Video Google: Efficient Visual Search of Videos

Josef Sivic and Andrew Zisserman

Department of Engineering Science
University of Oxford
Oxford, OX1 3PJ1
{josef,az}@robots.ox.ac.uk
http://www.robots.ox.ac.uk/∼vgg

Abstract. We describe an approach to object retrieval which searches for and localizes all the occurrences of an object in a video, given a query image of the object. The object is represented by a set of viewpoint invariant region descriptors so that recognition can proceed successfully despite changes in viewpoint, illumination and partial occlusion. The temporal continuity of the video within a shot is used to track the regions in order to reject those that are unstable.

Efficient retrieval is achieved by employing methods from statistical text retrieval, including inverted file systems, and text and document frequency weightings. This requires a visual analogy of a word which is provided here by vector quantizing the region descriptors. The final ranking also depends on the spatial layout of the regions. The result is that retrieval is immediate, returning a ranked list of shots in the manner of Google.

We report results for object retrieval on the full length feature films 'Groundhog Day' and 'Casablanca'.

1 Introduction

The aim of this work is to retrieve those key frames and shots of a video containing a particular object with the ease, speed and accuracy with which Google retrieves text documents (web pages) containing particular words. This chapter investigates whether a text retrieval approach can be successfully employed for this task.

Identifying an (identical) object in a database of images is now reaching some maturity. It is still a challenging problem because an object's visual appearance may be very different due to viewpoint and lighting, and it may be partially occluded, but successful methods now exist [7,8,9,11,13,14,15,16,20,21]. Typically an object is represented by a set of overlapping regions each represented by a vector computed from the region's appearance. The region extraction and descriptors are built with a controlled degree of invariance to viewpoint and illumination conditions. Similar descriptors are computed for all images in the database. Recognition of a particular object proceeds by nearest neighbour matching of the descriptor vectors, followed by disambiguating using local spatial coherence (such as common neighbours, or angular ordering), or global relationships (such as epipolar geometry or a planar homography).

J. Ponce et al. (Eds.): Toward Category-Level Object Recognition, LNCS 4170, pp. 127–144, 2006.

We explore whether this type of approach to recognition can be recast as text retrieval. In essence this requires a visual analogy of a word, and here we provide this by vector quantizing the descriptor vectors. However, it will be seen that pursuing the analogy with text retrieval is more than a simple optimization over different vector quantizations. There are many lessons and rules of thumb that have been learnt and developed in the text retrieval literature and it is worth ascertaining if these also can be employed in visual retrieval.

The benefits of this approach is that matches are effectively pre-computed so that at run-time frames and shots containing any particular object can be retrieved with no-delay. This means that any object occurring in the video (and conjunctions of objects) can be retrieved even though there was no explicit interest in these objects when descriptors were built for the video. However, we must also determine whether this vector quantized retrieval misses any matches that would have been obtained if the former method of nearest neighbour matching had been used.

Review of text retrieval: Text retrieval systems generally employ a number of standard steps [2]: the documents are first parsed into words, and the words are represented by their stems, for example 'walk', 'walking' and 'walks' would be represented by the stem 'walk'. A stop list is then used to reject very common words, such as 'the' and 'an', which occur in most documents and are therefore not discriminating for a particular document. The remaining words are then assigned a unique identifier, and each document is represented by a vector with components given by the frequency of occurrence of the words the document contains. In addition the components are weighted in various ways (described in more detail in section 4), and in the case of Google the weighting of a web page depends on the number of web pages linking to that particular page [4]. All of the above steps are carried out in advance of actual retrieval, and the set of vectors representing all the documents in a corpus are organized as an *inverted file* [22] to facilitate efficient retrieval. An inverted file is structured like an ideal book index. It has an entry for each word in the corpus followed by a list of all the documents (and position in that document) in which the word occurs.

A text is retrieved by computing its vector of word frequencies and returning the documents with the closest (measured by angles) vectors. In addition the degree of match on the ordering and separation of the words may be used to rank the returned documents.

Chapter outline: Here we explore visual analogies of each of these steps. Section 2 describes the visual descriptors used. Section 3 then describes their vector quantization into visual 'words', and sections 4 and 5 weighting and indexing for the vector model. These ideas are then evaluated on a ground truth set of six object queries in section 6. Object retrieval results are shown on two feature films: 'Groundhog Day' [Ramis, 1993] and 'Casablanca' [Curtiz, 1942].

Although previous work has borrowed ideas from the text retrieval literature for image retrieval from databases (e.g. [19] used the weighting and inverted file schemes) to the best of our knowledge this is the first systematic application of these ideas to object retrieval in videos.

(a) (b)

Fig. 1. Object query example I. (a) Top row: (left) a frame from the movie 'Groundhog Day' with an outlined query region and (right) a close-up of the query region delineating the object of interest. Bottom row: (left) all 1039 detected affine covariant regions superimposed and (right) close-up of the query region. (b) (left) two retrieved frames with detected regions of interest and (right) a close-up of the images with affine covariant regions superimposed. These regions match to a subset of the regions shown in (a). Note the significant change in foreshortening and scale between the query image of the object, and the object in the retrieved frames. For this query there are four correctly retrieved shots ranked 1, 2, 3 and 9. Querying all the 5,640 keyframes of the entire movie took 0.36 seconds on a 2GHz Pentium.

