INRIA-LEARs video copy detection system

Matthijs Douze, Adrien Gaidon, Hervé Jégou, Marcin Marszalek, Cordelia Schmid

INRIA-LEAR'S VIDEO COPY DETECTION SYSTEM

Matthijs Douze[1]*, Adrien Gaidon*[2]*, Herve Jegou*[1]*, Marcin Marszałek*[2] *and Cordelia Schmid*[1,2]

INRIA LEAR, LJK
Grenoble, France

[1]*Copyright detection task* [2]*High-level feature extraction task*

1. INTRODUCTION

A video copy detection system is a content-based search engine [1]. It aims at deciding whether a query video segment is a copy of a video from the indexed dataset or not. A copy may be distorted in various ways. If the system finds a matching video segment, it returns the name of the database video and the time stamp where the query was copied from.

Fig. 1 illustrates the video copyright detection system we have developed for the TRECVID 2008 evaluation campaign. The components of this system are detailed in Section 2. Most of them are derived from the state-of-the-art image search engine introduced in [2]. It builds upon the bag-of-features image search system proposed in [3], and provides a more precise representation by adding 1) a Hamming embedding and 2) weak geometric consistency constraints. The HE provides binary signatures that refine the visual word based matching. WGC filters matching descriptors that are not consistent in terms of angle and scale. HE and WGC are integrated within an inverted file and are efficiently exploited for all indexed frames, even for a very large dataset. In our best runs, we have indexed 2 million keyframes, represented by 800 million local descriptors.

We give some conclusions drawn from our experiments in Section 3. Finally, in section 4 we briefly present our run for the high-level feature detection task.

2. COPY DETECTION SYSTEM

An overview of the system we have designed for this task is given Fig. 1. Each step is identified by a circled number. The feature extraction part ②-⑤ is illustrated Fig. 2. The components of this system are detailed hereafter.

2.1. Frame extraction ①

The first step of our system extracts frames from videos. Two frame extraction methods have been used:

- *Uniform subsampling:* a fixed number of frames per time unit is extracted. Using this method, we have extracted 2.5 frames per second, i.e., one frame out of 10.

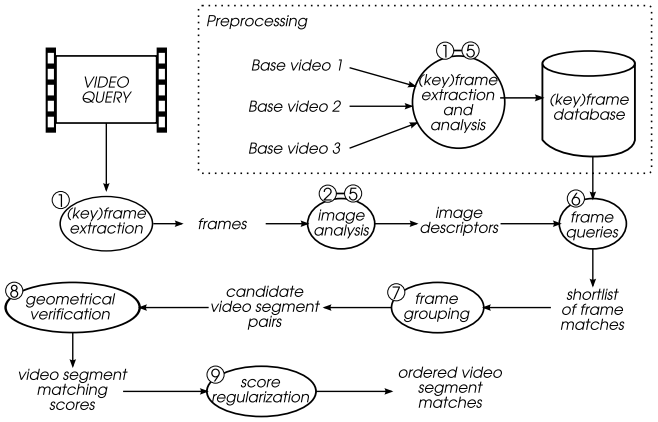


Fig. 1. Overview of our copyright detection system

  This is the method we have used in our runs `STRICT` and `SOFT`.

- *Stable keyframes:* Here, we extract only a few representative keyframes per shot. The main advantage of this method is to produce a limited number of frames (1 frame every 6 s on average).

In our preliminary experiments, we observed that the stable keyframe selection caused an insufficient number of frame matches. Therefore, for our `KEYSADVES` run, we used an *asymmetric sampling strategy*:

- stable keyframe were extracted on the dataset side, producing a relatively small set of frames.

- the query frames were extracted using uniform subsampling.

