A Simple and Fast Word Spotting Method

Alon Kovalchuk, Lior Wolf, and Nachum Dershowitz
The Blavatnik School of Computer Science, Tel Aviv University
Email: {kovalchuk,wolf,nachum}@post.tau.ac.il

***Abstract*—A simple and efficient pipeline for word spotting in handwritten documents is proposed. The method allows for extremely rapid querying, while still maintaining high accuracy. The dataset images that are to be queried are preprocessed by a simple binarization operation, followed by the extraction of multiple overlapping candidate targets. Each binary target, as well as the binarized query, is resized to fit a fixed-size rectangle and represented by conventional image descriptors. Then, a cosine similarity operator—followed by maximum pooling over random groups—is used to represent each target or query as a concise 250D vector. Retrieval is performed in a fraction of a second by nearest-neighbor search within that space, followed by a simple suppression of extra overlapping candidates.**

I. Introduction

Recent large-scale digitization and preservation efforts have made huge numbers of images of historical manuscripts readily available over the internet. Unfortunately, optical character recognition (OCR) for handwritten documents is notoriously difficult, making it nigh impossible to search within those images for something in particular. As an alternative, one can perform image-based search, a form of "query by example". Given a query *image* of a word, one seeks all sub-images that contain occurrences of that same word within the dataset of documents. With multi-writer collections, where handwriting can differ significantly from document to document, this task can be quite challenging.

The method we propose for this word-spotting task is both simple to implement and fast to run. We use very basic versions of standard operations for binarization, delineating bounding boxes, resizing, feature extraction, similarity measurement, and nearest-neighbor retrieval. Yet, this simple pipeline leads to a remarkably effective (good precision) and efficient (tenth-of-a-second retrieval time) search engine.

The outline of the process is as follows: In a preparatory step, all the dataset images that are to be queried are binarized. Then many candidate target word-like sections of the images are isolated. These typically include many overly-large and overlapping target segments. Each binarized target is resized to fit into a Procrustean bed (that is, a fixed-size rectangle) and represented by conventional image descriptors. For accuracy, a concise subset of targets for comparison is chosen by maximum pooling over random groups, using standard cosine similarity distances.

For each query, as it is posed, the same binarization, resizing, feature extraction and comparison are applied to the query image. Retrieval is accomplished by a simple nearest-neighbor search within that space. Only the best match of any set of overlapping candidates is returned.

Resizing makes the method scale invariant, which is crucial with diverse collections deriving from multiple documents—often poorly preserved, written by multiple authors, and perhaps digitized under differing conditions and containing significant noise. Target patches are not clustered. And words in the images are not tagged in any way: thus the method works unsupervised. This makes the method easily applicable to many manuscript datasets. Nor need queries derive from the dataset itself. They can even be synthetic and produced from digital text rendered in an archaic font.

Words need not be pre-segmented, as in many methods, though accurate word segmentation can help somewhat. Rather, a generous set of potential targets is constructed, which over-compensates for lack of segmentation, as it includes almost all actual word segments and many more. This popular modern approach also helps makes the method widely applicable.

The proposed pipeline is quite robust, as it does not depend on the specific choices of component methods. Thus, it can easily accommodate more sophisticated binarization techniques (necessary for collections of degraded manuscripts, such as the Dead Sea Scrolls [1]) or alternate features (such as SIFT, which does well for unconnected scripts, for example). Furthermore, it is an easy matter to add preprocessing (e.g., query jittering [2], [3]) or post-processing stages (rerank the results, or winnow a quickly-obtained list of results by a slower—but perhaps more accurate—method).

The next section briefly surveys some recent approaches to word spotting. Then, in Section 3, we explain our method step by step, followed by a section devoted to experimental results on the standard (two-writer, longhand) George Washington [4][1] and (typewritten) Lord Byron [5][2] benchmarks. We conclude with a brief discussion of possible improvements and extensions.

