Stable hyper-pooling and query expansion for event detection

Matthijs Douze
INRIA Grenoble

Jérôme Revaud
INRIA Grenoble

Cordelia Schmid
INRIA Grenoble

Hervé Jégou
INRIA Rennes

Abstract

This paper makes two complementary contributions to event retrieval in large collections of videos. First, we propose hyper-pooling strategies that encode the frame descriptors into a representation of the video sequence in a stable manner. Our best choices compare favorably with regular pooling techniques based on k-means quantization. Second, we introduce a technique to improve the ranking. It can be interpreted either as a query expansion method or as a similarity adaptation based on the local context of the query video descriptor. Experiments on public benchmarks show that our methods are complementary and improve event retrieval results, without sacrificing efficiency.

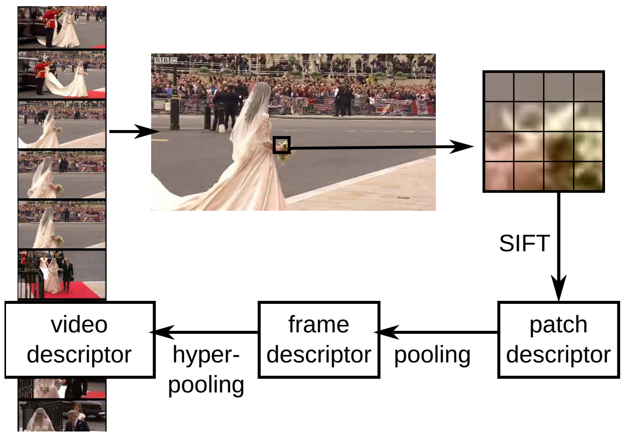


Figure 1. Our first contribution: local descriptors are extracted and pooled to produce frame descriptors. These are *hyper-pooled* to form a single vector video representation. We address the stability of the pooling, as techniques used to encode local patch descriptors are not stable enough.

1. Introduction

Event retrieval in large collections of videos is an emerging problem. Given a query video, the goal is to retrieve all the videos associated with the same event. Solutions to this problem address needs of individual and professional users, since the video content is not necessarily annotated with relevant tags, and pure text retrieval is not precise enough.

This paper considers the unsupervised setup. In the supervised case, considered, for example, in the multimedia event detection (MED) task of the TRECVID campaign [13], the system additionally receives a set of annotated videos representing the event. In the unsupervised case, only the query video is provided.

Most state-of-the-art systems [5, 12, 18] first extract individual frame descriptors, which are then averaged to produce the video representation. In the context of image representations, pooling strategies are intensively used to reduce the loss of information induced by simply summing local descriptors [3, 4]. This paper introduces a *hyper-pooling* approach for videos, that encodes frame descriptors into a single vector based on a second layer clustering, as illustrated in Figure 1. This stage is similar in spirit to the hyper-features [1], that produce mid-level representations of images.

Most of the research effort has been devoted on the pooling at the image level. Recently, Cao *et al.* [5] proposed scene-aligned pooling that fuses the frame descriptors per scene before computing a kernel based on these scenes. They soft-assign each frame to 16 scene categories (using GIST descriptors) and then sum the frame descriptors using the weights of this soft-assignment. Similarly, a recent method [18] based on circulant matching can be interpreted as a hyper-pooling technique in the frequency domain, where the average frame is complemented with frequency vectors, thereby incorporating temporal information. The method was shown effective for copy detection, but for event retrieval it is only slightly better than simple average pooling, which is also used in several of the top-ranking systems of TRECVID MED 2012 [12].

Hyper-pooling is more challenging than pooling of local descriptors: as noticed in [5], vector quantization of frame descriptors is prone to noise and, therefore, pooling techniques developed for image representation are not appropriate. The first contribution of our paper is to introduce a video hyper-pooling method which relies on a stable and reliable vector quantization technique adapted to high-dimensional frame descriptors. It is particularly adapted to videos comprising many shots, where average pooling tends to become less effective. Our approach dispatches the vectors into different cells and aggregates them in these cells.

Our second contribution is a query expansion technique for event retrieval in videos when no training data is provided (unsupervised case). In contrast with many approaches in image retrieval, it does not rely on any spatial verification. Yet, it turns out to be also effective for image retrieval, see Section 5.2.

The paper is organized as follows. Section 2 introduces the datasets and the frame representation that we use in this paper. Section 3 analyzes the properties of our frame descriptors to design better pooling schemes. Section 4 introduces our query expansion technique. Both contributions are evaluated in the experimental section 5.

2. Background

This section presents our baseline for event retrieval, the method introduced recently by Revaud *et al.* [18]. We also describe the datasets used for our experimental evaluation.

