VIVA lab - University of Ottawa at TRECVID 2009
Content Based Copy Detection

Gerhard Roth, Robert Laganière, Martin Bouchard, Ilias Lakhmiri, and Tarik Janati
gerhardroth@rogers.com, laganier,bouchard@site.uottawa.ca
University of Ottawa, School of Information Technology and Engineering
Ottawa, K1N 6N5, Canada

Abstract

1. *Briefly, what approach or combination of approaches did you test in each of your submitted runs?*

   - VIVAlab-uOttawa.v The video search approach is based on a method that finds keypoints in each video frame, and then uses a descriptor of keypoint counts in 16 (4 by 4) equally sized regions of the images.
   - VIVAlab-uOttawa.a The audio-only copy detection search scheme was based on the computation of a coherence function, using intermediate features in the ITU-R BS.1387 Perceptual Evaluation of Audio Quality (PEAQ) standard.
   - VIVAlab-uOttawa.m The combined audio-video used the fact that audio was detected in a database segment simply to boost the video scores.

2. *What if any significant differences (in terms of what measures) did you find among the runs?*
   The video only approach performs well in terms of the mean F1 measure, which is a consequence of the construction of the algorithm. Indeed, because a frame is represented with only 16 bytes, it was possible to compare each frame of the query with each frame of the video database. As a result, our algorithm is one of the best in terms of accuracy. The balanced profile performs at almost exactly the median in terms of NDCR, while the NOFA profile is slightly worse than the median performance. The audio only approach is slightly worse than the median in terms of NDCR. The F1 results are not significant in this case because of the very low number of positive detection that were submitted. The audio detection method has a very low false positive detection rate but, unfortunately a very high false negative too. The combined results were lower than what we expected; we still have to analyze the root cause of this performance.

3. *Based on the results, can you estimate the relative contribution of each component of your system/approach to its effectiveness?*
   Currently the video analysis seems to be doing most of the work, but we do not believe that our combination of the audio and video search methods is optimal.

4. *Overall, what did you learn about runs/approaches and the research question(s) that motivated them?*
   With so many different possible solutions to copy detection deciding which is the best is a function of the goals. Our goal is to have an approach that provides good performance, is easy to parallelize, works quickly in search mode, and has low storage requirements per video frame of the database. We believe that we have achieved this goals, but we have not yet found a satisfactory way of combining the audio video results.

1. Introduction

Duplicate video detection is an important and new topic that provides an alternative to watermarking for copyright control and other applications. The TRECVID video copy detection task provides a way to compare and evaluate the performance of different solutions to this problem. While the performance evaluation of TRECVID is important [8], by its nature copyright detection is an underconstrained problem, and there are many possible solutions that have similar performance. To discriminate between these solutions we believe that it is necessary to add some evaluation criteria that are not related simply to performance. Below we list some features that we believe should be part of a successful copy detection system.

First, it is clear that all systems will execute an indexing stage to extract search information from the video, and then the search stage will use this information to do the matching to a given video query. Since the indexing stage is done off-line, it can be complex and time consuming. However, given the rapidly increasing amount of video data that is being created, the amount of index information saved per video database entry should be as small as possible. By contrast, the search process that tries to match the index information from a query to the database indexes should be as fast as possible. In practice a copy detection system will need to run on parallel hardware, such as multiple CPUs or even GPUs. Therefore the search process should also be easy to parallelize and it should be as simple as possible.

Keeping these extra criteria in mind, we will look at the current set of solutions to the problem of copy detection. In general, they can be divided into two groups, those that use global features and those that use local features. Typically global features, such as moments, or image statistics, can be extracted quickly and efficiently, and they provide compact index information [6, 7]. They are also efficient in terms of search, but they are not discriminatory enough to provide good performance. They also have problems dealing with certain transformations that are used in TRECVID. Given the success of local features such as SIFT in image matching and search, it is natural to attempt to use these features in video copy detection [5]. Their extraction is usually slow, which is not a problem, since this is done off-line. Local features are indeed discriminatory and perform well, but they have required a considerable amount of storage per image frame, since there is usually a high dimensional descriptor associated with local feature point. While they are tolerant to some types of transformations (such as gamma change—) they are not tolerant to global transformations (such as image flip).

