Three things everyone should know to improve object retrieval

Relja Arandjelović Andrew Zisserman
Department of Engineering Science, University of Oxford
{relja,az}@robots.ox.ac.uk

Abstract

The objective of this work is object retrieval in large scale image datasets, where the object is specified by an image query and retrieval should be immediate at run time in the manner of Video Google [28].

We make the following three contributions: (i) a new method to compare SIFT descriptors (RootSIFT) which yields superior performance without increasing processing or storage requirements; (ii) a novel method for query expansion where a richer model for the query is learnt discriminatively *in a form suited to immediate retrieval through efficient use of the inverted index; (iii) an improvement of the image augmentation method proposed by Turcot and Lowe [29], where only the augmenting features which are spatially consistent with the augmented image are kept.*

We evaluate these three methods over a number of standard benchmark datasets (Oxford Buildings 5k and 105k, and Paris 6k) and demonstrate substantial improvements in retrieval performance whilst maintaining immediate retrieval speeds. Combining these complementary methods achieves a new state-of-the-art performance on these datasets.

1. Introduction

We consider the problem of large scale particular object retrieval where the goal is to retrieve all images containing a specific object in a large scale image dataset, given a query image of that object. This goal is required to be performed in near-real time so that users can interactively browse the dataset or search using images from their mobile phones, for example, as in Google Goggles.

Many works have addressed this problem, starting with setting up the standard framework [18, 23, 28] where an image is represented using a bag-of-visual-words (BoW), and images are ranked using term frequency inverse document frequency (tf-idf) computed efficiently via an inverted index.

However, an object in a target image can fail to be retrieved for a number of reasons using this standard pipeline, these include: feature detection drop-out; noisy descriptors; inappropriate metrics for descriptor comparison; or loss due to descriptor quantization. All of these failings have received attention over the past few years. Quantization introduces two problems: information about the original descriptor is lost, and corresponding descriptors may be assigned to different visual words. To address the first of these problems, Jégou *et al*. [8, 10] have moved towards keeping a more accurate approximation of the original descriptor by Hamming embedding or product quantization. To address the second problem, methods include soft-assignment [7, 10, 24, 30] and explicitly learning quantization variability [15, 17]. Others have learnt better visual descriptors (than SIFT) [32] or better metrics for descriptor comparison and quantization [25]. Overcoming these quantization problems improves recall.

The query expansion methods of [5, 6], where BoW vectors from spatially verified regions are used to issue new queries, address the problem of feature detection drop out in addition to quantization and noise on the descriptor, and again improves recall. The principal limitation of query expansion is that it relies on the query to yield a sufficient number of high precision results in the first place. Database-side feature augmentation [29] is a natural complement to query expansion where images in the database are augmented offline with all features of images containing the same view of the object. Again this improves recall. The precision (but not the recall) of the pipeline is improved by using spatial consistency to rerank a tf-idf short list [23, 28] or to rank all images with common visual words [8], or other (re)ranking variations [9, 27].

Against this background of considerable progress, we make the following three novel contributions:

1. RootSIFT: We show that using a square root (Hellinger) kernel instead of the standard Euclidean distance to measure the similarity between SIFT descriptors leads to a dramatic performance boost in all stages of the pipeline. This change is simple to implement in just a few lines of code, and it does not require any additional storage space as the conversion from SIFT to RootSIFT can be done online.

978-1-4673-1228-8/12/$31.00 ©2012 IEEE

2. Discriminative query expansion: Current methods for query expansion combine the BoW vectors of the spatially verified results, *e.g*. by averaging. We show that using a linear SVM to discriminatively learn a weight vector for re-querying yields a significant improvement over the standard average query expansion method [6], whilst maintaining immediate retrieval speeds through efficient use of the inverted index.

3. Database-side feature augmentation: The database-side feature augmentation of [29] is very powerful but suffers from not taking into account the spatial configuration of augmenting features. We show that if the *visibility* of the augmenting features is taken into account (using spatial verification by a homography) then this simple extension provides a significant improvement in performance compared to the original method.

In each case these methods can substantially boost the retrieval performance, and can simply be "plugged into" the standard object retrieval architecture of Philbin *et al*. [23] (BoW, inverted index, tf-idf, spatial consistency re-ranking) without increasing processing time. Indeed, RootSIFT and discriminative query expansion do not even increase the storage requirements.

In sections 3–5 we describe each of these methods in detail and demonstrate their performance gain using the method of [23] as a baseline on the Oxford Buildings 5k and 105k image dataset benchmarks as a running example. The methods are combined and compared to the state of the art in section 6. We conclude with giving recommendations for the design of object retrieval systems based on their performance, computational efficiency, storage requirements and ease of implementation of the various methods.

