Object retrieval with large vocabularies and fast spatial matching

James Philbin[1], Ondřej Chum[1], Michael Isard[2], Josef Sivic[1] and Andrew Zisserman[1]
[1]Department of Engineering Science, University of Oxford
[2]Microsoft Research, Silicon Valley
{james,ondra,josef,az}@robots.ox.ac.uk misard@microsoft.com

Abstract

In this paper, we present a large-scale object retrieval system. The user supplies a query object by selecting a region of a query image, and the system returns a ranked list of images that contain the same object, retrieved from a large corpus. We demonstrate the scalability and performance of our system on a dataset of over 1 million images crawled from the photo-sharing site, Flickr [3], using Oxford landmarks as queries.

Building an image-feature vocabulary is a major time and performance bottleneck, due to the size of our dataset. To address this problem we compare different scalable methods for building a vocabulary and introduce a novel quantization method based on randomized trees which we show outperforms the current state-of-the-art on an extensive ground-truth. Our experiments show that the quantization has a major effect on retrieval quality. To further improve query performance, we add an efficient spatial verification stage to re-rank the results returned from our bag-of-words model and show that this consistently improves search quality, though by less of a margin when the visual vocabulary is large.

We view this work as a promising step towards much larger, "web-scale" image corpora.

1. Object retrieval from a large corpus

We are motivated by the problem of retrieving, from a large corpus of images, the subset of images that contain a query object.[1] In practice, no algorithm will be able to make a perfect binary determination of whether or not an image lies in the query subset, and in fact even human judges may disagree on this due to occlusion, distortion, *etc*. We therefore address the slightly different problem of ranking each image in the corpus to determine the likelihood that it contains the query object, and aim to return to the user some prefix of this ranked list, in descending rank order.

A naive and inefficient solution to this task would be to formulate a ranking function and apply it to every image in the dataset before returning a ranked list. This is very computationally expensive for large corpora and the standard method in text retrieval [4, 8] is to use a bag of words model, efficiently implemented as an inverted file data-structure. This acts as an initial "filtering" step, greatly reducing the number of documents that need to be considered.

Recent work in object based image retrieval [20, 24] has mimicked simple text-retrieval systems using the analogy of "visual words." Images are scanned for "salient" regions and a high-dimensional descriptor is computed for each region. These descriptors are then quantized or clustered into a vocabulary of visual words, and each salient region is mapped to the visual word closest to it under this clustering. An image is then represented as a bag of visual words, and these are entered into an index for later querying and retrieval. Typically, no spatial information about the image-location of the visual words is used in the filtering stage.

Despite the analogy between "visual words" and words in text documents, the trade-offs in ranking images and web pages are somewhat different. An image query is generated from an example image region and typically contains many more words than a text query. The words are "noisier" however: in the web search case the user deliberately attempts to choose words that are relevant to the query, whereas the choice of words is abstracted away by the system in the image-retrieval case, and cannot be understood or guided by the user. Consequently, while web-search engines usually treat every query as a conjunction, object-retrieval systems typically include images that contain only, for example, 90% of the query words, in the filtered set.

The biggest difference, however, is that the visual words in an image-retrieval query encode vastly more spatial structure than a text query. A user who types a three-word text query may in general be searching for documents containing those three words in any order, at any positions in the document. A visual query however, since it is selected from a sample image, automatically and inescapably in-

[1]The query object is specified by a user selecting part of a query image, so it is really a "query region" however we will refer to it as an object to avoid overloading the term region.

cludes visual words in a spatial configuration corresponding to some view of the object; it is therefore reasonable to try to make use of this spatial information when searching the corpus for different views of the same object.

In this paper we investigate two directions for improving visual object-retrieval performance. In both cases we are guided by the constraint that the methods should be scalable to extremely large image corpora.

1. Improving the visual vocabulary. As noted above, image-based retrieval systems extract high-dimensional region descriptors from images, then cluster these to form a vocabulary of visual words. Early systems [24] used a flat k-means clustering that was effective but difficult to scale to large vocabularies. More recent work has used cluster hierarchies [17] and greatly increased the visual-word vocabulary size [20] using them. We show in section 3 that flat k-means can be scaled to similarly large vocabulary sizes by the use of approximate nearest neighbor methods. As will be demonstrated, this method has similar complexity to the vocabulary tree, but far superior performance. For the approximate nearest neighbors we employ a random forest algorithm [5, 15, 23]. This algorithm has recently been extensively used for supervised classification [19, 25] and unsupervised matching [21].