2 Viewpoint Invariant Description

Two types of viewpoint covariant regions are computed for each frame. The first is constructed by elliptical shape adaptation about a Harris [5] interest point. The method involves iteratively determining the ellipse centre, scale and shape. The scale is determined by the local extremum (across scale) of a Laplacian, and the shape by maximizing intensity gradient isotropy over the elliptical region [3,6]. The implementation details are given in [11,15]. This region type is referred to as Shape Adapted (SA).

The second type of region is constructed by selecting areas from an intensity watershed image segmentation. The regions are those for which the area is approximately stationary as the intensity threshold is varied. The implementation details are given in [10]. This region type is referred to as Maximally Stable (MS).

Two types of regions are employed because they detect different image areas and thus provide complementary representations of a frame. The SA regions tend to be centred on corner like features, and the MS regions correspond to blobs of high contrast with respect to their surroundings such as a dark window on a grey wall. Both types of regions are represented by ellipses. These are computed at twice the originally detected region size in order for the image appearance to be more discriminating. For a 720×576 pixel video frame the number of regions computed is typically 1,200. An example is shown in Figure 1.

Each elliptical affine invariant region is represented by a 128-dimensional vector using the SIFT descriptor developed by Lowe [7]. In [12] this descriptor was shown to be superior to others used in the literature, such as the response of a set of steerable filters [11] or orthogonal filters [15], and we have also found SIFT to be superior (by comparing scene retrieval results against ground truth [18]). One reason for this superior performance is that SIFT, unlike the other descriptors, is designed to be invariant to a shift of a few pixels in the region position, and this localization error is one that often occurs. Combining the SIFT descriptor with affine covariant regions gives region description vectors which are invariant to affine transformations of the image. Note, both region detection and the description is computed on monochrome versions of the frames, colour information is not currently used in this work.

To reduce noise and reject unstable regions, information is aggregated over a sequence of frames. The regions detected in each frame of the video are tracked using a simple constant velocity dynamical model and correlation. Any region which does not survive for more than three frames is rejected. This 'stability check' significantly reduces the number of regions to about 600 per frame.

3 Building a Visual Vocabulary

The objective here is to vector quantize the descriptors into clusters which will be the visual 'words' for text retrieval. The vocabulary is constructed from a subpart of the movie, and its matching accuracy and expressive power are evaluated on the entire movie, as described in the following sections. The running example is for the movie 'Groundhog Day'.

The vector quantization is carried out here by K-means clustering, though other methods (K-medoids, histogram binning, etc) are certainly possible.

3.1 Implementation

Each descriptor is a 128-vector, and to simultaneously cluster all the descriptors of the movie would be a gargantuan task. Instead a random subset of 437 frames is selected. Even with this reduction there are still 200K descriptors that must be clustered.

The Mahalanobis distance is used as the distance function for the K-means clustering. The distance between two descriptors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, is then given by

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top \Sigma^{-1} (\boldsymbol{x}_1 - \boldsymbol{x}_2)}.$$

The covariance matrix Σ is determined by (i) computing covariances for descriptors throughout tracks within several shots, and (ii) assuming Σ is the same for all tracks (i.e. independent of the region) so that covariances for tracks can be aggregated. In this manner sufficient measurements are available to estimate all elements of Σ. Details are given in [18]. The Mahalanobis distance enables the more noisy components of the 128–vector to be weighted down, and also decorrelates the components. Empirically there is a small degree of correlation. As is standard, the descriptor space is affine transformed by the square root of Σ so that Euclidean distance may be used.

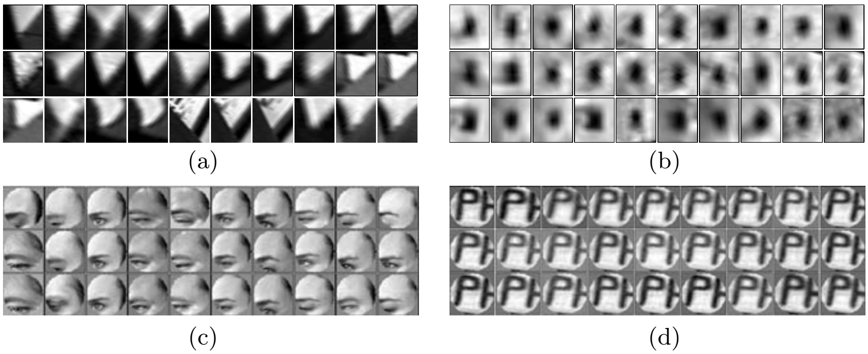


Fig. 2. Samples of normalized affine covariant regions from clusters corresponding to a single visual word: (a,c,d) Shape Adapted regions; (b) Maximally Stable regions. Note that some visual words represent generic image structures, e.g. corners (a) or blobs (b), and some visual words are rather specific, e.g. an eye (c) or a letter (d).

