An image-based approach to video copy detection with spatio-temporal post-filtering

Matthijs Douze, Hervé Jégou, Cordelia Schmid

An image-based approach to video copy detection with spatio-temporal post-filtering

Matthijs Douze, Hervé Jégou and Cordelia Schmid

***Abstract*—This paper introduces a video copy detection system which efficiently matches individual frames and then verifies their spatio-temporal consistency. The approach for matching frames relies on a recent local feature indexing method, which is at the same time robust to significant video transformations and efficient in terms of memory usage and computation time. We match either keyframes or uniformly sampled frames. To further improve the results, a verification step robustly estimates a spatio-temporal model between the query video and the potentially corresponding video segments.**

Experimental results evaluate the different parameters of our system and measure the trade-off between accuracy and efficiency. We show that our system obtains excellent results for the TRECVID 2008 copy detection task.

I. INTRODUCTION

The task of video copy detection determines if a given video (query) has a duplicate in a set of videos [1]. This problem has received increasing attention in recent years, due to major copyright issues resulting from the widespread use of peer-to-peer software and users uploading video content on online sharing sites such as YouTube and DailyMotion.

Query videos may be distorted in various ways. Common distortions are scaling, compression, cropping and camcording. If the system finds a matching video segment, it returns the name of the database video and the time stamp at which the query was copied.

Some previous research [2], [3] has been carried out on datasets of more than ten thousand hours of videos. In this case, the indexing structure must be stored in part on disk. Others [4], [5] address the problem of detecting repeated subsequences, such as advertising clips, from a video stream. In this case the video quality is usually high and the deformations consistent across sequences. Therefore, a simple global description in combination with hashing suffices to represent and match the videos. Here, we assume that the dataset is smaller than one thousand hours and that the transformations are severe. In this setup, a system can store the main indexing structure in RAM, which is fast. It is also possible to use a dense and precise video description to find copies that are difficult to identify.

These hypotheses are relevant in practical situations. When monitoring user-generated content for copyright, the dataset typically consists of a few current "hot items", like shots of recent sports events or the last block-buster movie. The elements of the database are those with the highest commercial value. In this context, from the user's point of view, the video upload time is the limiting factor. The video post-processing (format conversion, copyright monitoring) may be slow (up to 2 or 3 times the playback time) without degrading the user experience.

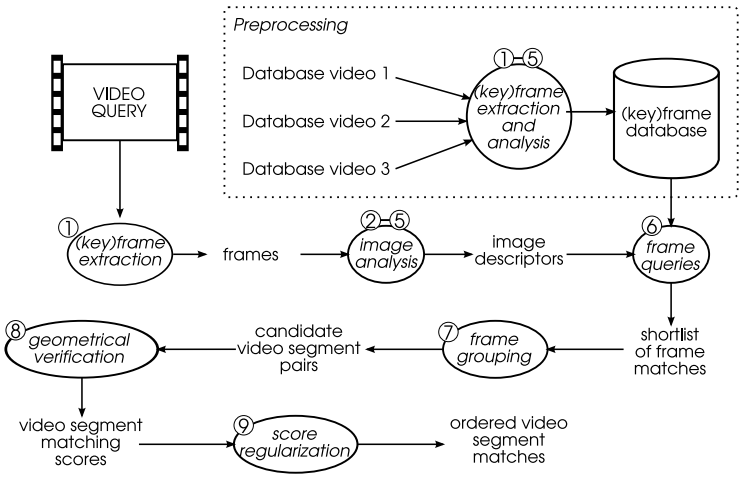
In this paper we present a system which addresses the problem of searching for strongly deformed videos in relatively small datasets. An overview of our system is shown in Figure 1. The first step consists in extracting local signatures for a subsample of frames from the video. We extract local patches with the Hessian-Affine region detector [6] and describe them with the SIFT or CS-LBP descriptors [7], [8]. A query is then performed using a structure derived from text retrieval: the inverted file. These steps were first proposed in the so-called *Video-Google* approach of [9], [10], which popularized the bag-of-features representation for image and video retrieval. In this approach, the feature descriptor representation is simply a quantization index, called *visual word*. Here, we use a recent extension [11] of this approach. In this method, the visual word is augmented with a binary signature that refines it. Note that concurrent approaches [12], [13] were proposed to compute binary signatures. We chose the *Hamming Embedding* method of [11] because it can be easily used in a bag-of-features framework.

The second refinement is the use of partial geometrical information, based on the *weak geometry consistency* (WGC) method [11]. The WGC check is integrated in the inverted file and efficiently exploited for all indexed frames, even for a very large dataset: in this paper, we have indexed up to 2 million frames, represented by 800 million local descriptors. This is in contrast with image matching techniques like the epipolar geometry estimation [14], which can only be used for a limited set of frame comparisons, even when using a simplified geometry model [7], [15].

Matched frames are grouped into sequences. Similar to [1], [3], a spatio-temporal model is estimated robustly. Our spatio-temporal model first determines the temporal shift based on 1D Hough voting. We, then, determine the spatial component by estimating a 2D affine transformation between the matching video sequences. We introduce a normalization of the scores to make them comparable across queries. The scoring approach is similar in spirit to measuring burstiness [16] which has shown excellent results for image retrieval.

This paper is organized as follows. Section II describes our approach for frame indexing, i.e., how to extract, analyze and index frames. In section III, corresponding frames are grouped into video sequences that are robustly matched with a spatio-temporal model and for which adjusted matching scores are computed. The experimental section IV presents an analysis of the key parameters and measures the trade-off between accuracy, memory usage and efficiency. It also gives the results obtained with our approach on the TRECVID 2008 copy detection task [17].

All authors are with INRIA. E-mail: firstname.lastname@inria.fr



① The system selects frames from the input videos. We use two frame extraction methods, *uniform sampling* and *keyframes* (Subsection II-A).

②–⑤ The selected frames are processed to produce an image description based on local features (Subsection II-B).

⑥ The descriptors are access keys into a database of local features (Subsection II-C). It outputs a shortlist of the most likely corresponding frames.

⑦ The matching frames are grouped temporally into candidate video segment matches (Subsection III-B).

⑧ Video segments are filtered with a spatio-temporal model (Section III). Each segment is assigned a matching score, and segments for which the score is too low are removed.

⑨ The segment matches are post-processed to produce a score that is comparable across queries (Subsection III-D).

Fig. 1. Overview of our video copy detection system. Each processing step is identified by a circled number.