2. Incorporating spatial information into the ranking. Ideally we would like to verify that the target and query image regions were generated by the same object/scene region. Correspondence issues have been well studied both in the transformations required and in their estimation [14]. For example, two views of a rigid object are related by epipolar geometry, two views of a planar patch are related by a homography, *etc*. Such mappings can be computed from correspondences of salient regions between the target and query images. Indeed, correspondences (to verify a match) can even be extended to deformations [11]. In practice for collections such as consumer photographs, it may not be necessary to consider such general geometric transformations. This is a question that can be assessed empirically by re-ranking on transformations of varying generality, and to this end we investigate a set of ranking methods, all of which can be implemented efficiently, in section 4. We find that we are able to deal with image variations such as lighting, shadows and partial occlusions without explicitly modeling them.

For a concrete application, we choose searching on building facades and architectural features. This can be very challenging because of the near ambiguities that arise from repetitions of architectural building blocks: windows, doors etc. We use photos from the photo-sharing site Flickr [3], as this contains many examples of the typical building shots that tourists capture.

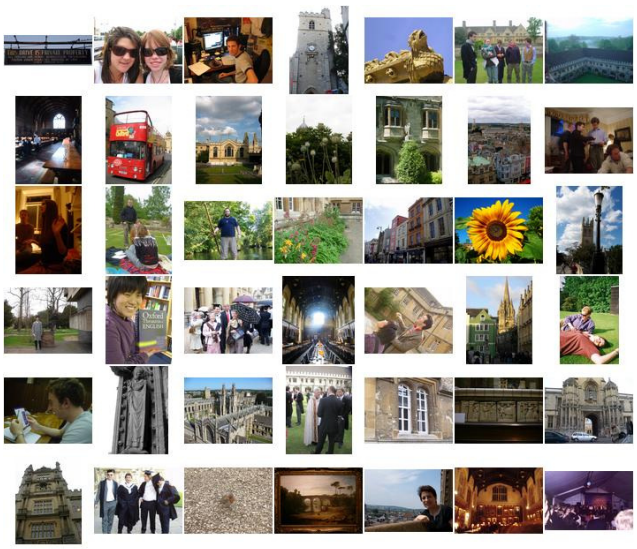


Figure 1. 42 randomly sampled images from the 5K dataset. Note that the dataset contains difficult distractors which may easily be confused with those used in the query set.

Dataset	# images	# features	Size of descriptors
5K	5,062	16,334,970	1.9 GB
100K	99,782	277,770,833	33.1 GB
1M	1,040,801	1,186,469,709	141.4 GB
Total	1,145,645	1,480,575,512	176.4 GB

Table 1. The number of images, features and descriptor sizes for each dataset.

2. The Datasets, Evaluation & Implementation

Oxford 5K dataset. To evaluate performance when comparing different visual vocabularies and spatial rankings, we have collected a set of images comprising 11 different Oxford "landmarks" – by landmark here we mean a particular part of a building – together with distractors. Images for each landmark were retrieved from Flickr [3], using queries such as "Oxford Christ Church" and "Oxford Radcliffe Camera." We also retrieved further distractor images by seaching on "Oxford" alone. The entire dataset consists of 5,062 high resolution (1024×768) images. Sample images from the dataset are shown in figure 1.

For each landmark we chose 5 different query regions, as shown in figure 2. The five queries are used so that retrieval performance can be averaged over any individual query peculiarities.

We obtain ground truth manually by searching over the entire dataset for the 11 landmarks. Images are assigned one of four possible labels: (1) *Good* – a nice, clear picture of the object/building. (2) *OK* – more than 25% of the object is clearly visible.(3) *Junk* – less than 25% of the object is visible, or there is a very high level of occlusion or distortion. (4) *Absent* – the object is *not present*. The number of occurrences of the different landmarks range between 7 and 220 *good* and *ok* images. The dataset of images and the