2. Video-only Copy Detection Approach

Looking at the past solutions we have come to the conclusion that a combination of local and global features is the best way to attack the problem of copy detection. For this reason in each video frame in the database we extract a set of SURF feature points. The SURF algorithm is a modern scale invariant feature detector [3], similar to the well known SIFT features. The SURF features extracted from a single video frame are shown in Figure 1 (marked by a cross). Now as is usual for such local features, there is a high dimensional feature vector associated with each of these feature points (most often, 64 integers per interest point). We do not use these feature vectors since this would represent too much information to store, and a feature-based comparison across all frames would be prohibitive in terms of computational load. Instead our feature vector for a single frame is simply a count of the number of SURF points in each of sixteen (four by four) equal quadrants. Since there are rarely more than 255 SURF features in a quadrant, we store only sixteen bytes for each video frame (in the rare cases where more than 255 features are detected, we fix the value at 255). Experiments with different image partitioning showed that this sixteen byte descriptor is surprisingly discriminatory. One reason is the actual number of SURF feature points changes with the characteristics of the image, another is that when we search for a match we compare a set of image descriptors (i.e. over many frames). Thus we use a combination of local (SURF) and global (16 counts) features, which is both compact and discriminatory. This descriptor is also inherently invariant to resizing (up to a certain extent) and can be made invariant to flipping and mirroring just by changing the order of the count

vector. SURF features are also quite resistant to image deterioration and coding artefacts. For each video database entry there is an associated index file of size sixteen bytes times the number of frames. This is a very compact representation as Gigs of video files can be represented by just a few Mb of data. This video index file is created in a pre-processing phase at approximately 3 to 5 times the video frame rate.

2.1 Video Similarity Measurement

Now to actually compare a video query to a database entry we must compare their sequences of SURF feature counts. This is shown diagrammatically in Figure 2. First we build a cross table where we compute the normalized difference between every query frame, and database frame. If there are K query frames, and N video frames, then this takes $O(KN)$ time. However, we subsample the database frames (since typically $N >> K$), so the time to compute each normalized difference is very small. It is also clear that the process of creating this crosstable of scores can be easily parallelized.

To find the best match we first threshold each of these scores in the crosstable which transform this table into a binary form. We found that the exact value of this threshold is not too critical in the detection of similar video segment. Then we simply find the longest positive diagonal sequence in the crosstable, which is the longest video match. However, in order to take into account the dropped frames and possible outlier frames, we allow a certain number of negative entries in the indentified matching diagonal. This is done by skipping over a small number of frames in the diagonal sequence which do not pass the comparison threshold. In this way a small number of bad matches does not cause an otherwise long matching video sequence to be ignored.

The best match produces a length (number of matched frames) and an average difference for each overlapping frame (the normalized L1 difference). In some of our experiments we have used the match length as the decision criteria, but this does not deal with the fact that some matches have a very short match length, but have a very low frame difference. This implies that the decision threshold should be a function of both the match length and the average frame distance. To validate this hypothesis, we computed these values for a small number of videos (approximately 100) that we visually classified as a positive or negative detection, as shown in Figure 4. In this figure the triangles are the incorrect matches (false positives) and the rectangles the correct matches (true positives) as obtained by manual inspection. Using only the match length (say more than 200 frames) as an evaluation criteria would eliminate many true positives, so we choose the shown exponential curve as the new evaluation criteria. What this means is that matches that are shorter in length will be accepted if they have a lower difference score, which is sensible. Unfortunately, we did not have time to include this new criteria in our video only results, and for reasons that are not clear when used in the combined a/v submission the performance was not improved. We still need to investigate this aspect of the matching process.