II. Frame indexing

This section presents our approach for indexing individual frames, i.e., steps ① to ⑥ in the overview Figure 1. We first describe our strategy for sampling frames, then give details on the extraction of local features and on the bag-of-features representation.

A. Frame sampling ①

Processing and indexing all frames from the query and/or database videos would be too costly and also inefficient due to the temporal redundancy. We, therefore, subsample the frames. As we aim at matching two video sequences, the time lag between sampled frames of the two videos should be low. This ensures that the scene does not change too much and remains recognizable by the image matching engine. In our experiments, we use either of two frame sampling techniques.

Uniform sampling selects a fixed number of frames per second. The impact of the sampling rate is studied in the experimental section IV.

Keyframes are characteristic frames of the video; they should be repeatable to obtain similar frames for two matching videos. To extract keyframes, we detect shot boundaries by measuring graylevel changes on a spatio-temporal slice of the video and thresholding them [18]. We sample a frame at a fixed offset after each detected shot boundary. Shot boundaries are repeatable and the offset (0.5 s) ensures that the sampled frame is not disturbed by compression artifacts around the boundary itself. This samples a frame on average every 6 s.

In the presence of strong deformations, using keyframes is not sufficient to guarantee retrieval of the correct video in the database. A simple solution is to sample the query and the database videos differently, i.e., to apply an *asymmetric sampling* strategy: keyframes are sampled for the database videos, whereas the query video is uniformly and densely sampled. Compared with a keyframe sampling strategy on both sides, we loose the ability to use shot boundaries as recognizable timestamps; on the other hand, many query videos do not include a single shot boundary, so these would be impossible to retrieve in this symmetric setting. The asymmetric strategy does not increase the volume of the database and keeps the maximum time lag short (since the highest sampling rate determines the maximum time lag).

B. Local features ②–③

Local features are extracted individually for each sampled frame of the video. Figure 2 ②-③ presents the descriptor extraction. Our image search system is based on local invariant descriptors [6], [7]. We detect salient interest points and then describe the pattern of the surrounding regions. Such a local description can independently match small parts of video frames, which is required, for example, to resist pattern insertions. In the following, we present the approach used in our system for detecting regions of interest and computing descriptors.

② **Detector:** We detect Hessian-Laplace regions [6] using the software of [19] with default parameters. This region detector is invariant to several image transformations:

- Scale change: The Hessian interest point detector in combination with automatic scale selection [20] is invariant to scale changes.
- Image rotation: Interest points are invariant to image rotation, as they are extracted based on the determinant of the Hessian matrix. To make the corresponding patches invariant, they are rotated so that the dominant orientation is aligned with a common direction.
- Noise: Detection is performed at multiple scales, to be robust to high-frequency noise, blurring, encoding artifacts, etc.

Affine invariance of the interest regions [6] could be of interest in the case of camcording (re-filming a movie from the screen, which often entails perspective distortions). However, affine normalization makes the descriptors less discriminative and robust. In our experiments, on a dataset of mainly non perspective transformations, the results with an affine invariant detector were below those with a simple scale invariant one. We, therefore, chose not to use affine invariance in our system.
③ **Descriptor:** We use SIFT [7] or center-symmetric local binary patterns (CS-LBP) [8]. Both are computed on local image patches. They do not use color, and are normalized

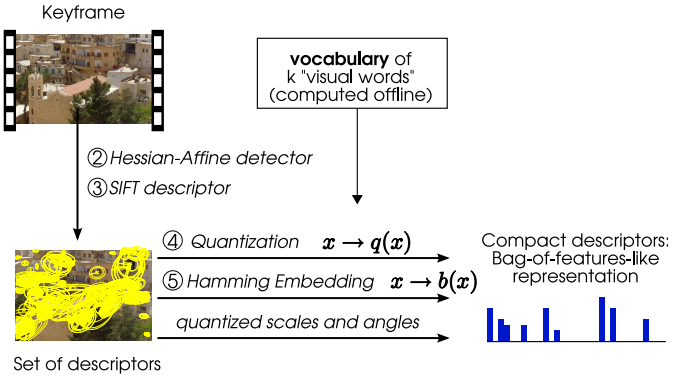


Fig. 2. Image representation for a frame: descriptor extraction and conversion to a compact representation.

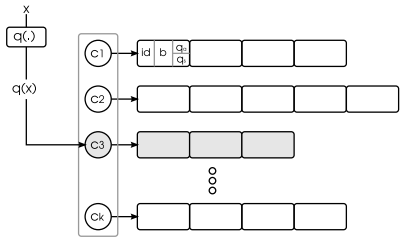


Fig. 3. Inverted file structure. Each descriptor is assigned to the closest visual word and represented by its image index, its binary signature and its quantized angle and scale.

to be invariant to affine illumination changes of the patches. The description is, therefore, invariant to most photometric changes. Both SIFT and CS-LBP produce 128-dimensional vectors. For SIFT we used the implementation of [7] with default parameters. Our implementation of CS-LBP is significantly faster than the public SIFT implementation and provides comparable results.

C. Bag-of-features and Hamming Embedding ④–⑥

Our image indexing system builds on the state-of-the-art image search engine proposed in [11], which improves the "Video-Google" system introduced by Sivic and Zisserman [9]. The key steps of our system (Figure 2 ④-⑤) are detailed below.

Visual codebook generation (offline): The quantizer partitions the space of descriptors into Voronoi cells. Each cell is identified by a representative point: the centroid. Centroids are often called "visual words" belonging to a "visual vocabulary". Our visual vocabulary has been generated with k-means clustering on a subset of descriptors from the video database. We have used $k = 200\,000$ visual words in all our experiments, as a large vocabulary increases the query speed [15].

④ **Assigning the descriptors to visual words:** For the indexed frames, each local descriptor is assigned to its closest visual word. This quantization step amounts to representing a descriptor by the corresponding centroid index $q(x)$. During the query, each descriptor is assigned to *several* closest visual words (the multiple assignment of [16]). This asymmetric assignment strategy improves the accuracy without increasing the memory usage of the indexing structure.

⑤ **Hamming Embedding:** Given the visual word $q(x)$ of a descriptor x, we only know that x belongs to a quantization cell. Due to the high dimensionality of the descriptors, comparing descriptors with the cell index is not very precise, i.e., quite different descriptors match.