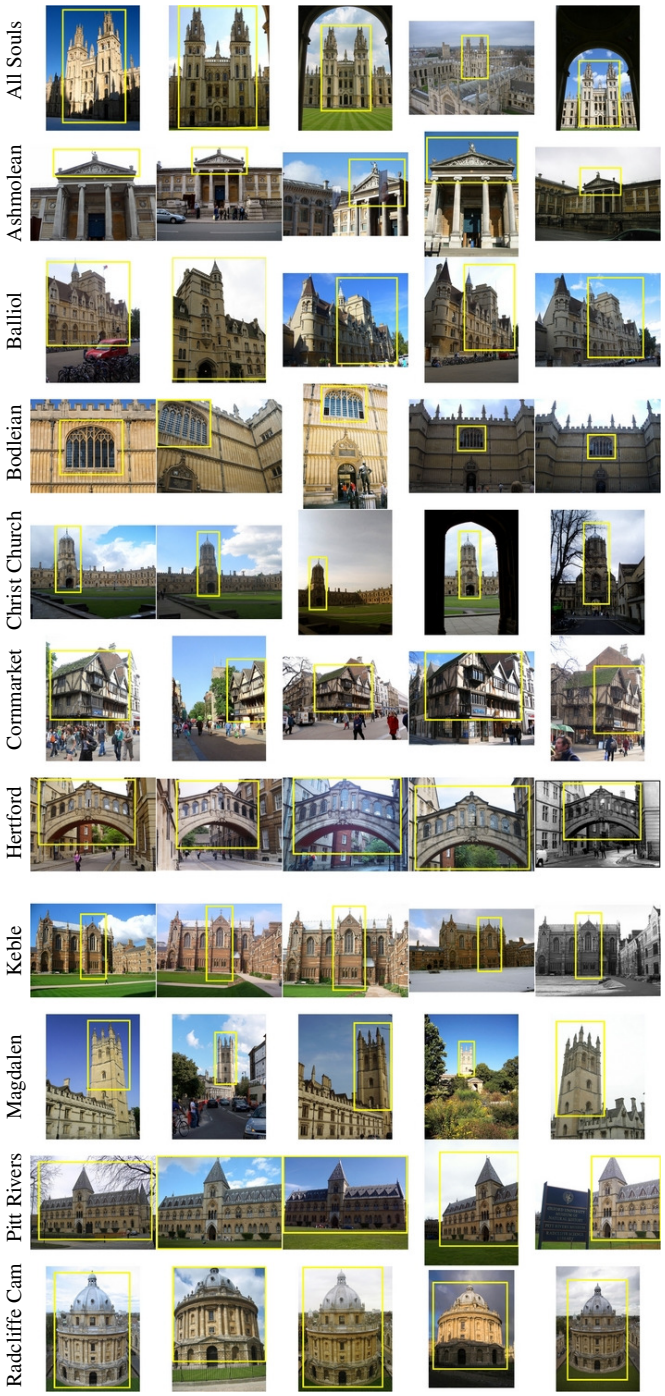


Figure 2. All 55 query images used in the ground truth evaluation. Each row shows different queries for the same scene landmark. Note the large variation in scale of the query regions and the variation in position, lighting, *etc*. of the images themselves.

ground truth labelling are available at [1].

In addition to this labelled set, we use two other datasets to stress-test retrieval performance when scaling up. These consist of images crawled from Flickr's list of most popular tags. The images in our datasets will not in general be disjoint when crawled from Flickr, so we remove exact duplicates from the sets. We then assume that these datasets contain no occurrences of the objects being searched for, so they act as *distractors*, testing both the performance and scalability of our system.

100K dataset. This data was crawled from Flickr's 145 most popular tags and consists of 99,782 high resolution (1024×768) images.

1M dataset. This data was crawled from Flickr's 450 most popular tags and consists of 1,040,801 medium resolution (500×333) images.

Table 1 summarizes the relative sizes of the datasets.

2.1. Performance evaluation

To evaluate the performance we use the average precision (AP) measure computed as the area under the precision-recall curve for a query. Precision is defined as the ratio of retrieved positive images to the total number retrieved. Recall is defined as the ratio of the number of retrieved positive images to the total number of positive images in the corpus. An ideal precision-recall curve has precision 1 over all recall levels and this corresponds to an average precision of 1.

We compute an average precision score for each of the 5 queries for a landmark, averaging these to obtain a mean Average Precision (mAP) score. The average of these mAP scores is used as a single number to evaluate the overall performance.

In computing the average precision, we use the *Good* and *Ok* images as positive examples of the landmark in question, *Absent* images as negative examples and *Junk* images as null examples. These null examples are treated as though they are not present in the database – our score is unaffected whether they are returned or not.

2.2. Implementation

This section overviews our bag-of-visual-words real-time object retrieval engine.

Image description. For each image in the dataset, we find affine-invariant Hessian regions [18]. Typically there are 3,300 regions detected on an image of size 1024×768. For each of these affine regions, we compute a 128-D SIFT descriptor [16]. The number of descriptors generated for each of our datasets is shown in table 1.

A sample of the visual descriptors are quantized and then used to index the images for the search engine. In section 3.3 we describe the number of descriptors and words used for quantization, but in all cases the visual vocabulary is computed on the 5K dataset.

Search Engine. Our search engine uses the vector-space model [7] of information-retrieval. The query and each document in the corpus is represented as a sparse vector of term (visual word) occurrences and search proceeds by calculat-

ing the similarity between the query vector and each document vector, using an L_2 distance. We use the standard tf-idf weighting scheme [7], which down-weights the contribution that commonly occurring, and therefore less discriminative, words make to the relevance score.

For computational speed, the engine stores word occurrences in an index, which maps individual words to the documents in which they occur. In the worst case, the computational complexity of querying the index is linear in the corpus size, but in practice it is close to linear in the number of documents that match a given query, generally a major saving. For sparse queries, this can result in a substantial speedup, as only documents which contain words present in the query need to be examined. The scores for each document are accumulated so that they are identical to explicitly computing the similarity.

