Advancing Large Scale Object Retrieval

D.Phil Thesis

Robotics Research Group
Department of Engineering Science
University of Oxford

Supervisor:
Professor Andrew Zisserman

Relja Arandjelović
Christ Church

2013

Relja Arandjelović
Christ Church

Doctor of Philosophy
December 2013

Advancing Large Scale Object Retrieval

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. Such a system has a wide variety of applications including object or location recognition, video search, near duplicate detection and 3D reconstruction. The task is very challenging because of large variations in the imaged object appearance due to changes in lighting conditions, scale and viewpoint, as well as partial occlusions.

A starting point of established systems which tackle the same task is detection of viewpoint invariant features, which are then quantized into visual words and efficient retrieval is performed using an inverted index. We make the following three improvements to the standard framework: (i) a new method to compare SIFT descriptors (RootSIFT) which yields superior performance without increasing processing or storage requirements; (ii) a novel discriminative method for query expansion; (iii) a new feature augmentation method.

Scaling up to searching millions of images involves either distributing storage and computation across many computers, or employing very compact image representations on a single computer combined with memory-efficient approximate nearest neighbour search (ANN). We take the latter approach and improve VLAD, a popular compact image descriptor, using: (i) a new normalization method to alleviate the burstiness effect; (ii) vocabulary adaptation to reduce influence of using a bad visual vocabulary; (iii) extraction of multiple VLADs for retrieval and localization of small objects. We also propose a method, SCT, for extremely low bit-rate compression of descriptor sets in order to reduce the memory footprint of ANN.

The problem of finding images of an object in an unannotated image corpus starting from a textual query is also considered. Our approach is to first obtain multiple images of the queried object using textual Google image search, and then use these images to visually query the target database. We show that issuing multiple queries significantly improves recall and enables the system to find quite challenging occurrences of the queried object.

Current retrieval techniques work only for objects which have a light coating of texture, while failing completely for smooth (fairly textureless) objects best described by shape. We present a scalable approach to smooth object retrieval and illustrate it on sculptures. A smooth object is represented by its imaged shape using a set of quantized semi-local boundary descriptors (a bag-of-boundaries); the representation is suited to the standard visual word based object retrieval. Furthermore, we describe a method for automatically determining the title and sculptor of an imaged sculpture using the proposed smooth object retrieval system.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Relja Arandjelović, Christ Church

Acknowledgements

I am very grateful to my supervisor, Professor Andrew Zisserman, for sharing his knowledge and keeping my PhD on a good track. His extensive expertise, as well as passion, devotion, enthusiasm and optimism have made this thesis possible. I am looking forward to continuing to learn from him even more during my postdoc.

I would also like to thank James Philbin for sharing his code and providing guidance at the beginning of my PhD, enabling me to immediately start working on interesting research problems.

The labmates and friends from VGG have been invaluable in my everyday life. Amr has always been there to help with encountered problems, as well as to relax by going to play some of our many games of pool (score to date: Relja 294, Amr 147; exactly 2:1, I was right!). Special thanks also go to the remaining VGG-twins, Yusuf and Karen. Joining the lab was made easier by helpful and friendly Arpit, Varun, Patrick and Maria-Elena, while the lab continued to be a fun place due to Mircea, Chai (a.k.a. Yuning), Max, Carlos, Ken, Omkar, Tomas, Meelis, Elliot, Victor, Lubor, Eric, Marcin, Alonzo, Minh, Matthew, Esa, Nataraj, Siddhartha, Mayank and Karel. I wish them all the best in life and hope to keep in touch.

Most importantly, I want to thank my girlfriend Marianna for making rainy Oxford feel sunny, friends for life Miloš, Igor, Dionyssis and Evangelos who though far away have remained important in my life, and, of course, my brother Ognjen and parents Nataša and Đorđe to whom I dedicate this thesis.

Contents

Chapter 1

Introduction

1.1 Objectives and motivation

The objective of this thesis is to create systems capable of performing large scale image search purely based on visual information. The search has to be fully automatic and executed in near-real time when searching large scale databases containing millions of unannotated images. We focus on searching for particular object instances such as landmarks, paintings, sculptures, logos, books covers, etc. Since objects can be imaged from different viewpoints and at various scales, they typically occupy only a part of an image; our goal is thus not to retrieve similar images or scenes, but to search on the object level. Note that we do not consider classes of objects, such as buildings in general, but we target specific objects, such as Buckingham Palace.

Such a large scale visual search system has a wide variety of applications, some of which are listed next.

What is this? A user can take a photo with his mobile phone and ask the system for more information about the object in the photo. For example, a museum visitor can obtain additional information about a piece of art, a buyer can look up the prices for the imaged item at other shops, or an autonomous robot can determine how to interact with the object. Google Goggle is an example of such a system.

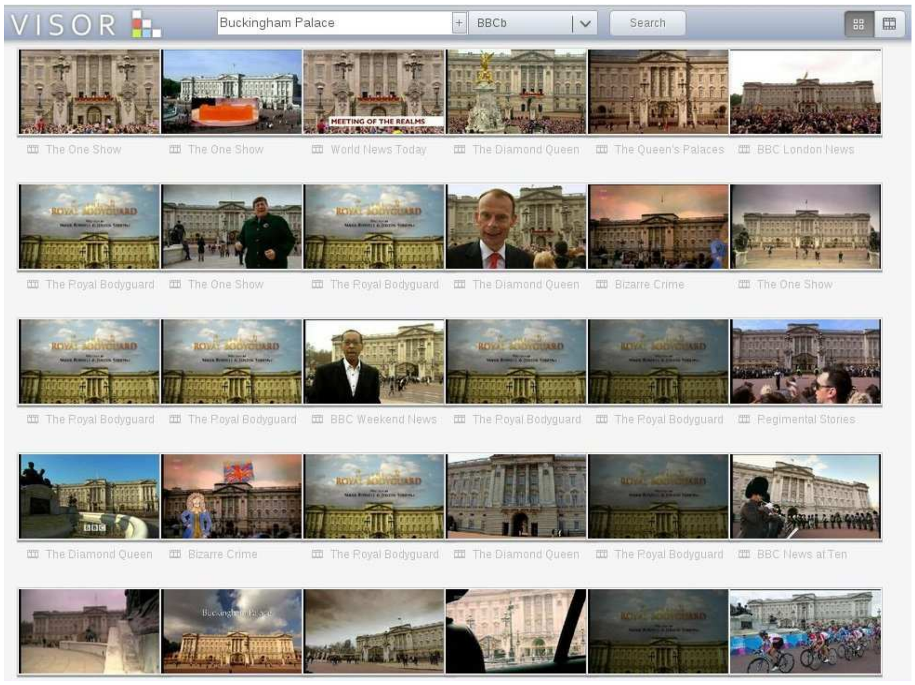
Where am I? Geolocalization of an image can be performed via object retrieval, by searching a large database of images which contain location information, for example Google Street View images or images with GPS metadata. This is useful for devices which don't have GPS or GPS usage is expensive (e.g. due to large battery use), or for improving the localization accuracy of a system which has consumer-grade GPS. Furthermore, the system is, unlike GPS, capable of functioning indoors or even at other planets.

Automatic annotation. Personal photos can be tagged automatically with objects or places in order to facilitate easy search and navigation through ever increasing personal photo collections. For example, one could easily find an answer to the query "my photos of Christ Church" without having to remember the albums and dates of all visits to that Oxford college.

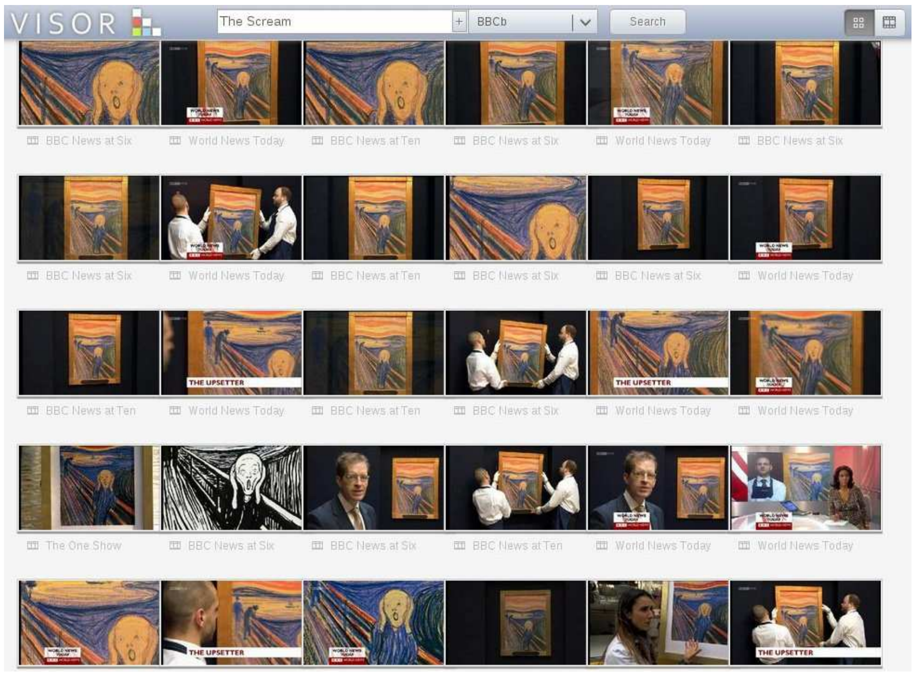
Augmented reality. With upcoming products such as Google Glass, recognizing objects in real time can be used to provide additional useful information to the user.

Video search. Huge amounts of video data are generated all the time making it impossible to manually annotate all of it. A visual retrieval system is useful, for example, to production teams of news programmes, as they can easily find clips or images to reuse from their own archives.

Tool for other Computer Vision systems. There are a large number of Com-



(a) query: Buckingham Palace



(b) query: The Scream

Figure 1.1: **Searching BBC videos.** Top results for on-the-fly visual search with textual queries "Buckingham Palace" and "The Scream" (section 6) in a database containing 125 days (3000 hours) of video footage. GUI created by Ken Chatfield.

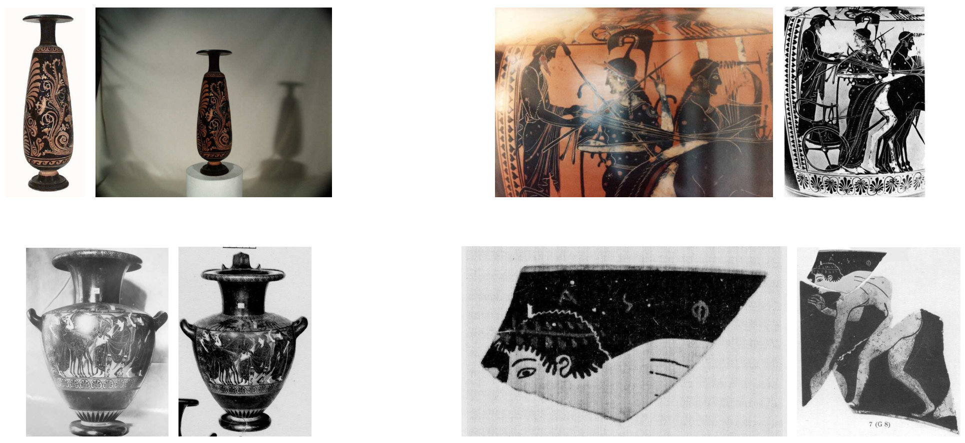


Figure 1.2: **Vase copies.** Automatically detected pairs of images of the same vase which were erroneously assigned different vase IDs in the database meta-data.

puter Vision systems which benefit from visual retrieval. For example, automatic 3D reconstruction usually starts with visual clustering (via retrieval) of large image sets, while inpainting and deblurring can be helped by retrieving images containing the imaged objects.

1.1.1 Specific applications

In this section we describe a few specific applications of object retrieval, some of which we worked on.

Video search. We are collaborating with BBC to provide them with video search functionality. This is useful to, for example, their production teams – when creating a reportage the production teams often buy stock photos or videos of a particular object or place. Searching for objects inside the BBC video collection enables them to reuse their own material instead. Figure 1.1 shows examples of this functionality.

Database consistency checking. A large image database can be checked for

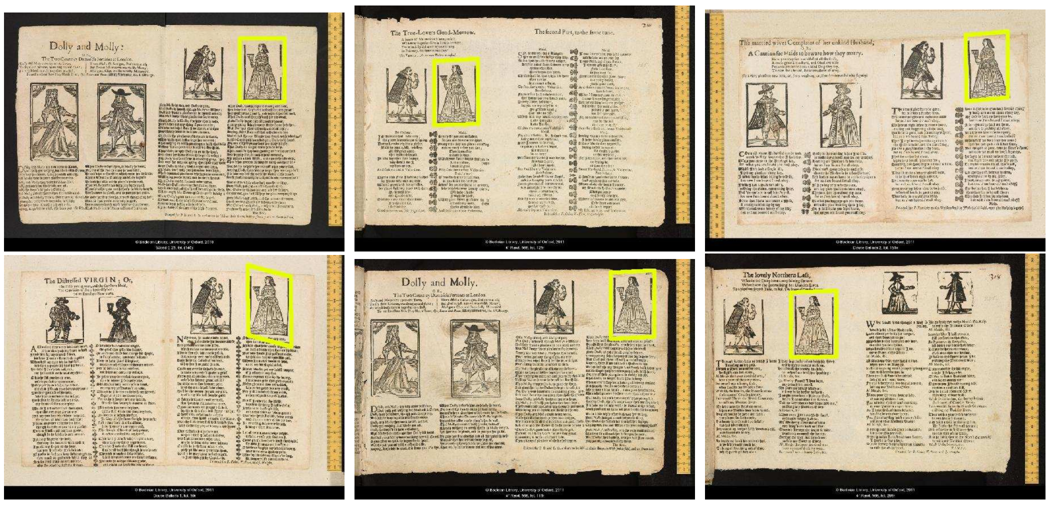


Figure 1.3: **Searching printed illustrations.** Top left image is the query, the remaining images show the top retrieved results; regions of interest are overlaid in yellow.

consistency by automatically clustering images of same objects and checking if the clusters correspond to meta-data well. We performed this task on a database of antique Greek vases for CLAROS, the project which collect art from six museums from four European countries. Figure 1.2 shows some of the inconsistencies discovered by our method.

Theft detection. Museums in badly developed or war torn countries are often looted. A database of photos of items in endangered museums can be created and visual search can be used to detect a stolen object.

Searching printed illustrations. Early printed books contain illustrations created by stamping an inked wooden block onto paper. Visual search can be used to retrieve other paper sheets containing the same illustrations, which provides useful information to cataloguers, bibliographers and art historians. For example, the condition of the woodblocks can be used to date the sheet relative to other sheets for which the date is known, or the identity of the printer, printing place or relationships between printers can be determined. We have created such a visual retrieval system[1] which is actively used by art historians; example retrieval results are shown in figure 1.3. See (Bergel et al., 2013) for more details about the project.

1.2 Challenges

There are many challenges in large scale object retrieval, here we list a few.

Robustness to changes in appearance. An object retrieval system has to be able to retrieve images of the queried object despite large changes in the object's appearance. The change can be caused by varying lighting conditions, different physical properties of the cameras used to capture the images, digital artifacts (e.g. due to JPEG compression), etc. The object can be rotated, imaged from varying viewpoints and it can appear at different scales. Furthermore, significant occlusions or cropping should be handled, as well as background clutter. Some of these are illustrated in figure 1.4. Changes to the object itself are possible, for example a building façade can deteriorate with time. In the case of early printed books (see the previous section), where "objects" of interest are illustrations created by stamping an inked wooden block onto paper, each "object" has different appearance: the amount of ink can vary, the woodblock can deteriorate, and the paper can be non-uniformly deformed; see figure 1.5.

Handling various object types. Objects of interest are inherently visually different: some objects are best described in terms of their colour, others in terms of

[1]Available on the Bodleian library website at http://ballads.bodleian.ox.ac.uk/ .

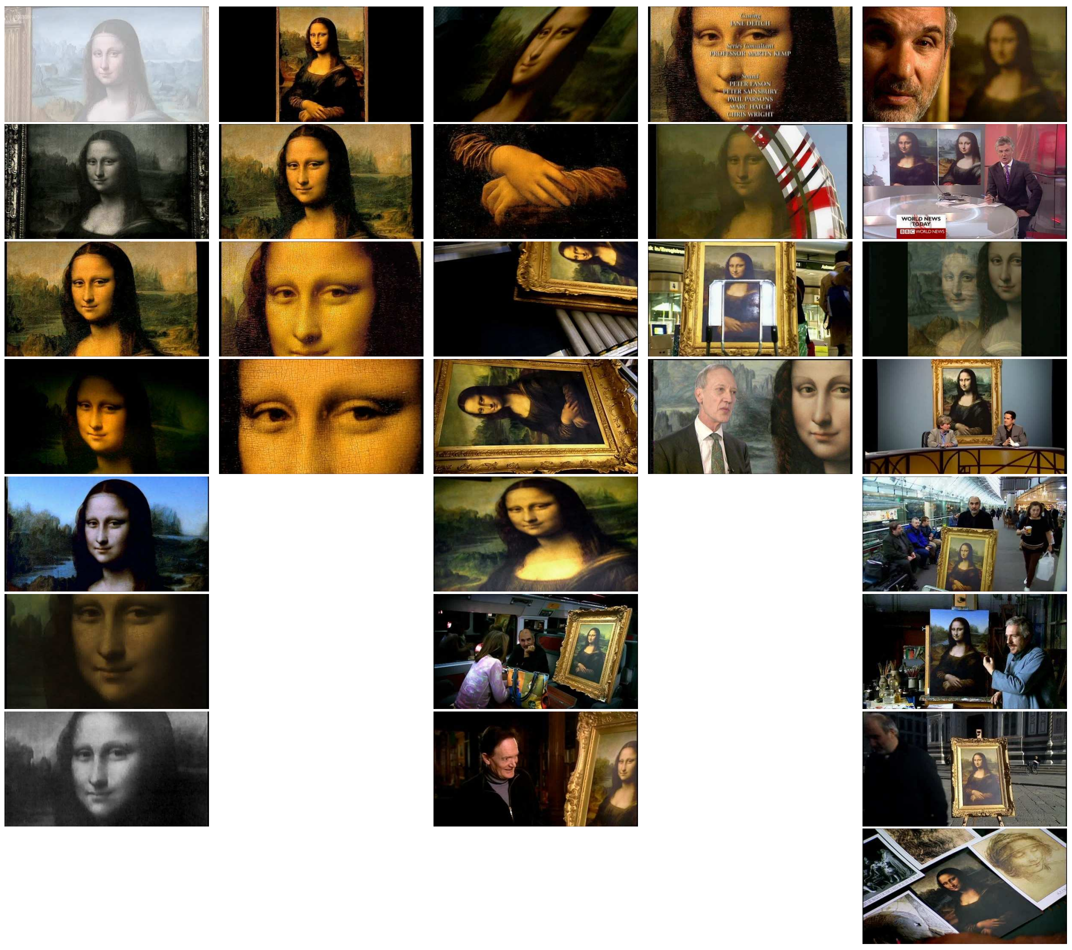


Figure 1.4: **Object retrieval challenges.** Columns show various challenging situations for object retrieval, from left to right: changes in lighting and colour, scale, viewpoint and cropping, occlusions, and background clutter.

their texture, while yet others are best defined by their shape. An ideal retrieval system should be able to handle all types of objects.

Total recall. For some applications it is sufficient to retrieve only a single relevant image to the query (e.g. to answer the "What is this?" question from the previous section). However, in many cases it is required to return several, if not all, images of the queried object; examples include accurate geolocalization, 3D reconstruction,

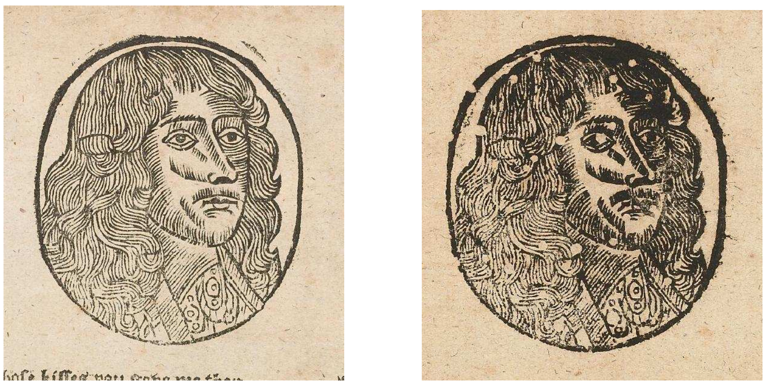


Figure 1.5: **Printed illustration changes.** Early printing was performed by stamping an inked wooden block onto paper. The two images show illustrations created using the same woodblock at different points in time. The amount of ink used is very different, and several wormholes developed before the printing of the right illustration (manifested as small circular patches with no ink).

database consistency checking, etc. While it is often relatively easy to retrieve a single correct image, the task of retrieving all relevant images, while maintaining good precision (i.e. not returning many incorrect results) is considerably harder.

Speed and scalability. Retrieval in a dataset containing millions of images 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. This is a difficult task as it requires efficient recognition algorithms, as well as compact image representation since all required data should fit inside the computer's RAM memory. This is because data transfer rates are orders of magnitude slower for hard disks than for RAM, thus making hard disk access infeasible for the near-real time operation requirement.

1.3 Contributions and thesis outline

In this section, we list some of the key contributions made in this thesis. Contributions 1-3 (chapter 4) are improvements of bag-of-words based object retrieval, although contribution 1 (RootSIFT) is generally applicable across Computer Vision. In chapter 5, contributions 4-6, we propose improvements to compact image representations, namely the VLAD (Jégou et al., 2010b) descriptor, used for very large scale object retrieval, though applicable to classification tasks as well. Chapter 6 (contribution 7) presents a method to boost recall by making use of multiple query images. Retrieval of smooth textureless objects, which are not handled well in current systems, is discussed in chapter 7 (contributions 8 and 9). Finally, a method for very efficient descriptor compression, useful for approximate nearest neighbour search which is ubiquitous in object retrieval, is presented in chapter 8 (contribution 10). The ten key contributions of this thesis are discussed next.

1. RootSIFT. We introduce a new method to compare SIFT descriptors (Lowe, 2004) which yields superior performance without increasing processing or storage requirements (section 4.2). SIFT can be effortlessly replaced by RootSIFT in all Computer Vision systems.

2. Discriminative query expansion. 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 (section 4.3). This is the first time that discriminative learning methods have been employed in the area of large scale retrieval.

3. Database-side feature augmentation. Turcot and Lowe (2009) propose a method which augments database images with visual information from related images. We improve on it by taking into account the visibility of the augmenting features in the augmented image (section 4.4).

4. Intra-normalization of VLAD descriptors. New normalization scheme for VLAD that addresses the problem of burstiness (Jégou et al., 2009b), where a few large components of the VLAD vector can adversely dominate the similarity computed between VLADs (section 5.3). The new normalization is simple, and always improves retrieval performance.

5. Vocabulary adaptation for VLAD descriptors. Retrieval performance suffers in the case where the cluster centres used for VLAD are not consistent with the dataset – for example they were obtained on a different dataset or because new images have been added to the dataset. In section 5.2 we propose an efficient, simple, method for improving VLAD descriptors via vocabulary adaptation, without the need to store or recompute any local descriptors in the image database.

6. Extraction of multiple VLAD descriptors per image. In section 5.4 we improve retrieval performance for small objects by extracting multiple VLAD descriptors per image. Furthermore, we propose a method of sub-VLAD localization where the window corresponding to the object instance is estimated at a finer resolution than the VLAD tiling.

7. Use of multiple query images. Current systems, for example Google Goggles, concentrate on querying using a single view of an object, e.g. a photo a user takes with his mobile phone, in order to answer the question “what is this?”. In chapter 6 we consider the somewhat converse problem of finding *all* images of an object given that the user knows what he is looking for; so the input modality is text, not an image. Given a textual query (e.g. "coca cola bottle"), our approach is to first obtain multiple images of the queried object using textual Google image search. These images are then used to visually query the target database to discover images containing the object of interest.

8. Bag-of-Boundaries. Standard retrieval systems are quite successful in recognizing a wide variety of objects, however they all assume that objects have a light coating of texture. In chapter 7 we describe a scalable approach to smooth object retrieval by representing objects via a set of semi-local boundary descriptors.

9. Sculpture naming. In section 7.5 we describe a retrieval based method, namely a combination of bag-of-words and bag-of-boundaries, for automatically determining the title and sculptor of an imaged sculpture.

10. Set compression tree. We introduce a novel encoding method which is able to accurately compress 1 million descriptors using only a few bits per descriptor. The large compression rate is achieved by not compressing on a per-descriptor basis, but instead by compressing the set of descriptors jointly. In chapter 8 we describe the encoding, decoding and use for nearest neighbour search, all of which are quite straightforward to implement. The method, tested on standard benchmarks, achieves superior performance to a number of state-of-the-art approaches.

1.4 Publications

Contributions 1–3 were presented in CVPR 2012 (Arandjelović and Zisserman, 2012a). Improvements to VLAD (contributions 4–6) were presented in CVPR 2013 (Arandjelović and Zisserman, 2013a). The work employing multiple query images (contribution 7) was presented in BMVC 2012 (Arandjelović and Zisserman, 2012c). Smooth object retrieval work (contribution 8) was presented in ICCV 2011 (Arandjelović and Zisserman, 2011), while the sculpture naming work (contribution 9) was presented and nominated for the best paper at ICMR 2012 (Arandjelović and Zisserman, 2012b). Set compression tree (contribution 10) was published as a technical report in 2013 (Arandjelović and Zisserman, 2013b) and is under review for PAMI.

Other papers published during the course of this PhD include: (i) work on the TRECVid project which uses contributions 1 and 7 – TRECVid 2011 (McGuinness et al., 2011), TRECVid 2012 (Aly et al., 2012b), TRECVid 2013 (Aly et al., 2013) and ICMR 2013 (McGuinness et al., 2013); (ii) work on visual retrieval systems helpful to researchers in humanities (Bergel et al., 2013, Rahtz et al., 2011); (iii) work on efficient epipolar geometry estimation used for spatial reranking, published in BMVC 2010 (Arandjelović and Zisserman, 2010), but excluded from this thesis due to lack of space.

Chapter 2

Literature Review

In this chapter we review development of large scale object retrieval. All methods are required to provide good retrieval performance while executing queries in near real-time. The demand for fast retrieval typically imposes that all data has to be stored in RAM since hard disk access is too slow, which in turn restricts storage requirements as RAM is a relatively scarce resource. The retrieval performance is assessed in terms of precision and recall (section 3.2). Some approaches focus on high precision, namely high quality of top retrieved results, while other aim at large recall, i.e. retrieving all relevant results.

Section 2.1 provides an overview of methods which store some information for each local image descriptor of every image in the database. These methods are capable of searching a few million images on a single high-end computer in near real-time.

When scaling up to even larger databases, containing tens of millions or even billions of images, one is left with two choices: (i) distribute the computation and storage across many computers, thereby increasing the cost of the system, or (ii) stop storing data on a per-descriptor level and use a very compact global image representation. Methods which employ the latter solution are reviewed in section 2.2.

2.1 Retrieval by matching local descriptors

The question that naturally arises in all areas of Computer Vision is how to represent an image in a computer. The representation has to be robust to large changes in imaging conditions, scale changes and differences in camera viewpoint, as well as significant occlusion. It is beyond the scope of this thesis to review the large body of work investigating various low-level image representations, so we choose only to provide a very brief description of the standard techniques widely adopted by the Computer Vision community.

In the presence of occlusion, background clutter and changes in scale and camera viewpoint, two images containing the same object can contain large regions which cannot be found in both images. The standard approach to account for this difficulty is to extract many local patches from an image which are then used as a proxy for image representation. This procedure relies on the assumption that two similar images will share a significant amount of local patches which can be matched against each other. The local patches are typically extracted in one of two ways: (i) on a dense grid and at various scales (to provide scale invariance), or (ii) from regions obtained from a detector, engineered to provide sparse regions focusing on “interesting” parts of the image and ensuring scale invariance at the detection stage. The dense method is not applicable for large scale retrieval when information is stored on a per-patch level since it extracts orders of magnitude more patches than sparse methods; therefore sparse patch extraction is used. Popular region detectors include Difference of Gaussians, DoG (Lowe, 2004), Maximally Stable Extremal Regions, MSER (Matas et al., 2002) and affine invariant detectors (Mikolajczyk and Schmid, 2004b, Schaffalitzky and Zisserman, 2002), as well as very fast detectors designed to be used in real-time, like SURF (Bay et al., 2006), FAST (Rosten et al., 2010) and BRISK (Leutenegger et al., 2011).

Deciding if two images contain the same object is based on how well the sets of extracted image patches match. It is necessary to be able to quantify the similarity between patches in order to measure the similarity between patch sets, and this is done via computing similarities, e.g. scalar products of patch descriptors. As raw pixel colour or intensity is clearly not robust to changes in imaging conditions, the standard approach is to engineer descriptors that are by design robust to such changes, as well as being discriminative enough to distinguish between different patches. After the seminal work of Lowe (2004) which introduced the very effective SIFT descriptor, local descriptors generally describe the spatial distribution of pixel intensity gradients in a patch. Similar approaches include GLOH (Mikolajczyk and Schmid, 2004a), SURF (Bay et al., 2006), DAISY (Tola et al., 2008), CONGAS (Zheng et al., 2009), BRIEF (Calonder et al., 2010), as well as RootSIFT, our own improvement of SIFT (see section 4.2). Histogram of Oriented Gradients, HOG (Dalal and Triggs, 2005), also has a similar flavour but is mostly used in object detection as a representation of an entire object or its part. Building on top of gradient-based descriptors, a few works proposed learning discriminative local descriptors (Brown et al., 2011, Lepetit et al., 2005, Philbin et al., 2010, Simonyan et al., 2012).

Given sets of local descriptors extracted from all database images and the query image, large scale object retrieval reduces to efficient matching of the query descriptor set to the database descriptor sets. The following sections provide an overview of the choices for the similarity metric between the descriptor sets, as well as computationally and memory efficient algorithms which enable real-time ranking of database images according to the chosen similarity. Section 2.1.1 presents an approach inspired by successes in large scale text retrieval, while methods in section 2.1.2 rank images based on precise descriptor matching. In section 2.1.3 the spatial configuration of local descriptors is exploited, while section 2.1.4 considers expanding the descriptor sets with ones of related images.

2.1.1 Bag-of-Words: A text retrieval motivated approach

The first effective and scalable approaches to object retrieval, initiated by the "Video Google" work of Sivic and Zisserman (2003), but still very popular, were inspired by the success of text retrieval, where current web search engines, like Google and Bing, are capable of instantly searching billions of web pages. This section discusses text retrieval motivated methods for image search, starting with a brief introduction to text retrieval, followed by its application to image search and further developments.

2.1.1.1 Text retrieval

A text document is represented in a computer using the vector space model (Salton and McGill, 1986), also known as "bag-of-words", BOW (Manning et al., 2008). Namely, each document is regarded as an unordered collection (a "bag") of words and represented as N_w-dimensional histogram of word occurrences, where N_w is the number of words in a language. This scheme is called *term frequency weighting* as the value of each histogram bin is equal to the number of times the word appears in the document.

As some words, like *the* and *and*, naturally occur more often than others, they are less informative than infrequent words. To account for the imbalanced word frequencies, dimensions (i.e. words) in the vector space model are weighted based on how informative they are. The commonly used weighting is the *inverse document frequency*, where information is quantified as $log\frac{N_D}{N_i}$, where N_D is the size of the document corpus and N_i is the number of documents in which word i appears. The overall bag-of-words representation is thus weighted by multiplying the term frequency (tf) with the inverse document frequency (idf) giving rise to the tf-idf weighting (Manning et al., 2008). Extremely frequent words, “stop words”, can be removed entirely in order to reduce storage requirements and query time.

Similarity between documents is computed as the cosine similarity between their tf-idf weighted bag-of-words representations, which naturally normalizes for document length. Documents typically contain only a small subset of all available words in a language, making the representation very sparse. For efficient storage and retrieval, it is beneficial to precompute a data structure called the *inverted index* (Manning et al., 2008), which contains a *posting list* for each word; a posting list records all documents containing a particular word along with the respective term frequencies. The list of documents which contain query words, and thus have a non-zero similarity with the query, is quickly obtained by traversing the relevant posting lists, while the similarities are efficiently computed by accumulating the products between the normalized tf-idf weights of database documents and the query tf-idf.

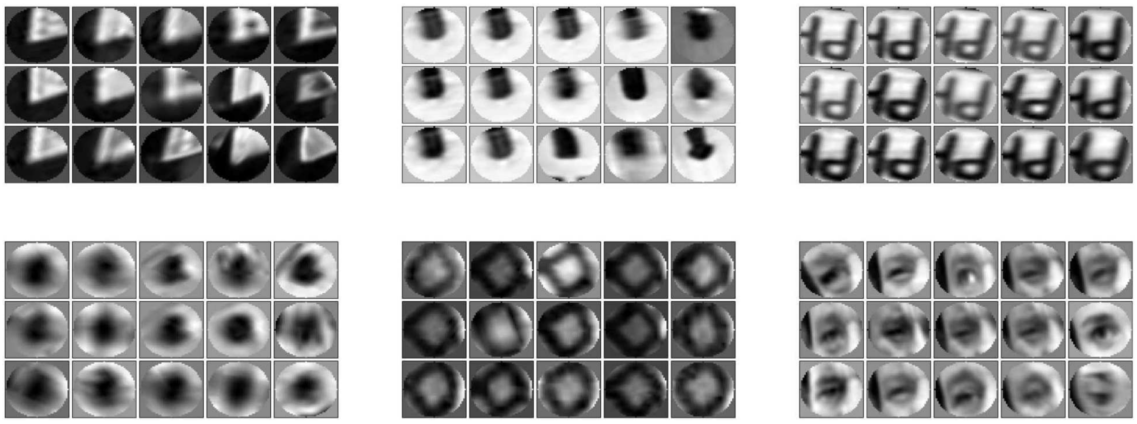


Figure 2.1: **Visual words.** Each block shows normalized patches assigned to the same visual word. Taken from (Sivic and Zisserman, 2009).

2.1.1.2 Visual word based image retrieval

The work of Sivic and Zisserman (2003) is the first one to apply text retrieval ideas to image search[1]. A major obstacle for applying text retrieval framework to image retrieval is the fact that text documents are naturally broken into words, while no such natural segmentation exists for images. However, as discussed earlier in this chapter, a set of local descriptors can be extracted from an image. Sivic and Zisserman (2003) introduce the concept of "visual words" where local descriptors are vector quantized into a predefined "visual vocabulary" obtained using k-means; examples of visual words are shown in figure 2.1. Images are represented with sparse, tf-idf weighted, bag-of(-visual)-words histograms and object retrieval is performed efficiently through the use of an inverted index. They demonstrate sub-second retrieval in a 4000 image database and a 10k visual vocabulary.

[1]More strictly, the paper was actually considering the problem of visual video search, however the problem was cast as a visual image search problem by extracting one frame of a video per second.

Nister and Stewenius (2006) followed by Philbin et al. (2007) demonstrate that using a large visual vocabulary is very beneficial for specific object retrieval as it yields less false positive matches for a query descriptor than when using a small vocabulary. Furthermore, a large vocabulary also improves the search speed as the bag-of-words histograms become sparser. In contrast to a 10k vocabulary in (Sivic and Zisserman, 2003), Nister and Stewenius (2006) and Philbin et al. (2007) use a 1M-16M vocabulary. The computational complexity of k-means is $O(N_w N_d)$, where N_w and N_d are the sizes of the visual vocabulary and the training descriptor set respectively; note that $N_d \geq N_w$ is required and thus the complexity is larger than $O(N_w^2)$. Therefore, it is intractable to compute exact k-means for the vocabulary sizes suggested by (Nister and Stewenius, 2006, Philbin et al., 2007). Nister and Stewenius (2006) tackle this problem by building a vocabulary tree, which is essentially a hierarchical k-means (HKM) algorithm, and demonstrate impressive sub-second search in a 1M dataset of images. (Philbin et al., 2007) show that using an approximate k-means (AKM) algorithm to build the vocabulary, namely k-means with approximate nearest neighbour search by using randomized k-d trees, significantly outperforms hierarchical k-means due to decreased quantization effects. The complexity of both methods is $O(N_d log(N_w))$ making them appropriate for building large visual vocabularies.

2.1.1.3 Alleviating quantization problems

Increasing the vocabulary size increases the distinctiveness of a visual word but it also decreases its repeatability, as slightly different descriptors can be assigned to different visual words thus contributing zero votes to the similarity of the respective images. This is particularly the case when the vocabulary is created on an independent dataset from the target one, causing a drop in performance observed in (Nister and Stewenius, 2006, Philbin et al., 2008, Schindler et al., 2007). Schindler et al. (2007) propose to always build the vocabulary on the target set and in fact construct it from a subset of training descriptors which are found to be most informative. However, this approach does not solve the adverse quantization effects, nor is it always applicable as image databases can grow with time thus requiring periodical rebuilding of the vocabulary and re-indexing of the entire database. Philbin et al. (2008) propose to "soft assign" each descriptor to multiple nearest visual words, as opposed to a "hard assignment" to the nearest visual word, where the words are weighted by how close they are to the descriptor. Though very effective, this method significantly increases storage requirements and search time. Jégou et al. (2010a) propose to instead just soft assign query descriptors thus not increasing storage requirements at all compared to the hard assignment baseline, while increasing the search time less than (Philbin et al., 2008). The method of Nister and Stewenius (2006) has the flavour of soft assignment as it is capable of assigning non-zero weights to more than one visual word, namely the words which have common ancestors in the vocabulary tree with the query word. However, the weights are independent of their closeness to the query descriptor. Torii et al. (2013) detect repeated patterns in an image and propose to hard assign the member descriptors, as their number assures fair assignment ("natural" soft assignment) into visual words representative of the pattern; descriptors which are not part of any repeated structure are soft assigned as in (Philbin et al., 2008).

Mikulik et al. (2010) and Makadia (2010) propose to learn similarities between visual words. These methods automatically gather training data consisting of sets of matched descriptors in a large image database: Mikulik et al. (2010) mine the database for similar descriptors which pass geometric verification (see section 2.1.3), while Makadia (2010) track features in adjacent Google Street View images. Two words are deemed similar if they often co-occur in the training set, signifying they are often used to describe the same underlying physical data (i.e. they are "visual synonyms"). The estimated similarities are then used at retrieval time by effectively expanding the set of query words with their visually related words. This scheme does not take into account the location of the descriptor inside a visual word Voronoi cell, unlike in (Philbin et al., 2008), but it does alleviate quantization errors when similar descriptors are assigned to different words. Note that Philbin et al. (2008) attempt a related scheme where the set of words extracted from image patches is expanded by words produced by jittering the patches, somewhat like ASIFT (Morel and Yu, 2009). However, their results are inferior probably due to the fact that the image patches are artificially created while (Mikulik et al., 2010) and (Makadia, 2010) use descriptors extracted from real physically observed patches. A related approach to (Makadia, 2010, Mikulik et al., 2010) is the one of Bergamo et al. (2013), where the training set of matching descriptors is obtained by structure from motion, and the visual vocabulary is constructed to discriminate between various "classes" of features via a random forest. This method lies in the middle of the spectrum of approaches which identify that the original Euclidean distance used to compare local descriptors might not be perfect. At one end of the spectrum, Mikulik et al. (2010) and Makadia (2010) apply their methods after the fact, when a visual vocabulary is already constructed. In the middle, Bergamo et al. (2013) modify vocabulary building to group truly similar descriptors together. Finally, Philbin et al. (2010) and Simonyan et al. (2012) try to learn better descriptors, for which the Euclidean

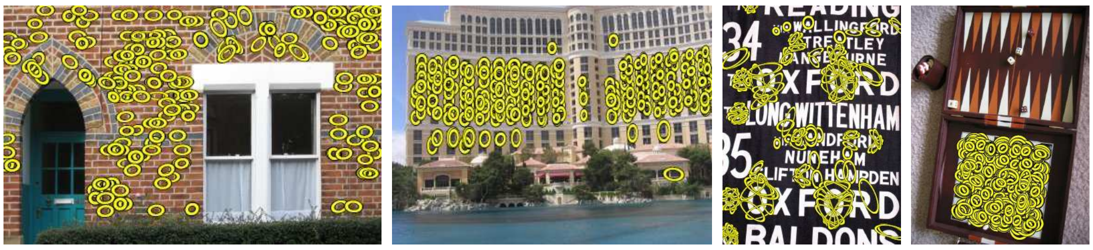


Figure 2.2: **Burstiness.** Instances of the most bursty visual word of each image are displayed. Taken from (Jégou et al., 2009b).

distance is more appropriate, before the vocabulary is built, thus not needing any modification of the standard vocabulary construction nor post-processing.

