Compact Video Description for Copy Detection with Precise Temporal Alignment

Matthijs Douze, Hervé Jégou, Cordelia Schmid, Patrick Pérez

Compact video description for copy detection with precise temporal alignment

Matthijs Douze[1], Hervé Jégou[2], Cordelia Schmid[1], and Patrick Pérez[3]

[1] INRIA Grenoble, France
[2] INRIA Rennes, France
[3] Technicolor Rennes, France

Abstract. This paper introduces a very compact yet discriminative video description, which allows example-based search in a large number of frames corresponding to thousands of hours of video. Our description extracts one descriptor per indexed video frame by aggregating a set of local descriptors. These frame descriptors are encoded using a time-aware hierarchical indexing structure. A modified temporal Hough voting scheme is used to rank the retrieved database videos and estimate segments in them that match the query. If we use a dense temporal description of the videos, matched video segments are localized with excellent precision.
Experimental results on the TRECVID 2008 copy detection task and a set of 38000 videos from YouTube show that our method offers an excellent trade-off between search accuracy, efficiency and memory usage.

1 Introduction

We consider the problem of searching a transformed query video, or part of this query, in a large database of videos. This is important, in particular, for detecting video copies that may be illegally delivered on peer-to-peer networks and user generated content sites such as YouTube. The most common transformations observed in practice are camcording and re-encoding, though sophisticated video post-processing is also encountered.

In recent evaluations [1, 2], the use of local descriptors [3–6] combined with a frame voting system appeared to be the most successful architecture for video copy detection. These state-of-the-art systems search individually for each local descriptor of the query video in a structure indexing all local descriptors of the video database. The typical memory requirement associated with representing the set of local descriptors of a video frame ranges from 1 to 10 Kbytes. This seriously limits the number of video frames that can be indexed in practice. Therefore, the video frames are subsampled, which reduces the capability to find very

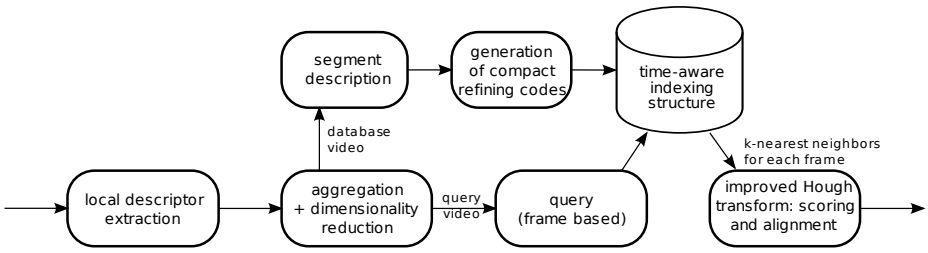

Fig. 1. Overview of our video copy detection method. Local descriptors of a video frame are aggregated into a single vector and the dimension of this vector is reduced. The videos to be indexed are encoded using a temporal-aware indexing scheme. No encoding is applied to the query frame descriptors. A weighted temporal Hough transform provides the final ranking of the database videos w.r.t. the query.

short clips and to determine the precise localization of the query in the database videos. Furthermore, even with subsampling, very large video datasets (several thousands hours of videos) cannot be handled efficiently.

The objective of this paper is to address these scalability and localization issues, while maintaining a high recognition rate. Figure 1 gives an overview of our approach for querying and matching video segments. The individual steps are:

1. Local descriptors are extracted from video frames (either query or database video) and subsequently aggregated into a single vector. This aggregation process is similar to recent approaches for descriptor aggregation [7, 8] which outperform the popular bag-of-features (BOF) representation [9] with descriptors of lower dimension.
2. The dimensionality of this frame descriptor is reduced with either a technique based on a multiplication with a sparse matrix or principal component analysis (PCA).
3. On the database side, the reduced descriptors are encoded within an indexing structure that takes into account the temporal regularity. The video is split in segments. A first description is computed for a segment by minimizing a fidelity criterion for frames of this segment. In the spirit of [10], this segment descriptor is refined by a code based on a product quantizer.
4. Each frame's approximate description is refined by encoding the difference between the frame descriptor and the vector describing the segment it belongs to.
5. A modified temporal Hough voting scheme is used to fuse the votes obtained at the frame level. Its main difference with the conventional

method is that the votes are weighted so that their contribution is penalized if 1) the query frame has received a large amount of votes and 2) the database frame has voted several times.

The paper is organized as follows. The frame description method is introduced in Section 2. Section 3 describes how frame descriptors are indexed and retrieved when a query frame descriptor is submitted. The voting scheme is presented in Section 4. The contributions of the different steps are evaluated in Section 5. Furthermore, we compare to the state of the art on the TRECVID 2008 benchmark, and obtain top results in terms of localization accuracy. The scalability of the approach is demonstrated by experiments on 38000 YouTube videos represented by more than 200 million frames. We show that these videos are indexed in less than 5GB of memory.

