.181	0.180	0.109
bell tower	0.127	0.150	0.162	**0.171**	0.114
archaeological sites	**0.343**	0.223	0.244	0.263	0.218
city square	**0.396**	0.075	0.172	0.188	0.175
musical instruments	0.286	0.043	0.279	0.297	**0.438**
bridge	0.223	**0.250**	0.223	0.230	0.165
crown	0.266	0.066	0.277	0.279	**0.296**
Average	**0.348**	0.154	0.230	0.247	0.202

belonging to 14 categories: *church, statue, manuscript, capitel, rose window, mosaic, altar, inscription, bell tower, archaeological sites, city square, musical instruments, bridge* and *crown*. For each category we downloaded approximately 700 images: 5 representative images are chosen as queries, approximately 140 are used for testing and the rest is used to compute the codebooks needed by Bag of Words approaches. This leads to 70 queries, 1894 testing images and 7631 images for codebook generation.

For each query we sort all the testing images using cosine similarity and report the Mean Average Precision (MAP). We compare our solution with 2 other methods: Bag of Words on SIFT using different codebook sizes and a color histogram on RGB values computed using 512 bin obtained with K-means (referred as $SIFT_N$, where N is the number of clusters, and RGB_{hist} in the following). For all the methods we extract dense descriptors with step of 3 pixels computing the descriptors on a Spatial Pyramid of 1×1, 2×2, 1×3 following [40]. Results are reported in Table 3.

The GOLD descriptor outperforms the other solutions in the majority of classes, proving its efficacy for image retrieval. As expected, the Bag of Words approach on SIFT is more effective the more cluster are computed but performance tends to saturate after a certain number of clusters. The color histogram solution obtains very good results only for classes with a very specific color like *crowns* and *musical instruments*.

To show some qualitative results, we selected a subset of images from the Treccani dataset and we have automatically downloaded about 500 images from Google Images, using the automatically extracted caption as the query string. The GOLD descriptor, introduced in Section 5, is used to visually describe the retrieved images and the corresponding query. Cosine similarity is again used to sort the results, so as to present the user only a small amount of relevant pictures, showed in Fig. 10. Google Images is usually able to re-

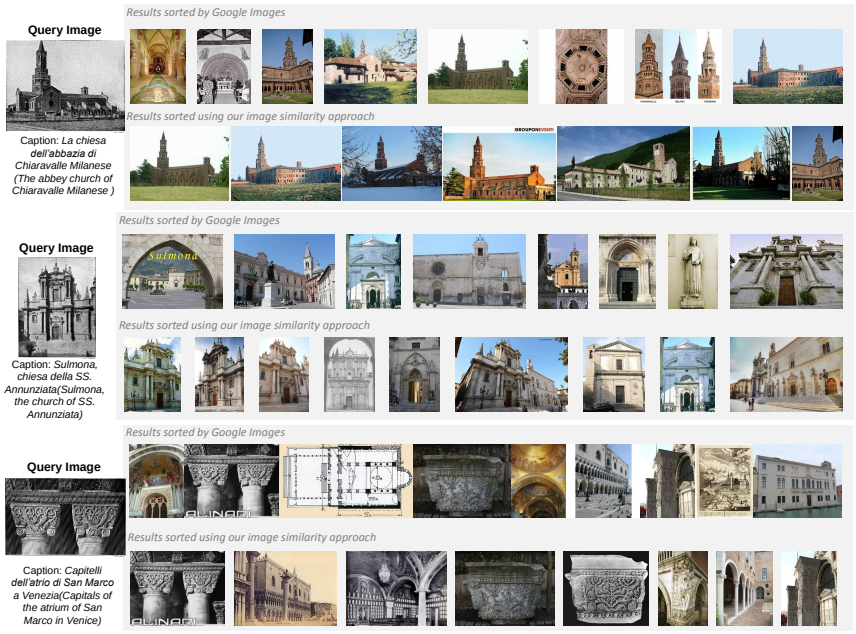


Fig. 10 Similar images web search: the query image is placed on the left, the first few images given by Google Images when querying the image caption are sorted using the proposed approach. The results of the system are satisfactory when querying with the first two images, the last row instead, shows a failure case, where a lot of unrelated content is also presented to the user.

