An Automatic Word-spotting Framework for Medieval Manuscripts

Ruggero Pintus
CRS4, Italy

Ying Yang
Yale University, United States

Enrico Gobbetti
CRS4, Italy

Holly Rushmeier
Yale University, United States

***Abstract*—We present a completely automatic and scalable framework to perform query-by-example word-spotting on medieval manuscripts. Our system does not require any human intervention to produce a large amount of annotated training data, and it provides Computer Vision researchers and Cultural Heritage practitioners with a compact and efficient system for document analysis. We have executed the pipeline both in a single-manuscript and a cross-manuscript setup, and we have tested it on a heterogeneous set of medieval manuscripts, that includes a variety of writing styles, languages, image resolutions, levels of conservation, noise and amount of illumination and ornamentation. We also present a precision/recall based analysis to quantitatively assess the quality of the proposed algorithm.**

I. Introduction

In the last decade many museums and libraries around the world have undertaken a massive digitization of vast corpora of valuable handwritten historical documents, ranging from old Greek texts to modern manuscripts. Well known examples are George Washington's papers at the Library of Congress and Isaac Newton's papers at the University of Cambridge Library. Besides the goal of Cultural Heritage (CH) preservation, the aim is also to make this data available to scholars, curators, and the general public. To provide access and facilitate navigation of such on-line databases, each captured document must be labeled and indexed. Traditionally, the standard way to perform this task is to manually create a structured meta-data file for each book or collection. Automation is desirable, since manual indexing is very expensive, and becomes impractical as the size of repositories grows. An important, basic task that is needed is the automatic recognition of characters or words, to perform labeling, indexing, or transcription.

Unfortunately, although state-of-the-art techniques for Optical Character Recognition (OCR) are very powerful when applied to printed text, they are highly likely to fail on historic handwritten documents, which are in general significantly degraded due to ink bleed-through, faded ink, stained paper, and other aging factors. In addition, the high degree of variability in handwriting, and the noise in such old manuscripts, has forced humanities researchers to transcribe those pages by hand. Thus, automatic handwriting recognition still remains a very challenging problem for the Computer Vision (CV) community.

In this scenario, word-spotting approaches try to overcome the limits of the aforementioned text reading methods, by relying on image matching metrics and more general object recognition frameworks. Word-spotting considers a collection of documents as a collection of word images, and tries to group similar word occurrences into clusters. This class of algorithms enables different technical solutions for document analysis, e.g. a word retrieval system that requires neither a previous transcription [1], nor a partial transcription tool that helps scholars in a semi-automatic, incremental environment. The power and reliability of word-spotting have been proved in many application domains, ranging from automatic mail sorting in modern handwriting, to historical document retrieval given a single word image, on-line searching in CH collections, and book integration within a digital library [2].

There are many classes of handwritten manuscripts, each with different features to be exploited and challenges to overcome. As a result, there is no "one size fits all" method that is optimal for all manuscripts. Our project, in collaboration with humanist scholars, focuses on Western medieval manuscripts. These have the advantage of regular, horizontal text lines. However, they have the disadvantage of being primarily composed of thick vertical strokes that are only slightly modified to differentiate characters, making recognition tasks peculiarly difficult. There are huge numbers of manuscripts of this type, scattered through libraries around the world. There are a large world-wide community of scholars interested in these manuscripts for studying history, law, literature and art. Our work is in an environment in which the manuscripts are being scanned and processed to be made available in a web-based service for a variety of cross-collection studies. Our approach takes advantage of being able to segment manuscripts in a pre-process that can facilitate the many types of study being conducted by our humanist colleagues.

We present a completely automatic and scalable framework to perform query-by-example word-spotting in medieval manuscripts. In particular, we do not require any manual intervention of the user to produce a large amount of annotated data to train the algorithm. Given a set of images from a handwritten book and an image of a query word, the algorithm first segments each page into text lines [3], to remove all the irrelevant *non-text* data. Then, it performs a decolorization that enhances image contrast before feature extraction and image matching. Finally, we adopt, modify and integrate the state-of-the-art approach of Almazán et al. [4], which finds the occurrences of the query word by searching similar regions across each single line separately, based on Histogram of Oriented Gradients (HOG) descriptors. The results are exported in a lightweight, open, data-interchange annotation format [5]. Compared with previous word-spotting systems, the proposed paper and approach provides the following major contributions:

- **Automatic and scalable word-spotting framework.** Our novel automatic word-spotting framework exploits the automatic computation of average text leading value. All parameters are automatically computed, and we do not require any user intervention for a manuscript dependent

tuning. We feed the word-spotting algorithm with the automatically segmented text lines. This allows us to get rid of all non-text data, and reduce the system in-core memory footprint, making the framework, designed to be compliant with the standards of document layout analysis web services, scalable to very large image databases.

- **Cross-manuscripts search.** Relying on text leading information makes the proposed algorithm independent from the size of the text and the image capture resolution. The adaptive nature of our framework allows us to perform cross-manuscript word-spotting, by taking the query from a book and searching for occurrences in another.
- **Evaluation.** In addition to presenting an improvement over the state-of-the-art, we present an extensive study of algorithm performance, to evaluate our pipeline with a heterogeneous corpus content, that includes different classes of writing styles, languages, image resolutions, levels of conservation, noise and amount of illumination and ornamentation. Our precision/recall-based analysis quantitatively assesses the quality of the proposed algorithm.

Although not all the techniques presented here are entirely novel in themselves, their elaboration and combination in a single, completely automatic, unified system is non trivial, and represents a substantial enhancement to the state-of-the-art. The proposed approach allows us to produce a particularly compact and efficient system for document analysis. It is very simple for CV researchers to implement and integrate it into a local or web-based applications (e.g., transcription environment), and it can be a very helpful tool for daily work of scholars, curators and CH practitioners in the field of historic document study and interpretation.