2 Video description

2.1 Local description

The first step of our video description extracts a set of local features for each frame. The same approach is used to extract descriptors for the query and database videos. Here, regions are obtained with the scale-invariant Hessian detector [11] and described by CSLBP [12]. Similar to SURF [13] and DAISY [14], this descriptor is a variant of the SIFT descriptor which provides comparable results and reduces the computation time significantly: extracting local descriptors from a frame takes about 35 ms on one 2.4GHz processor core. Note that for large databases, the time for feature extraction is not the critical operation at query time, because it only depends on the query length, not the database size.

2.2 Local descriptor aggregation: non probabilistic Fisher Kernel

Given a set of local descriptors $\{x_1, \ldots, x_i, \ldots\}$ for each video frame, it is impossible to store all of them in memory in the case of large scale video search, even if only a few bytes are required for each of them [15, 16]. In general, several hundreds of local descriptors are extracted for each frame.

We, therefore, aggregate the local descriptors into a single vector. We adopt a variant [8] of the Fisher Kernel image representation introduced by Perronnin et al. [7]. The resulting vector, called vector of locally

aggregated descriptors (VLAD), provides a compact yet effective representation of images. Assuming that a k-means codebook with k centroids $\{c_1, \ldots, c_j, \ldots, c_k\}$ has been learned, we obtain the VLAD descriptor for a frame, denoted by μ, as follows:

1. As for the bag-of-features representation, each local descriptor x_i of the frame is assigned to the closest centroid in the codebook, i.e., to the quantization index $\mathrm{NN}(x_i) = \arg\min_j ||x_i - c_j||$.
2. Given the set of descriptors assigned to a centroid c_j, the vector μ^j is obtained by summing the differences between these descriptors and the centroid:
$$\mu^j = \sum_{i:\mathrm{NN}(x_i)=j} x_i - c_j. \quad (1)$$
3. The VLAD descriptor associated with a frame is obtained by concatenating the vectors μ^j into a single vector.
4. As proposed in [17] for the Fisher Kernel image representation, we apply a power-law normalization to the components to reduce the contribution of the most energetic ones. Here, we use the signed square root of each component. The vector is subsequently L2-normalized and is denoted μ in the following.

The resulting vector is of dimension k times the dimensionality of the input vector, e.g., $k \times 128$ for the CSLBP descriptor. For the same codebook size, the dimensionality of the descriptor is significantly larger than for the bag-of-features representation [9]. However, the VLAD description is already discriminant for low values of k in contrast to BOF, which requires very large codebooks (up to 1 million) to provide the best results [18, 19]. Therefore, the dimensionality of the VLAD vector is typically lower than for BOF. It is worth noting that this representation can be seen as a non-probabilistic version of the Fisher kernel. In the latter, a Gaussian mixture model and soft assignment are used instead of k-means centroids, and additional information (variance and count) are used to obtain a richer (but longer) representation.

2.3 Dimensionality reduction of frame descriptors

Local descriptor aggregation results in one VLAD descriptor per video frame. This descriptor is highly dimensional: for a typical value of $k = 64$, the vector μ has $D = 128 \times k = 8192$ components. Such a vector is difficult to index due to its dimensionality [8]. We, therefore, use and compare two different methods to reduce the dimensionality:

1. Principal component analysis (PCA) allows to reduce the dimension D of the VLAD descriptor to a smaller dimension d [8]. The vector μ is multiplied with a projection matrix $\boldsymbol{M}$ formed by the first principal eigenvectors of an empirical covariance matrix. The PCA matrix is pre-multiplied with a random rotation to "whiten" the output;
2. Alternatively, we define $\boldsymbol{M}$ as a $d \times D$ sparse matrix obtained as $\boldsymbol{M} = \boldsymbol{P}\boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ is a $D \times D$ random permutation matrix and $\boldsymbol{P}$ is a $d \times D$ aggregation matrix that sums several consecutive components. For example with $D = 6$ and $d = 2$, a possible matrix M is:

$$\underbrace{\begin{bmatrix} 0\,1\,1\,0\,1\,0 \\ 1\,0\,0\,1\,0\,1 \end{bmatrix}}_{\boldsymbol{M}} = \underbrace{\begin{bmatrix} 1\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1 \end{bmatrix}}_{\boldsymbol{P}} \times \underbrace{\begin{bmatrix} 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix}}_{\boldsymbol{\sigma}} . \tag{2}$$

The two techniques are evaluated in Section 5. The advantage of using a structured matrix is that there is no need for a training stage. The dimensionality reduction is also cheaper to compute, because the multiplication is more efficient with the sparse matrix $\boldsymbol{M}$ than with the full matrix obtained by PCA. However, during the search, the dimensionality reduction typically has a low computing cost compared to the other steps.