About 6K clusters are used for Shape Adapted regions, and about 10K clusters for Maximally Stable regions. The ratio of the number of clusters for each type is chosen to be approximately the same as the ratio of detected descriptors of each type. The number of clusters was chosen empirically to maximize matching performance on a ground truth set for scene retrieval [18]. The K-means algorithm is run several times with random initial assignments of points as cluster centres, and the lowest cost result used.

Figure 2 shows examples of regions belonging to particular clusters, i.e. which will be treated as the same visual word. The clustered regions reflect the properties of the SIFT descriptors which penalize intensity variations amongst regions less than cross-correlation. This is because SIFT emphasizes orientation of gradients, rather than the position of a particular intensity within the region.

The reason that SA and MS regions are clustered separately is that they cover different and largely independent regions of the scene. Consequently, they may be thought of as different vocabularies for describing the same scene, and thus should have their own word sets, in the same way as one vocabulary might describe architectural features and another the material quality (e.g. defects, weathering) of a building.

4 Visual Indexing Using Text Retrieval Methods

In text retrieval each document is represented by a vector of word frequencies. However, it is usual to apply a weighting to the components of this vector [2], rather than use the frequency vector directly for indexing. Here we describe the standard weighting that is employed, and then the visual analogy of document retrieval to frame retrieval.

The standard weighting is known as 'term frequency–inverse document frequency', *tf-idf*, and is computed as follows. Suppose there is a vocabulary of V words, then each document is represented by a vector

$$\boldsymbol{v}_d = (t_1, ..., t_i, ..., t_V)^\top$$

of weighted word frequencies with components

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

where n_{id} is the number of occurrences of word i in document d, n_d is the total number of words in the document d, n_i is the number of documents containing term i and N is the number of documents in the whole database. The weighting is a product of two terms: the *word frequency* n_{id}/n_d, and the *inverse document frequency* $\log N/n_i$. The intuition is that word frequency weights words occurring often in a particular document, and thus describes it well, whilst the inverse document frequency downweights words that appear often in the database.

At the retrieval stage documents are ranked by their normalized scalar product (cosine of angle)

$$f_d = \frac{\boldsymbol{v}_q{}^\top \boldsymbol{v}_d}{\sqrt{\boldsymbol{v}_q{}^\top \boldsymbol{v}_q}\,\sqrt{\boldsymbol{v}_d{}^\top \boldsymbol{v}_d}} \qquad (1)$$

between the query vector $\boldsymbol{v}_q$ and all document vectors $\boldsymbol{v}_d$ in the database.

In our case the query vector is given by the visual words contained in a user specified sub-part of an image, and the frames are ranked according to the similarity of their weighted vectors to this query vector.

4.1 Stop List

Using a stop list analogy the most frequent visual words that occur in almost all images are suppressed. The top 5% and bottom 5% are stopped. In our case the very common words are due to large clusters of over 3K points. These might correspond to small specularities (highlights), for example, which occur throughout many scenes. The stop list boundaries were determined empirically to reduce the number of mismatches and size of the inverted file while keeping sufficient visual vocabulary.

Figure 4 shows the benefit of imposing a stop list – the very common visual words occur at many places in the image and are responsible for mis-matches. Most of these are removed once the stop list is applied. The removal of the remaining mis-matches is described next.

4.2 Spatial Consistency

Google increases the ranking for documents where the searched for words appear close together in the retrieved texts (measured by word order). This analogy is

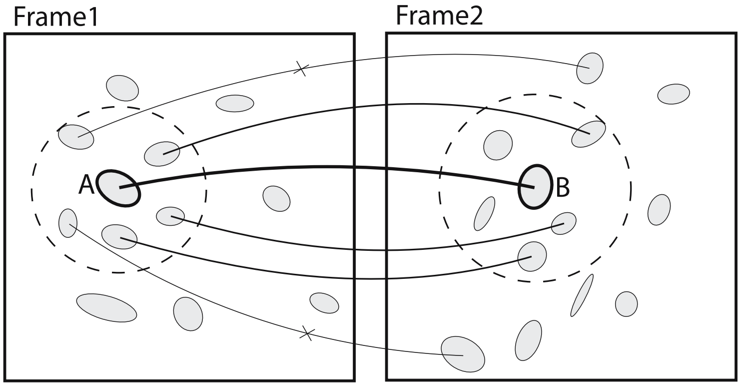


Fig. 3. Illustration of spatial consistency voting. To verify a pair of matching regions (A,B) a circular search area is defined by the k (=5 in this example) spatial nearest neighbours in both frames. Each match which lies within the search areas in both frames casts a vote in support of match (A,B). In this example three supporting matches are found. Matches with no support are rejected.

especially relevant for querying objects by an image, where matched covariant regions in the retrieved frames should have a similar spatial arrangement [14,16] to those of the outlined region in the query image. The idea is implemented here by first retrieving frames using the weighted frequency vector alone, and then re-ranking them based on a measure of spatial consistency.