Another direction for decreasing descriptor quantization effects relies on finer representation of each original descriptor; methods following this strategy are reviewed in section 2.1.2.

2.1.1.4 Modifying the image similarity metric

The commonly used tf-idf measure (Sivic and Zisserman, 2003) of image similarity assumes that observed visual words in an image are independent of each other. However, as pointed out by Jégou et al. (2009b), visual words often appear in bursts – if a visual word appears in an image, it is more likely it will appear again; see figure 2.2. Thus, each instance of a particular word in an image adds progressively less information, which should be accounted for in the representation of an image, and is ignored by basic tf-idf weighting which assumes visual word instances are independent. Jégou et al. (2009b) propose to discount the burstiness effect by square-rooting the term frequency (tf) of each word[2]. Square-rooting can also be viewed as a first

[2]The actual discounting strategy also depends on the strength of the match which is measured using Hamming embedding, discussed in section 2.1.2. For clarity, here we discuss only the method

order approximation to the intersection kernel (Vedaldi and Zisserman, 2011), which would measure the maximal possible number of word matches between two images. For example, if two images contain 2 and 3 instances of a particular visual word, the similarity contribution of the corresponding dimension in the BoW histogram would be $2 * 3 = 6$ (discarding normalization and idf for clarity). The intersection kernel would model the fact that the maximal possible number of matches is $min(2, 3) = 2$, while the square-rooting yields a score of $\sqrt{2} * \sqrt{3} = 2.45$.

Doubek et al. (2010) show that burstiness or, more specifically, repeated patterns can in fact be used to describe an image. They detect repeated patterns and create a descriptor of the repeating element which is then used to retrieve similar images. This is particularly useful for man-made objects like façades of modern buildings. Furthermore, as discussed in section 2.1.1.3, repeated structures can be used as a way to naturally soft assign descriptors to visual words (Torii et al., 2013).

Apart from the burstiness effect which considers cases when a particular visual word is repeated in an image, the tf-idf word independence assumption is also invalidated when sets of words often appear together. The argument towards modifying tf-idf similarity metric to account for co-occurring words is analogous to the one for burstiness: given visual word A appears in an image, if visual word B frequently co-occurs with A and is observed in the image, its information content is small. Chum and Matas (2010b) propose a scalable method for discovering frequently co-occurring visual words by using MinHash (Broder, 1997), and modifying the scoring function to discount them. Cummins and Newman (2008) build a probabilistic model for location recognition such that dependencies between visual words are accounted for using the Chow Liu algorithm (Chow and Liu, 1968).

which does not use Hamming embedding.

Jégou and Chum (2012) note that the tf-idf similarity of two images only takes into account visual words which appear in both of them, while the information that the both images *do not contain* some visual words is almost discarded. They propose to modify the BoW vectors by deducting a fraction of their mean (the mean is taken across the image corpus), which increases the similarity (scalar product between BoW vectors) which contain co-missing words.

As discussed in section 2.1.1.1, the inverse document frequency (idf) for a visual word i is computed as $log\frac{N_D}{N_i}$, where N_D is the size of the document corpus and N_i is the number of documents (images) in which word i appears. The computation of N_i can be expressed as $\sum_d N_{i,d}^p$ where $N_{i,d}$ is the number of times word i appears in document d, and $p = 0$ (defining $0^0 = 0$). Zheng et al. (2013) propose to use other values of p as this can provide a finer measure of the information contained in each word. For example, a word which appears in every image of the database 10 times should have a smaller weight than a word which appears in every image once; under the standard ($p = 0$) idf the two words have the same weight while this is not the case with $p > 0$.

Jégou et al. (2007) and Qin et al. (2011) argue that the ranking of database images should take into account the local density of BoW vectors in the tf-idf vector space. For example, let us consider the case of querying with image Q and deciding how to rank images A and B which are both equally distant from Q under the tf-idf measure. If the local density of BoW vectors is large around A, meaning that many database images are similar to it, while the density is small around B, it means that it is “harder” to be similar to B than to A. Thus, it should be more significant that B is similar to Q than that A is, and therefore B should be ranked before A, despite having equal tf-idf scores. Jégou et al. (2007) propose to adjust local distance measures for each image such that the average distance to its local neighbours is equal to a database-wide constant, thus normalizing local BoW density. In the above example, distance between Q and B would be decreased relative to the distance between Q and A. On the other hand, Qin et al. (2011) rank highly a "close" set of images which is an expanded set of k-reciprocal nearest neighbours of the query; for the above example it is likely that B would be in the close set while A would not as Q would not be in the set of A's k-nearest neighbours.

None of the methods reviewed so far take negative data into account, which can be used to discriminate between good and bad retrievals. Our work in section 4.3 is the first one to employ discriminative learning for object retrieval; next we describe methods which have the same motivation. For location recognition, Gronat et al. (2013) train a linear classifier for each image in the database in the manner of Exemplar SVMs (Malisiewicz et al., 2011). To train a classifier for a single database image, "hard negatives" are mined through the use of GPS information associated with database images. Retrieval is then performed by ranking database images based on the calibrated classification scores obtained by applying every classifier to the query image. The paper reports 2 s query time for a database with 25k images as all database classifiers are dense and thus brute force computation of scalar products is required. However, it is easy to make the approach as scalable as the standard bag-of-words retrieval system by using the dual form representation of the SVM classifier and efficiently computing scalar products between the query BoW and support vector BoWs, which are sparse, using an inverted index[3]. Cao and Snavely (2013) apply a similar strategy to (Gronat et al., 2013), also for location

[3]The suggestion was made by this author

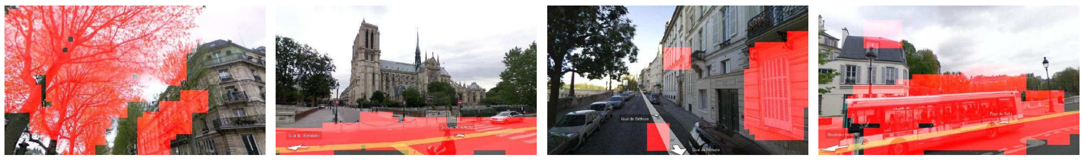


Figure 2.3: **Confusing regions.** Examples of detected confusing regions by Knopp et al. (2010).

recognition, where instead of GPS information they use an automatically created image graph to discover “hard negatives”.

2.1.1.5 Feature selection

Some features are more confusing than others and identifying these can improve retrieval performance, as well as reducing storage requirements if they are removed from the index. Knopp et al. (2010) identify confusing image regions in every database image and remove all features from it. For a database image, the confusing regions are detected by querying with it and discovering spatially localized groups of features which are responsible for a large number of false matches. Examples of detected confusing regions are shown in figure 2.3. The procedure assumes availability of labelled training data, specifically the work uses GPS information like in Gronat et al. (2013). Training a classifier which automatically weights individual features according to their discriminative power can be viewed as “soft” feature selection, and indeed the method of Gronat et al. (2013) (section 2.1.1.4) outperforms (Knopp et al., 2010).

On the other hand, Turcot and Lowe (2009) propose to discover useful features instead of confusing ones. A feature is deemed to be useful if it is matched with high confidence (using spatial verification, section 2.1.3) to any other feature in the database; all non-useful features are removed. Although the method presents good results on a standard retrieval benchmark, it suffers from inability to retrieve objects which only appear in one or very few images in the database, as no relevant feature will survive this overly aggressive removal strategy.

Kang et al. (2011) measure descriptor distinctiveness based on estimating the local intrinsic descriptor dimensionality. A distinctive visual vocabulary is built by assigning larger weights to distinctive features, and non-distinctive descriptors are removed from the database and the query.

If the image database is actually constructed by sampling frames from videos, features can be tracked across frames and those features which do not appear in enough consecutive frames are deemed to be unstable and are not added to the database (Sivic and Zisserman, 2003). The descriptors of stable features are averaged to increase the signal to noise ratio.

2.1.1.6 Near duplicate image detection

It is often useful to discover near duplicate images in a large image corpus. For example, it can be used for copyright infringement detection, reduction of storage requirements by removing redundant information from the corpus, or diversification of retrieval results making them more pleasing to a user. Experiments of Wang et al. (2013) show that nearly 30% of images on the internet have duplicates.

Chum et al. (2007a) define near duplicate images as images which have many common visual words. Images are thus represented as sets of visual words, i.e. word counts are discarded, and the measure of image similarity is the Jaccard index: the similarity of sets A_1 and A_2 is $sim(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}$. Fast estimation of set similarity is performed using MinHash (Broder, 1997). The algorithm computes a “sketch” (n-tuple of hashes) for each image such that the probability of “sketch collision” for two images, i.e. all hashes in the two sketches are identical, is equal to $sim(A_1, A_2)^n$. By generating several sketches per image and grouping together images with a significant number of sketch collisions, accurate and efficient near duplicate image detection can be performed. As an extension of this work, (Chum et al., 2008) propose a way of incorporating the tf-idf weights in the MinHash algorithm. A fast method of generating MinHash sketches by using the inverted index is presented by Chum et al. (2008). Zhao et al. (2013b) propose to use, in addition to sketch collisions, information about matching local descriptors to compute a finer measure of image similarity, much like methods in section 2.1.2.

Approaches which use global image features exist (Liu and Rosenberg, 2008, Wang et al., 2013) but are limited to detecting only almost exact duplicates, with no tolerance to cropping or rotation. Methods which employ compact image representations (reviewed in section 2.2) are also successfully used for near duplicate detection (Douze et al., 2010, Li et al., 2008).

2.1.1.7 Pushing the boundaries of scalability

When discussing large scale retrieval it is important to ask the question – just how scalable are bag-of-words based approaches? In our work we have successfully run a BoW-based retrieval system on a single server which searches the entire dataset of 3.5 million images in less than a second (including spatial reranking, section 2.1.3). Although this performance is quite impressive, the RAM and CPU requirements scale linearly with the number of features in the database. For every feature in the dataset, the inverted index has to store the identifier of the image (imageID) the feature was extracted from, encoded as a 32-bit number. For a database of 3M images, assuming 1k visual words per image, just storing the bare inverted index takes 12 GB of RAM. Thus, it is not possible to use the same BoW architecture in order to search a 100 times larger dataset, as it is very expensive, if not impossible, to have a single 1.2 TB RAM machine. In addition to the imageID, retrieval systems also typically store some additional information on a per-feature basis, most commonly the geometry of the local feature (e.g. its location in the image, scale or ellipse parameters if affine covariant detectors, like the ones of Mikolajczyk and Schmid (2004b), are used) which is required for spatial reranking (section 2.1.3).

Again borrowing ideas from text retrieval, it is possible to compress the inverted index (Zhang et al., 2008, Zobel and Moffat, 2006), for example by compressing sorted imageIDs in posting lists using run-length coding; Jégou et al. (2009a) report a four-fold reduction in index with this technique. Perd'och et al. (2009) present a method for efficient lossy compression of local feature geometry, reporting a compression ratio of four.

The retrieval process can be parallelized easily by distributing the storage of posting lists across multiple machines. Stewenius et al. (2012) demonstrate an impressive system capable of searching 94 billion images using a “large” (unspecified) number of computers. The system scores images in less than 3 seconds, where the bottleneck, which takes 2 s, is the transmission of relevant posting lists over the network.

2.1.2 Accurate large-scale descriptor matching

As noted by work reviewed in section 2.1.1.3, for good retrieval performance it is crucial to have a very large visual vocabulary (Nister and Stewenius, 2006, Philbin et al., 2007), thus having a precise representation of the original descriptors which enables high accuracy matching between query and database descriptors. However, using a large vocabulary sacrifices recall of descriptors relevant to the query due to quantization errors, which is often addressed by soft assignment of descriptors into visual words (Jégou et al., 2010a, Makadia, 2010, Mikulik et al., 2010, Philbin et al., 2008). This discussion suggests that the standard bag-of-words retrieval system with a large visual vocabulary can be viewed in a different light – it is essentially a system which performs approximate nearest neighbour (ANN) search for every query descriptor. The ANN search strategy is to assign a query descriptor to the nearest cluster ("visual word") and deem all database descriptors which fall into the same cluster as nearest neighbours of the query. There is no way of discriminating between the retrieved descriptors as all the information that is kept about a descriptor is the assigned visual word, or multiple words in the case of soft assignment, so matches are just weighted using idf based on how uncommon they are. With these observations in mind, a multitude of methods, which will be reviewed next, aim at employing a better ANN search strategy in order to accurately match all query descriptors against database descriptors, under the constraints of near real-time operation and reasonable memory consumption. Note that all these methods consume more memory than BoW-based methods as it is necessary for them to store at least all the information BoW methods store (e.g. imageID and geometry information, see section 2.1.1.7)

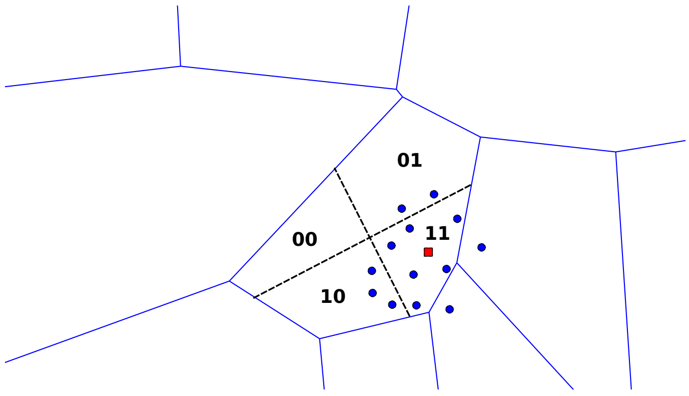


Figure 2.4: **Hamming embedding.** Voronoi cells associated with each visual word are shown in blue. The binary strings correspond to the Hamming embedding associated with each subspace, the square and circles represent the query descriptor and its nearest neighbours, respectively.

Lowe (2004) presented an early approach to object retrieval via accurate descriptor matching. The work uses a k-d tree for ANN search for query descriptors and demonstrates (for the time) impressive results on a database of 32 images containing 40k local descriptors. However, this approach is not scalable as it increases memory usage by 128 bytes (size of the typical representation of a SIFT descriptor) per local descriptor; just storing all descriptors in a database of 3 million images would thus take 384 GB, compared to a total of 12 GB required by a BoW system (section 2.1.1.7). If one is willing to use thousands of machines to store the required data in RAM and respond to user queries, Aly et al. (2011) show a method of parallelizing k-d tree construction and nearest neighbour search for querying in a dataset of 100 million images on 2k computers.

Jégou et al. (2008) propose to use a similar architecture to BoW-based systems, with the addition of a compact binary signature associated with every descriptor in the posting list. This signature, termed Hamming embedding, is used to disambiguate BoW matches by requiring that the Hamming distance between the query and matched signatures is small enough. The signature is computed by performing random orthogonal projections of the descriptor and encoding whether the projected value is smaller or larger than the median for the corresponding visual word, thus essentially describing which quadrant of visual word's Voronoi cell the descriptor falls in (figure 2.4). Hamming embedding in a typical setup only adds 8 bytes per descriptor, while Hamming distance computation can be done very efficiently with dedicated instructions on modern CPUs. This approach has been successfully applied to an image localization task by Sattler et al. (2012).

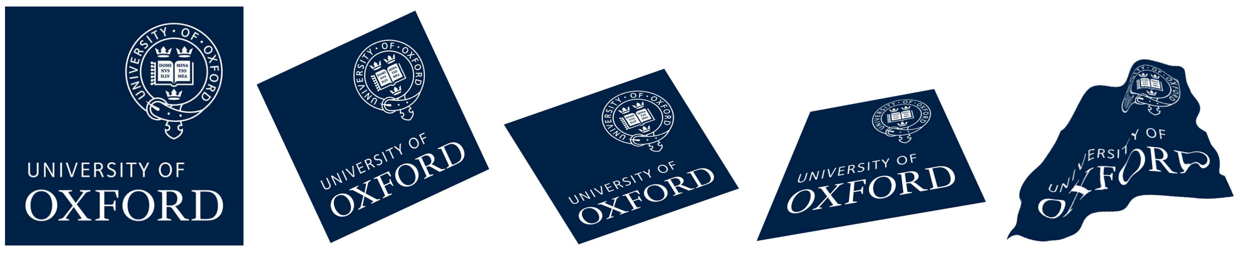
The work of Jégou et al. (2011a) proposes a fast memory-efficient method for ANN search similar to their preceding Hamming embedding (HE) work. The HE binary signature is replaced with a compressed version of the descriptor using product quantization (PQ), a state-of-the-art vector compression method. PQ compression is performed by splitting vectors into chunks and vector quantizing each chunk independently; Ge et al. (2013) and Norouzi and Fleet (2013) optimize the PQ performance by pre-rotating vectors into a space where PQ is most effective. ANN search is performed by exploring several nearest cluster centres to the query descriptor, analogous to soft assignment into visual words, and computing the distance between the query descriptor and the compressed descriptors in each posting list. Jégou et al. (2011b) apply this ANN search algorithm to object retrieval by finding approximate k-nearest neighbours of query descriptors and having them vote for their respective images; several promising scoring strategies based on estimates of matched descriptor distances are investigated. The work also proposes the use of k-reciprocal nearest neighbours for more precise descriptor matching in order to account for varying density of the descriptor space, much like methods reviewed in section 2.1.1.4 (Jégou et al., 2007, Qin et al., 2011) but applied on local descriptors and not BoW vectors. Qin et al. (2013) employ the same ANN search procedure but matched features cast a probabilistic vote for the corresponding images, where the probability that two features are correctly matched given their distance is estimated from a training set of matching and non-matching descriptors.

An alternative ANN search method is proposed by Babenko and Lempitsky (2012), where coarse clusters are defined by product quantization, instead of vector quantization used by Jégou et al. (2011a). An efficient procedure for prioritized exploration of the clusters is employed, followed by, like in (Jégou et al., 2011a), fine vector comparison using product quantization.

Aly et al. (2012a) use a k-d tree to perform ANN search, but unlike (Lowe, 2004), where all the original SIFT descriptors are kept, they only store compact descriptor signatures, making the method scalable. Object recognition is performed by finding the nearest neighbour of every query feature and voting for the corresponding images. The method demonstrates good performance but, as it only fetches a single nearest neighbour for each query descriptor, it is only useful for applications where one is interested in the very top retrieval result, without attempting to retrieve all images containing the queried object; an example of such an application is Google Goggles.

2.1.3 Enforcing spacial consistency

In both previous sections 2.1.2 and 2.1.1 we have purely been concentrating on image representations and similarity measures which regard an image as a collection of local descriptors, potentially quantized into visual words. However, an image is not just an unordered set of patches, and, as noted in the original work introducing



(a) Original image (b) Similarity (c) Affine (d) Projective (e) Non-rigid

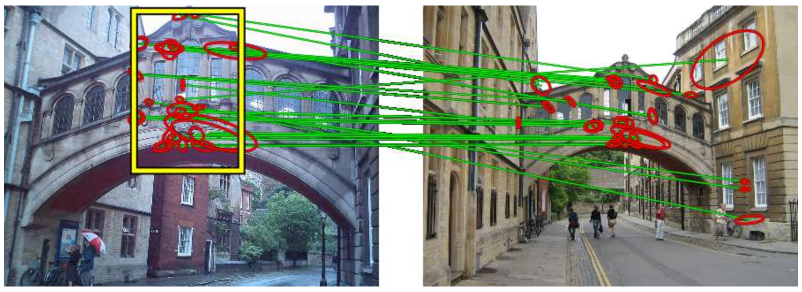
Figure 2.5: **Geometric transformations.** (a) The original image. (b) Similarity transformation (translation, rotation, isotropic scaling): relative orientations of lines are unchanged, a square is imaged as a square. (c) Affine transformation (similarity + non-isotropic scaling and sheer): parallel lines remain parallel, square is imaged as a parallelepiped. (d) Projective transformation (affine + perspective): straight lines remain straight, square is imaged as a quadrilateral. (e) Non-rigid transformation: free form transformation of the original image.

bag-of-visual-words (Sivic and Zisserman, 2003), making use of spatial information can be very beneficial for improving retrieval performance. Note that in order to be able to use spatial information, the retrieval system needs to store additional data on a per-feature basis, namely the location of the feature in the image and usually the scale or the shape of the local region; Perďoch et al. (2009) show how to efficiently compress this information.

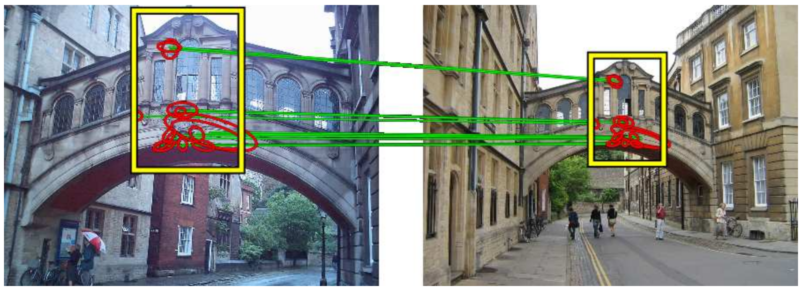
Retrieval quality can be improved by enforcing spatial consistency between the query and a retrieved database image. Here we provide a brief overview of various spacial relationships between images depicting the same object, full details are available in (Hartley and Zisserman, 2004). If the imaged object is rigid and planar, two images of the object are related by a projective transformation (figures 2.5a-2.5d); the hierarchy of projective transformations, starting from the simplest followed by more expressive ones, is discussed next. Similarity transformation captures translation, rotation and isotropic scaling (figure 2.5b) and can be computed from two point correspondences. Affine transformation, which captures similarity, also allows for non-isotropic scaling (figure 2.5c) and is fully specified with three point correspondences. Finally, a general projective transformation also allows for perspective effects (figure 2.5d) and can be computed from four point correspondences. Apart from modelling relationship between images of planar objects, projective transformation also relates images of a rigid scene taken from the same point in space (i.e. stationary camera allowed to rotate in 3D and zoom). Images of the same rigid scene taken from different locations are related via epipolar geometry, which can be computed from seven point correspondences. Finally, images of non-rigid objects and scenes cannot be characterized with a single global relation, however in reality it is often true that local spatial neighbourhoods are preserved across images (figure 2.5e).

Spatial Reranking. The first set of methods propose to use a standard retrieval algorithm, like for example BoW (Sivic and Zisserman, 2003) or BoW with Hamming embedding (Jégou et al., 2008), and apply a relatively costly reranking step which measures spatial consistency between the query and matched database features only on top retrieved results; the length of this "short-list" is typically 200-1k images. The aim of the spatial reranking is to reorder the short-list so that false matches are moved to its end, thus improving (decreasing) the false positive rate.

The original BoW system by Sivic and Zisserman (2003) measures local spatial consistency of matches between the query and a database image like in (Schmid and Mohr, 1997, Zhang et al., 1995), namely neighbouring descriptor matches reinforce each other by casting votes; matches with no vote are discarded and images are reranked based on the number of surviving votes. The construction of feature



(a) All visual word matches



(b) Spatially verified matches

Figure 2.6: **Spatial verification.** (left) The query image and region of interest are shown on the left; (right) A retrieved database image.

neighbourhood is not scale invariant since it is defined as the nearest 15 features, however in practice the method can handle scale changes due to the very loose requirement that only one of the neighbourhood words needs to support a match. The method can also handle non-rigid deformations, but the voting strategy can be too strict as it penalizes even correctly matched descriptors which are scattered around the image (i.e. local neighbourhood structure is lost) but still in a consistent spatial arrangement with the query.

A rigid affine transformation is fitted very efficiently by Philbin et al. (2007) using RANSAC (Fischler and Bolles, 1981), where only a single pair of matched visual words is used to propose an affine transformation between the query and a database image by exploiting the local shape of affine covariant regions (Mikolajczyk and

Schmid, 2004b, Schaffalitzky and Zisserman, 2002) and assuming images are upright. In fact, the fitting of an affine transformation is so efficient that no random sampling in RANSAC is needed (resulting in "NOSAC") as all transformation proposals can be explored. Reranking is based on the number of inliers with the best affine transformation found by NOSAC, while Philbin et al. (2008) propose a soft voting scheme when soft assignment into visual words is used. Results which have more than a minimal number of inliers are deemed to have passed spatial verification. Figure 2.6 shows an example of a spatially verified pair of images.

Tolias and Avrithis (2011) propose Hough Pyramid Matching for spatial reranking, namely they efficiently group matches which yield consistent similarity transformations, where one pair of matched features proposes a similarity transformation. A hierarchical structure is used to group matches thus resulting in an algorithm which is only linear in the number of putative correspondences, unlike the quadratic complexity of RANSAC used in (Philbin et al., 2007). The algorithm provides an order of magnitude speedup over (Philbin et al., 2007) enabling reranking of a 10 times larger short-list with the same time budget.

Finally, Stewenius et al. (2012) employ a distributed system with a large number of machines to rank their entire 94 billion image dataset based on the number of spatially consistent visual word matches with the query, thus effectively extending the short-list for reranking to the entire dataset.

Spatial Ranking. Previously reviewed works concentrated on reranking a fixed number of top results obtained with a standard retrieval method; the length of the short-list is typically constant due to a a fixed upper limit on acceptable time a query can take. Thus, as image databases increase in size spatial verification is expected to visit diminishing portions of the dataset, and therefore be decreasingly effective. The next set of methods propose to incorporate spatial information at the original ranking stage, thus alleviating this problem.

Jégou et al. (2008) propose to rank images based on weak geometric consistency (WGC), namely, instead of accumulating tf-idf scores on an image level, they are accumulated in a Hough-like voting scheme in the space of similarity transformations between the query and a database image. For memory and CPU efficiency, matched descriptors vote independently for scale and rotation components of the transformation, thus only weak geometric consistency is enforced. As such, the method still benefits from applying spatial reranking which measures spatial consistency of matches more strictly.

Zhao et al. (2010), Zhang et al. (2011) and Shen et al. (2012a) assume the only allowed transformation between a query and desired database images is translation and thus perform Hough voting in the space of 2-D translations. Shen et al. (2012a) propose to handle transformations which are more general than the overly simple translation, specifically similarity transform is used, by querying with sets of transformed query images. For example, to handle similarity transforms (i.e. translation, scaling and rotation), the query ROI is scaled and rotated by various predefined scales and rotations, and a separate query is issued for every transformed query ROI. The method is thus much more computationally expensive than the baseline BoW system or WGC, as typically 64 synthesized queries (when using 8 quantization levels for scale and rotation each) need to be issued for every query. This method is applied "in reverse" in a subsequent work (Shen et al., 2012b) to segment out the query object. Namely, the authors consider a dataset of pre-segmented out objects (e.g. products on shopping websites) which at query time cast votes for the location of the query ROI. The segmentation masks of a few top retrieved images are aggregated and used to initialize GrabCut segmentation algorithm (Rother et al., 2004). After the precise segmentation of the object is obtained, the query is re-issued to improve retrieval quality.

Cao et al. (2010) propose to use spatial-bag-of-features where an image is encoded as sets spatial histograms which capture the spatial ordering of visual words under various linear and circular projections. The position of the bin with the largest count is chosen as the origin in order to provide translation invariance, much like the way rotation invariance is commonly ensured for local descriptors, e.g. SIFT (Lowe, 2004). In the case when a large vocabulary is used, most of the spatial histograms contain only one non-empty bin which is chosen as the first bin for translation invariance. As all spatial information is lost in this scenario, it is not clear how the method distinguishes itself from non-spatial BoW.

Other approaches (Chum et al., 2009, Jiang et al., 2012, Lee et al., 2010, Wang et al., 2011, Wu et al., 2009, Zhang et al., 2010) focus on augmenting the local descriptor with a description of the distribution of visual words in the spatial neighbourhood. Retrieval is then performed by matching the augmented descriptors. These methods are limited to detection of near duplicate images (section 2.1.1.6) as they are typically sensitive to feature drop-outs and scale changes.

2.1.4 Query expansion and feature augmentation

Despite all techniques used to improve retrieval quality that have been reviewed thus far, it is still unrealistic to expect even the most advanced algorithm to retrieve all images containing the query object. There are many causes for this: region detection can fail or imprecisely localize the region, features can be missing from the query or database images due to occlusions, patch description is imperfect, descriptor matching is only approximate due to ANN search, information is lost in descriptor quantization and compression, etc. The following set of methods rely on the assumption that the query object appears in more than one database image, which enables sharing of information across related images. The information sharing can alleviate many of the aforementioned problems, for example, if an object is partially occluded in one image it can borrow the occluded descriptors from another image of the same object.

The first set of methods consider query expansion, a standard approach used to boost text retrieval performance (Buckley et al., 1995, Salton and Buckley, 1999), and adapt it to the visual search task. Query expansion is a form of blind relevance feedback, where top spatially verified results ("expansion set") are used to build a richer model of the query object and reissue the improved query (Chum et al., 2007b); see figure 2.7. It is essential that spatial verification is performed in order to obtain only very confident matches to expand the query with, as if any false result is used it is likely to cause topic drift – the inferred model can diverge from the original query. Chum et al. (2007b) compare several query expansion methods, amongst others the now standard average query expansion (AQE) which uses the average tf-idf vector of the query and the expansion set to perform a new query. Other methods are also proposed in the same work, as well as by Shen et al. (2012a), but they involve issuing multiple additional queries, so AQE has become the de facto standard. Chum et al. (2011) extend this work by performing automatic prevention of expansion failure by examining the quality of the expansion set, as well automatically increasing the

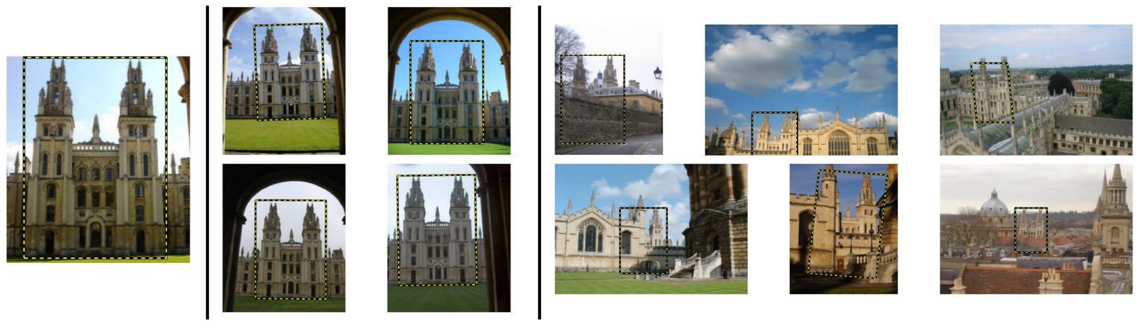


Figure 2.7: **Query expansion.** The image on the left shows the query while the middle four images represent the top retrieved results using BoW. The images on the right show correctly found results after average query expansion, which were not found by the original BoW method. Taken from (Chum et al., 2007b).

query region of interest by inferring the relevant spatial context. Qin et al. (2011) also use a form of query expansion where images which are not in the expansion set ("close set", see section 2.1.1.4) are reranked based on their minimal distance to the set. Tolias and Jégou (2013) present a method which relies on precise matching of query descriptors using Hamming Embedding (section 2.1.2) to form the expansion set, replacing the commonly used, time consuming, spatial verification. This work performs average query expansion on the level of local descriptors instead of the tf-idf vectors used in the original scheme (Chum et al., 2007b).

The second set of methods consider database-side feature augmentation, namely, instead of expanding the query they expand and make better models of the database images. Turcot and Lowe (2009) automatically construct the image graph for the database (Philbin and Zisserman, 2008), where nodes represent images, while edges signify that images contain an object in common. The graph is then used to share information across the nodes – the BoW vector for each image is augmented with the visual word counts of all neighbouring images in the graph. The procedure does not increase storage cost, apart from having to store the image graph which is negligible, as the augmentation does not need to be performed explicitly. Instead, the tf-idf score of the augmented BoW vectors can be computed at run time as the average score of neighbouring non-augmented BoW vectors. Schindler et al. (2007) use a similar strategy for image localization where image scores are averaged across a local neighbourhood, which is defined by physical distance.

Torii et al. (2011) propose a related method where points along image graph edges are retrieved. Every point along an edge is assigned a BoW vector equal to an affine combination of the end-nodes' BoW vectors, based on the distances to the end nodes. None of the point BoWs need to be stored explicitly as the point closest to the query can be found at run time by scoring only the nodes (images) and employing similar ideas to (Turcot and Lowe, 2009). The method is useful for image localization where the database is geotagged and the image graph is constructed from a 2-D map. The top retrieved point then corresponds to the best estimate of the query image location.

2.2 Scaling Up: Very compact image representations

Retrieval approaches based on matching local descriptors (section 2.1) by definition require some information to be stored on a per-feature level. As such, it is impossible to apply the same techniques when scaling up to image corpora containing more than tens of millions of images, without incurring a significant cost of distributing the retrieval system across thousands of machines (section 2.1.1.7). This section reviews methods which discard feature-level information and instead use a very compact global representation (description) of an entire image. Retrieval is performed by searching the database of global image descriptors for the descriptor which is closest to the global descriptor of the query. The search is performed either by a linear scan through the dataset if this is plausible (e.g. Jégou and Chum (2012) show that searching through 1 million 128-D descriptors can be done in 6 ms), or by memory-efficient approximate nearest neighbour search (section 2.1.2). It should be noted that spatial reranking (section 2.1.3) cannot be performed in this setting as shape of local features is discarded along with all per-feature data, and thus it is only possible to take the spatial layout of an image into account by incorporating it into the global descriptor itself.

A popular global descriptor is GIST (Oliva and Torralba, 2001) where a single SIFT-like descriptor is computed over the entire image. However, as demonstrated by Douze et al. (2009), GIST is a rigid descriptor of the entire image and is thus mostly applicable to near duplicate detection and not to object retrieval where robustness to partial occlusion, scale changes, cropping and rotation is required. Therefore, successful methods for object retrieval mainly focus on deriving global descriptors from local descriptors, and we review these next.

Chum et al. (2007a, 2008), Jégou et al. (2009a) and Jégou and Chum (2012) start from a BoW representation and try to compress it. Chum et al. (2007a) and Chum et al. (2008) compute a compact image representation by applying the MinHash algorithm on the BoW vectors (section 2.1.1.6), however this method only manages to retrieve very similar BoWs thus making it is mostly applicable to near duplicate detection. Jégou et al. (2009a) extract MiniBOFs by random aggregation of binarized

BoW vectors. Jégou and Chum (2012) achieve impressive results by concatenating multiple BoW representations followed by whitening and dimensionality reduction with PCA. The multiple BoWs are obtained by using multiple visual vocabularies (obtained by different random initializations of k-means) or vocabularies of different sizes.

2.2.1 Aggregating descriptors

The bag-of-words encoding captures the distribution of local descriptors in the descriptor space by recording their count in various areas of the space; the areas are defined by clustering descriptors into visual words[4]. Methods reviewed in this section store further information about the distribution of descriptors in the descriptor space. This is typically done by aggregating descriptors assigned to same visual words to obtain their means or other moments. The resulting global image descriptor is dense, and can often be successfully compacted by performing dimensionality reduction via PCA.

2.2.1.1 Fisher vectors

Perronnin and Dance (2007) employ the Fisher Vector (FV) encoding (Jaakkola and Haussler, 1998) for a visual classification task. Local descriptors in an image are assumed to be independent samples from a Gaussian Mixture Model (GMM). The Fisher Vector is constructed by taking the gradient of their log-likelihood with respect to the GMM parameters, namely the mixture weights, means and (diagonal)

[4]Due to the lack of a better expression, in this section we use the term "visual word" loosely to denote an area of descriptor space.

covariances. Perronnin and Dance (2007) further show that the derivative with respect to the mixture weights resembles the BoW encoding as it counts the number of descriptors softly assigned to each mixture component (resembling a visual word). Computing the gradient with respect to each Gaussian mean corresponds to aggregating all residuals (vector differences between descriptors and Gaussian means) for descriptors assigned to the mixture component, downscaled by the variance of the component. However, strictly speaking, all descriptor residuals affect all gradients (i.e. there is no assignment to a mixture component), but are weighted by their posterior probability of being generated by the respective components. The gradient with respect to the covariances captures the second moment of the descriptor distribution in an analogous manner.

The Fisher Vector is finally computed as the whitened gradient vector, in order to normalize the dynamic range of various dimensions. Perronnin et al. (2010c) note that the derivatives with respect to the mixture weights do not add much information and are thus discarded, making the final dimensionality of the FV equal to $2KD$, where D is the local descriptor dimensionality and K is the number of mixture components ("visual words").

For the same vector dimensionality, it was found that, for the image retrieval task, the gradients with respect to the variances also do not provide much information (Jégou et al., 2010b, Jégou et al., 2012): the FVs which record the gradients with respect to means and variances (dimensionality: $2KD$) perform similarly to the FVs which record the gradients with respect to the means only, with a doubled number of components $K' = 2K$ (dimensionality: $K'D = 2KD$). The resulting FV is quite similar to super-vector encoding of Zhou et al. (2010).

Perronnin et al. (2010c) provide further insights into FVs, showing that, under some simplifying assumptions, the image-independent information ("background" descriptors) is automatically discarded from the encoding thus keeping only the image-specific information. Perronnin et al. (2010a) apply Fisher Vectors for large scale image retrieval and make connections with the commonly used tf-idf weighting (section 2.1.1.1) – the similarity of two FVs is proportional to the product of frequencies of visual words in the two images, downweighted by the average frequency of the word. Perronnin et al. (2010a) therefore apply power normalization in the manner of Jégou et al. (2009b) (section 2.1.1.4) in order to discount the effect of bursty features. Power normalization is computed by transforming each element v_i of the vector using the formula: $sign(v_i)|v_i|^\alpha$, followed by L2 normalization; a commonly used value for α is 0.5 (Jégou et al., 2012, Perronnin et al., 2010c), also referred to as signed square rooting (SSR).

2.2.1.2 Vector of locally aggregated descriptors (VLAD)

Jégou et al. (2010b) propose VLAD, a global image descriptor motivated by the success of Fisher Vector encoding (section 2.2.1.1). It is computed by aggregating, for each visual word, all residuals (vector differences between descriptors and cluster centres) of descriptors assigned to the same visual word. VLAD can be seen as a simplification of the FV encoding – instead of a GMM it uses a visual vocabulary (built using k-means like for BoW), descriptors are hard-assigned to visual words and there is no inverse weighting by covariances of each visual word. Due to its simplicity and hard-assignment, VLAD is faster to compute than FV. As with FV (the flavour which only records the first order statistics), the final dimensionality of a VLAD vector is KD, where K is the size of the visual vocabulary and D is the dimensionality of the local descriptor.

In the original scheme of Jégou et al. (2010b) the VLAD vectors are L2 normalized. Subsequently, signed square rooting (SSR) normalization was introduced (Jégou and Chum, 2012, Jégou et al., 2012), following its use by Perronnin et al. (2010a) for Fisher Vectors.