trieve, within the first 500 images, the exact object we are looking for, but the amount of unrelated images is still too high. Visually sorting the retrieved images deeply impacts the quality of the first few images presented to the user, and a high quality color image depicting the object of interest is often available. Sometimes the caption used as the online search query contains "strong" keywords that produce biased results. For example, the third query of Fig. 10 does not produce satisfactory results, in which the majority of images depict San Marco square, and not the capitals cited in the caption.

7 Conclusion

We proposed a method to automatic extract illustrations from digitized documents. Starting from the assumption that text regions are characterized by a strong horizontal repeating pattern we introduced a descriptor based on the autocorrelation of squared blocks that has demonstrate to be effective even with drawing and charts. We compared our method with Tesseract and we showed it is able to detect challenging illustration also when the state-of-the-art fails. Our proposal represents an useful tool for a future enrichment of historical manuscripts with renovated contents. Starting from the appearance

of the extracted images, and eventually exploiting the keywords contained in their captions, is in fact possible to automatically retrieve similar images, for example from the web.

References

1. Smith R (2007) An Overview of the Tesseract OCR Engine. In: International Conference on Document Analysis and Recognition (ICDAR), Washington, DC, USA, pp 629–633.
2. Chen N, Blostein D (2007) A Survey of Document Image Classification: Problem Statement, Classifier Architecture and Performance Evaluation. Int J Doc Anal Recogn 10:1–16.
3. Clausner C, Pletschacher S, Antonacopoulos A (2011) Aletheia - An Advanced Document Layout and Text Ground-Truthing System for Production Environments. In: International Conference on Document Analysis and Recognition (ICDAR), pp 48–52.
4. Ha J, Haralick R, Phillips I (1995) Recursive X-Y cut using bounding boxes of connected components. In: International Conference on Document Analysis and Recognition (ICDAR), volume 2, pp 952–955 vol.2.
5. Cesarini F, Lastri M, Marinai S, et al (2001) Encoding of modified X-Y trees for document classification. In: International Conference on Document Analysis and Recognition (ICDAR), pp 1131–1136.
6. Appiani E, Cesarini F, Colla A, et al (2001) Automatic document classification and indexing in high-volume applications. Int J Doc Anal Recogn 4:69–83.
7. Pavlidis T, Zhou J (1991) Page Segmentation by White Streams. In: International Conference on Document Analysis and Recognition (ICDAR), pp 945–953.
8. Esposito F, Malerba D, FA L, et al (2000) Machine Learning for Intelligent Processing of Printed Documents. Journal of Intelligent Information Systems 14:175–198.
9. Kise K, Sato A, Iwata M (1998) Segmentation of Page Images Using the Area Voronoi Diagram. Comput Vis Image Understand 70:370–382.
10. Agrawal M, Doermann D (2009) Voronoi++: A Dynamic Page Segmentation Approach Based on Voronoi and Docstrum Features. In: International Conference on Document Analysis and Recognition (ICDAR), pp 1011–1015.
11. Winder A, Andersen T, Smith E (2011) Extending Page Segmentation Algorithms for Mixed-Layout Document Processing. In: International Conference on Document Analysis and Recognition (ICDAR), pp 1245–1249.
12. Sebastiani F, Ricerche CND (2002) Machine Learning in Automated Text Categorization. ACM Computing Surveys 34:1–47.
13. Chen K, Yin F, Liu C (2013) Hybrid Page Segmentation with Efficient Whitespace Rectangles Extraction and Grouping. In: International Conference on Document Analysis and Recognition (ICDAR), pp 958–962.
14. Lazzara G, Levillain R, Geraud T, et al (2011) The SCRIBO Module of the Olena Platform: A Free Software Framework for Document Image Analysis. In: International Conference on Document Analysis and Recognition (ICDAR), pp 252–258.
15. Diligenti M, Frasconi P, Gori M (2003) Hidden Tree Markov Models for Document Image Classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 25:2003.
16. Baldi S, Marinai S, G S (2003) Using treegrammars for training set expansion in page classification. In: International Conference on Document Analysis and Recognition (ICDAR), pp 829–833.
17. Wang Y, Phillips I, Haralick R (2006) Document zone content classification and its performance evaluation. Pattern Recogn 39:57–73.
18. Meng G, Zheng N, Song Y, et al (2007) Document Images Retrieval Based on Multiple Features Combination. In: International Conference on Document Analysis and Recognition (ICDAR), volume 1, pp 143–147.
19. Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. IEEE Trans on Pattern Anal Mach Intell 27:1615–1630.