Spatial consistency can be measured quite loosely simply by requiring that neighbouring matches in the query region lie in a surrounding area in the retrieved frame. It can also be measured very strictly by requiring that neighbouring matches have the same spatial layout in the query region and retrieved frame. In our case the matched regions provide the affine transformation between the query and retrieved image so a point to point map is available for this strict measure.

We have found that the best performance is obtained in the middle of this possible range of measures. A search area is defined by the 15 nearest spatial neighbours of each match, and each region which also matches within this area casts a vote for that frame. Matches with no support are rejected. The final score of the frame is determined by summing the spatial consistency votes, and adding the frequency score f_d given by (1). Including the frequency score (which ranges between 0 and 1) disambiguates ranking amongst frames which receive the same number of spatial consistency votes. The object bounding box in the retrieved frame is determined as the rectangular bounding box of the matched regions after the spatial consistency test. The spatial consistency voting is illustrated in figure 3. This works very well as is demonstrated in the last row of figure 4, which shows the spatial consistency rejection of incorrect matches. The object retrieval examples presented in this chapter employ this ranking measure and amply demonstrate its usefulness.

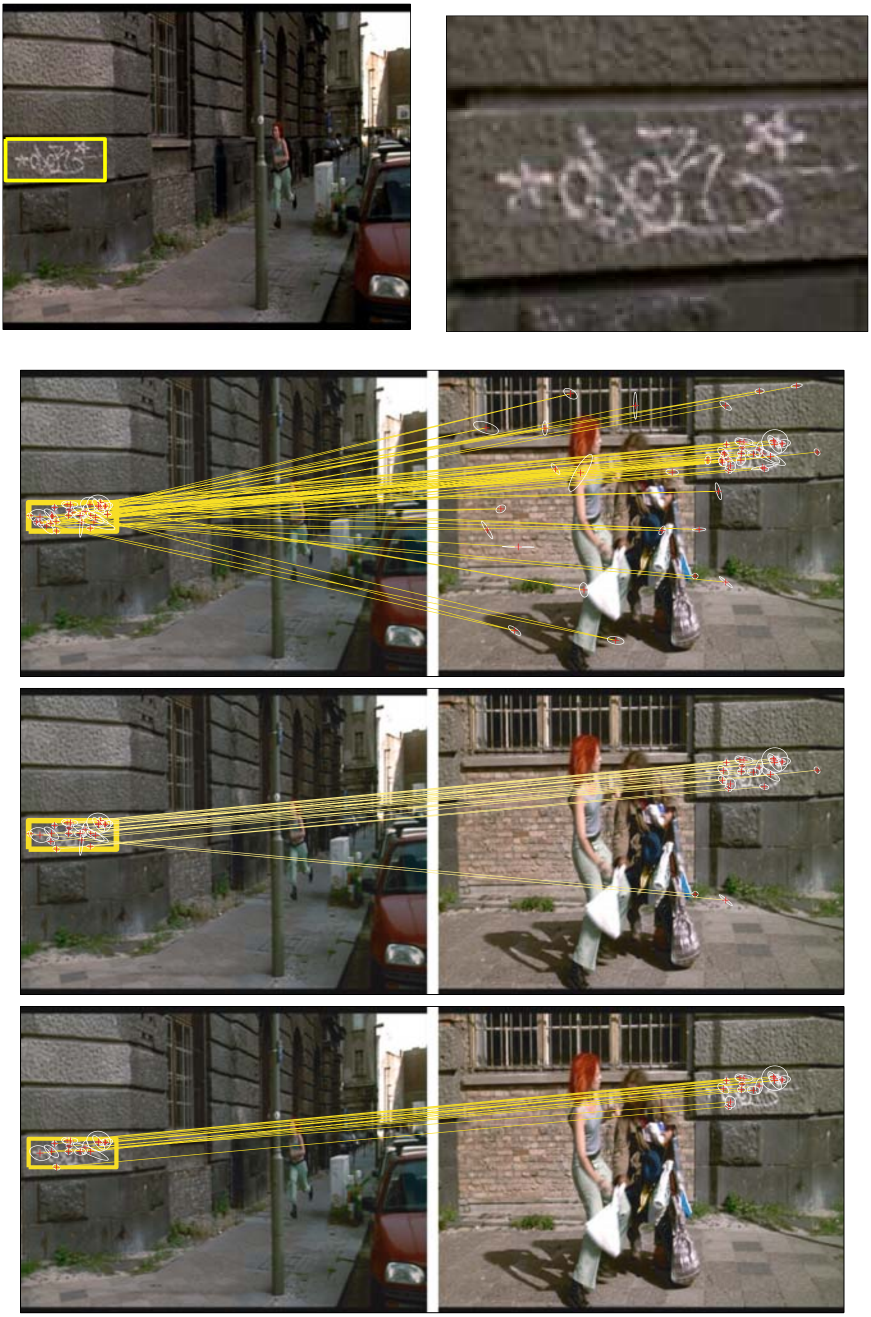


Fig. 4. Matching stages. Top row: (left) Query region and (right) its close-up. Second row: Original matches based on visual words. Third row: matches after using the stop-list. Last row: Final set of matches after filtering on spatial consistency.

