Segmentation-Free Word Spotting with Exemplar SVMs

Jon Almazán[a], Albert Gordo[1b], Alicia Fornés[a], Ernest Valveny[a]

[a] *Computer Vision Center – Dept. Ciències de la Computació*
Universitat Autònoma de Barcelona, 08193 Bellaterra (Barcelona), Spain
[b] *INRIA Grenoble - Rhône-Alpes research centre*
655 Avenue de l'Europe, 38330 Montbonnot, France

Abstract

In this paper we propose an unsupervised segmentation-free method for word spotting in document images. Documents are represented with a grid of HOG descriptors, and a sliding-window approach is used to locate the document regions that are most similar to the query. We use the Exemplar SVM framework to produce a better representation of the query in an unsupervised way. Then, we use a more discriminative representation based on Fisher Vector to rerank the best regions retrieved, and the most promising ones are used to expand the Exemplar SVM training set and improve the query representation. Finally, the document descriptors are precomputed and compressed with Product Quantization. This offers two advantages: first, a large number of documents can be kept in RAM memory at the same time. Second, the sliding window becomes significantly faster since distances between quantized HOG descriptors can be precomputed. Our results significantly outperform other segmentation-free methods in the literature, both in accuracy and in speed and memory usage.

Keywords: word spotting, segmentation-free, unsupervised learning, reranking, query expansion, compression

1. Introduction

This paper addresses the problem of query-by-example word spotting: given a dataset of document images and a query word image, the goal is to identify and retrieve regions of those documents where the query word may be present. From decades ago this problem has attracted a lot of interest from the computer vision community [1, 2, 3, 4, 5], since making handwritten texts or ancient manuscripts available for indexing and browsing is of tremendous value. Interesting applications for word spotting are, for example, retrieving documents with a given word in company files, or searching online in cultural heritage collections stored in libraries all over the world. In this work, we specially focus on the unsupervised word-spotting problem, where no labeled data is available for training purposes. This is a very common scenario since correctly labeling data is a very costly and time consuming task.

[1] Part of this work was done while A. Gordo was a PhD student at the Computer Vision Center in Barcelona, Spain.

Email addresses: almazan@cvc.uab.es (Jon Almazán), albert.gordo@inria.fr (Albert Gordo), afornes@cvc.uab.es (Alicia Fornés), ernest@cvc.uab.es (Ernest Valveny)

URL: www.cvc.uab.es/~almazan (Jon Almazán)

Traditionally, word-spotting systems have followed a well defined flow. First, an initial layout analysis is performed to segment word candidates. Then, the extracted candidates are represented as sequences of features [6, 7, 8]. Finally, by using a similarity measure – commonly a Dynamic Time Warping (DTW) or Hidden Markov Model (HMM)-based similarity –, candidates are compared to the query word and ranked according to this similarity. Examples of this framework are the works of Rath and Manmatha [2] and Rodríguez-Serrano and Perronnin [5]. One of the main drawbacks of these systems is that they need to perform a costly and error prone segmentation step to select candidate windows. Any error introduced in this step will negatively affect the following stages, and so it is desirable to avoid segmenting the image whenever possible. Unfortunately, since the comparison of candidate regions, represented by sequences, is based on costly approaches such as a DTW or a HMM, it is not feasible to perform this comparison exhaustively with a sliding-window approach over the whole document image.

Late works on word spotting have proposed methods that do not require a precise word segmentation, or, in some cases, no segmentation at all. The recent works of [3, 4] propose methods that relax the segmentation problem by requiring only a segmentation at the text line level. One of their drawbacks, however, is that they require a large amount of annotated data to train the Hidden Markov Models [3] or the Neural Networks [4] that they use. Additionally, the line segmentation still has to be very precise to properly encode the lines.

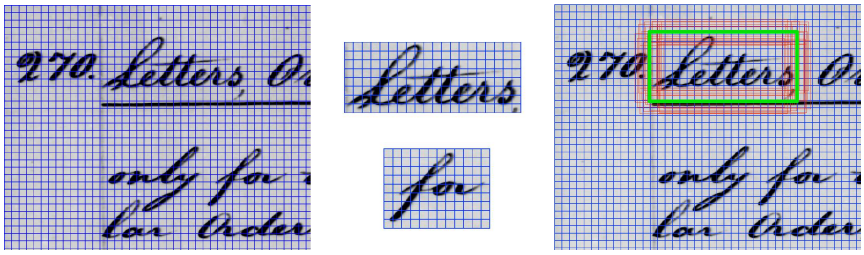
In [9], Gatos and Pratikakis perform a fast and very coarse segmentation of the page to detect salient text regions. Queries are represented with a descriptor based on the density of the image patches.Then, a sliding-window search is performed only over the salient regions of the documents using an expensive template-based matching.

The methods proposed by Leydier et al. [10] and Zhang and Tan [11] avoid segmentation by computing local keypoints over the document images. While [10] represents the document images with gradient-based features, [11] uses features based on the Heat Kernel Signature. The main drawback of these approaches is that they use a costly distance computation, which is not scalable to large datasets.

The work of Rusiñol et al. [12] avoids segmentation by representing regions with a fixed-length descriptor based on the well-known bag of visual words (BoW) framework [13]. In this case, comparison of regions is much faster since a dot-product or Euclidean distance can be used, making a sliding window over the whole image feasible. To further improve the system, unlabeled training data is used to learn a latent space through Latent Semantic Indexing (LSI), where the distance between word representations is more meaningful than in the original space. Rothacker et al. [14] exploits the use of the BoW representation to feed a HMM and avoids the segmentation step by means of a patch-based framework. Comparison of regions is slower than the BoW-based approach of [12], so it could not be directly applied in a large-scale scenario, but, thanks to the sequential encoding of the BoW features of the HMM, they obtain a more robust representation of the query.

Howe [15] uses a generative model for word appearance from a single positive sample, resulting in an example of a *one-shot learning* approach. The model consists of a set of nodes connected via springlike potentials, and arranged in a tree structure whose a priori minimum energy configuration conforms to the shape of the query word.

In this work we focus on this family of segmentation-free word-spotting approaches and we argue that previously described current methods can be improved in several ways. **First, they can be improved in the choice of low level features.** The features of [6, 7, 8, 14] produce sequence representations, which are usually slower to compare than fixed-length representations. The work of [9] uses a descriptor based on the patch density, which is insufficient to capture all



(a) HOG grid. (b) (c)

Figure 1: a) Grid of HOG cells. Only one small part of the image is shown. b) Two random queries of the George Washington dataset. The windows adjust around the HOG cells. c) A query (in green) and some positive samples (in red) used to learn the Exemplar SVM. To avoid clutter, only some of the positive windows are shown.

the fine-grained details. The bag of visual words approach of [12] fixes the size of the scanning window for efficiency reasons, which makes the accuracy of the method very dependent of the size of the query. We address these issues by using a sliding-window approach based on HOG descriptors [16], which have been shown to obtain excellent results when dealing with large variations in the difficult tasks of object detection and image retrieval. The document images are represented with a grid of HOG descriptors, where each cell has the same size in pixels. (*cf*. Fig 1(a)). In this case, we do not have a fixed window size; instead, the window size is adjusted to the query size, covering several HOG cells (*cf*. Fig 1(b)).

HOG descriptor provides a good trade-off between speed and memory requirements and discriminative power. However, more discriminative representations exist, but cannot be used in practice when dealing with large volumes of data – such as those of large-scale datasets or sliding-window setups. We propose to apply a common solution that consists of, first, using affordable features (*i.e*. HOG) to produce an initial ranking over all the dataset, and then, use more powerful and expensive representations to rerank only the most promising candidates returned by the first stage. Since only a small subset of the whole dataset is scanned, it is feasible to use more expensive features. In the case of word spotting, however, it is not clear which features would be better suited for this task. Therefore, we have included in the experiments a comparison of low-level features to represent the word images with the objective of analyzing their use on a reranking step. This reranking step can significantly improve the accuracy with a cost that does not depend on the number of documents on the dataset.