To address this problem, we have used the Hamming Embedding method proposed in [11]. The key idea is to represent a descriptor by both the index $q(x)$ and a binary signature $b(x)$ (here of length 64), where $b(.)$ is the Hamming Embedding function associated with the visual word $q(x)$. It is designed so that the Hamming distance

$$h(b(x), b(y)) = \sum_{i=1..64} |b_i(x) - b_i(y)| \qquad (1)$$

between two descriptors x and y lying in the same cell approximately provides the same nearest neighbors as those obtained by the Euclidean distance $\|x - y\|_2$. A descriptor is now represented by $q(x)$ and $b(x)$. The descriptor matching function f_{HE} is defined as

$$f_{\text{HE}}(x, y) = \begin{cases} w(h\,(b(x), b(y))) & \text{if } q(x) = q(y) \\ & \text{and } h\,(b(x), b(y)) \leq h_{\text{t}} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where h_{t} is a fixed Hamming threshold and $w(.,.)$ is a soft weighting function that gives higher scores to smaller Hamming distances [16]. In our system we set $h_{\text{t}} = 22$. If we apply this threshold h_{t} on non matching images, only one descriptor out of 3 million is considered a match (93.5% of the cell's descriptors are filtered out by the binary signature check).

Given a query frame with m' descriptors $y_{i'}$, $i' = 1..m'$, the score associated with frame j is given by

$$s_j = \alpha_j \sum_{i'=1..m'} \sum_{i=1..m_j} f_{\text{HE}}\,(x_{i,j}, y_{i'})\,, \qquad (3)$$

where m_j is the number of descriptors of frame j. The normalization factor α_j is typically computed as $\alpha_j = 1/\sqrt{N' N_j}$, where N' (resp. N_j) is the L2 norm of the bag-of-features vector of the query image (resp. database image j). In addition, a weighting scheme is included to reduce the effect of visual bursts [16].

⑥ **Inverted file:** In order to compute the score of Equation 3 efficiently, the entire set of descriptors of the video dataset is stored in a structure similar to the inverted file used in text retrieval, and used in the image search system of [9]. This structure is composed of k lists of descriptor entries, each list being associated with a visual word. This greatly reduces the complexity, because only the descriptors assigned to the same quantizer centroid as the query descriptor are processed.

We store *one entry per descriptor* in the inverted list of the corresponding visual word. The entry contains:

- the image identifier j ;
- the binary signature $b(x)$;
- the quantized dominant orientation $q_{\text{a}}(x)$;
- the quantized scale $q_{\text{s}}(x)$.

The resulting structure (Figure 3) uses 12 bytes per local descriptor, see [11] for details. In addition to the filtering steps based on $q(x)$ and $b(x)$, the differences in orientation and log-scale are estimated for each point match. Matches that are not consistent in terms of rotation and scaling with the dominant

hypothesis for database image j are removed. This strategy is the weak geometry consistency (WGC) method of [11].

The output of all frame queries is a set of tuples

$$(t_{\mathrm{q}}, b, t_{\mathrm{b}}, s_{\mathrm{f}}), \tag{4}$$

where t_{q} and t_{b} are the timestamps of the matched query and database frames, b is the database video (b and t_{b} are computed from the identifier j), and s_{f} is the score of the frame match, computed by Equation 3 after WGC has been applied.

III. Spatio-temporal verification

The goal of spatio-temporal verification is to score video segment matches using geometric information, in the spirit of geometric ranking [15] and registration [3].

A. The spatio-temporal transformation

A general spatio-temporal transformation that maps points from a frame of a database video $(x_{\mathrm{b}}, y_{\mathrm{b}}, t_{\mathrm{b}})$ to a query frame $(x_{\mathrm{q}}, y_{\mathrm{q}}, t_{\mathrm{q}})$ is given by

$$(x_{\mathrm{q}}, y_{\mathrm{q}}, t_{\mathrm{q}}) = \big(M_{\mathrm{x}}(x_{\mathrm{b}}, y_{\mathrm{b}}, t_{\mathrm{b}}), M_{\mathrm{y}}(x_{\mathrm{b}}, y_{\mathrm{b}}, t_{\mathrm{b}}), M_{\mathrm{t}}(t_{\mathrm{b}})\big). \tag{5}$$

We restrict this transformation as much as possible, both to simplify the parameter estimation and to avoid overfitting. We assume that

- the spatial transformation between the query and the database video is approximately constant in time, similar to [1]: $M_{\mathrm{x}}(x_{\mathrm{b}}, y_{\mathrm{b}}, t_{\mathrm{b}}) = M_{\mathrm{x}}(x_{\mathrm{b}}, y_{\mathrm{b}})$, $M_{\mathrm{y}}(x_{\mathrm{b}}, y_{\mathrm{b}}, t_{\mathrm{b}}) = M_{\mathrm{y}}(x_{\mathrm{b}}, y_{\mathrm{b}})$. This implies in the case of camcording that the video camera is fixed.
- the spatial model is affine:

$$\begin{bmatrix} M_{\mathrm{x}}(x_{\mathrm{b}}, y_{\mathrm{b}}) \\ M_{\mathrm{y}}(x_{\mathrm{b}}, y_{\mathrm{b}}) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_{\mathrm{b}} \\ y_{\mathrm{b}} \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}. \tag{6}$$

  Such an affine model is sufficient for the most common transformations. Camcording results in a full homography, but can in most cases be approximated by an affine transformation [15].
- the temporal model is a simple time shift: $M_{\mathrm{t}}(t_{\mathrm{b}}) = t_{\mathrm{b}} + \delta t$. This suffices for common attacks. Problematic are slow-motion or fast-forward (which correspond to a 1D affine transformation), and re-editing a video using cuts (the transformation is non-continuous).

Our two approximations (homography $\approx$ 2D affine and temporal affine $\approx$ time shift) are sufficiently accurate for most video queries. Introducing some tolerance into the parameter estimation handles the noise due to these approximations. Given this simplified transformation model, the transformation is obtained in two steps: 1) estimation of the temporal parameter δt and 2) of the spatial affine transformation $(a_{11} \dots a_{23})$.

B. Temporal grouping ⑦

The algorithm in Section II returns a set of frame matches $(t_{\mathrm{q}}, b, t_{\mathrm{b}}, s_{\mathrm{f}})$, see Equation 4. Each match results in a hypothesis $(b, \delta t)$, where $\delta t = t_{\mathrm{q}} - t_{\mathrm{b}}$, which aligns the query with the database video b.