2.2 Video Transformation Processing

A considerable part of the copy detection process is in dealing with the different types of transformations. The transformations which simply modify the image without inserting or deleting any content (blur, rencoding, gamma noise) are dealt with automatically by the SURF feature point detection process. In other words, we rely on the invariance of the SURF detection process to deal with these transformations, and do not specifically do any other processing. However, the remaining transformations actually change the video by inserting or detecting part of the image, and must be dealt with explicitly. This is done by first computing a mask file for each query video which simply consists of a binary value computed from the variance of each pixel over the entire video. This is shown in Figure 5, where the mask file is displayed for a given query video. Essentially this mask file marks areas of the image which are not changing over the entire query video. The counts for image quadrants which are masked are then adjusted during the search process so that the masked parts of the image are ignored. This allows us to deal with transformations such as crop, shift, and various types of insertion (text or still picture). Different types of mask files for text insertion, crop and shift are shown in Figure 6.

The process of dealing with an inserted video is more complex; here we attempt to detect whether a video is inside another video, and if so we extract this video and process it separately. The process of dealing with an image flip is shown in Figure 7, and with our representation this simply requires us to mirror the descriptor for each video frame of the query. There is also specific processing for the ratio and shift transformations, since these require an adjustment of the region sizes when comparing the image descriptors. Finally, the dropping of frames is dealt with implicitly by the matching process which skips over a small number of frames in a sequence that do not pass the match test.

3. Audio-only Copy Detection Approach

The audio-only copy detection search scheme was based on the computation of a coherence function, using intermediate features in the ITU-R BS.1387 Perceptual Evaluation of Audio Quality (PEAQ) standard. A comprehensive analysis of the PEAQ standard can be found in [4], a C code implementation of the basic version of the standard is available online [1], and a Matlab implementation is also available online [2] (although it was not used for our work). The idea of using PEAQ intermediate features was mostly to include psychoacoustic effects such as critical bands, frequency masking and loudness.

1. For the audio component of both the reference database videos and the audio queries, some feature files were first produced making use of the following steps:

   - conversion of all the audio components from the MPEG 1 Layer 2, 44.1 kHz, stereo format to the linear PCM 44.1 kHz stereo WAV format, using the *ffmpeg* utility.
   - computation of the PEAQ intermediate variables called "specific loudness patterns" (eq. (68) in [4]) for 109 critical bands i.e. frequency bands:
     - the PEAQ standard was designed using psychoacoustic criteria and it assumes that input signals have been sampled at 48 kHz. In this project, signals sampled at 44.1 kHz were provided instead as the inputs of the PEAQ code, in order to avoid the extra computations required for sampling rate conversion. This obviously introduces some inaccuracies in the psychoacoustic model.
     - in the PEAQ code, the signals are processed by frames of size 2048 samples (46.4 ms at 44.1 kHz), with 50 % overlap (23.2 ms)
     - the PEAQ code was modified so that the two stereo channels from an input file are averaged (in the time domain) to produce a single mono channel, to reduce the number of features produced and the resulting search complexity
     - to further reduce the number of features and the resulting search complexity, the specific loudness patterns kept as features were made of the maximum (for each critical band) of the patterns obtained from X consecutive overlapping input frames. Moreover, a jump of Y overlapping input frames was used between each of these "maximum" specific loudness patterns produced. The values X=6 and Y=6 were used in the final implementation, resulting in specific loudness patterns each built from 162.5 ms of samples with a time shift of 139.3 ms between them.
     - the mean of each resulting specific loudness pattern (computed over all critical bands) was removed from the pattern before its values were stored to feature files, and the square root of the total energy of each resulting mean-removed pattern was also stored as extra information in the feature file.