Second, spotting methods can be improved in the learning of a more discriminative space. In [12], LSI is used to learn a latent space where words and documents are more related. However, learning a semantic space with LSI may be too conditioned to the words used in the unsupervised training stage, and adapting to new, unseen words may be complicated. Instead, we propose to perform this unsupervised learning once the query has been observed, and adapt the learning to the particular query. For this task, we propose to use a similar approach to the Exemplar SVM framework of [17, 18]. Additionally, to further improve the representation of the query, we propose to combine reranking with a query expansion step, which uses the best candidates after the reranking step to construct a new, more informative representation of the query, and use it to query again the dataset to further improve the results. As a consequence of this reranking step it may be necessary to train several exemplar models per query. We propose to use a fast solver based on Stochastic Gradient Descent (SGD) to considerably speed up the

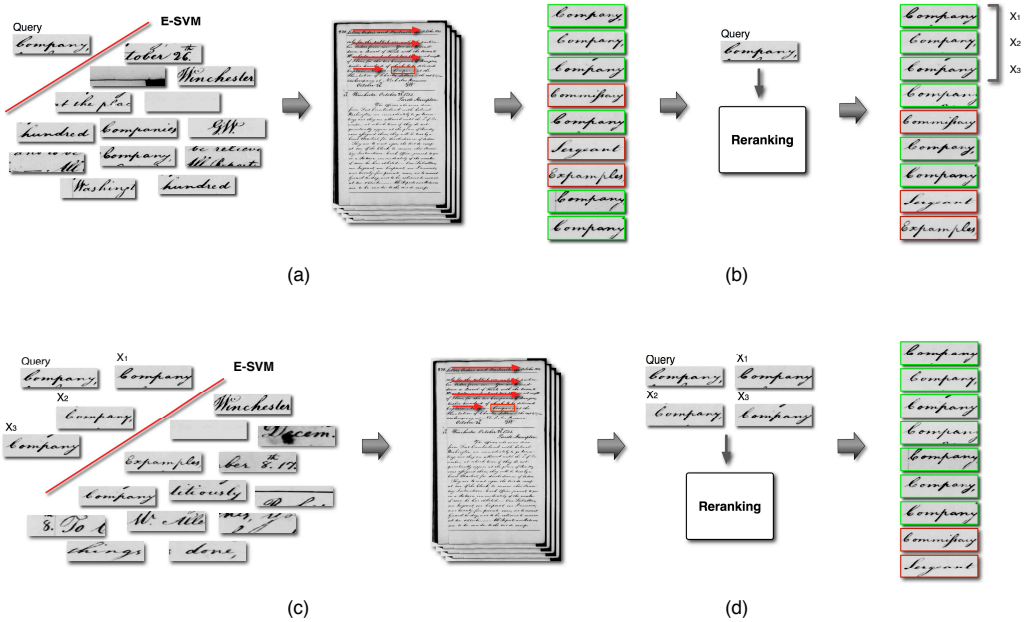


Figure 2: General scheme of the method proposed. (a) E-SVM training and sliding-window search. (b) First reranking of the best retrieved regions. (c) E-SVM retraining and sliding-window search applying query expansion with the first reranked regions. (d) Second reranking using the expanded training set.

training process.

Finally, these methods can be improved in the cost of storing the descriptors of all the possible windows of all the dataset items. Assuming HOG descriptors of 31 dimensions represented with single-precision floats of 4 bytes each (*i.e.*, 124 bytes per HOG), and 50,000 cells per image, storing as few as 1,000 precomputed dataset images would require 5.8GB of RAM. Since documents will not fit in RAM memory when dealing with large collections, we are left with two unsatisfying options: either recomputing the descriptors of every document with every new query, or loading them sequentially from a hard disk or a solid-state drive (SSD). Any of these approaches would produce a huge performance drop in the speed at query time[3]. To address this problem, we propose to encode the HOG descriptors using Product Quantization (PQ) [19]. Encoding the descriptors with PQ would allow us to reduce the size of the descriptors and to preserve a much larger amount of images in RAM at the same time. As a side effect, computing the scores of the sliding window also becomes significantly faster.

In summary, we present a full system for efficient unsupervised segmentation-free word spotting that will be shown to outperform existing methods in two standard datasets. The use of HOG templates provides a very natural model, and its discriminative power is improved through the use of Exemplar SVMs with SGD solvers. The use of PQ drastically improves the efficiency of the system at test time. Finally, the use of more informative features in combination with reranking and query expansion improves the final accuracy of the method at a reduced cost. A general scheme of the method can be seen in Figure 2.

[3] A similar point can be argued about the methods of [9], [12] or [14].

A preliminary version of our system was described in [20]. The system described here extends the work of [20] in the following ways: the comparison of different low-level features for word representation, the introduction of reranking and query expansion to improve performance, the analysis of different configurations of the PQ compression method, the use of a faster Stochastic Gradient Descent-based solver for learning the Exemplar SVM and new experiments with the full system integrating all these steps.

The rest of the paper is organized as follows. Section 2 describes the basic configuration of a HOG-based word-spotting approach. Section 3 extends it to make use of the Exemplar SVM framework, and Section 4 introduces the use of PQ to compress the HOG descriptors of the document. Then, reranking and query expansion are described in Section 5 and Section 6. Finally, Section 7 deals with the experimental validation and Section 8 concludes the paper.

2. Baseline System

Word spotting – and, particularly, handwritten word spotting – is a challenging problem for descriptors because they have to deal with many sources of variability such as deformations, different styles, *etc*. Moreover, in a large-scale and segmentation-free scenario, it is mandatory to use descriptors that are both fast to compute and compare, and that could be integrated in a sliding-window based search. Traditionally, features for word spotting have been based on sequences [6, 7, 8] and compared using methods such as Dynamic Time Warping or Hidden Markov Models. The rationale behind this is that these types of variable-length features would be able to better adapt to the word they represent and capture information independently of deformations caused by different styles, word length, *etc*. The main drawback of these methods is that comparing representations is slow, with usually a quadratic cost with respect to the length of the feature sequence, which makes them impractical in large-scale scenarios as well as in a sliding-window setup.

Interestingly, it has been shown that these variable-length features do not always lead to the best results and can be outperformed by fixed-length representations. The work of Perronnin and Rodríguez-Serrano [21] exploits the Fisher kernel framework [22] to construct the Fisher Vector of a HMM, leading to a fixed-length representation that outperforms standard HMM over variable-length features. Rusiñol et al. [12] represent document image regions with a descriptor based on the well-known bag of visual words framework [13] over densely extracted SIFT descriptors, obtaining a reasonable performance in segmentation-free word spotting. Unfortunately, although these features are superior to some sequence-based ones, they still remain impractical for a sliding-window setup since computing the descriptors for every window is usually quite slow. Rusiñol et al. [12] address this issue by fixing the window size and precomputing offline the descriptors of all possible windows of that size. This, however, makes the results very dependent on the size of the query, since very small or very large queries will not correctly adapt to the precomputed windows.

Because of these reasons here we advocated for the use of HOG descriptors [16], which have been shown to obtain excellent results in difficult tasks such as pedestrian and object detection and image retrieval (see, *e.g*., [23] or [18]). Although in general HOG descriptors cannot compete in terms of accuracy with other more powerful features, they are very fast to compute and compare, and are particularly suited for problems that require a sliding-window search such as this one, providing a very good trade-off between accuracy and speed. Now, as we will show in Section 5, it is actually possible to use more powerful and discriminative features during a

second step to rerank the best candidate words yielded by the sliding-window approach at a very reasonable cost, overcoming the main deficiency of HOG descriptors for this task.

In our system, the document images are divided in equal-sized cells (see Fig 1a) and represented with HOG histograms, which encode local gradients. We follow [23] and use HOG histograms of 31 dimensions (9 contrast insensitive orientation features, 18 contrast sensitive orientation features and 4 features to reflect the overall gradient energy around the cell) to represent each cell. Queries are represented analogously using cells of the same size in pixels. In this case, as opposed to [12], we do not have a fixed window size; instead, the window size depends on the query size, covering $H_q \times W_q$ HOG cells (*cf*. Fig 1(b)), leading to a descriptor of $H_q \times W_q \times 31$ dimensions. Note that the number of cells in a query, and therefore the final dimensionality of the descriptor, depends on the size of the query image. The score of a document region $\mathbf{x}$ of $H_q \times W_q$ cells with respect to the query $\mathbf{q}$ is then calculated as a 3D convolution, $\text{sim}(\mathbf{q}, \mathbf{x}) = \mathbf{q} * \mathbf{x}$. This could also be understood as calculating the dot-product between the concatenated HOGs of the query $\bar{\mathbf{q}} = \text{vec}(\mathbf{q})$ and the concatenated HOGs of the document region $\bar{\mathbf{x}} = \text{vec}(\mathbf{x})$, *i.e*., $\text{sim}(\mathbf{q}, \mathbf{x}) = \bar{\mathbf{q}}^{\mathbf{T}}\bar{\mathbf{x}}$, but we remark that at test time we perform a convolution instead of concatenating the descriptors explicitly. Following this approach, we can compute the similarity of all the regions in the document image with respect to the query using a sliding window and rank the results. To avoid returning several windows over the same region, Non-Maximum Suppression (NMS) is performed to remove windows with an overlap over union larger than 20%.

