SIFT-Based Local Image Description Using Sparse Representations

Joaquin Zepeda #1, Ewa Kijak *2, Christine Guillemot #3

# *INRIA, Centre Rennes - Bretagne Atlantique*
Rennes, France
1 joaquin.zepeda@irisa.fr
3 christine.guillemot@irisa.fr

* *Université de Rennes 1, IRISA*
INRIA, Centre Rennes - Bretagne Atlantique, Rennes, France
2 ewa.kijak@irisa.fr

***Abstract*—This paper addresses the problem of efficient SIFT-based image description and searches in large databases within the framework of local querying. A descriptor called the *bag-of-features* has been introduced in [1] which first vector quantizes SIFT descriptors and then aggregates the set of resulting codeword indices (so-called *visual words*) into a histogram of occurence of the different visual words in the image. The aim is to make the image search complexity tractable by transforming the set of local image descriptor vectors into a single sparse vector as sparsity particularly permits efficient inner product calculations. However, aggregating local descriptors into a single histogram decreases the discerning power of the system when performing local queries. In this paper, we propose a new approach that aims to enjoy the complexity benefits of sparsity while at the same time retaining the local quality of the input descriptor vectors. This is accomplished by searching for a sparse approximation of the input SIFT descriptors. The sparse approximation yields a sparse vector per local SIFT descriptor, and helps preserving local description properties by using each sparse-transformed descriptor independently in a voting system to retrieve indexed images. Our system is shown experimentally to perform better than histogram based systems under query locality, albeit at an increased complexity.**

I. Introduction

Content-based image retrieval [2] aims at finding an image in a database by using a query made of a user-selected image region instead of a text. The problem can be seen as consisting of two (not necessarily disjoint) sub-problems: (i) designing a good image descriptor (one that is invariant to image transformations) and (ii) designing a good way of indexing such descriptors (one that permits low-complexity match retrieval). Regarding the first problem, *local descriptors* are the most appropriated [2]: multiple such descriptors are obtained each from a local region of the underlying image. The most successful local descriptor [3] is the SIFT descriptor [4]. It consists of a 128 dimensional histogram of angles of the differential gradients of the pixels' intensity. To address the second problem, Sivic and Zisserman have recently introduced the so-called *bag-of-features* description [1]. This approach allows using the *inverted file* index of the text search community [5] for content-based image retrieval with tractable complexity.

The *bag-of-features* image description is constructed as follows. A set of N_I local SIFT descriptors $\underline{s}_d$, $d = 1, \ldots, N_I$ is first computed on MSER regions [6] that are the most stable affine covariant regions [6]. These SIFT vectors are then quantized (using vector quantization) on codewords (called *visual words*) of a codebook trained on SIFT descriptors extracted from a large image dataset. Let q_d, $d = 1, \ldots, N_I$, be the quantization indices of the N_I local descriptors $\underline{s}_d$ of an image. The *bag-of-features* $\underline{w}$ is defined as a weighted histogram of the q_d indices. For sufficiently large codebooks, $\underline{w}$ is very sparse. This approach based on vector quantization which selects the nearest codeword in the vector quantization codebook, has been generalized in [7] by keeping multiple nearest codewords. Thanks to the sparsity of $\underline{w}$, the *bag-of-features* approach reduces the search complexity, compared with the initial SIFT local descriptors: only elements at common non-zero positions need to be multiplied, when computing inner-product distances between the query descriptors and the descriptors stored in the database. The sparse vectors can indeed be stored in a row-major matrix structure, known as *inverted file* in the text search community, whose row entries allow to access all descriptor vectors having a non-zero component at the corresponding row position. This structure permits efficient access to the data, as descriptors in the database having coefficients at positions common to the query vectors coefficients are stored contiguously in memory.