There have been several subsequent improvements to VLAD, including ours in chapter 5, and we review them next. Many of these are also applicable to Fisher Vectors due to their similarity with VLAD. Delhumeau et al. (2013) argue that all residuals should contribute equally to VLAD, which is not the case in the original scheme as descriptors closer to cluster centres have smaller residuals. Therefore, Delhumeau et al. (2013) L2 normalize all residuals before they are aggregated. Chen et al. (2011) argue that descriptors which are close to boundaries of visual words are unlikely to be repeatable (i.e. a very similar descriptor is likely to be assigned to a different word), and simply remove these “outliers”. They also investigate different per-cluster residual aggregation methods, namely (the original) sum, mean and median, and find that mean aggregation works the best.

Delhumeau et al. (2013) revisit power normalization – all other encoding steps are invariant to rotation in the descriptor space, i.e. rotating SIFT descriptors prior to vocabulary construction and VLAD encoding results in unchanged retrieval performance, since the computed VLADs are just rotated versions of the “canonical” ones. However, power normalization is not rotationally invariant and therefore Delhumeau et al. (2013) investigate the optimal choice of the rotation. They argue that the largest eigenvectors capture the main bursty patterns and therefore the rotation with PCA basis is expected to be a good choice. Furthermore, in order to capture a large variety of bursty patterns, they propose to perform a different rotation for every visual word. Thus, Delhumeau et al. (2013) rotate the residuals inside each visual word according to the local PCA basis; the PCA for each visual word is computed using all descriptors assigned to the visual word.

Delhumeau et al. (2013) and Zhao et al. (2013a) obtain a large boost in retrieval performance by computing local descriptors densely. Furthermore, in order to capture some geometric information, Zhao et al. (2013a) pool local descriptors based on their dominant orientation. A VLAD descriptor is computed for each quantized dominant orientation (a typical number of quantization levels is 8), and the VLADs are then concatenated to form Covariant-VLAD (CVLAD). Two CVLADs are compared by searching over all possible relative rotations and picking the one which yields the largest scalar product. The similarity computation can be sped up by computing it in the frequency domain, like in (Revaud et al., 2013).

VLAD can be successfully compacted using PCA for dimensionality reduction, in some cases PCA even increases retrieval performance (Jégou et al., 2012) as it removes some of the noise from the data. Another very successful method of reducing the dimensionality of VLAD descriptors is the one of Jégou and Chum (2012) – multiple VLAD vectors, computed using multiple visual vocabularies obtained by different random initializations of k-means, are concatenated together, followed by dimensionality reduction via PCA and whitening. Using multiple vocabularies somewhat alleviates quantization effects and was successfully applied in the same work on BoW vectors.

Chapter 3

Datasets and Evaluation

This chapter describes standard, community accepted methods for evaluating large scale object retrieval, which are used throughout this thesis. Publicly available datasets used for evaluation are described next, followed by the definition of the evaluation metric.

3.1 Datasets

3.1.1 Oxford buildings

Introduced by Philbin et al. (2007), it consists of 5062 high-resolution images automatically crawled from Flickr using queries such as "Oxford Christ Church", "Oxford Radcliffe Camera" and "Oxford". Ground truth was obtained manually for 11 landmarks.

Images of a certain landmark are labelled as *Good* if it is clearly visible, *OK* if more than 25% is visible and *Junk* if less than 25% of the landmark is visible or the image is severely distorted. *Junk* images are ignored for a particular query, i.e. they are not considered to be positives or negatives and are effectively removed from the list of retrieved images before computation of retrieval performance metrics.

There are 55 queries, 5 per landmark, where each query is defined in terms of a query image and a bounding box. All of the queries are shown in figure 3.1.

It is quite a challenging dataset 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.

3.1.2 Paris buildings

Analogously to Oxford 5k, 6392 images of Paris were obtained from Flickr and 55 (5 for each of the 11 chosen landmarks) queries are used for evaluation (Philbin et al., 2008). 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 computing a visual vocabulary from it while evaluating performance on Oxford 5k.

3.1.3 Holidays

Contains 1491 high-resolution images containing personal holiday photos with 500 queries containing only a few positive examples per query (Jégou et al., 2008). This dataset is more targeted at large scale *image* retrieval rather than *particular object* retrieval due to limited changes in viewpoint and scale, and queries are defined

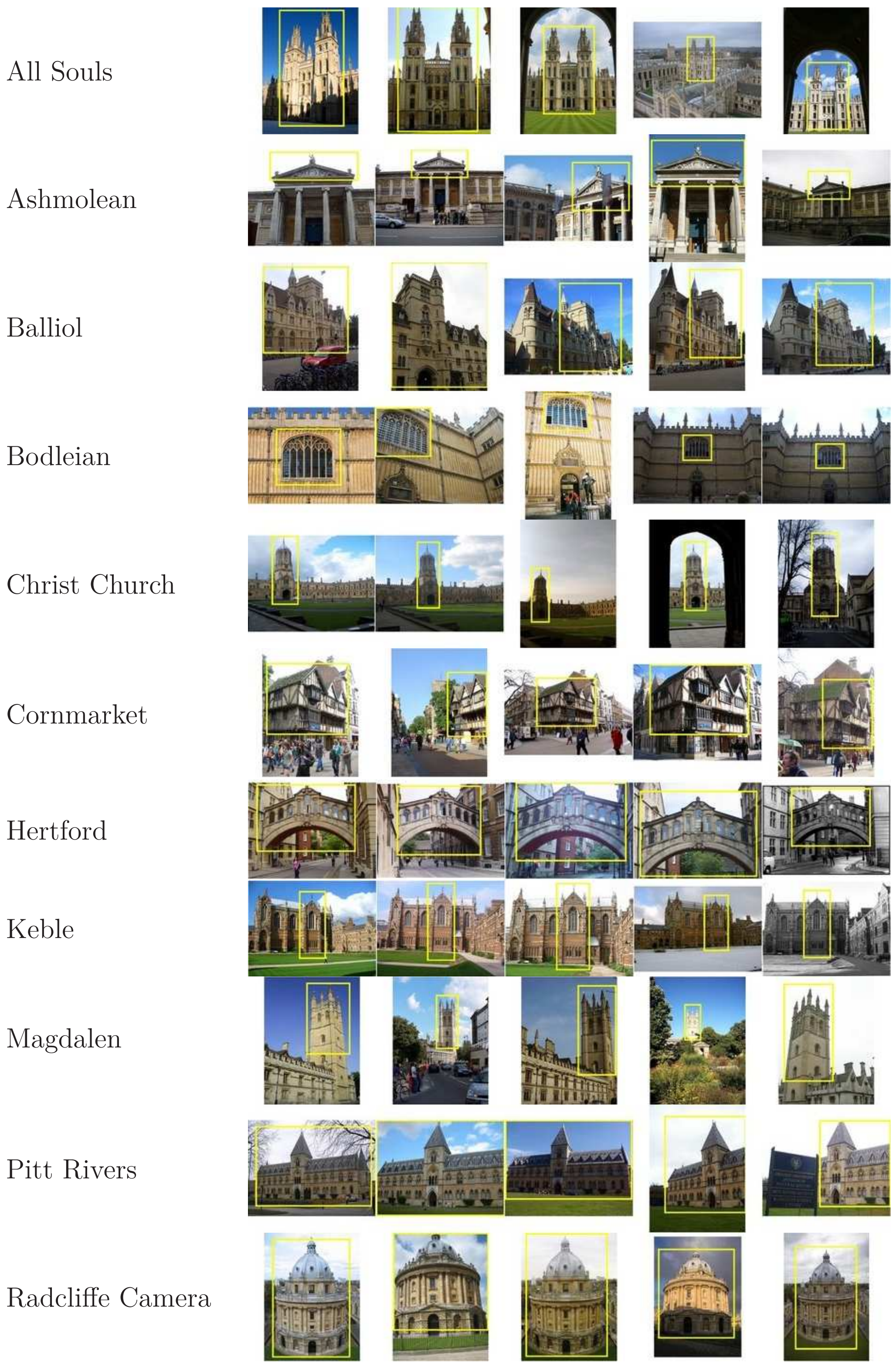


Figure 3.1: **Oxford dataset.** Landmarks and queries used for evaluation.

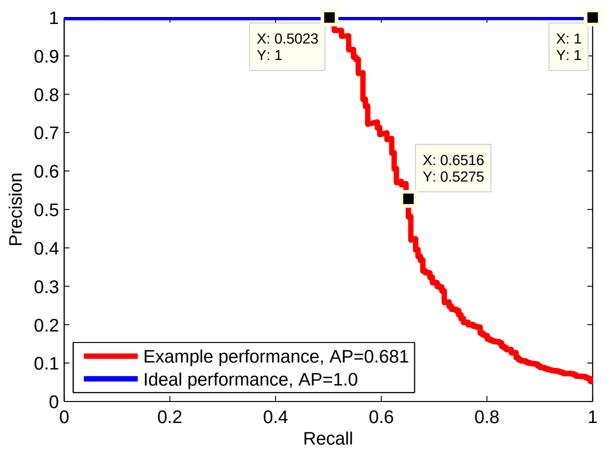


Figure 3.2: **Precision-recall (PR) curve.** The blue line shows an ideal PR curve with precision=1.0 at recall=1.0, meaning that all positives are retrieved and all retrieved images are true positives. The red line shows a more realistic PR curve where precision=1.0 up to recall of 0.5023, meaning that no false positives are encountered when 50% of all positives are retrieved. To achieve recall of 53% precision is sacrificed with 35% of retrieved images being false positives.

only in terms of complete images and not specific image regions (objects). The visual vocabulary is typically trained on an independent dataset, Flickr 60k. The basic dataset is usually appended with another 1M Flickr images to test large scale retrieval, forming the *Holidays+Flickr1M* dataset.

Note that the query image is ignored in retrieval results, unlike for Oxford and Paris datasets where it is counted as a positive.

3.2 Evaluation procedure

Retrieval quality for a single query is measured in terms of precision-recall (PR) curves (an example is shown in figure 3.2). Precision is defined as the proportion of true positives in the retrieved images; recall is the ratio of retrieved true positives to the total number of positives for the query. In simple terms, precision measures the purity of the retrieved list, while recall measures what fraction of the total number of known positives is discovered.

Certain applications, like Google Goggles, which try to answer the question "what is this?" are only interested in maximising precision as obtaining a single correct result is sufficient to recognise the object. For such applications, a useful performance measure for a single query is precision@R, i.e. the precision when R images are retrieved (e.g. $R = 1$ in the extreme).

Large recall is also commonly required, for example 3D reconstruction requires a large number of relevant images in order to build an accurate 3D model. Average precision (Philbin et al., 2007) (AP) summarises the precision-recall curve, and thus the precision vs recall trade-off, by measuring the area under the curve; AP of 1.0 represents the ideal case. The corresponding measure for a collection of queries is the mean average precision (mAP).

Chapter 4

Improving Bag-of-Words Retrieval

We consider the problem of near-real time large scale particular object retrieval. Many works have addressed this problem (section 2.1); the standard approach is to represent an image 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 (section 2.1.1).

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 and are addressed by methods reviewed in sections 2.1.1.3, 2.1.1.4, 2.1.2 and 2.1.4.

We make the following three novel contributions in this chapter:

1. RootSIFT: In section 4.2 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.

2. Discriminative query expansion: Query expansion methods (section 2.1.4), 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. Current methods for query expansion combine the BoW vectors of the spatially verified results, e.g. by averaging. In section 4.3 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 (Chum et al., 2007b), whilst maintaining immediate retrieval speeds through efficient use of the inverted index.

3. Database-side feature augmentation: 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 (Turcot and Lowe, 2009) 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. Though very powerful, this method suffers from not taking into account the spatial configuration of augmenting features. In section 4.4 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. (2007) (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 4.2–4.4 we describe each of these methods in detail and demonstrate their performance gain using the method of Philbin et al. (2007) 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 4.5. 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.

4.1 Baseline retrieval system

We follow the standard BoW retrieval framework described in (Philbin et al., 2007). We use affine-Hessian interest points (Mikolajczyk and Schmid, 2004b), 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 Philbin et al. (2007). 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 Chum et al. (2007b) (described in detail in section 4.3) achieves a mAP of 0.726 on Oxford 105k compared to the original 0.711 (Chum et al., 2007b). 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 (Chum et al., 2011, Mikulik et al., 2010, Philbin et al., 2008) (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.

4.2 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 (Lowe, 2004), 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 neighbour methods (essential for systems with very large vocabularies) can still be used; as can soft assignment of descriptors to visual words (Jégou et al., 2010a, Philbin et al., 2008), query expansion, and other extensions which only require Euclidean distance on SIFT (Jégou et al., 2008, 2010b, Mikulik et al., 2010, Philbin et al., 2010).

Retrieval Method	SIFT		RootSIFT	
	Ox5k	Ox105k	Ox5k	Ox105k
Philbin et al. (2007): tf-idf ranking	0.636	0.515	0.683	0.581
Philbin et al. (2007): tf-idf with spatial reranking	0.672	0.581	0.720	0.642
Chum et al. (2007b): average query expansion (AQE)	0.839	0.726	0.850	0.756
Turcot and Lowe (2009): database-side feature augmentation (AUG)	0.776	0.711	0.827	0.759
This chapter: discriminative query expansion (DQE)	**0.847**	0.752	0.861	0.781
This chapter: spatial database-side feature augmentation (SPAUG)	0.785	0.723	0.838	0.767
This chapter: SPAUG + DQE	0.844	**0.795**	**0.881**	**0.823**

Table 4.1: **Retrieval performance (mAP) of various proposed methods.** We use our implementation of all listed methods (Chum et al., 2007b, Philbin et al., 2007, Turcot and Lowe, 2009) 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 4.4.

The dramatic improvement in performance is shown in table 4.1, 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 (Perronnin et al., 2010b, Vedaldi and Zisserman, 2010). 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. (2005) 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 (2010) uses Jeffrey's divergence to compare SIFT vectors concentrating on descriptor compression, Pele and Werman (2008) use a variant of the Earth Mover's Distance or a quadratic χ^2 metric (Pele and Werman, 2010).

4.3 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 4.1 illustrates schematically how negative data can benefit DQE over average query expansion.

Table 4.1 compares the DQE method to our implementation of the average query expansion (AQE) of Chum et al. (2007b). It can be seen that DQE is consistently superior to AQE. The performance gain is particularly evident with increasing dataset

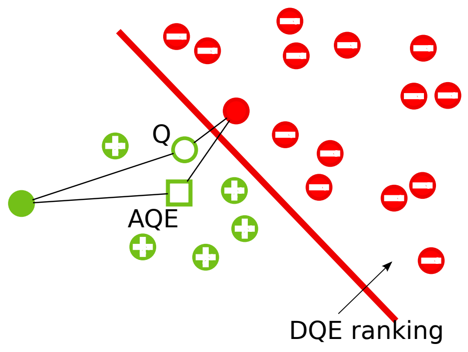


Figure 4.1: **Discriminative query expansion (DQE).** Illustration of the BoW feature space. True positives are shown as green circles and true negatives as red circles. Spatially verified images used to expand the query (i.e. “known” positives) are shown with pluses inside, low ranking images (i.e. “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.

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 (Chang and Lin, 2011). 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 4.1 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 (Chum et al., 2011, Knopp et al., 2010) has proposed methods for identifying words that are confusing for a given query image, and thus should not be used. In (Chum et al., 2011) these confusing words are then removed when re-querying, though the actual "expansion" is still performed by simply averaging

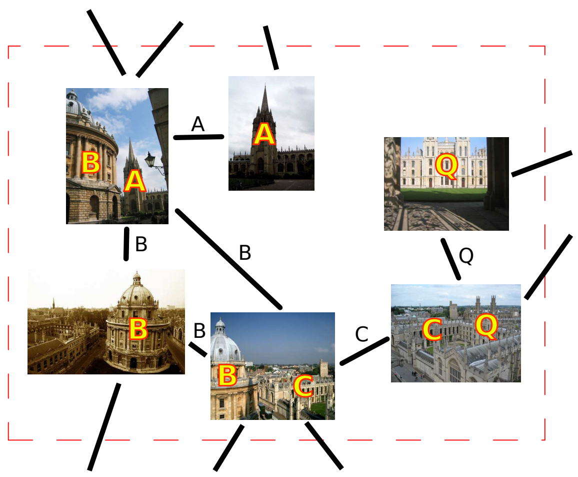


Figure 4.2: **Image graph.** Part of an image graph, nodes represent images while edges signify that images contain a common object; objects are labelled A, B, C and Q.

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).

4.4 Database-side feature augmentation

Turcot and Lowe (2009) use a matching graph to improve retrieval performance by augmenting the BoW for each image with the visual word counts of all neighbouring images in the graph. In a matching graph (Philbin and Zisserman, 2008) nodes represent images, while edges signify that images contain an object in common (figure 4.2). 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

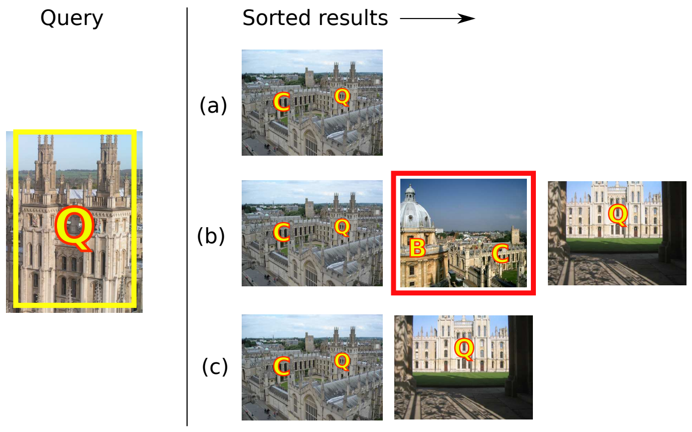


Figure 4.3: **Database-side for feature augmentation.** Retrieval performance of augmentation methods: the queried object is highlighted in yellow on the left-most image. (a) tf-idf retrieval results when not using image graph information, a challenging image is not retrieved; (b) retrieved images using the method of Turcot and Lowe (2009) and graph shown in figure 4.2: recall is improved but precision decreases as the false positive (highlighted in red) is augmented with all the visual words of its highly ranked neighbouring image; (c) our method shows increased recall while maintaining high precision since images are only augmented with visual words from relevant neighbouring regions.

errors.

The procedure of Turcot and Lowe (2009) 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, Turcot and Lowe (2009) augment the BoW vector with *all* visual words from neighbouring 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 4.2). 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 4.3.

Table 4.1 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 (Turcot and Lowe, 2009), 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 Turcot and Lowe (2009) 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 neighbouring images (neighbouring 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 neighbouring images are used for augmentation; our extension requires explicit augmentation as one image can contribute different words to different neighbours 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 (Turcot and Lowe, 2009), and illustrates that the original approach indeed introduces a large number of irrelevant and possibly detrimental visual words.

Implementation details

We use the approach of Philbin and Zisserman (2008) 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., 2007) and an edge is constructed to each spatially verified image. An alternative graph construction method which employs hashing (Chum and Matas, 2010a) 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

	SIFT		RootSIFT	
AUG variant	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 4.2: **Idf recomputation for database-side feature augmentation.** Retrieval performance of our implementation of the Turcot and Lowe (2009) database-side augmentation method (AUG) using just tf-idf or with spatial re-ranking (SR). Recomputing the idf using the augmented dataset always improves performance.

method (Turcot and Lowe, 2009) and our extension benefit from idf recomputation, as shown in Table 4.2. 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%.

4.5 Results and discussion

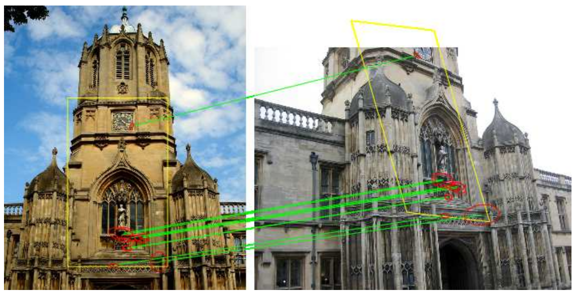
In this section we discuss and compare the three new methods to related work and the state of the art. Comparing absolute performance to previous publications is difficult as the results depend on a number of important implementation details (even though the same benchmark image datasets are used). There 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 Perďoch et al. (2009) 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, Paris 6k (section 3.1) image datasets.

4.5.1 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 chapter. In every single case RootSIFT significantly outperforms the standard L2 distance for SIFT comparison; we only show a subset of these experiments in table 4.1. Note that the simple tf-idf scheme with RootSIFT and without spatial reranking outperforms tf-idf using SIFT with spatial reranking. Figure 4.4 shows examples of matched image patches using SIFT and RootSIFT, demonstrating that RootSIFT yields more complete matches enabling better object localization

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. (2010) 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 (Philbin et al., 2010). The mAPs are (Philbin et al. (2010)/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 still more room for improvement by combining the root transformation with learning; this was indeed achieved after the publication of this work by Simonyan et al. (2013b).


(a) SIFT (L2 distance): 10 matches

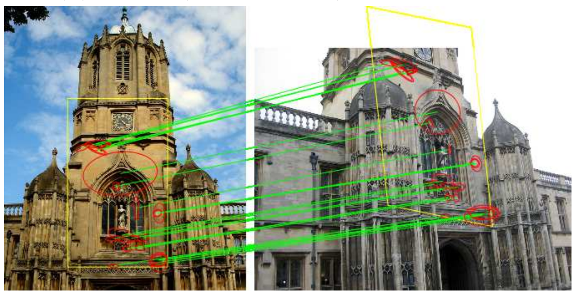

(b) RootSIFT: 26 matches

Figure 4.4: **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.

4.5.2 Final retrieval system

We combine all the proposed improvements into one system and evaluate its performance. RootSIFT (section 4.2) 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 4.4). At query time we extract RootSIFT descriptors 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

	Method	F	V	SA	Ox5k	Ox105k	Paris6k
a	Philbin et al. (2008)	M	S		0.801	0.708	N/A
b	Philbin et al. (2008)	M	S	√	0.825	0.718	N/A
c	Qin et al. (2011)	M	S		0.814	0.767	0.803
d	This chapter	M	S		**0.881**	**0.823**	**0.850**
e	Philbin et al. (2008)	M	I		0.654	0.562	N/A
f	Philbin et al. (2008)	M	I	√	**0.719**	**0.605**	N/A
g	This chapter	M	I		0.714	0.602	**0.660**
h	Perd'och et al. (2009)	P	I		0.784	0.728	N/A
i	Perd'och et al. (2009)	P	I	√	0.822	0.772	N/A
j	Mikulik et al. (2010)	P	$\star$	√	**0.849**	**0.795**	**0.824**
k	Chum et al. (2011)	P	I		0.827	0.767	0.805
l	This chapter	P	I		0.809	0.722	0.765
m	Perd'och et al. (2009)	P	S		0.901	0.856	N/A
n	Perd'och et al. (2009)	P	S	√	0.916	0.885	N/A
o	This chapter	P	S		**0.929**	**0.891**	**0.910**

Table 4.3: **Comparison of the combined method (section 4.5.2) with state-of-the-art.** *F* denotes the feature detector used: *M* for (Mikolajczyk and Schmid, 2004b) and *P* for (Perd'och et al., 2009). *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 $\star$ for the case of (Mikulik et al., 2010) 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 (Philbin et al., 2008) or (Chum et al., 2011, Mikulik et al., 2010, Perd'och et al., 2009) 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.

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 4.3).

Table 4.3 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 (2004b) or Perd'och et al. (2009)), and learning the visual vocabulary from two different datasets (from Oxford 5k or Paris 6k).

First, (a–d), we report on using the Mikolajczyk and Schmid (2004b) 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 (Philbin et al., 2008), 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).

Third, (h–l), using the feature detector of Perďoch et al. (2009) boosts the performance even when using an independent vocabulary. The combined method, (l), does not top the results of (Chum et al., 2011, Mikulik et al., 2010, Perďoch et al., 2009), (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. (2010), (j), where a very large vocabulary is generated and similarities between the words are

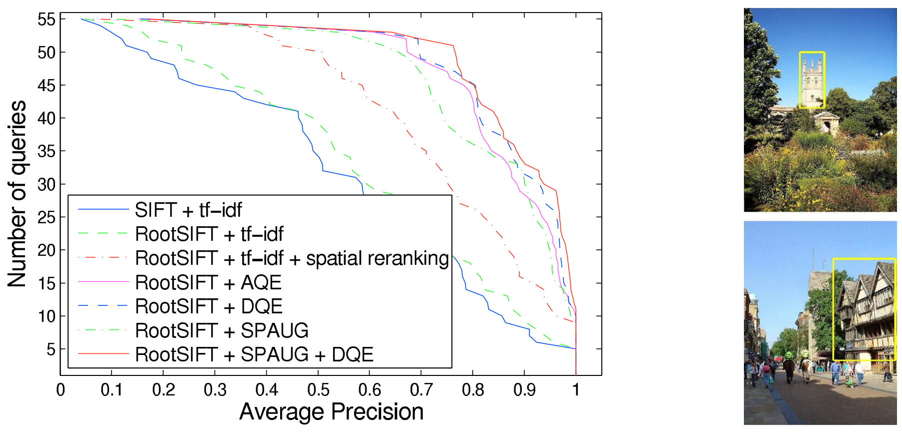


Figure 4.5: **Retrieval performance of various methods.** (left) The number of queries that achieve a specific AP or higher. Tested on Oxford 105k with the feature detector of Perďoch et al. (2009) and Oxford 5k vocabulary. (right) Two badly performing queries for all methods.

learnt.

Finally, (m–o), use the Perďoch et al. (2009) 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 4.5 (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 4.5 (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.

4.6 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 Philbin et al. (2008) 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. (2010a) and Jain et al. (2011)'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.

4.7 Impact

The work described in this chapter was published in CVPR 2012, and for a bit more than a year it has made significant impact in the Computer Vision community.

RootSIFT. As a result of its superiority, RootSIFT is rapidly replacing SIFT in retrieval systems (Delhumeau et al., 2013, Fernando and Tuytelaars, 2013, Jégou and Chum, 2012, Qin et al., 2013, Tolias and Jégou, 2013, Zhao et al., 2013a,b) as well as non-retrieval applications including classification (Garg et al., 2013, Juneja et al., 2013, Krause et al., 2013), face recognition (Simonyan et al., 2013a), facial landmark localization (Zhou et al., 2013), tracking (Prankl et al., 2013), event retrieval (Douze et al., 2013, Revaud et al., 2013), localization (Torii et al., 2013) and 3D mapping (Endres et al., 2013).

Discriminative query expansion (DQE). Our work is the first one to employ discriminative learning for object retrieval; a few subsequent methods with the same motivation are mentioned here. Chen et al. (2012) confirm that DQE outperforms average query expansion and further improve on it by learning an ensemble of linear SVMs in a boosting framework. Gronat et al. (2013) and Cao and Snavely (2013) employ discriminative learning in location recognition (section 2.1.1.4). Douze et al. (2013) consider event retrieval using VLAD descriptors and confirm that DQE is superior to AQE in this case as well; they also propose another discriminative query expansion scheme.

Chapter 5

Improving Compact Image Representations

Like the previous chapter, the topic of this chapter is also large scale object retrieval. The focus of this chapter, however, is towards very large scale retrieval where, due to storage requirements, very compact image descriptors are required and no information about the original local descriptors can be accessed directly at run time. Section 2.2 reviews related work in this area.

We start from VLAD, the state-of-the art compact descriptor introduced by Jégou et al. (2010b) for this purpose (section 2.2.1.2), and make three novel contributions:

1. Intra-normalization: We propose a new normalization scheme for VLAD that addresses the problem of burstiness (Jégou et al., 2009b), where a few large components of the VLAD vector can adversely dominate the similarity computed between VLADs. The new normalization is simple, and always improves retrieval performance.

2. Multi-VLAD: We study the benefits of recording multiple VLADs for an image and show that retrieval performance is improved for small objects (those that cover only a small part of the image, or where there is a significant scale change from the query image). Furthermore, we propose a method of sub-VLAD localization where the window corresponding to the object instance is estimated at a finer resolution than the VLAD tiling.

3. Vocabulary adaptation: We investigate the problem of vocabulary sensitivity, where a vocabulary trained on one dataset, A, is used to represent another dataset B, and the performance is inferior to using a vocabulary trained on B. We propose an efficient, simple, method for improving VLAD descriptors via vocabulary adaptation, without the need to store or recompute any local descriptors in the image database.

The first two contributions are targeted at improving VLAD performance. The first improves retrieval in general, and the second partially overcomes an important deficiency – that VLAD has inferior invariance to changes in scale (compared to a BoW approach). The third contribution addresses a problem that arises in real-world applications where, for example, image databases grow with time and the original vocabulary is incapable of representing the additional images well.

In sections 5.2–5.4 we describe each of these methods in detail and demonstrate their performance gain over earlier VLAD formulations, using the Oxford Buildings 5k and Holidays image dataset benchmarks as running examples. The methods are combined and compared to the state-of-the-art for larger scale retrieval (Oxford 105k and Flickr1M) in section 5.5.

5.1 VLAD review, datasets and baselines

We first briefly remind the reader of the original VLAD computation and subsequent variations (a more detailed review is available in section 2.2.1.2), and then overview the datasets that will be used for performance evaluation and those that will be used for vocabulary building (obtaining the cluster centres required for VLAD computation).

5.1.1 VLAD

VLAD is constructed as follows: regions are extracted from an image using an affine invariant detector, and described using the 128-D SIFT descriptor. Each descriptor is then assigned to the closest cluster of a vocabulary of size k (where k is typically 64 or 256, so that clusters are quite coarse). For each of the k clusters, the residuals (vector differences between descriptors and cluster centres) are accumulated, and the k 128-D sums of residuals are concatenated into a single $k \times 128$ dimensional descriptor; we refer to it as the unnormalized VLAD.

In the original scheme (Jégou et al., 2010b) the VLAD vectors are L2 normalized. Subsequently, a signed square rooting (SSR) normalization was introduced (Jégou and Chum, 2012, Jégou et al., 2012), following its use by Perronnin et al. (2010a) for Fisher Vectors. To obtain the SSR normalized VLAD, each element of an unnormalized VLAD is sign square rooted (i.e. an element x_i is transformed into $sign(x_i)\sqrt{|x_i|}$) and the transformed vector is L2 normalized. We will compare with both of these normalizations in the sequel, and use them as baselines for our approach.

5.1.2 Benchmark datasets and evaluation procedure

The performance is measured on two standard and publicly available image retrieval benchmarks, Oxford buildings and Holidays (section 3.1). For both, a set of predefined queries with hand-annotated ground truth is used, and the retrieval performance is measured in terms of mean average precision (mAP, section 3.2).

We follow the standard experimental scenario of (Jégou and Chum, 2012) for all benchmarks: for Oxford 5k and 105k the detector and SIFT descriptor are computed as in (Perďoch et al., 2009); while for Holidays(+Flickr1M) the publicly available SIFT descriptors are used.

Vocabulary sources. Three different datasets are used for vocabulary building (i.e. clustering on SIFTs): (i) Paris 6k (section 3.1.2), which is often used as an independent dataset from the Oxford buildings (Chum et al., 2011, Jégou and Chum, 2012, Philbin et al., 2008); (ii) Flickr60k (Jégou et al., 2008), which contains 60k images downloaded from Flickr, and is used as an independent dataset from the Holidays dataset (Jégou et al., 2008, 2009b, 2010b); and, (iii) 'no-vocabulary', which simply uses the first k (where k is the vocabulary size) SIFT descriptors from the Holidays dataset. As k is typically not larger than 256 whereas the smallest dataset (Holidays) contains 1.7 million SIFT descriptors, this vocabulary can be considered independent from all datasets.

5.2 Vocabulary adaptation

In this section we introduce cluster adaptation to improve retrieval performance for the case where the cluster centres used for VLAD are not consistent with the dataset

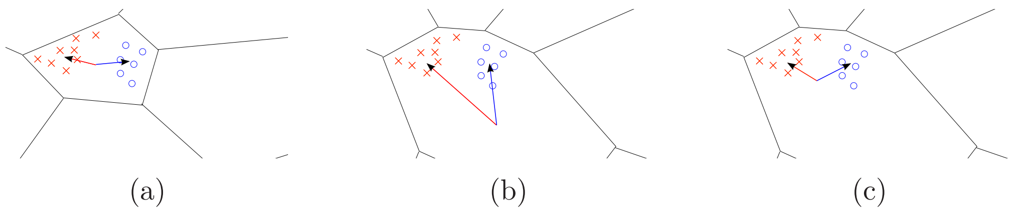


Figure 5.1: **VLAD similarity measure under different clusterings.** The Voronoi cells illustrate the coarse clustering used to construct VLAD descriptors. Red crosses and blue circles correspond to local descriptors extracted from two different images, while the red and blue arrows correspond to the sum of their residuals (differences between descriptors and the cluster centre). Assume the clustering in (a) is a good one (i.e. it is representative and consistent with the dataset descriptors), while the one in (b) is not. By changing the clustering from (a) to (b), the sign of the similarity between the two images (from the cosine of the angle between the residuals) changes dramatically, from negative to positive. However, by performing cluster centre adaptation the residuals are better estimated (c), thus inducing a better estimate of the image similarity which is now consistent with the one induced by the clustering in (a).

– for example they were obtained on a different dataset or because new data has been added to the dataset. As described earlier (sections 2.2.1.2 and 5.1.1), VLAD is constructed by aggregating differences between local descriptors and coarse cluster centres, followed by L2 normalization. For the dataset used to learn the clusters (by k-means) the centres are *consistent* in that the mean of all vectors assigned to a cluster over the entire dataset is the cluster centre. For an individual VLAD (from a single image) this is not the case, or course, and it is also not the case, in general, for VLADs computed over a different dataset. As will be seen below the inconsistency can severely impact performance. An ideal solution would be to recluster on the current dataset, but this is costly and requires access to the original SIFT descriptors. Instead, the method we propose alleviates the problem without requiring reclustering.

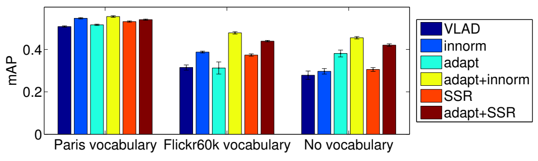
The similarity between VLAD descriptors is measured as the scalar product between them, and this decomposes as the sum of scalar products of aggregated residuals for each coarse cluster independently. Consider a contribution to the similarity for one particular coarse cluster k. We denote with $x_k^{(1)}$ and $x_k^{(2)}$ the set of all descriptors in image 1 and 2, respectively, which get assigned to the same coarse cluster k. The contribution to the overall similarity of the two VLAD vectors is then equal to:

$$\frac{1}{C^{(1)}} \sum_i (x_{k,i}^{(1)} - \mu_k)^T \frac{1}{C^{(2)}} \sum_j (x_{k,j}^{(2)} - \mu_k) \tag{5.1}$$

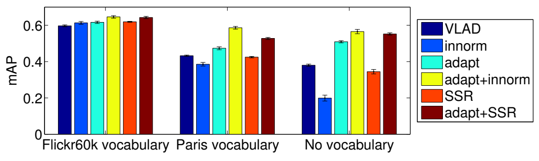
where μ_k is the centroid of the cluster, and $C^{(1)}$ and $C^{(2)}$ are normalizing constants which ensure all VLAD descriptors have unit norm. Thus, the similarity measure induced by the VLAD descriptors is increased if the scalar product between the residuals is positive, and decreased otherwise. For example, the sets of descriptors illustrated in figure 5.1a are deemed to be very different (they are on opposite sides of the cluster centre) thus giving a negative contribution to the similarity of the two images.

It is clear that the VLAD similarity measure is strongly affected by the cluster centre. For example, if a different centre is used (figure 5.1b), the two sets of descriptors are now deemed to be similar thus yielding a positive contribution to the similarity of the two images. Thus, a different clustering can yield a completely different similarity value.

We now introduce *cluster centre adaptation* to improve residual estimates for an inconsistent vocabulary, namely, using new adapted cluster centres $\hat{\mu}_k$ that are consistent when computing residuals (equation (5.1)), instead of the original cluster centres μ_k. The algorithm consists of two steps: (i) compute the adapted cluster



(a) Oxford 5k benchmark (section 3.1.1)



(b) Holidays benchmark (section 3.1.3)

Figure 5.2: **Retrieval performance.** Six methods are compared, namely: (i) baseline: the standard VLAD, (ii) intra-normalization, *innorm* (section 5.3), (iii) centre adaptation, *adapt* (section 5.2), (iv) *adapt* followed by *innorm*, (v) baseline: signed square rooting *SSR*, (vi) aided baseline: *adapt* followed by *SSR*. Each result corresponds to the mean result obtained from four different test runs (corresponding to four different clusterings), while error bars correspond to one standard deviation. The results were generated using RootSIFT (chapter 4.2) descriptors and vocabularies of size $k = 256$.

centres $\hat{\mu}_k$ as the mean of all local descriptors in the dataset which are assigned to the same cluster k; (ii) recompute all VLAD descriptors by aggregating differences between local descriptors and the adapted centres $\hat{\mu}_k$. Note that step (ii) can be performed without actually storing or recomputing all local descriptors as their assignment to clusters remains unchanged and thus it is sufficient only to store the descriptor sums for every image and each cluster.

Figure 5.1c illustrates the improvement achieved with centre adaptation, as now residuals, and thus similarity scores, are similar to the ones obtained using the original clustering in figure 5.1a. Note that for an adapted clustering the cluster centre is indeed equal to the mean of all the descriptors assigned to it from the dataset. Thus, our cluster adaptation scheme has no effect on VLADs obtained using consistent clusters, as desired.

To illustrate the power of the adaptation, a simple test is performed where the Flickr60k vocabulary is used for the Oxford 5k dataset, and the difference between the original vocabulary and the adapted one measured. The mean magnitude of the displacements between the $k = 256$ adapted and original cluster centres is 0.209, which is very large keeping in mind that RootSIFT descriptors (chapter 4.2) themselves all have a unit magnitude. For comparison, when the Paris vocabulary is used, the mean magnitude of the difference is only 0.022.

Results. Figure 5.2 shows the improvement in retrieval performance obtained when using cluster centre adaptation (*adapt*) compared to the standard VLAD under various dataset sources for the vocabulary. Center adaptation improves results in all cases, especially when the vocabulary was computed on a vastly different image database or not computed at all. For example, on Holidays with Paris vocabulary the mAP increases by 9.7%, from 0.432 to 0.474; while for the no-vocabulary case, the mAP improves by 34%, from 0.380 to 0.509. The improvement is smaller when the Flickr60k vocabulary is used since the distribution of descriptors is more similar to the ones from the Holidays dataset, but it still exists: 3.2% from 0.597 to 0.616. The improvement trends are similar for the Oxford 5k benchmark as well.

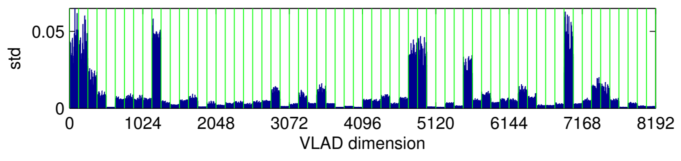
Application in large scale retrieval. Consider the case of real-world large-scale retrieval where images are added to the database with time. This is the case, for example, with users uploading images to Flickr or Facebook, or Google indexing images on new websites. In this scenario, one is forced to use a fixed precomputed vocabulary since it is impractical (due to storage and processing requirements) to recompute too frequently as the database grows, and reassign all descriptors to the newly obtained clusters. In this case, it is quite likely that the obtained clusters are inconsistent, thus inducing a bad VLAD similarity measure. Using cluster centre adaptation fits this scenario perfectly as it provides a way of computing better similarity estimates without the need to recompute or store all local descriptors, as descriptor assignment to clusters does not change.