2.1. Datasets

EVVE[1]. We mainly use the EVVE dataset to evaluate our method. EVVE is an event retrieval benchmark: given a single video of an event, it aims at retrieving videos related to the event from a dataset. Depending on the event, it requires to recognize same objects or scenes, specific video footage, and/or characteristic motion of the event.

Oxford5k Buildings[2]. Although the main focus of this work is on video analysis, we also investigate whether our findings extend to images. To do so, we use the Oxford5k Building dataset, which is a standard benchmark for image retrieval of the same object or building [16].

Evaluation. For both datasets we have used public descriptors and the evaluation code provided by the authors on their websites. This makes a direct comparison with the state of the art possible.

Both benchmarks use the same evaluation protocol: videos (or images) from a predefined query set are submitted to the retrieval system, which returns a list of results from the dataset. Results are then flagged as true or false positives. The ranked results are evaluated in terms of average precision [16]. Following common practice, the mean average precision (mAP) over queries is used as a synthetic performance measure.

2.2. Image and video descriptors

VLAD image descriptor. On the Oxford5k dataset we extract Hessian-Affine interest points and describe them with SIFT descriptors. The SIFT vectors are reduced to 64 D with a PCA, as in [9]. This assembles most of the descriptor variance in the first components. The baseline pooling and aggregation method for these PCA-reduced SIFTs is the VLAD descriptor [8].

MVLAD frame descriptor. On the EVVE dataset, image descriptors are extracted for every video frame. They are based on dense SIFT descriptors, that are aggregated into multiple VLAD descriptors with distinct vocabularies [7].

To produce the MVLAD descriptor, the concatenated VLADs are reduced with a PCA, followed by a whitening stage that divides each component with its corresponding standard deviation. Therefore, the first components of the MVLAD correspond to the most significant orientations in the descriptor space, but all components have equal variance after re-normalization.

Mean MVLAD video descriptor (MMV). Mean MVLAD is a fixed-size video descriptor averaging the set of MVLADs describing the frames of the video. The cosine similarity is used to compare the query with all the database video descriptors.

Although this aggregation method is simple and produces a short descriptor (typically 512 dimensions), for event recognition it provides competitive results compared to a more sophisticated method based on Fourier domain comparison [18].

3. Stable hyper-pooling

The objective of this section is to provide a strategy to encode a sequence of frame descriptors of a video clip, based on a pooling strategy. This pooling strategy consists of two stages: hashing and encoding.

Hashing. Let us consider a set $\mathcal{X} = (\mathbf{x}_t)_t,\ \mathbf{x}_t \in \mathbb{R}^d$ of d-dimensional vectors, as for example the MVLAD descriptors of Section 2. A hashing function

$$q : \mathbb{R}^d \to [1 \dots k],\ \mathbf{x} \mapsto q(\mathbf{x}) \tag{1}$$

maps a vector $\mathbf{x}$ to an index $q(\mathbf{x}) \in [1 \dots k]$ (or, equivalently, to the cell containing all points in $\mathbb{R}^d$ assigned to that index). The k-means quantizer is the most standard choice. It is fully defined by a set of k centroids $\{\mathbf{c}_1, \dots, \mathbf{c}_k\}$, learned on a training set, and uses a nearest neighbor assignment rule to find the index.

Encoding. We use the VLAD pooling technique [8]. It encodes a set of local descriptors into a single vector representation as follows. First, each descriptor is assigned to a cell by k-means hashing, as in Equation 1. Its centroid is subtracted to produce a residual vector. All residuals associated with the same index are added to produce a d-dimensional vector

$$\mathbf{x}_t^j(\mathcal{X}) = \sum_{\mathbf{x}_t \in \mathcal{X} : q(\mathbf{x}_t) = j} \mathbf{x}_t - \mathbf{c}_j. \tag{2}$$

[1]http://pascal.inrialpes.fr/data/evve/

[2]http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/

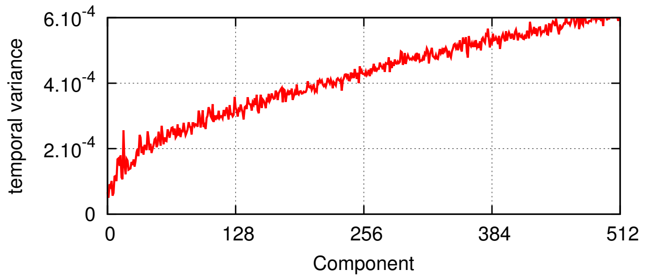


Figure 2. Temporal variance $E_{X_t}\left[(X_{t+1}-X_t)^2\right]$ per dimension of the MVLAD descriptor. The components are ordered by decreasing PCA eigenvalues. Note that after whitening, $E_X\left[(X_t)^2\right]$ is constant for all components (and $\bar{X}_t = 0$). The curve shows that the temporal variance is lower for larger eigenvalues, as predicted by an isotropic noise model.