Hypotheses vote in the temporal Hough space of the corresponding database video b with the 1D value δt. The time shift for each database video is obtained as the local maximum in the Hough space—the 1D histogram H_b of δt, see Figure 4. The histograms have a bin size of $t_{\mathrm{bin}} = 0.5\,\mathrm{s}$ and are soft-assigned.

A shortlist of $(b, \delta t)$ hypotheses is obtained as the bins with the highest scores (here 500 hypotheses). We group together the frame matches that vote for a hypothesis, then we split the groups into short segments (less than $t_{\mathrm{seg}} = 60\,\mathrm{s}$). This separates matches that have similar time shifts, but correspond to different parts of a video. This removes the influence of incorrect non contiguous matches. Additionally, the resulting time segments are short enough to be verified based on geometric spatial matching, as explained below.

C. Spatial verification ⑧

Given a group of temporally consistent frame matches, we now estimate the spatial transformation between the matched frames of a group. Here, we estimate a 2D affine transformation between video segments. Our approach is an extension of [7] to videos. The outline of the algorithm is as follows:

1) take all point matches $(p_{\mathrm{q}}, p_{\mathrm{b}})$ from the matching frames, using Equation 2. Our estimation simultaneously uses point matches from all matching frames, as we assume a time-independent spatial model. Compared to pure image matching, this increases the number of point matches and improves the estimation quality.
2) estimate possible similarity transformations (4 DOF) from all matching points with a Hough transform. Each matching point pair $(p_{\mathrm{q}}, p_{\mathrm{b}})$ allows to estimate a similarity transformation based on point location, characteristic scale and dominant orientation (see Section II-B). Each point match votes with the estimated parameters in a 4D soft-assigned histogram
3) compute and score possible affine transformations. Each non-empty bin of the 4D histogram corresponds to a hypothesis for a similarity transformation. Based on Equation 6, the affine transform parameters (a_{ij}) are estimated in the least squares sense from the point matches that contribute to the bin. The score for this transformation is the sum of the scores of all the point matches that agree with this transformation, weighted by the geometric likelihood score $w(a)$ defined in Equation 8.
4) select the maximum score over all possible hypotheses. If the maximum score over all bins, denoted by s_{v}, is above a threshold, a positive video segment is returned together with the affine transformation (a_{ij}) and the frames matches that correspond, up to tolerance, to the estimated transformation.

The weight $w(a)$ of the transformation measures the geometrical likelihood of an affine transformation. We express the affine transformation of Equation 6 with another set of parameters:

$$e^{\sigma} \begin{bmatrix} e^{r} & \\ & e^{-r} \end{bmatrix} \begin{bmatrix} \cos(\alpha) & -\sin(\alpha + \alpha_2) \\ \sin(\alpha) & \cos(\alpha + \alpha_2) \end{bmatrix} \begin{bmatrix} x_{\mathrm{b}} \\ y_{\mathrm{b}} \end{bmatrix} + \begin{bmatrix} x_{\mathrm{t}} \\ y_{\mathrm{t}} \end{bmatrix}. \tag{7}$$

The new parameters have a clear geometrical meaning.

- $(x_{\mathrm{t}}, y_{\mathrm{t}})$ encodes a translation. Translations do not penalize the weight w;
- σ is the global scale. We penalize too strong scale factors (more than a factor 2);

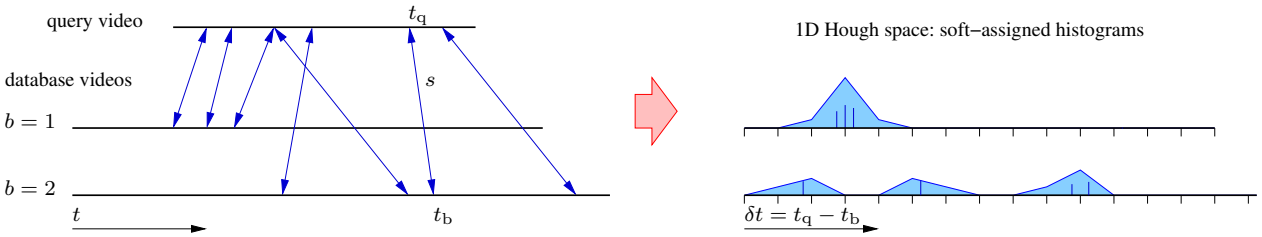


Fig. 4. Left: Frame matches between a query video and two database videos. Right: corresponding δt Hough histograms. They are soft-assigned, the tics indicate the quantized values. There are more frame matches between the query and video 2, but the matches with video 1 are temporally consistent.

- r is an anisotropic scaling. Such a scaling occurs quite often, due to a conversion between video aspect ratios (4:3, 16:9, 2.21:1, etc.);
- $\alpha \in [-\pi, \pi)$ is a global rotation of the frame, which is unlikely to appear in copied videos;
- α_2 marks a skewed video, which is even less likely.

Note that if all parameters are zero, the transformation is the identity. The weight $w(a)$ is computed as:

$$w(a) = \exp\left(-\frac{\sigma^2}{w_\sigma^2} - \frac{r^2}{w_r^2} - \frac{\alpha^2}{w_\alpha^2} - \frac{\alpha_2^2}{w_{\alpha_2}^2}\right). \quad (8)$$

The penalty coefficients $(w_\sigma, w_r, w_\alpha, w_{\alpha_2})$ are adjusted on a validation set. The weight $w(a)$ allows the system to remove weak video segment correspondences that are geometrically improbable.

D. Score aggregation strategy ⑨

We now have a set of matching segment pairs and a score s_v for each of them. This score depends on the number of matching descriptors between the pair of segments, i.e., it is the weighted sum of all matching descriptors scores (Equation 2). This score varies from one query to another, depending on the number of interest points as well as the length of the segments.

To address this problem, we have introduced a frame score normalization procedure. It reduces the contribution of query frames that match well with several videos from the dataset, in the spirit of burstiness scoring for descriptors [16]. We, first, compute the sum t_f of all frame matching scores s_f associated with a given query frame (Equation 4). We, then, update the score of a frame match by

$$s_f := s_f \times \left(\frac{s_f}{t_f}\right)^2. \quad (9)$$

Hence, if a query frame votes for only one dataset video frame, the score s_f is not modified. Conversely, if a frame votes with similar strength for different videos, the contribution of this frame to the final score is greatly reduced.