With large corpora of images, memory usage becomes a major concern. To help ameliorate this problem, the inverted index is stored in a space-efficient binary-packed structure. Additionally, when main memory is exhausted, the engine can be switched to use an inverted index flattened to disk, which caches the data for the most frequently requested words.

For example, for a vocabulary size of 1M words, our search engine implementation can query the combined `5K+100K` datasets in approximately 0.1s for a typical query and the inverted index consumes 1GB of main memory. The size of the index for the combination of the `5K+100K+1M` datasets is 4.3GB, larger than our available main memory, so we use an offline version of the index flattened to disk. Querying this corpus from disk takes around 15s – 35s for a typical query.

3. Visual vocabularies from scalable clustering

Generating clusters for such a large quantity of data presents challenges to traditionally used algorithms. Even sub-sampling 5% of the `100K` dataset would still require clustering roughly 28 million 128 dimensional descriptors. The size of the data essentially rules out methods such as mean-shift, spectral and agglomerative clustering. Even "simpler" clustering algorithms such as exact k-means fail to scale to this kind of size. Some work has been done on accelerating exact k-means [10], but this requires $\mathcal{O}(K^2)$ extra storage, where K is the number of cluster centers, rendering it impractical for our purposes.

In this work, we compare the performance of two different clustering methods: (a) approximate k-means, and (b) hierarchical k-means [20]. The two methods are described in detail below.

3.1. Approximate k-means (AKM)

The first method is an alteration to the original k-means algorithm. In typical k-means, the vast majority of computation time is spent on calculating nearest neighbours between the points and cluster centers. We replace this exact computation by an approximate nearest neighbor method, and use a forest of 8 randomized k-d trees [5, 15, 23] built over the cluster centers at the beginning of each iteration to increase speed. We use randomized k-d tree code, optimized for matching SIFT descriptors, supplied by Lowe [16]. Usually in a k-d tree, each node splits the dataset using the dimension with the highest variance for all the data points falling into that node and the value to split on is found by taking the median value along that dimension (although the mean can also be used). In the randomized version, the splitting dimension is chosen at random from among a set of the dimensions with highest variance and the split value is randomly chosen using a point close to the median. The conjunction of these trees creates an overlapping partition of the feature space and helps to mitigate quantization effects, where features which fall close to a partition boundary are assigned to an incorrect nearest neighbour. This robustness is especially important in high-dimensions, where due to the "curse of dimensionality" [22], points will be more likely to lie close to a boundary.

A new data point is assigned to the (approximately) closest cluster center as follows. Initially, each tree is descended to a leaf and the distances to the discriminating boundaries are recorded in a single priority queue for all trees [6]. Then, we iteratively choose the most promising branch from all trees and keep adding unseen nodes into the priority queue. We stop once a fixed number of tree paths have been explored. This way, we can use more trees without significantly increasing the search time.

The algorithmic complexity of a single k-means iteration is now reduced from $\mathcal{O}(NK)$ to $\mathcal{O}(N\log(K))$, where N is the number of features being clustered from. Our tests have shown that at least for moderate values of K, the percentage of points assigned to different cluster centers differs from the exact version by less than 1%, motivating the approach. Note that this method has the same time and memory complexity as the hierarchical vocabulary tree clustering of [20] described below. If a scaling-up beyond reasonable memory requirements is needed, the top level branches of each tree can be distributed to different machines.

3.2. Hierarchical k-means (HKM)

Nistér and Stewénius [20] propose generating a "vocabulary tree" using a hierarchical k-means clustering scheme (also called tree structured vector quantization [13]). On the first level of the tree, all data points are clustered to a small number ($K = 10$) of cluster centers. On the next level, k-means (with $K = 10$ again) is applied within each of the partitions independently. The result is K^n clusters at the n-th level. For example, using a branching factor of 10 with 6 levels results in 1M leaf nodes. A new data point is assigned by descending the tree. Instead of assigning each data point to the single leaf node at the bottom of the tree, the points

Clustering parameters		mAP	
# of descr.	Voc. size	k-means	AKM
800K	10K	0.355	0.358
1M	20K	0.384	0.385
5M	50K	0.464	0.453
16.7M	1M		0.618

Table 2. Comparison of the performance of exact k-means to our AKM method on the 5K dataset, using different numbers of training descriptors and clusters.

can additionally be assigned to some internal nodes which their path from root to leaf passes through. This can help mitigate the effects of quantization error, for cases when the data point lies close to the Voronoi region boundary for each cluster center.

It is important to note that traditional flat k-means minimizes the total distortion between the data points and their assigned, closest cluster centers, whereas the hierarchical tree minimizes this distortion only locally at each node and this does not in general result in a minimization of the total distortion.