The VLAD vector is obtained by concatenating all these vectors

$$V(\mathcal{X}) = \left[{\mathbf{x}_t^1}^\top \dots {\mathbf{x}_t^k}^\top\right]^\top, \tag{3}$$

and is ℓ_2-normalized.

VLAD uses the k-means centroids both for pooling and encoding the descriptors. We consider a more general formulation, in which we separate the hashing function $q(.)$ from the encoding of the descriptors falling in each of the cells. In other terms, the assignment rule is now considered independently from the calculation of $\mathbf{x}_t^j$.

In the following, we discuss the specificity of our frame descriptors. Then, we evaluate different hashing strategies, including the standard k-means. This leads to a technique relying on stable components. We, finally, discuss some relationships with existing works in different contexts, and briefly evaluate our method in the context of image indexing.

3.1. Properties of the MVLAD frame descriptors

The frame descriptors used as input of our pooling scheme have specific properties. Their mean is the null vector before ℓ_2-normalization. This property is still almost satisfied after normalization.

MVLADs have high *intrinsic* dimensionality because they are obtained by PCA from large descriptors. As a result, two vectors assigned to the same cell are almost orthogonal, as shown by comparing the average distance of two vectors assigned to the same cell: for $k = 32$, it is typically 1.389, *i.e.*, close to the value $\sqrt{2} \approx 1.414$ obtained with orthogonality. For the same reason, most vectors are close to the boundaries, which affects the stability of the assignment.

Another consequence is that the expectation of the vectors assigned to a given cell is close to the null vector. Instead of summing the residual, it is therefore reasonable to

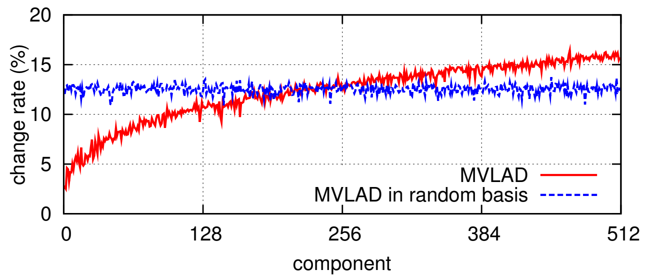


Figure 3. Empirical probability that a MVLAD component changes sign between two successive frames. For reference, we also provide the change rate of the sign in an arbitrary basis instead of the PCA basis.

simplify the computation of $\mathbf{x}_t^j$ in Eqn. 2:

$$\mathbf{x}_t^j(\mathcal{X}) = \sum_{\mathbf{x}_t \in \mathcal{X} : q(\mathbf{x}_t)=j} \mathbf{x}_t. \tag{4}$$

This makes it possible to use quantizers that are not based on centroids. Note that for $k = 1$, Eqn. 4 is equivalent to computing the MMV vector.

Impact of whitening. The MVLAD components are whitened (divided by their standard deviation, see Section 2.2) before being ℓ_2-normalized. This has two consequences. First, all components have an equal contribution on average when using the cosine as a similarity measure. Second, assuming that, before whitening, the frame descriptor is altered by a small and non time-consistent isotropic noise, the components of the MVLAD associated with the largest eigenvalues are more stable over time than those associated with small eigenvalues. This is empirically confirmed by Figure 2, which reports, per component, the variance of the difference vector $\mathbf{X_{t+1}} - \mathbf{X_t}$ for a video clip.

3.2. Evaluation of frame hashing strategies

The previous discussion and variance analysis suggests that the pooling strategy should exploit the most stable (first) components of the MVLAD descriptor.

Hashing: design criteria. Unlike in bag-of-words, the objective of the hash function is not to find the best approximation of the vector, as the encoding stage will be considered independently. Our motivation is to optimize two contradictory criteria:

1. The stability of the assignment;

2. The richness of the representation.

The stability is maximized by a trivial solution, *i.e.*, by simply assigning all vectors to the same cell, as in the MMV baseline described in Section 2. However, this solution is not likely to capture the different aspects of a video sequence formed of multiple shots. The second criterion is difficult to evaluate independently from the stability. It depends on the subsequent coding technique and on the task. As a simple criterion, we measure the empirical entropy of the distribution of indexes (higher is better). Although not directly related to the richness of the representation, entropy reflects how the frame vectors are spread over the different clusters. It is maximized when the vectors are evenly dispatched over the different indexes.