We further modify the baseline in two ways. First, instead of using the HOG descriptors directly, we reduce their dimensionality down to 24 dimensions with PCA. We observed no loss in accuracy because of this, probably because of the "simplicity" of text patches. Second, instead of calculating the dot-product, we are interested in the cosine similarity, *i.e*., calculating the dot-product between the L2 normalized descriptors. The cosine similarity is a typical choice in document retrieval, and we observed experimentally that L2 normalizing the vectors can indeed make a significant difference in the results. Note that the L2 normalization is performed at the region level, not at the cell level. Fortunately, we do not need to explicitly reconstruct the regions to normalize the descriptors, since $\text{sim\_cos}(\mathbf{q}, \mathbf{x}) = (\frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|})^T(\frac{\bar{\mathbf{x}}}{\|\bar{\mathbf{x}}\|}) = \frac{1}{\|\bar{\mathbf{q}}\|}\frac{1}{\|\bar{\mathbf{x}}\|}\bar{\mathbf{q}}^T\bar{\mathbf{x}} = \frac{1}{\|\bar{\mathbf{q}}\|}\frac{1}{\|\bar{\mathbf{x}}\|}\mathbf{q} * \mathbf{x}$ Therefore, we can calculate the $\text{sim}(\mathbf{q}, \mathbf{x})$ score with a convolution without explicitly concatenating the descriptors and later normalize it with the norms of the query and the document region. The norm of the query can in fact be ignored since it will be constant for all the document regions and therefore does not alter the ranking. As for the region patch, we can accumulate the squared values while performing the convolution to construct, *online*, the norm of the region patch without explicitly reconstructing it.

3. Exemplar Word Spotting (EWS)

In the previous section we introduced how HOG descriptors can be used in the framework of a sliding-window based approach for word spotting. There, we used a basic retrieval setting based on the cosine similarity. We note however, that the cosine similarity, despite being a reasonable option, may not be the optimal way to compare document regions, and that learning a better metric may yield significant improvements. In [12], this is achieved by learning, *offline*, a latent space through LSI, where the cosine similarity is an appropriate measure. However, this may be too conditioned to the words used in the unsupervised training stage, and adapting to new, unseen words may be complicated. Additionally, it is not clear how we could adapt this latent learning to our grid of HOGs framework.

Here we take a different approach and propose to learn, *at query time*, a new representation of the query, optimized to maximize the score of regions relevant to the query when using the dot-product. This new representation can be understood as *weighting* the dimensions of the region descriptors that are relevant to the query. We achieve this goal by means of the Exemplar SVM framework [17, 18]. Let us assume that we have access to a set $\mathcal{P}$ of positive regions that are relevant to the query. These are described as the concatenation of their PCA-compressed HOG descriptors, and are L2 normalized. Analogously, let us assume that we have access to a set $\mathcal{N}$ of negative regions that are not relevant to the query. In this case, we could find a new representation $\mathbf{w}$ designed to give a high positive score to relevant regions, and a high negative score to non-relevant regions when using the dot-product with L2 normalized regions. This can be casted as an optimization problem as

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum_{\{x_p, y_p\} \in \mathcal{P}} L(y_p \mathbf{w}^T \mathbf{x}_p) + C_2 \sum_{\{x_n, y_n\} \in \mathcal{N}} L(y_n \mathbf{w}^T \mathbf{x}_n), \tag{1}$$

where $y_p = 1$, $y_n = -1$, $L(\mathbf{x}) = \max(0, 1 - \mathbf{x})$ is the hinge loss and C_1 and C_2 are the cost parameters of relevant and non-relevant regions. This is very similar to the standard SVM formulation and can be solved by standard solvers such as LIBLINEAR [24]. The classifier bias can be considered implicitly by augmenting the samples as $\mathbf{x} = [\mathbf{x}\ \textit{bias\_multiplier}]$, where *bias_multiplier* usually equals 1. Solving this optimization produces a weight vector $\mathbf{w}$, which can be seen as a new representation of the query designed to maximize the separation between relevant and non-relevant regions. As in the baseline system, we can rearrange the terms at test time so that $\text{sim}(\mathbf{w}, \frac{\mathbf{x}}{\|\bar{\mathbf{x}}\|}) = \frac{1}{\|\bar{\mathbf{x}}\|}\text{sim}(\mathbf{w}, \mathbf{x})$, where $\text{sim}(\mathbf{w}, \mathbf{x})$ can be calculated without reconstructing the region vectors and $\|\bar{\mathbf{x}}\|$ can be calculated online while performing the convolution. Note that, once learned, the classifier bias is constant for every dataset sample given a query, and therefore it is not needed to rank the words. However, considering the bias during the learning may lead to a better $\mathbf{w}$ model, and so it is important to include it during training even if afterwards it is not explicitly used.

Unfortunately, in most cases we will not have access to labeled data, and so $\mathcal{P}$ and $\mathcal{N}$ will be unavailable. To overcome this problem, $\mathcal{P}$ is constructed by deforming the query, similarly to what is done in [18]. In our case, we slightly shift the window around the query word to produce many almost identical, shifted positive samples (see Fig 1(c)). As a side effect, at test time, sliding windows that are not completely centered over a word will still produce a high score, making the approach more resilient to the sliding window granularity. To produce the negative set $\mathcal{N}$, we sample random regions over all the documents after filtering those with very low norm. Note that, since we do not have access to segmented words, we can not guarantee that a given negative region will contain a complete word or a word at all. This is different from unsupervised methods that perform word segmentation such as that of [5]: even if they do not use labeled data, they have access to the bounding boxes of training words. As in [18], positive samples could also appear in this randomly chosen negatives set.

In the most basic setup, this model needs to be learned only once per query independently of the number of documents on the dataset, and so the learning time becomes small compared to the complete retrieval time if the number of documents in the dataset is not small. Still, using a batch solver such as LIBLINEAR [24] as was done in [20] may require between one and two seconds per query, which is not negligible. This may be even worse when using query expansion (*cf*. Section 6), since under this setup it is needed to learn the exemplar model not just once but at least twice per query. Here we propose to use a Stochastic Gradient Descent (SGD)

implementation that can very significantly reduce the training time while retaining good accuracy results. The key idea behind SGD is that the classifier is updated one sample at a time based on the (sub-)gradient of the objective function O (Equation (1) in our case), *i.e.*, $\mathbf{w_{(t+1)}} = \mathbf{w_{(t)}} - \eta \frac{\nabla O}{\partial \mathbf{w_{(t)}}}$, where η is the learning rate or step rate. In our case, that leads to the following update:

$$\mathbf{w_{(t+1)}} = (1 - \lambda\eta)\mathbf{w_{(t)}} + \eta\delta_{it} y_i \mathbf{x}_i, \tag{2}$$

where $\delta_{it} = 1$ if $L(y_i {\mathbf{w_{(t)}}}^T \mathbf{x}_i) > 0$ and 0 otherwise, and λ is a regularization parameter. We consider the set $\mathcal{PN} = \mathcal{P} \cup \mathcal{N}$, and we randomly sample pairs $\{\mathbf{x}_i, y_i\} \in \mathcal{PN}$ and use them to update $\mathbf{w}$ using Equation (2). The initial weight vector $\mathbf{w_{(0)}}$ is initialized randomly from a Normal distribution of mean 0 and variance $1/\sqrt{d}$, where d is the dimensionality of $\mathbf{w}$. The process is repeated for several iterations until convergence. Instead of setting the C_1 and C_2 weights of equation (1) directly, we follow the approach of [25] and control the proportion of negatives to positives samples used at every iteration. As we will see experimentally, the SGD solver leads to results comparable to LIBLINEAR while being approximately 10 times faster.