2.2. Features ②–③

The core of our image system is based on local invariant descriptors. Such a description is able to match small parts of

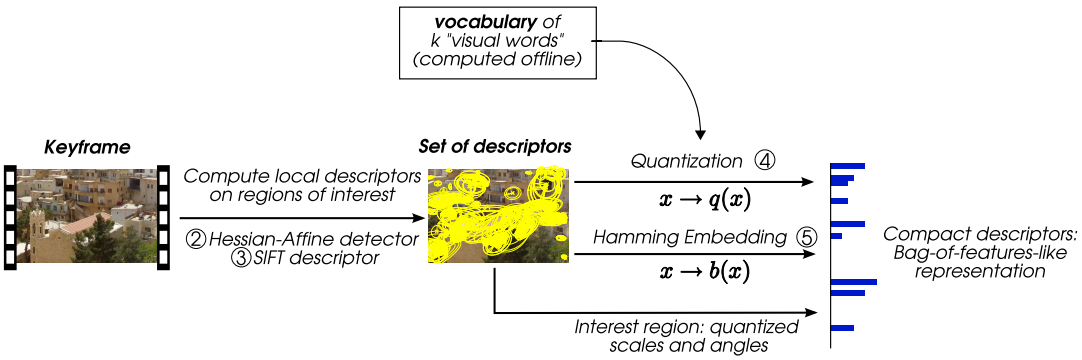


Fig. 2. Feature extraction from the frames: descriptor generation and conversion to a compact representation.

video frames, which is necessary to handle the picture-in-picture transformation. It is also appropriate to handle difficult global transformations, such as camcording, pattern insertion or combined transformations.

We have extracted only one type of features from the video. This extraction is performed in two steps: detecting regions of interest and computing descriptors on those.

② **Detector:** We used the Hessian-Affine region extractor of [4], using the software of [5] with the default parameters. This region extractor is invariant to several image transformations:

- *Scale invariance:* The Hessian interest point detector is invariant to scale changes. It is used together with automatic scale selection, as first proposed in [6].
- *Orientation invariance:* It is obtained by computing the dominant gradient orientation. The extracted regions are rotated such that this dominant orientation is aligned with a common direction.
- *Affine invariance:* This one is obtained using the iterative algorithm proposed in [4].

The affine invariance provided by the Hessian-Affine detector is obviously of interest for the camcording transformation. It is also useful for any type of transformations, because the frame extraction may not extract exactly the same frames on corresponding videos. In this case, the affine normalization allows the system to match frames from the same shot even if the camera position is not exactly the same, or if the scene or the objects have moved between the considered frames.

③ **Descriptor:** We use the SIFT descriptor [7], which produces 128-dimensional vectors. The description has been generated using the implementation [5] with default parameters.

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

Our core image system builds upon the state-of-the-art image search that we proposed in [2]. This paper improves the so-called "Video-Google" bag-of-features image search system first introduced by Sivic and Zisserman [3]. The key steps of our system are detailed below.

Visual codebook generation (off-line): The quantizer is a partitioning of the space of descriptors. It is defined by a set of centroids. Each centroid is called a "visual word", belonging to a "visual vocabulary". Our visual vocabulary has been generated using the k-means algorithm learned on a subset of descriptors from the video database. We have used $k = 200000$ visual words in all our experiments and runs.

④ **Assigning the descriptors to visual words:** Each SIFT descriptor of a given frame is assigned to the closest visual word. This quantization step amounts to representing a descriptor by the corresponding centroid index $q(x)$. On the query side, instead of choosing only the nearest neighbor, each descriptor is assigned to *several* nearest visual words. This strategy is similar to the multiple descriptor assignment proposed in [8], except that we perform multiple assignment for the query only, not for the indexed video dataset. This limits the memory usage of the frame indexing structure.

⑤ **Hamming Embedding:** At this point, a given descriptor x is represented by the corresponding quantization cell $q(x)$. Because of the high dimensionality of the descriptors, comparing them with the cell index is not precise enough: although around 99.9995% are filtered out, quite different descriptors can still match.

To address this problem, we have used the Hamming Embedding method proposed in [2]. The key idea is to represent a descriptor by both the index $q(x)$ and a binary signature $b(x)$ of length 64, where $b(.)$ is the Hamming Embedding function associated with the visual word $q(x)$. It is designed such that the Hamming distance

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

between two descriptors x and y lying in the same cell reflects the Euclidean distance $d(x, y)$. A descriptor is now represented by $q(x)$ and $b(x)$. The descriptor matching function f_{HE} is then defined as

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

where $h(.,.)$ is the Hamming distance defined in Eqn. 1, $h_t = 22$ is a fixed Hamming threshold and $w(.,.)$ is a soft weighting function that gives higher scores to smaller Hamming distances. Using the threshold h_t on non matching images, only one descriptor out of 15 millions is considered a match (98.8% of the cell's descriptors are filtered out by the binary signature check).