1. **Pre-processing (off-line)**
   - Detect affine covariant regions in each keyframe of the video. Represent each region by a SIFT descriptor (section 2).
   - Track the regions through the video and reject unstable regions (section 2).
   - Build a visual dictionary by clustering stable regions from a subset of the video. Assign each region descriptor in each keyframe to the nearest cluster centre (section 3).
   - Remove stop-listed visual words (section 4.1).
   - Compute tf-idf weighted document frequency vectors (section 4).
   - Build the inverted file indexing structure (section 5).

2. **At run-time (given a user selected query region)**
   - Determine the set of visual words within the query region.
   - Retrieve keyframes based on visual word frequencies (section 4).
   - Re-rank the top $N_s(= 500)$ retrieved keyframes using the spatial consistency check (section 4.2).

Fig. 5. The Video Google object retrieval algorithm

Other measures which take account of the affine mapping between images may be required in some situations, but this involves a greater computational expense.

5 Object Retrieval Using Visual Words

We first describe the off-line processing. A feature length film typically has 100K-150K frames. To reduce complexity one keyframe is used per second of the video. Descriptors are computed for stable regions in each keyframe (stability is determined by tracking as described in section 2). The descriptors are vector quantized using the centres clustered from the training set, i.e. each descriptor is assigned to a visual word. The visual words over all frames are assembled into an inverted file structure where for each word all occurrences and the position of the word in all frames are stored.

At run-time a user selects a query region. This specifies a set of visual words and their spatial layout. Retrieval then proceeds in two steps: first frames are retrieved based on their tf-idf weighted frequency vectors (the bag of words model), then they are re-ranked using spatial consistency voting. The frequency based ranking is implemented using the Matlab sparse matrix engine. The spatial consistency re-ranking is implemented using the inverted file structure. The entire process is summarized in figure 5.

It is worth examining the time complexity of this retrieval architecture and comparing it to that of a method that does not vector quantize the descriptors.

The huge advantage of the quantization is that all descriptors assigned to the same visual word are considered matched. This means that the burden on the run-time matching is substantially reduced as descriptors have effectively been pre-matched off-line.

In detail, suppose there are N frames, a vocabulary of V visual words, and each frame contains R regions and M distinct visual words. $M < R$ if some regions are represented by the same visual word. Each frame is equivalent to a vector in $\mathbb{R}^V$ with M non-zero entries. Typical values are $N = 10,000$, $V = 20,000$ and $M = 500$. At run time the task is to compute the score of (1) between the query frame vector $\boldsymbol{v}_q$ and each frame vector $\boldsymbol{v}_d$ in the database (another situation might be to only return the n closest frame vectors). The current implementation exploits sparse coding for efficient search as follows. The vectors are pre-normalized (so that the denominator of (1) is unity), and the computation reduces to one dot product for each of the N frames. Moreover, only the $m \leq M$ entries which are non-zero in both $\boldsymbol{v}_q$ and $\boldsymbol{v}_d$ need to be examined during each dot product computation (and typically there are far less than R regions in $\boldsymbol{v}_q$ as only a subpart of a frame specifies the object search). In the worst case if $m = M$ for all documents the time complexity is $O(MN)$.

If vector quantization is *not* used, then two architectures are possible. In the first, the query frame is matched to each frame in turn. In the second, descriptors over all frames are combined into a single search space. As SIFT is used the dimension D of the search space will be 128. In the first case the object search requires finding matches for *each* of the R descriptors of the query frame, and there are R regions in each frame, so there are R searches through R points of dimension D for N frames, a worst case cost of $O(NR^2D)$. In the second case, over all frames there are NR descriptors. Again, to search for the object requires finding matches for *each* of the R descriptors in the query image, i.e. R searches through NR points, again resulting in time complexity $O(NR^2D)$.

Consequently, even in the worst case, the vector quantizing architecture is a factor of RD times faster than not quantizing. These worst case complexity results can, of course, be improved by using efficient nearest neighbour or approximate nearest neighbour search [9].

6 Experiments

In this section we evaluate object retrieval performance over the entire movie. The object of interest is specified by the user as a sub-part of any keyframe. In part this retrieval performance assesses the expressiveness of the visual vocabulary, since invariant descriptors from the test objects (and the frames they appear in) may not have been included when clustering to form the vocabulary.