II. Background

Much effort has been devoted to research on word spotting for multi-author multi-style document images; a few recent

[1] At http://www.iam.unibe.ch/fki/databases/iam-historical-document-database/washington-database.

[2] At http://books.google.com/books?id=u6poWVzCIWsC.

examples are [6], [7], [8], [9]. Other works, with their own set of problems and less relevant here, deal with words embedded in outdoor photographs, e.g., [10].

Dynamic time warping (DTW) and hidden Markov models (HMMs) are two popular training techniques. An example of the former is [11] and of the latter is [12]. Many recent HMM-based systems are supervised and pre-segmented. Regrettably, these techniques are time consuming.

Two approaches are possible in searching for occurrences of words in documents: one can first segment the text into words and then compare each target word with the query, or one can search for a match to the query using a sliding window of some sort. There is substantial literature on word segmentation, including, for example, [13]. An example of word spotting among segmented images is [14]; among the works that do not require segmentation are [15] and [2].

As pointed out in [2], one of the main drawbacks of prior segmentation is that it is error-prone and mistakes negatively affect the following stages. On the other hand, an exhaustive comparison of the whole dataset using a sliding window is unfeasible for costly matching methods like DTW. An in-between approach is to work with multiple overlapping target regions, as in [16]. Using multiple candidates, as we do, for the purpose of reducing the number of false positives that sliding-windows approaches can engender, is a current trend in computer vision; see [17], [18] and others.

A second dimension that distinguishes work in this area is whether training examples are used for learning or whether the method is unsupervised. Our method is unsupervised. Similarly, the bag-of-features method of [5] works within a segmentation-free framework; it samples the space of patches uniformly, but uses local descriptors.

Our method is inspired by the work of Liao et al. [3]. This face recognition method does not follow the conventional pipeline of detection, followed by alignment, followed by representation and classification. Instead, it consists of a feed-forward network aimed at performing recognition directly from the unsegmented image. The design of that system relies on a recently theory of visual recognition proposed by Poggio [19], which provides a computational model of the ability of biological vision-systems to recognize based on only few training examples. Since in word spotting there is typically only one example (the query), and since little is known regarding the location of candidate words, this framework is extremely appealing for the task.

What is common to our work and that of Liao et al. is the use of local descriptors across candidate patches and the use of maximum pooling. However, the domains are very different and our system is much simpler than that of [3]. Local jittering is not used here; our hierarchy is flat and very efficient even without using PCA or hashing; our training data is unsupervised and pooling is done randomly; no classifiers are used to learn similarities; and we employ a mechanism for identifying candidate targets.

III. Method Description

A. Design Choices

One of the strengths of the proposed method lies in its simplicity. The set of candidate targets is extracted by a straightforward process. Then, the same processing pipeline is applied to both the candidate targets and the query image. Identifying matching targets is performed via a nearest neighbor search.

The process of max-pooling, which is central to the success of our method, relies on the extraction, in an unsupervised manner, of target candidates from a set of training images. In practice, we take the training set to be the set of dataset images. Since no optimization is performed by our method, and since the cosine similarity used is bounded by 1, we expect (and observe) no negative outcomes to this design choice.

Extracting candidate targets from the dataset images: We are given a set of dataset images to query. When following a sliding-window approach there are tens of thousands of possible targets to consider, or more, depending on the number of scales and the stride length. Such a liberal approach leads to an increase in the number of false matches as well as to an increase in the computational demands. Instead, following a recent trend in object recognition and a long tradition in document analysis, we extract a set of candidate targets.