2. Making use of the feature files produced in the previous step, for each audio query coherence functions γ were then computed between sequences of G consecutive feature frames from the audio query and sequences of G consecutive feature frames from each reference database file. Moreover, to reduce the search

complexity, in the audio query a shift of *Z* feature frames was used between each sequence of consecutive frames to be compared. The values *G*=2 and *Z*=32 were used in the final implementation, resulting in feature sequences covering 325 ms and a sequence shift of 5.2 s between the feature sequences used in the query file. The coherence functions were computed as:

$$\gamma = \frac{1}{G} \sum_{g=1}^{G} \left(\frac{\sum_{k=0}^{108} l_{ref,g}[k]\, l_{query,g}[k]}{\sqrt{\sum_{k=0}^{108} \left(l_{ref,g}[k]\right)^2} \sqrt{\sum_{k=0}^{108} \left(l_{query,g}[k]\right)^2}} \right)$$

where $l_{ref,g}[k]$ is the gth specific loudness pattern in a feature sequence from a given file in the reference database, and $l_{query,g}[k]$ is the gth specific loudness pattern in a feature sequence from the considered audio query.

For each audio query, the audio-only copy detection search found the highest coherence score γ over all the files in the reference database, returning the corresponding locations of the matching sequences of features in the audio query and the reference database. This approach produced only one result per query. Other alternatives would have been to keep several of the highest coherence scores produced for each audio query, and also to measure for how many consecutive feature frames or sequences a high coherence score was found, allowing us to make use of this information for copy detection. Finally, for both the No-Fault and the Balanced profiles of the audio-only copy detection search, based on the above and on the definition of the Normalized Detection Cost Rate (NCDR) [8] for the TRECVID 2009 copy detection task, it was decided to use a common threshold of 0.992855.

4. Conclusion

In terms of performance our video search ranked in the middle in terms of the HDCR for both balanced and NOFA submissions. As we have stated, we believe that the proper use of our new evaluation criteria would improve our NDCR scores. However, in terms of mean F1 our approach is one of the best, as would be expected from a system that matches all video frames. But more importantly we believe that this method satisfies our design goals, and we are in the process of implementing it on GPU hardware.

References

[1] http://www-mmsp.ece.mcgill.ca/documents/downloads/afsp/afsp-v8r2.tar.gz.

[2] http://www-mmsp.ece.mcgill.ca/documents/downloads/pqevalaudio/pqevalaudio-v1r0.tar.gz.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[4] P. Kabal. An examination and interpretation of itu-r bs.1387: Perceptual evaluation of audio quality. *TSP Lab Technical Report, Dept. Electrical & Computer Engineering, McGill University*, 2003.

[5] Y. Ke and R. Sukthankar. An efficient parts-based near-duplicate and sub-image retrieval system. In *IEEE Transactions on Audio, Speech and Language Processing*, volume 16, pages 396–407, 2008.

[6] C. Kim and B. Vasudev. Spatiotemporal sequence matching for efficient video copy detection. In *IEEE Trans. Circuits Syst. Video Technology*, volume 15, pages 127–132, 2005.

[7] A. Kimura and K. Kashino. A quick search method for audio signals based on a piecewise linear representation of feature trajectories. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, 2004.

[8] 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*, pages 321–330, New York, NY, USA, 2006. ACM Press.