The stability is measured by the rate of successive frames assigned to the same $q(x)$, see Figure 3. This reflects the temporal stability of the assignment, as most successive frames belong to the same shot and are expected to be hashed similarly.

Hashing functions. Let $\mathbf{x} = [f_1, \ldots, f_d]^\top$ be a frame descriptor. We consider the following choices for the hashing function $q(.)$.

1– k-means serves as a baseline.

2– Partial k-means (PKM) is the same as k-means except that it uses only the m first components of MVLAD, *i.e.*, those corresponding to the largest eigenvalues. The assignment relies on the subvector $[f_1, \ldots, f_m]$ instead of the full vector, which aims at producing a more consistent assignment (see previous discussion).

3– Sign of stable components (SSC). We consider the hashing function

$$q : \mathbb{R} \to \{0,1\}^m \equiv [1 \ldots k] \quad (5)$$

$$\mathbf{f} \mapsto \left[\text{sign}(f_1), \ldots, \text{sign}(f_m)\right]^\top, \quad (6)$$

which produces a binary code, or equivalently, an integer index value in $[1 \ldots 2^m]$. For instance, by taking the 5 first components, we obtain $k = 2^5$ different cells. This is similar to a code produced by locality sensitive hashing, except that we rely on the first components of the descriptors, as in [15]. This is preferable for the same reason as PKM, since the first components are more robust to a small variation of the input frame.

4– KD-tree. The previous choices are motivated by the stability criterion. Instead, if we want to optimize the entropy of the random variable associated with the index values, we can use a kd-tree computed on the first m components, and learned on a distinct set. On the learning set, the entropy is equal to $\log_2 k = m$ by construction.

Comparison of hashing schemes. Table 1 compares the four strategies with respect to the stability in time, measured by the probability that the hashing key changes between two frames, and the diversity, measured by the entropy. The k-means, used in pooling schemes such as bag-of-words or VLAD, is less stable over time and has also a lower diversity than its partial counterpart. As expected, the kd-tree has

Table 1. Evaluation of stability and diversity of different hashing techniques on a sample set of video clips. $k = 32$ for all functions. Random indexes are included for reference.

hashing	# components	change rate (%)	entropy
k-means	512 (all)	12.41	4.320
PKM	5 first	11.74	4.671
SSC	5 first	12.85	4.683
kd-tree	5 first	15.24	4.839
random	0	96.88	5.000

the highest entropy, as this criterion is maximized on the learning. However, it is the most unstable approach.

The two approaches employing stable components, *i.e.*, PKM and SSC, both offer an interesting trade-off between stability and entropy. Partial k-means is the best, but SSC does not require any learning stage if frame descriptors are already reduced by PCA. The stability is illustrated on a video excerpt in Figure 4: PKM and SSC are visually more stable than the kd-tree and k-means quantizers.

3.3. Encoding

After hashing the frames descriptors, the descriptors are aggregated per quantization cell. The construction of the $d \times k$-dimensional video descriptor proceeds as follows.

1. We use the simple sum proposed in Equation 4 to aggregate the frames descriptors within a cell.
2. We adopt the component-wise power-law normalization [14] proposed for the Fisher kernel and also employed with VLAD [9]. This cancels an effect resulting from the self-similarity in videos: long static shots dominate the similarity.
3. The resulting vector is ℓ_2-normalized and vectors are compared with the inner product, in order to compare the video descriptors with cosine similarity.

3.4. Discussion

The pooling technique based on stable components is not specific to video, as it plays the same role as the quantizer used in VLAD to dispatch the set of descriptors into several cells. SSC is similar in spirit to the fixed structured rectangular lattice proposed in a classification context [19], where bag-of-“fixed”-words histograms were produced.

We have measured the performance on the Oxford5k [16] object retrieval benchmark, *i.e.*, by using PKM and SSC as a pooling scheme for local descriptors transformed by PCA. The regular VLAD pooling techniques gives mAP=40.0% with 64 centroids, while PKM and SSC give 33.3% and 29.2%, respectively.