A video segment score s_v is then obtained as the sum of its frame scores divided by the number of query video frames. This score is finally updated by taking into account the set of matching videos by

$$s_v := s_v \times \left(\frac{s_v}{m_v}\right)^2, \quad (10)$$

where m_v is the highest score obtained among all the matching video segments. This update normalizes the video matching score with respect to the best match and reduces the number of false positives when a decision threshold is used (as is the case for the NDCR measure, see Section IV).

IV. Experiments

We first evaluate the parameters of our system. Next we discuss how our system can cope with the TRECVID 2008 copy detection attacks. Finally, we present our results for the TRECVID 2008 evaluation.

A. Parameter optimization

Validation dataset: As the validation set provided by TRECVID 2008 was too small and not challenging enough, we have created our own validation dataset. We have implemented a video transformation tool based on the transformations specified for the TRECVID 2008 copyright evaluation task[1]. Note that, as required by the evaluation procedure [17], we have not tuned our system on the TRECVID test videos. The algorithm to build the validation dataset is:

1) select 38 random videos (total 21 hours 11 min) from the TRECVID dataset: this dataset, denoted by B, will be indexed;
2) select 31 random subsequences from an independent video source (episodes of "Buffy the Vampire Slayer"). The length of these subsequences ranges from 21 to 179 seconds. This is the distractor dataset D;
3) insert into each of the 20 first videos of D a subsequence (14 to 56 s) drawn randomly from a video in B: this is the sequence that has to be retrieved. The other videos in D are unchanged. This produces the dataset Q_0;
4) transform each video from Q_0 with 5 combinations of the most challenging attacks from the TRECVID specifications (camcording + blurring, picture-in-picture + text overlay, pixelization + low-bitrate encoding, pixel noise + low-bitrate encoding, black margin addition + blurring). This results in the query set Q (total 3 hours 54 min).

The videos are analyzed with a sampling rate of one frame out of two (12.5 frames per second) and the CS-LBP descriptor.

Evaluation measure:

The system retrieves all videos contained in Q. This results in a set of tuples

$$(q, t_q, t'_q, b, t_b, t'_b, s_v), \quad (11)$$

[1]Code available at http://lear.inrialpes.fr/software.

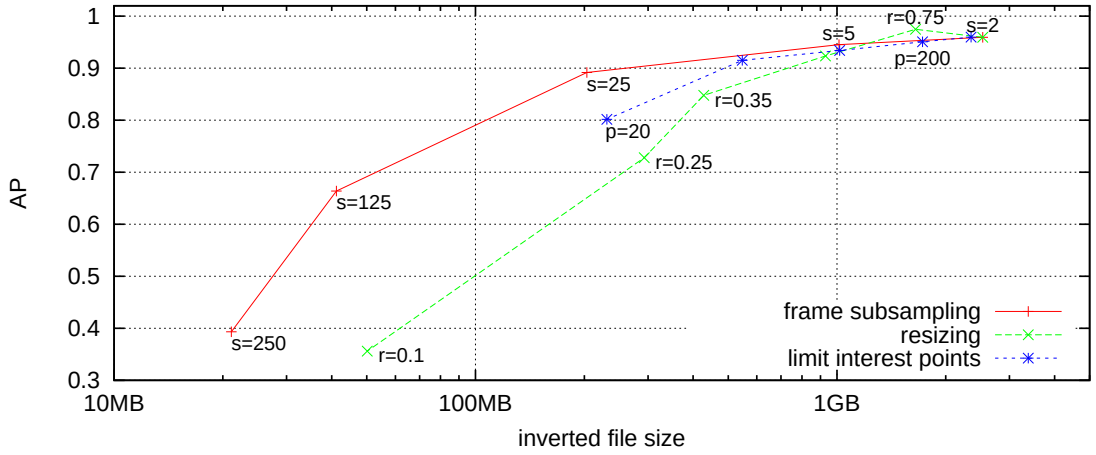


Fig. 5. Inverted file size and retrieval performance for various operating points of the system defined by frame subsampling, frame resizing and number of interest points.

meaning that time range $[t_q, t'_q]$ of $q \in Q$ is found to correspond to time range $[t_b, t'_b]$ of $b \in B$ with a confidence score of s_v.

For such results the TRECVID evaluation document [21] defined a time-based precision and recall. Within the time range $[t_b, t'_b]$, there is a true and a false positive part. The true positive part is the intersection of $[t_b, t'_b]$ with the subsequence that was extracted at Step 3 to produce q. The *recall* is defined as the ratio of the total length of the true positive parts over the total length of material that was copied. The *precision* is the ratio of the total length of true positive parts over the total length of found copies.

By removing the results with scores s_v below a threshold, several operating points with different tradeoffs between precision and recall can be obtained. In the TRECVID copy detection evaluation, a specific operating point was chosen to compute the F1 measure. Here, we prefer to compute the average precision (AP) over all operating points (i.e. the area under the precision-recall curve).

Experiments: The validation dataset was used

- to design the temporal and geometrical verification used in the re-ranking stage ⑧ and optimize the parameters of Equation 8;
- to adjust the scoring strategy ⑨ to produce scores that are consistent across queries, leading to Equations 9 and 10;
- to find a good trade-off between dataset size and accuracy.

In the following, we analyze this trade-off. For our validation videos, the optimal detection parameters give almost perfect results (AP=0.990). This is at the cost of a large inverted file system (2.5 GB) and a slow processing time.

In the following, we analyze the impact of the parameters on the size of the inverted file. The parameters are only varied for the database videos, as the query videos are always analyzed with the maximum quality (asymmetric description, see II-A). The impact of the parameters is shown in Figure 5:

- the parameter p is a limit on the number of keypoints detected per frame (they are selected baed on their cornerness);
- the parameter r is a scaling factor applied to the image. Lower-resolution images have fewer keypoints and are faster to preprocess;

Stage	frame subsampling $s=2$	frame subsampling $s=25$	asymptotic complexity
database descriptors	222:25	17:47	$O(L_b)$
query descriptors	41:03	41:03	$O(L_q)$
frame search	68:52	9:02	$O(L_q L_b)$
spatio-temporal verification	4:16	0:12	$O(L_q)$
total for a query	115:56	50:39	$O(L_q L_b)$
slowdown w.r.t video time	30×	13×	

TABLE I
RUNTIME OF THE PROCESSING STAGES, FOR ONE PROCESSING CORE (HOURS:MINUTES). THE DATABASE CONTAINS $L_b = 21$ HOURS 11 MIN OF VIDEO AND THERE ARE 155 QUERIES OF $L_q = 3$ HOURS 54 MIN IN TOTAL. COMPLEXITIES ARE INDICATED WITH RESPECT TO THESE LENGTHS.