2. Evaluation, datasets and baselines

The performance of particular object retrieval systems is evaluated using standard and publicly available image datasets containing hundreds of thousands of images each. Several representative queries are defined and ground truth is manually compiled. The single score for the recognition performance is the mean average precision (mAP), where the mean is taken over all queries.

Oxford Buildings [23] Contains 5062 high-resolution images automatically downloaded from Flickr [2]. It defines 55 queries (consisting of an image and query region of interest) used for evaluation (5 for each of the 11 chosen Oxford landmarks) and it is quite challenging due to substantial variations in scale, viewpoint and lighting conditions. The basic dataset, often referred to as *Oxford 5k*, is usually appended with another 100k Flickr images to test large scale retrieval, thus forming *Oxford 105k* dataset.

Paris Buildings [24] Analogously to Oxford 5k, 6392 images were obtained from Flickr [2] and 55 queries are used for evaluation. As it contains images of Paris it is considered to be an independent dataset from Oxford 5k and thus commonly used to test effects of training a visual vocabulary on it while evaluating performance on Oxford 5k.

2.1. Baseline retrieval system

We follow the standard BoW retrieval framework described in [23]. We use affine-Hessian interest points [16], a vocabulary of 1M vision words obtained using approximate k-means, and spatial re-ranking of the top 200 tf-idf results using an affine transformation. Our most recent implementation of the system achieves a mAP of 0.672 on the Oxford 5k dataset compared to the original 0.657 of [23]. This is the baseline system that we will compare to as we introduce new methods in the sequel.

Our most recent implementation of the average query expansion method from [6] (described in detail in section 4) achieves a mAP of 0.726 on Oxford 105k compared to the original 0.711 [6]. Note, although the original paper described several methods for query expansion (*e.g*. transitive closure, multiple image resolution), the average method has emerged as the standard to compare to [5, 17, 24] (it has similar performance to the others, and is faster at run time as the other methods involve issuing several new queries). Hence, we use it as our baseline for query expansion in the subsequent comparisons.

For consistency reasons (using the same visual vocabulary, and various parameters of spatial reranking and query expansion) we compare our improvements to our most recent implementation of the baseline systems.

3. RootSIFT: Hellinger distance for SIFT

It is well known for areas such as texture classification and image categorization, that using Euclidean distance to compare histograms often yields inferior performance compared to using measures such as χ^2 or Hellinger. SIFT was originally designed to be used with Euclidean distance [14], but since it is a histogram the question naturally arises as to whether it would also benefit from using alternative histogram distance measures. We show that using the Hellinger kernel does indeed bring a great benefit.

In the following it will be helpful to make use of the standard connection between distances (metrics) and kernels. Suppose $\mathbf{x}$ and $\mathbf{y}$ are n-vectors with unit Euclidean norm ($\|\mathbf{x}\|_2 = 1$), then the Euclidean distance $d_{\mathrm{E}}(\mathbf{x},\mathbf{y})$ between them is related to their similarity (kernel) $S_{\mathrm{E}}(\mathbf{x},\mathbf{y})$ as

$$d_{\mathrm{E}}(\mathbf{x},\mathbf{y})^2 = \|\mathbf{x}-\mathbf{y}\|_2^2 = \|\mathbf{x}\|_2^2+\|\mathbf{y}\|_2^2-2\mathbf{x}^\mathsf{T}\mathbf{y} = 2-2S_{\mathrm{E}}(\mathbf{x},\mathbf{y})$$

where $S_{\mathrm{E}}(\mathbf{x},\mathbf{y}) = \mathbf{x}^\mathsf{T}\mathbf{y}$, and the last step follow from $\|\mathbf{x}\|_2^2 = \|\mathbf{y}\|_2^2 = 1$. We are interested here in replacing the Euclidean similarity/kernel by the Hellinger kernel.

The Hellinger kernel, also known as the Bhattacharyya's coefficient, for two $L1$ normalized histograms, $\mathbf{x}$ and $\mathbf{y}$ (*i.e.* $\sum_i^n x_i = 1$ and $x_i \geq 0$), is defined as:

$$H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \sqrt{x_i y_i}$$

SIFT vectors can be compared by a Hellinger kernel using a simple algebraic manipulation in two steps: (i) $L1$ normalize the SIFT vector (originally it has unit $L2$ norm); (ii) square root each element. It then follows that $S_{\mathrm{E}}(\sqrt{\mathbf{x}}, \sqrt{\mathbf{y}}) = \sqrt{\mathbf{x}}^{\mathsf{T}} \sqrt{\mathbf{y}} = H(\mathbf{x}, \mathbf{y})$, and the resulting vectors are $L2$ normalized since $S_{\mathrm{E}}(\sqrt{\mathbf{x}}, \sqrt{\mathbf{x}}) = \sum_i^n x_i = 1$. We thus define a new descriptor, which we term *RootSIFT*, which is an element wise square root of the $L1$ normalized SIFT vectors. The key point is that comparing RootSIFT descriptors using Euclidean distance is equivalent to using the Hellinger kernel to compare the original SIFT vectors: $d_{\mathrm{E}}(\sqrt{\mathbf{x}}, \sqrt{\mathbf{y}})^2 = 2 - 2H(\mathbf{x}, \mathbf{y})$.