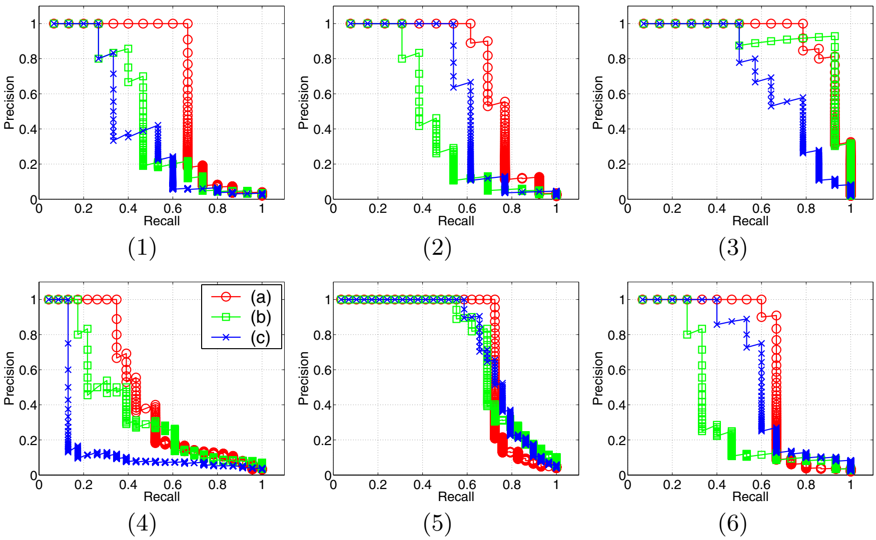
Baseline method: The performance is compared to a baseline method implementing standard frame to frame matching. The goal is to evaluate the potential loss of performance due to the descriptor quantization. The same detected regions

Object	# of keyframes	# of shots	# of query regions
1 Red Clock	138	15	31
2 Black Clock	120	13	29
3 Frames sign	92	14	123
4 Digital clock	208	23	97
5 Phil sign	153	29	26
6 Microphone	118	15	19

Fig. 6. Query frames with outlined query regions for the six test queries with manually obtained ground truth occurrences in the movie Groundhog Day. The table shows the number of ground truth occurrences (keyframes and shots) and the number of affine covariant regions lying within the query rectangle for each query.

and descriptors (after the stability check) in each keyframe are used. The detected affine covariant regions within the query area in the query keyframe are sequentially matched to all 5,640 keyframes in the movie. For each keyframe, matches are obtained based on the descriptor values using nearest neighbour matching with a threshold on the distance. Euclidean distance is used here. Keyframes are ranked by the number of matches and shots are ranked by their best scoring keyframes.

Comparison on ground truth: The performance of the proposed method is evaluated on six object queries in the movie Groundhog Day. Figure 6 shows the query frames and corresponding query regions. Ground truth occurrences were manually labelled in all the 5,640 keyframes (752 shots). Retrieval is performed on keyframes as outlined in section 4 and each shot of the video is scored by its best scoring keyframe. Performance is measured using a precision-recall plot for each query. Precision is the number of retrieved ground truth shots relative to the total number of shots retrieved. Recall is the number of retrieved ground truth shots relative to the total number of ground truth shots in the movie. Precision-recall plots are shown in figure 7. Results are summarized using Average



	Object 1	Object 2	Object 3	Object 4	Object 5	Object 6	Average
AP freq+spat (a)	0.70	0.75	0.93	0.50	0.75	0.68	**0.72**
AP freq only (b)	0.49	0.46	0.91	0.40	0.74	0.41	**0.57**
AP baseline (c)	0.44	0.62	0.72	0.20	0.76	0.62	**0.56**

Average precision (AP) for the six object queries.

Fig. 7. Precision-recall graphs (at the shot level) for the six ground truth queries on the movie Groundhog Day. Each graph shows three curves corresponding to (a) frequency ranking followed by spatial consensus (circles), (b) frequency ranking only (squares), and (c) baseline matching (stars). Note the significantly improved precision at lower recalls after spatial consensus re-ranking (a) is applied to the frequency based ranking (b). The table shows average precision (AP) for each ground truth object query for the three different methods. The last column shows mean average precision over all six queries.

Precision (AP) in the table in figure 7. Average Precision is a single valued measure computed as the area under the precision-recall graph and reflects performance over all recall levels.

It is evident that for all queries the average precision of the proposed method exceeds that of using frequency vectors alone – showing the benefits of the spatial consistency in improving the ranking. On average (across all queries) the frequency ranking method performs comparably to the baseline method. This demonstrates that using visual word matching does not result in a significant loss in performance against the standard frame to frame matching.

Figures 1, 8 and 9 show example retrieval results for three object queries for the movie 'Groundhog Day', and figure 10 shows example retrieval results for black and white film 'Casablanca'. For the 'Casablanca' retrievals, the film

Fig. 8. Object query example II: Groundhog Day. (a) Keyframe with user specified query region in yellow (phil sign), (b) close-up of the query region and (c) close-up with affine covariant regions superimposed. (d-g) (first row) keyframes from the 1st, 4th, 10th, and 19th retrieved shots with the identified region of interest shown in yellow, (second row) a close-up of the image, and (third row) a close-up of the image with matched elliptical regions superimposed. The first false positive is ranked 21st. The precision-recall graph for this query is shown in figure 7 (object 5). Querying 5,640 keyframes took 0.64 seconds.

is represented by 5,749 keyframes, and a new visual vocabulary was built as described in section 3.