II. Related work

The word-spotting problem has always attracted the interest of the pattern recognition community; the seminal works in the field were in speech recognition [6], while the first application of word-spotting to handwritten text was presented a little later by Manmatha et al. [7]. Since then, the importance of indexing and browsing old handwritten books leads to a numerous interesting contributions [8], [9], making it an active area of research. Space does not allow an exhaustive review of the literature. Instead, we only discuss in detail the state-of-the-art techniques closely related to ours.

Two major classes of handwritten word-spotting approaches are: template-based and learning-based. Template-based approaches compute the similarity between the image of a query word with a set of labeled template images [10]. Although performance is independent of the complexity of the language or the alphabet, it is difficult to extend to a general case (e.g., with an unknown out-of-vocabulary word). Learning-based approaches are more robust and more generalizable [11]–[13]. Rodriguez et al. [14] presented a statistical word-spotting framework, which uses two types of Hidden Markov Models(HMMs) to characterize words, and a Gaussian Mixture Model (GMM) to compute the score normalization. These techniques have been applied to a wide variety of input writing styles. Leydier et al. [15] introduced a text search algorithm designed for ancient books, and, in particular, for medieval handwritten text in Latin and Semitic alphabets. They compute local differential features, and use a cohesive elastic matching method to find similar regions that share the same informative parts. Fischer et al. [16] propose a learning-based word-spotting system that uses character HMMs to find word occurrences in a medieval manuscript. Although they are efficient approaches, the main problem with those methods is that they require a significant manual effort to produce annotated data to train the Hidden Markov Models [17] or the Neural Networks [18]; moreover state-of-the-art evaluations have been carried out with relatively small databases, with a low number of input pages or lines. Conversely, we adopt the recent word-spotting algorithm of Almazán et al. [4], and we modify it to make it adapt to the text size and the acquisition resolution of the input. Their approach is particularly efficient, since it does not need any user defined a-priori segmentation or annotated data. It takes the query word, automatically computes the learning model, and outputs the word occurrences in the book, and it is based on the use of compressed HOG descriptors, and on Exemplar Support Vector Machines (SVMs) coupled with Stochastic Gradient Descent (SGD) solvers. This method requires the user to define the best cell size for HOG computation, which depends on the text character dimension and the image capture resolution. By exploiting the text leading value computation, which gives us an estimation of text size in pixel, we modify the original Almazán's technique to make the choice of cell size completely automatic and, more important, adaptive wrt the input manuscript. Compared to the current state-of-the-art methods, this automatic, adaptive behavior improves the applicability of our framework, by further enabling a cross-manuscript word-spotting. Moreover, we propose to feed the word-spotting algorithm with the automatically segmented text lines [3]. By getting rid of all *non-text* data (background, parchment areas, figures and ornamentation), we are able to drastically reduce the in-core memory footprint of the original pipeline. This modification makes the framework more scalable to work with larger image sets.

As in many vision techniques, the vast majority of previous methods require a pre-processing of the input images, to prepare and adapt the signal to particular algorithmic requirements. A lot of image-matching based pipelines apply a color space conversion, and in particular a dimensionality reduction to luminance space. A simple conversion might remove some important gradient-based features that are key for matching, so more advanced techniques are needed. Color-to-gray conversion can be performed locally or globally. Locally, the algorithm considers iso-color pixels differently to enhance local contrast [19]–[21]. The main drawback of local approaches is that they could produce a strong appearance bias in constant color regions, which might be converted inhomogeneously [22]; this makes local methods unsuitable for our pattern recognition task. A global decolorization technique is preferable [22], [23], which takes an image as a whole, and applies a color-to-gray mapping function to all the pixels. In particular, we rely on the global decolorization method presented by Grundland et al. [24], which is very efficient in terms of computational time, and proved to be very suitable for matching or recognition algorithms when gradient-based features are extracted [25]. We modify Grundland's original approach to better fit our problem. The algorithm automatically computes the average text leading for each page [26], and we use this value to drive the decolorization, since it is proportional to the size of the relevant color contrast

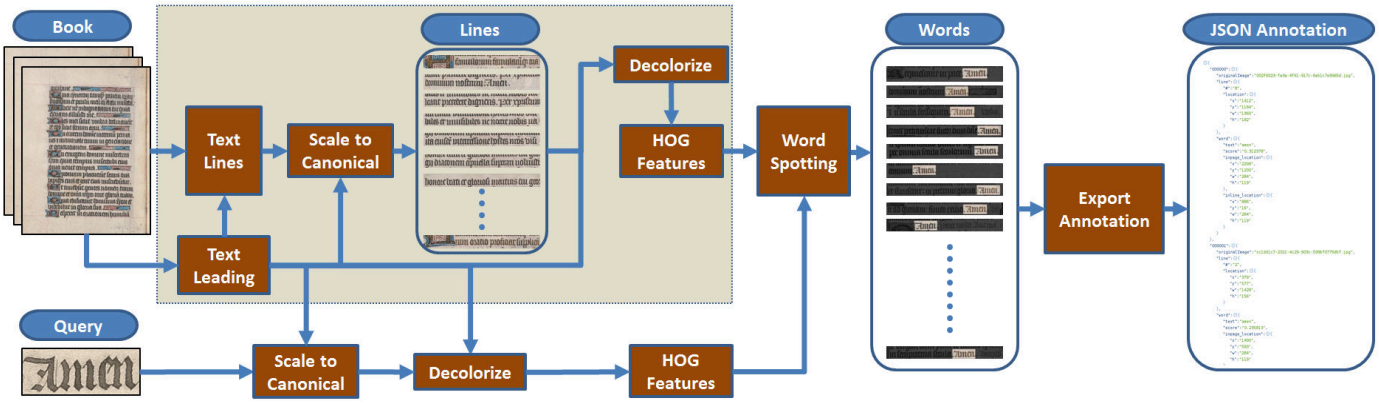


Fig. 1. **Pipeline Overview.** Our pipeline is completely automatic, and it does not require any user intervention or user defined parameter. Operations in the shaded region are performed only once per book, independent of the number of queries. The input are a book and a query image. First we segment the book in text lines. We use the average text leading to drive the decolorization step and the word-spotting algorithm. We export retrieved words data as a text file in the JavaScript Object Notation (JSON) format.

features we want to preserve after the color-to-gray conversion. Although Grundland's method was recently used for printed text recognition in natural images [27], [28], to the best of our knowledge, it is the first time it was tested in an ancient handwriting recognition framework. Further, to be more convenient for scholars, our framework has been designed to be compliant with web systems like the well known Mirador [29] developed by the Stanford University, which is an open source image, Javascript and HTML5 viewer that delivers high resolution images in a workspace that enables comparison of multiple images from multiple repositories. It is fully compatible with the International Image Interoperability Framework (IIIF [30]).