The descriptor $f \in \Re^d$ of reduced dimensionality is obtained by multiplying the matrix $\boldsymbol{M}$ with the VLAD descriptor μ and by L2-normalizing it. The resulting vector f is used in the following as the frame descriptor. The dot product is used as a similarity measure.

3 Indexing frame descriptors with temporal integration

The objective of this section is to provide a compact representation and an efficient search mechanism for the frames of a database video. Let us consider a video to be indexed[4], for which we have extracted a set $f_1, \ldots, f_t, \ldots, f_T$ of d-dimensional descriptors using the method introduced in Section 2. The individual descriptors will be approximated, indexed and searched using three successive refinement levels:

1. joint description of a group of contiguous frames: all the frames associated with the same time segment have the same coarse approximation;

[4] We treat several videos as a single long video, except that we constrain a segment of frames not to cross video boundaries.

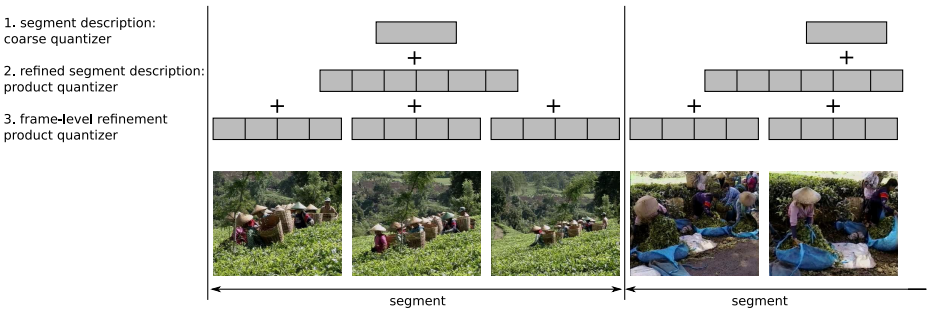

Fig. 2. Hierarchical representation of the frame descriptors. At levels 1 and 2, segments are represented by a frame descriptor. At level 1, the descriptor is quantized to a single index, that is refined at Level 2 by a product quantizer code. At level 3, the individual frames of the segment are represented as a refinement of their segment descriptor. Typically, a segment descriptor is encoded in 4+64 bytes, while each frame is refined by a 16-byte code.

2. refinement of this video segment descriptor;
3. refinement of the individual frame descriptors.

Each of these levels provides an improved approximation of an indexed frame descriptor, as illustrated by Figure 2. This procedure is not used for the frames of the query, i.e., their VLAD descriptors are not approximated. Our approach is, to some extent, similar in spirit to the method proposed in [10]. However, a major difference is the integration of the temporal aspect into the indexing scheme.

3.1 Level 1: Coarse segment description

Due to the temporal dependency between frames, contiguous frame descriptors of a video shot are similar. We exploit this property by providing a shared description for consecutive frames. Let us define a segment $\{t_\mathrm{b}, \ldots, t_\mathrm{e}\}$ as an interval of consecutive frames, for which the same level 1+2 approximation of the descriptor is used.

If segments are of fixed size, we have $t_\mathrm{e} = t_\mathrm{b} + 1/r - 1$, where r is the ratio between the number of segments and the total number of frames in the video. The first approximation is given by a coarse vector quantizer $q_\mathrm{c}(.)$, for which the codebook $\mathcal{C}_\mathrm{c} = \{c_1, \ldots, c_L\}$ is learned using a spherical k-means algorithm. The coarse quantization index $i_\mathrm{c}(t_\mathrm{b} : t_\mathrm{e})$ associated with the segment $\{t_\mathrm{b}, \ldots, t_\mathrm{e}\}$ aims at best representing the set of frame descriptors $\{f_{t_\mathrm{b}}, \ldots, f_{t_\mathrm{e}}\}$ with respect to the total square error, i.e.,

$$i_\mathrm{c}(t_\mathrm{b} : t_\mathrm{e}) = \arg\min_{i \in \mathcal{C}_\mathrm{c}} \sum_{t=t_\mathrm{b}:t_\mathrm{e}} ||f_t - c_i||^2, \tag{3}$$

which is equivalent to

$$c_{i_c(t_b:t_e)} = q_c\left(r \sum_{t=t_b:t_e} f_t\right) \tag{4}$$

The vector $c_{i_c(t_b:t_e)}$ is the level-1 approximation of the frame descriptors in the segment $\{t_b, \ldots, t_e\}$. When searching the nearest neighbors of a given query descriptor, only the database descriptors associated with the k_c closest elements in $\mathcal{C}_c$ are explored.