3.3. Results on comparing vocabularies

Our goal is to evaluate the retrieval performance of visual vocabularies built using the two clustering methods described above. Here, we test only the filtering stage of the retrieval system, i.e. retrieval is performed using only the inverted file (including the tf-idf weighting), and no ranking using the spatial configuration of regions is used. We perform three main experiments. Firstly, we compare performance using AKM to flat k-means. This is to establish how much, if any, performance is lost by the approximation. Secondly, we compare AKM to HKM. Thirdly, we investigate how the performance using AKM degrades as we scale up the number of images in the corpus.

k-means vs AKM. For the small 5K dataset, we compare AKM to exact k-means, using varying amounts of subsampled data and cluster centers with identical cluster initialization. These results are given in table 2, and show that our approximate method gives very similar performance to exact k-means, differing in mAP by less than 1% and outperforming k-means in two cases. This justifies the use of AKM as an effective proxy for exact k-means, but with a fraction of the computational cost.

HKM vs AKM. We compare our method to HKM in two ways. First, we compare performance on the Recognition Benchmark introduced by [20]. This consists of 10,200 images split into four image groups of the same scene taken from different viewpoints. A perfect result is to return, given a query, the other three images from that query's group before images from other groups. This is expressed as an average over the number of the top four correctly returned, taken over all possible query images. We also display a graph, showing how the query performance changes as increasingly large subsets of the data are searched over. To train our clusters, we use identical training and testing descriptors to [20] provided at [2], and an L_1 distance to compute the ranking. From table 3, we see that for the same number of visual words, our method significantly outperforms the hierarchical method.

Method	Scoring Levels	Average Top
HKM	1	3.16
HKM	2	3.07
HKM	3	3.29
HKM	4	3.29
AKM		**3.45**

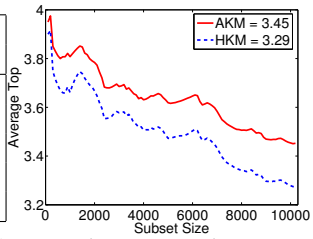


Table 3. A comparison of the AKM and HKM on the Recognition Benchmark of [20] using the descriptors for training and testing provided by the authors of [20]. "HKM" is the hierarchical k-means quantization, where the numbers are taken from [2]. "AKM" is the result of our approximate k-means clustering. Both methods use a vocabulary of 1M visual words and an L_1 distance.

Method	Dataset	mAP	
		Bag-of-words	Spatial
(a) HKM-1	5K	0.439	0.469
(b) HKM-2	5K	0.418	
(c) HKM-3	5K	0.372	
(d) HKM-4	5K	0.353	
(e) AKM	5K	0.618	0.647
(f) AKM	5K+100K	0.490	0.541
(g) AKM	5K+100K+1M	0.393	0.465

Table 4. Vocabulary comparison over the three datasets. For the HKM method, the number of levels used for scoring is listed in the method name. All methods use 1M cluster centers, generated from all 16.7M descriptors in the 5K dataset. The "spatial" method is described in section 4.

Second, we have also compared the performances of the two methods on our own 5K dataset, shown in table 4, rows (a)–(e), using our descriptors. Here, we have used our own implementation of HKM which we have found gives almost identical figures on the dataset from [2]. The AKM method clearly outperforms the best HKM method, by 0.618 to 0.439 mAP. This might be attributed to quantization effects of the vocabulary tree – data points may be suffering from bad initial splits close to the root of the vocabulary tree. As a result, descriptors arising from the same object/scene region in different images can be assigned (due to e.g. noise) to different clusters. Hierarchical scoring might partially overcome this problem, but we find that the hierarchical scoring actually hurts the performance of the HKM method. However, if we switch the vector scoring to use the L_1 distance (instead of L_2), we find that the hierarchical scoring improves performance, but doesn't produce as good a result as in the L_2 case (0.427 best L_1 vs. 0.439 best L_2). Clearly, more work is needed to understand the HKM performance here.

Vocab Size	Bag of words	Spatial
50K	0.473	0.599
100K	0.535	0.597
250K	0.598	0.633
500K	0.606	0.642
750K	0.609	0.630
1M	**0.618**	**0.645**
1.25M	0.602	0.625

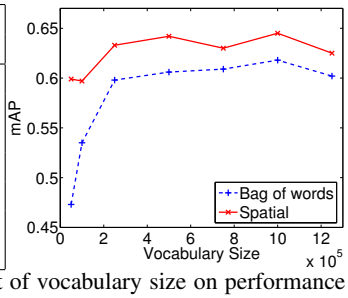


Table 5. Examining the effect of vocabulary size on performance for the 5K dataset. Each vocabulary is trained using AKM on all 16.7M descriptors. There is a performance peak at 1 million words. The spatial verification consistently improves performance.