4. Feature Compression

One important drawback of sliding-window based methods such as this or [9, 12] is the cost of recomputing the descriptors of every image with every new query. As we will see during the experimental evaluation (*cf*. Section 7), computing the HOG descriptors of an image can take between 50% and 90% of the total test time per document, and this has to be recomputed for every new query. A possible alternative could be precomputing and storing the HOG descriptors. However, this is usually not a feasible option because of the large amount of memory that would be necessary to maintain them. Assuming HOG descriptors of 31 dimensions represented with single-precision floats of 4 bytes each (*i.e.*, 124 bytes per HOG), and 45,000 cells per image, storing as few as 1,000 precomputed dataset images would require 5.2GB of RAM. Even if we compress the HOG descriptors with PCA down to 24 dimensions, we can barely fit 250 documents in 1GB of Memory. Since documents have to be kept in RAM to be rapidly accessed, storing large collections of documents would be extremely expensive or directly unfeasible.

In this section we propose to encode the HOG descriptors by means of Product Quantization (PQ) [19]. This technique has shown excellent results on approximate nearest neighbor tasks [26, 27], maintaining a high accuracy while drastically reducing the size of the signatures. After compression, we can store up to 24,000 images per GB of RAM, depending on the PQ configuration. As a side effect, because of the dimensionality reduction, the sliding-window comparisons will also be faster: without PQ, we can analyze approximately 5 documents per second, and that is excluding the time to compute the HOG descriptors of each page. After PQ, we can analyze approximately 70 document images per second, an almost 15 fold improvement in speed.

In the following section we will first give an overview of PQ, and then we will describe how PQ fits in our system, both in the exemplar training and the sliding-window stages.

4.1. Product Quantization

In vector quantization the goal is to find a finite number of reproduction values $C = \{\mathbf{c}_1, \ldots \mathbf{c}_k\}$ (usually through k-means), and then represent the vector with the the closest reproduction value, *i.e.*, given a vector $\mathbf{x} \in \mathbb{R}^D$ and a quantizer $\mathfrak{q}$, then

$$\mathfrak{q}(\mathbf{x}) = \underset{\mathbf{c}_i \in C}{\operatorname{argmin}}\, d(\mathbf{x}, \mathbf{c}_i), \tag{3}$$

where d is usually the Euclidean distance. Assuming k centroids, then quantized vectors can be encoded using as few as $\log_2 k$ bits by storing the indices of the centroids and not the centroids themselves, which are kept in a different table. Unfortunately, vector quantization is not possible when the dimensionality of the vectors is not trivially low: as noted in [19], to encode a descriptor of 128 dimensions using only 0.5 bits per dimension, we would need to compute 2^{64} centroids, which is not feasible.

PQ addresses this issue by quantizing groups of dimensions independently. Given a set of D-dimensional vectors, each vector is split sequentially into m groups of D/m dimensions, and then, a different quantizer is learned for every group of sub-vectors. Assuming k centroids per group, this leads to $m \times k$ centroids. Now, given vector $\mathbf{x}$ that we want to encode, we denote with $\mathbf{x}^j$ the j-th group of $\mathbf{x}$, and with $\mathbf{c}_{ji} \in C_j$ the i-th centroid learned from group j, then

$$\mathfrak{q}_j(\mathbf{x}) = \underset{\mathbf{c}_{ji} \in C_j}{\operatorname{argmin}} \; d(\mathbf{x}^j, \mathbf{c}_{ji}), \tag{4}$$

and $\mathfrak{q}(\mathbf{x}) = \{\mathfrak{q}_1(\mathbf{x}), \mathfrak{q}_2(\mathbf{x}), \ldots, \mathfrak{q}_m(\mathbf{x})\}$. The final codification of vector $\mathbf{x}$ results from the concatenation of the indices of the m centroids. In this case, to produce a b bits code, each quantizer needs to compute only $2^{b/m}$ centroids, which is reasonable if m is chosen appropriately.

In our case, we use PQ to encode our PCA-HOG descriptors of 24 dimensions. In the experiments section we compare the performance when using different number m of groups of $k = 256$ centroids, *i.e.*, 8 bits to represent each one.

One advantage of PQ is that to compute the distance between a query $\mathbf{q}$ and a (quantized) document x from a dataset – represented by m indices to the centroids table –, it is not necessary to quantize the query or to explicitly decode the quantized document. We can, at query time, pre-calculate a look-up table $\ell(j, i) = d(\mathbf{q}^j, \mathbf{c}_{ji})$. Note that this table does not depend on the number of elements of the dataset, only on the number of groups m and centroids k, and so the cost of computing it is negligible if the number of documents is large. Once this look-up table has been constructed, we can calculate the distance between query $\mathbf{q}$ and a quantized document x as $\sum_{j=1}^{m} \ell(j, x_j)$, where x_j is the j-th index of x, without explicitly reconstructing the document.

To learn the Exemplar SVM, we need to create the sets $\mathcal{P}$ and $\mathcal{N}$. As seen in Figure 1(c), positive samples do not exactly align with the precomputed cells, and therefore we need to recompute the descriptors for those regions. Note that this is a small amount of descriptors compared to the whole image. The negative samples, however, can be chosen so that they align with the precomputed grid of HOGs, decoded, and then fed to the SVM training method. Similarly to [28], we decode the features and learn our classifier on the decoded data. When calculating the sliding window, we no longer have to compute the dot-product between the query and the HOG descriptor to obtain the score of a cell, and it is enough to perform m accesses to the look-up table to obtain that score. Intuitively, this could lead to comparisons between 24 (with $m = 1$) and 4 (with $m = 6$) times faster with respect to the 24-dimensional HOG descriptors. In practice, we have obtained, depending on the number m of groups, up to 15 fold speed-ups.

5. Reranking

The main advantage of the HOG-based method proposed in this paper is that it can be efficiently computed over a large dataset of images in combination with a sliding window-based search. This allows one to apply it over non-segmented documents, which is mandatory in many real scenarios where pre-segmenting the words without errors is not possible. One drawback,

though, is that HOG features are in clear disadvantage in a direct comparison in performance with other more complex and informative image representations that unfortunately cannot be applied to all the dataset due to their cost.

In this paper we propose to improve the effectivity of the HOG-based framework using reranking, a popular technique applied in image retrieval [29, 30]: it consists of applying a second ranking step that considers only the best windows retrieved by an initial efficient ranking step, and that uses more discriminative (and costly) features. These features cannot be used over the whole dataset due to their cost, but it is feasible to use them only on a small subset of windows. Common methods in image retrieval use *geometrical verification*, which applies strong spatial constrains, ensuring that both query image and images retrieved contain the same scene [29]. This can be ensured because different views of the same scene can be seen as an affine transformation of their points, and therefore techniques such as Ransac can be used to verify this transformation accurately. However, due to the variability and inconsistency in handwriting, transformations are not affine and we cannot apply this technique in our case.

We therefore analyze other representations that are appropriate for our problem. In the recent [31], the authors perform a similar analysis, showing that a bag of words representation using SIFT descriptors can outperform classic approaches such as DTW based on sequence features as well as graph-based or pseudo-structural descriptors. We build on that work, and analyze how the HOG features compare with respect to DTW using the popular Vinciarelli features [7][4] as well as an encoding based on the bag of words framework, the Fisher Vector (FV) [32].

The Vinciarelli features are extracted by computing local descriptors using a horizontal sliding-window approach over the image. At each region, the local window is divided in a 4×4 grid and the density of each region is computed, leading to a window descriptor of 16 dimensions. The descriptor of the word image is a sequence of such 16-dimensional descriptors, where the exact number depends on the length of the image to represent. Due to their variable length, methods such as DTW are necessary to compare these descriptors.

The Fisher Vector [32] can be seen as a bag of words that captures not only the visual word count but also higher order statistics. The FV was recently shown to be a state-of-the-art encoding method for several computer vision tasks such as image retrieval and classification [33].

To the best of our knowledge, the only work that applies unsupervised reranking in a word-spotting context is that of [34]. In this case, a bag of words was used to represent images and to perform hashing, which allowed a fast but noisy retrieval. On a second stage, spatial pyramids were used to improve the representation adding spatial information, which was missing on the indexing stage. Note that in this case the reranking uses the same features, and the spatial pyramid can be seen as a way to calibrate the score of each region of the word independently. In our case, we are not just adding spatial information but using more powerful and discriminative features.

6. Query Expansion

Although we focus on a completely unsupervised case, and therefore only one image representation of the query is usually available at first, this framework could be improved if several instances of the query word became available, since a more discriminative model could be learned. For example, we could make use of query expansion, a technique popular in instance-level image retrieval works [29, 30, 35].

[4]These features are more discriminative than the ones used in [31].