We have tested both a fixed and adaptive number of frames per segment. Several variants for selecting keyframes have been tested in the adaptive case. Best results were obtained when constructing the segments based on the consistency of the k-nearest neighbors in $\mathcal{C}_c$ for the frame descriptors. However, experimental results showed that no variant is better than uniform subsampling. We, therefore, only use segments of fixed size in the following.

3.2 Level 2: segment descriptor refinement

The index associated with a given video segment is not precise, as an approximation with a centroid in $\mathcal{C}_c$ introduces a large quantization error. Therefore, similar to [10], we refine this first approximation by using a product quantizer q_f, whose codebook[5] is denoted by $\mathcal{C}_f$. The total number of centroids implicitly defined by a product quantizer composed of m_f subquantizers having L_f centroids each is equal to $(L_f)^{m_f}$. This quantizer aims at reducing, over the set of frames associated with a given segment, the average energy of the error vector $f_t - c_{i_c(t_b:t_e)}$ made by the first approximation $q_c(f_t) = c_{i_c(t_b:t_e)}$. The new approximation of a frame descriptor f_t associated with the segment $\{t_b, \ldots, t_e\}$ is, therefore, of the form

$$f_t \approx c_{i_c(t_b:t_e)} + c'_{i_f(t_b:t_e)}, \tag{5}$$

where the centroid $c'_{i_f(t_b:t_e)} \in \mathcal{C}_f$ is obtained by the minimization

$$c'_{i_f(t_b:t_e)} = \arg\min_{c'_i \in \mathcal{C}_f} \sum_{t=t_b:t_e} ||f_t - c_{i_c(t_b:t_e)} - c'_i||^2. \tag{6}$$

[5] A product quantizer decomposes the space into a Cartesian product of low dimensional subspaces and quantizes each space separately. As a result, learning codebooks and searching the quantization index have a low complexity even for very large codebooks. The codebook $\mathcal{C}_f$ has not to be stored explicitly.

The minimization is efficiently done using the decomposition associated with the product quantizer. Note that this quantizer q_f is more precise than the coarse quantizer used in the first stage, because the set of centroids $\mathcal{C}_\mathrm{f}$ that is implicitly defined by the product quantizer is large: it is $2^{8\times 64}$ for the typical 64-byte codes we use ($m_\mathrm{f} = 64, L_\mathrm{f} = 256$). The product quantizer decomposition is used to obtain a complexity comparable to that of a quantizer comprising L_f elements.

3.3 Level 3: refinement of individual frame descriptors

So far, the frames of a segment are described by the same approximation. We now refine the description of each individual frame f_t by using another refinement product quantizer q_r induced by m_r subquantizers with L_r centroids each. This quantizer encodes the error resulting from the two previous approximations by minimizing the quantization error of f_t. For a time instant t such that $t \in \{t_\mathrm{b}, \ldots, t_\mathrm{e}\}$, this is done by quantizing the residual error vector $f_t - c_{i_\mathrm{c}(t_\mathrm{b}:t_\mathrm{e})} - c'_{i_\mathrm{f}(t_\mathrm{b}:t_\mathrm{e})}$. The frame descriptor f_t is therefore approximated by

$$\hat{f}_t = c_{i_\mathrm{c}(t_\mathrm{b}:t_\mathrm{e})} + c'_{i_\mathrm{f}(t_\mathrm{b}:t_\mathrm{e})} + q_\mathrm{r}(f_t - c_{i_\mathrm{c}(t_\mathrm{b}:t_\mathrm{e})} - c'_{i_\mathrm{f}(t_\mathrm{b}:t_\mathrm{e})}). \tag{7}$$

3.4 Search procedure

Searching a query frame vector y in a database of frame descriptors $\mathcal{B} = \{f_1, \ldots, f_T\}$ proceeds in a hierarchical manner.

1. The k_c nearest neighbors of y in $\mathcal{C}_\mathrm{c}$ identify the segments to be considered: only those associated with one of the selected k_c indexes are explored.
2. For each vector f_i in the set of selected lists, the distance approximation

$$\mathrm{l}_2(f_i, y) = \mathrm{l}_2\left(f_i - q_\mathrm{c}(f_i), y - q_\mathrm{c}(f_i)\right) \approx \mathrm{l}_2\left(q_\mathrm{f}\left(f_i - q_\mathrm{c}(f_i)\right), y - q_\mathrm{c}(f_i)\right) \tag{8}$$

   is efficiently obtained from the quantizer indexes $q_\mathrm{f}\left(f_i - q_\mathrm{c}(f_i)\right)$ by exploiting ADC method of [10]. The best segments corresponding to a query vector are found based on the approximation of the square distance of Equation 8. This step returns the set of the k_f nearest segment descriptors.
3. The query frame descriptor y is now compared to all the approximated $\hat{f}_t$ frame descriptors associated with the k_f segments found in the previous stage. This step returns a set of k_r nearest frame descriptors.