5.3 Intra-normalization

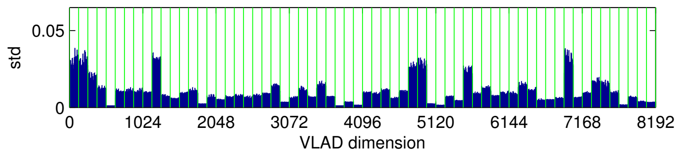
In this section, it is shown that current methods for normalizing VLAD descriptors, namely simple L2 normalization (Jégou et al., 2010b) and signed square rooting (Perronnin et al., 2010a), are prone to putting too much weight on bursty visual features, resulting in a suboptimal measure of image similarity. To alleviate this problem, we propose a new method for VLAD normalization.

The problem of bursty visual elements was first noted in the bag-of-visual-words (BoW) setting (Jégou et al., 2009b): a few artificially large components in the image descriptor vector (for example resulting from a repeated structure in the image such as a tiled floor) can strongly affect the measure of similarity between two images, since the contribution of other important dimensions is hugely decreased. This problem was alleviated by discounting large values by element-wise square rooting the BoW vectors and re-normalizing them. In a similar manner VLADs are signed square root (SSR) normalized (Jégou and Chum, 2012, Jégou et al., 2012). Figure 5.3 shows the effects these normalizations have on the average energy carried by each dimension in a VLAD vector.

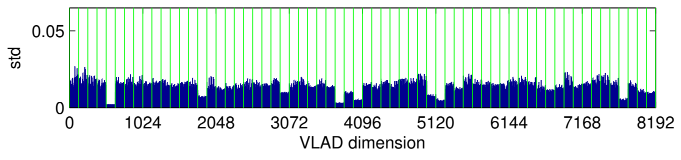
We propose here a new normalization, termed *intra-normalization*, where the sum of residuals is L2 normalized *within* each VLAD block (i.e. sum of residuals within a coarse cluster) independently. As in the original VLAD and SSR, this is followed by L2 normalization of the entire vector. This way, regardless of the amount of bursty image features their effect on VLAD similarity is localized to their coarse cluster,



(a) Original VLAD normalization (L2)



(b) Signed square rooting (SSR) followed by L2



(c) Intra-normalization (innorm) followed by L2

Figure 5.3: **The effect of various normalizing schemes for VLAD.** The plots show the standard deviation (i.e. energy) of the values for each dimension of VLAD across all images in the Holidays dataset; the green lines delimit blocks of VLAD associated with each cluster centre. It can be observed that the energy is strongly concentrated around only a few components in the VLAD vector under the original L2 normalization scheme (5.3a). These peaks strongly influence VLAD similarity scores, and SSR does indeed manage to discount their effect (5.3b). However, even with SSR, it is clear that the same few components are responsible for a significant amount of energy and are still likely to bias similarity scores. (c) Intra-normalization completely alleviates this effect (see section 5.3). The relative improvement in the retrieval performance (mAP) is 7.2% and 13.5% using *innorm* compared to *SSR* and *VLAD*, respectively. All three experiments were performed on Holidays with a vocabulary of size $k = 64$ (small so that the components are visible) learnt on Paris with cluster centre adaptation.

and is of similar magnitude to all other contributions from other clusters. While SSR reduces the burstiness effect, it is limited by the fact that it only *discounts* it. In contrast, intra-normalization fully suppresses bursts, as witnessed in figure 5.3c which shows absolutely no peaks in the energy spectrum.

Discussion. The geometric interpretation of *intra-normalization* is that the similarity of two VLAD vectors depends on the *angles* between the residuals in corresponding clusters. This follows from the scalar product of equation (5.1): since the residuals are now L2 normalized the scalar product depends only on the cosine of the differences in angles of the residuals, not on their magnitudes. Chen et al. (2011) have also proposed an alternative normalization where the per-cluster mean of residuals is computed instead of the sum. The resulting representation still depends on the magnitude of the residuals, which is strongly affected by the size of the cluster, whereas in *intra-normalization* it does not. Note that all the arguments made in favor of cluster centre adaptation (section 5.2) are unaffected by intra-normalization. Specifically, only the values of $C^{(1)}$ and $C^{(2)}$ change in equation (5.1), and not the dependence of the VLAD similarity measure on the quality of coarse clustering which is addressed by cluster centre adaptation.

Results. As shown in figure 5.2, intra-normalization (innorm) combined with centre adaptation (adapt) always improves retrieval performance, and consistently outperforms other VLAD normalization schemes, namely the original VLAD with L2 normalization and SSR. Center adaptation with intra-normalization (adapt+innorm) significantly outperforms the next best method (which is adapt+SSR); the average relative improvement on Oxford 5k and Holidays is 4.7% and 6.6%, respectively. Compared to SSR without centre adaptation our improvements are even more evident: 35.5% and 27.2% on Oxford 5k and Holidays, respectively.

5.4 Multiple VLAD descriptors

In this section we investigate the benefits of tiling an image with VLADs, instead of solely representing the image by a single VLAD. As before, our constraints are the memory footprint and that any performance gain should not involve returning to the original SIFT descriptors for the image. We target objects that only cover a small part of the image (VLAD is known to have inferior performance for these compared to BoW), and describe first how to improve their retrieval, and second how to predict their localization and scale (despite the fact that VLAD does not store any spatial information).

The multiple VLAD descriptors (MultiVLAD) are extracted on a regular 3×3 grid at three scales. 14 VLAD descriptors are extracted: nine (3×3) at the finest scale, four (2×2) at the medium scale (each tile is formed by 2×2 tiles from the finest scale), and one covering the entire image. At run time, given a query image and region of interest (ROI) covering the queried object, a single VLAD is computed over the ROI and matched across database VLAD descriptors. An image in the database is assigned a score equal to the maximum similarity between any of its VLAD descriptors and the query.

As will be shown below, computing VLAD descriptors at fine scales enables retrieval of small objects, but at the cost of increased storage (memory) requirements. However, with 20 bytes per image (Jégou et al., 2010b), 14 VLADs per image amounts to 28 GB for a 100 million images, which is still a manageable amount of data that can easily be stored in the main memory of a commodity server.

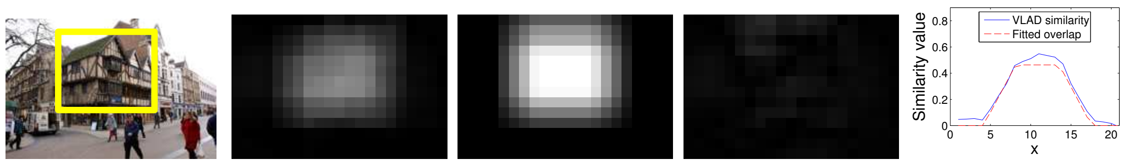
To assess the retrieval performance, additional ROI annotation is provided for the Oxford 5k dataset, as the original only specifies ROIs for the query images. Objects are deemed to be small if they occupy less than 300×300 pixels squared. Typical images in Oxford 5k are 1024×768, thus the threshold corresponds to the object occupying up to about 11% of an image. We measure the mean average precision for retrieving images containing these small objects using the standard Oxford 5k queries.

We compare to two baselines a single 128-D VLAD per image, and also a $14 \times 128 = 1792$-D VLAD. The latter is included for a fair comparison since MultiVLAD requires 14 times more storage. MultiVLAD achieves a mAP of 0.102, this outperforms the single 128-D VLAD descriptors, which only yield a mAP of 0.025, and also the 1792-D VLAD which obtains a mAP of 0.073, i.e. a 39.7% improvement. MultiVLAD consistently outperforms the 1792-D VLAD for thresholds smaller than 400^2, and then is outperformed for objects occupying a significant portion of the image (more than 20% of it).

Implementation details. The 3×3 grid is generated by splitting the horizontal and vertical axes into three equal parts. To account for potential featureless regions near image borders (e.g. the sky at the top of many images often contains no interest point detections), we adjust the outer boundary of the grid to the smallest bounding box which contains all interest points. All the multiple VLADs for an image can be computed efficiently through the use of an integral image of unnormalized VLADs.

5.4.1 Fine object localization

Given similarity scores between a query ROI and all the VLADs contained in the MultiVLAD of a result image, we show here how to obtain an estimate of the corresponding location within the result image. To motivate the method, consider figure 5.4 where, for each 200×200 subwindow of an image, VLAD similarities (to the VLAD of the target ROI) are compared to overlap (with the target ROI). The correlation is evident and we model this below using linear regression. The



(a) Image with the ROI (b) VLAD similarities (c) Region overlaps (d) Residuals (e) 1-D slice

Figure 5.4: **Variation of VLAD similarity with region overlaps.** (b) The value plotted at each point (x, y) corresponds to the VLAD similarity (scalar product between two VLADs) between the VLAD of the region of interest (ROI) in (a) and the VLAD extracted from the 200×200 pixel patch centred at (x, y). (c) The proportion of each patch from (b) that is covered by the ROI from (a). (d) Residuals obtained by a linear regression of (c) to (b). (e) A 1-D horizontal slice through the middle of (b) and (c). Note that residuals in (d) and (e) are very small, thus VLAD similarities are very good linear estimators of region overlap.

procedure is similar in spirit to the interpolation method of Torii et al. (2011) for visual localization.

Implementation details. A similarity score vector $\mathbf{s}$ is computed between the query ROI VLAD and the VLADs corresponding to the image tiles of the result image's MultiVLAD. We then seek an ROI in the result image whose overlap with the image tiles matches these similarity scores under a linear scaling. Here, overlap $\mathbf{v}(r)$ between an ROI r and an image tile is computed as the proportion of the image tile which is covered by the ROI. The best ROI, r_{best}, is determined by minimizing residuals as

$$r_{best} = \underset{r}{\operatorname{argmin}} \min_{\lambda} ||\lambda \mathbf{v}(r) - \mathbf{s}|| \tag{5.2}$$

where any negative similarities are clipped to zero. This approach yields the ROI in figure 5.5. Regressed overlap scores mimic the similarity scores very well, as shown by small residuals in figure 5.4d and 5.4e.

Note that given overlap scores $\mathbf{v}(r)$, which are easily computed for any ROI r, the

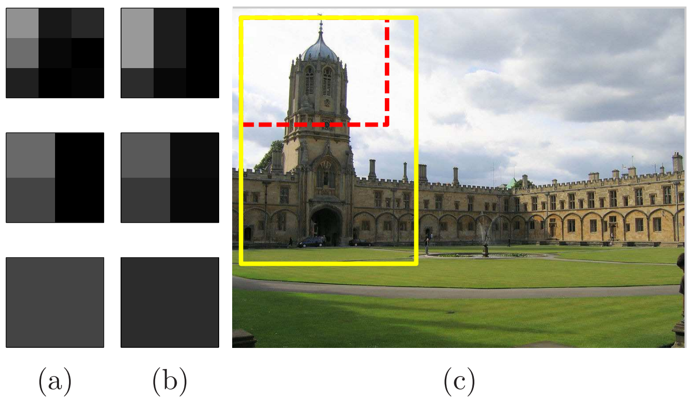


Figure 5.5: **Fine object localization.** (a) The similarity score between the query VLAD and all of the multiple extracted VLADs for the image shown in (c), i.e. $\mathbf{s}$ in the main text. (b) The overlap scores which approximate the real similarity scores in (a) the best, i.e. $\lambda\mathbf{v}(r_{best})$ in the main text. (c) The resulting image with estimated object position. The red dashed rectangle shows the tile corresponding to the largest similarity score (top left tile at the finest scale), the yellow rectangle shows the best estimate for the object location using our generative model.

inner minimization in (5.2) can be solved optimally using a closed form solution, as it is a simple least squares problem: the value of λ which minimizes the expression for a given r is $\lambda = \frac{\mathbf{s}^T\mathbf{v}(r)}{\mathbf{v}(r)^T\mathbf{v}(r)}$.

To solve the full minimization problem we perform a brute force search in a discretized space of all possible rectangular ROIs. The discretized space is constructed out of all rectangles whose corners coincide with a very fine (30 by 30) regular grid overlaid on the image, i.e. there are 31 distinct values considered for each of x and y coordinates. The number of all possible rectangles with non-zero area is $\binom{31}{2}^2$ which amounts to 216k.

The search procedure is very efficient as least squares fitting is performed with simple 14-D scalar product computations, and the entire process takes 14 ms per image on a single core 3 GHz processor.

Localization accuracy. To evaluate the localization quality the ground truth and estimated object positions and scales are compared in terms of the overlap score (i.e. the ratio between the intersection and union areas of the two ROIs), on the Oxford 5k dataset. In an analogous manner to computing mean average precision (mAP) scores for retrieval performance evaluation, for the purpose of localization evaluation the average overlap score is computed for each query, and averaged across queries to obtain the mean average overlap score.

For the region descriptors we use MultiVLAD descriptors with centre adaptation and intra-normalization, with multiple vocabularies trained on Paris and projected down to 128-D. This setup yields a mAP of 0.518 on Oxford 5k.

The *fine* localization method is compared to two baselines: *greedy* and *whole image*. The *whole image* baseline returns the ROI placed over the entire image, thus always falling back to the "safe choice" and producing a non-zero overlap score. For the *greedy* baseline, the MultiVLAD retrieval system returns the most similar tile to the query in terms of similarity of their VLAD descriptors.

The mean average overlap scores for the three systems are 0.342, 0.369 and 0.429 for the *whole image*, *greedy* and *fine* respectively; the *fine* method improves the two baselines by 25% and 16%. Furthermore, we also measure the mean average number of times that the centre of the estimated ROI is inside the ground truth ROI, and the *fine* method again significantly outperforms others by achieving a score of 0.897, which is a 28% and 8% improvement over *whole image* and *greedy*, respectively. Figure 5.6 shows a qualitative comparison of *fine* and *greedy* localization.

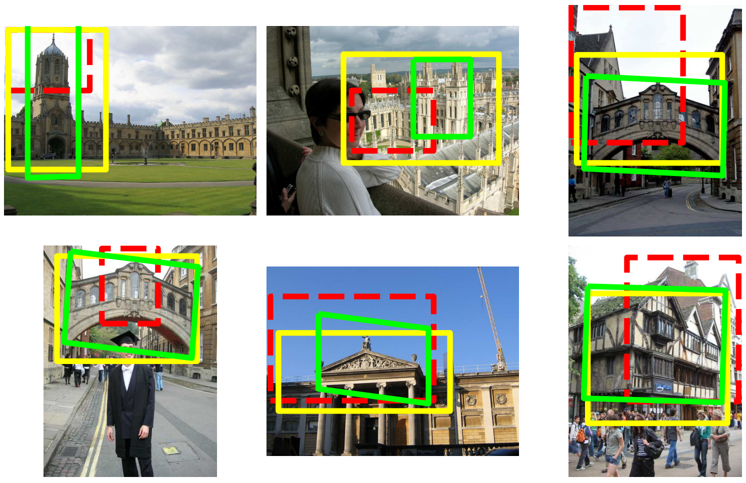


Figure 5.6: **Fine versus greedy localization.** Localized object: ground truth annotation (green); greedy method (red dashed rectangles); best location using the fine method of section 5.4.1 (yellow solid rectangles).

5.5 Results and discussion

In the following sections we compare our two improvements of the VLAD descriptor, namely cluster centre adaptation and intra-normalization, with the state-of-the-art. First, the retrieval performance of the full size VLAD descriptors is evaluated, followed by tests on more compact descriptors obtained using dimensionality reduction, and then the variation in performance using vocabularies trained on different datasets is evaluated. Finally, we report on large scale experiments with the small descriptors. For all these tests we used RootSIFT descriptors clustered into $k = 256$ coarse clusters, and the vocabularies were trained on Paris and Flickr60k for Oxford 5k(+100k) and Holidays(+Flickr1M), respectively.

Full size VLAD descriptors. Table 5.1 shows the performance of our method against the current state-of-the-art for descriptors of medium dimensionality (20k-D to 30k-D). Cluster centre adaptation followed by intra-normalization outperforms

Method	Holidays	Oxford 5k
BoW 200k-D (Jégou et al., 2012, Sivic and Zisserman, 2003)	0.540	0.364
BoW 20k-D (Jégou et al., 2012, Sivic and Zisserman, 2003)	0.452	0.354
Improved Fisher Vectors (Perronnin et al., 2010a)	0.626	0.418
VLAD (Jégou et al., 2010b)	0.526	-
VLAD+SSR (Jégou et al., 2012)	0.598	0.378
Improved det/desc: VLAD+SSR (Jégou et al., 2012)	-	0.532
This chapter: adapt+innorm (mean)	**0.646**	**0.555**
This chapter: adapt+innorm (single best)	**0.653**	**0.558**

Table 5.1: **Full size image descriptors (i.e. before dimensionality reduction): comparison with state-of-the-art.** Image descriptors of medium-dimensionality (20k-D to 32k-D) are compared in terms of retrieval performance (mAP) on the Oxford 5k and Holidays benchmarks. Reference results are obtained from the paper of Jégou et al. (2012). For fair comparison, we also include our implementation of VLAD+SSR using the detector (Perďoch et al., 2009) and descriptor (section 4.2) which give significant improvements on the Oxford 5k benchmark. The mean results are averaged over four different runs (corresponding to different random initializations of k-means for vocabulary building), and the single best result is from the vocabulary with the highest mAP.

all previous methods. For the Holidays dataset we outperform the best method (improved Fisher Vectors (Perronnin et al., 2010a)) by 3.2% on average and 4.3% in the best case, and for Oxford 5k we achieve an improvement of 4.3% and 4.9% in the average and best cases, respectively.

Small image descriptors (128-D). We employ the state-of-the-art method of Jégou and Chum (2012) (Multivoc) which uses multiple vocabularies to obtain multiple VLAD (with SSR) descriptions of one image, and then perform dimensionality reduction, using PCA, and whitening to produce very small image descriptors (128-D). We mimic the experimental setup of Jégou and Chum (2012), and learn the vocabulary and PCA on Paris 6k for the Oxford 5k tests. For the Holidays tests they do not specify which set of 10k Flickr images are used for learning the PCA. We use the last 10k images from the Flickr1M (Jégou et al., 2008) dataset.

Method	Holidays	Oxford 5k
GIST (Jégou et al., 2012)	0.365	-
BoW (Jégou et al., 2012, Sivic and Zisserman, 2003)	0.452	0.194
Improved Fisher Vectors (Perronnin et al., 2010a)	0.565	0.301
VLAD (Jégou et al., 2010b)	0.510	-
VLAD+SSR (Jégou et al., 2012)	0.557	0.287
Multivoc-BoW (Jégou and Chum, 2012)	0.567	0.413
Multivoc-VLAD (Jégou and Chum, 2012)	0.614	-
Reimplemented Multivoc-VLAD (Jégou and Chum, 2012)	0.600	0.425
This chapter: adapt+innorm	**0.625**	**0.448**

Table 5.2: **Low dimensional image descriptors: comparison with state-of-the-art.** 128-D dimensional image descriptors are compared in terms of retrieval performance (mAP) on the Oxford 5k and Holidays benchmarks. Most results are obtained from the paper of Jégou et al. (2012), apart from the recent multiple vocabulary (Multivoc) method (Jégou and Chum, 2012). The authors of Multivoc do not report the performance of their method using VLAD on Oxford 5k, so we report results of our reimplementation of their method.

As can be seen from table 5.2, our methods outperform all current state-of-the-art methods. For Oxford 5k the improvement is 5.4%, while for Holidays it is 1.8%.

Effect of using vocabularies trained on different datasets. In order to assess how the retrieval performance varies when using different vocabularies, we measure the proportion of the ideal mAP (i.e. when the vocabulary is built on the benchmark dataset itself) achieved for each of the methods.

First, we report results on Oxford 5k using full size VLADs in table 5.3. The baselines (VLAD and VLAD+SSR) perform very badly when an inappropriate (Flickr60k) vocabulary is used achieving only 68% of the ideal performance for the best baseline (VLAD+SSR). Using adapt+innorm, apart from improving mAP in general for all vocabularies, brings this score up to 86%. A similar trend is observed for the Holidays benchmark as well (see figure 5.2).

Method \vocabulary	Ox5k	Paris	Flickr60k
VLAD	0.519	0.508 (98%)	0.315 (61%)
VLAD+SSR	0.546	0.532 (97%)	0.374 (68%)
VLAD+adapt	0.519	0.516 (99%)	0.313 (60%)
VLAD+adapt+SSR	0.546	0.541 (99%)	0.439 (80%)
VLAD+adapt+innorm	**0.555**	**0.555 (100%)**	**0.478 (86%)**

Table 5.3: **Effect of using different vocabularies for the Oxford 5k retrieval performance.** Column one is the ideal case where retrieval is assessed on the same dataset as used to build the vocabulary. Full size VLAD descriptors are used. Results are averaged over four different vocabularies for each of the tests. The proportion of the ideal mAP (i.e. when the vocabulary is built on Oxford 5k itself) is given in brackets.

We next report results for 128-D descriptors where, again, in all cases Multivoc (Jégou and Chum, 2012) is used with PCA to perform dimensionality reduction and whitening. In addition to the residual problems caused by an inconsistent vocabulary, there is also the extra problem that the PCA is learnt on a different dataset. Using the Flickr60k vocabulary with adapt+innorm for Oxford 5k achieves 59% of the ideal performance, which is much worse than the 86% obtained with full size vectors above. Despite the diminished performance, adapt+innorm still outperforms the best baseline (VLAD+SSR) by 4%. A direction of future research is to investigate how to alleviate the influence of the inappropriate PCA training set, and improve the relative performance for small dimensional VLAD descriptors as well.

Large scale retrieval. With datasets of up to 1 million images and compact image descriptors (128-D) it is still possible to perform exhaustive nearest neighbour search. For example, in (Jégou and Chum, 2012) exhaustive search is performed on 1 million 128-D dimensional vectors reporting 6 ms per query on a 12 core 3 GHz machine. Scaling to more than 1 million images is certainly possible using efficient

Method	Holidays + Flickr1M	Oxford 105k
BoW (Jégou et al., 2012, Sivic and Zisserman, 2003) (200k-D)	0.315*	-
VLAD (Jégou et al., 2010b) (8192-D)	0.320*	-
VLAD (Jégou et al., 2010b) (8192-D ANN)	0.241	-
VLAD+SSR (Jégou et al., 2012) (96-D)	0.310*	-
VLAD+SSR (Jégou et al., 2012) (8192-D ANN)	0.370	-
Reimplemented Multivoc-VLAD (Jégou and Chum, 2012) (128-D)	0.370	0.354
This paper: adapt+innorm (128-D)	**0.378**	**0.374**

Table 5.4: **Large scale retrieval: comparison with state-of-the-art.** Mean average precision values estimated from graphs in corresponding papers are marked with an asterisk (*). Methods which use approximate nearest neighbour search are marked with *ANN*. The performance of the reimplemented Multivoc-VLAD method (Jégou and Chum, 2012) is reported as the original paper does not state the mAP for either of the datasets.

approximate nearest neighbour methods.

The same 128-D descriptors (adapt+innorm VLADs reduced to 128-D using Multivoc) are used as described above, results are reported in table 5.4. On Oxford 105k we achieve a mAP of 0.374, which is a 5.6% improvement over the best baseline, being (our reimplementation of) Multivoc VLAD+SSR. There are no previously reported results on compact image descriptors for this dataset to compare to. On Holidays+Flickr1M, adapt+innorm yields 0.378 compared to the 0.370 of Multivoc VLAD+SSR; while the best previously reported mAP for this dataset is 0.370 (using VLAD+SSR with full size VLAD and approximate nearest neighbour search (Jégou et al., 2012)). Thus, we set the new state-of-the-art on both datasets here.

5.6 Conclusions and recommendations

We have presented three methods which improve standard VLAD descriptors over various aspects, namely cluster centre adaptation, intra-normalization and Multi-VLAD.

Cluster centre adaptation is a useful method for large scale retrieval tasks where image databases grow with time as content gets added. It somewhat alleviates the influence of using a bad visual vocabulary, without the need of recomputing or storing all local descriptors.

Intra-normalization was introduced in order to fully suppress bursty visual elements and provide a better measure of similarity between VLAD descriptors. It was shown to be the best VLAD normalization scheme. However, we recommend intra-normalization always be used in conjunction with a good visual vocabulary or with centre adaptation (as intra-normalization is sometimes outperformed by SSR when inconsistent clusters are used and no centre adaptation is performed). Although it is outside the scope of this chapter, VLAD with intra-normalization also improves image classification performance over the original VLAD formulation.

The two methods set a new state-of-the-art over all benchmarks investigated here: Oxford 5k and Holidays for both mid-dimensional (20k-D to 30k-D) and small (128-D) descriptors; and for Oxford 105k and Holidays+Flickr1M benchmarks for small (128-D) descriptors.

Finally, we have also presented a MultiVLAD method for retrieving and localizing objects that only extend over a small part of an image.

5.7 Impact

The work described in this chapter was published in CVPR 2013. Intra-normalization of VLAD vectors was developed concurrently by Tolias et al. (2013); the work was published later in 2013.

Chapter 6

Using Multiple Query Images to Boost Recall

The objective of this chapter is to retrieve all images containing a specific object in a large scale image dataset. This is a problem that has seen much progress and success over the last decade (chapters 2 and 4), with the caveat that the starting point for the search has been a single query image of the specific object of interest. In this work we make two changes to the standard approach: first, our starting point for specifying the object is text, as we are interested in probing data sets to find known objects; and second, and more importantly for the development of novel algorithms, we search the dataset using multiple image queries and collate the results into a single ranked list.

The limitation of current approaches used to boost recall, which are based on query expansion (QE, section 2.1.4) within the data set, is that they rely on the query to yield a sufficient number of high precision results in the first place. In more detail, in QE an initial query is issued, using only the query image, and confident matches, obtained by spatial verification, are used to re-query. There are three problems with this approach: firstly, it is impossible to gain from QE if the initial query fails. Secondly, if the dataset does not contain many images of the queried object QE cannot boost performance. Finally, it is not possible to obtain images from different views of the object as these are never retrieved using the initial query, for example querying using an image of a building façade will never yield results of its interior.

More generally current BoW retrieval systems miss images that differ too much from the query in aspect (side vs front of a building), age (antiquarian photos may be missed if too much has changed between the target image and query), weather conditions, extreme scale changes, etc. Using multiple images of the object to query the database naturally alleviates to some extent all of these problems.

One of the principal contributions of this chapter is an algorithm to overcome these current shortcomings by combining multiple queries in a principled manner (section 6.1). The other principal contribution is the implementation of a real time demonstration system which generates query images automatically starting from text using Google image search (section 6.2.3).

Related work. In content-based image retrieval (CBIR) for categories (but not for specific objects) it is quite common to use a set of images to represent a query specified by text. A standard method is to obtain a set of images from a labelled corpus corresponding to that query (Heller and Ghahramani, 2006) or training images from a web search (Fergus et al., 2005, Torresani et al., 2010). Other standard approaches in CBIR can also result in a set of images representing the query: in relevance feedback the user selects from a set of images proposed from the target corpus, e.g. in the PicHunter system (Cox et al., 2000); in query expansion the original text query can be enhanced (e.g. by synonyms) and thereby result in multiple queries; one form of query expansion is to simply issue new queries using high ranked images from an initial search, a form of blind relevance feedback.

Many methods for combining (or fusing) ranked lists have been developed, these can either use only the rank of the items in the list (e.g. Borda count by Aslam and Montague (2001)), or the score as well if this is available (Shaw and Fox, 1994).

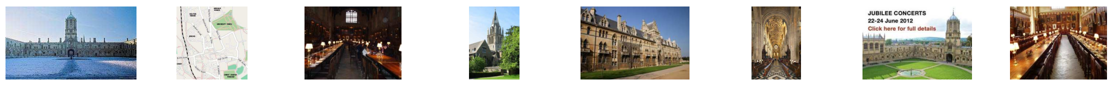
6.1 Retrieval using multiple query images

A question arises as to how to use multiple query images (the query set), as current systems only issue a single query at a time. We propose five methods for doing this; methods (i) and (ii) use the query set jointly to issue a single query, while methods (iii)-(v) issue a query for each image in the query set and combine the retrieved results. The five methods are described next.

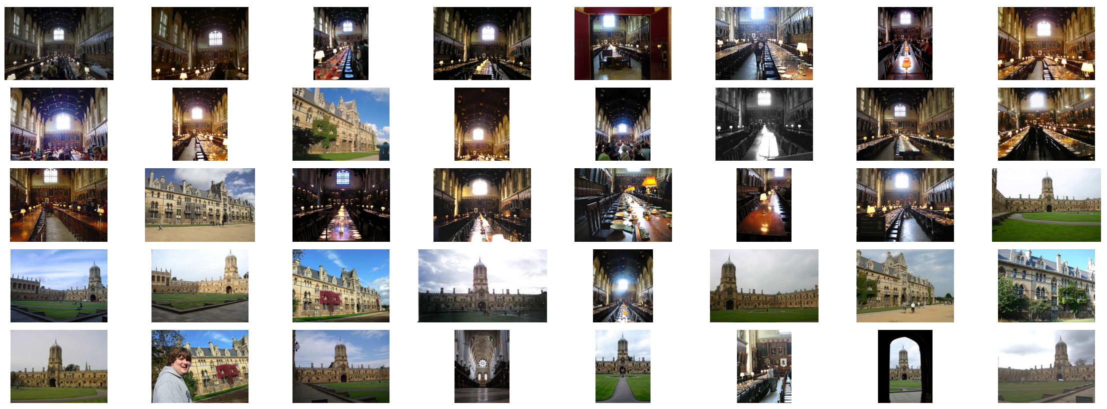
6.1.1 Retrieval methods

(i) Average query (Joint-Avg). Similar to the average query expansion method of Chum et al. (2007b), the bag-of-words representations of all images in the query set are averaged together. The average BoW vector is used to query the database by ranking images based on the tf-idf score.

(ii) SVM over all queries (Joint-SVM). Similar to the discriminative query expansion method (section 4.3), a linear SVM is used to discriminatively learn a



(a) Top 8 Google Image results for the textual query "Christ Church, Oxford"



(b) Top 40 retrieved results from the Oxford 5k dataset for the query "Christ Church, Oxford"

Figure 6.1: **Multiple query retrieval.** Images downloaded from Google using the "Christ Church, Oxford" textual query (a) are used to retrieve images of Christ Church college in the Oxford Buildings dataset (b). All the top 40 results of (b) do show various images of Christ Church (the dining hall, tourist entrance, cathedral and Tom tower). This illustrates the benefit of issuing multiple queries in order to retrieve all images of the queried object. Note that the noise in images retrieved from Google (the second image in (a) shows a map of Oxford) did not affect retrieval.

weight vector for visual words online. The query set BoWs are used as positive training data, and BoWs of a random set of 200 database images form the negative training data. The weight vector is then used to efficiently rank all images in the database.

(iii) Maximum of multiple queries (MQ-Max). A query is issued for each BoW vector in the query set independently and retrieved ranked lists are combined by scoring each image by the maximum of the individual scores obtained from each query.

(iv) Average of multiple queries (MQ-Avg). Similar to (iii) but the ranked lists are combined by scoring each image by the average of the individual scores obtained from each query.

(v) Exemplar SVM (MQ-ESVM). Originally used for classification (Malisiewicz et al., 2011), this method trains a separate linear SVM for each positive example. The score for each image is computed as the maximal score obtained from the SVMs.

6.1.2 Spatial reranking

Precision of a retrieval system can be improved by reranking images based on their spatial consistency with the query (section 2.1.3). Since spatial consistency estimation is computationally relatively costly, only a short-list of top ranked results is reranked. We use the spatial reranking method of Philbin et al. (2007) which reranks images based on the number of visual words consistent with an affine transformation (inliers) between the query and the database image.

Here we explain how to perform spatial reranking when multiple queries are used. For fair comparison of different methods it is important to fix the total number of spatial transformation estimations, we fix it to $R = 200$ per image in the query set of size N.

For methods *Joint-Avg* and *Joint-SVM* which perform a single query each, reranking is performed on the top R results. Images are ranked based on the average number of inliers across images in the query set. The number of spatial transformation estimations is thus $N \times R$.

For methods *MQ-Max*, *MQ-Avg* and *MQ-ESVM* which issue N queries each, reranking is performed for each query independently before combining the retrieved lists. For a particular query (one of N), reranking is done on the top R results using only the queried image. The number of spatial transformation estimations is thus, again, $N \times R$.

6.2 Implementation description

6.2.1 Standard BoW retrieval system

We have implemented the standard framework of Philbin et al. (2007) with some recent improvements that are discussed next. RootSIFT (section 4.2) descriptors are extracted from the affine-Hessian interest points, we use the recent implementation of the affine-Hessian feature detector (Mikolajczyk and Schmid, 2004b) by Perďoch et al. (2009) as it was shown to yield superior retrieval results. The descriptors are quantized into 1M visual words obtained using approximate k-means. Given a single query, the system ranks images based on the *term frequency inverse document frequency* (tf-idf) score (Sivic and Zisserman, 2003). Spatial reranking is performed on the top 200 tf-idf results using an affine transformation (Philbin et al., 2007) as described above.

6.2.2 Implementation details for multiple query methods

Here we give implementation details for the proposed methods (section 6.1). For the discriminative approaches (*Joint-SVM* and *MQ-ESVM* methods), the query set forms the positive training examples, while the negative set comprises 200 random database images. For training of a linear SVM classifier we use LIBSVM (Chang and Lin, 2011). The learnt weight vector is used to efficiently rank all images in the database based on their signed distance from the decision boundary. This can be done efficiently using the inverted index in the same way as when computing the tf-idf score, as both operations correspond to computing the scalar product between a weight vector and the BoW histograms of the database images. In order for retrieval to be fast, the learnt weight vector should be sparse. To ensure this we use the same approach as in (section 4.3), namely, the BoW vectors of negative images are truncated (and renormalized) to only include words that appear in at least one positive example.

For the *MQ-ESVM* case, as in (Malisiewicz et al., 2011), scores of individual SVMs have to be calibrated so that they can be compared with each other. This is done by fitting a sigmoid function to the output of each SVM individually (Platt, 1999), to try to map scores to 0 and 1 for negatives and positives, respectively. For the negative data required for calibration we use a set of 200 random images (different from the one used in exemplar SVM training), while for calibration positives we use the spatially verified positives for the given query. Note that it is not possible to evaluate *MQ-ESVM* without spatial reranking, as spatial transformations need to be estimated for the calibration procedure.

6.2.3 Building a real-time system

We have built a system which can respond to user text queries in real-time. After a user enters the query text, a textual Google image search is performed using the publicly available API provided by Google. Each of the top retrieved results, we use eight, is processed independently in a separate thread – the image is downloaded and a bag-of-visual-words description is obtained as discussed in section 6.2.1. Then, the processed query set is used to present the user with a ranked list of results obtained by using one of the methods introduced in section 6.1. Note that the methods which issue multiple queries and then merge the retrieved results (*MQ*-) can be easily parallelized as each query can be executed in an independent thread.

The entire process from typing words to retrieving relevant images takes less than 10 seconds. The bottle-neck is the Google API call which can take up to 3 seconds, along with downloading images from their locations on the internet. The actual querying, once the query set BoWs are computed, takes a fraction of a second.

6.3 Evaluation and Results

In this section we assess the retrieval performance of our multiple query methods by comparing them to a standard single query system, and compare them to each other.

6.3.1 Datasets and evaluation procedure

The retrieval performance of proposed methods is evaluated using standard and publicly available image and video datasets, we briefly describe them here.

Oxford Buildings. This dataset is described in section 3.1.1. 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 the retrieval performance is measured in terms of mean average precision (mAP). The standard evaluation protocol needs to be modified for our task as it was originally set up to evaluate single-query methods. We perform 11 queries, one per each predefined landmark; the performance is still measured using mAP.

Our methods are evaluated in two modes of operation depending on the source of the query set: one using the five predefined queries per landmark (Oxford queries, OQ), and one using the top 8 Google image search results for the landmark names (Google queries, GQ), chosen by the user to make sure the images contain the object of interest. The images in the Oxford building dataset were obtained by crawling Flickr, so we append a "-flickr" flag to the textual Google image search in order to avoid downloading exactly the images from the Oxford dataset which would artificially boost our performance.

TrecVid 2011. This dataset contains 211k keyframes extracted from 200 hours of low resolution footage used in the TrecVid 2011 known-item search challenge Paul et al. (2011) (the IACC.1.B dataset). As there is no ground truth available for this dataset we only use it to assess the retrieval performance qualitatively.

6.3.2 Baselines

Due to the lack of multiple query methods, comparison is only possible to methods which use a single image to query. For the Oxford queries (OQ) case the queries are the 55 predefined ones for the dataset. The two proposed baselines use exactly the same descriptors and vocabulary as our multiple query methods.

Single query. A natural baseline to compare to is the system of Philbin et al.

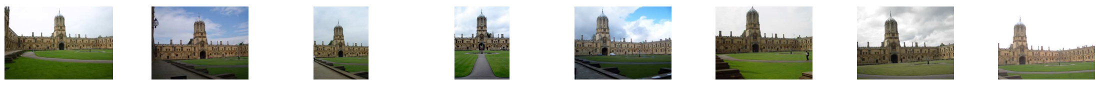
(2007) with extensions of section 6.2.1. For the Google queries (GQ) case the query is the top Google image result which contains the object of interest.

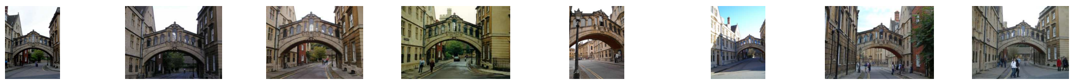
Best single query. The *single query* method is used to rank images using each query from the query set (the same query sets are used as for our multiple query methods) and the best performing query is kept. This method cannot be used in a real-world system as it requires an oracle (i.e. looks up ground truth).

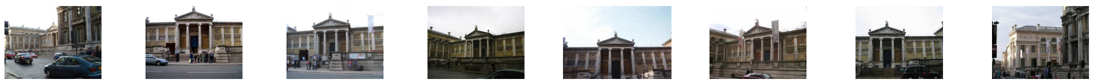
6.3.3 Results and discussion

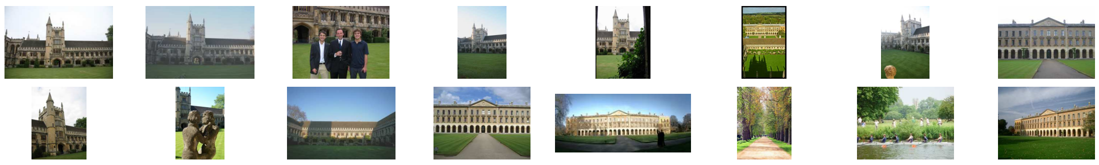
Figure 6.2 shows a few examples of textual queries and the retrieved results. Note the ability of the system to retrieve specific objects (e.g. the Tom Tower of Christ Church college in figure 6.2a) as well as sets of relevant objects (e.g. different parts of Christ Church college in figure 6.1) without explicitly determining the specific/general mode of operation.

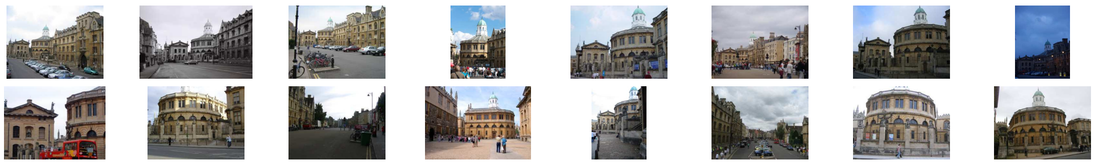
Table 6.1 shows the retrieval performance on the Oxford 105k dataset. It can be seen that all the multiple query methods are superior to the "single query" baseline, improving the performance by 29% and 52% for the Oxford queries and Google queries (with spatial reranking), respectively. It is clear that using multiple queries is indeed very beneficial as the best performance using Oxford queries (0.937) is better than the best reported result using a single query (0.891 achieved in section 4.5.2); it is even better than the state-of-the-art on a much easier Oxford 5k dataset (section 4.5.2: 0.929). All the multiple query methods also beat the "best single query" method which uses ground truth to determine which one of the images from the query set is best to be used to issue a single-query.


