TRECVID 2007 by the Brno Group
High Level Feature Extraction
&
Shot Boundary Detection

Adam Herout, Vítězslav Beran, Michal Hradiš, Igor Potúček, Pavel Zemčík, Petr Chmelař

A. Structured Abstract:

High Level Feature Extraction

1. The runs:
   - **A_brU_1** – *features* extracted from each frame; SVM per-frame classifier trained on frames in each shot; simple decision tree judging shots based on per-frame results
   - **A_brV_2** – same as A_brU_1, but SVM trained on all training data (the first run divided the training data to training and cross-validation datasets), with SVM configured from the previous run
2. Significant differences between the runs:
   - As expected, the second run performed generally better, in some cases notably better (which is slightly surprising, because besides the amount of training data, there was no change)
3. Contribution of each component:
   - The low-level features appear to be *good enough*, though their number is relatively large and having more time we would experiment with reduction of the feature vector size (now 572 low level features)
   - We considered using some mid-level features based on existing solutions the group has, such as face detection, car detection, etc., but for time constraints did not employ these in the feature vector
   - The per-frame classification seems to suffer greatly from mis-annotated frames (whole shots are considered to share the same annotation information in our system) and could be the weakest point of the system
   - The per-shot decision making seems to be sufficient given the data coming from the per-shot classification
4. Overall comments:
   - see further in the paper

Shot Boundary Detection

We describe our approach to cut detection where we use AdaBoost boosting algorithm to create a detection classifier from a large set of features which are based on few simple frame distance measures. First, we introduce the reasons which led us to use AdaBoost algorithm, then we describe the set of features and we also discuss the achieved result. Finally, we present the possible future improvements to the current approach.

B. Overall Information on Brno Image Processing Group

The image and video processing group "video@fit" is a part of Department of Computer Graphics and Multimedia (DCGM), Faculty of Information Technology (FIT), Brno University of Technology. Its focus is on research, development, and application of algorithms of image processing, videosequence processing, and computer vision. The group is responsible for teaching of Image processing and Computer vision courses in M.Sc. programme at FIT.

The group participates in the research of several projects, such as EU IST CareTaker or AMIDA projects, or locally funded projects, such as GA AVCR (Czech Academy of Sciences Grant Agency) RIPAC, or GACR (Grant Agency of the Czech Republic) "Algorithms of Image Recognition".

The research areas of video@fit include:

- Automatic video editing and summarization,
- algorithms for detection and tracking of objects and human body parts,
- acceleration of image processing and computer vision in hardware,
- detection and positioning of human face and other objects,
- industrial and traffic applications of image processing,
- evaluation of trajectories and complex video events and actions.

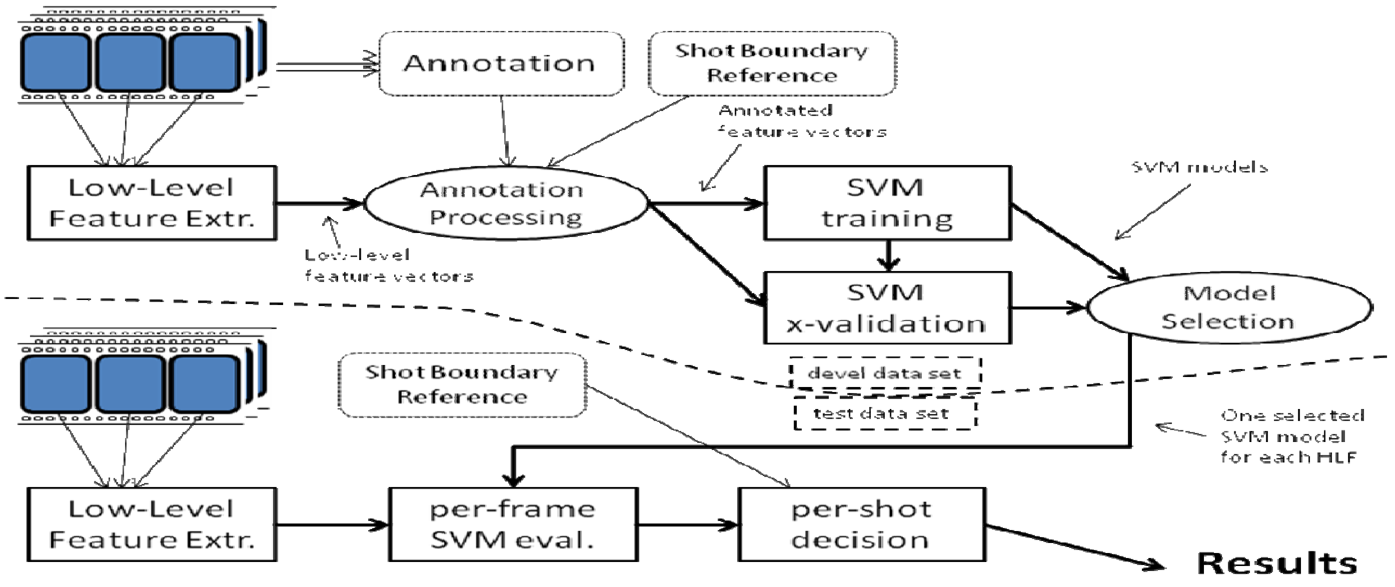
Furhter information on the group, publications, research projects, and contact can be found at http://www.fit.vutbr.cz.