This low performance mirrors the observation made by Philbin *et al.* [17], who evaluated lattice-based pooling for retrieval. It is not surprising, as SIFT descriptors have a relatively low intrinsic dimensionality, and their assignment is

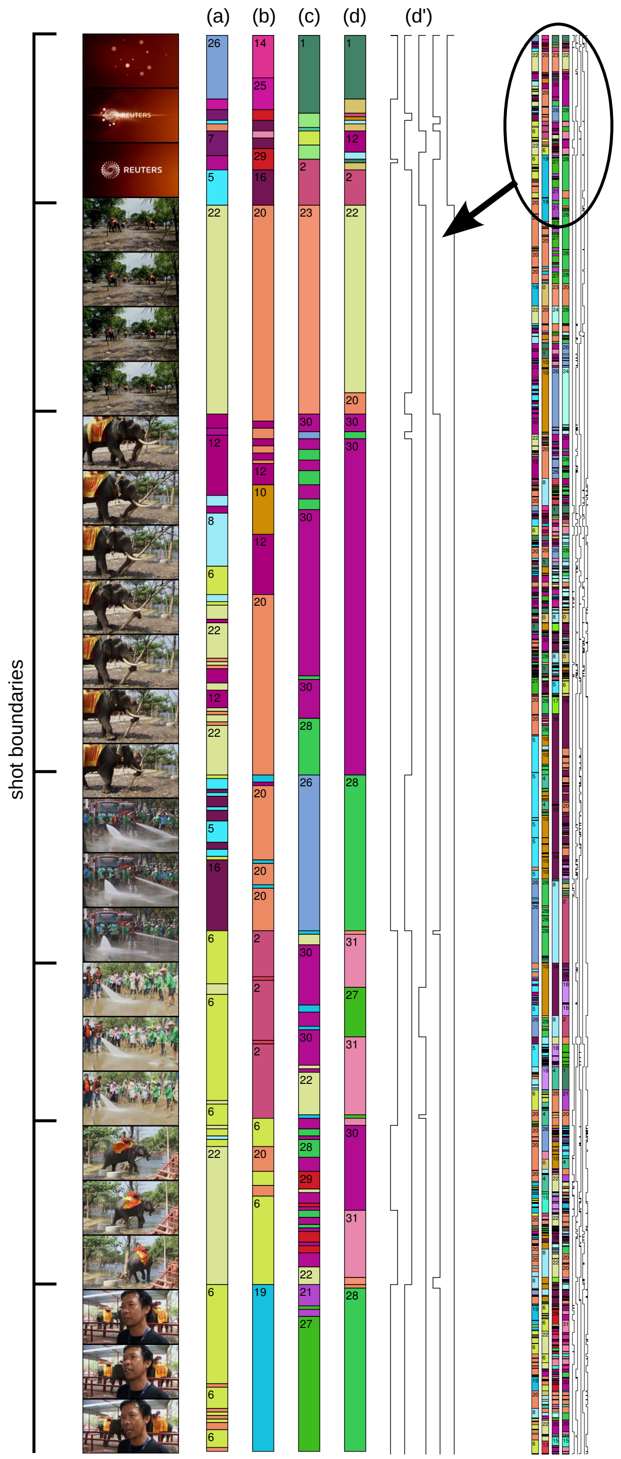


Figure 4. Temporal stability of different space partitioning approaches with $k = 32$ cells, color = hash value. (a) k-means, (b) PKM, restricted to the first 5 components, (c) kd-tree based hashing, on 5 components, (d,d') SSC hashing: (d) = hash index, (d') = the assignment to the individual bits. Note that most shot boundaries trigger a modification of the hash values.

stable enough. For pooling local descriptors, it is therefore more important to produce the best approximation of the descriptors.

Therefore, we insist that our SSC method is better suited to high-dimensional vectors, like those considered for video hyper-pooling. By design, it relies on stable components, which are in limited number. It is better to avoid building large vocabularies, because the required additional bits become less stable, as shown in Figure 3.

4. Query expansion: similarity in context

Query expansion (QE) refers to re-ranking procedures that operate in two stages. The first stage is standard: The nearest neighbors of the query vector are retrieved. In the second stage, a few reliable neighbors are fused with the query to produce an *expanded* query that is submitted in turn in a second retrieval stage. QE is effective for datasets comprising many positives per query, which guarantees that the expanded query is indeed better than the initial one. Both EVVE and Oxford5k satisfy this condition.

Existing visual query expansion approaches [2, 6, 11] employ a geometric matching procedure to select the relevant images used in the expansion. For very large sets of videos, it is, however, not reasonable[3] to store individually all descriptors along with their spatial and temporal positions in the different frames. Therefore, we consider only methods that rely on global descriptors, including those obtained by aggregating local descriptors as in [5, 18].

Average query expansion (AQE). We start from the AQE method introduced in [6] for bag-of-words (BoW) image descriptors. It is similar to its counterpart in text information retrieval. It averages the BoW descriptor describing the query with those of the shortlist of nearest neighbors retrieved by the first stage:

$$q_{\text{aqe}} = \frac{q + \sum_{b \in N_1} b}{1 + |N_1|}, \tag{7}$$

where q is the query vector, N_1 the neighborhood of q in the database. A new nearest-neighbor search in the dataset is performed using q_{aqe} as a query vector. Note that AQE shares some similarities with the NL-means denoising techniques, in which a patch is de-noised as a weighted average of similar patches in the image.