III. Framework Overview

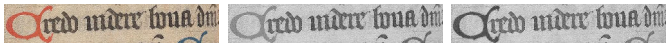
Fig. 1 depicts an overall view of the pipeline of the proposed word-spotting framework. The algorithm is given an entire book as a set of images, and a sub-region from one of those pages that contains the user selected query word (e.g., the word *Amen*). First we automatically compute the average text leading of the book [26], then we apply an automatic segmentation step to extract text lines from the input manuscript [3]. We first scale query and line images to a canonical size. In a decolorization step we convert them to a high contrast monochromatic signal, and we compute their compressed HOG descriptors. *The highlighted part in the figure is executed just once per-book.* Then for each query word the word-spotting algorithm takes those gray-scale images and returns a ranked sequence of lines, which contain the query word, together with the region of the line surrounding that word. The automatically computed text leading value is used to adaptively set parameters in the text line extraction, scaling, decolorization and word-spotting steps. We export annotations about retrieved words in the JavaScript Object Notation (JSON) format, for lightweight data sharing purposes.

IV. Method

Input data. The proposed word-spotting framework has two main inputs: the set of all page images from an handwritten book; the query image, which is a sub-region of one manuscript page that contains the user selected query word. It does not require any limiting constraints on the nature of that data; the layout of the input images may contain portraits, capital letters, figures, ornamentation, overlapping lines and touching characters. The condition of the page is not restricted: the manuscript could be degraded by background noise, bleed-through or faded-out ink, stained, or affected by other kinds of damage and aging. We only rely on two minor assumptions that are typically met in a general scenario, and that are required to apply the per-page text leading computation [26] and the text line segmentation [3]. First, text must be quasi-horizontal, or a pre-processing step (e.g., Papandreu et al. [31]) must be performed to correct its orientation before being used within the framework (note in extending to other languages this requirement could be quasi-vertical, e.g., Chinese language). Generally, this is also a classic choice in the capture setups and the page visualization systems of museums and digital libraries. Secondly, the variability of the per-page text leading (inter-line distance) across the book is assumed to be Gaussian, i.e., it is reliable to model it as a mean value and a variance. Moreover, as opposed to many word-spotting frameworks, we do not need any data previously annotated by the user. Given these assumptions, our pipeline is completely automatic, without any manuscript dependent parameters tuned manually by the user. This enables our framework to be integrated into an unsupervised system, such as a web service. For display purposes only, throughout this section we use and show pages from the book BeineckeMS310 available at the Yale University's Beinecke Rare Book and Manuscript Digital Library [32].

Text leading computation and text line segmentation. We pre-process each input page in order to extract the average text leading [26] T_L for the book. This gives an estimation of the text size, which will be used to drive all the steps in our pipeline, text line extraction, adaptive scaling, image decolorization and word-spotting, making them completely automatic and avoiding any type of user defined input (see Fig. 1 and below). For each page, we extract the lines of text by using the method of Pintus et al. [3]. It first computes a text region segmentation, and extract main text blocks. Then, for each block, it uses a Projection Profiles (PP) approach [33], which has been applied only to text features (SIFT [34]), to robustly extract text lines. In word-spotting applications a pre-segmentation of the data is generally not preferable, since a dataset with a arbitrarily complex layout might results in bad region classification [4]. However, for a huge number

of layouts that are similar to those used in handwritten medieval manuscripts, line retrieval has proven to be very robust and reliable, with an average *Precision* value $> 96\%$, and an average *Recall* value $> 98\%$ [3]. Line retrieval allows us to remove irrelevant data, such as background, figures without text, and ornamentation. Converting the input data from a series of document images to a series of line images also reduces memory occupancy and computational time. By applying an adaptive scaling factor χ to the query image and to all text line images, we transform them into a canonical size. The scaling factor is proportional to the text leading, and in all our experiments we set $\chi = {}^{\tilde{T}_L}/_{T_L}$, where the canonical text leading $\tilde{T}_L = 100$.



(a) (b) (c)

Fig. 2. **Decolorization.** Comparison between color-to-gray conversions: the original RGB color image (a); the luminance (b); the higher contrast signal obtained with the decolorization step (c).

Decolorization. Before passing the query image and all text lines to the word-spotting algorithm, we convert them to a gray-scale signal. Since the word-spotting method is based on HOG features, we would like to obtain a single channel that is more suitable and efficient when coupled with a gradient-based descriptor. We adopt the decolorization technique of Grundland et al. [24]. It is a real-time conversion that proposed a new dimensionality reduction strategy, i.e., Predominant Component Analysis. It results in the estimation of a predominant chromatic axis, which captures, in a mono-dimensional space, the contrast information that would be otherwise lost in the luminance. The decolorization algorithm is controlled by three parameters: the degree of image enhancement (λ), which specifies how much the chromatic contrast will be taken into account, or how much the predominant chromatic channel will influence the final picture (if $\lambda = 0$ the gray-scale signal corresponds with the luminance value); the size of relevant image features in pixels (σ); the proportion of outlier pixels, which is related to image noise (η). As in [24], we set $\lambda = 0.5$, which represents a quite high improvement of the chromatic contrast. Our manuscript images are acquired in a professional, almost noise-free, setup, so in our experiments we always set $\eta = 0.001$. Generally, the default size of color contrast feature is set to $\sigma = \sqrt{2min\left(im\_width, im\_height\right)}$, where im_width and im_height are the number of image columns and rows. However, in our case we have the value of the average text leading T_L (inter-line distance). We modify the original Grundland's approach [24] by exploiting this a-priori knowledge about the text size, and we adaptively set the value of $\sigma = {}^{1}/_{\alpha}T_L$, where α has been estimated in a data-driven manner from several ground truths (see section V). The comparison between color-to-gray conversions in the case of a manuscript image is in Fig. 2, where we show the original color image (Fig. 2(a)), the luminance (Fig. 2(b)), and the higher contrast signal obtained with the decolorization step (Fig. 2(c)). In the simple example, it is very easy to see how the gradient between the capital letter *'C'* in the word *'Credo'* and the background has been increased by the proposed decolorization algorithm.