Nonetheless, the sparsity of *bag-of-features* is attained by sacrificing the locality of the input descriptors $\underline{s}_d$ during the histogram construction. This is particularly harmful in the context of local querying. Treating local requests is an important functionality of image search systems as it makes possible to seek particular objects. In this paper, two methods are described which, rather than building a histogram to induce sparsity, transforms each local image SIFT descriptor $\underline{s}_d$ into a sparse vector $\underline{x}_d$ individually. In the first approach, the sparse representation of each local SIFT descriptor is derived by keeping the k-nearest codewords in a trained codebook. In contrast with [7], the image is described here by the resulting set of N_I sparse vectors rather than by a histogram of these vectors. The second method uses sparse decomposition algorithms [8], [9], [10] and in particular the basis pursuit algorithm [11] to derive a sparse approximation of each local SIFT descriptor. Note that sparse approximations have already been considered in pattern recognition problems, e.g., for finding approximate nearest neighbors search in [12], or for constructing a mathematical model for the manifold composed of pattern's transformations in [13]. The resulting manifold is then a continuous, connected set of points in pattern space. Querying using a given input pattern thus amounts to finding the manifold that is the *closest* (eg., in the euclidean sense) to the input pattern (a single point in pattern space).

The complexity and the performance of proposed search methods are assessed comparatively to the *bag-of-features* approaches [1], [7] as a function of the query locality (i.e., size of the query area). The comparison is performed using analytical derivations as well as simulations on an image database.

The rest of the paper is organized as follows. Section II briefly reviews the *bag-of-features* systems and analyses their behavior under local querying. Section III presents the sparse local descriptors constructed by taking the k-NN approximations of the SIFT vectors. It then presents the local descriptors constructed by using the basis pursuit algorithm. The voting system based on the inverted files principles which has been used to assess the performance of the local descriptors in the image retrieval application is then described. A complexity analysis, in comparison with the *bag-of-features*, is also given. Section IV gives performance evaluation results as a function of the complexity and the locality of the search using the Holidays image database.

For notational clarity, we use un-emphasized letters like a to denote scalars, while underlines as in $\underline{a}$ denote vectors. The i-th element of a vector $\underline{a}$ is denoted as a_i. The k-th vector in a set of vectors is $\underline{a}_k$, and its i-th element is denoted $a_{k,i}$.

II. Semi-Local Searches Using Bag-of-Features

In this section, after reviewing the *bag-of-features* systems, we carry out an analysis of the score of a correct response under local queries to illustrate how the locality property of the initial SIFT descriptors is affected by the *bag-of-features* algorithm.

Searches using local descriptors require establishing correspondences between sets of descriptor vectors, associated to the query image and to one of many indexed images. Local description algorithms can yield hundreds or even thousands of descriptors per image. In addition, the descriptor vectors are high dimensional, 128-dimensional for the case of the SIFT descriptor [4]. This high density of local descriptors combined with their high-dimensionality renders an exhaustive search of matches impossible under reasonable time constraints. Their high-dimensionality further proscribes traditional data structures that have complexity that can grow exponentially with dimensionality.

Due to the high-dimensionality d of local descriptors $\underline{s}$ ($d = 128$ for the case of SIFT) and their large density per image (hundreds or thousands), the *bag-of-features* scheme [1], [7] aims to reduce search complexity by reducing searches to time and memory efficient inner product calculations between sparse vectors $\underline{w}^1$. Let N_I be the number of local descriptors in an image, and $|D|_c$ the codebook size with $|D|_c >> d$. The i-th component of a *bag-of-features* $\underline{w}^1$ of size $|D|_c$ is written as:

$$w^1 i = \frac{1}{\alpha} f^1{}_i \log \frac{1}{f^{DB}{}_i}, \quad i = 1, \ldots, |D|_c \tag{1}$$

where $f^1{}_i$ is the frequency of occurrence of the codeword (or *visual word*) i in the image which results from the vector quantization of its local descriptors $\underline{s}_j$; $f^{DB}{}_i$ is the frequency of occurrence of images containing the codeword i within the database, and α is a normalization constant. Let $\underline{x}^1_j$ be the all zero vector with a single unit coefficient at the position given by the selected codeword index and note that the codeword image frequency vector $\underline{f}^1$ of size $|D|_c$ can be written as $\frac{1}{N_I}\sum_{j=1}^{N_I} \underline{x}^1_j$.