- the parameter s is the frame subsampling rate. By default, it is set to 2 (12.5 frames per second are processed).

Overall, the results show that if storage space and time are important, it is better to reduce the sampling rate than the quality of the image analysis. Interestingly, the best results are obtained by reducing the frame size to 3/4. This is probably an artifact due to one of the more difficult transforms, which included a rescaling of 1/2.

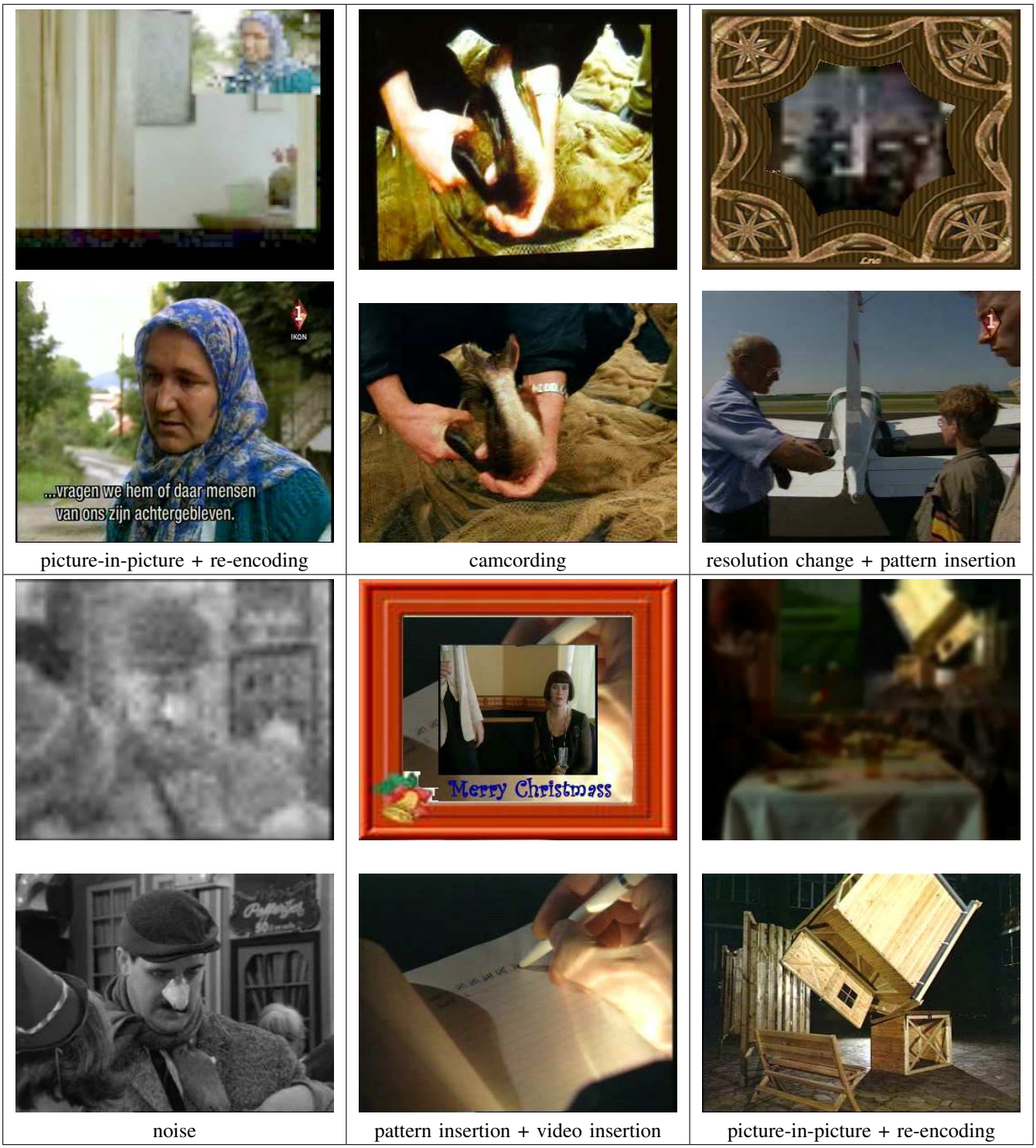
Table I shows the processing time required by each stage of a query. For this small dataset, the descriptor computation and quantization take most of the time. With larger datasets, frame querying becomes dominant. Note that all algorithms can easily be multi-threaded.

B. *Handling of* TRECVID *attacks*

The image matching part of our system (stages ②-⑥ and ⑧) was developed to handle pictures of natural scenes seen under different viewing conditions. In the following, we review how it responds to the TRECVID transformations and the adaptations we have made to our system to handle them.

Frame dropping: As our system is based on frame matching (without motion information), it is not disturbed by dropped frames.

Change of gamma/contrast: The CS-LBP and SIFT descriptors are robust to this change, as they are invariant to affine transformations of the illumination.



picture-in-picture + re-encoding | camcording | resolution change + pattern insertion

noise | pattern insertion + video insertion | picture-in-picture + re-encoding

Fig. 6. Example frames of the transformed videos that our system recognizes correctly (top) and of the corresponding database videos (bottom).

Blur, blocks, re-encoding, noise: We observe that, taken alone, these transformations do not degrade the quality of the frame matching. This is due to the multi-scale detection of interest points: the transformations have little influence on large-scale points, which remain stable.

Camcording, occlusions, cropping: Camcording and partial occlusion represent moderate changes in viewing conditions. Local descriptors can cope with occlusions and crops, as they remain unchanged for part of the image. Figure 6 shows a few examples of partial visibility (more than half of the image in not visible) to which our system is robust.

Speed change: The sequences are accelerated or slowed down by up to $\pm 20\%$. This has an effect on ⑦: for distantly matched frames, the δt values are different, and may vote for different bins in the δt histogram. A solution is to compute a 2D histogram $(\delta t, f)$ which additionally estimates the speedup factor f like in [1]. However, we found this unnecessary as the histogram bins (t_{bin} in Section III-B) are large enough with respect to the specified length of the sub-videos.