C. Task 2: High Level Features Extraction

This section describes the solution to Task 2 “High Level Feature Extraction” structured as follows: section 1 describes the overall scheme of the system, section 2 describes the low level features used as the input of the per-frame classification engine, which is described in section 3. The per-shot decision process is described in section 4, and the performance of the system is shortly analyzed in sections 5 and 6.

C.1. Structure of the System

The structure of our high-level feature extraction system is shown in the following figure.



The most important parts of the system shown in the figure are these:

Low-level feature extractors – These are described in section C.2. Several feature extractors are employed, their feature vectors are concatenated to make a per-frame feature vector. Note that the features are relatively generic. Feature range normalization is part of the feature extraction process.

SVM training and cross-validation – Using grid-search, the SVM kernels are optimized and for each high-level feature to be searched, one model is selected by the Model Selection module. The SVM training and evaluation process is shortly described in section C.3.

Per-frame SVM evaluation – Evaluates the low-level feature vector for each frame in the testing dataset based on the selected SVM model for each high level feature.

Per-shot decision – Judges the set of per-frame classifications based on the shot-boundary reference to make a decision on each shot of the testing video dataset. (see section C.4).

This system can be described as a brute-force approach to high-level feature extraction, since the low-level features are rather generic than built to match specially the high-level concepts looked for. Also the main classification machine (SVM) is generic. The only specialized part of the system is the per-shot decision making subsystem, which constitutes a very simplistic decision tree.

C.2. Low-level Features Extracted from the Frames

The low-level features used for frame description utilized in the system are –color histogram based on HSV color model and multi-scale gradient distribution of a frame intensity.

Color Histogram Based on HSV Color Model

The color histogram contains statistical information about color distribution in terms of frequency of hues and saturations in the frame (using HSV color model). The better spatial description is achieved by dividing the frame into several patches. The frame division is not adaptive, so the patches have a similar size. Each patch is processed separately; the histogram is computed and normalized.

Multi-scale Gradient Distribution

The histogram of gradient orientations serves as other part of the feature vector. First, the frame gradients are computed. Then each gradient contributes to the histogram bin according to its orientation. The contributions are weighted by the gradient magnitude. The gradients are computed on different frame resolutions so also lower-frequency structures contribute to final feature vector.

The resolution of both the color histogram and gradient histogram, the resolution of the frame grid and amount of frame scale levels are all the descriptor parameters.

Composed Per-Frame Feature Vector and Its Processing

All the partial feature vectors mentioned in the previous text are concatenated to make a per-frame feature vector which serves as input to the per-frame classifier described in C.3. This feature vector is normalized across the whole data-set, i.e. maxima and minima for separate features are found in the training dataset and the features are independently re-scaled so that these maxima and minima correspond to 0.0 or 1.0 respectively. The scaling factors are used for the test data, which can then scaled exceed the normal interval (0-1). These infrequent cases were not found harmful for evaluation by the SVM classifier.

C.3. Per-Frame SVM Classifier

Libsvm 2.84 was used for svm classification of separate frames – one classifier was trained for each high level feature to be detected. Given shot annotations were used for all the frames in a given shot. The svm classifier was trained on 20% of the training data provided, and cross-evaluated on the resting 80%. This cross-evaluation was used for selection of proper parameters of the SVM kernels.

The SVM training and parameters selection took majority of the development time (thousands of hours sequentially) and this would be the part most likely to get speeded up in future versions of the system.