3.5 Complexity

The cost of searching a frame descriptor in a database containing T frames is expressed in terms of the number C_{dist} of regular d-dimensional vector comparisons and the amount C_{mem} of memory scanned in the indexing structure. These are given by

$$C_{\text{dist}} = L + k_{\text{c}}\, L_{\text{f}} + \frac{k_{\text{f}}}{r} \tag{9}$$

and

$$C_{\text{mem}} = \alpha\, \frac{k_{\text{c}}}{L}\, r\, T\, m_{\text{f}}\, \log_2 L_{\text{f}} + \frac{k_{\text{f}}}{r}\, m_{\text{r}} \log_2 L_{\text{r}}, \tag{10}$$

where $\log_2 L_{\text{f}} = \log_2 L_{\text{r}} = 8$ bits $= 1$ byte in our case. The factor $\alpha \geq 1$ accounts for the fact that the probability to assign a frame descriptor to an index is not uniform over $\mathcal{C}_{\text{c}}$. As observed in [18] in the context of the BOF representation, this increases the expectation of the number of elements that are processed. We measured that $\alpha \approx 8.4$ with our parameters. Note that our calculation of C_{dist} assumes that the level-2 search is optimized by using look-up tables computed on-the-fly.

4 Temporal alignment: improved Hough transform

Once each query frame has been matched to k_{r} putative frames of the database, the video search matches a sequence of query descriptors to sequence(s) from the database. This sequence matching can be cast in terms of temporal sequence alignment and addressed by dynamic programming techniques, such as dynamic time warping. However, this type of approaches requires a complete frame-to-frame matching cost matrix, which is not feasible at this stage of the detection system. Furthermore, they require a good initialization of the starting and endpoint of the matching sequences.

Simplified approaches can be used instead, e.g., partial alignment [20] or classic voting techniques, such as Hough transform or RANSAC. We adopt a temporal Hough transform [6], as it allows the efficient estimation of 1D temporal shifts. Individual frame votes are re-weighted according to suitable normalizations. In particular, re-weighting is used to reduce the harmful influence of temporal "burstiness", i.e., the presence of query frames that match strongly with multiple, possibly unrelated frames in the database. This is similar to the burstiness of local descriptors in images [21].

4.1 Hough transform

As output of the indexing structure, we have a set of matches, each of them represented by a score $s(\tau, t) > 0$ between the query frame timestamp τ and the database timestamp t. This score is given by the inner product between the query frame descriptor and the approximation of the database descriptor in Equation 7. We set to 0 the scores of frames that are not matched:

$$s(\tau, t) = \begin{cases} \langle y_\tau, \hat{f}_t \rangle & \text{if } t \text{ is retrieved by the index} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The temporal Hough transform consists in computing a histogram $h(\delta)$ to accumulate these scores for all $\delta = t - \tau$ hypotheses. Denoting by $\mathcal{Y}$ and $\mathcal{B} = \{1, \ldots, T\}$ the sets of time instants on the query side and the database side, respectively, the score is formally obtained as

$$h(\delta) = \sum_{\tau \in \mathcal{Y}} s(\tau, \tau + \delta), \tag{12}$$

where $s(\tau, t) = 0$ if $t \notin \mathcal{B}$. Peaks (maximum values) are then searched in the histogram h. We typically select 100 peaks and then apply non maximum suppression, i.e., peaks that are closer than 1 minute to a stronger one are discarded. For each peak identified by δ, the boundaries of the matching segments are then identified by collecting all the matches (τ, t) associated with a hypothesis δ such that $|\tau - t - \delta| < 10$. The final score is the sum of scores of these matches.

4.2 Handling the temporal burstiness

As mentioned in Section 3, consecutive video frames are often very similar, and so are their descriptors. This temporal dependency biases the scores returned by the Hough histogram, as bursts of votes occur for some frames, both on the query and database. This emphasizes them, i.e., they gather an abnormally large amount of scores.

We address this problem by modifying the scoring strategy in a way that mitigates this effect, in the spirit of the re-weighting scheme proposed in [6]. This is done by updating the score, prior to the Hough histogram computation in two steps:

$$s_1(\tau, t) = s(\tau, t) / \sqrt{\sum_{\tau \in \mathcal{Y}} s(\tau, t)} \qquad \text{and} \qquad s_2(\tau, t) = s_1(\tau, t) / \sqrt{\sum_{t \in \mathcal{B}} s_1(\tau, t)}, \tag{13}$$

where the computation is done efficiently by considering only the non-zero score values in the summations. The updated score s_2 is used instead of the original scores in Equation 12. We will show in Section 5 that this procedure significantly improves the quality of the Hough estimation.