Processing time: The region detection, description and visual word assignment takes about 20 seconds per frame (720×576 pixels) but can be done off-line. The average query time for the six ground truth queries on the database of 5,640 keyframes is 0.82 seconds with a Matlab implementation on a 2GHz pentium. This includes the frequency ranking and spatial consistency re-ranking. The spatial consistency re-ranking is applied only to the top $N_s = 500$ keyframes ranked by the frequency based score. This restriction results in no loss of performance (measured on the set of ground truth queries).

The query time of the baseline matching method on the same database of 5,640 keyframes is about 500 seconds. This timing includes only the nearest

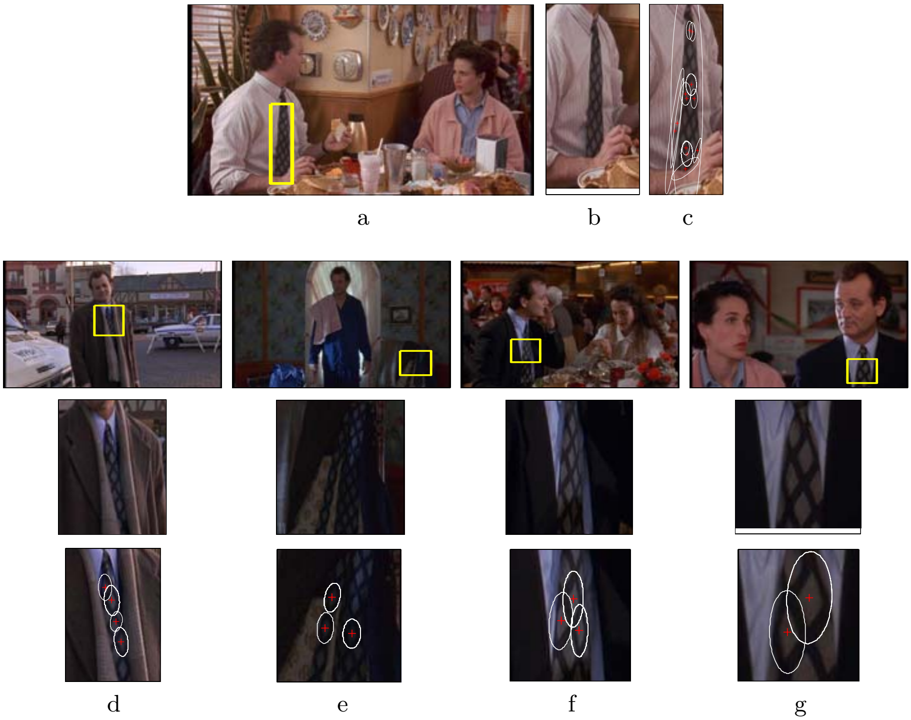
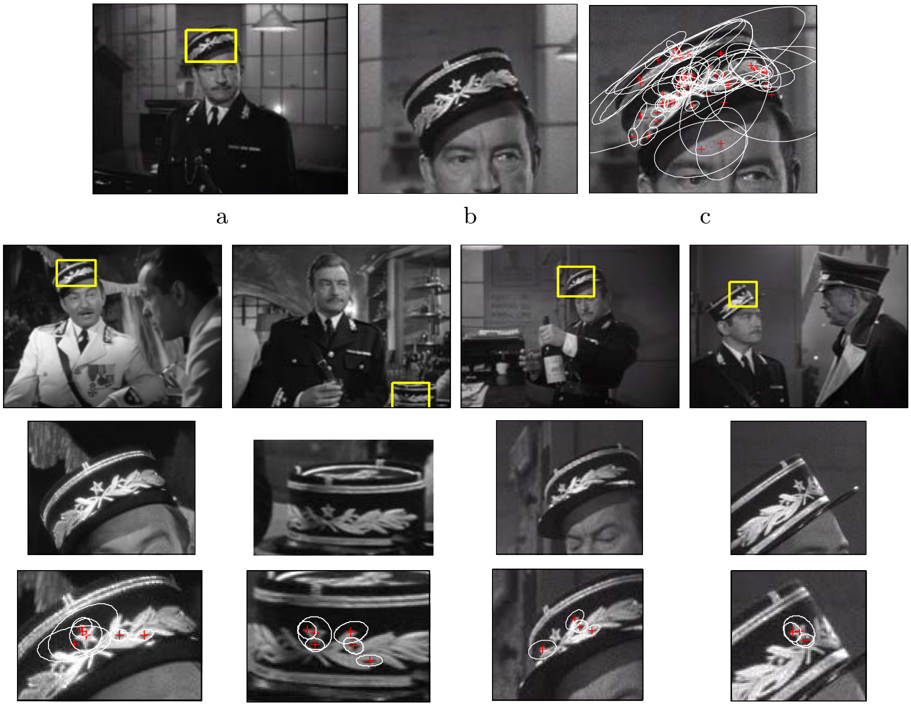