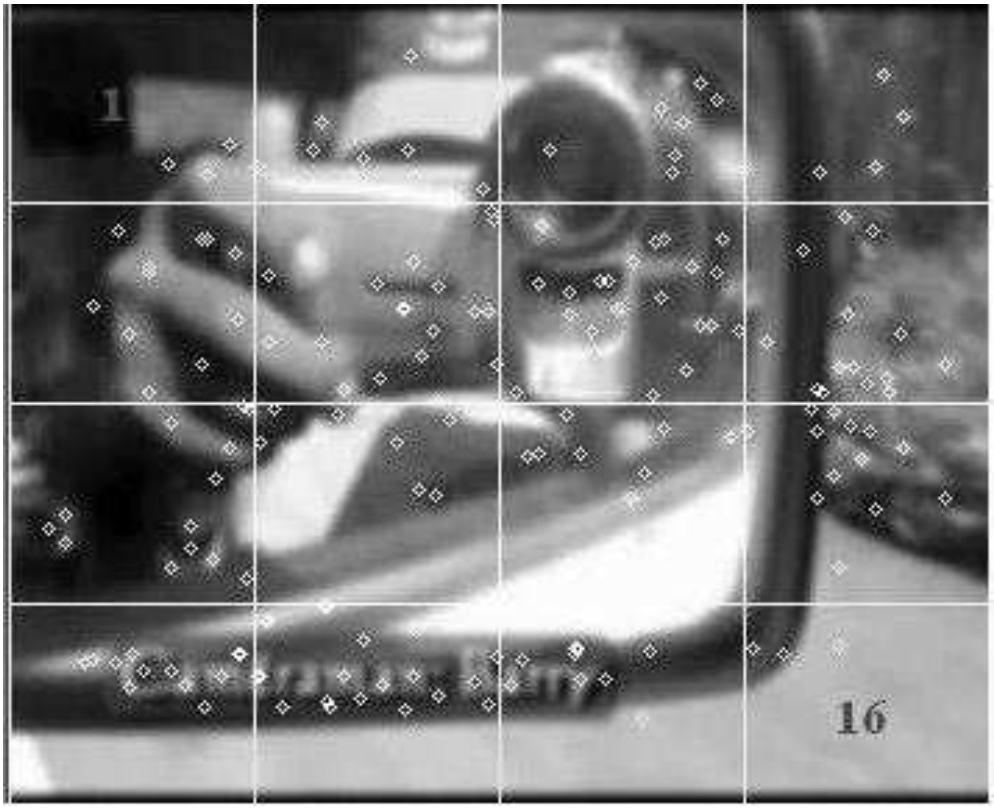

Figure 1. The features (SURF points) in a single frame. Our descriptor is simply the number of points in each quadrant [3,7,...,3]