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

$$s_j^* = \frac{1}{\sqrt{m_j\, m'}} \sum_{i'=1..m'} \sum_{i=1..m_j} f_{\text{HE}}\left(x_{i,j}, y_{i'}\right), \tag{3}$$

where m_j is the number of descriptors of the frame j.

⑥ **Inverted file:** In order to compute the score of Eqn. 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 [3]. This structure is composed of k lists of descriptor entries, each corresponding to a visual word. This greatly reduces the complexity, because only the descriptors assigned to the same quantizer centroid as the query descriptor are checked.

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

- the image id ;
- the binary signature $b(x)$;
- the quantized dominant orientation $qa(x)$;
- the quantized scale $qs(x)$.

The resulting structure is illustrated Fig. 3. The memory usage 12 bytes per local descriptor, see [2] for details. Note that, for a given query descriptor x, the set of entries associated with the corresponding visual word $q(x)$ is analyzed. According to Eqn. 2, only the dataset descriptors that are consistent in terms of the binary signature $b(x)$ will produce a vote. In addition to the filtering steps based on $q(x)$ and $b(x)$, the difference of orientations and log-scales are estimated for each frame. This is done in order to use WGC [2], i.e., geometrical information is used for all descriptors for further filtering.

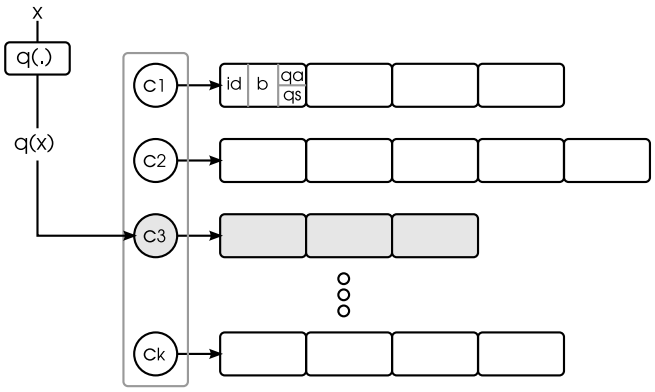


Fig. 3. Modified inverted file structure.

2.4. Frame grouping ⑦

At this point, we have a set of matching frames between the query video and database video sequences. Compared to the voting functions used in [1], we estimate the temporal and geometrical parameters separately. We compute a Hough transform to estimate them.

Each frame match indicates a potential resulting video b (the one from which it has been extracted), a time shift δt that aligns the query with this database video, and also a score s computed from the inverted file. This gives a weighted vote $(b, \delta t, s)$ for video b and time shift δ_t. These votes are accumulated in a 2D Hough histogram in $(b, \delta t)$.

We make a shortlist of 400 $(b, \delta t)$ hypotheses from the largest bins of the Hough histogram, and collect the frame matches that vote for each hypothesis. If some frame matches in a group are too far apart in time (more than 1 minute), the group is split.

2.5. Geometrical verification ⑧

This stage aims at re-scoring the video segment matches using more geometrical information, i.e. using the positions of the interest points, in the spirit of the re-ranking stage of [9].

We assume that the geometrical transform between the query and the video of the database is approximately constant in time, similar to [1]. This requires, for example, that the video camera does not move while camcording. We hence estimate the transform directly from the descriptor matches. This is more robust than matching two images, as point matches are accumulated over several frames.

We use Lowe's method [7] to estimate a 2D affine transform between the video segments:

- approximate similarity hypotheses are generated using a 4D Hough space, i.e. using a similarity transformation model ;
- a full affine transform is estimated from the point

matches that "agree" with each similarity transformation;

- the best hypothesis, i.e. the one that corresponds to most of the matches, is retained.

We introduce priors in the hypothesis ranking stage: in the case of TRECVID the scale factor is between 1 and 2 and that there is no big rotation or skew, etc.

2.6. Score aggregation strategy ⑨

As an output of the previous stage, we have obtained a set of segments and a score for each of them. This score corresponds to the total number of inliers (matching descriptors) for all matching frames normalized by the duration of the segment. We observed on our validation set that this score was quite different from one query to another.