(a) Tom Tower, Christ Church, Oxford


(b) Bridge of Sighs, Oxford


(c) Ashmolean Museum, Oxford


(d) Magdalen College, Oxford


(e) Broad Street, Oxford

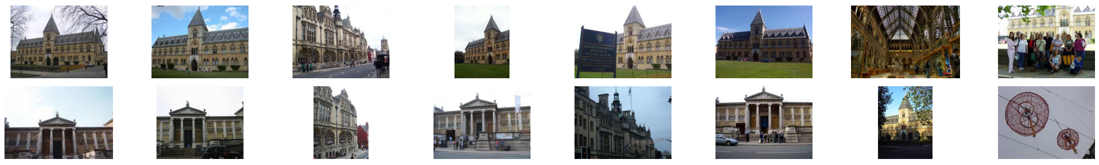

(f) Museum, Oxford

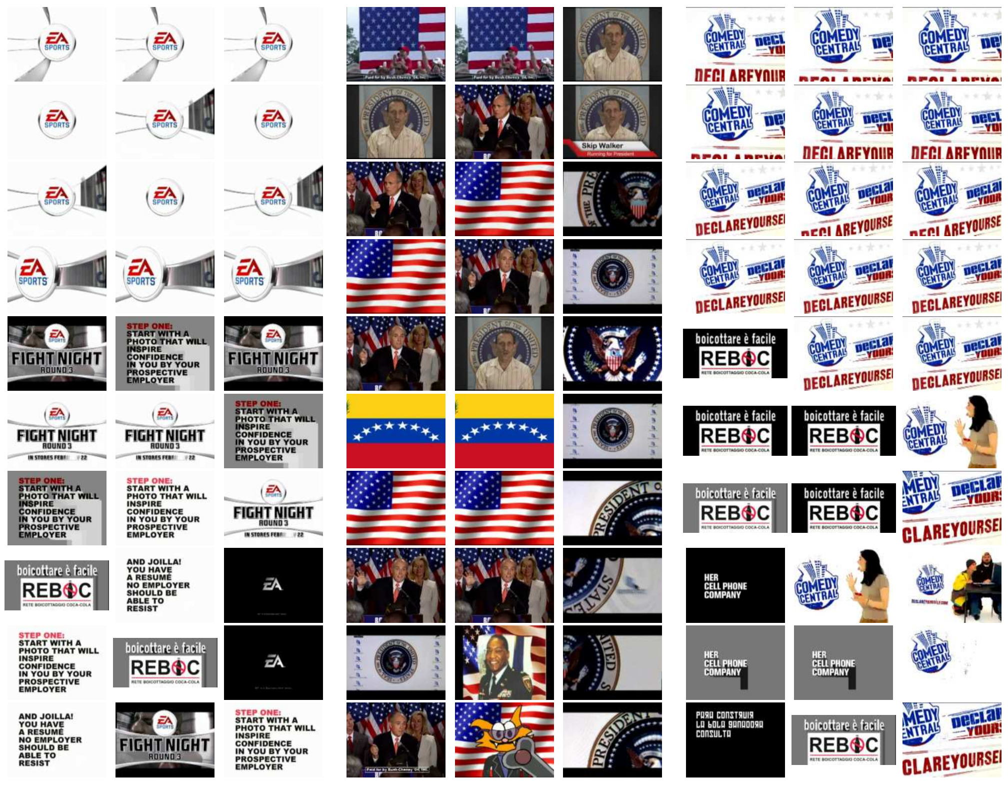
Figure 6.2: **Query terms and top retrieved images from the Oxford 5k dataset.** The captions show the textual queries used to download images from Google to form the query set. The top 8 images were used, without any user feedback to select the relevant one; the results are generated with the *MQ-Max* method. Specific (a-c) and broad (d-f, figure 6.1) queries are automatically handled without special considerations; note that (a) is a more specific version of the query in figure 6.1. (f) searching for "Museum, Oxford", which is a broader query than (c), yields in the top 16 results photos of three Oxford museums and a photo from the interior of one of them.

	Google queries (GQ)		Oxford queries (OQ)	
	Without SR	With SR	Without SR	With SR
Single query	0.464	0.575	0.622	0.725
Best single query ("cheating")	0.720	0.792	0.791	0.864
Joint-Avg	0.834	0.873	0.886	0.933
Joint-SVM	0.839	0.875	0.886	0.926
MQ-Max	0.746	0.850	0.826	0.929
MQ-Avg	0.834	0.868	0.888	0.937
MQ-ESVM	N/A	0.846	N/A	0.922

Table 6.1: **Retrieval performance (mAP) of the proposed methods on the Oxford 105k dataset.** SR stands for spatial reranking. The "Oxford queries" (OQ) and "Google queries" (GQ) columns indicate the source of query images, the former being the 5 predefined query images and the latter being the top 8 Google images which contain the queried object. The details of the evaluation procedure, baselines and proposed methods are given in sections 6.3.1, 6.3.2 and 6.1, respectively. All proposed methods significantly outperform the "single query" baseline, as well as the artificially boosted "best single query" baseline.

From the quantitative evaluation it is clear that multiple query methods are very beneficial for achieving higher recall of images containing the queried object, however it is not yet clear which of the five proposed methods should be used as all of them perform very well on the Oxford 105k benchmark. Thus, we next analyse the performance of various methods qualitatively on the TrecVid 2011 dataset, and show three representative queries and their outputs in figure 6.3.

The clear winner is the *MQ-Max* method – this is because taking the maximal score of the retrieved lists enables it to rank an image highly based on a strong match with a single query image from the query set. The other two methods which average the scores down-weight potential challenging examples even if they match very well with one query image, thus only retrieving "canonical" views of an object. For example, all methods work well for the "EA sports logo" query (figure 6.3a) and



(a) EA sports logo (b) Presidential seal (c) Comedy central logo

Figure 6.3: **Multiple query retrieval on TrecVid 2011 dataset.** (a)-(c) show three different textual queries and retrieval results. Within one example, each column shows a ranked list of images (sorted from top to bottom) for a particular method. Left, middle and right columns show *Joint-SVM*, *MQ-Avg* and *MQ-Max* methods, respectively. *MQ-Max* is clearly the superior method.

retrieve the common appearances of the object (represented in 7 out of 8 images in the query set). However, only the *MQ-Max* method manages to find the extra two "unusual" and challenging examples of the logo in silver on a black background.

It is also interesting to compare *MQ-Avg* with *Joint-SVM* in order to understand whether it is better to issue multiple queries and then merge the resulting ranked lists (the *MQ-* approaches), or to have a joint representation of the query set and perform a single query (the *Joint-* approaches). Figure 6.3 shows that the "multiple queries" approach clearly performs better. The argument for this is similar to the arguments we made in favour of the *MQ-Max* method, namely that it is beneficial to be able find close matches to each individual query image. Furthermore, we believe that the spatial reranking procedure (section 6.1.2) of the *MQ-* methods is more efficient – estimation of a spatial transformation between a query image and a short-list is conducted on the short-list obtained from the corresponding query image, while for the *Joint-* methods, where only a single "global" short-list is available, many attempts at spatial verification are wasted on using irrelevant query images. Another positive aspect of the "multiple queries" methods is that they can be parallelized very easily – each query is independent and can be handled in a separate parallel thread.

We note that the discriminative methods perform slightly better than the corresponding non-discriminative ones, i.e. *Joint-SVM* and *MQ-ESVM* outperform *Joint-Avg* and *MQ-Max*, respectively. However, the difference in our examples was not significant, so due to ease of implementation we recommend the use of the non-discriminative methods.

Finally, taking all aspects into consideration, we conclude that the method of choice for multiple query retrieval is *MQ-Max*, where each image from the query set is queried on independently and max-pooling is applied to the retrieved sets of results.

6.4 Conclusions

We have investigated a number of methods for using multiple query images and find that approaches that issue multiple independent queries and combine the results outperform those that jointly model the query set and issue a single query. Of the multiple independent query methods *MQ-Max* was found to perform best in terms of retrieving the more unusual instances.

Also, we have built a system which can, in real-time, retrieve images containing a specific object from a large image database starting from a text query. Using Google image search (or Bing or Flickr image search etc) in this way to obtain sample query images opens up a very flexible way to immediately explore unannotated image datasets.

6.5 Impact

The work described in this chapter was published in BMVC 2012. Motivated by our approach, Chen et al. (2012) and Fernando and Tuytelaars (2013) also propose methods which use multiple query images for object retrieval. Fernando and Tuytelaars (2013) employ mining for representative visual patterns in the query image set. Their results show that our work performs very well compared to their highly complex method.

Chapter 7

Smooth Object Retrieval: Sculptures

Recognizing specific objects, such as buildings, paintings, CD covers etc is to some extent a solved problem – provided that they have a light coating of texture (section 2.1). However, as has been noted for quite some time (Mikolajczyk et al., 2005, Moreels and Perona, 2006), there are two classes of specific objects for which current methods fail completely: wiry objects (Carmichael and Hebert, 2003) and smooth (fairly textureless) objects (Neven, Google, 2011). This chapter addresses the smooth object class.

Our goal in this chapter is to raise smooth objects to the first class status that lightly textured specific objects have: to be able to recognize these objects under change of lighting; and under change of scale and viewpoint; and to be able to build scalable retrieval systems. In this work we consider smooth objects that are three dimensional (3D), and will use sculptures as our illustration. We are interested in

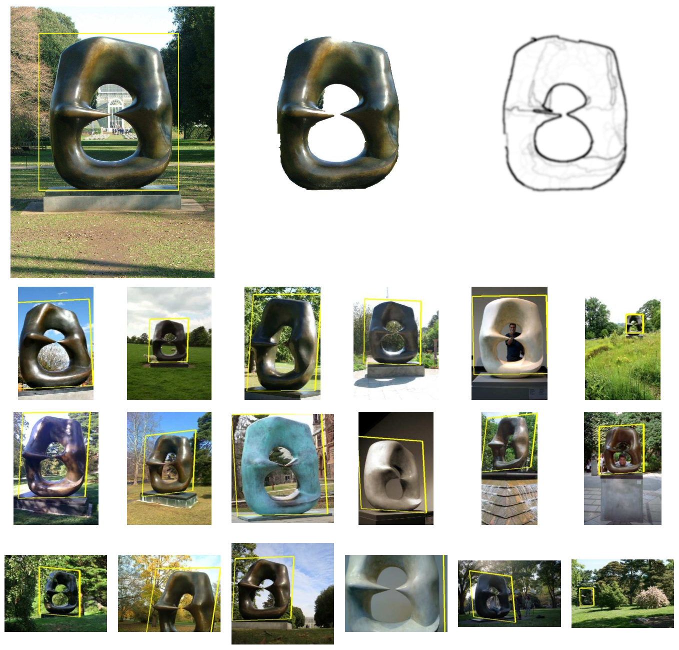


Figure 7.1: **Smooth sculpture retrieval using a bag of boundaries (BoB).** Top row: (left) a sculpture by Henry Moore selected by a user-outlined query; (middle) automatically segmented sculpture (section 7.1.1); (right) the boundary and internal edges are represented using semi-local descriptors (section 7.1.2) and indexed using a BoB (section 7.2). Bottom three rows: 18 of the retrieved images in rank order (before any false positives) showing the BoB's robustness to scale, viewpoint, lighting, colour and material variations. Note, at least seven different instances of the sculpture are retrieved, made out of at least three different materials.

matching objects of the same shape, and for sculptures, where the same form may be produced multiple times, this means that two instances may have the same shape but differ in size and even material. For example, Henry Moore routinely made the same sculptural form in bronze and marble.

3D smooth objects also bring with them the additional issue that their boundaries (internal and external) depend on viewpoint since they are defined by tangency with the line of sight (Koenderink, 1990). This means that the imaged shape can vary continuously with viewpoint, and we address this for the moment by a view based representation.

To this end we develop a new representation for smooth objects that encodes their boundaries (internal and external) both locally and at multiple scales (section 7.1.2). This representation is inspired by the shape context descriptors of Belongie and Malik (2002) and also by the silhouette representation used by Agarwal and Triggs (2006). We show that this representation is suitable for matching smooth 3D objects over scale changes, and is tolerant to viewpoint change and segmentation failures.

However, the representation cannot be employed directly for objects in an image due to the overwhelming number of edges and boundaries in the background (from clutter, trees, people etc). Instead it is first necessary to improve the signal to noise (where signal is the sculpture) by segmenting the image to isolate the sculpture as foreground. We show that this can be accomplished quite successfully using a combination of unsupervised segmentation into regions (Arbelaez et al., 2009) and supervised classification of the regions (Hoiem et al., 2005) (section 7.1.1).

In section 7.2 we show that the boundary representation can be vector quantized into a form suitable for large scale retrieval in a manner analogous to visual words (section 2.1.1).

Related work

Early approaches to 2D object recognition focused on objects which were fairly textureless (e.g. spanners). Recognition was based on shape, via explicit shape matching (Grimson and Lozano-Pérez, 1987, Huttenlocher and Ullman, 1987) or comparing geometric invariants (Rothwell et al., 1992). Many current approaches that target smooth objects focus on fast detection by matching templates (Hinterstoisser et al., 2012, Hsiao and Hebert, 2013), edgelets (short straight edge segments) chains (Damen et al., 2012, Ferrari et al., 2006), or learnt shape models (Ferrari et al., 2010). They require many training images for each object taken from different viewpoints (e.g. Hinterstoisser et al. (2012) use 2000) and induce large storage requirements. Therefore, these methods are not directly applicable to large scale retrieval. However, they can potentially be used as a post-processing step for removing false matches, akin to spatial reranking (section 2.1.3).

Sculpture naming

In section 7.5 we describe a retrieval based method for automatically determining the title and sculptor of an imaged sculpture. This is a useful problem to solve, but also quite challenging given the variety in both form and material that sculptures can take, and the similarity in both appearance and names that can occur.

Our approach is to first visually match the sculpture and then to name it by harnessing the meta-data provided by Flickr users. To this end we make the following three contributions: (i) we show that using two complementary visual retrieval methods (one based on visual words, the other on boundaries) improves both retrieval and precision performance; (ii) we show that a simple voting scheme on the tf-idf weighted meta-data can correctly hypothesize a sub-set of the sculpture name (provided that the meta-data has first been suitably cleaned up and normalized); and (iii) we show that Google image search can be used to query expand the name sub-set, and thereby correctly determine the full name of the sculpture.

7.1 Sculpture representation

In this section we describe the representation of the object boundary by a set of semi-local descriptors. In order to obtain this representation from a (cluttered) image it is first necessary to partition the image into sculpture and non-sculpture regions. This segmentation has two benefits: it improves the signal to noise, and also it provides an approximate scale for the descriptor computation. We begin with the segmentation, and then develop the boundary representation in section 7.1.2.

7.1.1 Segmentation

The goal of the segmentation is to separate sculptures as foreground from the background. This is quite a challenging task since sculptures can be made from various materials including bronze, marble and other stone, and plastics. Their surface can be natural or finished in some way such as polishing (for stone) or buffered (for bronze) or even a light texture (e.g. deliberate chisel textures). The colour can include white, brown, specular highlights (on bronze), and even green (for algae or moss on outdoor installations). These must be distinguished from backgrounds that can have quite similar appearances including textureless sky, pavements and walls.

To achieve this segmentation we employ a supervised classification approach, engineering a feature vector that represents the appearance, shape and position of sculptures (relative to the image boundary). The segmentation proceeds in three steps: first, an over-segmentation of the image into regions (super-pixels); second, each super-pixel is classified into foreground (sculpture) or background to give an initial segmentation; and third, post-processing is used to filter out small connected

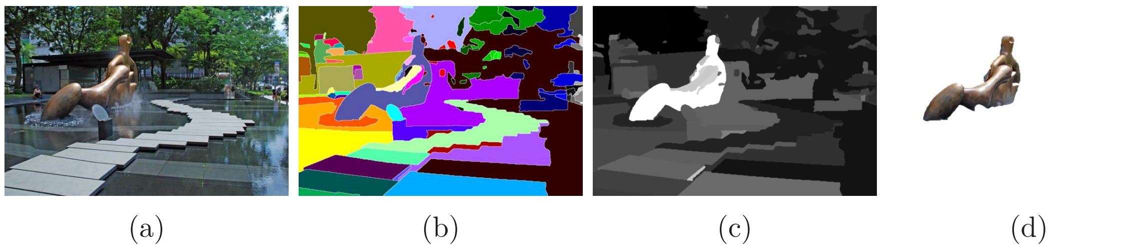


Figure 7.2: **Automatic sculpture segmentation.** (a) An image from the Sculptures 6k test set. (b) Over-segmentation (Arbelaez et al., 2009). (c) Classifier output, scaled for display. (d) Final segmentation.

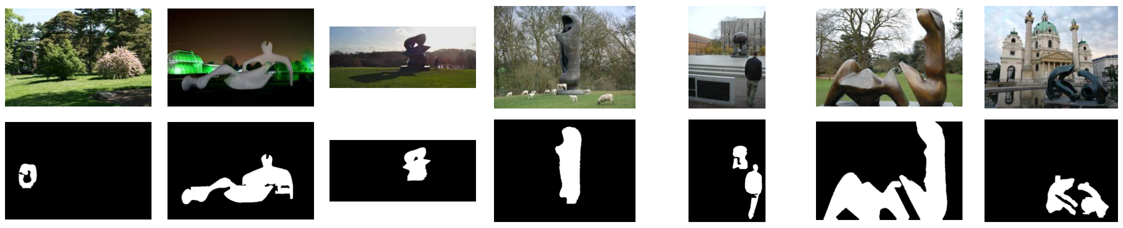


Figure 7.3: **Examples of automatic sculpture segmentation.** Top row shows images from the Sculptures 6k test set, bottom row shows the fully automatic segmentation.

components and obtain the final segmentation. Figure 7.2 illustrates these steps. Note, we do not attempt to group the super-pixels but simply classify them independently. We now describe these steps in more detail.

1. Super-pixels. We use the method and code from Arbelaez et al. (2009) which generates a hierarchy of regions based on the output of the gPb contour detector (Maire et al., 2008). This provides a partition of the image into a set of closed regions for any threshold. We use a threshold of 50 (out of 255) which yields about 58 super-pixels per image on average. A typical example of the super-pixels is shown in figure 7.2b.

2. Classification. For training and testing of the super-pixel classifier, 300 random images are selected from the Sculptures 6k training set, and segmented into super-pixels. The images are divided randomly into a training and validation set, each containing 150 images. Each super-pixel is then manually labelled into one of three classes: contained within a sculpture (positive example), not containing any sculpture pixels (negative example), or containing both sculpture and non-sculpture pixels (ignored completely). Small segments (less than 50×50 pixels) are also ignored in order to emphasize the correct classification of large segments.

Each super-pixel is described by a 3208 dimensional feature vector. This represents the appearance (colour, texture), shape and position of the segment (see below). A linear SVM classifier is trained on the annotated super-pixels from the training images, and its performance measured on the validation images. The histogram parts of the feature vector are compared using a χ^2 kernel, but using the efficient linear approximation of Vedaldi and Zisserman (2010) enables a linear SVM to be used for these as well. The linear SVM leads to both fast training and testing.

The feature vector consists of: (i) the median gradient magnitude – this feature is typically very informative as its value is usually small for smooth object segments; (ii) four binary features indicating whether the segment is touching one of the image boundaries – in order to more easily distinguish sky, ceiling, wall and floor from smooth sculptures; (iii) colour represented by vector-quantized (using k-means, dictionary size 1600) HSV, and the mean HSV of the segment – this helps to identify the materials that sculptures are made of; and (iv) a bag of SIFT (Lowe, 2004) visual words computed densely at multiple scales (dictionary size 1600, image patches with sides of 16, 24, 32 and 40, spacing of 2 pixels) – used for texture description, and also useful to identify sculpture material.

The super-pixel classifier has an accuracy of 96% on the training images, and 87% on the validation images. This results in a segmentation overlap score (intersection over union) of 0.78 on the training and 0.70 on the validation images.

3. Post-processing. The positive super-pixels are grouped using connected components, and small connected components (less than 50×50 pixels) of the foreground are removed. This does not significantly change the mean overlap score, but it removes many 'floating' and erroneous segments.

Examples of automatically segmented images are given in figure 7.3. These results show quantitatively and qualitatively that the automatic segmentation succeeds in its main objective of significantly increasing the signal (sculpture) to noise (other clutter) ratio.

7.1.2 Boundary descriptor

We develop a new shape descriptor suited for smooth object representation. Constructing such a descriptor is a challenging task as it needs to represent shape rather than texture or colour, be robust enough to handle lighting, scale and viewpoint changes, but simultaneously discriminative enough to enable object recognition. Additionally it should be extracted locally in order to be robust to occlusions and segmentation failures.

For an object, two types of descriptors are computed by sampling the object boundaries (internal and external) at regular intervals in the manner of Belongie and Malik (2002). They are (i) a HoG (Dalal and Triggs, 2005) descriptor, and (ii) a foreground mask occupancy grid. The scale of the descriptor is determined from the scale of

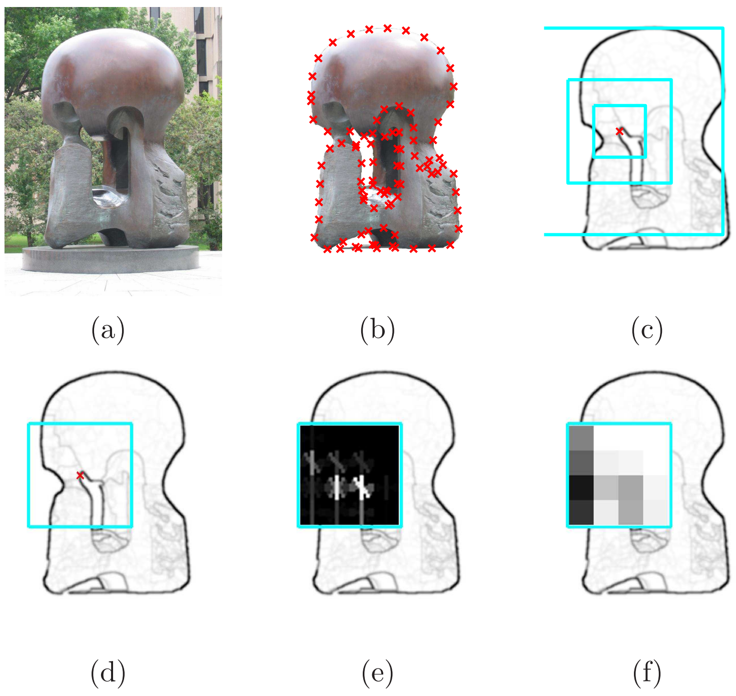


Figure 7.4: **Boundary descriptor extraction.** (a) original image; (b) automatically segmented image (section 7.1.1) overlaid with the centres for the boundary descriptors; (c) boundary image with three different scaled descriptors centred at the same point; (d) support region for a single descriptor; (e) HOG descriptor for the region in (d); (f) occupancy grid for the region in (d). See section 7.1.2 for details.

the object. In order to represent the boundary information locally (e.g. the curvature, junctions) and also the boundary context (e.g. the position and orientation of boundaries on the other side of the object), the descriptors are computed at multiple scales. We use HoG computed on the gPb image here (rather than shape-context or SIFT for example) as we wish to represent both the position and orientation of the boundaries, and also their magnitude. Figure 7.4 illustrates the entire descriptor extraction process.

In order to extract this representation from an image, it is first segmented into foreground (sculpture) and background as described above in section 7.1.1; and then the descriptor centres are obtained by sampling prominent foreground object boundaries and internal gPb edges at uniform intervals. The multiple scales of the descriptor are computed relative to the size of the foreground segmentation.

Implementation details

The first part of the descriptor uses 4×4 HoG cells, each containing 8×8 pixels (i.e. gPb patches are scaled to 32×32 pixels for HoG computation) and contrast insensitive spatial binning into 9 orientations, making the HoG part of the boundary description 324 dimensional (9 blocks each with 2×2 cells with 9 orientations). The HoG descriptor is L2 normalized in order to be able to compute similarities using Euclidean distance.

The second part of the descriptor is a 4×4 occupancy grid, where the value of a cell represents the proportion of pixels belonging to the foreground. The Hellinger kernel is used to compute similarities between these descriptors, i.e. the 16 dimensional descriptor is L1 normalized followed by square rooting each element thus producing a L2 normalized vector; the similarities are then computed using Euclidean distance.

The two parts of the descriptor are simply concatenated together making a 340 dimensional L2 normalized vector (the L2 norm being equal to 2).

The descriptor centres are obtained by sampling prominent (stronger than 30 out of 255) foreground object boundaries at uniform intervals. The interval lengths are determined as the maximum of 10 pixels and 1/50 of the foreground object perimeter. The descriptor scales are set to be 1, 4 and 16 times 1/10 of the foreground object area. Note that even though the largest scale descriptor is 1.6 larger than the object it does not in general cover the entire object as it is often computed at the external boundary, objects are usually elongated or not convex. The number of extracted descriptors per image is 450 on average.

Descriptor properties and matching

As the descriptor operates purely on boundary and segmentation data it is fairly unaffected by light, colour and texture changes. Scale invariance is obtained by computing the descriptor at multiple scales relative to the size of the foreground object they belong to. The descriptor is not rotation invariant but this can easily be alleviated by orienting the patch according to the boundary curve tangent.

The descriptors are matched between images using Euclidean distance. Note, even though three descriptors (at different scales) are computed at each of the sampled boundary points, these descriptors are matched independently in the subsequent processing – we do not explicitly enforce consistency between them. Figure 7.5 shows two examples of correctly matched sculptures, while figure 7.6 shows three typical retrieval results. Apart from illustrating robustness to lighting, colour, texture and scale differences, they also show that the descriptor is quite insensitive to significant viewpoint changes. There are three main reasons for this behaviour, firstly, the description is semi-local and even under significant viewpoint change it can be expected that some boundaries (and thus the descriptors) remain unchanged. Secondly, even though the object silhouette can change drastically between views, internal edges, which our method takes into account, can be unaffected. Finally, HoG cells inherently allow for some deformation in the position and orientation of boundaries.

The semi-local descriptors can be matched directly between object boundaries. However, in the following section we describe how this set of descriptors is rep-

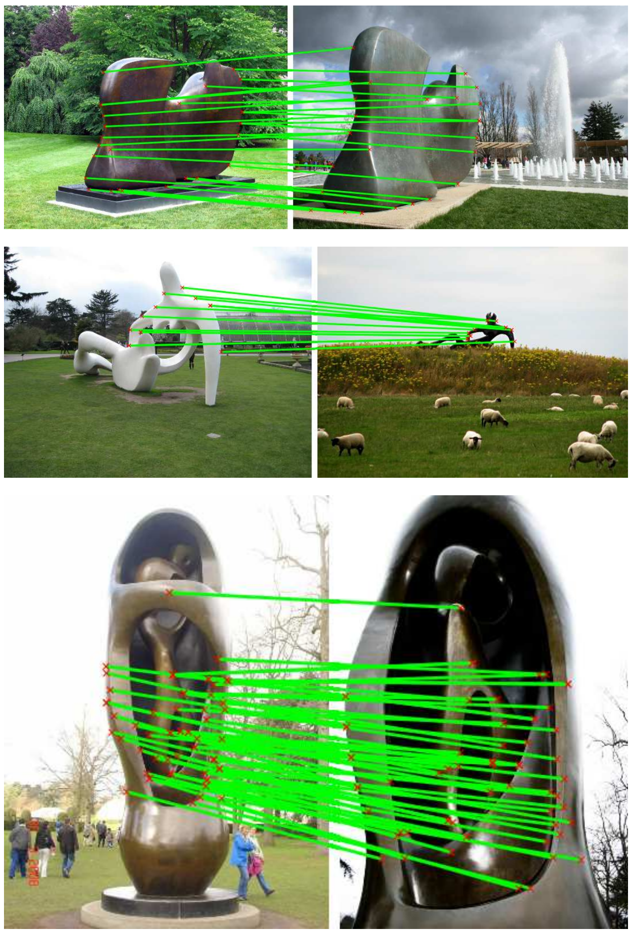


Figure 7.5: **Boundary descriptor matches.** Three examples of correctly matched sculptures using the semi-local boundary descriptor (section 7.1.2). Significant lighting, scale and viewpoint changes are handled well. Note that the images contain different sculpture instances but the shapes are identical and are successfully matched. Matches shown are after spatial verification (section 7.2).

resented as a histogram (by vector quantization and counting) in the manner of Agarwal and Triggs (2006).

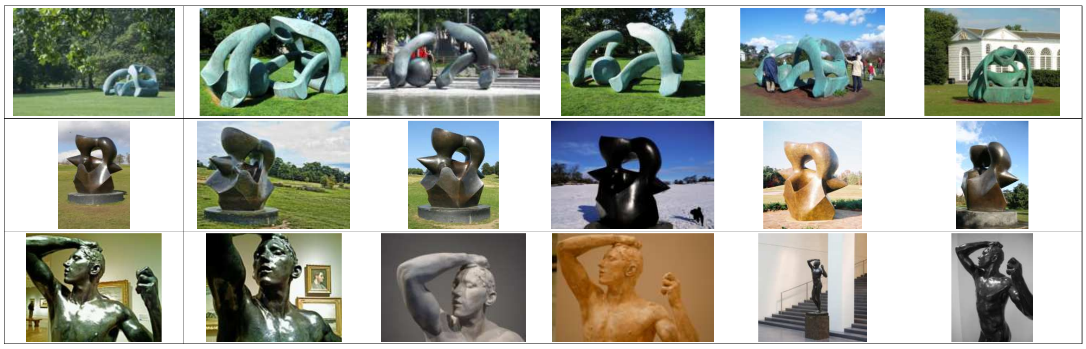


Figure 7.6: **Viewpoint invariance.** Each row shows one query (the left image), and the other five images are samples from the retrieved results. These results are typical, and demonstrate the viewpoint tolerance of the semi-local boundary representation.

7.2 Retrieval procedure

Here we use the standard retrieval pipeline of Philbin et al. (2007), but instead of representing the image as bag-of-visual-words (BoW) based on SIFT descriptors (Lowe, 2004) computed at affine covariant regions (Mikolajczyk et al., 2005), we develop a *bag-of-boundaries (BoB)* representation. For each image, boundary descriptors are extracted as described in section 7.1.2 and vector quantized using k-means; a histogram of these quantized descriptors (which we will also refer to as 'words') is then used to represent an image. Note, this is a bag representation as no information about the spatial position of the descriptors is recorded in the histogram. A query BoB is compared to other BoBs in the dataset using the standard tf-idf (section 2.1.1) measure. The tf-idf scores can be computed efficiently for each image in the database using an inverted index, which enables real-time retrieval in large databases.

As shown in (Philbin et al., 2007) spatial verification and re-ranking of the top tf-idf retrieved results can be done efficiently and proves to be useful as it improves precision by ensuring spatial consistency between query and retrieved images. We adopt the same model for the geometry relation, namely an affine transformation. However, as the objects of interest are highly three-dimensional the affine model of the transformation is only approximate here, so only a very loose affine homography is fitted (i.e. large reprojection errors are tolerated) in order not to reject correct matches. We follow the procedure of Philbin et al. (2007) of first using a single (boundary) word match to determine a restricted affine transformation (in this case translation and scaling only), followed by fitting a full affine transformation to the inliers.

7.2.1 Implementation details

The BoB vocabulary is obtained from the Sculpture 6k test set descriptors. The test set generates 1.4M descriptors, we chose the vocabulary size to be 10k.

We spatially verify the top 200 results using a loose affine homography, tolerating reprojection errors of up to a 100 pixels. We also propose a new scoring system where the score of the geometrically verified image for a given query is computed as follows:

$$\text{score} = \text{tf-idf} + \alpha n + \beta \frac{n}{n_q} \frac{n}{n_r} \tag{7.1}$$

where n_q and n_r are the number of words in the query and result images, respectively, and n is the number of verified matches. The proposed score is a generalization of the commonly used scoring scheme of (Philbin et al., 2007) which corresponds to $\alpha = 1$ and $\beta = 0$. Our system accounts for the fact that images with many features are likely to have many spatially verified words and removes the bias from these images by considering the number of matches relative to the total number of features in the image.

7.3 Dataset and evaluation

Sculptures 6k: We have collected a new image dataset in order to evaluate performance of smooth object retrieval methods. The dataset was obtained in a similar manner to the widely used Oxford Buildings dataset (section 3.1.1): images containing sculptures were automatically downloaded from Flickr[1] using queries such as “Henry Moore Reclining Figure”, “Henry Moore Kew Gardens” and “Rodin Thinker”. The dataset has 6340 high resolution (1024×768) images.

The dataset is split equally into a train and test set, each containing 3170 images. For each set 10 different Henry Moore sculptures are chosen as query objects, and for each of these objects 7 images and query regions are defined, thus providing 70 queries for performance evaluation purposes. None of the 10 training set sculptures is present in the test set, whilst for the 10 test set sculptures mostly these are not present in the training set though there are a few occurrences as some images contain more than one sculpture (e.g. images taken in a museum). As well as the images containing these 10 sculptures in each set there are many images containing other sculptures or indeed no sculptures at all; figure 7.7 shows a random sample of the images. These images act as distractors in retrieval. A sub-set of the test set queries is shown in figure 7.8.

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

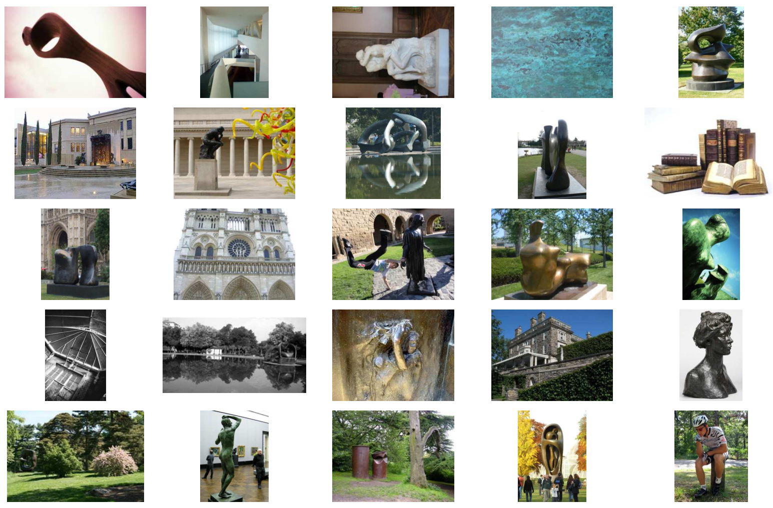


Figure 7.7: **Random samples from the Sculptures test dataset.** Note the variety of sculptures. Many of the images do not contain a sculpture while some contain people imitating the pose of a sculpture (e.g. bottom right image where a man is impersonating Rodin's *Thinker*).

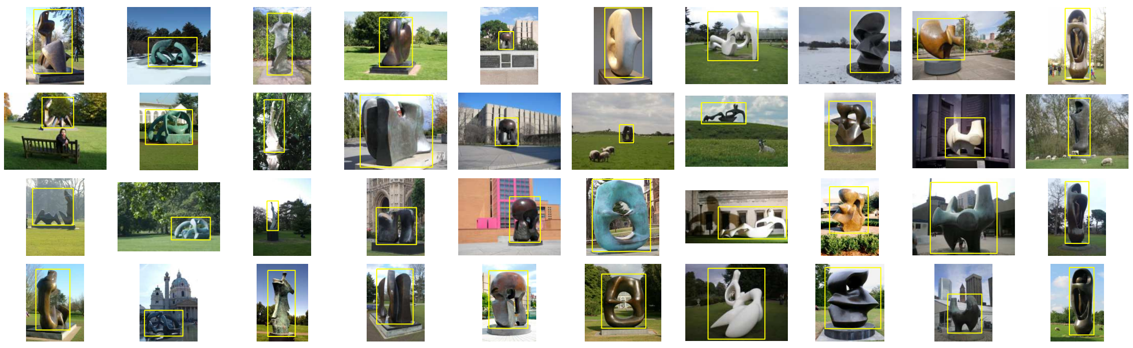


Figure 7.8: **Test dataset query images.** 40 query images (out of 70) used for evaluation in the Sculptures 6k test dataset. Each column shows 4 (out of 7) query images for one sculpture. Note the large variations in scale, viewpoint, lighting, material and background.

For each query we have manually compiled the ground truth dividing all images into *Positives*, *Negatives* and *Ignores*: (i) *Negative* – No part of the queried sculpture is present. (ii) *Positive* – More than 25% of the queried sculpture surface is visible. (iii) *Ignore* – Less than 25% of the queried sculpture surface is visible, but the queried sculpture is present. Note that our definition of the *same sculpture* relationship requires two sculptures to have identical shapes, however it does not require them to be the *same instances* – they can be constructed of different materials, made in different sizes and displayed at different locations. Sculptures are 'highly' three-dimensional, unlike the building facades used in the Oxford Buildings dataset. For this reason the ground truth matches are *view specific* and vary over the different queries of the same sculpture. For example, it is unreasonable to expect to retrieve an image of a sculpture given an image taken from its opposite side. For each query the number of positive matches can vary from 5 to a maximum of 112, with a mean of 53.4.

The *Sculptures 6k* dataset with all the images and ground truth is available online[2].

Performance evaluation: As in the case of the Oxford Buildings dataset, retrieval quality is evaluated using mean average precision (mAP) over all the queries. As in the INRIA Holidays (section 3.1.3) evaluation, the query image is not counted as a positive return (it is in the Oxford Buildings evaluation). In the mAP computation *Ignores* are not counted as positive or negative.

7.4 Results

To evaluate the performance of smooth object retrieval methods we follow the procedure outlined in section 7.3. The mean average precision (mAP) is computed over 70 queries on the test dataset.

Due to the lack of smooth object retrieval systems we use the standard affine-

[2]http://www.robots.ox.ac.uk/~vgg/research/sculptures/

Method name	Spat. rerank	mAP	A.q.t.
Baseline 1 (Philbin et al., 2007)		0.080	0.05 s
Baseline 1 (Philbin et al., 2007)	√	0.094	0.30 s
Baseline 2 (background removed)		0.081	0.03 s
Baseline 2 (background removed)	√	0.086	0.11 s
BoB without segmentation		0.253	0.01 s
BoB without segmentation	√	0.323	0.16 s
BoB with segmentation		0.454	0.01 s
BoB with segmentation	√	**0.502**	0.28 s

Table 7.1: **Retrieval performance.** Comparison of two baseline bag of visual word methods (section 7.4) and the bag-of-boundaries (BoB) method (section 7.2). Mean average precision (mAP) scores and average query times (A.q.t.) are shown. The mAP scores correspond to the best choice of parameters (vocabulary size and reranking parameters α, β) for each method individually.

Hessian/visual word system of Philbin et al. (2007) as a baseline (BL1). As a second baseline (BL2), we discard all visual words on the background (i.e. visual words are only included if their centres are in the automatically segmented foreground region). This is in order to give a fair comparison against our boundary representation which uses the foreground/background segmentation.

Retrieval performance

The mAP scores for the two baselines and our method are shown in table 7.1. As expected, there is a complete failure of the two baselines for smooth object retrieval. Note that BL2 perform slightly worse than BL1 (after spatial reranking with an affine homography) – this is due to the fact that many true positives in BL1 are actually obtained by matching the background of the sculpture installation instead of the actual queried sculptures. Note that none of the methods which usually improve retrieval performance can be hoped to help the two baselines: (i) query expansion (section 2.1.4) is only possible when the initial method yields high precision results which is certainly not the case here, (ii) soft vector quantization and (iii) learning a better vocabulary (section 2.1.1.3) both assume the descriptors to be appropriate for the task in hand which we demonstrate is not the case.