Query expansion is based on improving the representation of the query using retrieved and verified images from the dataset. In the work of Chum et al. [29] a number of the best ranked images from the original query are verified using strong spatial constrains, and the validated images are combined into a new query. They use BoW [13] as the image representation and they average together the query BoW vector and BoW vectors of the new images. Arandjelović and Zisserman [30] introduce a discriminative query expansion approach, where negative data is also taken into account and a classifier is trained. In a different work [36], they use multiple queries to retrieve images extracted from a text search in Google images and they propose different ways to combine either their representations or their retrieved results. Query expansion has also been applied to word spotting, although in combination with relevance feedback, by Rusiñol and Lladós [37]. Their system asks the user to cast several queries instead of a single one and combines the results by testing different strategies. To the best of our knowledge, there exist no word-spotting methods that perform query expansion in a completely unsupervised way.

Here, like in [29], we assume that the best results retrieved correspond to the query and can be used to improve its representation. However, due to the variability and inconsistency of handwriting text, we cannot apply the geometric constrains generally used in natural images. Instead, we use as a verification step the reranking process described in Section 5. Once results retrieved by the sliding-window search are reranked using more informative features, we define a set $\mathcal{X}$, composed by the k best windows retrieved and the original query. This is used as the set of positive examples of the query model. Although we have no absolute guarantees that the set $\mathcal{X}$ will contain only positive samples, we observed a significant improvement of the accuracy on the tested datasets.

In this paper we explore two different ways of combining and exploiting this new set of positive examples in the HOG-based Exemplar SVM framework:

Single-Exemplar: This approach consists of training a single Exemplar SVM with $\mathcal{X}$ as the set of positive images. We build set $\mathcal{P}$ of relevant examples by applying the shifting deformation of the window as described in Section 3 and showed in Figure 1(c) to every sample in $\mathcal{X}$, producing many almost identical, shifted windows. Training results in a single weight vector $\mathbf{w}$, which is used again to retrieve new windows.

Multi-exemplar: It consists of training one Exemplar SVM, as detailed in Section 3, for every sample in $\mathcal{X}$. It results in a set $\mathbf{W}$ that contains the weight vector $\mathbf{w}_i$ of every Exemplar SVM. Finally, retrieved ranked lists are combined by scoring each window by the average of the individual scores obtained from each Exemplar SVM: $\frac{1}{k}\sum_{i=0}^{k}\text{sim}(\mathbf{w}_i, \mathbf{x})$.

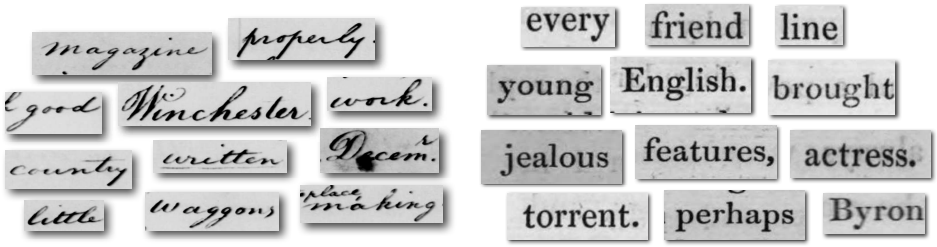
Finally, set $\mathcal{X}$ can also be used to improve the FV representation of the query and perform a second reranking process. We use as the new representation of the query the average FV of the representations extracted from the samples in set $\mathcal{X}$. Then, it is used to rerank the first regions retrieved by the *expanded model* of the query, as it is described in Section 5.

7. Experiments

7.1. Experimental Setup

Datasets and Performance Evaluation. We evaluate our approach on two public datasets: The George Washington (GW) dataset [38, 2] and the Lord Byron (LB) dataset [12][5]. These

[5]We obtained the exact images and groundtruth after direct communication with the authors of [12].



(a) GW (b) LB

Figure 3: Examples of the words contained in the (a) GW and (b) LB datasets.

datasets are comprised of 20 pages each and contain approximately 5,000 words. George Washington (Figure 3(a)) contains handwritten text while Lord Byron (Figure 3(b)) only contains typewritten text. We follow a similar protocol as used in [12]: each word is considered as a query and used to rank all the regions of every document in the dataset. However, as opposed to [12], the query image, if retrieved, is removed from the retrieved results and not considered in the performance evaluation. This is consistent with most other works on word spotting. For compatibility reasons, however, we will also report results without removing the query using our final system to ease the comparison with [12], since it is the work most related to ours. A region is classified as positive if its overlap over union with the annotated bounding box in the groundtruth is larger than 50%, and negative otherwise. For every document, we keep only the 1,000 regions with the highest score and perform NMS to avoid overlaps of more than 20%. Finally, we combine the retrieved regions of all the documents and rerank them according to their score. We report the mean Average Precision (mAP) as our main measure of accuracy. The mAP is a standard measure in retrieval systems and can be understood as the area below the precision-recall curve.

Parameters. To compute the HOG grid, we use cells of 12 pixels, which resulted in a reasonable trade-off in performance and time consumption. On both datasets, using this cell size produces approximately 45,000 cells for each document. For the sliding-window search we have also fixed the step size to one cell. An extended analysis on the effect of different cell sizes and step sizes can be found in [20]. When performing the unsupervised learning of PCA and PQ, we randomly sample 10,000 HOGs from all the documents, after filtering those with very low norm, and 1.5 million SIFTs for the unsupervised learning of the GMM for the FV representation. When learning the Exemplar SVM, we used the Stochastic Gradient Descent-based solver of the JSGD package [25] and fixed the step size η to 10^{-3} and the regularization parameter λ to 10^{-5} based on the results of a small subset of 300 queries. This provided reasonable results on both datasets, although better results could be obtained by fine-tuning this parameter individually for each dataset. We produce 121 positive samples (11 shifts in horizontal $\times$ 11 shifts in vertical) for each query (and for each expanded example in case that query expansion is used) and 7,744 negative samples (64 times the number of positive samples of the query). Increasing the number of negative samples led to a very slight improvement in the accuracy, but we did not consider it worth the extra cost during training. Note that [17, 18] propose to use several iterations of hard negative mining. However, we did not experience any gain in accuracy by doing so. Finally, we evaluate reranking using different k for selecting the *top-k* retrieved results to be reranked and query expansion using different number of examples to expand the query.

We used MATLAB's built-in profiler to measure the running times of the different sections of our pipeline on an Intel Xeon running at 2.67GHz using one single core. We noticed that the profiler added approximately a 10% overhead that we have not subtracted. Therefore, the actual times may be slightly faster than reported. Note that, although we used MATLAB, the core sections (computing HOG and FV descriptors, training the SVM, calculating the scores with a sliding window, and the NMS) were implemented in C. When reporting the sliding-window times, these already include the time to perform the ranking and the NMS.

7.2. Word Spotting Results and Discussion

In [20], we already analyzed the influence of the parameters of the basic configuration of the proposed approach: the cell size of the HOG descriptor and the step size of the sliding-window search. Here, we will focus on evaluating the impact of the new contributions included in this paper. First, we will analyze the performance of different features for representing word images. For that, we will use a simplified version of the datasets where the document words have already been segmented using the ground truth annotations. Then, in a segmentation-free setting and relying on a sliding-window approach, we will analyze the influence of the SGD solver and the different configurations of PQ, reranking and query expansion. Finally, we will compare the whole system with recent word-spotting methods.

Comparison of Features. We begin our experiments by testing the proposed method on a simplified setting, where the document words have already been segmented using the ground truth annotations. In this setting, we test the effectivity of our HOG and HOG+SVM approaches, and compare it with a descriptor based on Vinciarelli features with a DTW distance, as well as with a FV representation. We also study the computational needs of each approach. The objective of this test is twofold. First, to show that, although the accuracy of the proposed HOG+SVM approach may be outperformed by the Vinciarelli features or the FV, the accuracy of the HOG+SVM is reasonably high. Second, to highlight that neither the Vinciarelli nor the FV approaches can be used in a segmentation-free approach due to their computational costs, but can still be used to rerank a short list of results produced by the HOG+SVM approach.

Under this setup, different words may have different numbers of HOG cells, which affects their dimensionality. To compare a query and a document word of different sizes, the document word is first slightly enlarged by a 10%, and then the query word is "searched" inside the document word using a sliding-window approach. The best window score is used as a measure between the query and the word.

To compute the Vinciarelli features, we experimented with several region sizes and used the one that yielded the best results, which gives it an slightly unfair advantage. To compute the FV we densely extract SIFT features from the images and use a Gaussian mixture model (GMM) of 16 Gaussians. To (weakly) capture the structure of the word image, we use a spatial pyramid of 2×6 leading to a final descriptor of approximately $25,000$ dimensions.