Difference of Neighborhood (DoN). We extend AQE by subtracting the average of a larger neighborhood of the query, $N_2 \supset N_1$. This amounts to computing

$$q_{\text{don}} = \frac{q + \sum_{b \in N_1} b}{1 + |N_1|} - \frac{\sum_{b \in N_2} b}{|N_2|}. \tag{8}$$

[3]The dataset of 100,000 video clips used in the evaluation section is represented by 300 million frames, or 2200 billion local descriptors!

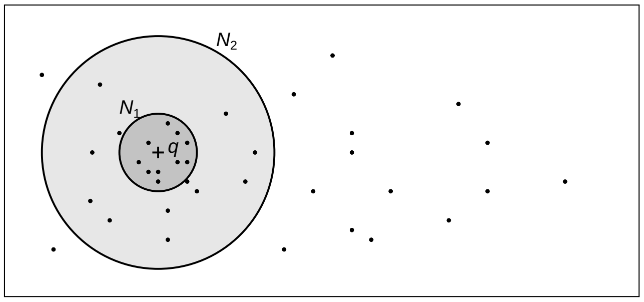


Figure 5. Query vector q is averaged with the points in its neighborhood N_1, and the average of the larger neighborhood N_2 is subtracted from it (Equation 8).

The rationale for this is similar to the difference-of-Gaussians applied in image processing:

- The average over a small neighborhood N_1, assuming that it contains relevant answers, reduces the noise in the query descriptor, and includes new information on nearby descriptors (as in AQE);
- Subtracting a larger neighborhood increases the contrast with descriptors that are likely to be negative results.

The neighborhood is defined either by finding a predefined number of nearest neighbors for the query, or by selecting the points that are closer than a threshold.

This method is related to the Rocchio algorithm, as used for relevance feedback [10]. In this formulation, same-class examples are taken from a small neighborhood N_1, and examples from different classes are assumed to be in N_2, a neighborhood large enough to guarantee that it contains a majority of negatives.

In contrast with our method, Rocchio's algorithm takes the whole set of documents, excluding the same-class examples, as negatives. We find this to be suboptimal (see Section 5). Also, we do not weight the positive and negative terms, to avoid another parameter to tune.

5. Experiments

We evaluate three aspects of our method: (i) the performance of our hyper-pooling; (ii) the improvement due to our query expansion method; and (iii) the behavior of the method when videos are merged with clutter.

5.1. Hyper-pooling

Table 2 compares the different quantizers mentioned in Section 3 in a retrieval setup, without considering query expansion at this stage.

Dimension of MMV. The MVLAD and MMV descriptors we use here are relatively low dimensional (512 D). However, increasing the output dimension of the PCA to 2048 D does not improve the performance.

Table 2. Performance of hyper-pooling on EVVE. The figure m is the number of dimensions of the frame descriptor used by the quantizer. MA/k = 4/32 means 32 quantization cells and a multiple assignment of 4, dim is the video descriptor's dimension.

Quantizer	m	MA/k	dim	mAP
MMV [18] baseline			512	33.3
MMV	-	-	2048	33.0
Fisher Vector	-	-	16384	28.6
kmeans	512	1/32	16384	34.0
kd-tree	5	1/32	16384	33.7
random	0	1/32	16384	32.7
PKM	5	1/32	16384	34.6
SSC	5	1/32	16384	34.3
kmeans	512	4/32	16384	34.0
random	0	4/32	16384	31.6
PKM	5	4/32	16384	**36.2**
SSC	5	4/32	16384	**36.3**

We also experimented with Fisher vectors [9] as frame descriptors. With a mixture of 256 Gaussians, we obtain descriptors of 16384 D, but their performance is below that of 512 D MVLAD.

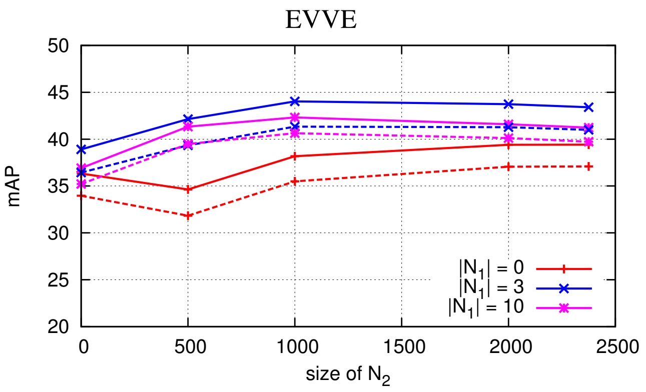
Overall, this shows that the performance of averaged frame descriptors based on SIFT saturates, irrespective of the descriptor size.