Word-spotting. The word-spotting algorithm takes as input the gray-scale version of the segmented text lines and the monochromatic converted query image. We modify the

(a) (b) (c) (d) (e) (f)

Fig. 3. **User query variability.** The size of the manual query, which depends on the background discarded by the user (a)-(d), capital letters (f), or ascenders and descenders (e).

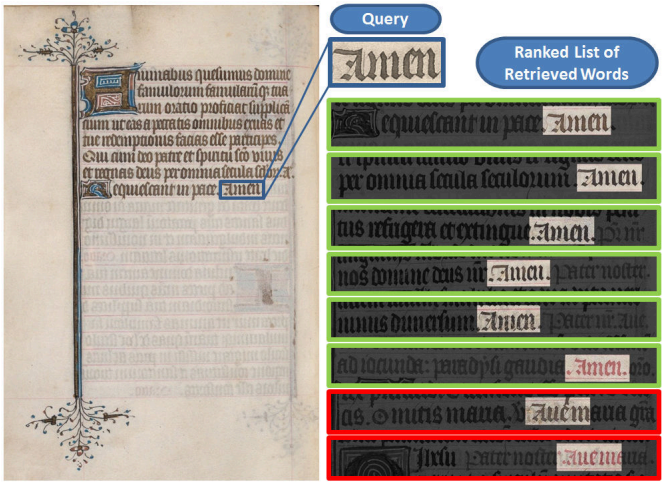


Fig. 4. **Word-spotting.** Example of the word-spotting pipeline applied to the word *'Amen'*. We show one original image with the zoomed version of the query word, and a ranked list of retrieved word occurrences. 'Green' lines are *true positive* outcomes, while 'red' lines are the *false positives*.

approach of Almazán et al. [4] to make it adapt to the text size and the acquisition resolution of the manuscript. To encode local gradient, the query image and the lines are characterized by HOG descriptors. They are particularly reliable for object detection and image retrieval [35]–[37], they are very fast to compute and to compare, and they are very suitable for a search technique based on sliding window. To save memory, the HOGs are compressed by using a combination of Principal Component Analysis (PCA) and Product Quantization (PQ) [38]. These compressed descriptors are used to train a classifier for the query word in an Exemplar Support Vector Machine (Exemplar SVM) framework [39], [40]. One of the most important parameters is the size of the cell for the computation of HOGs, which defines the optimal scale of the most relevant features in the text. This value is crucial to properly drive the machine learning process for a good query classification. In the original paper [4] the authors manually chose a fixed size of 12×12 pixel. Here, we are interested in a computation of the cell size that is completely automatic and independent of both the character size and the manuscript acquisition resolution in pixel. We exploit the computation of the text leading across the book, since the cell size (and so the size of the relevant features) must be correlated with the dimension of the text, or, more precisely, of the character strokes. Undersized cells will capture very variable signal, depending more on image noise or writing variability; oversized cells will produce very similar gradient histograms, which makes it difficult for the classifier to separate the classes and to reach a convergence. Therefore, we modify the algorithm to take into account the text leading value computed in the pre-processing step. Given the canonical text leading $\tilde{T}_L = \chi T_L$, we use a cell size equal to ${}^{1}/_{\alpha}\tilde{T}_L$. The unique and manuscript independent value of the scaling factor α has been computed from several ground truth datasets of medieval manuscripts (Sec. V). Another way to compute an estimation

of the HOG grid cell is to measure the height of the user selected area of the query word. However, this value can vary randomly for several reasons, e.g., the changing ratio between the inter-line distance and the height of characters, the user's decision to discard the background space between text lines (reducing the height of the selected area), or the presence of capital letters, and, more important, ascenders or descenders (Fig. 3). So, given the assumption of a Gaussian text leading distribution across the book [3], relying on an automatic estimation of the text leading is preferable. We also modify the Almazán's technique to exploit the advantages of the text line segmentation. First, we restrict the user selection to lie within the corresponding line; the query word image will be the intersection between the user selected area and the rectangular box of the corresponding line. In this way, we apply a clipping constraint on the manual intervention, and we more properly limit the variability of user input. Moreover, compared with the original method, we also improve performance by feeding the word-spotting pipeline only with relevant image regions, i.e., segmented lines of text. By discarding *non-text* regions such as background, figures or ornamentation, we greatly reduce the memory footprint as well. As an example of the word-spotting results, we selected the query word image *'Amen'* from one page, and the algorithm will return all the lines where the same word occurs (Fig. 4). The 'green' lines are the *true positives*, while the 'red' lines are *false positives*. As it is easy to appreciate, the algorithm wrongly retrieves words such as *'Ave m(aria)'*, whose character sequence has a very similar appearance to the query.


```
{
    "000000" : {
       "originalImage" : "002f6023-fa3a-4f61-917c-6eb1c7e8965d.jpg",
        "line" : {
            "#" : "8",
            "location" : { "x" : "1319", "y" : "1106", "w" : "1270", "h" : "170" }
        },
        "word" : {
            "text" : "amen",
            "score" : "0.286207",
            "inpage_location" : { "x" : "2148", "y" : "1131", "w" : "263", "h" : "107" },
            "inline_location" : { "x" : "829", "y" : "25", "w" : "263", "h" : "107" }
        }
    },
     "000001" : {
        "originalImage" : "b8dbb6ec-c829-45be-88b7-65e355f3541c.jpg",
        "line" : {
            "#" : "3",
            "location" : { "x" : "1030", "y" : "675", "w" : "1363", "h" : "145" }
        },
        "word" : {
            "text" : "amen",
            "score" : "0.228226",
            "inpage_location" : { "x" : "1355", "y" : "700", "w" : "263", "h" : "107" },
            "inline_location" : { "x" : "325", "y" : "25", "w" : "263", "h" : "107" }
        }
    }
}
```


Fig. 5. **JSON annotation.** JSON annotation structure for the first two ranked occurrences of the word *'Amen'*. JSON is an open standard, ideal data-interchange language, which is very suitable for digital library websites.