5 Experiments

5.1 Datasets and evaluation protocol

Trecvid'08. This dataset contains 200 hours of Dutch television broadcasts. It was used for the copy detection pilot task in the TRECVID'08 evaluation campaign. A set of 134 query clips was extracted from the dataset and 67 clips from other videos were added as negatives, i.e., with no corresponding videos in the database. Some clips were embedded in a distractor video and all were then transformed with 10 different transformations, see Table 2. As a result, 2010 queries with varying degrees of difficulty are used to evaluate a system.

The performance measure used to evaluate the search quality in the TRECVID competition is the *Normalized Detection Cost Ratio* (NDCR), which integrates the cost of missing a true positive with the cost of retrieving an incorrect video. It is equal to 0 if all the true positives are returned before the false positives (no matter how many there are) and lower values of the NDCR correspond to better results. A result video segment is considered as a true positive if it overlaps with the ground-truth. We have used this measure[6] to compare our results with those obtained by the participants of the TRECVID'08 evaluation, see Subsection 5.3.

STV (Small TRECVID). In order to evaluate the parameters of our approach, we created a reduced version of the TRECVID dataset, referred to in the following as STV. It uses a subset of 21 h of the videos. From these videos we extracted a set of 100 clips not used by in the TRECVID'08 queries, embedded them in an independent distractor set and transformed them with some of the most challenging transformations.

This dataset is smaller than the TRECVID'08 dataset. However, the transformations are more challenging on average. We, therefore, obtain comparable conclusions with reduced runtime. Furthermore, using this dataset for parameter evaluation avoids optimizing parameters on the TRECVID query set, and provides a fair comparison with the state of the art on this dataset.

[6] NIST (the institute organizing TRECVID) provided the software to compute this measure as well as the results of the other participants.

YouTube. In order to evaluate the scalability of our system, we have collected a dataset of YouTube videos. We downloaded a total of 38,000 videos from YouTube, corresponding to 189 million frames (or 2100 h). Most of the videos have a similar resolution as the TRECVID ones (about 352*288) and the number of interest points extracted per frame is similar (about 300 per frame on average). These videos are used as distractors in our large scale experiment in Section 5.4.

Evaluation measures. In addition to the NDCR measure used for TRECVID, we have used two additional measures to evaluate performance, localization accuracy and average precision. The localization accuracy for a result segment is measured as the overlap between the ground-truth and the returned result: $\Omega = |T_{\text{gt}} \cap T_{\text{found}}|/|T_{\text{gt}} \cup T_{\text{found}}|$. If the match is incorrect, $\Omega = 0$, and if the localization of a match is perfect, $\Omega = 1$. Better matches have a higher overlap.

We have used the overall Average Precision (AP) as a quality measure. The results returned for all queries are evaluated together ranked by their scores. A match is considered a true positive if the overlap measure is above 0.5. A precision-recall curve evaluates the results as a function of the score. The area under this curve is the AP measure.

5.2 Impact of parameters

Unless stated otherwise, for the parameters introduced in Sections 2 and 3, we have used the following values:

Descriptor	$D = 128 \times k = 8192$	$d = 2048$	$r = 1/10$
Coarse quantizer	$L = 2048$		$k_{\text{c}} = 64$
Fine quantizer	$L_{\text{f}} = 256$	$m_{\text{f}} = 64$	$k_{\text{f}} = 128$
Refinement	$L_{\text{r}} = 256$	$m_{\text{r}} = 16$	$k_{\text{r}} = 32$

In the following, we measure the impact of these parameters. The performance is reported for the STV dataset.

The dimensionality reduction is evaluated on the two first levels of the method, i.e., without frame grouping ($r = 1$) nor refinement (Levels 1+2 only). Our aggregation method is compared with PCA to reduce the $D = 8192$ dimensions of the frame descriptor to $d = 2048$. For this operating point, dimensionality reduction with an aggregator (see Section 2.3) gives AP=87.7, and the PCA-based dimensionality reduction achieves AP=90.1. In the following, we use the PCA-based dimensionality reduction.