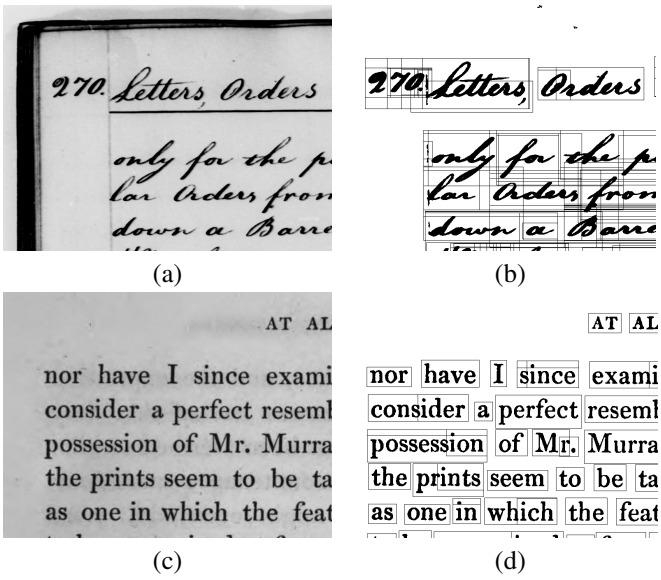
Our target prediction solution is heuristic rather than learned. The images are first binarized by thresholding the input image at a value which equals 85% of the mean pixel intensity.[3] Connected components are computed, and connected components that are either too small (noise) or too big (stains and page margins) are discarded.

The candidate targets are formed as groups (of arbitrary size) of connected components that satisfy four criteria:

1) The bounding box containing all the components within the group is not wider or taller than predetermined thresholds. This bounding box must also have an area larger than a third threshold.
2) The projection of the pixels of all the components onto the x-axis does not have a gap larger than a threshold.
3) The projection of the centers of mass of each component to the y-axis does not have a gap larger than another threshold.
4) There is no component outside the group that lies entirely within the bounding box of the group's components such that its center of mass is not the rightmost, leftmost, topmost or lowest of all centers of mass associated with the group components.

The suitable groups of connected components are collected, given an image, by iterating over all connected

[3]Here, and in almost all subsequent design choices we opt for the simplest solution; a more sophisticated solution can clearly lead to better performance.



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

Figure 1. (a) A page from the George Washington dataset. (b) The binarized page with candidate targets overlayed. Candidate regions for one connected component might intersect other connected components. For example, the numeral 0 produces a candidate target that cuts the numeral 7 vertically. (c,d) Similarly for the Lord Byron dataset.

components. Each time, the connected component at hand is considered to be the top-left component of the group. The subset of the connected components that fall within the width and height limits set by the first criterion are considered and smaller subsets are constructed by scanning from left to right until a large enough gap is met up with. During this process, each subset formed is evaluated based on these criteria.

The resulting candidate targets are typically dense and often overlapping. See Fig. 1. On a typical image from the George Washington dataset, there are around 7500 candidate targets, which is about 30 times the number of actual words. On a typical Lord Byron page, which is typewritten, the candidate-target to word ratio is only two.

Representing binary image patches: The process of representing an image patch as a vector is illustrated in Fig. 2. It is applied to the binarized version of the query and to each of the binary candidate targets, formed, as detailed above, as a union of a group of connected components.

From the binary patch, the minimal bounding box containing all black pixels is extracted and placed within a wider patch such that there is a margin of a fixed size around the bounding box. The extended patch is resized (by image interpolation) to a patch of a fixed size. Using a regular grid, the fixed sized patch is divided into multiple non-overlapping cells that are each encoded by the HOG descriptor [20] and by the LBP descriptor [21]. The HOG descriptors of all cells are concatenated and the resulting vector is normalized to an Euclidean norm of 1. The same process is applied to all LBP descriptors. The two vectors are then concatenated to

(a) (b) (c)

Figure 2. The patch normalization process. A patch (a) and its binarized version (b). (c) Resized patch with grid overlayed. All patches of candidate targets and of queries are resized to the same size regardless of their size and aspect ratio. Note that the binary patches do not contain pixels from connected components that do not belong to the candidate target.

form a single vector $v \in \mathbb{R}^d$.

A matrix $M \in \mathbb{R}^{n \times d}$, which consists of the vector representations (same as v) for n random candidate targets from the dataset, is then considered. The vector v is transformed to a vector $u \in \mathbb{R}^n$ by means of a linear projection: $u = Mv$. In other words, the normalized descriptor vector is represented by its similarities to a predetermined set of exemplars.