The *bag-of-features* approach has been extended in [7] by computing the vector $\underline{f}^k$ as

$$\underline{f}^k = \frac{1}{N_I} \sum_{j=1}^{N_I} \underline{x}^k_j, \tag{2}$$

where $\underline{x}^k$ is a vector of l-0 norm $k > 1$ for which the positions of the k non-zero components correspond to the k nearest codewords (*visual words*) in the codebook. The values of the k non-zero components are given by

$$x^k{}_l = \frac{1}{\lambda} \exp(-\delta_l^2/\sigma^2), \quad l = 1, \ldots, k. \tag{3}$$

Here σ^2 is a fixed parameter tuned according to the used codebook, δ_l is the euclidean distance between $\underline{s}$ (the underlying local descriptor) and the l-th chosen codeword, and λ is an l-1 normalization constant [7]. The associated *bag-of-features* vector $\underline{w}^k$ is built similarly to $\underline{w}^1$ by substituting $\underline{f}^k$ in place of $\underline{f}^1$ in (1):

$$w^k{}_i = \frac{1}{\alpha} f^k{}_i \log \frac{1}{f^{DB}{}_i}. \tag{4}$$

Letting $k = 1$ reduces $\underline{w}^k$ [7] to $\underline{w}^1$ [1]. When irrelevant, we will discard the superscript k and use $\underline{w}$ to denote *bag-of-features* vectors in general.

The *bag-of-features* sparsity-inducing mechanism results in a loss of performance under local querying: the algorithm achieves sparsity by forming a single (global) histogram descriptor from a set of (local) input descriptors. To illustrate this, consider a local query $\underline{w}_q$ defined by a subset of descriptors of an indexed image. We can then decompose an indexed *bag-of-features* $\underline{w}$ in terms of background ($\underline{w}_B$) and query region ($\underline{w}_q$) *bag-of-features*. The number of times the codeword i occurs in the indexed image can be written as $N_I f_i = N_q f^q{}_i + (N_I - N_q) f^B{}_i$, where N_q is the number of descriptors in the query region, the *bag-of-features* $\underline{w}$ is thus given by:

$$w_i = \frac{\alpha_B}{\alpha} \cdot \frac{N_I - N_q}{N_I} \cdot w_{B\,i} + \frac{\alpha_q}{\alpha} \cdot \frac{N_q}{N_I} \cdot w_{q\,i}, \tag{5}$$

where α_B, α_q and α respectively denote the norm of the background $\underline{w}_B$, the query $\underline{w}_q$, and the entire image $\underline{w}$. The resulting score for the matching *bag-of-features* $\underline{w}$, given the local query *bag-of-features* $\underline{w}_q$, is then:

$$\langle \underline{w}, \underline{w}_q \rangle = \frac{\alpha_B (N_I - N_q)}{\alpha N_I} \langle \underline{w}_B, \underline{w}_q \rangle + \frac{\alpha_q}{\alpha} \cdot \frac{N_q}{N_I}. \tag{6}$$

This expression illustrates how the score is more and more impacted by the background *bag-of-features* $\underline{w}_B$ as the query region becomes more local. The score due only to the query region (right-hand term in the sum) will be contaminated by the correlation between query and background descriptors (left-hand side in the sum).

III. New Local Sparse Descriptors

In the previous section, we have shown how the *bag-of-features* system succeeds in reducing a complex query between images each being represented by sets of local descriptors into simple inner product distance calculation between sparse vectors $\underline{w}$. The approach nonetheless suffers from a loss in descriptor locality, as the set of local image descriptors were transformed into a single global descriptor. In this section, we present sparse descriptors that retain the locality of the input descriptors while at the same time enjoying the complexity benefits of sparsity. We propose two different approaches to get sparse representations: the first one is based on k-NN approximations, whereas the second one borrows sparse decomposition algorithm from the image compression and signal processing community. We also discuss the voting system and the complexity induced by having several descriptors per image.

A. *k-NN based descriptor*

The first descriptor is constructed by selecting the k nearest codewords as explained in Section II for the *bag-of-features*, but this time keeping the set of N_I individual sparse vectors instead of a single descriptor $\underline{w}$ per treated image. This approach yields N_I sparse descriptors $\underline{x}^k$ and a voting algorithm can be used to carry out queries. With this representation, codewords define cells in the SIFT description space. As the weighting coefficients of a sparse vector $\underline{x}^k$ are computed as a function of the euclidean distance between the SIFT descriptor and its k nearest codewords, the weighted vector $\underline{x}^k$ can be seen as a local coordinates of the SIFT descriptor relatively to its the k nearest neighbouring cells. It acts as an approximated localization of the SIFT descriptor in the description space. The distance between two sparse descriptors $\underline{x}^k$ thus approximates the distance between underlying SIFT descriptors in the description space.