RootSIFT is used in the specific object retrieval pipeline by simply replacing SIFT by RootSIFT at every point. The fact that RootSIFT descriptors are compared using Euclidean distance means that every step can be effortlessly modified: k-means can still be used to build the visual vocabulary (since it is based on Euclidean distance), approximate nearest neighbor methods (essential for systems with very large vocabularies) can still be used; as can soft assignment of descriptors to visual words [10, 24], query expansion, and other extensions which only require Euclidean distance on SIFT [8, 11, 17, 25].

The dramatic improvement in performance is shown in table 1, where for each step (*e.g.* adding query expansion, adding feature augmentation) using SIFT is compared with using RootSIFT. For example, on Oxford 105k the baseline system (tf-idf only) increases in performance from 0.515 to 0.581, and with spatial reranking included the improvement is from 0.581 to 0.642. These improvements come at virtually no additional cost, and no additional storage since SIFT can be converted online to RootSIFT with a negligible processing overhead.

Discussion. The RootSIFT transformation can be thought of as an explicit feature map from the original SIFT space to the RootSIFT space, such that performing the scalar product (*i.e.* a linear kernel) in RootSIFT space is equivalent to computing the Hellinger kernel in the original space. This approach has been explored in the context of kernel maps for SVM classifiers by [22, 31]. Explicit feature maps can be built for other additive kernels, such as χ^2, but we find little difference in performance from that of the Hellinger kernel when used in the specific object retrieval system.

The effect of the RootSIFT mapping is to reduce the larger bin values relative to the smaller bin values. The Euclidean distance between the original SIFT vectors can be dominated by these large values. After the mapping the distance is more sensitive to the smaller bin values. The importance of this "variance stabilizing transformation" has previously been noted by Winn *et al.* [33] for texton histograms.

Previous work has compared SIFT vectors with distances other than Euclidean, but an explicit feature map was not employed and so the benefits of simply being able to continue to use algorithms with Euclidean distance (*e.g.* k-means) were not apparent. For example, Johnson [12] uses Jeffrey's divergence to compare SIFT vectors concentrating on descriptor compression, Pele and Werman use a variant of the Earth Mover's Distance [19] or a quadratic χ^2 metric [20].

4. Discriminative query expansion

Query expansion can substantially improve the performance of retrieval systems. The average query expansion method proceeds as follows: given a query region, images are ranked using tf-idf scores and spatial verification is performed on a short list of high ranked results, also providing the location (ROI) of the queried object in the retrieved images. BoW vectors corresponding to words in these ROIs are averaged together with the query BoW, and this resulting query expanded BoW vector is used to re-query the database.

In contrast we introduce here a *discriminative* approach to query expansion where negative data is taken into account and a classifier trained. It proceeds as follows: BoW vectors used to enrich the query are obtained in exactly the same way as for average query expansion. These provide the positive training data, and images with low tf-idf scores provide the negative training data. A linear SVM is trained using these positive and negative BoW vectors to obtain a weight vector $\mathbf{w}$. The learnt weight vector is used to rank images by their distance from the decision boundary, *i.e.* if the image is represented by the BoW vector $\mathbf{x}$, then images are sorted on the value $\mathbf{w}^{\mathsf{T}}\mathbf{x}$. Ranking images using the learnt weight vector $\mathbf{w}$ can be carried out efficiently using the inverted index in much the same way as when computing tf-idf scores – both operations are just scalar products between a vector and $\mathbf{x}$. For the tf-idf scoring in average query expansion the vector used is the average query idf-weighted BoW vector, whilst for discriminative query expansion (DQE) it is the learnt weight vector $\mathbf{w}$.

Note, for DQE to be efficient it is essential that the weight vector is sparse. As discussed below, by a careful choice of negative data the obtained weight vector is at least as sparse as the one used in average query expansion. Thus, the method is at least as computationally efficient as average query expansion with an insignificant overhead of training a linear SVM. Figure 1 illustrates schematically how negative data can benefit DQE over average query expansion.