Fig. 9. Object query example III: Groundhog Day. (a) Keyframe with user specified query region in yellow (tie), (b) close-up of the query region and (c) close-up with affine covariant regions superimposed. (d-g) (first row) keyframes from the 1st, 2nd, 4th, and 19th retrieved shots with the identified region of interest shown in yellow, (second row) a close-up of the image, and (third row) a close-up of the image with matched elliptical regions superimposed. The first false positive is ranked 25th. Querying 5,640 keyframes took 0.38 seconds.

neighbour matching performed using linear search. The region detection and description is also done off-line. Note that on this set of queries our proposed method has achieved about 600-fold speed-up.

Limitations of the current method: Examples of frames from low ranked shots are shown in figure 11. Appearance changes due to extreme viewing angles, large scale changes and significant motion blur affect the process of extracting and matching affine covariant regions. The examples shown represent a significant challenge to the current object matching method.

Searching for objects from outside the movie: Figure 12 shows an example of searching for an object outside the 'closed world' of the film. The object (a Sony logo) is specified by a query image downloaded from the internet. The image was



a b c

d e f g

Fig. 10. Object query example IV: Casablanca. (a) Keyframe with user specified query region in yellow (hat), (b) close-up of the query region and (c) close-up with affine covariant regions superimposed. (d-g) (first row) keyframes from the 4th, 5th, 11th, and 19th retrieved shots with the identified region of interest shown in yellow, (second row) a close-up of the image, and (third row) a close-up of the image with matched elliptical regions superimposed. The first false positive is ranked 25th. Querying 5,749 keyframes took 0.83 seconds.

(1,2) (4)

Fig. 11. Examples of missed (low ranked) detections for objects 1,2 and 4. In the left image the two clocks (object 1 and 2) are imaged from an extreme viewing angle and are barely visible – the red clock (object 2) is partially out of view. In the right image the digital clock (object 4) is imaged at a small scale and significantly motion blurred. Examples shown here were also low ranked by the baseline method.

Fig. 12. Searching for a Sony logo. First column: (top) Sony Discman image (640×422 pixels) with the query region outlined in yellow and (bottom) close-up with detected elliptical regions superimposed. Second and third column: (top) retrieved frames from two different shots of 'Groundhog Day' with detected Sony logo outlined in yellow and (bottom) close-up of the image. The retrieved shots were ranked 1 and 4.

preprocessed as outlined in section 2. Searching for images from other sources opens up the possibility for product placement queries, or searching movies for company logos, or particular types of vehicles or buildings.

7 Conclusions

We have demonstrated a scalable object retrieval architecture by the use of a visual vocabulary based on vector quantized viewpoint invariant descriptors. The vector quantization does not appear to introduce a significant loss in retrieval performance (precision or recall) compared to nearest neighbour matching.

The method in this chapter allows retrieval for a particular visual aspect of an object. However, temporal information within a shot may be used to group visual aspects, and enable object level retrieval [17].

A live demonstration of the 'Video Google' system on a publicly available movie (Dressed to Kill) is available on-line at [1].

Acknowledgements

Funding for this work was provided by EC Project Vibes and EC Project CogViSys.

References

1. http://www.robots.ox.ac.uk/∼vgg/research/vgoogle/.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, ISBN: 020139829, 1999.

3. A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.
4. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *7th Int. WWW Conference*, 1998.
5. C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.
6. T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure. In *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, LNCS 800, pages 389–400, May 1994.
7. D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, September 1999.
8. D. Lowe. Local feature view clustering for 3D object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, pages 682–688. Springer, December 2001.
9. D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
10. J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 384–393, 2002.
11. K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*. Springer-Verlag, 2002.
12. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
13. S. Obdrzalek and J. Matas. Object recognition using local affine frames on distinguished regions. In *Proceedings of the British Machine Vision Conference*, pages 113–122, 2002.
14. F. Schaffalitzky and A. Zisserman. Automated scene matching in movies. In *Proceedings of the Challenge of Image and Video Retrieval, London*, LNCS 2383, pages 186–197. Springer-Verlag, 2002.
15. F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 1, pages 414–431. Springer-Verlag, 2002.
16. C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, May 1997.
17. J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, May 2004.
18. J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, October 2003.
19. D.M. Squire, W. Müller, H. Müller, and T. Pun. Content-based query of image databases: inspirations from text retrieval. *Pattern Recognition Letters*, 21:1193–1198, 2000.

20. D. Tell and S. Carlsson. Combining appearance and topology for wide baseline matching. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, LNCS 2350, pages 68–81. Springer-Verlag, May 2002.
21. T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proceedings of the 11th British Machine Vision Conference, Bristol*, pages 412–425, 2000.
22. I. H. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, ISBN:1558605703, 1999.