C.4. Per-Shot Decision Based on Per-Frame Results

For each high level feature separately, the results from per-frame classification are judged for each shot (dropping frames at the beginning and end of each shot to avoid mis-classification). Two quantities are observed in this decision process:

- Positive rate $r = \frac{P}{N}$ where P is the number of positively judged frames and N is number of all frames (excluding the dropped initial and tailing frames in each shot).
- Largest positive sequence s is the length of the longest sequence of positively judged frames.

Two thresholds are defined for these quantities r_t, s_t, exceeding either of them selected the shot being judged for output. These two threshold values were found experimentally on the testing data.

The output (positively classified) shots are sorted by the value of r, in cases their number would exceed the given TRECVID limit of 2000 positive shots, shots with largest r are selected.

C.5. Results, Future Work

The results of our system for the high-level feature extraction task in TRECVID 2007 were average or slightly below average.

We would have expected significantly better results, if we had the time to employ our object classifiers trained for different classes of objects (faces, vehicles, planes, animals, …) as the “low-level” features entering our per-frame classification.

During the training and evaluation process on the large cluster of computers, we monitored the performance of separate parts of the system and identified several aspects of the process as bottle-necks, most of which are relatively easy to eliminate or work around. That could speed up the processing several (potentially many) times, which would enable us to tune the system better and incorporate more mid-level features into the classification. Some of these improvements have already been implemented in our training/testing engines and are surely going to improve the results in future evaluation like this.

For future evaluations generally, we would like to focus on particular higher-level features that could benefit from our object classifiers and other technologies being developed for longer time, rather than build a general classification machine, uninformed about the nature of the classified high level features. On the other hand, the results of such uninformed machine were better than we expected, which could be interpreted in several ways. It surely compliments the selection of low-level features (though the exact set was not optimized in any way), and it shows that in many cases, simple low-level features such as those mentioned in section C.2 suffice for a simple base-line solution.

C.6. Conclusion

Our solution can be generally described as a brute-force approach, which relies on generic software pieces (several feature extractors, SVM library, training/evaluation framework for distributed computing), that solve the task in an "uninformed" way. For future implementations of HLF extraction for TRECVID or similar evaluations we intend to include object detectors and similar frame-processing engines to provide specialized and "informed" knowledge to the overall classification process. These will be represented as mid-level features entering the per-frame classifier. Also many speed-up optimizations have been suggested from the undertaken runs, which would enable more experimenting for future implementations.

D. Task 3: Shot Boundary Detection

We focused solely on cut detection in the TRECVID 2007 Shot Boundary Detection Task. We approached the problem purely as a pattern detection task using AdaBoost [1, 2, 3] learning algorithm to create the detection classifiers and a large set of simple features extracted from the video in the uncompressed domain.

The set of features is based on per pixel distance measures (difference of pixel intensities, squared difference and correlation) and difference of RGB histograms. These distance measures are computed on a regular grid with 4 lines and 4 columns giving 16 values for each distance measure. Additionally, the set of features is supplemented by mean, median, standard deviation and other values computed from the original 16 values. The features are extracted from a set of frame pairs in a local neighborhood giving total 2100 features.

We used total six hours of hand-annotated video sequences for training. Five hours of the training sequences were randomly chosen from the TRECVID 2007 Sound and Vision data which were supplemented by one hour of Czech Television broadcast. According to this, our shot boundary detection should belong to the category C.

D.1. AdaBoost classifier

AdaBoost and derived algorithms are often used for object detection [4] in computer vision where they achieve state-of-the-art results in both classification accuracy and classification speed. When detecting objects in images with AdaBoost, a large and over-complete set of features (180 000 in [4]) is extracted from the original data using simple filters (Haar-like features) which can be individually computed very rapidly. AdaBoost then creates a strong classifier which is a linear combination of relatively low number of simple weak classifiers (decision stumps, decision trees). As each of the weak classifiers produces decision based only on a single feature and only the features which are needed for classification are computed, the resulting strong classifier can be very fast. When used in this way, AdaBoost in fact performs feature selection.

Another pleasant property of the AdaBoost algorithm is that it can cope with relatively large numbers of samples. This supports generalization properties and can be further improved by using some kind of cascade of classifiers [4]. In each stage the cascade rejects background (non-face) samples that are already classified with enough confidence. The result of this is that only the most difficult samples propagate to the later stages of the cascade and if there are enough samples to bootstrap, the classification function can be reliably estimated even for these rare and difficult samples. This way, the cascade and similar techniques minimize False Alarm Rate which can drop below 1e-5 in face detection and also minimizes average number of the weak classifiers evaluated during classification.