Then, a max-pooling step takes place. The set of indices $[1..n]$ is randomly split into groups of size p. Call this random partition $P = \{C_i\}_{i=1}^{n/p}$. It is sampled once, uniformly from the space of all partitions, and applied for the purpose of max-pooling during the encoding of all patches. Given a vector v, this max-pooling is performed simply by considering one measurement per C_i that is the maximal value among all the indices of this subset in the vector v. Put differently, let u be the vector of length n/p that results from the max-pooling process as applied to the vector v. Then $u_i = \max_{j \in C_i} v_j$.

Note that in previous work, e.g., [3], max-pooling at the exemplar level is done on subsets of samples that belong to the same category or identity and not to a random subset of the samples. Employing such supervised max-pooling can boost performance. However, it also results in the loss of elegance, simplicity and practicality of using a completely unsupervised system.

Query: Given a query image, it is binarized, cropped by the minimal bounding box (connected components do not play a role here), embedded in a larger white patch and resized as explained above. Then the vector consisting of the multiple HOG and LBP descriptors is computed, multiplied by the matrix M and max-pooling is employed using the partition P. This vector is then compared, by means of $L2$ distance, to the similar vectors—computed in exactly the same manner—of all candidate targets.

Note that the set of vectors associated with the candidate targets is precomputed, which supports scalability. For

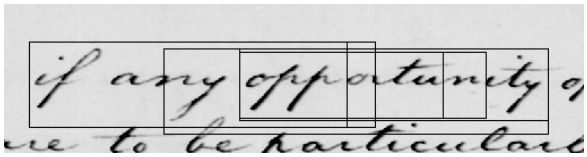


Figure 3. The overlapping target suppression mechanism. The multiple candidates read "opportunity", "if any oppo", "if any opp", "y opportunity", "opp", "oppo", "opportu", and "opportun". All share "opp" as biggest connected component. Since "opportunity" is ranked highest in the query of Fig. 4, the rest are discarded.

very big databases, kdtrees or other data-structures can be employed to perform rapid sub-linear search. However, for the scales required in the experiments herein, all samples are stored in the main memory without difficulty and direct computation of the distances is already very efficient.

All candidate targets are ranked in accordance with the computed $L2$ distance. Recall that there are many overlapping candidate targets. In order to eliminate multiple occurrences of the same target word as it appears in multiple candidate targets, we employ yet another simple heuristic. Out of all candidate targets that contain the same connected component as their largest component, we only consider the candidate with the highest rank (lowest distance). The rest of the targets that share the same maximal-area component are eliminated from the retrieved list. This suppression mechanism is illustrated in Fig. 3.

B. Implementation Details

There are various parameters used in the system. We have not yet performed any large-scale experiments to select these; rather, we chose reasonable values for each separately, based on observing a few patches. We believe, based on very limited tests, that the performance is not overly sensitive to these parameters.

The minimal size for connected components is 30 pixels, measured by counting the number of pixels in the binary image of the connected components. The maximal size is such that all connected component are smaller than 600 pixels in both dimensions. The sizes of the candidate targets are bounded (criterion 1) by a wide rectangle of 700×160 pixels. The maximal horizontal gap and the maximal vertical gaps are both set to 25 pixels (criteria 2,3).

The margin used in order to embed the patch's bounding box in a larger patch is 8 pixels in all four directions. This larger patch is resized (up or down) using bi-cubic interpolation to a patch of 160×56 pixels. Note that the binary image is transformed to a gray-value image due to this interpolation. The resized image is divided into a grid of 20×7 cells, each of 8×8 pixels.