Quantizers. On EVVE, quantizers that use fewer dimensions tend to perform slightly better. Increasing the number of centroids does not improve the performance significantly. The random quantizer returns meaningful results, probably due to the temporal redundancy of the frames. This also suggests that margin for improvement due to hyper-pooling is small for this task.

Multiple assignment. We compute several quantization indices for each vector and accumulate several entries in the resulting VLAD vector[4]. This improves the retrieval performance significantly, but only for the stable pooling methods. Otherwise, it introduces clutter that reduces the performance, as shown by the decreasing results of random pooling. Our PKM and SSC hyper-pooling methods improves the MMV mAP by **+2.9** and **+3.0** points.

In what follows and unless specified otherwise, we use SSC with 32 centroids and multiple assignment to 4 cells. With this setting, hyper-pooling outperforms (mAP=36.3% with SSC) the CTE method ([18], mAP=35.2%), a more

[4]To perform multiple assignment with SSC, we start from the quantization cell $q(f) = c \in \{0,1\}^m$ (Eqn. 6). We then define the cost of flipping some of the bits of c to $b \in \{0,1\}^m$ as $\sum_{i=1..k/c_i \neq b_i} |f_i|$. The multiple assignment cells are the ones with the lowest cost.



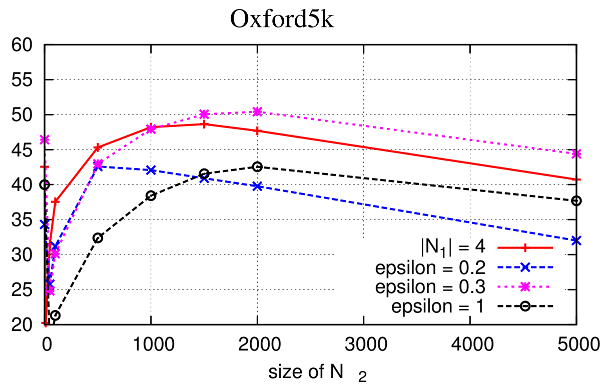


Figure 6. Parameters of the query expansion method. For EVVE, full lines: SSC quantizer, dashes: kmeans quantizer.

complex method exploiting temporal information.

5.2. Query expansion

Parameters. We, first, evaluate the impact of N_1 and N_2 on the EVVE dataset, see Figure 6. We set $|N_1| = 3$ and $|N_2| = 1000$ (Equation 8), which consistently gives good results.

QE on EVVE. Table 3 compare the results with QE on the EVVE and EVVE+100k datasets [18], using their frame descriptors. QE techniques are expected to improve the results, as the dataset contains many relevant results per query.

We compare DoN with AQE (see Section 4) and Discriminative Query Expansion (DQE). In DQE [2], a discriminative SVM classifier is learned by using the query and the 3 first retrieval results as positives and the 200 last ones as negatives (we set $C = 1$). These parameters are from the original paper and we found them experimentally to be optimal on EVVE. Then the classifier scores are used to re-rank the results.

AQE combined with our SSC method results in an improvement of +2.5 %, and DQE is better by +4.1 %. Our DoN query expansion approach outperforms these two approaches, as it improves by +7.7 % over SSC, *i.e.*, by **+10.7 %** over the MMV baseline. These findings are confirmed on EVVE+100k, where our method improves by more than +7 % over the best results [18] reported on this large-scale dataset.

Table 3. Performance with query expansion. The AQE and DoN query expansion techniques are defined in Equations 7 and 8.

dataset	MMV	Hyper-pooling			
		no QE	AQE	DQE	DoN
EVVE,	33.3	36.3	38.9	40.4	**44.0**
EVVE + 100k	22.0	26.5	30.1	30.2	**33.1**
Oxford5k	N/A	40.0	46.4		**50.4**

When applied to the MMV descriptor, the DoN query expansion gives a mAP of 40.0 on EVVE, see Table 4. Thus, the DoN increases the performance gap between MMV and SSC. This is probably due to that fact that QE methods increase the risk of false positives, requiring more discriminant input descriptors, *i.e.* SSC rather than MMV.

Table 4. SSC hyper-pooling and MMV in combination with DoN query expansion on EVVE.