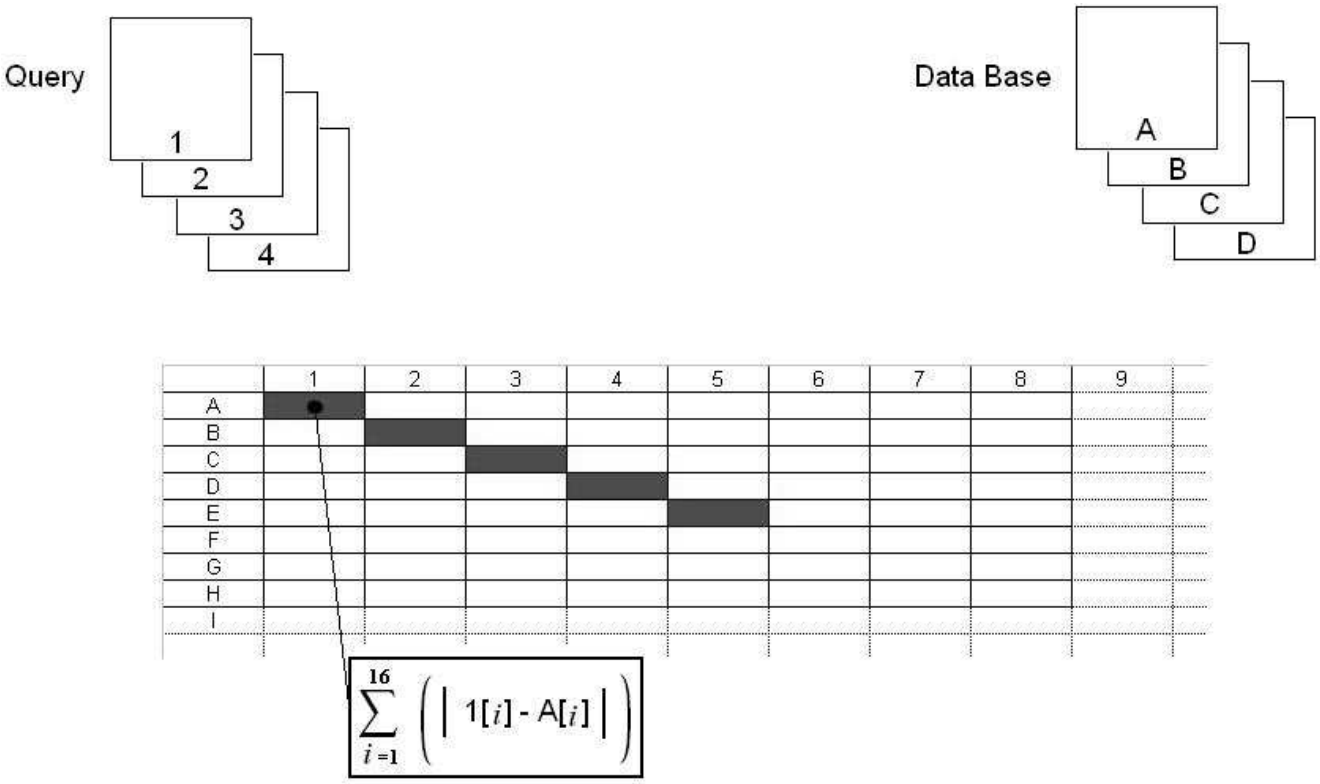

Figure 2. Measuring video similarity. Each frame of the query is compared with each frame of the input video. A cross-table of comparison of the vector of SURF counts is created; the similarity measure corresponds to the longest diagonal with a distance score less than a given threshold.

	A	B	C	D
1	S<thresh			
2		S<thresh		
3			S<thresh	
4				S>thresh

Figure 3. Finding a match in the cross-table of comparison of the vector of SURF counts

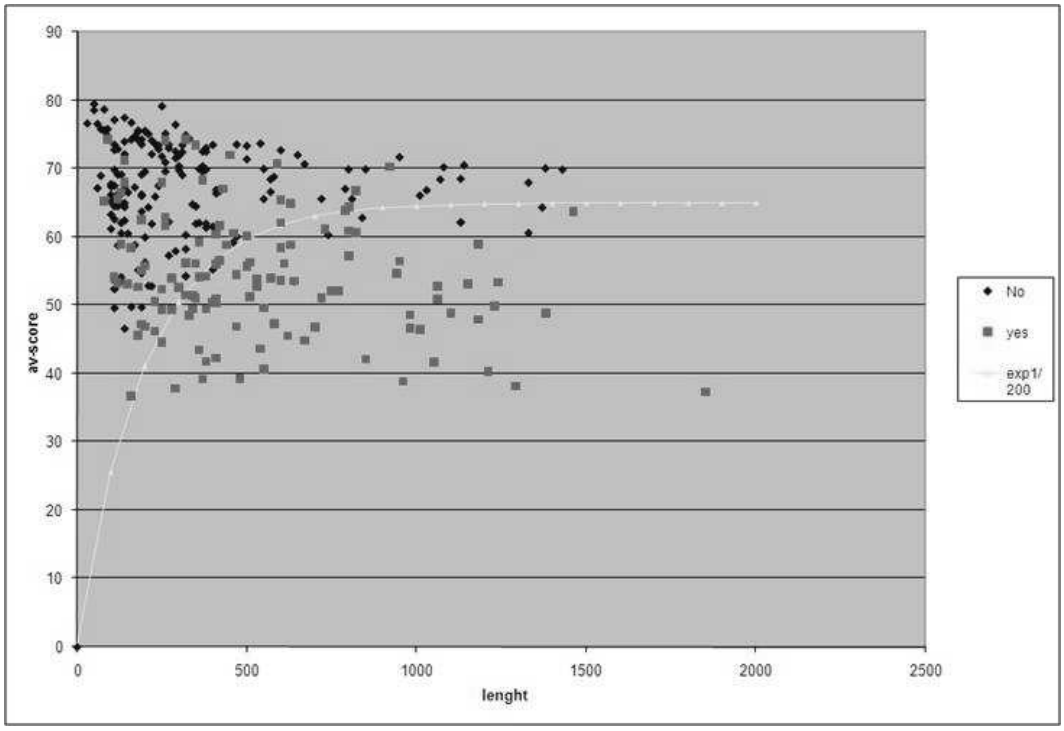

Figure 4. The thresholding criteria, based on match length and average match score