To address this problem, we have used a frame score normalization procedure. The objective of this step is to reduce the contribution of the query frames that receive high scores for several videos from the dataset. Therefore, we first compute the sum t_f of all scores associated with a given query frame. We then update the score s_f associated with a given match as

$$s_f := s_f \times \left(\frac{s_f}{t_f} \right)^2 . \quad (4)$$

Hence, if a query frame votes for only one dataset video frame, then $s_f = t_f$ and the score s_f is not modified. Conversely, if a frame receives many votes of similar strength for different videos, the impact of this frame on the final score is greatly reduced.

A video segment score s_v is then obtained by summing its frame scores and by dividing it by the number of query video frames. This score is finally updated using a strategy similar to Eqn. 4, as

$$s_v := s_v \times \left(\frac{s_v}{m_v} \right)^2 , \quad (5)$$

where m_v is the highest score obtained among all the video segments. This final score update penalizes the segments which are not ranked first. This reduces the number of false positives when a decision threshold is used, as done when using NDCR measure, see Section 3.

3. EXPERIMENTS

3.1. Validation datasets

As required by the evaluation procedure [10], we have not evaluated our system on the final test videos, nor merged the results across transformations. Because the validation set provided by TRECVID was too small and not difficult enough, we have used two other datasets to improve the feedback obtained when measuring the system accuracy.

Image dataset: We have used our own INRIA Holidays dataset [11] to improve our core image system. This dataset mainly contains personal Holiday photos. The remaining ones were taken on purpose to test the robustness to various transformations: rotations, viewpoint and illumination changes, blurring, etc. The dataset includes a very large variety of scene types (natural, man-made, water and fire effects, etc).

Video dataset: We have created a video validation set. We have implemented a video transformation tool based the transformations specified for the copyright evaluation task[1]. We have adjusted the transformation parameters to obtain difficult transformations.

- to design the temporal and geometrical verification used in the re-ranking stage ;

- to adjust the a scoring strategy to produce scores which are consistent across queries.

3.2. Handling of specific transforms

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

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 SIFT descriptor is invariant to this change, as it represents an approximately linear change in gray-levels.

Blur, blocks, re-encoding, noise: We observed that, when evaluated individually, such transformations do not disturb the image core system (which often performs better than an untrained human, see Fig. 4d). 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, crop: Camcording and partial occlusion are relatively easy versions of the changes in viewing conditions the image core system is able to handle, so there was no specific development required. We tightened the prior on the 2D transform estimated in ⑧ to allow only for small rotations and skews. Local descriptors handle occlusions and crops, as they remain unchanged for part of the image even if the rest changes.

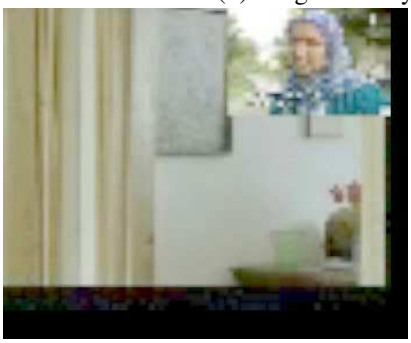
Fig. 4a) shows an example for an occluding pattern of more 1/2 of the image size to which our system is robust.

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

(a) large occlusion

(b) images with symmetric elements

(c) cluttered background



(d) hardly visible match (on the right: the base frame)

Fig. 4. Example frames from transformed videos.

Speed change: The sequences can be accelerated or slowed down by up to $\pm 20\%$. This has an effect on ⑦: for distant matched frames, the δt values are different, and may vote for different bins in the $(b, \delta t)$ histogram. A solution is to compute a 3D histogram $(b, \delta t, f)$ which additionally estimates the speedup factor f like in [1]. However, we found this unnecessary as the histogram bins in δt are large enough with respect to the specified length of the sub-videos.

Flip: The image core system is not robust to flipping (or any affine transform with a negative determinant). Indeed the affine and orientation normalizations of ② are not invariant this transformation. We handle this by querying the flipped video sequence. 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 (Fig. 4b).

Picture-in-picture: This transform is especially difficult to handle in combination with small-scale attacks (such as blur) where only a few very large interest points of the initial video are stable.

However, if a significant scale change is combined with a cluttered background video (Fig. 4c), the few robust points are outnumbered by the clutter points.

To address this issue, we have maintained a second database of half-sized videos and perform all queries in both bases (normal-sized and half-sized). Note that this adds an overhead of "only" 25% in ⑥, as the second base contains many less interest points.