Table 1 compares the DQE method to our implemen-

Retrieval Method	SIFT		RootSIFT	
	Ox5k	Ox105k	Ox5k	Ox105k
Philbin *et al*. [23]: tf-idf ranking	0.636	0.515	0.683	0.581
Philbin *et al*. [23]: tf-idf with spatial reranking	0.672	0.581	0.720	0.642
Chum *et al*. [6]: average query expansion (AQE)	0.839	0.726	0.850	0.756
Turcot and Lowe [29]: database-side feature augmentation (AUG)	0.776	0.711	0.827	0.759
This paper: discriminative query expansion (DQE)	**0.847**	0.752	0.861	0.781
This paper: spatial database-side feature augmentation (SPAUG)	0.785	0.723	0.838	0.767
This paper: SPAUG + DQE	0.844	**0.795**	**0.881**	**0.823**

Table 1. **Retrieval performance (mAP) of various proposed methods.** We use our implementation of all listed methods [6, 23, 29] in order to compare them consistently using the same visual vocabularies and sets of parameters. RootSIFT significantly outperforms SIFT for all investigated methods. The vocabularies are generated using the Oxford 5k descriptors and all methods apart from "tf-idf ranking" employ spatial reranking of the top 200 results. Note that for AUG and SPAUG we recompute the idf as described in section 5.

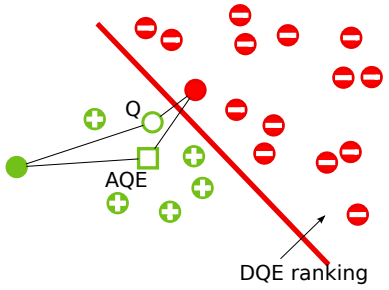


Figure 1. **Discriminative query expansion (DQE).** Illustration of the BoW feature space. True positives are shown as green circles and true negatives as red circles. Spatially verified images used to expand the query (*i.e*. "known" positives) are shown with pluses inside, low ranking images (*i.e*. "known" negatives) with minuses. Q and AQE denote the query and the average query expansion BoW vectors, respectively. A tf-idf AQE ranking sorts images based on their distance to the AQE vector, while DQE ranking sorts images by their signed distance from the decision boundary. As illustrated here, DQE correctly ranks the two images with unknown labels while AQE does not.

tation of the average query expansion (AQE) of Chum *et al*. [6]. It can be seen that DQE is consistently superior to AQE. The performance gain is particularly evident with increasing dataset size – for Oxford 5k DQE outperforms AQE by 1% and 1.3% for SIFT and RootSIFT respectively, while for Oxford 105k mAP improves by 3.6% and 3.3%.

Implementation details. The images used for negative data are the 200 with smallest non-zero tf-idf score for the query. These images are very unlikely to contain any positive instances. However, to avoid the weight vector becoming dense, the BoW vector corresponding to each image is first truncated to only include words that appear in at least one positive example. This is done to prevent many irrelevant negative words being brought in by the large number of negative images, which would then make the 1 million dimensional (*i.e*. the size of the vocabulary) weight vector dense, rendering re-querying inefficient. Other truncation or sparsity methods could be employed here, but we have found this simple procedure quite adequate. With this procedure the weight vector is at least as sparse as the one used in average query expansion as the set of considered words is identical. Note, all vectors are idf-weighted and L2-normalized before training.

The classifier is a linear SVM trained with LIBSVM [3]. The choice of linear (rather than a non-linear kernel) is for efficiency in training, and to have a linear weight vector for use with the inverted file. In training there is only one parameter to be optimized, namely the C weighting of the SVM cost function. The retrieval performance is very insensitive to this value – for RootSIFT and the Oxford 5k benchmark varying C between 0.001 and 1000 changed the mAP of 0.8608 by at most 0.0015. We thus choose $C = 1$. The entire overhead of using DQE instead of AQE (gathering negative training data and training the linear SVM) is 30 ms on average on a 3 GHz single core machine.

Discussion. It is interesting to note that, unlike any object retrieval method proposed to date, our method can actually benefit from adding more distractor images to the dataset to make it "more confusing". DQE could learn a better weight vector if the new images are picked as negative examples, while for other methods the performance would be expected to remain the same at best. As already noted, results in table 1 indeed show that the relative improvement of DQE compared to the average query expansion increases with the number of distractors in the dataset.

Recent work [5, 13] has proposed methods for identifying words that are confusing for a given query image, and thus should not be used. In [5] these confusing words are then removed when re-querying, though the actual "expansion" is still performed by simply averaging the BoW vectors and applying the tf-idf ranking scheme. The DQE goes further in two respects: first it learns the weighting for the positive words (rather than simply averaging), and second it has negative weights for confusing words (rather than simply ignoring them).

5. Database-side feature augmentation

Turcot and Lowe [29] use a matching graph to improve retrieval performance by augmenting the BoW for each im-

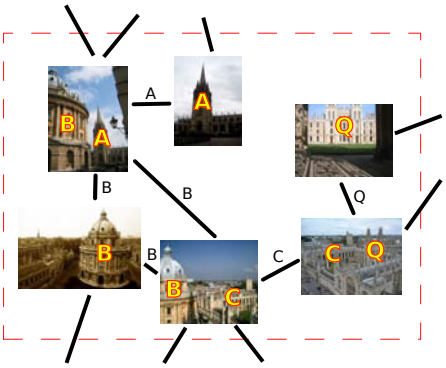


Figure 2. **Image graph.** Part of an image graph, nodes represent images while edges signify that images contain a common object; objects are labelled A, B, C and Q.