Our bag-of-boundaries (BoB) method proves to be very suitable for the task of smooth object retrieval, achieving more than a five fold increase in performance (0.502) over the best baseline (BL1, 0.094). The importance of the segmentation is shown by the 'with and without' comparison (i.e. in the 'with' case, only boundaries on the foreground region are used). There is 55% gain in performance for BoB when the foreground segmentation is used compared to using the entire image. On the other hand, the without segmentation performance is still quite respectable and demonstrates the robustness to background clutter. As would be expected, in some of the cases where automatic segmentation fails and the sculpture is assigned to background, the without case succeeds in retrieving the image. However, it is more prone to background clutter and less resistant to scale change as there is no scheme for automatic descriptor scale selection.

Examples of ranked retrieval results are given in figures 7.1, 7.6 and 7.9. They illustrate the appropriateness of the BoB system for the smooth object retrieval task as significant lighting, scale, viewpoint, colour and material differences are successfully handled.

Table 7.1 also gives the retrieval speed, tested on a laptop with a 2.67 GHz core i7 processor using only a single core. It can be seen that due to the inverted index implementation, the BoB representation enables real time retrieval to the same extent as visual words. The BoB representation is much sparser than the BoW (450 words per image for BoB compared to 2600 for BoW) making the entire storage

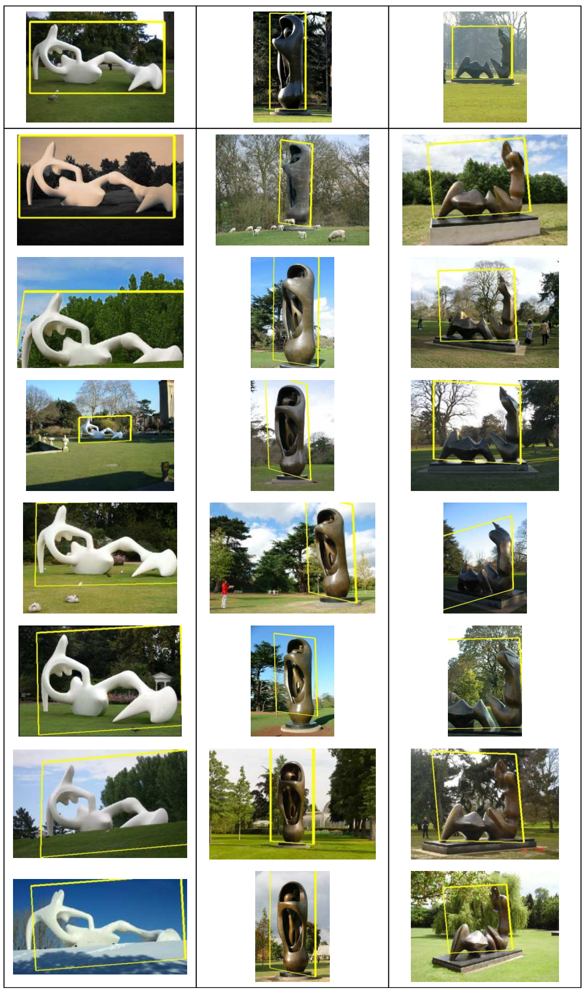


Figure 7.9: **Retrieval results.** Each column shows one retrieval result, the query image and ROI are shown in the first row, followed by the top 7 ranked retrieved images.

requirements for the system (inverted and forward indexes) a mere 20 MBs (in the BoW case this is 275 MBs). Our approach is thus much more scalable than the existing BoW ones as the BoB representation of up to 5 million images can fit into main memory on a system with 32 GB of RAM.

To further test the resistance to distractors, a larger scale retrieval test was performed by adding all 5062 images from the Oxford Buildings dataset to the testset.

The mAP performance only dropped to 0.451 (from 0.502) despite the variety of images in the distractor dataset.

Due to the semi-local HoG boundary description (figure 7.4) and the BoB representation, the matching is capable of handling the significant segmentation failures that are bound to happen in a fully automatic system. The semi-local property means that a proportion of the HoG descriptors computed on the boundary will still be valid (the proportion depending on the extent of the segmentation failure), and the BoB representation enables matches for images where only a subset of the quantized descriptors are in common. Thus, as can be seen in figure 7.10, retrieval can succeed both in the cases of under-segmentation (where HoG descriptors will be missing) and over-segmentation (where additional erroneous boundaries are generated).

Parameter and descriptor variation

The choice of descriptor scales is critical for retrieval performance as reducing the areas by a factor of 4 reduces the mAP from 0.502 to 0.404. The problem with using small descriptors is that they are too local thus mainly capturing the orientation of a single edge, which without surrounding boundary information is completely non-discriminative.

Not using descriptors centred on internal boundaries but keeping the internal boundary information in the remaining descriptors reduces the mAP to 0.469, while not taking internal boundaries into account at all decreases it further to 0.433. This proves that using internal boundaries is very beneficial for shape representation.

The system is quite insensitive to the number of HoG and occupancy grid cells, using a coarser grid (3×3) decreases the mAP from 0.502 to 0.485 while using a finer one (6×6) increases it slightly to 0.509. The slight increase in performance when

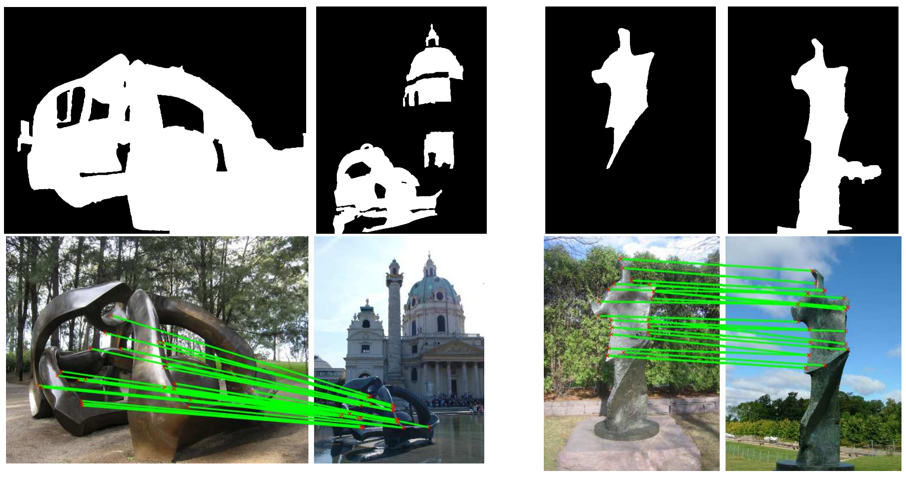


Figure 7.10: **Robustness to segmentation failures.** Two examples of correctly matched sculptures despite significant segmentation failures. In each case the segmentation is shown above the original image.

using a finer grid is not worth the large increase in descriptor dimensionality (from 340 to 936). Excluding the 16 dimensional occupancy grid part of the boundary descriptor decreases the mAP from 0.502 to 0.485, so it provides good value for the small descriptor dimensionality increase (from 324 to 340).

A main source of failure is due to the inevitable automatic segmentation mistakes, it is most prominent when the sculpture pixels in the query image are assigned to the background. This failure can be potentially alleviated by either segmenting the query image online or keeping descriptors for multiple alternative segmentations. The second failure mode is when all matched words occur spatially close to each other thus only effectively representing one part of the object which is not necessarily discriminative. A potential solution is to modify spatial verification to incorporate the information about the spatial distribution of matches, but this must be traded against robustness to occlusion and viewpoint change.

We investigate the effect of varying the reranking (section 7.2.1) parameter β for fixed $\alpha = 0$ on the mAP scores. The performance increases monotonically with β which effectively means the last portion of (7.1), which accounts for relative number of matched words, should dominate the reranking, while the tf-idf scores should be used for tie-breaking. Our reranking procedure, for the BoB method, always outperforms the reference reranking method (Philbin et al., 2007) which uses just the unnormalized number of inliers and tf-idf. When a 10 times smaller vocabulary is used the benefits of the proposed reranking method are even more apparent (0.449 compared to 0.391) – words are less discriminative allowing the reference reranking method to incorrectly verify images with many features more easily thus reducing its precision.

7.5 Name that sculpture

The goal of this section is to automatically identify both the *sculptor* and the name of the *sculpture* given an image of the sculpture, for example from a mobile phone. This is a capability similar to that offered by Google Goggles, which can use a photo to identify certain classes of objects, and thereby carry out a text based web search.

Being able to identify a sculpture is an extremely useful functionality: often sculptures are not labelled in public places, or appear in other people's photos without labels, or appear in our own photos without labels (and we didn't label at the time we took them because we thought we would remember their names). Indeed there are occasionally pleas on the web of the form "Can anyone help name this sculpture?".

Identifying sculptures is also quite challenging. Although Google Goggles can visually identify objects such as landmarks and some artwork, sculptures have eluded it to date (Neven, Google, 2011) because the visual search engine used for matching cannot handle smooth objects.

We divide the problem of identifying a sculpture from a query image into two stages: (i) visual matching to a large dataset of images of sculptures, and (ii) textual labelling given a set of matching images with annotations. Figure 7.11 shows an example. That we are able to match sculptures in images at all, for the first stage, is a result of combining two complementary visual recognition methods. First, the bag-of-boundaries method for recognizing 3D smooth objects from their outlines in cluttered images (section 7.2). Second, we note that there is still a role for interest point based visual matching as some sculptures do have texture or can be identified from their surroundings (which are textured). Thus we also employ a classical visual word based visual recognition system (section 2.1.1). The matching image set for the query image is obtained from the sets each of the two recognition systems returns (section 7.5.2).

The other ingredients required to complete the identification are a dataset of images to match the query image to, and annotation (of the sculptor and sculpture name) for the images of this dataset. For the annotated dataset we take advantage of the opportunity to harness the knowledge in social media sites such as Facebook and Flickr. As is well known, such sites can provide millions of images with some form of annotation in the form of tags and descriptions – though the annotation can often be noisy and unreliable (Quack et al., 2008). The second stage of the identification combines this meta-information associated with the matched image set in order to propose the name of the sculptor and sculpture. The proposed sculpture name is finally determined using a form of query expansion from Google image search.

The stages of the identification system are illustrated in figure 7.11. We describe the dataset downloaded from Flickr in section 7.5.1, and the method of obtaining the name from the meta-data and Google query expansion in section 7.5.3.

Others have used community photo collections to identify objects in images (Gammeter et al., 2009, Ivanov et al., 2010) and have dealt with the problems of noisy annotations (Jin et al., 2005, Wang et al., 2006). In particular, Gammeter et al. (2009) auto-annotated images with landmarks such as “Arc de Triomphe” and “Statue of Liberty” using a standard visual word matching engine. In (Gammeter et al., 2009), two additional ideas were used to resolve noisy annotations: first, the GPS of the image was used to filter results (both for the query and for the dataset); second, annotations were verified using Wikipedia as an Oracle. Although we could make use of GPS this has not turned out to be necessary as (i) sculptures are often sufficiently distinctive without it, and (ii) sculptures are sometimes moved to different locations (e.g. the human figures of Gormley’s “Event Horizon” or Louise Bourgeois’ “Maman”) and so using GPS might harm recognition performance. Similarly, using Wikipedia to verify sculpture matches has not been found to be necessary, and also at the moment Wikipedia only covers a fraction of the sculptures that we consider.

7.5.1 Sculpture naming dataset

The dataset provides both the library of sculpture images and the associated meta-data for labelling the sculptor and sculpture. A list of prominent sculptors was obtained from Wikipedia[3] (as of 24th November 2011 this contained 616 names).

[3]http://en.wikipedia.org/wiki/List_of_sculptors

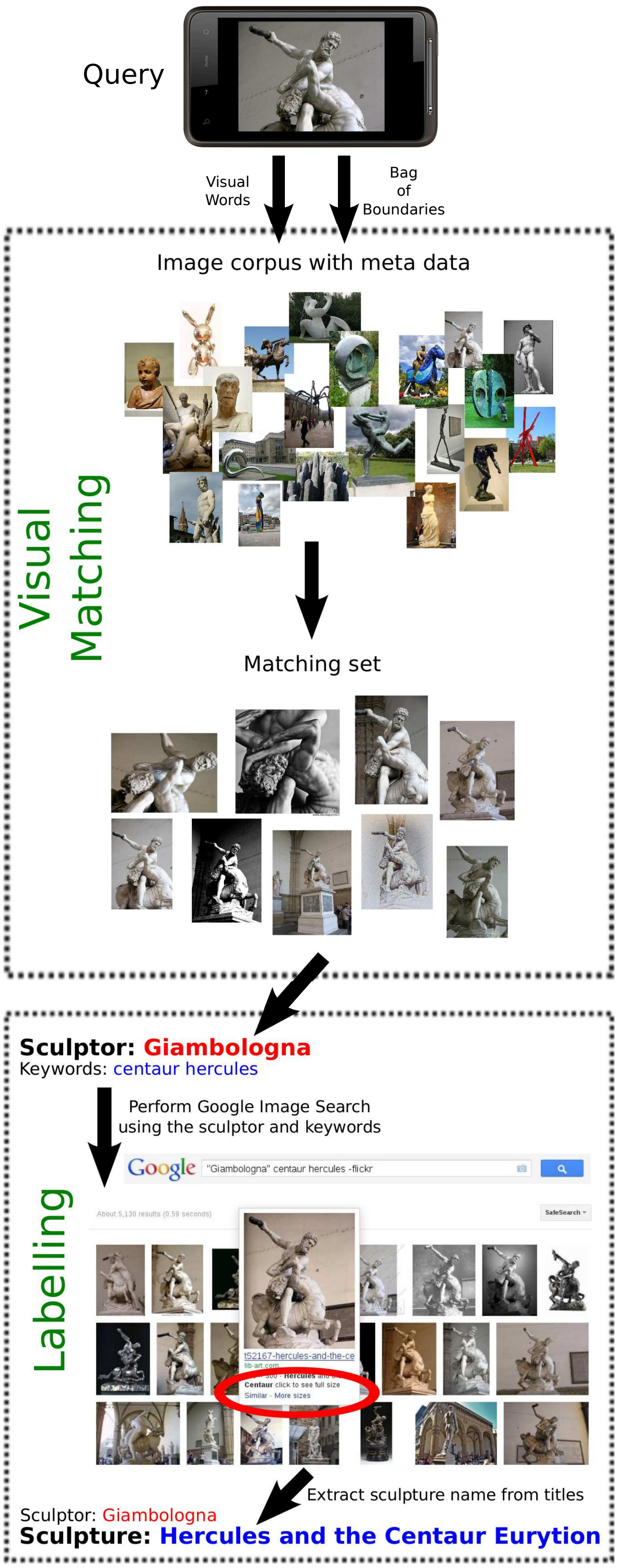

Query
Visual Words
Bag of Boundaries
Image corpus with meta data
Visual Matching
Matching set
Sculptor: Giambologna
Keywords: centaur hercules
Perform Google Image Search using the sculptor and keywords
Google
"Giambologna" centaur hercules -flickr
About 5,130 results (0.59 seconds)
SafeSearch
Labelling
Extract sculpture name from titles
Sculptor: Giambologna
Sculpture: Hercules and the Centaur Eurytion

Figure 7.11: Sculptor and sculpture identification: on-line system overview.

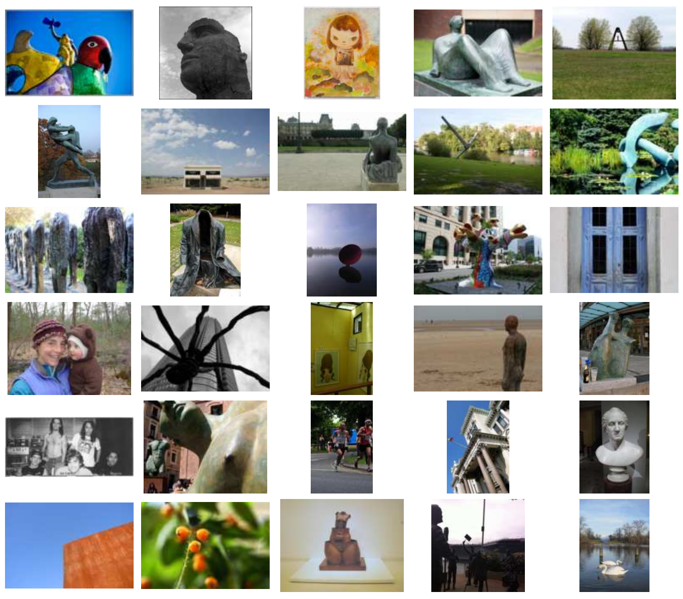


Figure 7.12: Random sample from the Sculptures 50k dataset.

This contains sculptors such as “Henry Moore”, “Auguste Rodin”, “Michelangelo”, “Joan Miró”, and “Jacob Epstein”. Near duplicates were removed from the list automatically by checking if the Wikipedia page for a pair of sculptor names redirects to the same entry. Only Michelangelo was duplicated (as “Michelangelo” and “Michelangelo Buonarroti”).

Flickr was queried using this list, leading to 50128 mostly high resolution (1024×768) images. Figure 7.12 shows a random sample. For each of the images textual meta data is kept as well. It is obtained by downloading the title, description and tags assigned to the image by the Flickr user who uploaded it. The textual query (i.e. sculptor name) used to retrieve an image is saved too. This forms the *Sculptures 50k dataset* used in this work.

Unlike the Sculptures 6k dataset (section 7.3) we did not bias this dataset towards smooth textureless sculptures.

7.5.2 Sculpture retrieval system

The first stage of the naming algorithm is to match the query image to those images in the *Sculptures 50k* that contain the same sculpture as the query. We briefly describe here the two complementary visual retrieval engines that we have implemented. In each case, a visual query is used as the input and the system returns a ranked list of matched images from the dataset, where the ranking is based on the number of spatial correspondences between the query and target image. The outputs of these two systems are then combined.

Bag-of-visual-words (BoW). We use a standard BoW retrieval system described in (Philbin et al., 2007) with affine-Hessian interest points (Mikolajczyk and Schmid, 2004b), RootSIFT descriptors (section 4.2), a vocabulary of 1M visual words obtained using approximate k-means, and spatial re-ranking of the top 200 tf-idf results using an affine transformation.

Bag-of-boundaries (BoB). We use the smooth object retrieval system described in section 7.2.

Combining BoB with BoW. The two described methods are complementary, BoW is well suited for retrieval of textured objects while BoB is adapted for smooth textureless objects defined mainly by their shape. Often only one of the systems is appropriate for a particular sculpture/query, but in a significant number of cases both systems manage to retrieve correct results; each of the cases can be observed in figure 7.14.

The two systems are combined by scoring each image by the maximum of the individual scores obtained from the two systems. In the situation where one system is capable of handling the query and the other is not, the latter system assigns low scores to all images while the former sets high scores to the relevant ones; the max combination rule thus correctly retrieves the relevant images by trusting the high scores of the former system. Note that our combination method merges the results of the two systems softly, i.e. no hard decisions are made about which system should be solely trusted for a particular query. This is because for some queries both systems are capable of functioning correctly, and the union of their matching sets (automatically obtained by the max combination method) is larger than any of the individual matching sets. We have not found it necessary to calibrate the scores obtained from the two systems.

The output of this stage is a *matching image set* which contains the highest ranked images of the combined list with score above a threshold of nine. Each image has an associated matching score and also the meta-data originally obtained from Flickr. This choice of threshold retains only the most reliable matches to the query. If this procedure yields no results, then the top match (highest ranked) is returned with low confidence.

7.5.3 Sculptor and sculpture identification

The goal of this work is to create a system which can automatically determine the sculptor and sculpture in a query photo. This is done by querying the Sculptures 50k database with the image, as described in section 7.5.2, and processing the textual meta information associated with the matching image set. Here we describe how to obtain the sculptor and sculpture names from this set of meta data.

7.5.3.1 Sculptor identification

It is simple to propose a candidate sculptor for each retrieved image: it is sufficient just to look up the textual query (i.e. sculptor name, see section 7.5.1) which was used to download that image from Flickr when the Sculptures 50k database was harvested. Given this set of putative sculptors we propose and compare two simple strategies to identify the actual sculptor of the query image: *Winner-Takes-All* and *Weighted Voting*.

Winner-Takes-All (WTA). The top match (highest ranked) is kept as the correct one and its sculptor identity is returned. Empirically, this scheme performs quite well, however it does have two shortfalls: it is prone to *label noise* and *retrieval failure*. In the *label noise* failure case the system cannot identify the sculptor correctly due to the mislabelled top match, which is a significant possibility when data is obtained in an unconstrained fashion, in our case from Flickr user annotations. *Retrieval failure* occurs if the top match is not actually of the same sculpture as the query. Both of these can be overcome to some extent by the following scheme.

Weighted Voting (WV). The scores of the top four images in the matching set are counted as weighted votes for the sculptor associated with that image; the sculptor with the largest sum of votes is chosen. This method can sometimes overcome both failure cases of the WTA scheme (label noise and retrieval failure) if the database contains more than one image of the same sculpture and they are correctly retrieved. As shown in section 7.5.4, this strategy outperforms Winner-Takes-All by 2%.

7.5.3.2 Sculpture identification

Unlike identifying the sculp*tor*, identifying the sculp*ture* requires finding distinctive words in the textual meta data associated with the matching image set. However, this data is variable, unstructured and quite noisy as it is supplied by Flickr users so it needs to be cleaned up and normalized. We first describe the filtering procedure that is performed off-line for data clean-up, and then the on-line sculpture naming applied to the matching image set.

1. Off-line: Meta data preprocessing. The data is cleaned up and normalized by the following steps. First, to reduce the problem of having different languages in the meta data, Microsoft's automatic translation API is used to detect the language and translate the text into English. This procedure overcomes sculptures being named in different languages, e.g. Rodin's "The Thinker" is also commonly referred to as "Le Penseur" in (the original) French.

Second, characters such as ,;:_-&/\()@ are treated as spaces in order to simplify word extraction and standardize the text. For example, Henry Moore's sculpture "Large Upright Internal External Form" contains the words "Internal External" which are also often found in variations such as "Internal-External" or "Internal/external"; all these cases are identical after the standardization step.

Only alphanumeric characters are kept and converted to lower case in order to simplify word correspondence. Of these, only words longer than 2 and shorter than 15 are kept so that typos, abbreviations and invalid words are filtered out. Some uninformative words are removed too, like "Wikipedia", "Wiki", "www", "com", "jpg", "img" etc. Also, only words which do not contain any digits are kept in order to filter out image file names often found in Flickr meta data, such as DSC12345, IMG12345, P12345, as well as the dates the photos were taken.

Lastly, the name of the sculptor is removed from the meta data in order to enable sculpture identification instead of just obtaining the sculptor name again.

2. On-line: sculpture naming. We start here with the meta-data associated with the matching image set for the query image. Only the meta-data from the images with the previously identified sculptor (section 7.5.3.1) is used in order to filter out potentially falsely retrieved images (i.e. those images that were in the original matching set, but do not contain the sculptor name selected by WTA or WV). There are two steps: (i) first, keywords, often containing the name or a part of the name, are identified, and second, the name is verified or corrected using Google by a form of query expansion.

The sculpture name, or particular words which can be used to uniquely identify the sculpture, are obtained by finding words which frequently occur in the titles and descriptions of the matching set, but are distinctive at the same time (for example typical stop-words such as “the” are common but not at all distinctive). This is achieved by the standard *term frequency inverse document frequency (tf-idf)* weighting, where each word is weighted by its frequency in the text and down-weighted by the logarithm of the number of documents in the entire database which contain it. We term the top scoring words *keywords*, and these identify the sculpture and are most commonly part of its name. However, further processing is required to obtain the full sculpture name from these keywords, for example it is required to put the words in the right order, add non-distinctive words as many sculpture names contain words like “the” and “and”, and correct possible mistakes.

Google based query expansion. The top two scoring keywords from the previous step, along with the name of the sculptor are used to issue a textual query on Google image search; the titles associated with the top Google image search results are then processed, as described next, to obtain the full sculpture name. The procedure is illustrated in figure 7.11, where the sculptor is identified as "Giambologna" and the top two keywords as "centaur" and "hercules", resulting in a Google image search query *"Giambologna" centaur hercules -flickr* ("-flickr" is added to obtain results independent of the Sculptures 50k dataset which is entirely downloaded from Flickr). The textual data associated with the top Google image search results is mined to obtain the full sculpture name "Hercules and the Centaur Eurytion".

The full sculpture name is obtained by "growing" the top scoring keyword using the titles associated with the top 15 Google image search results obtained via the Google API (note, only the titles from the Google image search API are used; the images themselves are not processed). The name is "grown" as follows: firstly, it is initialized to be the top keyword. Secondly, the name is iteratively expanded by the word which directly precedes or succeeds it in the most number of titles. In our running example (figure 7.11) the initial name is "centaur", the word "the" directly precedes it 8 times, "a", "with" and "beating" once each, and it is succeeded by "Nessus" trice and "Eurytion" twice; "the" thus has most support and is prefixed to the name to form the new one: "the centaur". Words are prefixed or appended to the name one by one; growing stops once a proposed name does not exist in more than one title, in order not to grow it to an entire title and overfit. The procedure yields many benefits: the name length is automatically determined (otherwise one would need to set a threshold on the tf-idf scores which is potentially hard to do), words are put in the correct order, new non-distinctive words like "the" and "and" which

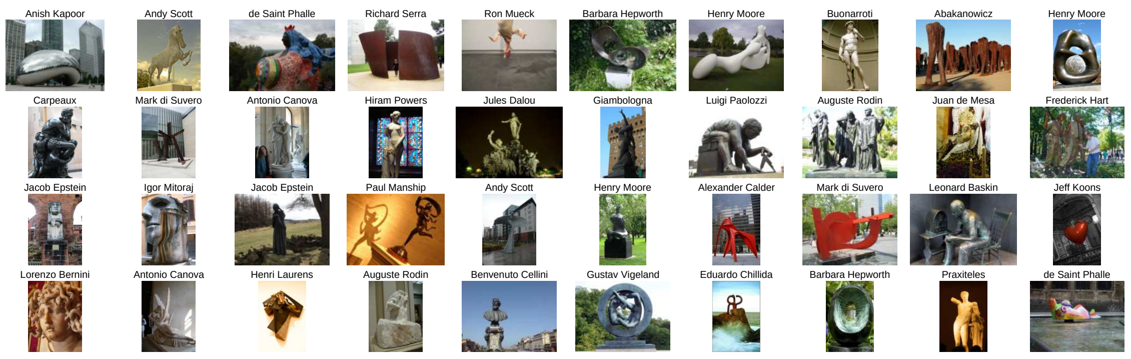


Figure 7.13: Random sample of evaluation query images (40 out of 200) used for evaluation, and sculptor names for each image (first names for some sculptors are cropped for display).

have a low tf-idf score are automatically inserted to form a meaningful sequence of words, etc.

7.5.4 Evaluation and results

The on-line system including all the stages takes only 0.8s from specifying the query image to returning the sculptor and sculpture name on a standard multi-threaded 2.8 GHz workstation. The memory required to store the (non-compressed) files for the retrieval system is 4.3 Gb. Of the run time, on average, 0.56s is used for visual matching, 0.23s for calling the Google API, and 2ms for sculpture and sculptor naming.

To evaluate the identification performance we have randomly selected 200 images of various sculptures from the Sculptures 50k dataset. For each of these we have manually confirmed that at least one more image of the same sculpture exists in the database, as otherwise identification would be impossible. A sample of these is shown in figure 7.13, illustrating the wide variety of sculpture images and sculptors

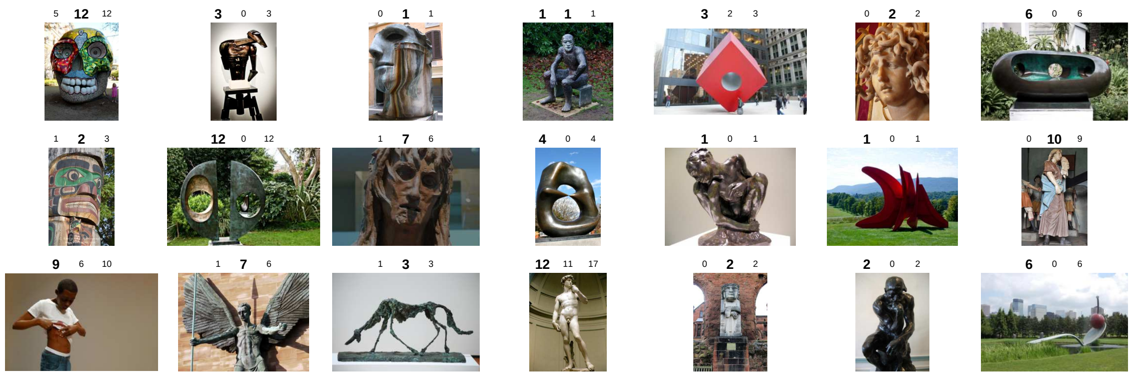


Figure 7.14: **Comparison of BoB and BoW methods for 21 (out of the total of 200) evaluation query images.** The numbers above each query image are the number of positives retrieved before the first negative for each of the methods; from left to right these are BoB, BoW and the combined method. A high number corresponds to better performance and indicates both that the first mistake was low ranked, and also that there are many examples of that sculpture in the *50k* dataset. Numbers shown in bold and larger font point out the better method to be used for the given image (BoB or BoW).

used for evaluation.

The system can be evaluated at three levels: visual matching, sculpt*or* naming and sculpt*ure* naming, and we report on each of these in turn.

7.5.4.1 Visual matching

Visual retrieval failures happen due to well known problems like extreme lighting conditions, inability to handle wiry objects, large segmentation failures (BoB method), interest point detector dropouts (BoW method), descriptor quantization etc. Segmentation can fail when the sculpture is not isolated from the background physically, for example draped with a sheet.

The visual matching is quantitatively evaluated here by reporting the proportion of queries for which the query sculpture appears in the top four retrieved results. Note that we use the same definition of the "same sculpture" relationship as in section 7.3, namely: the two sculptures are the same if they have identical shapes, but they do not need to be the *same instance*, as the same sculpture can be produced multiple times (and can be made in different sizes and from different materials).

The BoB and BoW methods successfully retrieve the correct sculpture within the top four results 60.5% and 63.5% of the time, respectively, while the combined method achieves 83.5%. The large performance boost obtained by combining the two methods demonstrates that they capture complementary information. Figure 7.14 shows query images and performances of BoB, BoW and the combined method.

7.5.4.2 Sculptor identification

The performance measure for sculptor identification (section 7.5.3.1) is the proportion of times the retrieved sculptor matches the sculptor of the query image. Recall that images were downloaded from Flickr by issuing textual queries with sculptor names, so the image-sculptor association is automatically available but potentially noisy (i.e. may not be the true sculptor).

The *Winner-Takes-All (WTA)* scheme correctly identifies the sculptor 154 times, i.e. achieves the score of 0.770, while *Weighted Voting (WV)* achieves 0.785, i.e. WV succeeds 94% of the times that the visual matching is correct. Compared to WTA, WV manages to overcome three retrieval failures and two noisy labellings, while introducing one mistake and changing an accidental positive into a negative.

In the case of WV, the BoB and BoW methods achieve 0.550 and 0.635, respectively, while the combined method achieves 0.785. If we instead imagine that there is an oracle that decides which retrieval method to trust to obtain the matching set for each query image a performance of 0.835 is achievable.

7.5.4.3 Sculpture identification

It is harder to evaluate sculpture identification (section 7.5.3.2) as sculptures often do not have distinctive names (e.g. many Henry Moore's sculptures are known simply as "Reclining Figure"), and query image descriptions are too noisy to be used as ground truth (unlike the case of sculptor naming, there is not a simple image-sculpture association available from the query used to download the data). As a proxy for evaluating the obtained sculpture names we perform a textual Google image search and check if an image of the queried sculpture is present in the top 15 results. We have manually done this evaluation for each of the 200 queries and recorded the proportion of times a hit was obtained.

The Google image search query is formed by concatenating the sculptor name (surrounded by citation marks), followed by the top two keywords obtained in the procedure from section 7.5.3.2, appended by "-flickr" in order to make sure we do not simply retrieve back images from our dataset as the text would certainly match. For the example shown in figure 7.11, the system returns "Giambologna" as the sculptor and the top words are "centaur" and "hercules", then the Google image search query used to evaluate these results is *"Giambologna" centaur hercules -flickr*. Note that the query string is identical to the one used for query expansion. The obtained search results (also shown in figure 7.11) contain many examples of the queried sculpture thus confirming identification success.

The combined method achieves a sculpture identification score of 0.615. This

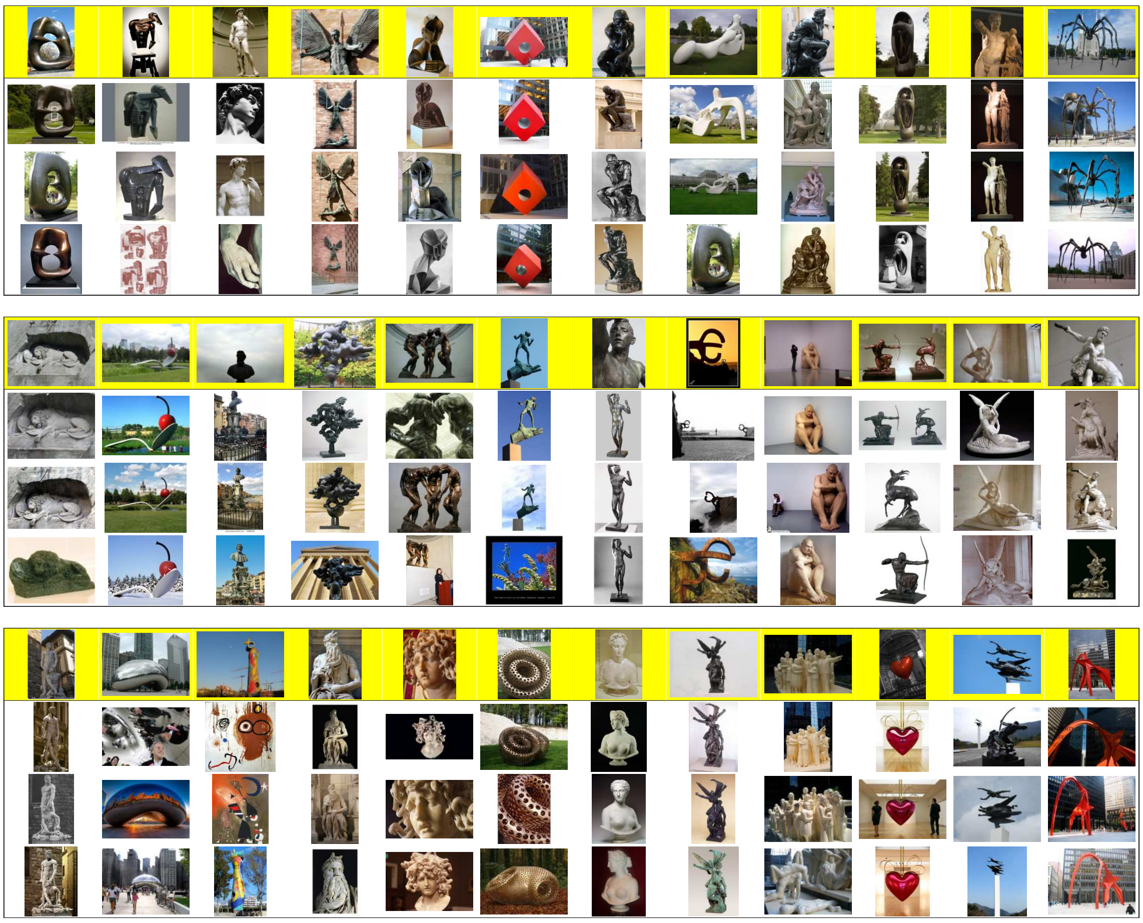


Figure 7.15: **Examples of sculpture naming evaluation.** Illustrations are laid out in three four-row blocks, each column in one block shows one example. For each example the top row shows the query image highlighted in yellow, while the remaining three rows show the top three Google image search results (section 7.5.4.2) using the identified keywords as textual queries (section 7.5.3.2). A wide variety of sculptures are correctly identified.

means that it succeeds 78% of the times that the sculptor identification is correct. Unlike other Flickr annotations we have found the annotations of sculpture images to be fairly reliable. For this reason, it has not been necessary to go to further efforts in overcoming noisy annotations such as (Jin et al., 2005, Wang et al., 2006). Qualitative examples in figure 7.15 demonstrate the effectiveness of our method.

The meta data clean up and normalization step (section 7.5.3.2) is very important since switching off the described preprocessing (while still using automatic translation and removing the sculptor name) causes the performance to drop by 19%, to 0.500. Even when identification still succeeds, the proportion of correct images retrieved in the top 15 Google image search results substantially decreases, the obtained keywords are much noisier and full sculpture names are substantially worse.

Meaningful name extraction. The procedure used to obtain the full sculpture name from identified keywords (section 7.5.3.2) has been found to work very well. The keywords are successfully put in order, for example the top two keywords "thinker the" are converted into "The Thinker", as well as grown into a meaningful sculpture name, for example the top two keywords "sons ugolino", "vulture strangling", "call arms", "lion lucerne' and' "rape sabine" are automatically and correctly converted into "Ugolino and His Sons", "Prometheus Strangling the Vulture II", "The Call to Arms", "The Lion of Lucerne" and "The Rape of the Sabine Women", respectively.

The fact that only the top keyword is used for name growing also means that mistakes from thresholding the keywords can be corrected. For example, the top two keywords for Michelangelo's "David" are "david" and "max", where the latter keyword is meaningless since the sculpture has a one-word title. The name growing procedure starts from "david" and stops at the very beginning correctly yielding "David", as no expansion was found with sufficient support. Finally, it is worth noting that the Google image search using an automatically generated textual query can also flag a failure when the search yields very few or no results.

Actual outputs of the full system on a subset of the evaluation queries are shown

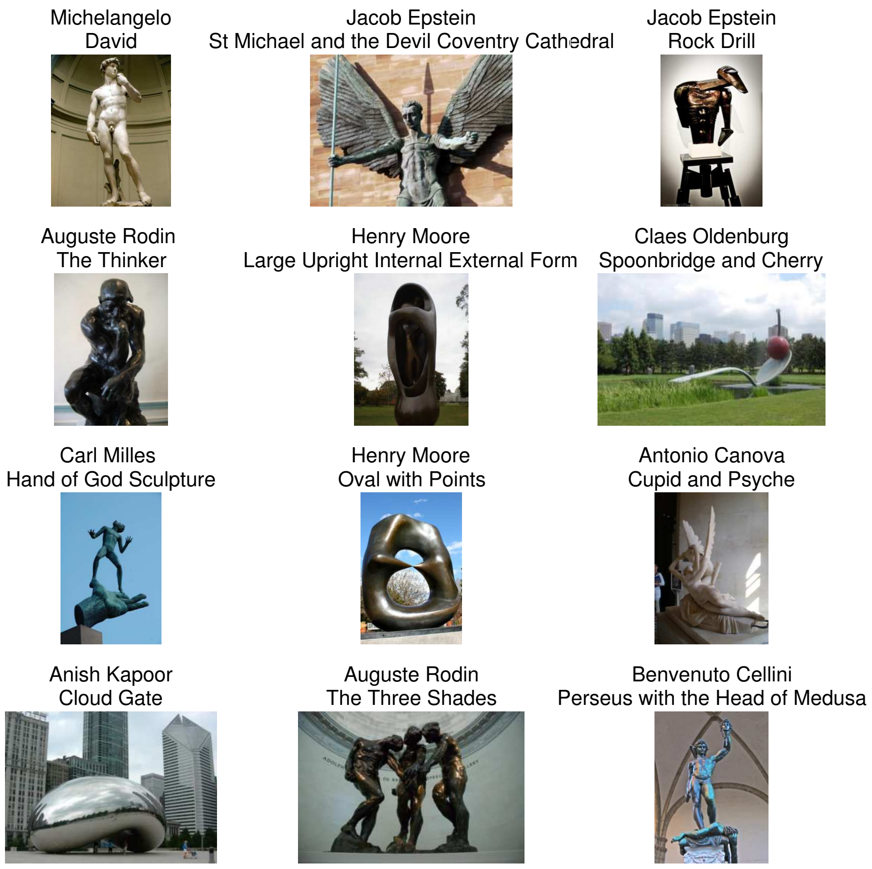


Figure 7.16: **Sculpture naming examples.** Automatically obtained sculptor and sculpture name are shown above each evaluation query image.

in figures 7.11 and 7.16. The complete set of results over all 200 evaluation queries are provided online[4].