The results are shown in Table 1.

Dataset	HOG	HOG+SVM	Vinciarelli+DTW	FV
GW	40.24	49.19	56.25	64.90
LB	75.37	83.04	83.47	91.75

Table 1: Retrieval performance in mAP of different descriptors for segmented words.

We observe how, indeed, the powerful FV features obtain the best results on both datasets

despite being a fixed-length feature. The Vinciarelli features with DTW outperform HOG and HOG+SVM on the GW dataset, suggesting that HOG is quite rigid for the type of variations found in this dataset, although it still achieves reasonable results. On the LB dataset, where variations are much smaller, HOG+SVM performs similar to the Vinciarelli features, despite the unfair advantage of the Vinciarelli features.

Regarding the computational costs, there are two separate issues: computing the descriptor given a document window, and comparing the descriptors. With our setup, searching a query in a document page requires on average to compute and compare about $40,000$ window descriptors. Because of this, computing and comparing window descriptors needs to be extremely fast to be feasible on a segmentation-free setting.

- Descriptor computation. On the HOG and HOG+SVM setup, the HOG cells of a page can be precomputed and stored offline. At test time, computing the descriptor of a window requires only to access the corresponding elements of the precomputed grid, and so the cost is negligible. A similar technique can be applied for the Vinciarelli features by precomputing an integral image of aggregated densities over a document page. At test time, the descriptor for a window can be directly computed based on the precomputed statistics. Unfortunately, such techniques cannot be applied to the FV formulation, and it needs to be calculated independently each time. With our optimized implementation, we can compute approximately 60 FVs per second. Computing the $40,000$ descriptors per page would require approximately 700 seconds, which makes it unfeasible. Precomputing the descriptors is also not possible, since only one page would require approximately 4Gb of memory.

- Descriptor comparison. On the case of HOG, HOG+SVM, and FV, comparing descriptors is fast since it is based on dot-products of vectors of same size. The distances between a FV query and $1,000$ FV words can be computed in less than 15 ms, and comparing the HOG descriptors is an order of magnitude faster due to their reduced dimensionality. However, comparing the Vinciarelli features is slower since it is based on DTW. Again, using our optimized DTW implementation in C, comparing one query against $1,000$ words takes on average 350ms, a hundred times slower than comparing HOGs.

Because of these reasons, the underlying costs of Vinciarelli and FV makes them unfit for a segmentation-free task. However, it is possible to perform the segmentation-free search using the HOG+SVM approach with a reasonable success, and use the FV features to rerank only the best scored candidates.

Influence of SVM solvers. Here we compare the results of the Exemplar SVM over HOG descriptors using a batch solver as was done in [20] and using an SGD implementation based on [25]. In this case, as we did in the previous experiment, we do not use the annotations to segment the words and rely on the sliding-window approach. We do not use neither PQ compression nor reranking or query expansion for this experiment. Results are shown in Table 2. We compare the accuracy in mAP for both datasets and compare it with the cosine approach, that does not use any training. We also compare the training time on the GW dataset – training times on the LB dataset are extremely similar. We can observe how the performances of both LIBLINEAR and the SGD implementation are very similar, with differences that are not significative when compared with the cosine approach, *i.e*. the baseline system. However, in terms of speed, the SGD implementation is almost ten times faster than LIBLINEAR. Through the rest of the experiments, we will use the SGD implementation.

	GW		LB
	mAP	Time (ms/query)	mAP
Cosine	31.86	-	66.34
LIBLINEAR	38.28	1,090	78.01
SGD	38.16	124	77.91

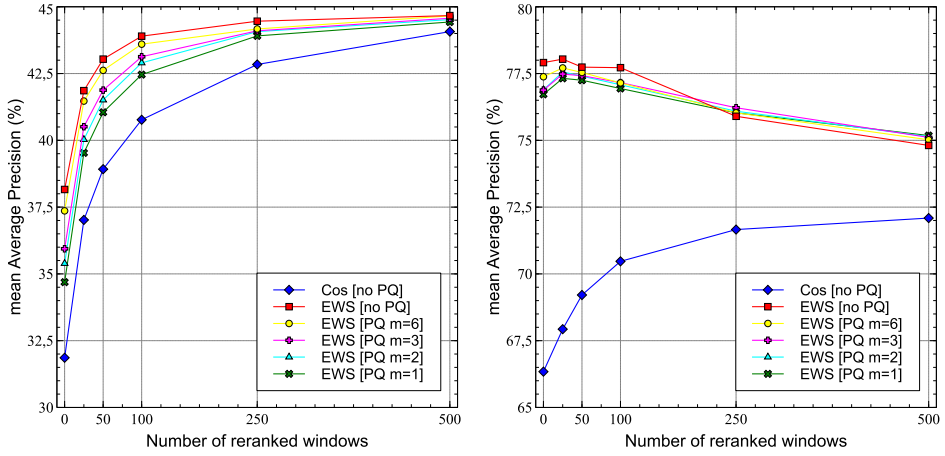
Table 2: Comparison of LIBLINEAR and SGD on terms of accuracy in mAP and training time in milliseconds.

Influence of PQ. We study the influence of PQ using different number of subquantizers, from $m = 1$, *i.e.*, one subquantizer of 8 bits for 24 dimensions (compression ration of 1:96), to $m = 6$, *i.e.*, one subquantizer of 8 bits for 4 dimensions (compression ratio of 1:16). We also consider the average time in milliseconds needed to scan one page using a sliding window. Results are shown in Table 3. After PQ, the accuracy of the methods suffers a small drop, especially when we use a single quantizer for each HOG feature, *i.e.*, $m = 1$. However, as long as we use more quantizers per cell, this drop is reduced until it becomes insignificant. The difference between applying PQ with $m = 6$ and not applying PQ is less than 1% absolute. Moreover, the sliding-window times become between 3 and 15 times faster, depending on the configuration, and a much larger number of documents can fit in memory at the same time. We should also consider that when using PQ one does not need to compute the HOG descriptor of every document for every new query since they are precomputed, saving approximately 500ms per document and query.

	GW			LB
	mAP	ms/doc	docs/GB	mAP
Cosine [no PQ]	31.86	218	~ 0.25K	66.34
EWS [no PQ]	38.16	218	~ 0.25K	77.91
EWS [PQ m=6]	37.36	86	~ 4K	77.38
EWS [PQ m=3]	35.94	45	~ 8K	76.88
EWS [PQ m=2]	35.39	34	~ 12K	76.89
EWS [PQ m=1]	34.69	14	~ 24K	76.72

Table 3: Comparison of different numbers of quantizers used in PQ on terms of accuracy in mAP, time to perform a sliding-window search in milliseconds per document, and number of pages that fits in one Gigabyte of memory. Since time and space consumption is extremely similar for both datasets we only report numbers for GW.

Effect of Reranking. We study the effect in performance obtained by the reranking process using FV as representation as a function of the number of the first retrieved windows that have been reranked. Figures 4(a) and 4(b) show the accuracy in mAP for different configurations of the method for both GW and LB datasets. We plot the performance of the method using cosine similarity and EWS with different configurations of the PQ compression (without compression and compressing using from $m = 1$ to $m = 6$ quantizers). We see that, as the number of windows increases, all the configurations using Exemplar SVMs converge in mAP independently of the amount of compression used, showing that the FV representation can compensate the high compression used during the first ranking step. On GW, accuracy increases as the number of windows increases and reaches a plateau at approximately 45% of mAP using 500 windows. The effect in LB is different. When using the inferior cosine similarity, reranking does indeed improve the results significantly. However, when using the Exemplar model, reranking actually decreases the performance when reranking more than 25 windows. We believe the reason is that, for the rigidity of this dataset, the very structured cell of HOG descriptors is more accurate than the FV



(a) GW (b) LB

Figure 4: Retrieval results in mAP for different configurations in the (a) George Washington and (b) Lord Byron datasets.

representation, that only uses weak geometrical structure through the spatial pyramid. Note that although in Table 1 we showed that FV obtained better results than HOG, the difference was quite small in this dataset.

Regarding the computational cost of reranking, the time that is needed to extract the FV of a given window and compute the new score similarity with the query is of approximately 20 milliseconds. The cost of performing the actual reranking is negligible since it only involves sorting a few tens or hundreds of values.