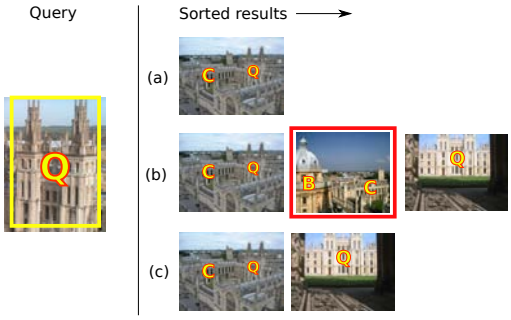


Figure 3. **Database-side for feature augmentation.** Retrieval performance of augmentation methods: the queried object is highlighted in yellow on the leftmost image. (a) tf-idf retrieval results when not using image graph information, a challenging image is not retrieved; (b) retrieved images using the method of Turcot and Lowe [29] and graph shown in figure 2: recall is improved but precision decreases as the false positive (highlighted in red) is augmented with all the visual words of its highly ranked neighboring image; (c) our method shows increased recall while maintaining high precision since images are only augmented with visual words from relevant neighboring regions.

age with the visual word counts of all neighboring images in the graph. In a matching graph [26] nodes represent images, while edges signify that images contain an object in common (figure 2). This approach is somewhat complementary to query expansion as it tries to overcome the same problems as query expansion but on the dataset side, namely feature detection drop outs, occlusion, noisy description and quantization errors.

The procedure of [29] could be thought of as query expanding each image of the dataset, and replacing the original BoW vector of the image, by its average query expanded version. However, rather than only augmenting the BoW vector with spatially verified visual words, [29] augments the BoW vector with *all* visual words from neighboring images. This can be dangerous because a significant number of augmenting visual words may not be visible in the original image (because they are outside the image). The problem is very common as unless two images are nearly identical, a large number of augmenting words will indeed not be visible (figure 2). Instead, we augment using only words estimated to be visible in the augmented image. This is simple to do, as an estimated homography between the two images is readily available from the spatial verification stage of the matching graph construction. The benefit of spatial database-side feature augmentation (SPAUG) is illustrated in figure 3.

Table 1 shows that SPAUG always significantly improves over the baseline. For example, on Oxford 105k with RootSIFT, SPAUG achieves a mAP of 0.767 whereas the baseline gets 0.642. It also consistently outperforms the original method [29], which does not perform the visibility check by 1.0 or 1.7% (for Oxford 5k or 105k respectively). These improvements are significant at these high values of mAP. It is not shown in the table, but SPAUG gives an improvement (of 3.6 or 4.5% respectively) over the tf-idf baseline even before spatial re-ranking is performed. This is important because the tf-idf ranking has to provide a sufficient number of results in order for spatial re-ranking to be beneficial – if there are almost no results then reranking achieves little.

Discussion. Although there is clearly a retrieval benefit in using the spatial homography, there is a cost in terms of additional storage requirement. The original augmentation method of [29] does not incur this cost as it does not need to explicitly augment BoW vectors in advance of a search. Instead, at run time the scalar product tf-idf score between the query and a dataset image is efficiently computed using the inverted index as usual, and then it is augmented by simply summing scores of neighboring images (neighboring according to the matching graph). This is equivalent to augmenting the vectors before tf-idf scoring due the distributivity of the scalar product. However, this is only possible because all visual words in neighboring images are used for augmentation; our extension requires explicit augmentation as one image can contribute different words to different neighbors according to the spatial overlap. This increases the storage requirement since the inverted index grows, however it is worth it given the improvement in retrieval performance. Note that for a particular BoW vector storage only increases when an augmenting word does not already appear anywhere in the augmented image, as if it does then only the count of that word needs to be incremented which does not impact on the inverted index size.

The spatially verified augmentation adds 4.4 words on average per existing word for the Oxford 105k dataset. This is 28% less than the original augmentation method [29], and illustrates that the original approach indeed introduces a large number of irrelevant and possibly detrimental visual words.

AUG variant	SIFT		RootSIFT	
	Ox5k	Ox105k	Ox5k	Ox105k
Orig. idf	0.712	0.662	0.780	0.709
Recomp. idf	0.734	0.673	0.785	0.720
Orig. idf + SR	0.755	0.708	0.820	0.752
Recomp. idf + SR	0.776	0.711	0.827	0.759

Table 2. **Idf recomputation for database-side feature augmentation.** Retrieval performance of our implementation of the Turcot and Lowe [29] d-base augmentation method (AUG) using just tf-idf or with spatial re-ranking (SR). Recomputing the idf using the augmented dataset always improves performance.