B. *Visual Sentences*

The second descriptor is constructed by running the basis pursuit algorithm [11] to find a sparse approximation of each input SIFT descriptor, thus obtaining a single sparse vector $\underline{x}^s$ per input descriptor $\underline{s}$. Similarly, the approach yields N_I sparse descriptors $\underline{x}^s$.

Let $\mathbf{D}$ be the matrix that contains the codewords $\underline{D}_k$ along its columns. The visual word $\underline{x}^1$ used when building the *bag-of-features* descriptor yields a coarse representation $\hat{\underline{s}}$ of the underlying SIFT descriptor $\underline{s}$ by specifying the one codeword $\underline{D}_k$ most similar to $\underline{s}$. This can be formalized as:

$$\hat{\underline{s}} = \mathbf{D}\underline{x}^1. \tag{7}$$

where $\underline{x}^1$ has a single, unit-valued, non-zero coefficient. This suggests using a better reconstruction $\hat{\underline{s}}_s$ of $\underline{s}$ by using a linear combination of codewords

$$\hat{\underline{s}}_s = \mathbf{D}\underline{x}^s, \tag{8}$$

where $\underline{x}^s$ is a real-valued sparse vector with multiple non-zero coefficients. The vector $\underline{x}^s$ is the proposed descriptor; we name it a *visual sentence*, an extension of the name *visual words* given to the vectors $\underline{x}^1$ in [1], as it defines a linear combination of visual words $\underline{D}_k$.

The problem of selecting the best coefficients and codewords (non-zero positions) for $\underline{x}^s$ is an open question. Motivated by (8), in this work we optimize the reconstruction error $|\mathbf{D}\underline{x}^s - \underline{s}|^2$ and sparsity of $\underline{x}^s$. Both characteristics are of importance: the reconstruction error is directly related to the performance of the descriptor while sparsity is related to complexity, as we will show later, analytically and experimentally. This optimization problem has been addressed extensively in the image processing literature [8], [9], [10], [11].

In our work, we use the basis pursuit algorithm [11] as it provides an optimal tradeoff between sparsity and reconstruction error. The algorithm formulates the selection of $\underline{x}^s$ as the following constrained optimization problem:

$$\underline{x}^s = \underset{\underline{x}^s}{\operatorname{argmin}} |\underline{s} - \mathbf{D}\underline{x}^s|^2 + h|\underline{x}^s|_1, \tag{9}$$

where $|\cdot|_1$ is the l-1 norm denoting the sum of coefficients of $\underline{x}^s$. The parameter h controls the tradeoff between sparsity and reconstruction error: the higher h is, the more sparse $\underline{x}^s$ is. It would be more natural to use the l-0 norm corresponding to the number of non-zero elements of $\underline{x}^s$. The l-1 norm is used nonetheless as it reduces the problem to a linear program that can be solved using standard mathematical routines.

The meaning of this representation is different from the k-NN based descriptor. Here, the sparse vector $\underline{x}^s$ represents an approximated reconstruction of the SIFT descriptor. Thus, the inner product between sparse vectors can be seen as a rough approximation of the descriptors correlation.

C. *Voting system and data structure*

The query score using *bag-of-features* is computed as the distance calculation (normalized inner product) between $\underline{w}$ vectors. Given that the sparse decomposition provide N_I sparse and local descriptors $\underline{x}^s$ (or equally $\underline{x}^k$), this ranking system no longer applies. It is replaced by the following voting system: for each of the N_q local descriptors $\underline{x}^s$ of a query image, the K closest descriptors among the whole database are retained. Each image having at least one descriptor among these K-nn will vote once. In the end, at most of $K \times N_q$

votes are distributed amongst all database images, and the cumulative votes for each image yields its score.