Annotation. Finally, we export the word-spotting results in a JSON annotation format [5], which includes all the information about the retrieved words. JSON is an open standard, ideal data-interchange language, since it is a text format that is completely independent of any other programming languages. It is based on universal data structures supported by the majority of other programming languages. It is typically employed in data transmission between a server and a web application and so turns out to be very suitable for digital library websites, e.g., to manage user requests and to return areas of a manuscript in response to a query. For this reason it is preferred over XML.

Fig. 5 reports the JSON annotation structure for the first ranked occurrence of the word *'Amen'* in Fig. 4. The first six digit number is the *ranking position*. Then, we have three elements: i.e. *originalImage*, *line*, and *word*. The *originalImage* is the name of the page image in the book containing the occurrence. In *line* we report its numbering in the page (#), and its physical location in the image, i.e., the upper-left corner (x, y) of the rectangular block that contains the line, and its width and height (w, h). Similarly, *word* contains the corresponding ASCII *text*, the computed matching *score* with the query word, and the location (rectangular area) of the retrieved occurrence in page and line coordinates.

V. Results

The algorithm has been tested on medieval manuscripts made available by the Yale University's Beinecke Rare Book and Manuscript Digital Library [32]. Those books are very heterogeneous, in terms of level of conservation (e.g., aging, ink bleed-through, noise), capture resolution, languages, writing styles and the amount of figures and ornamentation. In all contain about 270 pages and about 5000 lines. Our technique was implemented on Linux using MatLab, and tested on a Laptop PC with 4 Intel Core i7-4510U CPU @ 2.00GHz processors, and 8GB RAM. When the dataset is ready after the pre-processing step, which has been done once (highlighted part in Fig. 1), the speed of word-spotting (learning the query word and evaluating possible matches) is critical for integration into a web service. The time our algorithm takes to learn the query model and to evaluate possible occurrences is less than 5 seconds.

The validation of our pipeline in a quantitative way requires some ground truth data. Although some public ground truths exist, e.g., the most used are the George Washington (GW) [8], [41] and the Lord Byron (LB) dataset [42], they are relatively recent and, in the case of Lord Byron, machine printed. No ground truths are available for medieval manuscripts similar to ours, wuth challenges in terms of level of conservation, noise and amount of text and figures in each page. Hence, we manually produced segmented data for ground truth. In generic Information Retrieval evaluation, *Precision* and *Recall* are used. They are single-value metrics, which rely on a set of documents/items returned by the process. In our framework, the pipeline returns a ranked list of word occurrences, and, in order to take into account this ranking, it is preferable to use the *Average Precision* (*AP*) metrics instead. Given n retrieved words, the *AP* is computed as:

$$AP = \frac{\sum_{k=1}^{n} \left(P\left(k\right) \times rel\left(k\right)\right)}{RW} \tag{1}$$

where $P\left(k\right)$ is the *Precision* at cut-off k in the list, $rel\left(k\right)$ is a function equal to 1 if the item at rank k is a relevant word (otherwise 0), and RW is the number of all relevant words.

As mentioned in section IV, the algorithm adaptively set the HOG cell size proportional to the canonical text leading value $\tilde{T}_L$, i.e., ${}^{1}/_{\alpha}\tilde{T}_L$. We need to find a good and reliable estimation of α. In order to do so in a data-driven manner, we use the ground truth datasets, and launch the word-spotting algorithm for a range of α values. For each of them we compute the resulting *AP*. Then, we compute the best value $\hat{\alpha}$ as

$$\hat{\alpha} = \underset{\alpha}{arg\,max}\; f\left(\alpha\right) := \prod_{i=0}^{N} AP_i\left(\alpha\right) \tag{2}$$

TABLE I. WORD-SPOTTING STATISTICS.

Book Name	Word	Relevant Words	True Positives	False Positives	False Negatives	Average Precision
BeiMS310	*'Amen'*	55	51	369	4	84.5%
	'Oratio'	31	31	969	0	74.3%
	'nobis'	113	93	296	20	69.7%
	'pro nobis'	35	30	47	5	78.6%
	'ora pro nobis'	13	10	19	3	76.2%
BeiMS360	*'god'*	121	99	901	22	62.2%
	'lord'	177	155	440	22	82.7%
BodMS_2_11	*'Per'*	45	31	969	14	53.4%
	'psalmus_dd'	39	22	226	17	52.1%
Cross-manuscript						
BeiMS310 → BodMS_2_11	*'Amen'*	33	26	186	7	26.6%
BodMS_2_11 → BeiMS310	*'Oratio'*	32	32	606	0	64.9%
BodMS_2_11 → BeiMS310	*'nobis'*	114	64	8	50	52.3%