	KEYSADVES	STRICT	SOFT
number of indexed frames	95,411	2,080,446	
number of indexed descriptors	39,112,273	874,697,777	
shortlist length in ⑥	500	500	1500
keep top-ranked video only	no	yes	no

TABLE II
PARAMETERS OF OUR RUNS.

Flip: Our image matching approach is not robust to flipping (or any affine transform with a negative determinant), as the image features ② are not invariant to this transformation. To cope with flipping, both the query video and its flipped version are used to search the database. The results of the query and of the flipped query are merged in ⑦. Interestingly, video sequences and their flipped version often appear close together in the shortlist, presumably because typical scenes contain numerous symmetric objects.

Picture-in-picture: This transform is especially difficult to handle in combination with small-scale attacks (such as blur), because only few interest points detected at large scales in the initial video are stable. If a significant scale change is combined with a cluttered background video, the few robust points are outnumbered by the clutter points.

To address this issue, we have used a second database of half-sized videos and performed all queries in both databases (normal-sized and half-sized). Note that this adds a complexity and memory overhead of "only" 25% in ⑥, as the second database contains significantly less interest points.

Conclusions: Our frame matching approach was able to handle most of the transformations without requiring any adaptation to TRECVID. Only picture-in-picture and flip have been handled by performing additional computations (four queries to handle all combinations of flip and half-size) in steps ①-⑥. Note that they did not require a modification of the matching approach, i.e., it was not necessary to implement any explicit detection or adaptation for specific transformations.

C. TRECVID *copy detection results*

In this section we present the results obtained in the TRECVID 2008 copy detection challenge. The task was to perform 2000 queries (of 3 seconds to 1 minute) in a 200-hour video dataset. The query videos were produced from the dataset in a similar way as in Subsection IV-A. There were 10 transformations, for some of which results are shown in Figure 7.

We have submitted three different runs, see Table II. The run `KEYSADVES` uses only the keyframes of the videos instead of uniform sampling. The runs `STRICT` and `SOFT` sample uniformly 1 out of 10 frames. `STRICT` and `SOFT` have different precision-recall tradeoffs: `STRICT` returns only the result with the highest confidence and `SOFT` returns more results, some of which having low confidence scores. For all the runs, a uniform sampling rate of 1 out of 10 frames was used for querying. SIFT was used as local descriptor.

NDCR: The official detection accuracy measure of the copy-right detection task is the Normalized Detection Cost Ratio (NDCR)[21]. This measure is integrates the cost of missing a true positive and the cost of having to deal with false positives. Here, a result is considered a positive if there is an overlap between the returned subsequence and the ground truth subsequence, irrespective of how long the intersection is. The optimal cost threshold, i.e., the one minimizing this cost, is computed for each transformation. With the parameters used for the evaluation, the cost of false positives was much higher than that of missing a true positive. This explains why our run `STRICT` obtains better results than our run `SOFT` for all transformations.

The left part of table III gives the NDCR scores for our three runs, the two best scores among all other participants and the median of all runs. Note that the change in contrast, referred to by T5, is clearly an easy transformation, as two participants have obtained perfect results, in particular our run `STRICT`. This table shows the excellent performance of our approach: our run `STRICT` obtained the best results for all the transformations in terms of the NDCR measure.

Precision-Recall: The precision-recall curves are a standard way of measuring the performance of an information retrieval system. We have generated these curves for the most difficult transformations. Figure 7 gives, for these transformations, the precision-recall curves associated with the 5 best runs among all participants.

Localization accuracy: Localization accuracy was measured by the F1 measure. F1 is defined as the harmonic mean of precision and recall, with precision and recall obtained for the optimal threshold resulting from the NDCR measure computation. The time-based precision and recall are defined in Subsection IV-A.

This definition depends on the optimal decision threshold, and makes it impossible to compare the values of different runs as they include different videos. Indeed, the best runs in terms of the NDCR measure are penalized when computing the F1 measure because the most difficult queries are included into the score estimation. Nevertheless, it still provides a good indicator of the localization accuracy of a system. Results are presented in the right part of Table III. We can observe that a high sampling rate is important to obtain good results, i.e., our runs `STRICT` and `SOFT` are much better that our run `KEYSADVES`.

Conclusion: Our video copy detection system outperforms other submitted results on all transformations. This is due to a very accurate image-level matching. Run `KEYSADVES`, which is more scalable, shows that our system still obtains excellent results with a memory footprint and query time reduced 20 times.

ACKNOWLEDGMENTS

This work was partially funded by the ANR project RAFFUT, the GRAVIT project as well as the QUAERO project. We also thank ADVESTIGO and IRIM for their support.

REFERENCES

[1] 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 *Conference on Image and Video Retrieval*, 2007.

[2] A. Joly, C. Frélicot, and O. Buisson, "Content-based video copy detection in large databases: A local fingerprints statistical similarity search approach," in *International Conference on Image Processing*, 2005.

	NDCR (lower = better)						F1 (higher = better)					
	KEYADVES	STRICT	SOFT	BEST1	BEST2	MEDIAN	KEYADVES	STRICT	SOFT	BEST1	BEST2	MEDIAN
T1	0.328	**0.079**	0.126	0.363	0.385	0.763	0.672	0.948	0.928	**0.988**	0.869	0.657
T2	0.255	**0.015**	0.046	0.148	0.160	0.935	0.684	0.952	0.933	**0.990**	0.863	0.471
T3	0.220	**0.015**	0.015	0.076	0.087	0.567	0.667	0.950	0.918	**0.989**	0.906	0.758
T4	0.206	**0.023**	0.038	0.095	0.095	0.556	0.692	0.946	0.921	**0.987**	0.939	0.743
T5	0.213	**0.000**	0.012	**0.000**	0.027	0.350	0.672	0.949	0.916	**0.989**	0.936	0.774
T6	0.290	**0.038**	0.069	0.192	0.219	0.571	0.698	0.950	0.922	**0.992**	0.905	0.729
T7	0.317	**0.065**	0.115	0.436	0.498	0.810	0.701	0.941	0.914	**0.993**	0.863	0.698
T8	0.258	**0.045**	**0.045**	0.076	0.077	0.763	0.676	0.950	0.918	**0.991**	0.886	0.691
T9	0.266	**0.038**	0.080	0.173	0.176	0.951	0.681	0.951	0.929	**0.986**	0.860	0.552
T10	0.406	**0.201**	0.246	0.558	0.614	0.903	0.699	0.946	0.923	**0.864**	0.842	0.658

TABLE III
NDCR AND F1 MEASURES FOR OUR THREE RUNS (KEYSADVES, STRICT AND SOFT). THE VALUES GIVEN FOR BEST1 AND BEST2 ARE THE BEST ONES OBTAINED BY ALL OTHER PARTICIPANTS FOR EACH TRANSFORMATION. SIMILARLY, THE COLUMN MEDIAN INDICATES THE MEDIAN VALUE OF ALL PARTICIPANTS.