The sparsity of the descriptors is exploited for inner-product calculation. Indeed, finding the K nearest neighbors of a query descriptor $\underline{x}^s$ can be efficiently implemented using an inverted file index which is slightly different from the one used in the context of *bag-of-features*. The row entry j of the row-major matrix stores in a contiguous memory bin the j-th non-zero coefficient and the corresponding descriptor identifier of all the indexed $\underline{x}^s$. Only memory bins having the same index as the positions of the non-zero coefficients of the query descriptor need to be retrieved from memory (or disk) and processed.

D. Complexity Analysis

We now present an image search complexity analysis comparing sparse descriptors and *bag-of-features*. As a complexity measure we will use an estimate of the mean number of coefficients to retrieve given a set of query descriptors. This number further equals the mean number of multiplications to carry out when calculating all inner products between the query set and the indexed descriptors. If the query set consists of a single global query descriptor $\underline{w}$, this complexity measure is given by the cumulative sum of all the bins having en index corresponding to the k non-zero positions of the query descriptor. For the case of sparse descriptors, we will further sum bin sizes over all descriptors in the query set.

We first derive a complexity expression for sparse descriptors. Since multiple sparse descriptors are available per query, the complexity for the entire query will be the sum of per-descriptor complexity over all descriptors present in the query. Let I denote the number of images in the database, $\bar{N}_I$ the mean number of local descriptors per-image and $\bar{n}_s$ the mean number of non-zero coefficients in the sparse descriptors $\underline{x}^s$. An estimate of the mean bin size B_s can be defined as the total number of non-zero coefficients in the database divided by the total number of bins:

$$B_s = \bar{n}_s \bar{N}_I \cdot I / |\mathbf{D}|_c, \tag{10}$$

where $|\mathbf{D}|_c$ is the number of codewords available. For a single visual sentence, the number of multiplications to carry out is equal to the sum of bin sizes of all bins activated by the query. Thus, for a query comprising an average of $\bar{N}_q$ query descriptors, an estimate of the mean number of multiplications required can be taken as the total of non-zero query coefficients times the estimated mean bin size in (10):

$$\begin{aligned} M_s &= \bar{n}_s \bar{N}_q \cdot B_s \\ &= \bar{n}_s \bar{N}_q \cdot (\bar{n}_s \bar{N}_I \cdot I / |\mathbf{D}|_c) \\ &= \bar{n}_s^2 \cdot \bar{N}_q \bar{N}_I \cdot (I / |\mathbf{D}|_c). \end{aligned} \tag{11}$$

Considering next the case of *bag-of-features* scoring, the number of non-zero coefficients of a *bag-of-features* $\underline{w}$ is generally not equal to $\bar{N}_I$. Due to possible overlap in non-zero coefficient positions amongst the $\underline{x}^k$ in (3): each of the $\bar{N}_I$ descriptors in the image will *at most* contribute k new non-zero coefficients to the *bag-of-features* $\underline{w}$. We model the overlap in coefficients position with an overlap coefficient $\nu \in [1/\bar{N}_I,\ 1]$, thus writing the l-0 norm of a *bag-of-features* (built using all the N_I descriptors in the entire image) as $\nu \bar{n}_w \bar{N}_I$, where $\bar{n}_w$ denote the mean number of non-zero coefficients in $\underline{x}^k$. The mean bin size B_w (when indexing *bag-of-features* vectors $\underline{w}$) can thus be estimated as:

$$B_w = \nu \bar{n}_w \bar{N}_I \cdot I / |\mathbf{D}|_c. \tag{12}$$

Likewise, the number of bins activated by the query *bag-of-features* can be written as $\nu \bar{n}_w \bar{N}_q$, with $\nu \in [1/\bar{N}_q,\ 1]$. The complexity for *bag-of-features* scoring can then be written as:

$$\begin{aligned} M_w &= \nu \bar{n}_w \bar{N}_q \cdot B_w \\ &= \nu \bar{n}_w \bar{N}_q \cdot (\nu \bar{n}_w \bar{N}_I \cdot I / |\mathbf{D}|_c) \\ &= \nu^2 \bar{n}_w^2 \cdot \bar{N}_q \bar{N}_I \cdot (I / |\mathbf{D}|_c). \end{aligned} \tag{13}$$