Each HOG descriptor [20] is a vector in $\mathbb{R}^{31}$; each LBP descriptor is in $\mathbb{R}^{58}$. The length of the vector v, denoted d above, is therefore $31\times20\times7+58\times20\times7=12{,}460$. The number n of exemplars used is set to 3,750. These are chosen

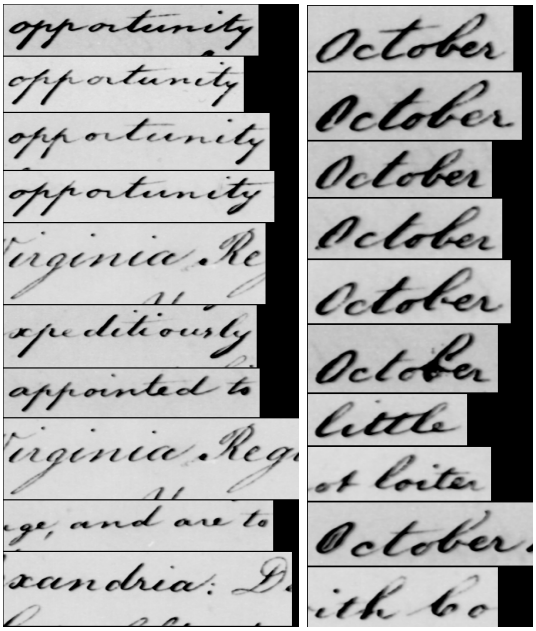


Figure 4. Top ten retrievals from the George Washington dataset for two sample queries. The query image is identical to the first result, only segmented manually. (left) The four occurrences of the word "opportunity" are ranked highest. (right) Three out of ten results are not from the 15 occurrences of "October".

at random from the dataset. We use a random partition P that contains 250 subsets of the indices $[1..3750]$ of size 15, resulting in patch representation vectors u in $\mathbb{R}^{250}$.

IV. Evaluation

Our approach is evaluated on two public datasets that we could find online: The George Washington (GW) dataset [4] (used in [22], [23]) and the Lord Byron (LB) [5] dataset. These datasets are comprised of 20 pages each and contain approximately 5,000 words. They are very different in nature: GW contains handwritten text while LB contains typewritten text. However, exactly the same system is used without any modification, and all parameters are fixed at the values described in Section III-B. We use the same evaluation protocol as [2], and so our results are directly comparable.

There is one query per annotated word image, see Figures 4, 5 for examples. For each query, all candidate targets are ranked and then filtered so that no two candidates with the same largest connected component appear in the list (see Section III-A). A candidate target is considered a true match to the query if it overlaps an annotated region of the same word as the query word by more than 50%. Overlap percent and success are defined based on the PASCAL detection criterion, which counts a detection to be correct if the ratio of intersection over the union of the two bounding boxes is greater than 0.5.

Following [5], [2], the query image is also considered to be a true positive. This benchmark design choice is