Implementation details. We use the approach of Philbin and Zisserman [26] to construct a matching graph of images in a dataset offline. Each image in the dataset is used as a query in a standard particular object retrieval system of Philbin *et al*. [23] and an edge is constructed to each spatially verified image. An alternative graph construction method which employs hashing [4] can be used for very large scale datasets where querying using each image in turn is impractical. When constructing the graph we do not include the query images used for evaluation of a dataset in order to simulate a real-life scenario where query images are not known at preprocessing time.

Since the augmenting words are considered to be equally important as the original visual words extracted from an image, we additionally recompute the *inverted document frequency* (idf) using the augmented dataset. Both the original augmentation method [29] and our extension benefit from idf recomputation, as shown in Table 2. As can be seen, recomputing idf values based on the augmented dataset provides a gain in all cases with median mAP improvement of 1.2%.

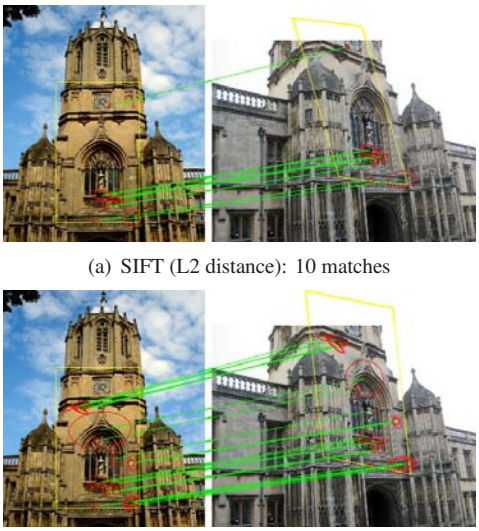
6. Results and discussion

In this section we discuss and compare the three new methods to related work and the state of the art. Comparing absolute performance to previous publications is difficult as the results depend on a number of important implementation details (even though the same benchmark image datasets are used). There factors include: (i) the feature detector used; (ii) the size of the vocabulary; and (iii) whether the vocabulary is learnt on the original dataset or another. For example, the substantial mAP performance improvement in [21] was principally due to using a better implementation of the Hessian-affine feature detector. Hence we take these factors into account in our comparison, using feature detectors provided by the authors or learning the vocabulary on different datasets as appropriate.

The methods are evaluated on the standard Oxford 5k and 105k [23], Paris 6k [26] (section 2) image datasets.

6.1. RootSIFT

We have tested RootSIFT over many retrieval methods, with and without spatial reranking, query expansion, soft assignment and all the methods described in this paper. In every single case RootSIFT significantly outperforms the standard L2 distance for SIFT comparison; due to space limitations we only show a subset of these experiments in table 1. Note that the simple tf-idf scheme with RootSIFT and without spatial reranking outperforms tf-idf using SIFT with spatial reranking. Figure 4 shows examples of matched image patches using SIFT and RootSIFT, demonstrating that RootSIFT yields more complete matches enabling better object localization



(a) SIFT (L2 distance): 10 matches

(b) RootSIFT: 26 matches

Figure 4. **Comparing matches with SIFT and RootSIFT.** Query image and region are shown on the left, a matching result with the estimated corresponding region of interest is shown on the right. RootSIFT yields more matches and the object localization is better.

RootSIFT may be thought of as a non-linear map on SIFT, and it is interesting to compare its performance to the non-linear projection learnt by Philbin *et al*. [25] from SIFT space into a space where L2 distance is more appropriate. This projection was modelled by a deep-belief network, and learnt discriminatively using training data. RootSIFT equals or outperforms this method on the three datasets used in [25]. The mAPs are ([25]/RootSIFT): 0.707/0.720 for Oxford 5k; 0.615/0.642 for Oxford 105k; and 0.689/0.689 for Paris 6k. Since RootSIFT achieves superior results using a simple non-linear root transformation, there is possibly still more room for improvement by combining the root transformation with learning.

6.2. Final retrieval system

We combine all the proposed improvements into one system and evaluate its performance. RootSIFT (section 3) is used to generate the visual vocabulary and hard assign descriptors to visual words, images are augmented with visual words of adjacent images in the image graph, but only with ones which back-project into the image region, and the inverse document frequency (idf) is recomputed (SPAUG, section 5). At query time we extract RootSIFT descriptors