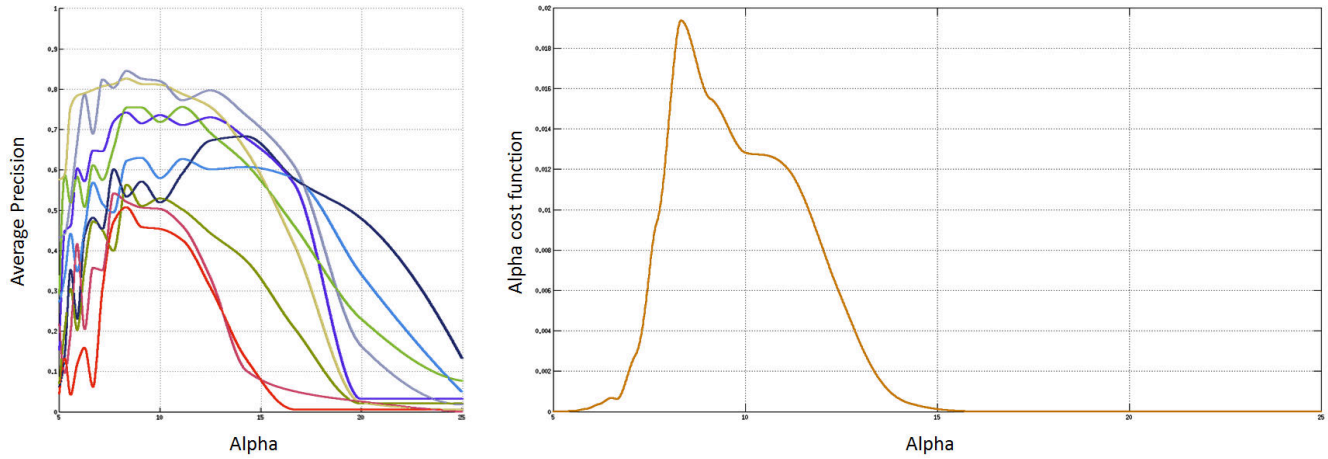
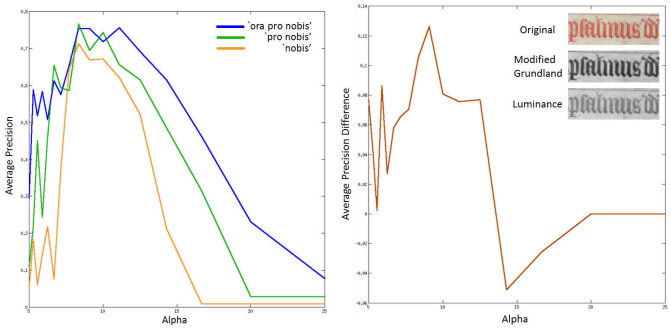


Fig. 6. **HOG cell size estimation.** Given a cell size equal to ${}^1/_{\alpha}T_L$, we show on the left the behavior of the *Average Precision* value wrt the variation of α in the case of input query words in table I. On the right we plot the cost function that we use to estimate the best α value (see eq. 2).

where N is the number of ground truths, and $AP_i(\alpha)$ is the AP function for the ground truth i. In Fig. 6 we show on the left all the $AP_i(\alpha)$ from the words in table I. The sensitivity of the AP wrt α is quite variable between one word and the other. On the right of the Fig. 6 we plot the cost function $f(\alpha)$, with its maximum at $\alpha \simeq 8.33$. We choose an adaptive HOG cell size equal to ${}^1/_8\tilde{T}_L$.



(a) AP and query length (b) AP and decolorization

Fig. 7. ***Average Precision* related to query length and decolorization.** (a) The *Average Precision* might depends on the query word length. We present the results for three queries, i.e., *'nobis'*, *'pro nobis'*, and *'ora pro nobis'*. The longer is the word, the better is the retrieval performance. (b) We plot an example of the difference between the *Average Precisions* computed by using our proposed decolorization algorithm and the standard luminance value. Around the working value of $\alpha = 8$, the gain is bigger than 10%. The modified Grundland's algorithm produces a higher contrast image than the standard luminance.

In Table I we report word-spotting statistics. For each word we request a ranked list of the most similar 1000 occurrences, and we list in the table the number of *Relevant Words*, *True Positives*, *False Positives*, *False Negatives*, and the resulting *AP*. The word length ranges from 3 to 11 letters, and, apart from one word in the cross-manuscript group, the *AP* ranges from 52% to 85%. These results are comparable to, and even better than, the state-of-the-art methods (see for instance statistics in Almazán et al. [4]), which have *AP* values ranging from 30% to 60% for the handwritten GW documents, and, more important, from 40% to 80% for the machine printed LB dataset. Moreover, our algorithm is capable of doing a cross-manuscript search with good *AP* scores (last three rows of Table I). The first book is the query provider, while the second is the book where word occurrences are retrieved. Two of the cross-manuscript *AP* values are comparable with the single manuscript searches. The low value of $AP \sim 27\%$ in the word *'Amen'* is due to a different way of writing the capital *'A'* from one manuscript to the other; however, we are able to spot 26 *True Positives* over 33 *Relevant Words*.

Generally, the *AP* values might also be dependent on the query word length. In Fig.7(a) we compare the *AP* plots for the words *'nobis'*, *'pro nobis'*, and *'ora pro nobis'*; we choose these three examples because there is not only a increment of the word length, but each query is also a subset of the next one, so that we can measure the improvement due to the new added part. We can easily appreciate how, across different values of α, the longer the query, the bigger the *AP*. Compared to GW and LB used by Almazán et al. [4], here we deal with both a bigger database of images, and a more challenging medieval handwriting style, which consists typically in a series of vertical strokes that make the letters very similar to each other (Fig. 8 and Fig. 3). This is the reason why some short words

like *'lord'* and *'Per'* have respectively bigger *AP* values than the word *'nobis'* and *'psalmus_dd'* (up-left of Fig. 8(b)), which does not contain enough strong distinguishing features. We also show an example comparison between using a standard color-to-gray conversion or the modified Grundland method. In Fig. 7(b), we plot the difference $\left(AP - \tilde{AP}\right)$, where AP is obtained with the proposed method, while $\tilde{AP}$ results from the same run but using the luminance instead. Around the working value of $\alpha = 8$ we obtain a gain in retrieval performance of more than 10%. The reason is that the original query has a color with a chromaticity close to the parchment, so that the luminance signal has a lower contrast value than that computed with the proposed modified Grundland's algorithm (see small figures in Fig. 7(b)).