Conclusions: We have not used any explicit detection of specific transforms, mainly due to time constraints. However, our core image system was able to handle most of these transformations without modification. The others (picture-in-picture, flip) have been handled by specifically performing separate computations (four queries to handle all combinations of flip and half-size) in steps ①-⑥.

For most videos, applying the entire chain with uniform frame sampling is an overkill. In these cases, the parameters used in our run `KEYSADVES` are sufficient.

The shortlist sizes of the output of steps ⑥ and ⑦ are important parameters. True positives may be lost by pruning too many hypotheses, but keeping too many of them disturbs the less robust steps that use them as input.

3.3. Results

Table 1 shows the differences between our three runs. Note that these runs only differ in the parameters used, except the run `STRICT` for which we have kept at most one video per query.

	KEYADVES	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 1. Parameters of our runs.

	KEYADVES	STRICT	SOFT	BEST1	BEST2	MEDIAN
T1	0.328	0.079	0.126	0.363	0.385	0.763
T2	0.255	0.015	0.046	0.148	0.160	0.935
T3	0.220	0.015	0.015	0.076	0.087	0.567
T4	0.206	0.023	0.038	0.095	0.095	0.556
T5	0.213	0.000	0.012	0.000	0.027	0.350
T6	0.290	0.038	0.069	0.192	0.219	0.571
T7	0.317	0.065	0.115	0.436	0.498	0.810
T8	0.258	0.045	0.045	0.076	0.077	0.763
T9	0.266	0.038	0.080	0.173	0.176	0.951
T10	0.406	0.201	0.246	0.558	0.614	0.903

Table 2. NDCR measures for our three runs (KEYSADVES, STRICT and SOFT). Lower values correspond to better scores. 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 NDCR value of all participants.

NDCR: The official detection accuracy measure of the copyright detection task is the Normalized Detection Cost Ratio (NDCR)[2]. This measure is a trade-off between the cost of missing a true positive and the cost of having to deal with false positives. The optimal cost threshold, i.e. the one minimizing this cost, is computed for each transformation. NDCR=0 indicates perfect results for the transformation considered. 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.

Table 2 gives the NDCR scores for our three runs, the two best scores among all other participants and the median run. Note that the change in contrast, referred to by T5, is clearly a quite easy transformation, as two participants have obtained perfect results (Our run STRICT and the run IMEDIA-Fusion). This table shows the relevance of our approach: our run STRICT obtain the best results for all the transformations.

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 representative transformations, mainly those where the participants have obtained the lowest values of NDCR. Fig. 5 gives, for these transformations, the precision-recall curves associated with the 5 best runs among all participants.

Localization accuracy: The accuracy of the localization 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.

This definition depends on the optimal decision threshold, and makes 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 most difficult queries are included into the score estimation. Nevertheless, it still provides a good indicator of the localization accuracy of a system.

Table 3 shows 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.

	KEYADVES	STRICT	SOFT	BEST1	BEST2	MEDIAN
T1	0.672	0.948	0.928	0.988	0.869	0.657
T2	0.684	0.952	0.933	0.990	0.863	0.471
T3	0.667	0.950	0.918	0.989	0.906	0.758
T4	0.692	0.946	0.921	0.987	0.939	0.743
T5	0.672	0.949	0.916	0.989	0.936	0.774
T6	0.698	0.950	0.922	0.992	0.905	0.729
T7	0.701	0.941	0.914	0.993	0.863	0.698
T8	0.676	0.950	0.918	0.991	0.886	0.691
T9	0.681	0.951	0.929	0.986	0.860	0.552
T10	0.699	0.946	0.923	0.864	0.842	0.658

Table 3. F1 measures for our three runs (KEYSADVES, STRICT and SOFT). Higher values correspond to better scores. The values given for BEST1 and BEST2 are the best obtained by all other participants for each transformation. Similarly, the column MEDIAN indicates the median F1 value of all participants.

4. HIGH LEVEL FEATURE EXTRACTION

We have submitted one method called **LEAR_basic_1** which classifies keyframe based on the approach of Zhang *et al.* [12]. We do not use temporal information or audio.