$$\sigma^2 = \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{n}$$

When σ = 0 draw the pixels as white

Otherwise draw the pixels as black

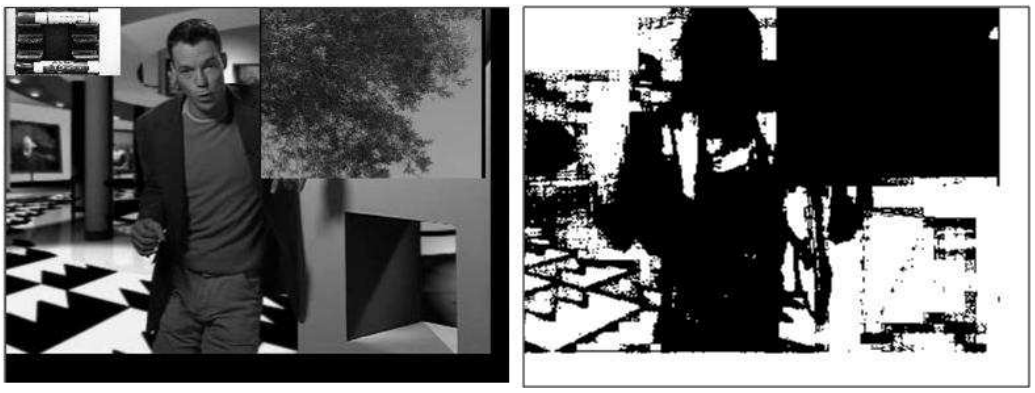

Figure 5. The mask creation process for a single query video

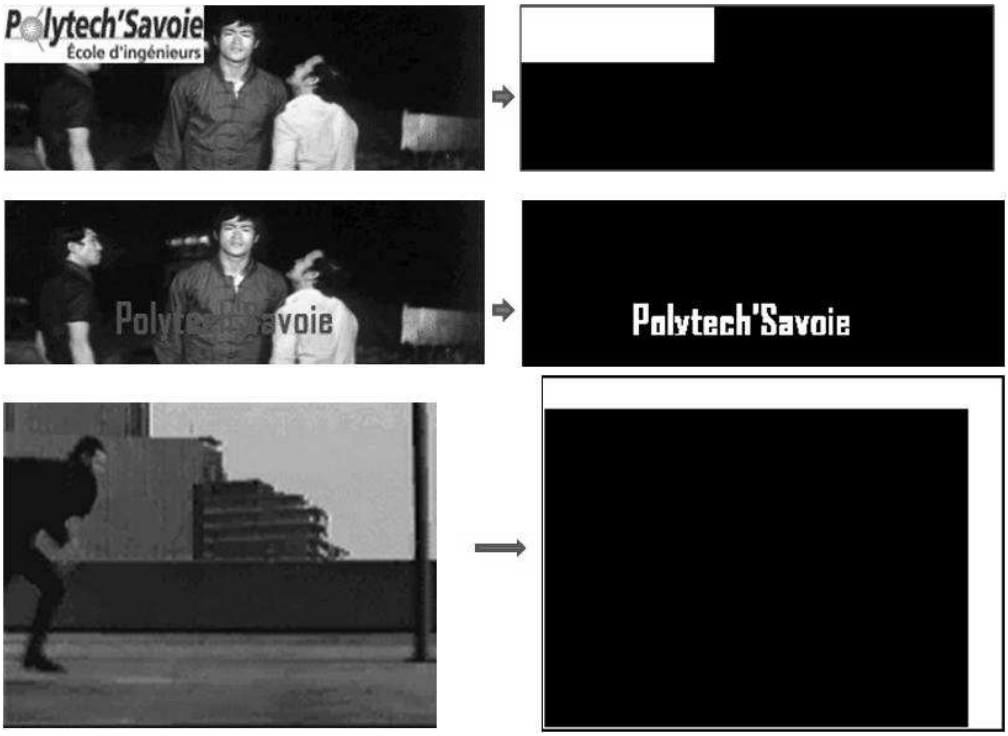

Figure 6. The masks for image and text insertion, along with a shift

Figure 7. The processing for dealing with image flips