Table 1. Evaluation of memory usage, search accuracy and timings on the STV dataset. The method "Levels 1+2, $r = 1$" indicates that frames are directly considered as segments in our method without any further refinement. The timings are given as a slowdown factor w.r.t. the "real" video time for one single 2.4GHz processor core (not including the frame description time).

structure/algorithm	total mem	C_{mem}	C_{dist}	AP	time
none/brute-force search	62 GB	62 GB	19 M	96.7	89 ×
Levels 1+2, $r = 1$	122 MB	25 MB	18432	90.1	2.52 ×
Levels 1+2, no refinement	12 MB	2.71 MB	18432	74.1	1.10 ×
Levels 1+2+3 ($m_{\text{r}} = 16$)	43 MB	2.73 MB	19710	91.2	1.33 ×
Levels 1+2+3 ($m_{\text{r}} = 32$)	73 MB	2.75 MB	19710	90.5	1.42 ×
Levels 1+2+3 ($m_{\text{r}} = 64$)	134 MB	2.79 MB	19710	91.7	1.43 ×

Indexing: impact of the quantization and of the refinement step. Table 1 shows the influence of the descriptor quantization and frame grouping on the search accuracy, memory usage and search time. Brute-force search gives an upper bound on the performance that can be achieved by using our frame descriptor, but is unreasonably expensive. We denote by *Levels 1+2* the methods that do not refine the segment level representation on the frame level. If $r = 1$, then frames are directly considered as segments, while *Level 1+2, no refinement* has the same effect as a subsampling, except that the average frame descriptor over a segment is used instead of a particular frame of this segment. This variant provides lower search quality, but is interesting to index very large datasets. One can observe that subsampling the video and indexing subsampled frames strongly degrades the performance, by 16 points of AP. The refinement improves the results. Short codes ($m_{\text{r}} = 16$ bytes) are sufficient to capture most of the possible improvement. Note that, for large databases, this last refinement stage (Level 3) is the limiting factor in terms of memory usage, even with the setting $m_{\text{r}} = 16$.

Burstiness handling. We have evaluated the impact of our vote regularization procedure which addresses the problem of burstiness (cf. section 4.2). This method significantly improves the results, as shown below:

bursts regularization	AP
none	83.8
database-side (using s_1 in Equation13)	84.3
full regularization (s_2)	91.2

Table 2. Evaluation of our method relative to other TRECVID'08 participants. The score is NDCR (the lower the better). The rank is obtained by taking the best run from each of the 22 participants.

no	transformation	best	second	ours	rank (/23)
1	camcording	0.079	0.363	0.224	2
2	picture in picture	0.015	0.148	0.321	4
3	insertion of patterns	0.015	0.076	0.079	3
4	strong re-encoding	0.023	0.095	0.064	2
5	change of gamma	0.000	0.000	0.023	3
6	photometric attacks	0.038	0.192	0.064	2
7	geometric attacks	0.065	0.436	0.140	2
8	3 random transformations from 6/7	0.045	0.076	0.437	5
9	5 random transformations from 6/7	0.038	0.173	0.693	5
10	5 random transformations	0.201	0.558	0.537	2

5.3 Comparison with state of the art

Table 2 compares the NDCR scores of our system with the best and second best run (from different participants) of the TRECVID'08 competition[7]. We also provide the rank associated with our score. One can see that our system is very competitive, in particular for the most interesting transformations encountered on a peer-to-peer network: camcording and re-encoding. Note that the best score for these transformations is obtained with the approach of [6]. This approach requires 10 GB of RAM against 300 MB used here, i.e., it is difficult to scale to very large video sets. Furthermore, it is more than 5 times slower than the approach presented in this paper (13× "real time" against 2.47× here on a single computing core).

We also compare our localization accuracy with the four best runs (from different participants) of the TRECVID'08 competition. We measure the accuracy on the 195 videos that were correctly retrieved by all 4 runs as well as by our method (i.e. the resulting segment has an overlap with the the ground-truth above 0.5). The measure used is average overlap. The results are:

rank of the participant	1	2	3	4	ours
mean overlap	0.952	0.858	0.846	0.884	0.973

We can observe that our approach localizes the segments very precisely, i.e., with a better precision than the competing approaches.

[7] The competitors can not be identified by name due to the non disclosure agreement of TRECVID.

5.4 Large scale experiment

The large scale experiments are performed on the YouTube dataset merged with our STV dataset. Figure 3 shows the AP obtained as the function of the growing database size (up to 2316 hours). When performing the experiments on the whole set, the index requires 4.6 GB of RAM to index 208 million frames. The search is slower on that scale: 23.5 real time for a single processor core. One can observe that the AP measure decreases as to be expected, but that results are still good, i.e., we obtain an AP=0.53 on the entire set. Typical retrieval results of our system are shown in Figure 4.