Scaling up with AKM. We explore a number of different vocabulary sizes for the 5K dataset in table 5. This shows a peak in performance at 1M visual words, although for large numbers of clusters, the performance curve appears quite flat and we predict the performance would not significantly degrade for moderately larger vocabularies.

We evaluate the scalability of our method on the 5K, 5K+100K and 5K+100K+1M datasets in table 4, rows (e)–(g), using the 1M words visual vocabulary. In going from the smallest dataset to the largest, a 226-fold increase in the number of images, the performance falls by just over 20%. We attribute this drop in performance to a lack of sufficient discrimination in the quantization for the larger dataset. As will be seen, this performance loss is ameliorated to some extent once spatial ranking is included.

4. Spatial re-ranking

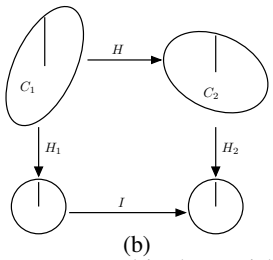
The output from performing a query on the inverted file described previously is a ranked list of images for a significant section of the corpus. We have until now considered the features in each image as a visual bag-of-words and have ignored the spatial configurations of features. We now investigate re-ranking the top-ranked results using spatial constraints. The spatial verification procedure estimates a transformation between the query region and each target image, based on how well its feature locations are predicted by the estimated transformation. We then re-rank target images based on the discriminability of the spatially verified visual words.

4.1. Transformations and their estimation

As is now standard in estimation algorithms on visual data, two types of measurement error must be considered: errors in a detected feature's position and shape; and errors due to outliers from mismatched or missing features, because of detector failure, occlusion, etc. The standard solution is to use the RANSAC algorithm [12]; this involves generating transformation hypotheses using a minimal number of correspondences and then evaluating each hypothesis based on the number of "inliers" among all features under that hypothesis.

Transformation	dof	Matrix
translation + isotropic scale	3	$\begin{bmatrix} a & 0 & t_x \\ 0 & a & t_y \end{bmatrix}$
translation + anisotropic scale	4	$\begin{bmatrix} a & 0 & t_x \\ 0 & b & t_y \end{bmatrix}$
translation + vertical shear	5	$\begin{bmatrix} a & 0 & t_x \\ b & c & t_y \end{bmatrix}$

(a)



(b)

Table 6. (a) The three affine sub-groups compared in the spatial re-ranking. (b) Computing H as $H_2^{-1}H_1$, preserving "upness" for the 5 dof case.

Typically, photos are taken from a restricted range of canonical views and we can use this prior information to speed up transformation estimation. We choose to use LO-RANSAC [9], a variant of RANSAC. It involves generating hypotheses of an approximate model and then iteratively re-evaluating promising hypotheses using the full transformation. By selecting a restricted class of transformations for the hypothesis generation stage and exploiting shape information in the affine-invariant image regions, we are able to generate hypotheses with only a *single* pair of corresponding features. This greatly reduces the number of possible hypotheses which need to be considered and significantly speeds up the matching procedure. We therefore choose to enumerate *all* such hypotheses, which removes the randomness from our algorithm, resulting in a deterministic procedure.

We compare three affine sub-groups for hypothesis generation, with degrees of freedom ranging between 3 and 5, that are listed in table 6(a). This is to evaluate whether or not there is any significant performance difference between transformation types. In each case we use a general (6 dof) affine transformation for the iterative re-estimation step of LO-RANSAC. The 3 dof transformation approximately covers situations such as a change in zoom or camera distance to the scene, but not foreshortening. The 4 dof transformation approximately covers foreshortening by either a horizontal or vertical scaling between views. The 5 dof transformation preserves the vertical direction and allows for anisotropic scaling and vertical shear. These three models take advantage of the fact that images are usually displayed on the web with the correct (upright) orientation. For this reason, we have not allowed for in-plane image rotations.

Implementation details. The 3 dof transformation (method (i) in the following results) is computed from a single region correspondence using the regions' centroids to estimate the translation, and each region's scale to estimate the isotropic scale change between the query region and the target image.

For the 4 dof transformation (method (ii)) from a single region correspondence, the scaling in the x direction is computed from the ratio of the regions' x extents (and similarly for the y scaling).

The 5 dof transformation (method iii) is estimated from

(a)

Method / Rerank N	100	200	400	800
i 3dof	0.468	0.492	0.522	0.556
ii 4dof	0.465	0.490	0.521	0.555
iii 5dof	0.467	0.491	0.526	0.560

(b)

Method / Rerank N	100	200	400	800
i 3dof	0.644	0.650	0.652	0.655
ii 4dof	0.646	0.656	0.659	0.661
iii 5dof	0.648	0.657	0.660	0.664

Table 7. Comparing the different transformation types on the 5K dataset. (a) Using a 20K visual vocabulary (bag of words = 0.385). (b) Using a 1M visual vocabulary (bag of words = 0.618).

a single correspondence of two elliptical regions, $C_1 \leftrightarrow C_2$. For the two regions, we consider the transforms H_1 and H_2 which will transform the ellipses to a unit circle (see figure 6(b)), such that the orientation of the unit vector in the y-direction is maintained ($(0,1)^\top$ is an eigenvector of the transform). The overall transform is then given by $H_2^{-1}H_1$.