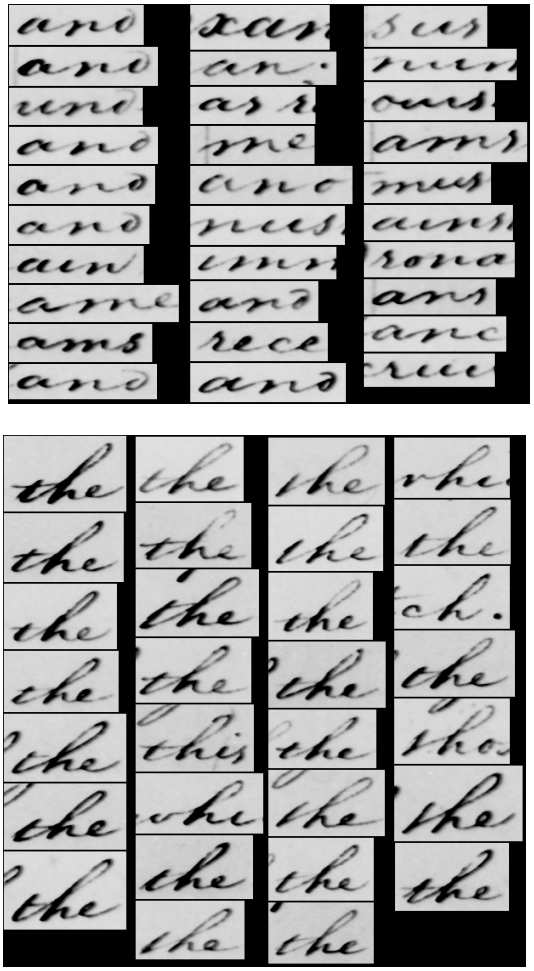


Figure 5. Top 30 retrievals from the George Washington dataset for top left "and" and "the". Short words pose a challenge to word spotting methods. Moreover, longer words that contain the query word are considered to be mistakes. Here, 23 results are mistaken for the top example, and 5 for the bottom one.

reasonable since we are not given ground-truth bounding boxes, and so the query image could be missing from the set of candidate targets. Based on reporting standards in the literature, we report the Mean Average Precision (MAP) retrieval score.

Table I shows the scores achieved by our method and several variants of it, as well as by baseline methods [5], [2] that employ the same benchmark protocol. We present results for our complete pipeline, the complete pipeline applied to segmented words (an easier task, especially for the handwritten GW), and to two variants that do not use max-pooling: one in which the same 3,750 random exemplars are used, and one in which a random subset of 250 exemplars is used. Our method, even without max-pooling, performs extremely well on the Lord Byron benchmark. In fact, max-pooling, while making the representation much more concise, hurts performance on this dataset. On the George Washington benchmark, however, max-pooling increases the accuracy significantly. In both cases, max-pooling produces a compact representation and helps reduce both the run time and the memory footprint.

Table I
MEAN AVERAGE PRECISION FOR VARIOUS METHODS.

Method	**GW**	**LB**
Efficient exemplar word spotting [2]	54.5%	85.5%
Segmentation-free word spotting [5]	30.5%	42.8%
Complete pipeline	50.1%	90.7%
Same applied to segmented words	66.3%	92.9%
Without max-pooling ($v \in \mathbb{R}^{3750}$)	48.8%	90.8%
Without max-pooling ($v \in \mathbb{R}^{250}$)	47.6%	90.7%

Table II
RUN TIME STATISTICS

Method/component	**GW**	**LB**
Number of queries	4,860	4,988
[2] all queries	5,058sec	4,159sec
[2] average per query	1.04sec	0.83sec
Our, all queries	158sec	46sec
Our, average per query	0.033sec	0.009sec
Our, single query	0.08sec	0.03sec
Preprocessing one page (ours)	46sec	3sec
Average memory per page (ours)	1,875KB	136KB

As noted in [2], a direct comparison with the methods of [22], [24] is not possible, as different protocols are used and the results are not comparable. In addition to the lack of agreed upon train/test splits, [24] does not include the query in the retrieved set. With regard to the exact task performed, neither of [22], [24] is segmentation-free. The reported results on the GW dataset are between 52% and 65% in [22], depending on the fold used for evaluation, while [24] reports 54%. Based on these statistics, the value of the DTW and HMM based methods is clear, and in the future we would like to employ our method as a quick short-list generating method that enables the rapid use of such methods as these two.

The running time of our method, using a straightforward matlab implementation based on VLFeat [25], is reported in Table II. It is compared to the runtime we obtained using the code of [2], which is shorter than that reported by the authors two years ago. Still, our method is two orders of magnitude faster. Note that multiple queries in our method are more efficient, on average, than a single query due to efficient matrix multiplication routines. The process of finding candidate targets dominates the preprocessing step. While not very efficient, it is run only once per dataset. As is evident, preprocessing is much more challenging for the George Washington dataset.