Failure analysis. Here we concentrate on problems related to sculpture naming given successful visual retrieval and sculptor naming.

(i) Bad annotation: The top retrieved results contain false or irrelevant annotation, or no annotation at all, rendering identification impossible.

[4] http://www.robots.ox.ac.uk/~vgg/research/sculptures/

(ii) Untitled sculpture: Many sculptures simply do not have names or they are not commonly known. For example, many sculptures made by Henry Moore are simply known as "Reclining figure", so even though the system correctly identifies these terms Google image search results are not guaranteed to show this particular reclining figure.

(iii) Place domination: The textual description is dominated by the sculpture location thus still potentially correctly specifying the sculpture but not providing a useful name for it; examples include names of museums or sculpture parks. This issue does not necessarily cause a failure since the information is often enough to uniquely identify a sculpture, for example: the top two words found by our method for Jacob Epstein's "St Michael and the Devil" in Coventry are "coventry" and "michael", all top 15 results from Google image search show the same sculpture.

(iv) Rare words dominate: Sometimes rare words, such as spelling errors, slang, unusual names etc, can dominate the results as they are deemed to be highly informative. On the other hand, the sculpture "We will" by Richard Hunt fails to be identified as both words are very common.

(v) Name lost in translation: In this case the name of the sculpture is most widely known in its original form, thus performing Google image search for its English translation fails to retrieve relevant results, even though the sculpture identity has been effectively correctly identified. In our 200 evaluation queries we haven't noticed catastrophic failures due to this problem, however it is possible it would occasionally prevent identification. One example in which the interference is significant, but not sufficient for identification to fail, is in the case of Joan Miró's "Woman and Bird" (original Catalan: "Dona i Ocell"); where the top two words are correctly identified as “woman” and “bird” yielding only 5 out of 15 correct top Google image search results, while searching for the original words “dona” and “ocell” gives 14.

(vi) Translation mistakes: Automatic translation fails to detect the correct language and/or to translate the text correctly into English. These are not necessarily catastrophic and in many cases the correct answer was obtained despite these failures (for example Rodin’s “Kiss” is identified as “Kiss” and, in (the original) French, “Baiser”).

(vii) Finally, our choice of evaluation measure can sometimes be a source of false negatives. For example, Gatzon Borglum’s monumental sculpture “Mount Rushmore” is correctly identified by our method, but searching on Google images for *“Gatzon Borglum” mount rushmore* mostly yields images of the sculptor with image descriptions such as “Gatzon Borglum, the sculptor of Mount Rushmore”.

In the light of this failure analysis and the noisiness of Flickr annotations, the achieved sculpture identification score of 0.615 demonstrates that the system really performs quite well.

7.6 Conclusions and further work

We have succeeded in our aim of raising 3D smooth objects to a first class specific object. This required both segmentation (a discriminative representation of the material appearance) and boundary representation. In doing this we have demonstrated that HoG can also be used as a descriptor for specific object retrieval (given suitably cleaned data), rather than solely as a descriptor where learning must be used. We have also established that 3D sculptures (as an example of 3D smooth objects) can be successfully retrieved using only a bag of boundary representation – without requiring any additional spatial information in the first instance.

We expect our framework to generalize to other classes of smooth objects, but new classifiers need to be trained to segment particular classes, e.g. plastic bottles, semi-transparent objects, etc.

Sculpture naming. Using this framework, we have demonstrated that sculptors and, with somewhat less success, sculptures can be named given a query image of a particular sculpture.

The next stage is to scale up the dataset further as having more examples of each sculpture in the corpus will overcome many of the failure cases of the sculpture naming. One avenue we are investigating is adding an authority score depending on the origin of the photo and text, e.g. the meta-data could have more authority if the photo is from Wikipedia rather than Google Image search or Flickr; or more authority if sculptures match when contributed by several different Flickr sources.

7.7 Impact

The work on smooth object retrieval was published in ICCV 2011, while the sculpture naming work was published in ICMR 2012 and was nominated for the best paper prize. Motivated by our approach, Arandjelović (2012)[5] also tackle the problem of recognizing smooth objects. They introduce a descriptor based on the local profile of boundary normals' directions; their descriptor achieves similar recogni-

[5]Note that the author of (Arandjelović, 2012) is not the same as the author of this thesis despite the same last name. The two authors are brothers.

tion performance to our boundary descriptor, while significant improvements are achieved by combining the two.

Google Goggles is not able to recognize smooth objects (Neven, Google, 2011), and thus Google is interested in the work presented in this chapter and have investigated it thoroughly (Neven, Google, 2012). They were able to replicate our results, but are interested in fine-grained recognition of general smooth objects (e.g. recognizing cars) for which they were unable to perform automatic segmentation needed by our approach.

Chapter 8

Extremely Low Bit-rate Nearest Neighbour Search

The goal of this work is a data structure to support approximate nearest neighbour search on very large scale sets of vector descriptors. The criteria we wish to optimize are: (i) that the memory footprint of the representation should be very small (so that it fits into main memory); and (ii) that the approximation of the original vectors should be accurate.

We introduce a novel encoding method, named a Set Compression Tree (SCT), that satisfies these criteria. It is able to accurately compress 1 million descriptors using only a few bits per descriptor. The large compression rate is achieved by not compressing on a per-descriptor basis, but instead by compressing the set of descriptors jointly. We describe the encoding, decoding and use for nearest neighbour search, all of which are quite straightforward to implement.

The method, tested on standard benchmarks (SIFT1M and 80 Million Tiny Images), achieves superior performance to a number of state-of-the-art approaches, including Product Quantization, Locality Sensitive Hashing, Spectral Hashing, and Iterative Quantization. For example, SCT has a lower error using 5 bits than any of the other approaches, even when they use 16 or more bits per descriptor. We also include a comparison of all the above methods on the standard benchmarks.

8.1 Introduction

Nearest neighbour search is ubiquitous in computer vision with numerous applications across the field. With ever larger data sets generating millions or billions of vector descriptors, two particular problems have become critical: (i) how to keep all the original vectors in main memory, and (ii) how to obtain the nearest neighbours of a query vector as fast as possible. Recently there have been two quite distinct threads of work aimed at addressing problem (i), which both proceed by obtaining a low dimensional representation of the original vectors, such that the distance between two low dimensional representations is a good approximation of the distance between the original vectors. The consequence is that the low dimensional vectors for the entire database then fit in main memory which in turn alleviates problem (ii) as no (expensive) hard disk access is required.

One thread is the product quantization of Jégou et al. (2011a). This follows on from earlier work on Hamming embedding (Jégou et al., 2008). Both were aimed at obtaining a more accurate distance between SIFT (Lowe, 2004) vectors than that obtained by straight forward k-means vector quantization of the entire vector and representation as visual words (Sivic and Zisserman, 2003). Product quantization divides the vector into sub-vectors, and then vector quantizes each sub-vector (if there are m sub-vectors each quantized independently into k cluster centres, then there are k^m possible code words with $m \cdot log_2(k)$ bits required to represent each vector). It was originally applied to SIFT vector matching, but has since been employed in large scale category retrieval (Sánchez and Perronnin, 2011) and used with inverted indexes for immediate retrieval (Babenko and Lempitsky, 2012, Rastegari et al., 2011) (again addressing problem (ii)).

The second thread is the binary string representation of vectors used for retrieval in the 80 Million tiny images series of papers (Gong and Lazebnik, 2011, Raginsky and Lazebnik, 2009, Torralba et al., 2008, Weiss et al., 2008, 2012). Here the goal is to represent the original vectors by short binary strings, such that the Hamming distance between the binary representations approximates the distance between the original vectors (or more accurately, that the ranking of Hamming distances equals the ranking of distances between the original vectors).

Current methods achieve reasonable performance only when 16 or 32 or more bits per descriptor are used, but none is capable of functioning at the extremely low bit-rate regime which is certainly needed for huge scale datasets. In this chapter we propose a *Set Compression Tree* (SCT) approach to lossy descriptor compression that is capable of accurately representing vectors using only 4 to 7 bits per vector. The key idea in SCT coding is not to store information directly on a per-descriptor basis, but instead to store information jointly across sets of the descriptors in the database. i.e. if the set of descriptors is $\{x_1, x_2, \ldots, x_n\}$ then the compressed set is $\text{com}\{x_1, x_2, \ldots, x_n\}$ rather than $\{\text{com}(x_1), \text{com}(x_2), \ldots, \text{com}(x_n)\}$. Because the code applies to a *set* of vectors, the number of bits required for *each* vector is far less. The coding is achieved through a simple and very fast scheme where the feature space is partitioned into disjoint cells in a k-d tree like manner using binary splits, such that each descriptor is uniquely associated with one cell and approximated by the centroid of that cell. The SCT coding is explained in section 8.2, and compared in section 8.3 to previous compression methods including: Product Quantization of Jégou et al. (2011a), Locality Sensitive Hashing (Andoni and Indyk, 2008) (LSH), Spectral Hashing of Weiss et al. (2008), SIK of Raginsky and Lazebnik (2009), and Iterative Quantization (ITQ) of Gong and Lazebnik (2011). As will be seen, in all cases SCT has superior performance at the very low bit end of the compression scale.

Such a low bit compression can find immediate application in multiple places, and we mention two use cases here: the first is direct representation of descriptors. These may be descriptors of local image regions (such as SIFT) or descriptors of the entire image (such as GIST (Oliva and Torralba, 2001), PHOW (Bosch et al., 2007), VLAD (Jégou et al., 2010b) etc). The second is in large scale object retrieval systems, where descriptors are first quantized and an inverted index then used for fast access, and the *residual* (the difference between the cluster centre and original descriptor) is compressed for finer descriptor matching (Jégou et al., 2008, 2011a).

As far as we know there has been no scheme similar to SCT proposed before. The closest work is Chandrasekhar et al. (2011), which, like SCT, exploits the fact that descriptor ordering is not important. However, their method is aimed at lossless compression for sets of binary vectors rather than the approximation for real vectors targeted here. Note, k-d trees have long been employed for efficient and approximate nearest neighbour algorithms (Amit and Geman, 1997, Muja and Lowe, 2009, Silpa-Anan and Hartley, 2004). Our objective is very different though – we aim for compression, whereas previous uses have stored all the original vectors (leading to an *increase* in storage requirements as the parameters of the tree must also be stored).

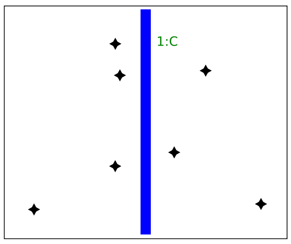
8.2 Set Compression Tree (SCT) encoding

The objective of the Set Compression Tree (SCT) is to provide an extremely low bit-rate lossy compression of a set of descriptors. It can be used for brute-force nearest neighbour search by decompressing the descriptors from the SCT encoded set one by one and comparing them to the query vector. Section 8.4 explains how SCTs can be used for approximate nearest neighbour search.

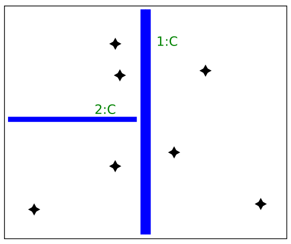
Current methods for obtaining a compressed representation of descriptors all focus on compressing each descriptor *individually*. For very low bit-rate scenarios this approach is infeasible as many descriptors get assigned to the same value thus making it impossible to discriminate between them. Nearest neighbour search in such a system would have extremely low precision.

We instead focus on compressing all descriptors *jointly* such that the amortized cost of storing a descriptor is extremely low. This is achieved by not storing any information on a per-descriptor basis, but representing all descriptors together via a single binary tree. All of the storage requirements for descriptor representation are contained by the encoding of this binary tree. The compression method, i.e. tree building and encoding is described here, followed by implementation details in section 8.2.1.

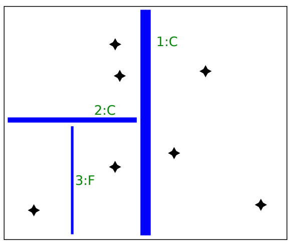
The method proceeds by dividing the descriptor space by a sequence of binary



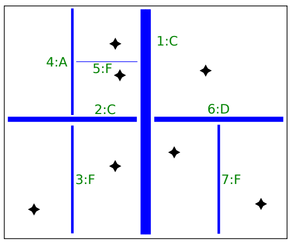
(a) 1st step; code = C



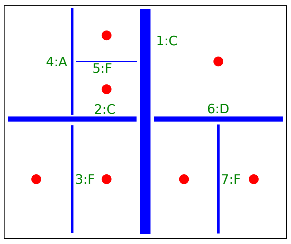
(b) 2nd step; code = C



(c) 3rd step; code = F



(d) Final tree



(e) Reconstruction

Symbol	Code	Number in child 1	Number in child 2
A	0000	$= 0$	> 1
B	0001	> 1	$= 0$
C	01	> 1	> 1
D	0010	> 1	$= 1$
E	0011	$= 1$	> 1
F	1	$= 1$	$= 1$

(f) Split outcomes and codes

Final tree encoding: CCFAFDF
Bitrate: $15/7 = 2.14$ bits per vector

Set tree encoding: 01 01 1 0000 1 0010 1

Figure 8.1: **SCT encoding.** The aim is to encode the seven 2-D vectors (*black stars*) by a sequence of binary partitions of the space (delimited by the outside rectangle). In *(a)-(e)* the space splits are shown in blue, thicker lines correspond to splits at levels closer to the root of the tree. *(a)-(c)* The first three steps in the tree construction. The construction (i.e. splitting bounding spaces) stops once all regions contain at most one descriptor; the final tree separating all seven vectors is shown in *(d)*. Next to each split its ordinal number in the sequence of splits is shown together with the code recording the outcome of the split. *(f)* The list of possible outcomes and their corresponding codes. For example, the second split *(b)* is shown as *2:C*, the table states that *C* means there is more than one vector in each of the newly created cells. *(e)* the vectors are represented by the centroid (*red circles*) of the final cells. Vector decompression is based solely on the tree, which is encoded by the sequence of split outcomes (15 bits, i.e. 2.14 bits per vector), and the centroids generate the reconstructed vectors.

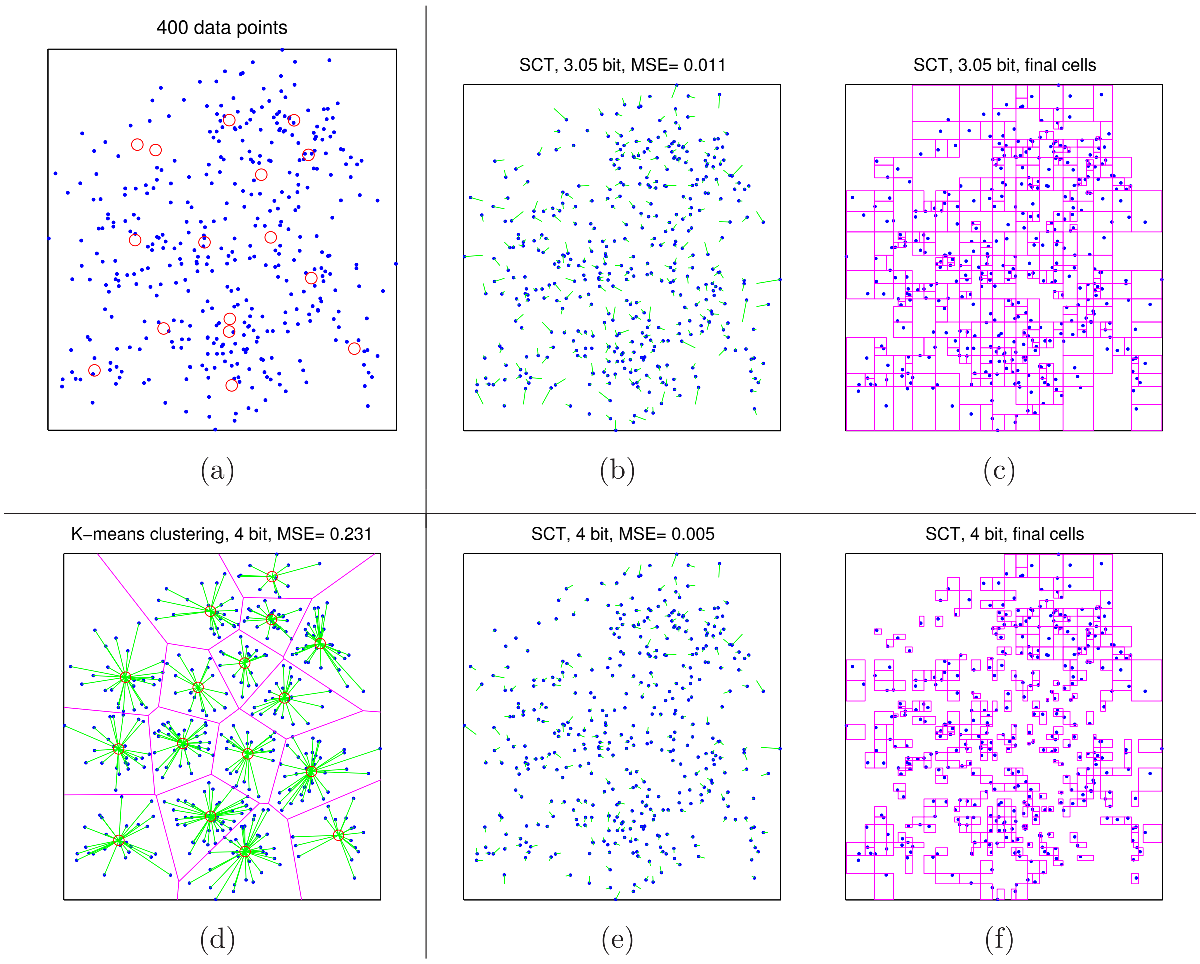


Figure 8.2: **Compression example.** All plots show a 2-D feature space, 400 synthetically generated 2-D points are shown in *blue*, *green* lines show the displacement between the original point and the approximation, *magenta* lines depict cells (i.e. uncertainties) associated with quantized descriptors. *(a)* The 400 points and the 16 GMM means (*red circles*) used to generate them. *(d)* Representing the data with 4 bits per point by using the closest cluster centre (16 clusters, *red circles*, are obtained through k-means); the errors (depicted by long *green* lines) are large. *(b)* SCT encoding; with as little as 3.05 bits per point the errors are very small, with MSE (mean squared error between the original and reconstructed points) being 21 times smaller than for k-means; *(c)* shows the cells associated with each point; the point is finally represented by the centroid of the cell. *(e)* Tree encoding; at the same bit-rate (4 bits per point) as the k-means method; MSE is 46 times smaller. *(f)* shows the cells associated with each point.

splits that are predetermined and independent of the data to be compressed. After a number of divisions, a cell will only contain a single vector, and that vector is represented by the centroid of the cell. The method is best explained through an example, and Figure 8.1 shows the compression and encoding steps for the case of seven descriptors in a 2D space. The resulting representation uses only 2.14 bits per descriptor.

The key idea underpinning SCT encoding is that a single split contributes information to many descriptors; for the example in figure 8.1(a) the first split, encoded with two bits, halves the space for all seven descriptors, so for 0.29 bits per descriptor their positional uncertainty is halved. If, instead, every descriptor is encoded individually halving the feature space would cost 1 bit per descriptor.

To summarize the method, the encoding algorithm starts from a root bounding space which contains all descriptors and proceeds by a sequence of binary space partitions. A bounding space S is divided in a predefined manner independent of the descriptors it contains, for example, an axis aligned split is chosen such that the longest edge of the space is divided in half (the split sequence is explained in full in section 8.2.1). The split of the space S partitions it into two spaces A and B (i.e. such that $A \cup B = S$ and $A \cap B = \varnothing$), then *all* that is recorded at each split is the "outcome" of that split, i.e. information on the number of descriptors in A and B, denoted as $|A|$ and $|B|$ respectively. All outcomes fall into six categories that are recorded (figure 8.1(f)): (i) $|A| = 0$, $|B| > 1$, (ii) $|A| > 1$, $|B| = 0$, (iii) $|A| > 1$, $|B| > 1$, (iv) $|A| > 1$, $|B| = 1$, (v) $|A| = 1$, $|B| > 1$, (vi) $|A| = 1$, $|B| = 1$. Note, a space is split only if the number of elements $|S|$ is larger than one. Consequently, options $\{|A| = 0, |B| = 1\}$ and $\{|A| = 1, |B| = 0\}$ are not possible since, $|S| > 1$

and $A \cup B = S$ thus $|A| + |B| > 1$.

After the encoding, the entire data set is simply represented by the sequence of splits outcomes. This sequence is all that is required to reconstruct the space partitions (and thereby the centroid of the cell which represents the original vector). Decoding is analogous to the encoding procedure: the process starts from the root bounding space retracing the same sequence of binary splits as the encoding. A bounding space S is divided in the same predefined manner as used in the encoding (e.g. by halving the longest edge) and the split outcome is obtained from the SCT. If any of the children cells contains only a single descriptor, it is reconstructed using the cell centroid. The splitting of remaining cells which are known to contain more than one descriptor is continued in the same fashion, until none remains and the whole set of descriptors will then have been decompressed.

Figure 8.2 illustrates the benefit of using the SCT representation. 400 2-D data points are generated by sampling from a Gaussian mixture model with 16 Gaussian components. Compressing this data with 4 bits per descriptor by approximating a point with its closest cluster centre, out of $2^4 = 16$ clusters, yields very large quantization errors. The centres in this example are obtained using k-means (note that at such a low bit-rate product quantization (Jégou et al., 2011a) degenerates to simple k-means), but any division of the space into 16 regions is bound to do similarly poorly. This is because on average $400/16 = 25$ points are assigned the same 4-bit code (corresponding to one region) and are thus completely indistinguishable between each other. However, quantizing the points jointly by sharing information between them performs very well, achieving a 21 times smaller mean squared error than k-means with only 3.05 bits per descriptor.

8.2.1 Implementation details

In this section technical details are discussed; they are sufficient to fully reproduce the experiments. The summary of all the steps of the SCT algorithm is given in figure 8.3.

In the case of nearest neighbour search there are three datasets involved: a training set, the database to be encoded, and the query set. The query set contains all the query vectors, and the database set contains the vectors on which nearest neighbour search will be performed, i.e. it contains the vectors that should be compressed. The training set is used to learn all the required parameters in an unsupervised way (for example PCA) and, in general, should be distinct from the database to be encoded in order to avoid implicitly storing database information in the learnt parameters. For an extreme example, one could "compress" a database of 1 million vectors by "learning" a 1 million dictionary identical to the database, and then representing the database using 20-bits per vector word ID thus achieving perfect representation at quite a low bitrate.

Requirements. An important requirement for SCT is that each component of the vector has a lower and an upper bound. The requirement is not very restrictive as this is the case with all commonly used descriptors, e.g. SIFT is bounded by 0 and 255 in each dimension.

Split choice. For a given cell, splits are determined in the following manner: The dimension to be split is chosen to be the one with the largest difference between the upper and lower bounds. It is straightforward to see that choosing to split the cell across its largest extent minimizes the expected approximation error for

(a) Training

1. Compute PCA, keep D principal components for the training set V
2. Use a random rotation matrix R $(D \times D)$ to rotate the training set to get $\tilde{V}$
3. For each dimension i in $[1, D]$:

    Store the distribution of values $\tilde{V}$ at dimension i, and lower (L_i) and upper (U_i) bounds

(b) Encoding

1. Rotate the D principal components of database descriptors by R
2. Create the root bounding space, bounded by $[L_i, U_i]$ for dimension i, and associate all database descriptors with it
3. Find a bounding space S (depth-first) containing more than one descriptor; if none found go to step (7)
4. Set $d = \arg\max_i (u_i^S - l_i^S)$ and $s = median(\{\tilde{V}_{i,d} | i : \tilde{V}_{i,d} \in [l_i^S, u_i^S]\})$
5. Create cells A^S and B^S by splitting cell S at dimension d and position s, move each descriptor from S into A^S or B^S depending on which space they are bounded by
6. Encode the split outcome, see figure 8.1(f), and go to step (3)
7. Compute optimal Huffman coding for the recorded frequency of split outcomes, store it using 18 bits and re-encode the tree by replacing old codes with new, optimal ones
8. Refine final non-empty bounding spaces by splitting them additionally, using one bit per split

(c) Decoding

1. Create the current cell C bounded by $[l_i^C, u_i^C] = [L_i, U_i]$ for dimension i. Set $|C|$ to > 1
2. If $|C| = 1$, refine C by reading codes created in (b.8) and output its centroid rotated by R^{-1}
3. If $|C| \leq 1$, set C to be equal to its parent, go to step (2)
4. If this is the first time C is visited:

    Create cells A^C and B^C by splitting dimension d at s, obtained in the same way as in (a.4)

    Decode the split outcome and assign $|A^C|$ and $|B^C|$ to values 0, 1 or > 1 accordingly
5. If A^C was not visited, set $u_d^C = s$ (i.e. $C \leftarrow A^C$) and go to step (2)
6. If B^C was not visited, set $l_d^C = s$ (i.e. $C \leftarrow B^C$) and go to step (2)
7. If parent exists, set C to be equal to it and go to step (2), otherwise exit as all nodes have been visited

(d) Nearest neighbour search

1. Project the query descriptor to the D principal components to obtain q
2. Use (c) to decode the tree, at step (c.2) compare q with the outputted value

Figure 8.3: **SCT algorithm summary.** Lower and upper bounds for cell S in dimension d are denoted as l_d^S and d_d^S, respectively. Training data is referred to as V, the value of vector i at dimension d is $V_{i,d}$. Note that for nearest neighbour search one would actually also rotate q in step (d.1) by R and modify step (c.2) not to rotate the outputted centroid by R^{-1}, thus improving the speed.

a descriptor (for example, imagine a degenerate 2-D cell where the extent in the x direction is long while the extent in the y direction is infinitesimal – splitting y would not be beneficial). Experiments confirm that pseudo-randomly choosing splitting dimensions (and recording the random seed in order to be able to reconstruct the tree) yields an inferior MSE to our scheme. The order of the splits is determined by always choosing the left child region first.

For a given cell and splitting dimension, the place of the split is determined with the aim of balancing the tree; a balanced tree is likely to produce similar sized cells for all descriptors thus resulting in similar magnitudes of reconstruction errors. The split is chosen according to the median value of the training data in the splitting dimension (clipped by the cell). Alternatively one can simply split the space in half, however this choice could lead to an unbalanced tree depending on the data distribution. A histogram for each component is retained to efficiently compute the median.

In summary, only the following information is stored from the training data in order to generate the split sequence: (i) upper and lower bounds for each dimension (in order to create the first, root cell); (ii) a histogram of training data values for each dimension (e.g. for SIFT, 128 histograms are stored).

Note, it is essential to have a cell splitting strategy that does not depend on the data inside the cell (as above). Otherwise the sequence of splits needs also to be stored, as one would do with a k-d tree, and this will incur a considerable storage cost.

Optimal binary encoding. The six outcomes of splits are encoded using optimal variable length binary codes. To achieve this tree encoding is performed in two stages. In the first stage splits are simply encoded using predefined suboptimal binary codes. During this stage, occurrence counts for each of the six outcomes are obtained. In the second stage, the tree is re-encoded by replacing the initial suboptimal codes with optimal ones. Huffman coding (Huffman, 1952) is used to obtain optimal variable length binary codes by utilizing the recorded frequency of the six split outcomes. Storing the Huffman tree requires 18 bits in total and is certainly worth the reduced storage requirement for the tree representation.

Finer representation. It is simple to obtain a finer representation of a descriptor by increasing the bit-rate: the cell associated with it can be additionally split with a rate of 1 bit per split, encoding on which side of the split the descriptor is. Based on the desired bit-rate, the additional available bits can be equally distributed to refine each of the final bounding spaces, however, it is better to bias the refinement towards large bounding spaces (i.e. descriptors which have been represented poorly).

Dimensionality reduction. SCT encoding is not appropriate to use when the dimensionality of the descriptor vector is large compared to $log_2 N$, where N is the number of descriptors to be compressed. For example, compressing 1 million descriptors is expected to yield a tree of depth 20 ($log_2(1M)$), as it is approximately balanced, meaning that at most 20 dimensions will be split in order to obtain a final bounding space for each descriptor. Trying to compress 128-D SIFT descriptors with 20 splits would result in large approximation errors as at least 108 dimensions would remain unchanged and thus not convey any information about the value of the descriptor in any of the 108 dimensions. For this reason it is recommended to first perform dimensionality reduction on the data. We chose to zero-centre the data followed by principal component analysis (PCA), keeping only D dominant directions. Since the variance of different components is not balanced we subject the D-dimensional vectors to a random rotation (Gong and Lazebnik, 2011, Jégou et al., 2010b), otherwise one would need to bias the choice of splitting directions towards the components which carry more information.

Obtaining the ID corresponding to a vector. As decompressing the SCT permutes input vectors according to the depth-first traversal of the tree, one might wonder how can one preserve the identity of the decompressed vectors without storing additional (costly) information. For example, in an image retrieval system, a retrieved feature vector (e.g. GIST (Oliva and Torralba, 2001), VLAD (Jégou et al., 2010b)) must be associated with its original image.

Here we distinguish two cases based on the nearest neighbour search strategy: (a) linear traversal, and (b) large scale retrieval using indexing.

In case (a), existing compression methods produce compressed vectors in the same sequence as input vectors, by for example product quantizing each vector in turn, and thus do not need to store any additional meta information to preserve the compressed vector identity (the original order may simply be alphabetical from file names). For example, returning the third compressed image descriptor means the system should return the *third image* in the image database. Thus the correspondence between compressed vector and original image is stored implicitly.

For SCT, the order of the reconstructed vectors depends on the order of the tree traversal and not the original order, thus SCT seemingly requires extra information to store the permutation of the vectors. However, this is ***not*** the case, for example the original images can simply be permuted so that the ‘canonical ordering’ of the images is the same as that of the decompression order.

More practically, the correspondence between the reconstructed vector order number can be stored in a look up table (LUT) that maps from order number to identity (e.g. its URL). *All* existing methods employ a LUT or some predefined ordering of the data. For example, an image retrieval system which represents each image using a single descriptor needs to be able to identify an image in some way from the retrieved descriptor. For images of Flickr or the web, a LUT is used to map between feature descriptor and URL.

In case (b), large scale retrieval, the situation is different, as the data is typically not traversed linearly. Instead, the dataset feature vectors may be grouped in some fashion, for example by vector quantizing, or accessed via an inverted index (Sivic and Zisserman, 2003) (an example is discussed in more detail in section 8.4). In such cases an explicit LUT is required (e.g. between the entries of a posting list and the URLs of the images). Again, this requirement applies to any compression method.

To summarise, in spite of permuting the order of the input vectors, the SCT is capable of preserving vector identities without any additional storage requirements over those of other compression methods.

8.3 Evaluation and results

In this section we compare the compression, accuracy and retrieval speed of SCT to a number of other standard methods.

8.3.1 Datasets and evaluation procedure

We evaluate the performance of the SCT on two standard datasets: (i) the SIFT1M dataset of Jégou et al. (2011a), and (ii) the 580k GIST descriptors used by Gong and Lazebnik (2011), which is a subset of the 80M Tiny Images dataset (Torralba et al., 2008). For both we follow the standard evaluation procedure of the authors, as summarized next.

1M SIFT descriptors (SIFT1M) (Jégou et al., 2011a). This dataset is commonly used for evaluating approximate nearest neighbour methods, with an emphasis on low bit-rate descriptor representations for image search, as it consists of SIFT (Lowe, 2004) descriptors. It contains 10k query descriptors, 100k descriptors used for training, and 1 million database descriptors. The descriptors are from the Holidays dataset (section 3.1.3) and Flickr images. Search quality is evaluated as the average recall of the first nearest neighbour at R retrievals (usually set to 100) for each query, i.e. the proportion of query vectors for which the Euclidean nearest neighbour using SIFT is ranked within the first R retrievals using the approximating method.

580k Tiny Images (Tiny580k) (Gong and Lazebnik, 2011). This dataset contains 580k images and is a subset of the Tiny Images dataset (Torralba et al., 2008); the images are represented by 384-D GIST descriptors (Oliva and Torralba, 2001). It is randomly split into 1k query descriptors and 579k database descriptors which are also used for training; performance is measured across five different random splits of the data.

For this dataset the search performance is evaluated in three different ways, we will refer to them as *mAP-50NN*, *AP-thres* and *mAP-thres*. The first method (*mAP-50NN*), is based on the precision-recall curves proposed by the creators of the Tiny Images dataset (Torralba et al., 2008). The 50 true nearest neighbours for each query are labelled as positives, and the performance is evaluated as mean average precision (mAP) across the queries.

The latter two criteria (*AP-thres* and *mAP-thres*) concentrate on measuring distance preservation from the original descriptor space (in this case GIST) to the new space, thus measuring the effectiveness of hashing schemes to preserve neighbourhood relations. For these cases, a set of "good neighbours" is determined by choosing a global distance threshold T (obtained by taking the mean distance to the 50th nearest neighbour), and labelling a descriptor as a positive for a particular query if its Euclidean distance from the query is lower than the threshold T. This method for ground truth generation was proposed by Weiss et al. (2008) and adopted by (Gong and Lazebnik, 2011, Raginsky and Lazebnik, 2009). However, there is a problem with the associated performance evaluation method, which we refer to as *AP-thres*. In this evaluation method there are Q queries, and an algorithm is evaluated over all pairs of query-database points (i.e. there are $Q \times N$ ground truth pairs, where N is the database size). All pairs are sorted based on their distance in the compressed space, and the AP computed from the precision-recall curve.

The problem is that the test data is extremely unbalanced: 50% of the queries have less than 22 positives, while 7% have more than 30k positives. This means that as long as a system performs well on the 7% of the queries (i.e. these are retrieved correctly and first) then it will reach a "good" AP value, regardless of its poor performance on 93% of the queries. As will be seen in section 8.3.3 a compression algorithm that simply clusters the database descriptors into 16 clusters using k-means (and so reduces the data to four bits) performs better under *AP-thres* than all the other methods with 16 bits. As a consequence of this insensitivity, the evaluation using *AP-thres* is quite misleading.

To account for this imbalance we propose to also measure performance in terms of mean average precision (mAP), i.e. the mean of the average precision for each query, and refer to this as *mAP-thres*.

In summary, the *mAP-50NN* criterion is the most useful one of the three when evaluating image retrieval as it directly measures the ability of a system to return relevant results for any query image.

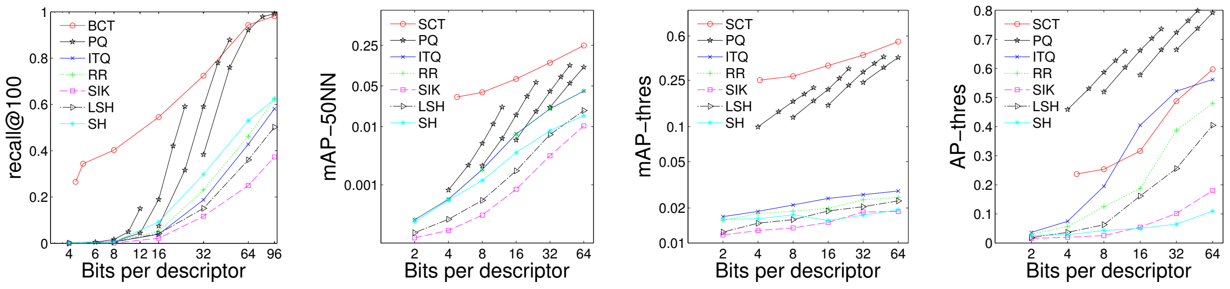
8.3.2 Baselines

SCT is compared to several state-of-the-art methods at various bit-rates, we briefly summarize them here.

(i) Product Quantization (PQ) (Jégou et al., 2011a): Each descriptor is split into m parts and each part is vector quantized independently using k cluster centres, thus having a $m \cdot log_2(k)$ bit code per descriptor.

(ii) Locality Sensitive Hashing (LSH) (Andoni and Indyk, 2008): The code is computed by taking the signs of descriptor projections onto random hyperplanes (normals are sampled from an isotropic Gaussian) with random offsets, the Hamming distance between descriptors encoded in such a way is closely related to the cosine similarity between the original vectors.

(iii) Shift Invariant Kernels (SIK) of Raginsky and Lazebnik (2009): The code is



(a) SIFT 1M (b) Tiny 580k (c) Tiny 580k (d) Tiny 580k

Figure 8.4: **Comparison of SCT with state-of-the-art.** Best viewed in colour; note the log scale for mAPs in *(b)* and *(c)*. The different curves associated with PQ correspond to different numbers of sub-quantizers (from left to right: 1, 2, 4, 8), the combinations of settings reproduce the ones used in the original paper (Jégou et al., 2011a). SCT outperforms all methods at low bit-rates for both datasets and all evaluation measures apart from *AP-thres*, where the test data is extremely unbalanced and the measure inappropriate (see the discussion in section 8.3.1). SCT continues to dominate all competing methods at higher bit-rates as well.

computed by taking the signs of random Fourier features with random offsets.

(iv) PCA with random rotation (RR) (Gong and Lazebnik, 2011, Jégou et al., 2010b): Data is zero-centred and the most significant D principal components are kept and randomly rotated. The D dimensional code is obtained by taking signs of each dimension.

(v) Iterative quantization (ITQ) (Gong and Lazebnik, 2011): Gong and Lazebnik use the RR method and then proceed to iteratively find the rotation which minimizes quantization errors.

(vi) Spectral Hashing (SH) (Weiss et al., 2008): The coding scheme is obtained deterministically by trying to ensure that similar training descriptors get hashed to similar binary codes. This is a NP-hard problem so their approach is to solve a relaxed optimization problem.

Methods (ii)-(vi) rank descriptors based on their Hamming distance from the query which is binarized in the same fashion. For product quantization (i) the asymmetric distance computation method (Jégou et al., 2011a) is used since Jégou et al. report it to be superior; the distance between the query and a database vector is approximated by the distance between the raw (non-quantized) query and the quantized representation of the database vector.

We use publicly available code for all of the baseline methods and replicate the results in (Gong and Lazebnik, 2011, Jégou et al., 2011a), these two papers together cover all the baseline methods.

8.3.3 Results and discussion

Figure 8.4 shows the comparison of SCT with the baseline methods. SCT clearly outperforms all competing methods at very low bit-rates on both datasets and using all evaluation metrics. For example, on the SIFT 1M dataset SCT achieves a recall@100 of 0.344 at 4.97 bits per descriptor, while the next best method, product quantization, achieves 0.005 at 6 bits, i.e. 69 times worse. Even at 16 bits, i.e. 3.2 times larger, PQ only reaches 55% of SCT performance at 4.97 bits per descriptor. The next best method after SCT is PQ, and PQ is then superior to all other methods.

Despite the very low bit-rate, SCT by construction represents *all* database descriptors distinctly, thus making it possible to distinguish between them and rank them appropriately. In contrast, all other methods, by quantizing each descriptor individually into a small number of bits, effectively identically represent a (possibly very large) set of database descriptors, as previously illustrated in Figure 8.2.