The 6 dof transformation is estimated from the centroids of the current inlier set, returned from the simpler transforms.

We have tried several different error functions [14] to determine inlier correspondences, including a one-way transfer error (from query to target and vice-versa), a two-way transfer error, and a two-way transfer error with a threshold on the expected. We find that the two-way transfer error with scale threshold performs the best on the data. In cases where a simpler one-way transfer error suffices, we can speed up verification when there is a high visual-word multiplicity between the two images (usually for smaller vocabularies). We merely need to find the spatially closest matching feature in the target to a particular query feature and check whether this single distance is less than the threshold. This is done using a 2D k-d tree, to provide logarithmic time search.

In all experiments where "spatial" is specified, we have used our spatial verification procedure to re-rank up to the top 1000 images. We consider spatial verification to be "successful" if we find a transformation with at least 4 inlier correspondences. We re-rank the images by scoring them equal to the sum of the idf values for the inlier words and place spatially verified images above unverified ones in the ranking. We abort the re-ranking early (after considering fewer than 1000 images) if we process 20 images in a row without a successful spatial verification. We find this gives a good trade-off between speed and accuracy.

4.2. Results on comparing spatial rankings

In this section, we evaluate the performance of our spatial re-ranking on the complete retrieval system for all datasets and compare the effects of the different transformation types described above.

Tables 7(a) and (b) show the mAP after applying different spatial ranking methods to the top 100, 200, 400 and 800 images returned by the initial filtering stage, for the 5K dataset, using a 20K vocabulary and a 1M vocabulary respectively. As shown, spatial re-ranking significantly improves the retrieval quality of the system, although the margin is less for the more discriminative vocabulary. As for the transformation types, in general, 5 dof outperforms 4 dof, which outperforms 3 dof, but the improvement is small. This may be because the affine re-estimation evens out the differences in transformation type.

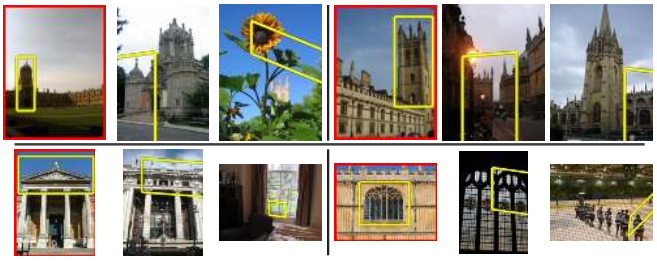


Figure 3. Examples of errors in retrieval for four different query images. The query image is shown outlined in red, with two highly ranked (top 100) false positives to the right. Some false positives are visually plausible, but others match due to visual ambiguities in the local regions and low numbers of matched visual words.

Table 4(e)–(g) show the results of the spatial re-ranking on the large datasets with consistent improvements of around 5%. This shows that even though our verification method only sees up to the top 1000 results, it is still able to sufficiently boost the retrieval quality.

Figure 4 shows example queries retrieved from the 5K dataset, using the complete retrieval system. The results demonstrate the considerable variations in viewpoint, scale, lighting and partial occlusion which are present in the corpus. Figure 3 illustrates examples of errors in retrieval.

5. Conclusions and further work

We demonstrate a scalable visual object-retrieval system that uses a corpus of photos taken from a public web site. The system returns photos from the corpus that contain a query object, despite substantial differences in lighting, perspective, image quality and occluders between the query and retrieved images.

We view this as a step towards the ultimate goal of building a web-scale retrieval system that will scale to billions of images. We have therefore concentrated on algorithms which are scalable, and which we believe are straightforward to distribute over a cluster of computers.

More evaluation is needed to determine how close we are to a ranking function that will correctly return matching objects as the corpus size grows. However, even if the ranking function is adequate, we believe that retrieval performance will not scale unless we find efficient ways to include spatial information in the index, and move some of the burden of spatial matching from the ranking stage to the filtering stage. We leave this problem as future work.