	Method	F	V	SA	Ox5k	Ox105k	Paris6k
a	Philbin *et al*. [24]	M	S		0.801	0.708	N/A
b	Philbin *et al*. [24]	M	S	√	0.825	0.718	N/A
c	Qin *et al*. [27]	M	S		0.814	0.767	0.803
d	This paper	M	S		**0.881**	**0.823**	**0.850**
e	Philbin *et al*. [24]	M	I		0.654	0.562	N/A
f	Philbin *et al*. [24]	M	I	√	**0.719**	**0.605**	N/A
g	This paper	M	I		0.714	0.602	**0.660**
h	Perdoch *et al*. [21]	P	I		0.784	0.728	N/A
i	Perdoch *et al*. [21]	P	I	√	0.822	0.772	N/A
j	Mikulik *et al*. [17]	P	⋆	√	**0.849**	**0.795**	**0.824**
k	Chum *et al*. [5]	P	I		0.827	0.767	0.805
l	This paper	P	I		0.809	0.722	0.765
m	Perdoch *et al*. [21]	P	S		0.901	0.856	N/A
n	Perdoch *et al*. [21]	P	S	√	0.916	0.885	N/A
o	This paper	P	S		**0.929**	**0.891**	**0.910**

Table 3. **Comparison of the combined method (section 6.2) with state-of-the-art.** *F* denotes the feature detector used: *M* for Mikolajczyk & Schmid [16] and *P* for Perdoch *et al*. [1, 21]. *V* signifies which dataset was used to generate the visual vocabulary: *S* for same as test dataset (*e.g*. Oxford 5k for Oxford 5k and Oxford 105k tests), *I* for an independent dataset (*e.g*. Paris 6k for the Oxford 5k and Oxford 105k tests) and ⋆ for the case of [17] where they learn word similarities based on mined SIFT correspondences in a 6M dataset. *SA* marks whether soft assignment was used or not. Spatial reranking is performed on the top 200 or 1000 tf-idf results for consistency with [24] or [5, 17, 21] respectively, which use these parameter values. For (g) the baseline system does not produce a good enough image graph so database-side feature augmentation is not used for this test.

from the interest region, hard assign them to the closest visual word and use the resulting sparse BoW representation to query the database. Fast spatial reranking is performed on the top tf-idf results, and spatially verified results are used to train a linear SVM to learn weights for visual words which represent the query object. The learnt weights are used to efficiently re-query the database and spatial reranking is performed again (DQE, section 4).

Table 3 shows a comparison of this combined method with previous results. The performance of the method is evaluated over three datasets, using two different feature detectors (Mikolajczyk and Schmid [16] or Perdoch *et al*. [1, 21]), and learning the visual vocabulary from two different datasets (from Oxford 5k or Paris 6k).

First, (a–d), we report on using the Mikolajczyk & Schmid [16] feature detector and a vocabulary generated from the test image dataset descriptors. Our combined method sets the new state-of-the-art on all three datasets. The best result for each dataset reported in previous publications is improved on by 6.8%, 7.3% and 5.9% for the Oxford 5k, 105k and Paris 6k datasets respectively. Note, these superior results are achieved without using soft-assignment (as used by (b)), so there are probably still improvements to be gained.

Second, (e–g), the vocabulary is now generated on an independent dataset. As reported previously [24], and evident here, this diminishes performance. Despite this, the combined method exceeds the state of the art (e) without soft-assignment by 9.2% and 7.1% (on Oxford 5k and 105k respectively), and almost equals the state-of-the-art including soft assignment (f).

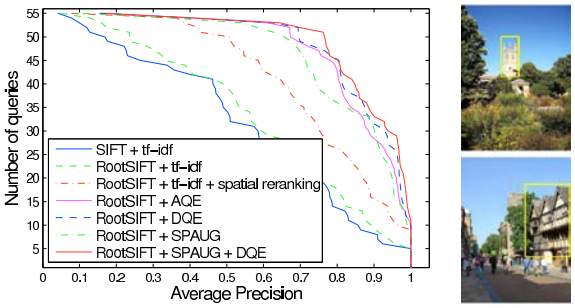


Figure 5. **Retrieval performance of various methods.** (left) The number of queries that achieve a specific AP or higher. Tested on Oxford 105k with the feature detector of [21] and Oxford 5k vocabulary. (right) Two badly performing queries for all methods.

Third, (h–l), using the feature detector of Perdoch *et al*. [21] boosts the performance even when using an independent vocabulary. The combined method, (l), does not top the results of [5, 17, 21], (i–k), due to problems of replicating their average query expansion score (our implementation gets a 4% worse mAP, probably caused by different parameter settings and heuristics). The performance on an independent vocabulary could be improved by using more powerful methods to construct the image graph for SPAUG: currently, it is computed without using DQE or soft-assignment. Another method which overcomes this independent vocabulary problem is that of Mikulik *et al*. [17], (j), where a very large vocabulary is generated and similarities between the words are learnt.

Finally, (m–o), use the Perdoch *et al*. [21] feature detector and a vocabulary generated from the test image dataset descriptors. The combined method sets the new state-of-the-art for the standard Oxford 5k and 105k, and Paris 6k benchmarks, achieving mAP scores of 0.929, 0.891 and 0.910 respectively.