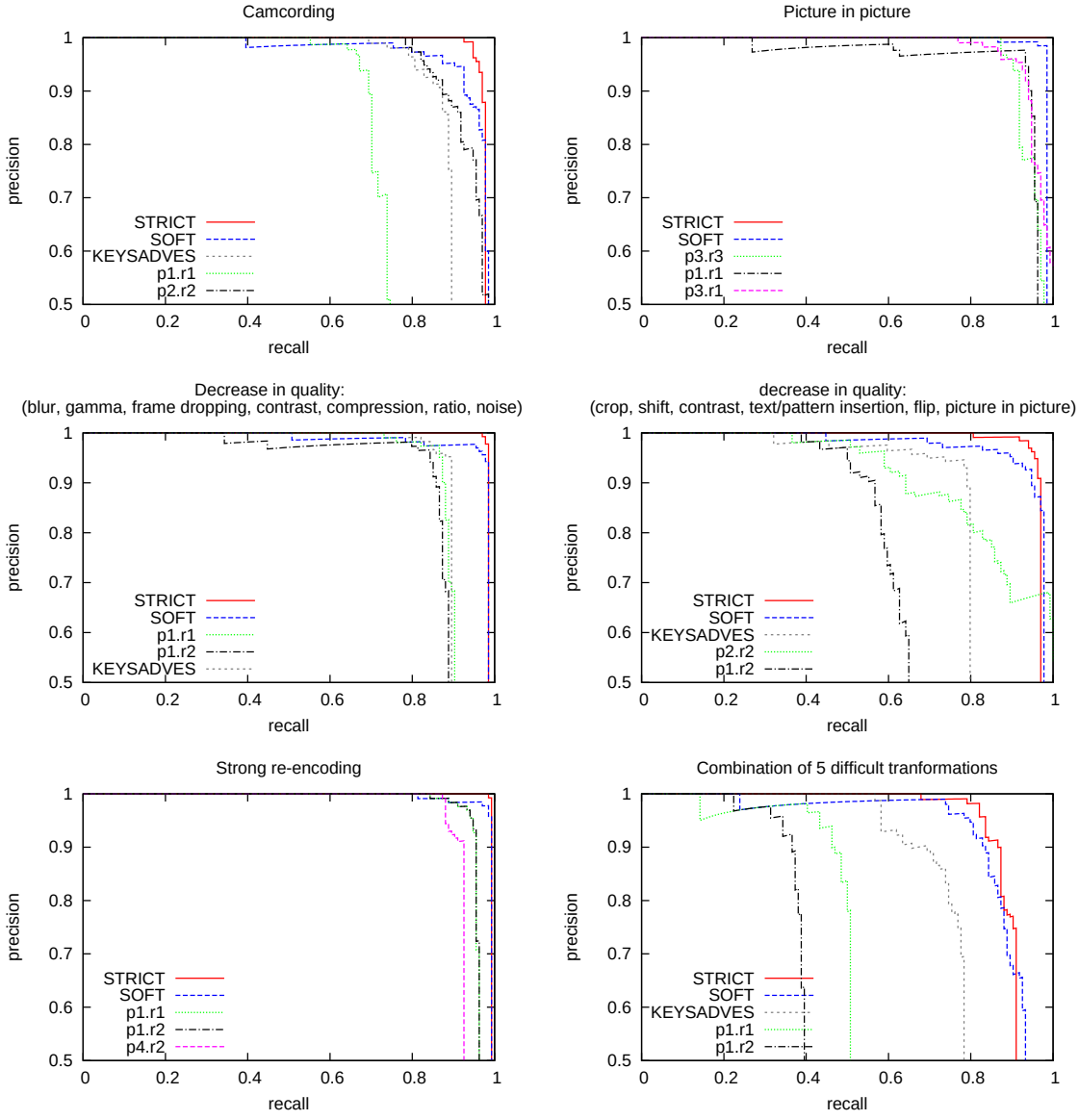


Fig. 7. Trecvid: precision-recall curves for the 5 best runs (based on the NDCR criterion) on 6 transformations. Runs from other participants are anonymized for legal reasons: the name "px.ry" corresponds to run y of participant x.

[3] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," *Transactions on multimedia*, vol. 9, pp. 293–306, 2007.

[4] X. Naturel and P. Gros, "Detecting repeats for video structuring," *Multimedia Tools and Applications*, vol. 38, no. 2, pp. 233–252, 2008.

[5] R. Lienhart and I. Döhring, "Mining TV broadcasts for recurring video sequences," in *Conference on Image and Video Retrieval*, 2009.

[6] 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.

[7] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] M. Heikkila, M. Pietikainen, and C. Schmid, "Description of interest regions with local binary patterns," *Pattern Recognition*, vol. 42, no. 3, pp. 425–436, 2009.

[9] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *International Conference on Computer Vision*, 2003.

[10] J. Sivic and A. Zisserman, "Efficient visual search for objects in videos," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 548–566, 2008.

[11] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *European Conference on Computer Vision*, 2008.

[12] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large databases for recognition," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[13] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems*, 2008.

[14] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Conference on Computer Vision and Pattern Recognition*, 2007.

[16] H. Jégou, M. Douze, and C. Schmid, "On the burstiness of visual elements," in *Conference on Computer Vision and Pattern Recognition*, 2009.

[17] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVID," in *ACM International Workshop on Multimedia Information Retrieval*, 2006.

[18] A. Saoudi and H. Essafi, "Spatio-temporal video slice edges analysis for shot transition detection and classification," *World Academy of Science, Engineering and Technology*, vol. 28, 2007.

[19] K. Mikolajczyk, "Binaries for affine covariant region descriptors." http://www.robots.ox.ac.uk/~vgg/research/affine/.

[20] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 77–116, 1998.

[21] W. Kraaij, P. Over, J. Fiscus, and A. Joly, "Final CBCD evaluation plan TRECVID 2008 (v1.3)," June 2008.