This means that, for example, using the Tiny 580k dataset and four bits per descriptor, on average a descriptor has the same representation as another 36k ones; it is impossible to rank the first 36k results for a given query, as any of the 36k factorial permutations is equally likely under these representations. This argument also shows that performance evaluation based on *AP-thres* is very misleading. For example, PQ under *AP-thres* achieves an AP of 0.459 with 4 bits per descriptor. The reason for such a "good" performance is the fact that the test data associated with *AP-thres* is extremely unbalanced, as explained in section 8.3.1.

Figure 8.5 shows qualitative retrieval results on the Tiny 580k dataset. As expected from the quantitative evaluation shown in figure 8.4, SCT performs much better than the state-of-the-art methods while representing descriptors using three times fewer number of bits.

Dimensionality reduction. Figure 8.6 shows the performance of SCT as a function of the number of principal components (PC) used and bit-rate. It can be seen that, as expected, at extremely low bit-rates performance drops with increasing number of PCs due to increasing approximation errors. However, increasing the bit-rate by further splitting final bounding spaces makes it again appropriate to use SCT with this larger number of PCs, which in turn improves the performance as the underlying data is represented more accurately using more PCs.

For a given number of PCs, it is also important to note that the SCT performance reaches the upper bound (i.e. the performance that is obtained by using raw descriptors with the same number of PCs) at quite low bit-rates. For example this point is reached at 32 bits per descriptor for 16 PCs, so only two bits per dimension are sufficient to encode a value which would commonly be represented using 32 or

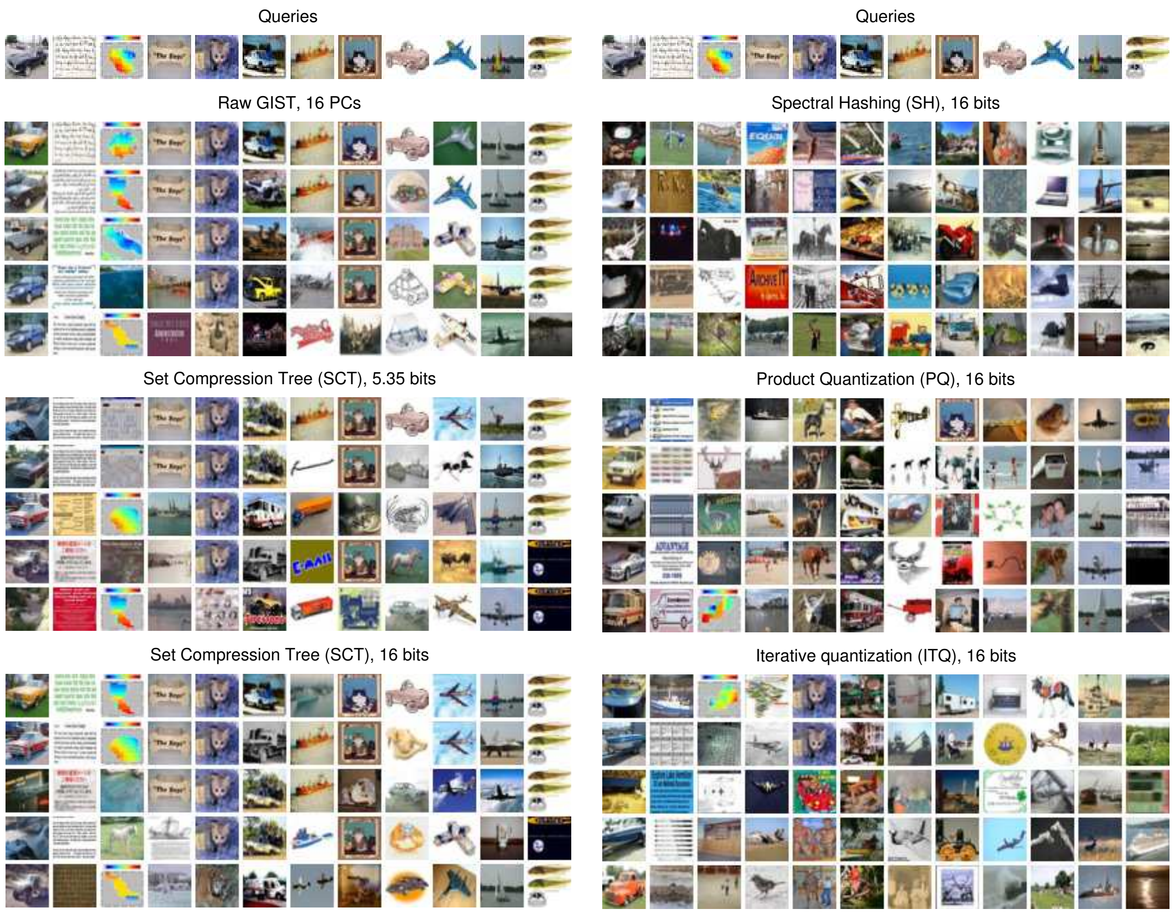

Queries
Raw GIST, 16 PCs
Set Compression Tree (SCT), 5.35 bits
Set Compression Tree (SCT), 16 bits
Queries
Spectral Hashing (SH), 16 bits
Product Quantization (PQ), 16 bits
Iterative quantization (ITQ), 16 bits

Figure 8.5: **Retrieval quality on Tiny 580k dataset.** Best viewed in colour. *First row* shows twelve example queries, each of the remaining *six blocks* show the top five retrieved images for each of the queries. Set Compression Tree (SCT) at only 5.35 bits per descriptor outperforms state-of-the-art methods, namely Spectral Hashing, Product Quantization and Iterative Quantization, while using three times less storage for descriptor representation. Using SCT with 16 bits per descriptor further improves retrieval quality. Note that other methods are not capable of even finding near-duplicate images at such low bit-rates.

64 bits (single or double floating point number).

Fast and low memory footprint of coding and decoding. All timings are measured on a laptop using a single 2.66 GHz processor. The un-optimized program compresses 1 million SIFT descriptors using 32 principal components in 14 seconds, while decompression and nearest neighbour search take 0.5 seconds. The search time scales linearly with the number of database descriptors, searching 580k GIST descriptors (again using 32 principal components) takes 0.26 seconds. In the decoding stage not all of the cells need to be stored in memory at once – the tree is traversed depth-first, so only a single cell representing the "current" node is kept at any one time. Once a leaf node or a node with already visited children is encountered, the current bounding space is reverted to its parent's bounding space. For this to be possible only a small amount of additional information needs to be maintained for each previous split in the current path from the root node, which is quite short as for 1 million descriptors the depth of a balanced tree is equal to 20.

8.4 Discussion and recommendations

In this section we discuss the properties of SCT, its advantages and disadvantages compared to other encoding methods, and three use cases for it.

Unique description. Every descriptor is assigned a unique bounding space, and all bounding spaces are disjoint. This means that even in areas of high descriptor density it is still possible to discriminate between descriptors, a characteristic which does not exist in any of the competing methods. This could potentially be used to disambiguate between descriptors using a second nearest neighbour test (Lowe, 2004).

Asymmetric in nature. As noted by Jégou et al. (2010b), it is beneficial not to have to quantize query vectors when performing nearest neighbour search, as this obviously discards relevant information. SCT is asymmetric at its core as query

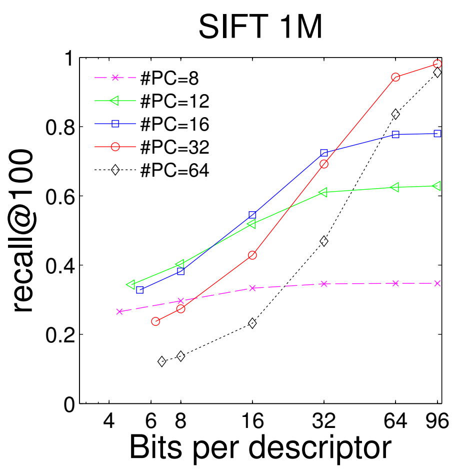


Figure 8.6: **Influence of the number of principal components (PC) on SCT performance with varying bit-rates.** At extremely low bit-rates performance drops with increasing number of PCs due to increasing approximation errors. However, increasing the bit-rate by further splitting final bounding spaces makes it again appropriate to use SCT with this larger number of PCs, which in turn improves the performance as the underlying data is represented more accurately using more PCs. The performance for each scenario (number of PCs) saturates at "large" bit-rates, each of the saturation levels has been verified to be equal to the actual performance when using raw database descriptors with the same number of PCs. This shows that SCT reaches maximal possible performance at what is actually quite a low bit-rate (see figure 8.4(a)).

vectors are compared directly to the reconstructed database vectors.

Making the representation finer. As described in section 8.2.1, by increasing the bit-rate: the cell associated with it can be additionally split with a rate of 1 bit per split, encoding on which side of the split the descriptor is. Other schemes are certainly possible, e.g. representing the residual (the difference between the original descriptor and the reconstruction) with any of the other methods such as product quantization, LSH or ITQ.

Small dimensionality requirement. As discussed in section 8.2.1, SCT is not applicable when the number of splits per descriptor is smaller than the data dimensionality, since in this case many of the dimensions are not split thus causing large approximation errors. To compress vectors of large dimensionality using the SCT requires dimensionality reduction via PCA. Note that the requirement for "small" descriptor dimensionality is not as limiting as it might seem as PCA is commonly used for compacting descriptors such as VLAD (Jégou et al., 2010b) or even SIFT (Sánchez and Perronnin, 2011). There are many cases where "small" descriptors are used, e.g. Google's CONGAS descriptors are 40-D (Zheng et al., 2009), Simonyan et al. (2012) achieve impressive performance with 60-D, while Jégou and Chum (2012) demonstrate very good performance when reducing 130k-D VLAD vectors using PCA down to 128-D.

Furthermore, all baselines apart from PQ perform dimensionality reduction: RR and ITQ start from PCA (bitrate equals the number of PCs), LSH and SIK use random projections or Fourier features (bitrate equals the number of projections), and SH learns the projections from training data (bitrate equals the number of projections). Thus all methods apart from PQ actually suffer from a much worse problem than SCT since, for example, for 8 bits ITQ is forced to use only 8 PCs.

Adding descriptors to the database. For applications where the image database grows in time it is important to be able to add descriptors efficiently to the SCT. Adding a single descriptor does not disturb the tree structure much (e.g. imagine the example in figure 8.1 where a point is left out and added to the tree after the tree is built) as the sequence of splits is independent of the compressed data. Thus, all the split outcomes will remain the same (i.e. the tree does not need to be re-encoded) apart from the split that creates the leaf node (cell) that contains the new descriptor. By construction, if the cell is not empty then it contains exactly one descriptor and new splits are added in the same manner as when the original tree was built, i.e. until a split is encountered which discriminates between the two descriptors, namely the “old” and the “new” one. Thus, adding a descriptor to the SCT is very efficient and only requires access to at most one original database descriptor (zero if the relevant leaf cell is empty). Note that for the same operation none of the baselines requires access to the original descriptors. However, this limitation is not significant as the original descriptors do not need to be stored in RAM but can be stored on disk.

Use cases. SCT has been demonstrated to perform well for the problem of large scale image retrieval (section 8.3.3), searching a database of 580k images (represented by GIST descriptors) in 0.26 seconds. Since query-time speed is linear in the number of descriptors, with no changes to the system up to 7 million images can be searched immediately (3 seconds per query) on a single core. SCT can easily be parallelized, thus enabling 30 million images to be searched on a typical quad-core computer. Note that storing 30 million descriptors at 6 bits per descriptor requires only 23 MB.

For larger scale databases with billions of images memory requirements would still remain low, however processing power would be the limiting factor as a linear scan through the data is infeasible. In this case one can, in the manner of Jégou et al. (2008, 2011a), vector quantize the database descriptors coarsely and use SCT to compress the residual. At search time the query descriptor is vector quantized and only compared to database descriptors quantized to the cluster through the use of an inverted index (Sivic and Zisserman, 2003), while the SCT encoded residual is used for refining the matches. Searching 1 billion images quantized into 1k clusters would thus take about half a second using a single core processor (i.e. to decompress a single cluster containing 1M descriptors).

The same system can be used for large scale object retrieval where database images are typically represented using 1k local descriptors (e.g. SIFT (Lowe, 2004)). For this use case a query image is also represented using 1k descriptors, thus the same number of nearest neighbour searches would need to be issued. Searching 1 million images by quantizing the 1 billion database descriptors into 100k clusters and using 4 processing cores would yield results in 1.25 seconds (i.e. to decompress 1k clusters each containing 10k descriptors).

Summary. The Set Compression Tree (SCT) hugely outperforms all competing methods at extremely low bit-rates, making it the only tool of choice for very large scale retrieval purposes, where it is critical for fast retrieval that all the relevant data fits in RAM.

Chapter 9

Conclusions

In this chapter we summarize the achievements of the work presented in this thesis and discuss future research directions.

9.1 Achievements

In this thesis we have proposed several methods for improving large scale object retrieval. In chapter 4 we have proposed three methods to improve bag-of-words based retrieval methods, resulting in setting the state-of-the-art performance for all considered datasets (Oxford 5k, Oxford 105k, Paris). These results have still not been topped, though various other methods achieve competitive results under different setups (e.g. using independent vocabularies, additional training data or larger storage requirements), including Mikulik et al. (2010) and Tolias and Jégou (2013). Furthermore, in the same chapter we have introduced a new way of comparing the SIFT descriptors (RootSIFT) which has proven useful in non-retrieval applications as well; section 4.7 discusses the impact of this work in more detail.

In chapter 5 we worked on improving compact image representations, used in very large scale retrieval systems. In particular, we proposed improvements to the VLAD descriptor which set the state-of-the-art retrieval performance for both mid-dimensional (20k-D to 30k-D) and small (128-D) descriptors. Some of these results have been topped by the more recent works of Zhao et al. (2013a) and Delhumeau et al. (2013).

We also proposed to improve object retrieval by using multiple query images (chapter 6) and achieved a substantial increase in recall via this approach. Furthermore, we implemented a system capable of answering textual queries on-the-fly, which downloads representative images from Google and uses them to visually query the target image corpus. This is an important and useful search modality, as recognized by the later work of Fernando and Tuytelaars (2013).

Smooth textureless objects have been ignored for quite some time in object retrieval, mainly because they are difficult to handle. In chapter 7 we have introduced a bag-of-boundaries representation and demonstrated good retrieval performance in a challenging dataset containing smooth sculptures. We have also combined the bag-of-boundaries method with the traditional bag-of-words to visually search general (i.e. not only smooth) sculptures. This system was used to automatically name the imaged sculpture by exploiting Flickr meta-data associated with retrieved images.

Finally, in chapter 8 we have proposed a method, Set Compression Tree (SCT), for compressing sets of descriptors, useful for large scale memory-efficient approximate nearest neighbour search. The main novelty of SCT is that descriptors are compressed jointly thus enabling sharing of information resulting in very large compression rates. Due to SCT's impressive performance, we believe that we have started a very promising line of research into joint descriptor compression.

9.2 Future work

This section discusses some potentially promising directions for future work, as well as burning issues in large scale object retrieval for which we believe it is important for researchers to solve. They are organized by relevant topics which loosely correspond chapters in this thesis.

Matching descriptors

- It has long been noted that the bag-of-words based methods are quite sensitive to the choice of a visual vocabulary (Philbin et al., 2008). Similar problems occur in memory-efficient approximate nearest neighbour search (ANN) which starts with vector quantization (Jégou et al., 2008, 2011a), namely, the ANN search performance suffers if the quantization is learnt on an independent image dataset. This behaviour is quite unsatisfactory as it means that search quality worsens as images are added to the corpus, which is often the case in real-life where the target image corpus grows with time (e.g. like it is the case with Google, Facebook, Flickr, etc), unless the visual vocabulary is periodically rebuilt and all database images are reindexed. We believe further research is needed in order to find a way of constructing and using a single universal vocabulary which would achieve close, or even better, performance compared to the vocabulary built on the target dataset.

- As discussed in section 2.1.2, large scale retrieval by performing approximate nearest neighbour search for query descriptors is a promising research direction. Even though a few methods have been proposed (Jégou et al., 2008, 2011b, Qin et al., 2013), it is still an open question how exactly to use the estimated descriptor distances to rank images.

- The work in this thesis has initiated the use of classification methods for large scale object retrieval (section 4.3). Even though a few approaches with the same motivation have appeared since (Cao and Snavely, 2013, Chen et al., 2012, Gronat et al., 2013), we believe further investigation into discriminative learning for instance retrieval is needed. Amongst others, it would be useful to investigate the following two problems. Firstly, mining hard negatives is known to be very beneficial for discriminative methods but it is not clear how to do it automatically in an object retrieval scenario, where no labelled training data is available (apart from using GPS information like in (Gronat et al., 2013)). Secondly, all current approaches rely on the bag-of-words framework; it would be interesting to investigate discriminative approaches for methods that rely on accurate descriptor matching (section 2.1.2).

- Failures of traditional systems based on matching local descriptors are often shocking to a human. It seems it should be “easy” to discover false positive retrievals as the matching descriptors do not cover the entire object of interest and there are many regions which do not match. However, in reality it is quite hard to automatically estimate if two regions should not match, due to imperfect feature detection and description, as well as potential partial occlusions.

- The problem of burstiness of visual features (Jégou et al., 2009b) has been addressed in past by accounting for its effect when computing the similarity of images without spatial reranking (section 2.1.1.4). We have noticed that current spatial verification techniques are quite prone to errors when bursty images are considered, falsely verifying a match purely due to the fact that there are many putative feature correspondences thus making it likely to find a geometric transformation which accidentally has sufficient support. Further improvements of spatial verification are required in order to prevent false matches for bursty images.

Multiple query images

- Though effective, our method for using multiple query images can be prone to mistakes caused by outlier query images. Thus, it is likely that preprocessing query images, in order to remove outlier images or features, would improve the retrieval quality. Recent work by Fernando and Tuytelaars (2013) mines representative patterns in query images in order to remove outlying features. However, this is not an easy task as one has to be careful not to remove rare but useful features and thus deteriorate recall; this is exactly what happened in some of our early experiments on this topic.

Smooth object retrieval

- We focused our work on sculptures and relied on being able to perform automatic segmentation, though our method is somewhat robust to some segmentation failures. However, having to perform automatic segmentation is unsatisfactory as no such step is required for the very successful retrieval of textured objects. In order to truly bring smooth object to the same level as textured objects, i.e. enable retrieval of *any* smooth object and not just particular classes of objects like sculptures, it seems necessary to abandon automatic segmentation. It is not clear how to tackle this problem, but we insist it is a very important problem to solve and ignoring it will damage the Computer Vision community.

Set compression

- As demonstrated in this thesis, joint compression of vectors is a promising research direction. We have applied our SCT algorithm on the problem of memory-efficient approximate nearest neighbour search, however, we believe many more uses exist. For example, extracting descriptors densely from an image can yield hundreds of thousands of descriptors, thus requiring megabytes of storage per image. SCT would enable compact storage of all dense descriptors for a large image corpus, this could be useful in order to, for example, preform dense 3D reconstruction.

- It should be possible to boost the performance of SCT by improving the way additional bits are used to refine the descriptor estimates. The refinement is done on a per-descriptor basis, so using a state-of-the-art method for compressing individual descriptors (e.g. product quantization of Jégou et al. (2011a)) instead of our naive method is almost guaranteed to improve the compression quality.

Bibliography

A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE PAMI*, 28(1):44–58, 2006. 117, 126

M Aly, M. Munich, and P. Perona. Distributed kd-trees for retrieval from very large image collections. In *Proc. BMVC.*, 2011. 31

M Aly, M. Munich, and P. Perona. Compactkdt: Compact signatures for accurate large scale object recognition. In *IEEE Workshop on Applications of Computer Vision*, 2012a. 33

R. Aly, K. McGuinness, S. Chen, N. E. O'Connor, K. Chatfield, O. M. Parkhi, R. Arandjelović, A. Zisserman, B. Fernando, T. Tuytelaars, J. Schwenninger, D. Oneata, M. Douze, J. Revaud, D. Potapov, H. Wang, Z. Harchaoui, J. Verbeek, and C. Schmid. AXES at TRECVID 2012: KIS, INS, and MED. In *TREC 2012 Video Retrieval Evaluation*, 2012b. 12

R. Aly, R. Arandjelović, K. Chatfield, M. Douze, B. Fernando, Z. Harchaoui, K. McGuiness, N. O'Connor, D. Oneata, O. Parkhi, D. Potapov, J. Revaud, C. Schmid, J.-L. Schwenninger, D. Scott, T. Tuytelaars, J. Verbeek, H. Wang, and A. Zisserman. The AXES submissions at TrecVid 2013. In *TRECVID Workshop*, 2013. 12

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997. 162

A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Comm. ACM*, 2008. 162, 176

O. Arandjelović. Object matching using boundary descriptors. In *Proc. BMVC.*, 2012. 157

R. Arandjelović and A. Zisserman. Efficient image retrieval for 3D structures. In *Proc. BMVC.*, 2010. 12

R. Arandjelović and A. Zisserman. Smooth object retrieval using a bag of boundaries. In *Proc. ICCV*, 2011. 12

R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. CVPR*, 2012a. 11

R. Arandjelović and A. Zisserman. Name that sculpture. In *ACM ICMR*, 2012b. 12

R. Arandjelović and A. Zisserman. Multiple queries for large scale specific object retrieval. In *Proc. BMVC.*, 2012c. 12

R. Arandjelović and A. Zisserman. All about VLAD. In *Proc. CVPR*, 2013a. 11

R. Arandjelović and A. Zisserman. Extremely low bit-rate nearest neighbor search using a Set Compression Tree. Technical report, Department of Engineering Science, University of Oxford, 2013b. 12

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *Proc. CVPR*, 2009. 117, 120

J. Aslam and M. Montague. Models for metasearch. In *Proc. SIGIR*, pages 276–284, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-331-6. doi: http://doi.acm.org/10.1145/383952.384007. 102

A. Babenko and V. Lempitsky. The inverted multi-index. In *Proc. CVPR*, 2012. 33, 161

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. ECCV*, May 2006. 15

S. Belongie and J. Malik. Shape matching and object recognition using shape contexts. *IEEE PAMI*, 24(24), 2002. 117, 122

A. Bergamo, S. N. Sinha, and L. Torresani. Leveraging structure from motion to learn discriminative codebooks for scalable landmark classification. In *Proc. CVPR*, 2013. 21

G. Bergel, A. Franklin, M. Heaney, R. Arandjelović, A. Zisserman, and D. Funke. Content-based image-recognition on printed broadside ballads: The Bodleian libraries' ImageMatch tool. In *IFLA World Library and Information Congress*, 2013. 6, 12

A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proc. ICCV*, 2007. 162

A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, 1997. 23, 28

M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE PAMI*, 33(1):43–57, 2011. 15

C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART. In *TREC-3 Proc.*, 1995. 40

M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proc. ECCV*, 2010. 15

S. Cao and N. Snavely. Graph-based discriminative learning for location recognition. In *Proc. CVPR*, 2013. 25, 76, 189

Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang. Spatial-bag-of-features. In *Proc. CVPR*, 2010. 39

O. Carmichael and M. Hebert. Shape-based recognition of wiry objects. In *Proc. CVPR*, 2003. 115

V. Chandrasekhar, S. Tsai, Y. Reznik, G. Takacs, D. Chen, and B. Girod. Compressing feature sets with digital search trees. In *International Workshop on Mobile Vision*, 2011. 162

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2:27:1–27:27, 2011. 63, 106

D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, H. Chen, R. Vedantham, R. Grzeszczuk, and B. Girod. Residual enhanced visual vectors for on-device image matching. In *Asilomar*, 2011. 47, 87

Y. Chen, X. Li, A. Dick, and A. Hengel. Boosting object retrieval with group queries. *IEEE Signal Processing Letters*, 2012. 76, 114, 189

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IT*, 14, 1968. 23

O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE PAMI*, 32(2):371–377, 2010a. 67

O. Chum and J. Matas. Unsupervised discovery of co-occurrence in sparse high dimensional data. In *Proc. CVPR*, 2010b. 23

O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proc. CIVR*, 2007a. 27, 43

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*, 2007b. 40, 41, 55, 56, 59, 61, 102

O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proc. BMVC.*, 2008. 28, 43

O. Chum, M. Perďoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *Proc. CVPR*, 2009. 39

O. Chum, A. Mikulik, M. Perďoch, and J. Matas. Total recall II: Query expansion revisited. In *Proc. CVPR*, 2011. 40, 57, 63, 71, 72, 80

I. J. Cox, M. Miller, T. Minka, T. Papathomas, and P. N. Yianilos. The bayesian image retrieval system, PicHunter: Theory, implementation and psychophysical experiments. *IEEE Transactions on Image Processing*, 2000. 102

M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 2008. 23

N. Dalal and B Triggs. Histogram of Oriented Gradients for Human Detection. In *Proc. CVPR*, volume 2, pages 886–893, 2005. 15, 122

D. Damen, P. Bunnun, A. Calway, and W. Mayol-Cuevas. Real-time learning and detection of 3d texture-less objects: A scalable approach. In *Proc. BMVC.*, 2012. 118

J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez. Revisiting the VLAD image representation. In *ACM Multimedia*, 2013. 47, 48, 75, 187

P. Doubek, J. Matas, M. Perďoch, and O. Chum. Image matching and retrieval by repetitive patterns. In *Proc. ICPR*, 2010. 23

M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of GIST descriptors for web-scale image search. In *Proc. CIVR*, 2009. 43

M. Douze, H. Jégou, C. Schmid, and P. Pérez. Compact video description for copy detection with precise temporal alignment. In *Proc. ECCV*, 2010. 28

M. Douze, J. Revaud, C. Schmid, and H. Jégou. Stable hyper-pooling and query expansion for event detection. In *Proc. ICCV*, 2013. 76

F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3D mapping with an RGB-D camera. *IEEE Transactions on Robotics*, PP(99):1–11, 2013. 76

R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *Proc. ICCV*, 2005. 101

B. Fernando and T. Tuytelaars. Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation. In *Proc. ICCV*, 2013. 75, 114, 187, 190

V. Ferrari, T. Tuytelaars, and L. V. Gool. Object detection by contour segment networks. In *Proc. ECCV*, 2006. 118

V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *IJCV*, 2010. 118

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981. 36

S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. I know what you did last summer: object-level auto-annotation of holiday snaps. In *Proc. ICCV*, 2009. 139

V. Garg, S. Chandra, and C. V. Jawahar. Sparse discriminative Fisher vectors in visual classification. In *Proc. CVPR*, 2013. 76

T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *Proc. CVPR*, 2013. 32

Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. CVPR*, 2011. 161, 162, 172, 174, 175, 177, 178

W. E. L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE PAMI*, 9(4):469–482, 1987. 117

P. Gronat, G. Obozinski, J. Sivic, and T. Pajdla. Learning and calibrating per-location classifiers for visual place recognition. In *Proc. CVPR*, 2013. 25, 26, 76, 189

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 34

K. A. Heller and Z. Ghahramani. A simple bayesian framework for content-based image retrieval. In *Proc. CVPR*, 2006. 101

S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of texture-less objects. *IEEE PAMI*, 2012. 117, 118

D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Proc. ICCV*, 2005. 117

E. Hsiao and M. Hebert. Gradient networks: Explicit shape matching without extracting edges. In *AAAI*, 2013. 117

D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. 171

D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. ICCV*, pages 102–111, 1987. 117

I. Ivanov, P. Vajda, L. Goldmann, J.-S. Lee, and T. Ebrahimi. Object-based tag propagation for semi-automatic annotation of images. In *ACM Multimedia information retrieval*, 2010. 139

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, pages 487–493, 1998. 44

M. Jain, H. Jégou, and P. Gros. Asymmetric hamming embedding. In *ACM Multimedia*, 2011. 75

H. Jégou and O. Chum. Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening. In *Proc. ECCV*, 2012. 23, 43, 44, 47, 48, 75, 79, 80, 85, 94, 95, 96, 97, 183

H. Jégou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *Proc. CVPR*, 2007. 24, 25, 33

H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proc. ECCV*, pages 304–317, 2008. 31, 35, 38, 50, 58, 80, 94, 160, 162, 184, 188, 189

H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *Proc. ICCV*, Sep 2009a. 29, 43

H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *Proc. CVPR*, Jun 2009b. 10, 22, 46, 77, 80, 85, 190

H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87(3):316–336, 2010a. 20, 30, 58, 75

H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010b. 9, 45, 46, 47, 58, 77, 79, 80, 85, 88, 94, 95, 97, 162, 172, 177, 181, 183

H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE PAMI*, 2011a. 32, 33, 160, 162, 167, 174, 176, 177, 178, 184, 188, 191

H. Jégou, M. Douze, and C. Schmid. Exploiting descriptor distances for precise image search. Technical report, INRIA, 2011b. 32, 189

H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local images descriptors into compact codes. *IEEE PAMI*, 2012. 45, 46, 47, 48, 79, 85, 94, 95, 97

Y. Jiang, J. Meng, and J. Yuan. Randomized visual phrases for object search. In *Proc. CVPR*, 2012. 39

Y. Jin, L. Khan, L. Wang, and M. Awad. Image annotations by combining multiple evidence and wordnet. In *ACM Multimedia*, 2005. 139, 152

M. Johnson. Generalized descriptor compression for storage and matching. In *Proc. BMVC.*, 2010. 60

M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proc. CVPR*, 2013. 76

H. Kang, M. Hebert, and T. Kanade. Image matching with distinctive visual vocabulary. In *IEEE Workshop on Applications of Computer Vision*, 2011. 27

J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *Proc. ECCV*, 2010. 26, 63

J. Koenderink. *Solid Shape.* MIT Press, 1990. 116

J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D object representations for fine-grained categorization. In *Proc. ICCV*, 2013. 76

D. C. Lee, Q. Ke, and M. Isard. Partition min-hash for partial duplicate image discovery. In *Proc. ECCV*, 2010. 39

V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. CVPR*, pages 775–781, 2005. 15

S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proc. ICCV*, pages 2548–2555, 2011. 15

X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *Proc. ECCV*, 2008. 28

T. Liu and H. A. Rosenberg, C. Rowley. Clustering billions of images with large scale nearest neighbor search. In *IEEE Workshop on Applications of Computer Vision*, 2008. 28

D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004. 9, 15, 30, 33, 39, 57, 121, 127, 160, 174, 181, 185

M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Proc. CVPR*, 2008. 120

A. Makadia. Feature tracking for wide-baseline image retrieval. In *Proc. ECCV*, 2010. 20, 21, 30

T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *Proc. ICCV*, 2011. 25, 104, 106

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. 16, 17

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC.*, pages 384–393, 2002. 15

K. McGuinness, S. Chen, M. Frappier, H. Lee, N. E. O'Connor, A. Smeaton, R. Aly, R. Ordelman, M. Kleppe, H. Beunders, R. Arandjelović, A. Vedaldi, A. Zisserman, M. Juneja, C. V. Jawahar, J. Schwenninger, S. Tschopel, and D. Schneider. AXES at TRECVID 2011. In *TREC 2011 Video Retrieval Evaluation*, 2011. 12

K. McGuinness, N.E. O'Connor, R. Aly, F. Dejong, K. Chatfield, O. Parkhi, R. Arandjelović, A. Zisserman, M. Douze, and C. Schmid. The AXES PRO video search system. In *ACM ICMR*, 2013. 12

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE PAMI*, 2004a. 15

K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 1(60):63–86, 2004b. 15, 29, 36, 56, 71, 72, 105, 142

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65 (1/2):43–72, 2005. 115, 127

A. Mikulik, M. Perďoch, O. Chum, and J. Matas. Learning a fine vocabulary. In *Proc. ECCV*, 2010. 20, 21, 30, 57, 58, 71, 72, 186

P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. *IJCV*, 2006. 115

J.-M. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2:438–469, 2009. 21

M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithmic configuration. In *Proc. VISAPP*, 2009. 162

Neven, Google. Machine learning in Google Goggles. In *keynote talk, ICML, http://techtalks.tv/talks/54457/*, 2011. 115, 138, 158

Neven, Google. Fine-grained classification of weakly textured objects. In *invited talk, The ECCV Workshop on Computer Vision for the Web*, 2012. 158

D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, pages 2161–2168, 2006. 18, 19, 20, 30

M. Norouzi and D. J. Fleet. Cartesian k-means. In *Proc. CVPR*, 2013. 32

A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001. 43, 162, 172, 174

O. Paul, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. F. Smeaton, and G. Quéenot. TRECVID 2011 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2011. 108

O. Pele and M. Werman. A linear time histogram metric for improved sift matchings. In *Proc. ECCV*, 2008. 60

O. Pele and M. Werman. The quadratic-chi histogram distance family. In *Proc. ECCV*, 2010. 60

M. Perďoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *Proc. CVPR*, 2009. 29, 34, 68, 71, 72, 73, 80, 94, 105

F. Perronnin and D. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, 2007. 44, 45

F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proc. CVPR*, 2010a. 46, 47, 79, 85, 94, 95

F. Perronnin, J. Sanchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *Proc. CVPR*, 2010b. 59

F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010c. 45, 46

J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *Proc. ICVGIP*, 2008. 41, 64, 67

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007. 19, 30, 36, 37, 49, 53, 55, 56, 59, 67, 104, 105, 108, 127, 128, 132, 137, 142

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. 20, 21, 30, 37, 50, 57, 58, 71, 72, 75, 80, 188

J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *Proc. ECCV*, 2010. 15, 21, 58, 69

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelyhood methods. *Advances in Large Margin Classifiers*, pages 61–74, 1999. 106

J. Prankl, T. M orwald, M. Zillich, , and M. Vincze. Probabilistic cue integration for real-time object pose tracking. In *Computer Vision Systems*, volume 7963 of *Lecture Notes in Computer Science*, pages 254–263. Springer Berlin Heidelberg, 2013. 76

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. 24, 25, 33, 41, 71

D. Qin, C. Wengert, and L. V. Gool. Query adaptive similarity for large scale object retrieval. In *Proc. CVPR*, 2013. 33, 75, 189

T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *Proc. CIVR*, 2008. 138

M. Raginsky and S. Lazebnik. Locality sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009. 161, 162, 175, 176

S. Rahtz, A. Dutton, D. Kurtz, G. Klyne, A. Zisserman, and R. ArandjeloviÄĞ. CLAROS – Collaborating on delivering the future of the past. In *Digital Humanities*, 2011. 12

M. Rastegari, C. Fang, and L. Torresani. Scalable object-class retrieval with approximate and top-k ranking. In *Proc. ICCV*, 2011. 161

J. Revaud, M. Douze, C. Schmid, and H. Jégou. Event retrieval in large video collections with circulant temporal encoding. In *Proc. CVPR*, 2013. 48, 76

E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE PAMI*, 32:105–119, 2010. 15

C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. *Proc. ACM SIGGRAPH*, 23(3):309–314, 2004. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1015706.1015720. 39

C. A. Rothwell, A. Zisserman, J. L. Mundy, and D. A. Forsyth. Efficient model library access by projectively invariant indexing functions. In *Proc. CVPR*, pages 109–114, 1992. 117

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1999. 40

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986. 16

J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Proc. CVPR*, 2011. 161, 183

T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. Image retrieval for image-based localization revisited. In *Proc. BMVC.*, 2012. 32

F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *Proc. ECCV*, volume 1, pages 414–431. Springer-Verlag, 2002. 15, 37

G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proc. CVPR*, 2007. 20, 42

C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE PAMI*, 19(5):530–534, May 1997. 35

J. A. Shaw and E. A. Fox. Combination of multiple searches. In *The Second Text REtrieval Conference (TREC-2)*, pages 243–252, 1994. 102

X. Shen, Z. Lin, J. Brandt, S. Avidan, and Y. Wu. Object retrieval and localization with spatially-constrained similarity measure and k-NN reranking. In *Proc. CVPR*, 2012a. 38, 40

X. Shen, Z. Lin, J. Brandt, and Y. Wu. Mobile product image search by automatic query object extraction. In *Proc. CVPR*, 2012b. 38

C. Silpa-Anan and R. Hartley. Localization using an image-map. In *Australasian Conf. on Robotics and Automation*, 2004. 162

K. Simonyan, A. Vedaldi, and A. Zisserman. Descriptor learning using convex optimisation. In *Proc. ECCV*, 2012. 15, 21, 183

K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *Proc. BMVC.*, 2013a. 76

K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. Technical report, Department of Engineering Science, University of Oxford, 2013b. 69

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, volume 2, pages 1470–1477, 2003. 16, 18, 19, 22, 27, 34, 35, 94, 95, 97, 105, 160, 173, 184

J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE PAMI*, 31(4):591–606, 2009. 18

H. Stewenius, S. H. Gunderson, and J. Pilet. Size matters: exhaustive geometric verification for image retrieval. In *Proc. ECCV*, 2012. 29, 37

E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *Proc. CVPR*, 2008. 15

G. Tolias and Y. Avrithis. Speeded-up, relaxed spatial matching. In *Proc. ICCV*, 2011. 37

G. Tolias and H. Jégou. Local visual query expansion: Exploiting an image collection to refine local descriptors. Technical Report RR-8325, INRIA, 2013. 41, 75, 186

G. Tolias, Y. Avrithis, and H. Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *Proc. ICCV*, 2013. 99

A. Torii, J. Sivic, and T. Pajdla. Visual localization by linear combination of image descriptors. In *International Workshop on Mobile Vision*, 2011. 42, 90

A. Torii, J. Sivic, T. Pajdla, and M. Okutomi. Visual place recognition with repetitive structures. In *Proc. CVPR*, 2013. 20, 23, 76

A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE PAMI*, 2008. 161, 174, 175

L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *Proc. ECCV*, pages 776–789, sep 2010. 101

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. 9, 26, 41, 42, 55, 59, 64, 65, 66, 67, 68

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proc. CVPR*, 2010. 59, 121

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE PAMI*, 2011. 23

C. Wang, F. Jing, L. Zhang, and H. Zhang. Image annotation refinement using random walk with restarts. In *ACM Multimedia*, 2006. 139, 152

X. Wang, M Yang, T. Cour, S. Zhu, K. Yu1, and T. X. Han. Contextual weighting for vocabulary tree based image retrieval. In *Proc. ICCV*, 2011. 39

X-J. Wang, L. Zhang, and C. Liu. Duplicate discovery on 2 billion internet images. In *IEEE Workshop on Large Scale Computer Vision*, 2013. 27, 28

Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. 161, 162, 175, 177

Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *Proc. ECCV*, 2012. 161

J. Winn, Criminisi, A., and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. ICCV*, 2005. 60

Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *Proc. CVPR*, 2009. 39

J. Zhang, X. Long, and T. Suel. Performance of compressed inverted list caching in search engines. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 2008. 29

S. Zhang, Q. Huang, G. Hua, S. Jiang, and Q. Gao, W. Tian. Building contextual visual vocabulary for large-scale image applications. In *ACM Multimedia*, 2010. 39

Y. Zhang, Z. Jia, and T. Chen. Image retrieval with geometry-preserving visual phrases. In *Proc. CVPR*, 2011. 38

Z. Zhang, R. Deriche, O. D. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995. 35

W.-L. Zhao, X. Wu, and C.-W. Ngo. On the annotation of web videos by efficient near-duplicate search. *IEEE Transactions on Multimedia*, 12(5):448–461, 2010. 38

W.-L. Zhao, H. Jégou, and G. Gravier. Oriented pooling for dense and non-dense rotation-invariant features. In *Proc. BMVC.*, 2013a. 48, 75, 187

W.-L. Zhao, H. Jégou, and G. Gravier. Sim-Min-Hash: An efficient matching technique for linking large image collections. In *ACM Multimedia*, 2013b. 28, 75

L. Zheng, S. Wang, Z. Liu, and Q. Tian. Lp-Norm IDF for large scale image search. In *Proc. CVPR*, 2013. 24

Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: building a web-scale landmark recognition engine. In *Proc. CVPR*, 2009. 15, 183

F. Zhou, J. Brandt, and Z. Lin. Exemplar-based graph matching for robust facial landmark localization. In *Proc. ICCV*, 2013. 76

X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proc. ECCV*, 2010. 45

J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2):6, 2006. 29