Our approach first extracts several image representations which differ in the image description and the spatial partitioning of the image (cf. section 4.1). These representations are then combined within a one-against-rest non-linear Support Vector Machine [13] as described in section 4.2.

We only use the publicly available keyframes (one per subshot) and their annotations resulting from the collaborative annotation conducted by Georges Quenot *et al.* [14]. We do not sample the shots for more frames.

4.1. Image representation

We densely sample the images using a multi-scale grid and use Harris-Laplace [4], Hessian, Harris-Harris and Laplacian [6] interest points detectors. We use the SIFT [7] descriptor to describe local interest regions.

[2]See http://www-nlpir.nist.gov/projects/tv2008/Evaluation-cbcd-v1.3.htm#eval

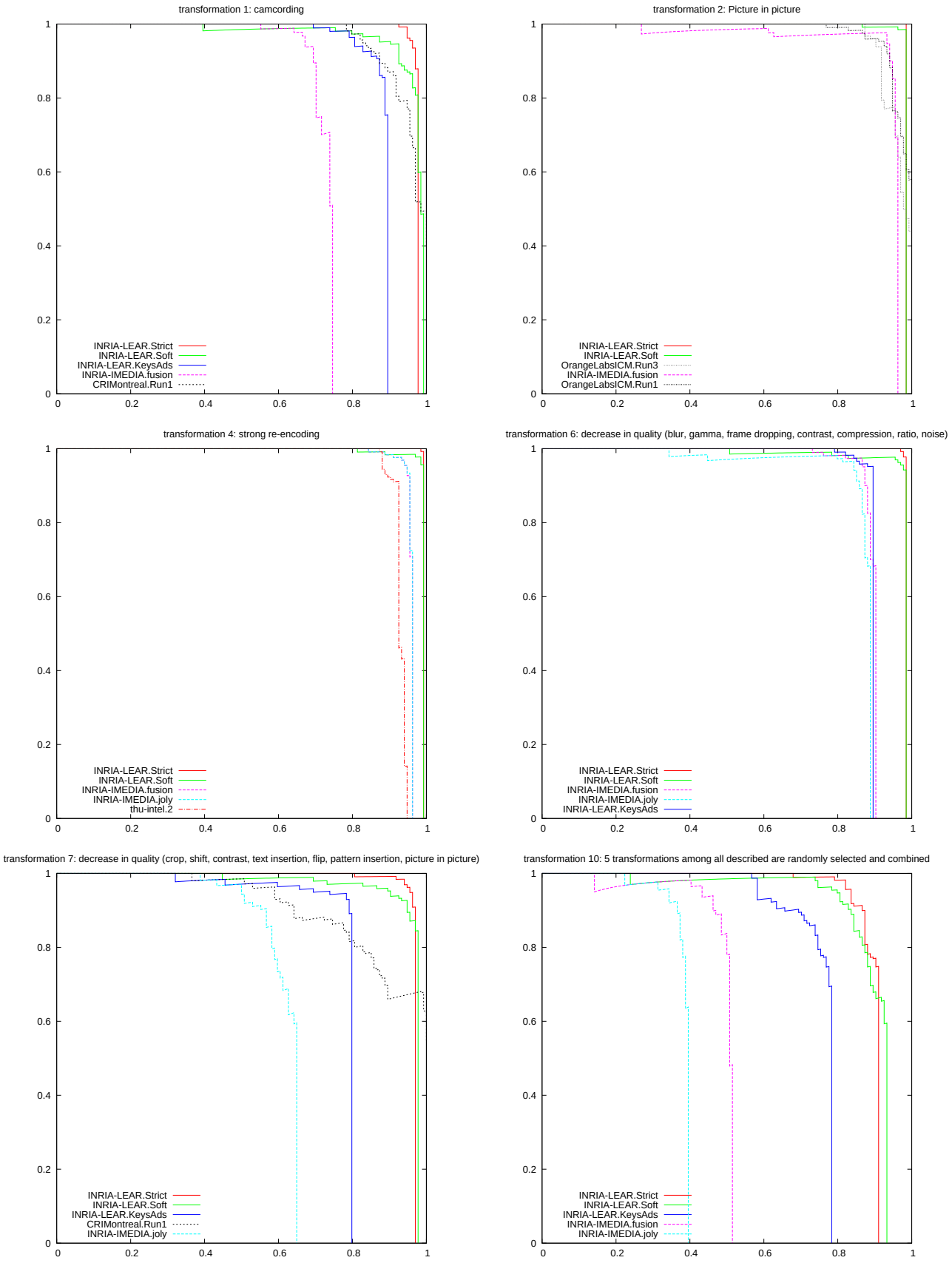


Fig. 5. Precision-recall curves for the 5 best runs for the NDCR measure

Given the descriptors extracted on the training data, we construct a visual vocabulary [15] in an unsupervised fashion with k-means clustering. For our experimental results, we set $k = 4000$. We then use these clusters (or visual words) to quantize all the descriptors of a given frame and count the occurrences of each visual word in this image. We finally obtain a frame representation based on a histogram of visual word occurrences. This representation is called a bag-of-words.

We use weak spatial information by sub-dividing the image into 1x1, 2x2 and horizontal 3x1 grids and appending the histogram of each grid cell into a single representation similar in spirit to the spatial pyramid of Lazebnik *et al.* [16]. Note that the 1x1 grid results in a standard bag-of-features representation.

To summarize, we use five different methods to select regions of interest, one descriptor and three spatial grids, as described above. This results in a total of 15 image representations, also called channels.

4.2. Classifier

Classification is performed with a non-linear Support Vector Machine [13] and takes as input the set of channels. We use a multi-channel extended Gaussian kernel :

$$K(X,Y) = \exp\left(-\sum_{ch} \gamma_{ch} D_{ch}(X,Y)\right)$$

where $D_{ch}(X,Y)$ is a similarity measure for channel ch. We use the χ^2 distance to measure the similarity between two bag-of-words $X = (x_1, \cdots, x_n)$ and $Y = (y_1, \cdots, y_n)$:

$$D(X,Y) = \frac{1}{2}\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}$$

For our **LEAR_basic_1** submission we follow the kernel design of Zhang *et al.* [12]: $\gamma_{ch} = \frac{1}{average(D_{ch})}$. We also fix the C parameter to the value suggested in their paper.