We provide some insights about the algorithm behavior in the case of *False positives*. In Fig. 9 we show some wrong retrieved occurrences (red) with four query words (blue) in table I: *'god'*, *'lord'*, *'Amen'*, and *'nobis'*. Some errors are due to parts of the words that share a subset of query letters, as *'(E)go'*, *'(G)od'*, *'(L)ord'*, *'(w)ord'*, *'Ame'*. Note that, even if some of them represent the same word we are looking for (e.g., *'(G)od'*, *'(L)ord'*, or *'Ame'*), we cannot include them among *True positives*; from a computer vision point of view we cannot consider the capitals *'G'* or *'L'* as characters similar to their lower case versions. Similarly, a three character word *'Ame'* cannot be treated as the whole query *'Amen'*. In other cases, the found word differs just by one or few letters, such as *'goo(d)'*, *'gol(d)'*, *'(b)lood'* (both for query *'god'* and *'lord'*), *'(g)lori(e)'*, *'lon(g)'*, and *'(f)lood'*. The group of letters *'gdo'* within the word *'kyngdom'* probably has been retrieved because of the three circles of the letters, ignoring the ascenders and descenders. Another important point is the effect on *False positives* of the aforementioned slight variability of character appearance due to the medieval writing style, which uses the vertical stroke as a basic graphical element. This is evident in the queries *'Amen'* and *'nobis'*. In the outcome *'Aude v(irgo)'* the two strokes of *'u'* and the first stroke of *'d'* are considered the letter *'m'*; the second stroke of *'d'* disappears in the letter *'e'*, and the two strokes of *'v'* are mis-interpreted as a *'n'*. The only slight and visible cues for the human eye are the ascender of the letter *'d'* and the connection between strokes on the bottom of *'v'*; otherwise the letter sequence is almost indistinguishable from the query. The same thing happens with the words *'Ave m(aria)'*. Here we have the same three letters, and the algorithm recognizes the *'v'* as a *'n'*. Similar visual interpretation errors occur with the retrieved *'Aude b(arbara)'*, where the ascenders of *'d'* and *'b'* are too small to make a difference in the recognition. The problem of the stroke usage explains well the bad matches for the query *'nobis'*. The small ascender of *'b'* makes the word as a sequence of seven strokes followed by a letter *'s'*, causing a lot of *False positives*.

Finally, we show some other word retrieval tests with words, single characters and *non-textual* elements. For each of them we depict the original image and the zoomed query, together with a subset of the *True Positives* returned by the algorithm. We choose a small word with three characters (*'Per'*), and a long seven-character word *'Psalmus'* (Fig. 10). We also demonstrate how the algorithm might work with other type of query input. In Fig. 10 we also launch the algorithm with a capital letter *'H'*, which is a single character, and a drawing as query data, i.e., the so called *'line filler'*, which indicates a region of the text with verses.

VI. Conclusions

We have presented a completely automatic and scalable framework to perform word-spotting in medieval handwritten books, in a single or cross-manuscript setup. The system has better performance than the state-of-the-art, and, unlike current solutions, does not require user intervention to manually define tunig parameters or to provide a large amount of annotated training data. This provides a simpe to implement pipeline that could be used and integrated in an unsupervised system, such as a web service associated with a digital library. We have also presented a quantitative and qualitative evaluation of the algorithm behavior in the case of an heterogeneous corpus content, that includes different kind of input data. Future work will focus better on improving the cross-manuscript capability, and on the investigation of other local image descriptors that might improve the final *AP* score.

Acknowledgments. This work was partially supported by the Digitally Enabled Scholarship with Medieval Manuscripts (DESMM) project funded by the Mellon Foundation (ydc2.yale.edu/). We also acknowledge the contribution of Sardinian Regional authorities for Data Intensive Computing activities.

References

[1] T. M. Rath, R. Manmatha, and V. Lavrenko, "A search engine for historical manuscript images," in *ACM SIGIR*, 2004, pp. 369–376.

[2] G. Nagy and D. P. Lopresti, "Interactive document processing and digital libraries." in *DIAL*, 2006, pp. 2–11.

[3] R. Pintus, Y. Yang, E. Gobbetti, and H. Rushmeier, "A TaLISMAN: Automatic Text and LIne Segmentation of historical MANuscripts," in *GCH*, October 2014, pp. 35–44.

[4] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Segmentation-free word spotting with exemplar svms," *Pattern Recognition*, vol. 47, no. 12, pp. 3967–3978, 2014.

[5] Json (javascript object notation). [Online]. Available: http://json.org/

[6] C. Myers, L. Rabiner, and A. Rosenberg, "An investigation of the use of dynamic time warping for word spotting and connected speech recognition," in *ICASSP*, vol. 5, 1980, pp. 173–177.

[7] R. Manmatha, C. Han, and E. M. Riseman, "Word spotting: A new approach to indexing handwriting," in *CVPR*, 1996, pp. 631–637.

[8] T. M. Rath and R. Manmatha, "Word spotting for historical documents," *IJDAR*, vol. 9, no. 2-4, pp. 139–152, 2007.

[9] D. Fernández-Mota, R. Manmatha, A. Fornés, and J. Lladós, "Sequential word spotting in historical handwritten documents," in *DAS*, 2014, pp. 101–105.

[10] Y. Leydier, A. Ouji, F. LeBourgeois, and H. Emptoz, "Towards an omnilingual word retrieval system for ancient manuscripts," *Pattern Recognition*, vol. 42, no. 9, pp. 2089–2105, 2009.

[11] F. Perronnin and J. A. Rodriguez-Serrano, "Fisher kernels for handwritten word-spotting," in *ICDAR*, 2009, pp. 106–110.

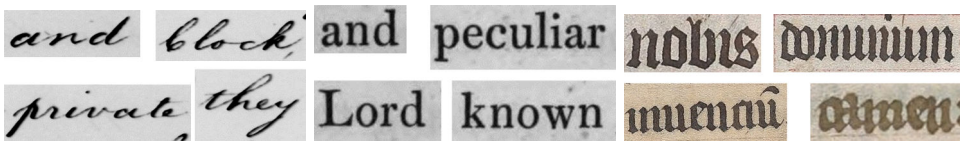
[12] S. Thomas, C. Chatelain, L. Heutte, and T. Paquet, "An information extraction model for unconstrained handwritten documents," in *ICPR*, 2010, pp. 3412–3415.