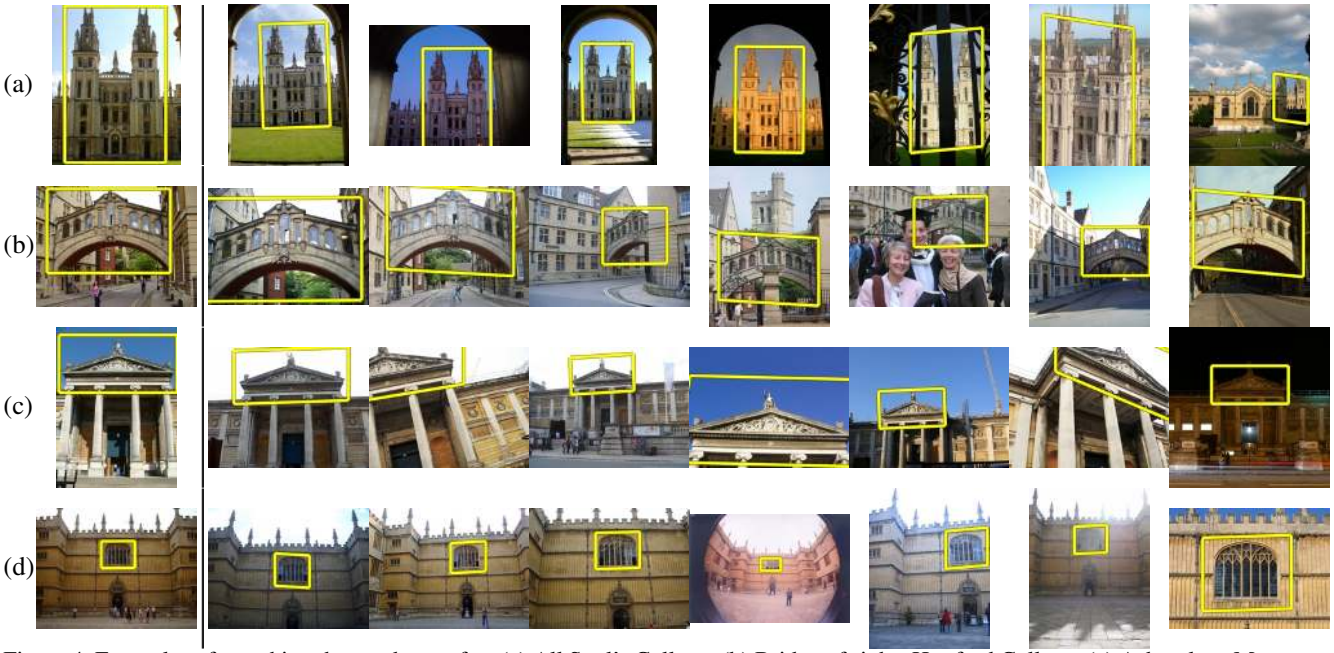


Figure 4. Examples of searching the 5K dataset for: (a) All Soul's College. (b) Bridge of sighs, Hertford College. (c) Ashmolean Museum. (d) Bodleian window. The query is shown on the left, with selected top ranked retrieved images shown to the right. All results displayed are returned before the first false positive for each query.

Acknowledgements. We thank David Lowe for discussions and for providing his k-d tree code and Henrik Stewénius for providing his dataset for comparison. We are grateful for support from the Royal Academy of Engineering, the EU Visiontrain Marie-Curie network, the EPSRC and Microsoft.

References

[1] http://www.robots.ox.ac.uk/∼vgg/data/.

[2] http://www.vis.uky.edu/∼stewe/ukbench/data/.

[3] http://www.flickr.com/.

[4] Y. Aasheim, M. Lidal, and K. Risvik. Multi-tier architecture for web search engines. *In Proc. Web Congress*, 2003.

[5] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computing*, 9(7):1545–1588, 1997.

[6] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[7] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, ISBN: 020139829, 1999.

[8] L. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *Micro, IEEE*, 23, 2003.

[9] O. Chum, J. Matas, and Š. Obdržálek. Enhancing RANSAC by generalized model optimization. In *Proc. ACCV*, 2004.

[10] C. Elkan. Using the triangle inequality to accelerate kmeans, 2003.

[11] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. In *Proc. ECCV*, 2004.

[12] M. A. Fischler and R. C. Bolles. Random sample consensus. *Comm. ACM*, 24(6):381–395, 1981.

[13] A. Gersho and R. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Boston, 1992.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[15] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. CVPR*, June 2005.

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[17] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *Proc. CVPR*, 2006.

[18] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 1(60):63–86, 2004.

[19] F. Moosman, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *NIPS*, 2006.

[20] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, 2006.

[21] Š. Obdržálek and J. Matas. Sub-linear indexing for large scale object recognition. In *Proc. BMVC.*, 2005.

[22] U. Shaft, J. Goldstein, and K. Beyer. Nearest neighbours query performance. Technical report, 1998.

[23] C. Silpa-Anan and R. Hartley. Localization using an image-map. In *Proc. ACRA*, 2004.

[24] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, Oct 2003.

[25] J. Winn and A. Criminisi. Object class recognition at a glance. In *In Video Proc. CVPR*, 2006.