V. CONCLUSION

Currently, as quality OCR technologies are still lacking when dealing with handwritten documents, and especially

with historical manuscripts, word-spotting technologies provide a useful substitute. In this work, we present an extremely efficient word-spotting method. Though this may not be the most accurate spotting technology in existence, it is certainly the simplest method providing a level of accuracy within reach of state-of-the-art ones. Specifically, no optimization is performed during preprocessing or during querying, and the representation is easily stored in conventional data structures. This is in contrast to methods that employ techniques such as DTW, HMM, or SVM.

Our method is also "trendy", as it relies on a module that generates many target proposals and employs max-pooling. Both techniques are employed very plainly: the candidates are extracted heuristically and not using learning techniques, and max-pooling is employed in a completely unsupervised manner and only once, in contrast to the deep convolutional networks that are now becoming ever so popular.

Our plans include experiments with the 350,000 pages of the Cairo Genizah and the 15,000 fragments of the Dead Sea Scroll collections. These documents are poorly preserved and were written in multiple hands and in multiple languages and styles.

The flexibility and extensibility of our method imply that it has the potential of becoming very useful in practice—either as is, or with some modifications, such as replacing the descriptors used or incorporating supervision, or as a plug-in method that can provide a short-list of candidates to be refined and verified by more demanding techniques.

Acknowledgment

The authors would like to thank the support of the Ministry of Science, Technology and Space through Israel-Taiwan grant number 3-10341.

References

[1] Y. Zarai, T. Lavee, N. Dershowitz, and L. Wolf, "Integrating copies obtained from old and new preservation efforts," in *ICDAR*, 2013.

[2] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Efficient exemplar word spotting," in *BMVC*, 2012.

[3] Q. Liao, J. Z. Leibo, Y. Mroueh, and T. Poggio, "Can a biologically-plausible hierarchy effectively replace face detection, alignment, and recognition pipelines?" *CoRR*, vol. abs/1311.4082, 2013.

[4] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character HMMs," *Pattern Recogn. Lett.*, vol. 33, no. 7, pp. 934–942, May 2012.

[5] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós, "Browsing heterogeneous document collections by a segmentation-free word spotting method," in *ICDAR*. IEEE, 2011.

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

[7] B. Gatos and I. Pratikakis, "Segmentation-free word spotting in historical printed documents," in *ICDAR*, 2009.

[8] J. A. Rodríguez-Serrano and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *ICFHR*, 2008.

[9] S. Espana-Boquera, M. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid HMM/ANN models," *PAMI*, 2011.

[10] K. Wang and S. Belongie, "Word spotting in the wild," in *ECCV*, 2010.

[11] T. M. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," in *ICDAR*, 2003, pp. 218–222.

[12] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "HMM-based word spotting in handwritten documents using subword models," in *ICPR*, 2010.

[13] R. Manmatha and N. Srimal, "Scale space technique for word segmentation in handwritten documents," in *Scale-Space Theories in Computer Vision*, 1999.

[14] J. Almazan, A. Gordo, A. Fornés, and E. Valveny, "Handwritten word spotting with corrected attributes," in *ICCV*, 2013.

[15] L. Rothacker, M. Rusiñol, and G. Fink, "Bag-of-features HMMs for segmentation-free word spotting in handwritten documents," in *ICDAR*, Aug 2013.

[16] A. J. Newell and L. D. Griffin, "Multiscale histogram of oriented gradient descriptors for robust character recognition," in *ICDAR*, 2011.

[17] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman, "Using multiple segmentations to discover objects and their extent in image collections," in *CVPR*, 2006.

[18] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, "Segmentation as selective search for object recognition," in *ICCV*, 2011.

[19] T. Poggio, "The computational magic of the ventral stream," *Nature Precedings*, 2012.

[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[21] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *PAMI*, 2006.

[22] T. Rath and R. Manmatha, "Word image matching using dynamic time warping," in *CVPR*, 2003.

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

[24] J. A. Rodríguez-Serrano and F. Perronnin, "A model-based sequence similarity with application to handwritten word spotting," *PAMI*, vol. 34, no. 11, pp. 2108–2120, 2012.

[25] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," 2008. [Online]. Available: http://www.vlfeat.org/