Given this high performance, we can now ask "What is being missed?". Figure 5 (left) shows the performance of various methods in terms of the number of queries on Oxford 105k that achieve a specific average precision or above. The combined method, (o), achieves an AP of 0.7 or higher for all but 3 (out of 55) queries. The two worst performing queries for the combined, and all other methods, are shown in figure 5 (right). The top one fails because the query object is quite small and the lighting very bright, so there are not many distinctive features in the image. The bottom one fails because the query object is imaged from an extreme viewpoint, whilst repetitive patterns yield some spatially verified false positives (*e.g*. on fence railings). Thus, retrieval on this dataset is still not saturated.

7. Conclusions and recommendations for retrieval system design

RootSIFT. Using RootSIFT instead of SIFT improved retrieval performance in every single conducted experiment. We highly recommend it to be used as it provides a performance boost at no cost – it is very easy to implement, does not increase storage requirements as SIFT can be converted to RootSIFT on the fly with negligible computational cost.

Note that RootSIFT is not specific to object retrieval – all systems which use SIFT (*e.g*. image classification, object detection) could potentially benefit from switching to RootSIFT and we encourage everyone to try it as the conversion is very simple to implement.

Discriminative query expansion (DQE). DQE consistently outperforms average query expansion (AQE). It is as efficient as AQE since SVM training is negligible and re-querying requires equal computational resources. Implementation complexity is only slightly increased compared to AQE due to the additional training stage, however this is insignificant as many SVM packages are publicly available. As there are no arguments against DQE we recommend it to be used instead of AQE in all situations.

To our knowledge this is the first time that discriminative learning methods have been employed in the area of large scale retrieval.

Database-side feature augmentation (AUG). AUG is a useful method of increasing recall. It is not computationally demanding at runtime, but it does require a lengthy preprocessing graph construction stage. We recommend it to be used as it is a natural complement to query expansion. Our extension to the basic method improves precision but increases storage requirements; this trade-off should be kept in mind when deciding whether to use it or not.

Query descriptors soft assignment. Soft assignment of descriptors to visual words alleviates the problems caused by descriptor quantization to some extent, but in the original implementation of [24] soft-assignment was applied to the database, thus leading to a large increase in storage requirements as the BoW vectors representing the images are consequently more dense (than using hard assignment). Instead, we recommend Jégou *et al*. [7, 10]'s approach of only soft assigning the query descriptors (not the database images) thus not changing the storage requirements and only marginally increasing the query processing time.

Acknowledgements. We thank Michael Isard for comments, and are grateful for financial support from ERC grant VisRec no. 228180.

References

[1] http://cmp.felk.cvut.cz/∼perdom1/code/index.html.
[2] http://www.flickr.com/.
[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2:27:1–27:27, 2011.
[4] O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE PAMI*, 32(2):371–377, 2010.
[5] O. Chum, A. Mikulik, M. Perdoch, and J. Matas. Total recall II: Query expansion revisited. In *Proc. CVPR*, 2011.
[6] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*, 2007.
[7] M. Jain, H. Jégou, and P. Gros. Asymmetric hamming embedding. In *ACM Multimedia*, 2011.
[8] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proc. ECCV*, 2008.
[9] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *Proc. CVPR*, Jun 2009.
[10] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87(3):316–336, 2010.
[11] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010.
[12] M. Johnson. Generalized descriptor compression for storage and matching. In *Proc. BMVC.*, 2010.
[13] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *Proc. ECCV*, 2010.
[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
[15] A. Makadia. Feature tracking for wide-baseline image retrieval. In *Proc. ECCV*, 2010.
[16] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 1(60):63–86, 2004.
[17] A. Mikulik, M. Perdoch, O. Chum, and J. Matas. Learning a fine vocabulary. In *Proc. ECCV*, 2010.
[18] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, 2006.
[19] O. Pele and M. Werman. A linear time histogram metric for improved sift matchings. In *Proc. ECCV*, 2008.
[20] O. Pele and M. Werman. The quadratic-chi histogram distance family. In *Proc. ECCV*, 2010.
[21] M. Perdoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *Proc. CVPR*, 2009.
[22] F. Perronnin, J. Sanchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *Proc. CVPR*, 2010.
[23] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.
[24] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*, 2008.
[25] J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *Proc. ECCV*, 2010.
[26] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *Proc. ICVGIP*, 2008.
[27] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors. In *Proc. CVPR*, 2011.
[28] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003.
[29] T. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD)*, 2009.
[30] J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *Proc. ECCV*, 2008.
[31] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proc. CVPR*, 2010.
[32] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *Proc. CVPR*, 2009.
[33] J. Winn, Criminisi, A., and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. ICCV*, 2005.