6 Acknowledgements

This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

References

1. P. Over, G. Awad, T. Rose, J. Fiscus, W. Kraaij, and A. Smeaton, "Trecvid 2008-goals, tasks, data, evaluation mechanisms and metrics," in Trecvid, 2008.
2. J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford, "Video copy detection: a comparative study," in *CIVR*. New York, NY, USA: ACM, 2007, pp. 371–378.
3. D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
4. K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," IEEE *Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.
5. A. Joly, "New local descriptors based on dissociated dipoles," in *CIVR*, 2007.
6. M. Douze, H. Jégou, and C. Schmid, "An image-based approach to video copy detection with spatio-temporal post-filtering," *IEEE Transactions on Multimedia*, vol. 12, no. 4, pp. 257–266, jun 2010. [Online]. Available: http://lear/pubs/2010/DJS10
7. F. Perronnin and C. R. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, June 2007.
8. H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.
9. J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, 2003, pp. 1470–1477.
10. H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 2010.
11. K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.
12. M. Heikkila, M. Pietikainen, and C. Schmid, "Description of interest regions with local binary patterns," *Pattern Recognition*, vol. 42, no. 3, pp. 425–436, mar 2009.

13. H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.
14. S. Winder, G. Hua, and M. Brown, "Picking the best Daisy," in *CVPR*, June 2009.
15. V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod, "Chog: Compressed histogram of gradients: A low bit-rate feature descriptor," in *CVPR*, June 2009.
16. M. Calonder, V. Lepetit, P. Fua, K. Konolige, J. Bowman, and P. Mihelich, "Compact signatures for high-speed interest point description and matching," in *ICCV*, September 2009.
17. F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-scale image retrieval with compressed Fisher vectors," in *CVPR*, 2010.
18. D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006, pp. 2161–2168.
19. J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *CVPR*, 2007.
20. M.-C. Yeh and K.-T. Cheng, "Video copy detection by fast sequence matching," in *CIVR*, 2009.
21. H. Jégou, M. Douze, and C. Schmid, "On the burstiness of visual elements," in *CVPR*, June 2009.

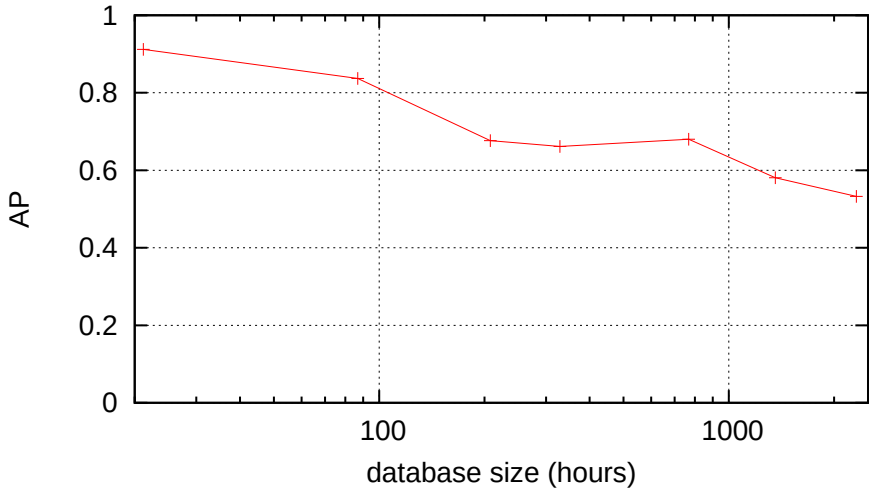

Fig. 3. Retrieval performance on the STV dataset combined with a varying number of videos from the remaining TRECVID videos and YouTube.

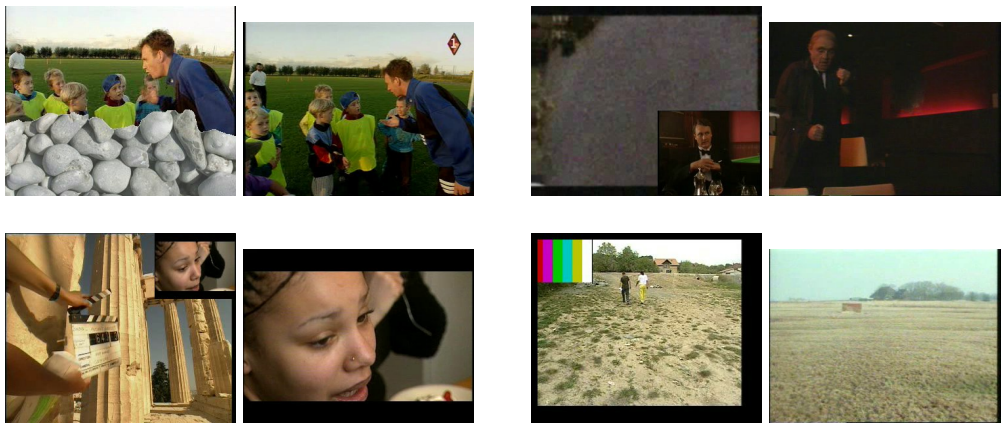

Fig. 4. Example results for video retrieval in our large scale dataset, (left) query and (right) best retrieved video. Lefts pairs: correct retrieval results. Right pairs: incorrect retrieval. Note the visual similarity between the queries and the retrieved videos.