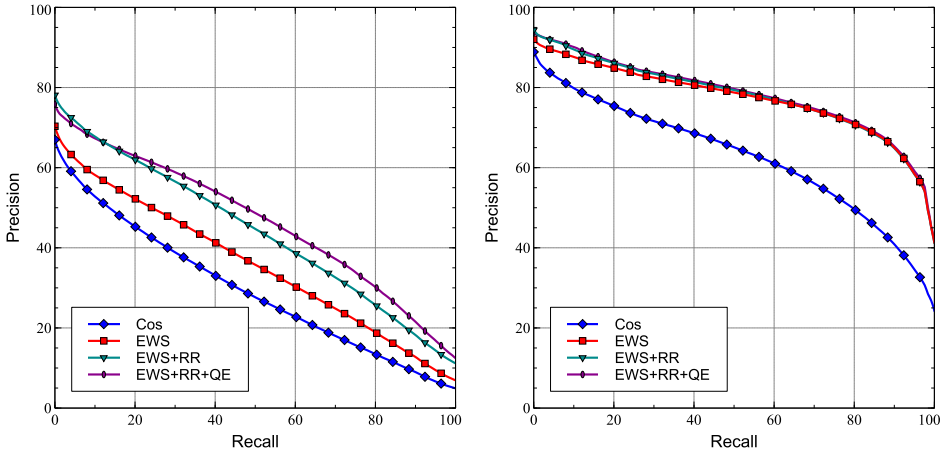
Effect of Query Expansion. Finally, we study the influence in performance of the query expansion process, as a function of the number of windows used to expand the set of positive samples, for both combination models proposed, *single-exemplar* and *multi-exemplar*. Here we combine the use of EWS, PQ and reranking previous and posterior to query expansion. We set the the number m of quantizers used in PQ to 3 as a good trade-off between accuracy and time consumption. We also set the number of windows used for the first reranking to 100 on the GW dataset and 25 on the LB dataset. This offers a reasonable trade-off based on the results reported in Figure 4. For the second reranking, we use 100 and 250 windows on the GW dataset and 25 on the LB dataset.

We show the results in Table 4. We see that expanding the query with the first retrieved windows after reranking leads to improved results. Interestingly, only a small number of windows has to be used: 2 in the GW dataset and 1 in LB; otherwise the results do not improve as much and can even worsen in the case of LB. One of the reasons behind this is that, for many queries, the number of relevant items is very small. On both GW and LB datasets, about 15% of the queries have 2 or less relevant items. On those cases, including more results in the query expansion will inevitably add negative samples to the set $\mathcal{X}$, degrading the results.

Regarding the ways to combine the new samples with the query, *multi-exemplar* has a very slight edge over *single-exemplar*, which is consistent with the main idea of Exemplar SVMs. However, note that *multi-exemplar* has to perform as many Exemplar SVM trainings as elements the set of positive samples contains, contrary to *single-exemplar*, which only requires one train-

	expanded samples		0	1	2	4
GW	EWS RR: 100 RR2: 100	single-exemplar	43.13	45.55	45.77	44.88
		multi-exemplar	43.13	45.63	45.94	45.32
	EWS RR: 100 RR2: 250	single-exemplar	44.10	46.33	46.53	45.62
		multi-exemplar	44.10	46.35	46.58	45.84
LB	EWS RR: 25 RR2: 25	single-exemplar	77.71	77.91	76.83	75.01
		multi-exemplar	77.71	78.35	77.56	76.14

Table 4: Influence of the number of examples to expand the query for different number of windows reranked. The number of windows reranked in the reranking previous to query expansion has been fixed to 100 for GW and 25 for LB.



(a) GW (b) LB

Figure 5: Precision-recall curves for different configurations in the (a) George Washington and (b) Lord Byron datasets.

ing. This could make, for some situations, not worth the extra cost.

Finally, in Figure 5 we show precision-recall plots of the different approaches that we presented on both GW and LB datasets. We observe how indeed the Exemplar learning has a very large influence in the results on both datasets, showing the importance of learning more discriminative signatures. On GW, the reranking and query expansion also lead to a significant improvement due to the superiority of the FV representation and the variability of the data, while on LB, where the variability is much smaller, reranking and query expansion only bring minimal improvements.

Comparison with other methods. We first compare our results with the segmentation-based methods of [38] and [5]. The authors of [38] report results between 52% and 65% mAP on the GW dataset depending on the particular fold they evaluate with, while [5] reports a 54% mAP, which should be compared to our 46.58% using reranking and query expansion. Note however that these results are not completely comparable since i) they use different query/database partitions, with smaller database sets, which benefits the mAP metric, and ii) they work on already segmented words, *i.e.*, they are not segmentation-free.

Additionally, in order to be able to compare with other segmentation-free methods that reports experiments in these datasets [12, 11, 14], we run again our method using the same final con-

figuration under their evaluation protocols. All these methods include the query in the retrieved results and therefore in the mAP computation. Moreover, [14] uses an overlap over union of 20% with the groundtruth to classify a window as positive, instead of the more traditional 50%, and [11] only considers as queries the words with more than 5 characters. Finally, the work of [15] also reports experiments in GW, but the results are not comparable since it performs a reranking step based on segmentation to avoid substring matching. We show the results in Table 5, where we can see that our baseline system already obtains quite reasonable results. Moreover, if we use our full system, including EWS, PQ, reranking and query expansion, we considerably outperform the work of [12] for both GW and LB datasets. With this full system, and using their respective protocols, we also outperform [14] and [11].

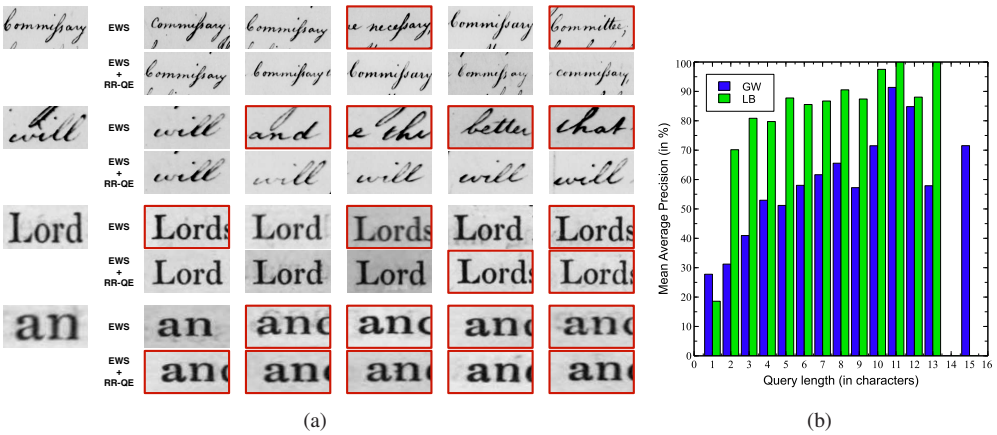
	GW	LB
Rusiñol et al. [12]	30.42	42.83
Cos noPQ	48.66	74.04
EWS+PQ	51.88	84.34
EWS+PQ+RR	57.46	84.51
EWS+PQ+RR+qe+RR2	59.13	84.04
Rothacker et al. [14] (overlap 20%)	61.10	–
EWS+PQ (overlap 20%)	59.51	–
EWS+PQ+RR+qe+RR2 (overlap 20%)	68.88	–
Zhang and Tan [11] (queries > 5 characters)	62.47	–
EWS+PQ (queries > 5 characters)	72.85	–
EWS+PQ+RR+qe+RR2 (queries > 5 characters)	82.23	–

Table 5: Retrieval performance in mAP and comparison with state-of-the-art when query is included in the results. Methods have been set to the best parameters.

Finally, Figure 6(a) illustrates some typical failure cases, such as confusions with similar-shaped words, giving too much weight to artifacts in the query word, or retrieving substrings from longer words. We also observe how reranking and query expansion address the first two issues by constructing more discriminative queries that are more informative and more independent of the background. The third problem is one drawback of the segmentation-free, sliding-window methods since, as opposed to DTW, they do not penalize matching a short word with a substring of a long word. As observed in Figure 6(b), this can very significantly reduce the mAP of short queries. However, this may be also seen as an interesting feature that could be exploited for sub-word searches.