The multi-class problem is addressed in a one-against-rest set-up. When there are multiple keyframes per shot (one per subshot), we simply consider the most confident of the decisions to be the confidence value for the entire shot.

4.3. Results

Over the 20 high level features we obtained an average precision of 9.6% (the best is 16.7%). We correctly retrieved 1403 true shots out of 4607. Our run is on position 25 given more than 150 submissions.

These results are encouraging, given that we can increase the performance by considering more frames per shot, by using audio information and by adding channels with color information. Our method shows that state of the art keyframe classifiers can compete with other techniques more specifically engineered for video processing.

Acknowledgments: We would like to acknowledge the ANR project RAFFUT, GRAVIT and QUAERO for their financial support. Thanks also to ADVESTIGO and IRIM for their help.

5. 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 *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*. New York, NY, USA: ACM, 2007, pp. 371–378.

[2] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *ECCV*, Oct 2008. [Online]. Available: http://lear.inrialpes.fr/pubs/2008/JDS08

[3] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, 2003, pp. 1470–1477.

[4] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *IJCV*, vol. 60, no. 1, pp. 63–86, 2004.

[5] K. Mikolajczyk, "Binaries for affine covariant region descriptors," http://www.robots.ox.ac.uk/~vgg/research/affine/, 2007.

[6] T. Lindeberg, "Feature detection with automatic scale selection," *IJCV*, vol. 30, no. 2, pp. 77–116, 1998.

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

[8] H. Jegou, H. Harzallah, and C. Schmid, "A contextual dissimilarity measure for accurate and efficient image search," in *CVPR*, 2007.

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

[10] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. New York, NY, USA: ACM Press, 2006, pp. 321–330.

[11] H. Jegou and M. Douze, "INRIA Holidays dataset," http://lear.inrialpes.fr/people/jegou/data.php, 2008.

[12] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *IJCV*, vol. 73, pp. 213–238, Jun 2007.

[13] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, MA, 2002.

[14] S. Ayache and G. Qunot, "Video corpus annotation using active learning," in *European Conference on Information Retrieval*, 2008, pp. 187–198.

[15] J. Willamowski, D. Arregui, G. Csurka, C. R. Dance, and L. Fan, "Categorizing nine visual classes using local appearance descriptors," in *IWLAVS*, 2004.

[16] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.