20. Lowe DG (2004) Distinctive Image Features from Scale-Invariant Keypoints. Int J Comput Vis 60:91–110.
21. van de Sande KEA, Gevers T, Snoek CGM (2010) Evaluating Color Descriptors for Object and Scene Recognition. IEEE Trans on Pattern Anal Mach Intell 32:1582–1596.
22. Csurka G, Dance CR, Fan L, et al (2004) Visual categorization with bags of keypoints. In: Statistical Learning in Computer Vision Workshop, pp 1–12.
23. Bao BK, Zhu G, Shen J, et al (2013) Robust Image Analysis With Sparse Representation on Quantized Visual Features. IEEE Trans Image Process 22:860–871.
24. Farquhar J, Szedmak S, Meng H, et al (2005) Improving "bag-of-keypoints" image categorisation: Generative Models and PDF-Kernels. Technical report, University of Southampton.
25. van Gemert JC, Geusebroek JM, Veenman CJ, et al (2008) Kernel Codebooks for Scene Categorization. In: Proceedings of the European Conference on Computer Vision(ECCV), pp 696–709.
26. Tuytelaars T, Schmid C (2007) Vector Quantizing Feature Space with a Regular Lattice. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp 1–8.
27. Philbin J, Chum O, Isard M, et al (2008) Lost in quantization: Improving particular object retrieval in large scale image databases. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp 1–8.
28. `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`.
29. `http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/`.
30. Journet N, Ramel J, Mullot R, et al (2008) Document image characterization using a multiresolution analysis of the texture: application to old documents. Int J Doc Anal Recogn 11:9–18.
31. Grana C, Borghesani D, Cucchiara R (2010) Automatic segmentation of digitalized historical manuscripts. Multimedia Tools and Applications :1–24.
32. Tuzel O, Porikli F, Meer P (2008) Pedestrian Detection via Classification on Riemannian Manifolds. IEEE Trans on Pattern Anal Mach Intell 30:1713–1727.
33. Pennec X, Fillard P, Ayache N (2006) A Riemannian Framework for Tensor Computing. Int J Comput Vis 66:41–66.
34. Martelli S, Tosato D, Farenzena M, et al (2010) An FPGA-based Classification Architecture on Riemannian Manifolds. In: DEXA Workshops, pp 215–220.
35. Chavel I (2006) Riemannian Geometry: A Modern Introduction. Cambridge Studies in Advanced Mathematics, Cambridge University Press.
36. Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. In: Proceedings of the European Conference on Computer Vision(ECCV), pp 143–156.
37. Chatfield K, Lempitsky V, Vedaldi A, et al (2011) The devil is in the details: an evaluation of recent feature encoding methods. In: British Machine Vision Conference, pp 76.1–76.12.
38. Vedaldi A, Zisserman A (2012) Efficient Additive Kernels via Explicit Feature Maps. IEEE Trans on Pattern Anal Mach Intell 34:480–492.
39. Safadi B, Quenot G (2013) Descriptor optimization for multimedia indexing and retrieval. In: International Workshop on Content-Based Multimedia Indexing, pp 65–71.
40. Lazebnik S, Schmid C, Ponce J (2006) Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp 2169–2178.