Comparing the resulting *bag-of-features* complexity above to that of sparse descriptors in (11) indicates that *bag-of-features* is more computationally efficient by a factor of $\frac{\bar{n}_s^2}{\nu^2 \bar{n}_w^2}$. The *bag-of-features* descriptors gain in performance for smaller l-0 norm [7], and thus this factor will tend to favor *bag-of-features*. We will see in the results section that for small l-0 norm the *bag-of-features* outperfom the local sparse descriptors in terms of complexity. However, a discerning descriptor under local queries is able to return better results for lower numbers of query descriptors (i.e., for lower values of $\bar{N}_q$). In the results section we evaluate the performance of different query systems as a function of $\bar{N}_q$ and verify that the sparse descriptors $\underline{x}^s$ yield better performance for smaller values of $\bar{N}_q$.

IV. Results

In this section, we compare experimentally the *bag-of-features* image description and retrieval system [1] to the proposed local sparse descriptors using the voting system described in section III-C. The experiments have been made with the holidays image database [14] which consists of 1491 images organized into 500 groups of images of varying size. For all methods, to compare with *bag-of-features* process, the initial SIFT descriptors are computed on MSER regions. In total, more than 3 millions local SIFT descriptors have been extracted (two thousand per image on average). For each image, the *bag-of-features*, the exponentially-weighted vectors $\underline{x}^k$ (3) and the visual sentences $\underline{x}^s$ have been computed.

The performance of the system has been evaluated as a function both of complexity and of the locality of the search. Each image has one or more relevant images in the database, and each image out of the 1491 images has been used as a query image. The locality of the search is controlled by varying the number N_q of query descriptors used, where the descriptors used are always taken to be those related to regions of the image closest to the image center. This choice of query region allows us to easily control the locality of the search, hence to evaluate its influence on performance. As the same query region has been used both for the reference and the proposed systems, this choice does not favor one or the other.

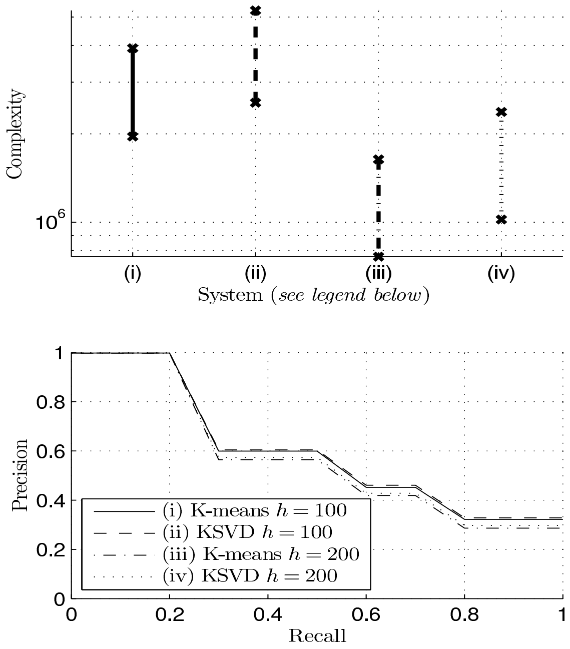

Fig. 1. Comparison of performance as a function of the dictionary training method. All systems use the visual sentences descriptor $\underline{x}^s$, where h is given in (9).