[13] V. Frinken, A. Fischer, and H. Bunke, "A novel word spotting algorithm using bidirectional long short-term memory neural networks," in *Artificial Neural Networks in Pattern Recognition*, 2010, pp. 185–196.

[14] J. A. Rodríguez-Serrano and F. Perronnin, "Handwritten word-spotting using hidden markov models and universal vocabularies," *Pattern Recognition*, vol. 42, no. 9, pp. 2106–2116, 2009.

[15] Y. Leydier, F. Lebourgeois, and H. Emptoz, "Text search for medieval manuscript images," *Pattern Recognition*, vol. 40, no. 12, pp. 3552–3567, 2007.

[16] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934–942, 2012.



(a) George Washington dataset (b) Lord Byron dataset (c) Medieval books

Fig. 8. **Writing style comparison.** Compared to the handwritten GW documents (a) [8], [41], and the machine printed LB words (b) [42], medieval writing style (c) is much more challenging, since it consists typically in a series of vertical strokes with very small changes to depict different characters. See also Fig. 3.

Fig. 9. ***False positives*.** We present four query words (blue) and some *False positives* (red). The errors are due to words containing same subset of letters, or the very low variability of character appearance due to the medieval writing style, which uses the vertical stroke as a basic graphical element, and its slight modification to depict different letters.

(a) (b) (c) (d)

Fig. 10. **Word-spotting results.** Example of the word-spotting pipeline applied to the three character word *'Per'*, the seven character word *'Psalmus'*, the capital letter *'H'*, and a drawing element, i.e., the so called *'Line filler'*. For each of them, we show one original image with the zoomed version of the query word, and a subset of the *True Positives* returned by the algorithm.

[17] ——, "Hmm-based word spotting in handwritten documents using subword models," in *ICPR*, 2010, pp. 3416–3419.

[18] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *PAMI*, vol. 34, no. 2, pp. 211–224, 2012.

[19] A. A. Gooch, S. C. Olsen, J. Tumblin, and B. Gooch, "Color2gray: salience-preserving color removal," in *ACM TOG*, vol. 24, no. 3, 2005, pp. 634–639.

[20] R. Bala and R. Eschbach, "Spatial color-to-grayscale transform preserving chrominance edge information," in *Color and Imaging Conference*, vol. 2004, no. 1, 2004, pp. 82–86.

[21] L. Neumann, M. Čadík, and A. Nemcsics, "An efficient perception-based adaptive color to gray transformation," in *CAe*. Eurographics, 2007, pp. 73–80.

[22] Y. Kim, C. Jang, J. Demouth, and S. Lee, "Robust color-to-gray via non-linear global mapping," in *ACM TOG*, vol. 28, no. 5, 2009, p. 161.

[23] K. Rasche, R. Geist, and J. Westall, "Re-coloring images for gamuts of lower dimension," *Computer Graphics Forum*, vol. 24, no. 3, 2005.

[24] M. Grundland and N. A. Dodgson, "Decolorize: Fast, contrast enhancing, color to grayscale conversion," *Pattern Recognition*, vol. 40, no. 11, pp. 2891–2896, 2007.

[25] L. Benedetti, M. Corsini, P. Cignoni, M. Callieri, and R. Scopigno, "Color to gray conversions in the context of stereo matching algorithms," *Machine Vision and Applications*, vol. 23, no. 2, pp. 327–348, 2012.

[26] R. Pintus, Y. Yang, and H. Rushmeier, "ATHENA: Automatic text height extraction for the analysis of text lines in old handwritten manuscripts," *ACM JOCCH*, 2014.

[27] R. Huang, S. Oba, S. Palaiahnakote, and S. Uchida, "Scene character detection and recognition based on multiple hypotheses framework," in *ICPR*, 2012, pp. 717–720.

[28] W. Dong, Z. Lian, Y. Tang, and J. Xiao, "Text detection in natural images using localized stroke width transform," in *MultiMedia Modeling*, 2015, pp. 49–58.

[29] "Mirador 2," Stanford University Library, 2015. [Online]. Available: http://sul-reader-test.stanford.edu/m2

[30] "International image interoperability framework," 2015. [Online]. Available: http://iiif.io/

[31] A. Papandreou, B. Gatos, G. Louloudis, and N. Stamatopoulos, "Icdar 2013 document image skew estimation contest (disec 2013)," in *ICDAR*, 2013, pp. 1444–1448.

[32] Beinecke, "Beinecke rare book and manuscript library - http://beinecke.library.yale.edu/," Yale University, 2013.

[33] M. Bulacu, R. van Koert, L. Schomaker, T. van der Zant *et al.*, "Layout analysis of handwritten historical documents for searching the archive of the cabinet of the dutch queen." in *ICDAR*, vol. 7, 2007, pp. 357–361.

[34] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[35] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[36] E. J. Crowley and A. Zisserman, "The state of the art: Object retrieval in paintings using discriminative regions," in *Proc. BMVC*, 2014.

[37] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros, "Data-driven visual similarity for cross-domain image matching," in *ACM TOG*, vol. 30, no. 6, 2011, p. 154.

[38] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *CVPR*, 2013, pp. 2946–2953.

[39] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011, pp. 89–96.

[40] E. Fetaya and S. Ullman, "Learning local invariant mahalanobis distances," *arXiv preprint arXiv:1502.01176*, 2015.

[41] T. M. Rath and R. Manmatha, "Word image matching using dynamic time warping," in *CVPR*, vol. 2, 2003, pp. II–521.

[42] M. Rusinol, D. Aldavert, R. Toledo, and J. Llados, "Browsing heterogeneous document collections by a segmentation-free word spotting method," in *ICDAR*, 2011, pp. 63–67.