8. Conclusions

In this paper we have shown how a combination of HOG descriptors and sliding windows can be used to perform segmentation-free, unsupervised word spotting, both on handwritten and machine-printed text. This method can be extended using Exemplar SVMs to represent the queries, improving the results at a minimum extra cost at query time. We have shown how the HOG descriptors can be aggressively compressed with Product Quantization with only a small loss in accuracy. Finally, we have shown that results can be improved by, first using more informative and discriminative features in a reranking step of the best windows retrieved, and second using some of these windows to expand the Exemplar SVM training set and improve the



(a) (b)

Figure 6: a) Failure cases. For every query we show a first row with the results retrieved by EWS and a second row with the results retrieved by EWS combined with reranking and query expansion. First query: words have a very similar shape. Second query: an artifact in the query leads to results with the same artifact. Third and fourth query: we detect the query word as a substring of a longer word. This is common when querying short words. Query expansion and reranking are able to alleviate some of the problems. b) Mean Average Precision as a function of the query length for the system combining EWS, PQ, reranking and query expansion.

query representation. We have obtained excellent results when comparing to other segmentation-free methods in the literature. We have published the MATLAB code implementation for training and testing the Exemplar Word Spotting [39], as well as experiments with other datasets [40], in the hope that it would ease the comparison for new works on word spotting.

Finally, we would like to note that the variability of the writing style on the datasets that have been used in the experiments is very small. This variability is usually much higher in a multi-writer scenario. For example, two writers may write the same word with very different widths, and only one window size will not be able to correctly capture both of them. We believe that, as presented, our method would have difficulties in this scenario. To overcome this problem, a possible solution would be to severely deform the query varying the stretch and the slant, and learn several Exemplar SVMs as in [17]. This would increase the query time, but not the memory required to store the datasets.

Acknowledgments

J. Almazán, A. Fornés, and E. Valveny are partially supported by the Spanish projects TIN2011-24631, TIN2009-14633-C03-03, TIN2012-37475-C02-02, by the EU project ERC-2010-AdG-20100407-269796 and by a research grant of the UAB (471-01-8/09).

References

[1] R. Manmatha, C. Han, E. M. Riseman, Word spotting: A new approach to indexing handwriting, in: IEEE Computer Vision and Pattern Recognition, 1996, pp. 631–637.

[2] T. Rath, R. Manmatha, Word spotting for historical documents, International Journal on Document Analysis and Recognition (2007) 139–152.

[3] A. Fischer, A. Keller, V. Frinken, H. Bunke, HMM-based word spotting in handwritten documents using subword models, in: International Conference on Pattern Recognition, 2010, pp. 3416–3419.

[4] V. Frinken, A. Fischer, R. Manmatha, H. Bunke, A novel word spotting method based on recurrent neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2012) 211–224.
[5] J. Rodríguez-Serrano, F. Perronnin, A model-based sequence similarity with application to handwritten word-spotting, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2012) 2108–2120.
[6] U.-V. Marti, H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems, International Journal of Pattern Recognition and Artificial Intelligence (2001) 65–90.
[7] A. Vinciarelli, S. Bengio, H. Bunke, Offline recognition of unconstrained handwritten texts using HMMs and statistical language models, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 709–720.
[8] J. Rodríguez-Serrano, F. Perronnin, Local gradient histogram features for word spotting in unconstrained handwritten documents, in: International Conference on Frontiers in Handwriting Recognition, 2008.
[9] B. Gatos, I. Pratikakis, Segmentation-free word spotting in historical printed documents, in: International Conference on Document Analysis and Recognition, 2009, pp. 271–275.
[10] Y. Leydier, A. Ouji, F. Lebourgeois, H. Emptoz, Towards an omnilingual word retrieval system for ancient manuscripts, Pattern Recognition 42 (2009) 2089–2105.
[11] X. Zhang, C. L. Tan, Segmentation-free Keyword Spotting for Handwritten Documents based on Heat Kernel Signature, in: International Conference on Document Analysis and Recognition, 2013, pp. 827–831.
[12] M. Rusiñol, D. Aldavert, R. Toledo, J. Lladós, Browsing heterogeneous document collections by a segmentation-free word spotting method, in: International Conference on Document Analysis and Recognition, 2011, pp. 63–67.
[13] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray, Visual categorization with bags of keypoints, in: Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision, 2004, pp. 1–22.
[14] L. Rothacker, M. Rusiñol, G. A. Fink, Bag-of-Features HMMs for Segmentation-free Word Spotting in Handwritten Documents, in: International Conference on Document Analysis and Recognition, 2013, pp. 1305–1309.
[15] N. R. Howe, Part-Structured Inkball Models for One-Shot Handwritten Word Spotting, in: International Conference on Document Analysis and Recognition, 2013, pp. 582–586.
[16] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Vision and Pattern Recognition, 2005, pp. 886–893.
[17] T. Malisiewicz, A. Gupta, A. Efros, Ensemble of Exemplar-SVMs for object detection and beyond, in: International Conference on Computer Vision, 2011, pp. 89–96.
[18] A. Shrivastava, T. Malisiewicz, A. Gupta, A. A. Efros, Data-driven visual similarity for cross-domain image matching, ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA) 30 (2011) 154:1–154:10.
[19] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2011) 117–128.
[20] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Efficient exemplar word spotting, in: British Machine Vision Conference, 2012, pp. 1–11.
[21] F. Perronnin, J. Rodríguez-Serrano, Fisher kernels for handwritten word-spotting, in: International Conference on Document Analysis and Recognition, 2009, pp. 106–110.
[22] T. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, in: Neural Information Processing Systems, 1999, pp. 487–493.
[23] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramaman, Object detection with discriminatively trained part based models, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 1627–1645.
[24] R. Fan, K. Chang, C. Hsieh, X. Wang, C. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.
[25] Z. Akata, F. Perronnin, Z. Harchaoui, C. Schmid, Good practice in large-scale learning for image classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2013) 507–520. `http://lear.inrialpes.fr/src/jsgd/`.
[26] H. Jégou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: IEEE Computer Vision and Pattern Recognition, 2010, pp. 3304–3311.
[27] H. Jégou, R. Tavenard, M. Douze, L. Amsaleg, Searching in one billion vectors: re-rank with source coding, in: International Conference on Acoustics, Speech, and Signal Processing , 2011.
[28] J. Sánchez, F. Perronnin, High-dimensional signautre compression for large-scale image classification, in: IEEE Computer Vision and Pattern Recognition, 2011, pp. 1665–1672.
[29] O. Chum, J. Philbin, J. Sivic, M. Isard, A. Zisserman, Total recall: Automatic query expansion with a generative feature model for object retrieval, in: International Conference on Computer Vision, 2007, pp. 1–8.
[30] R. Arandjelović, A. Zisserman, Three things everyone should know to improve object retrieval, in: IEE Computer Vision and Pattern Recognition, 2012, pp. 2911–2918.
[31] J. Lladós, M. Rusiñol, A. Fornés, D. Fernández, A. Dutta, On the influence of word representations for handwritten word spotting in historical documents, International Journal of Pattern Recognition and Artificial Intelligence 26

(2012).
[32] F. Perronnin, J. Sánchez, T. Mensink, Improving the Fisher Kernel for large-scale image classification, in: European Conference on Computer Vision, 2010, pp. 143–156.
[33] K. Chatfield, V. Lempitsky, A. Vedaldi, A. Zisserman, The devil is in the details: an evaluation of recent feature encoding methods, in: British Machine Vision Conference, 2011, pp. 1–12.
[34] R. Shekhar, C. Jawahar, Word Image Retrieval using Bag of Visual Words, in: International Workshop on Document Analysis Systems, 2012, pp. 297–301.
[35] O. Chum, A. Mikulík, M. Perdoch, J. Matas, Total recall II: Query expansion revisited, in: IEE Computer Vision and Pattern Recognition, 2011, pp. 889–896.
[36] R. Arandjelović, A. Zisserman, Multiple queries for large scale specific object retrieval, in: British Machine Vision Conference, 2012, pp. 1–11.
[37] M. Rusiñol, J. Lladós, The Role of the Users in Handwritten Word Spotting Applications: Query Fusion and Relevance Feedback, in: International Conference on Document Analysis and Recognition, 2012, pp. 55–60.
[38] T. Rath, R. Manmatha, Word image matching using dynamic time warping, in: IEEE Computer Vision and Pattern Recognition, 2003, pp. 521–527.
[39] J. Almazán, A. Gordo, Exemplar Word Spotting library, `http://almazan.github.io/ews/`.
[40] D. Fernández-Mota, J. Almazán, N. Cirera, A. Fornés, J. Lladós, BH2M: the Barcelona Historical Handwritten Marriages database, in: International Conference on Pattern Recognition, 2014.