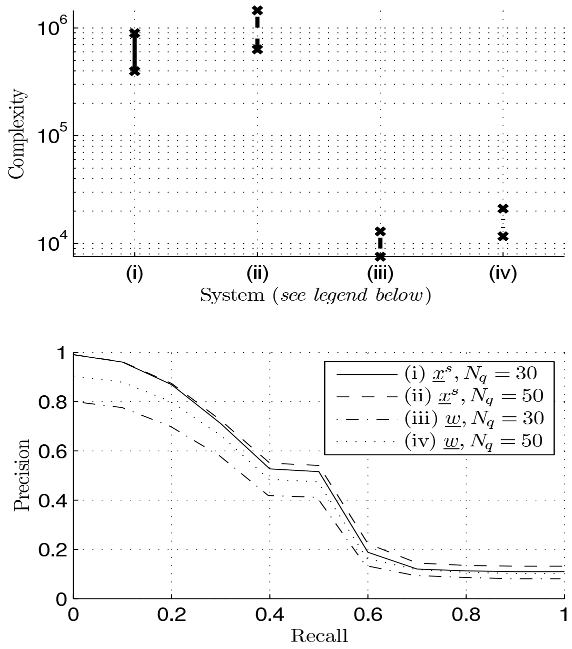

Fig. 2. Comparison of visual sentences vs. *bag-of-features* in terms of complexity and recall vs. precision when using $N_q = 30$ and $N_q = 50$. A value of $h = 250$ (cf. (9)) is used to build the visual sentences $\underline{x}^s$.

As a measure of performance we use average precision (over all queries) calculated at specified recall values (we use $R = 0, 0.1, 0.2, \ldots, 1$) [15]. The complexity is computed as the total number of multiplications required for the image query. Section III-D gives an estimate of the complexity expressed according to the mean number of non-zero coefficients of a query and the mean bin size of the index structure. However, in practice, the codewords are not uniformly distributed over the image database, leading to different bin sizes. Thus, the complexity varies from one query to another, depending on its sparsity and the size of bins needed to be retrieved. As a complexity measure, we thus present both the 10% and 90% complexity quartiles, computed over all queries.

The same codebook $\mathbf{D}$ has been used for the all the setups considered; it consists of $10,000$ codewords obtained by applying K-means on SIFT/MSER descriptors obtained from the Stewénius-Nistér image database [16]. Other training algorithms better suited to sparse approximation algorithms exist, notably the K-SVD algorithm [17]. These algorithms are nonetheless very costly and the available implementations yield prohibitive complexity for the dataset and codebook sizes concerned by description and indexing application. To gauge the potential impact of the codebook training method, we provide results for our proposed description/voting system when training a small codebook of $1,000$ atoms using both the K-SVD algorithm and the K-means algorithm. It can be noticed that the sparsity decreases along with the codebook size, resulting in a complexity increase. The results are plotted in Figure 1. For comparable complexity levels, the figure shows that the performance is not greatly affected by the training algorithm.

Figure 2 compares visual sentences to the *bag-of-features* when using N_q values of 30 and 50. The top graph represents the complexity quartiles as explained before, for each of the four experiments, while the bottom graph shows the search results quality in terms of recall and precision. While the performance of visual sentences remains stable when reducing the number N_q of query regions, the performance of *bag-of-features* degrades rapidly with reduced N_q. As expected, the improved performance for local queries comes at the price of increased complexity (top of Fig. 2), as the visual sentences descriptors require a greater l-0 norm and thus (i) a greater number of coefficients needs to be indexed and (ii) a greater number of index rows is activated by each query.

We also comparatively assess the two local sparse descriptors, the one constructed using a sparse approximation of the initial SIFT descriptor obtained with the basis pursuit algorithm, and the one constructed by considering the k nearest codewords in the codebook, with weighting coefficients given by (3). In this last method, the sparsity of the resulting vector $\underline{x}^k$ is clearly determined by the number k of nearest codewords chosen. When using the basis pursuit algorithm, the sparsity of vector $\underline{x}^s$ is controlled by the h parameter present in the optimization cost function (9). In order to compare these two approaches in terms of complexity, we set the parameters of the basis pursuit algorithm in order to obtain similar sparsity of $\underline{x}^s$ and $\underline{x}^k$. Figure 3 shows that the two methods perform comparably with similar complexity.