	MMV	SSC
no QE	33.3	36.3
DoN	40.0	44.0

QE on Oxford5k. We also evaluate our query expansion method for image retrieval on the Oxford5k dataset, using the classical VLAD descriptors ($k = 64$). In order to reflect the dataset's properties, we adjust the parameters by setting $N_1 = \{b \in B \mid \|b - q\| < \varepsilon\}$ with $\varepsilon = 0.3$ and $|N_2| = 2000$ (this is the best setting in Figure 6). The baseline result of 40.0 % is consistent with the original VLAD paper [9]. AQE improves by +6 % points, while our DoN improves by +10.4%. The N_2 subtraction without AQE obtains mAP=46.2 % (similar to AQE alone), which shows the merit of this term.

5.3. Resilience to clutter

We evaluate how hyper-pooling performs on videos when the meaningful data is merged with a lot of "clutter" footage. A variable amount of random video footage from Youtube is added at the beginning and end of EVVE queries. We, then, use the transformed queries. We report results on MMV, SSC with standard settings, and another version named SSC 512 D, in which we keep only 128 dimensions of the MVLAD frame descriptors and use 4 centroids. This version makes it possible to compare with MMV at the same descriptor dimensionality (512).

Retrieval on long videos. Figure 7 shows how the additional clutter impacts the retrieval performance. The hyper-pooled features always perform better than MMV and the

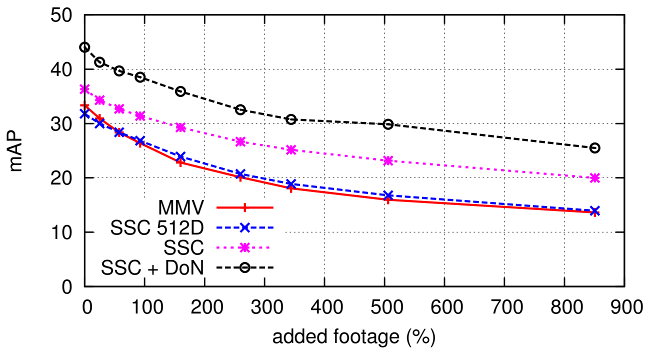


Figure 7. Performance of the different methods when EVVE query videos are embedded in random Youtube data. The length of added data is indicated as a percentage of time over the original query videos.

gap increases when the query videos become more cluttered. For a fixed descriptor size (512 dimensions), hyper-pooled features perform worse than MMV on small queries, but are slightly more robust with the additional clutter.

6. Conclusion

This paper proposes hyper-pooling strategies to produce a video descriptor from its frame descriptors. They are stable enough to deal with vectors of high dimensionality such as the recent MVLAD image descriptors. In particular, our SSC technique, albeit simple, improves the performance of event retrieval. In addition, we have introduced a query expansion method, which does not rely on geometrical verification. It is, therefore, both effective and efficient. These complementary methods establish a new baseline for unsupervised event retrieval in large collections of videos.

Acknowledgments. This work was supported by Quaero (funded by OSEO, French State agency for innovation), the European integrated project AXES, the MSR/INRIA joint project and the ERC advanced grant ALLEGRO.

References

[1] A. Agarwal and B. Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *ECCV*, 2006.

[2] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[3] Y. Boureau, F. Bach, L. Y., and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[4] Y. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: Multi-way local pooling for image recognition. In *ICCV*, 2011.

[5] L. Cao, Y. Mu, A. Natsev, S.-F. Chang, G. Hua, and J. R. Smith. Scene aligned pooling for complex video recognition. In *ECCV*, 2012.

[6] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, 2007.

[7] H. Jégou and O. Chum. Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening. In *ECCV*, 2012.

[8] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[9] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local descriptors into compact codes. In *Trans. PAMI*, 2012.

[10] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *ICML*, 1997.

[11] A. Joly and O. Buisson. Logo retrieval with a contrario visual query expansion. In *ACM Multimedia*, 2009.

[12] D. Oneata, M. Douze, J. Revaud, S. Jochen, D. Potapov, H. Wang, Z. Harchaoui, J. Verbeek, C. Schmid, R. Aly, K. Mcguiness, S. Chen, N. O'Connor, K. Chatfield, O. Parkhi, R. Arandjelovic, A. Zisserman, F. Basura, and T. Tuytelaars. AXES at TRECVid 2012: KIS, INS, and MED. In *TRECVID*, 2012.

[13] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, B. Shaw, W. Kraaij, A. F. Smeaton, and G. Quenot. Trecvid 2012 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2012.

[14] F. Perronnin, J.Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.

[15] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed Fisher vectors. In *CVPR*, Jun. 2010.

[16] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

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

[19] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *ICCV*, 2007.