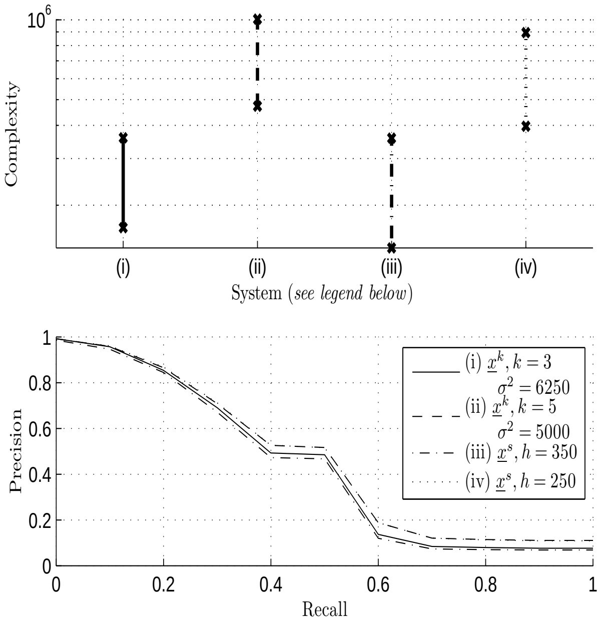

Fig. 3. Local descriptor voting using $\underline{x}^k$ with $k = 3$ and $k = 5$ and $\underline{x}^s$ with $k = 2.8$ and $k = 5.1$ (mean). The parameter h corresponds to (9) and σ and k correspond to (3).

V. Conclusion

In this paper, we have described two methods for efficient SIFT-based image description and searches in large databases in the context of local querying. The methods rely on a sparse representation of SIFT descriptors computed on MSER regions. A retrieval and voting system based on an inverted file has been developed and adapted to the two descriptors. These approaches benefit from a complexity decrease which result from the descriptor vector sparsity, while preserving the locality properties of the indexed descriptors. The analysis has indeed shown that local queries using these descriptors perform better than with the *bag-of-features* image descriptor and retrieval system, where a single histogram is constructed from multiple local descriptors and used as a single descriptor for the query region. The corresponding complexity is also increased, but the methods, by tuning the l_0-norm of the sparse descriptors, allow an easy control of the trade-off between complexity versus precision of the local queries. Note that the inner product distance may not be the most appropriate distance measure between sparse vectors, due to instability in the codeword selection process under slight variations of original vectors. Further work will be dedicated to study a more appropriate distance measure. Another possible improvement will be to exploit the property of codeword selection process using another sparse decomposition algorithm that gives different importance to the codewords selected.

References

[1] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” in *Proceedings of the International Conference on Computer Vision*, vol. 2, Oct. 2003, pp. 1470–1477. [Online]. Available: http://www.robots.ox.ac.uk/ vgg

[2] Z. Jia, L. Amsaleg, and P. Gros, “Content-based image retrieval from a large image database,” *Pattern Recogn.*, vol. 41, no. 5, pp. 1479–1495, 2008.

[3] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[4] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] J. Zobel and A. Moffat, “Inverted files for text search engines,” *ACM Comput. Surv.*, vol. 38, no. 2, p. 6, 2006.

[6] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, “A comparison of affine region detectors,” *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[7] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[8] S. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397–3415, Dec 1993.

[9] O. G. Sezer, O. Harmanci, and O. G. Guleryuz, “Sparse orthonormal transforms for image compression,” in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, 2008, pp. 149–152. [Online]. Available: http://dx.doi.org/10.1109/ICIP.2008.4711713

[10] P. Jost, P. Vandergheynst, and P. Frossard, “Tree-based pursuit: Algorithm and properties,” *Signal Processing, IEEE Transactions on*, vol. 54, no. 12, pp. 4685–4697, Dec. 2006.

[11] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[12] B. Ruf, E. Kokiopoulou, and M. Detyniecki, “Mobile museum guide based on fast SIFT recognition,” in *6th International Workshop on Adaptive Multimedia Retrieval*, ser. Lecture Notes in Computer Science. Springer, 2008.

[13] E. Kokiopoulou and P. Frossard, “Minimum distance between pattern transformation manifolds: Algorithm and Applications,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[14] INRIA, “Inria holidays dataset.” [Online]. Available: http://lear.inrialpes.fr/ jegou/data.php

[15] M. Boughanem and J. Savoy, *Recherche d’information état des lieux et perspectives*, H. Lavoisier, Ed. http://www.editions-hermes.fr/: Hermès Science Publications, 2008.

[16] H. Stewnius and D. Nistr, “Stewnius-Nistr dataset.” [Online]. Available: http://vis.uky.edu/ stewe/ukbench/

[17] M. E. Michal Aharon and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, pp. 4311–4322, 2006.