For TRECVID 2007, we used the AdaBoost learning algorithm with decision tree weak classifiers and with number of leaf nodes set to four. We did not use any variant of classifier cascade, because we did not have sufficient learning data.

D.2. Feature Extraction

A major advantage of the AdaBoost algorithm in the form we use it is that it is able to cope with very large number of features from which it selects only relatively few features for the final classifier. This gives us an opportunity to supply the learning algorithm with any feature we can think of and leave the selection of features for the learning algorithm. This property is used in the current cut detection system only partially and the system should benefit form adding new features in the future.

The ability of AdaBoost to choose only few features for the final classifier could be used to create very fast cut detectors. The high speed could be achieved by computing only the features which are needed by the classifier during the detection. However, we did not take advantage of this fact this year and perform feature extraction as an independent step.

As the weak learners which we use, need the features to be discretized to relatively low number of levels, we normalize the features and discretize them to 255 equidistant levels.

All features which we use are based on one of image distance measures. The distance measures which we use are:

Sum of pixel value difference:

$$D = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} \left| I_1(i,j) - I_2(i,j) \right|$$

Sum of pixel value squared difference:

$$D = \frac{\sum_{i=0}^{M}\sum_{j=0}^{N}\left(I_1(i,j)-I_2(i,j)\right)^2}{\sum_{i=0}^{M}\sum_{j=0}^{N}I_1(i,j)^2 \cdot \sum_{i=0}^{M}\sum_{j=0}^{N}I_2(i,j)^2}$$

Correlation:

$$D = \frac{\sum_{i=0}^{M}\sum_{j=0}^{N}\left(I_1(i,j)\cdot I_2(i,j)\right)}{\sum_{i=0}^{M}\sum_{j=0}^{N}I_1(i,j)^2 \cdot \sum_{i=0}^{M}\sum_{j=0}^{N}I_2(i,j)^2}$$

The last distance measure is difference of RGB histograms. The resolution of the histogram is 4x4x4 which gives total 64 bins.

To extract the features, image is first divided by a regular grid with four lines and four columns and the distance measures are computed for each of the bins. From these 16 values mean, median and standard deviation is computed. After this, the 16 values are sorted and divided into two halves for which mean, median and standard deviation is also computed. Finally, first and second derivation is computed for each of the 100 values. This gives total 300 features extracted form a pair of images.

In many previous works on shot boundary detection [5], the intra-frame distance measures are computed from some small local neighborhood. This extension should provide some resistance to abrupt non-cut events in the video like flashes, etc. We follow this idea and compute features for multiple frame pairs with increasing distance from the classified position. Additionally, we compute the features for few frame pairs before and after the actual position. These additional features should provide the classifier with enough information to precisely localize the cuts. There are seven frame pairs for which features are computed. The frame pairs are shown on figure 1. 300 features per seven frame pairs give total 2100 features.

For some of the runs, the set of features was additionally supplemented by ranks of the original features in small temporary neighborhood. The size of the neighborhood was set to 24 frames. The feature set with temporary ranks consisted of 4200 features.

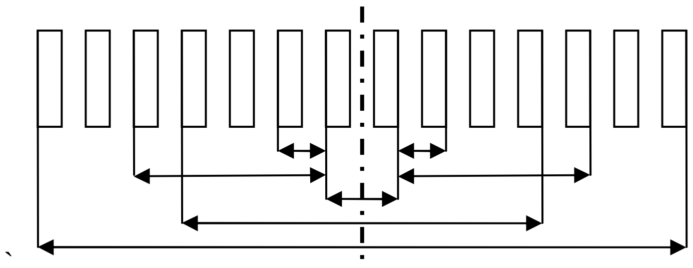


Figure 1. Frame pairs from which features are axtracted.

D.3. Training Data

Training data set consisted of five hours of randomly selected sequences from TRECVID 2007 task two development data and one hour of Czech television broadcast. The sequence selection was not completely random, but some video sequences containing ambiguous situations (eg. picture in picture) were discarded as were not sure how to handle such situations. The one hour of Czech television broadcast was added just in case the SBD test dataset will significantly differ from the task two development data. The training video sequences were hand-annotated using a video annotation tool which allowed faster then real-time annotation of shot boundaries. There are approximately 5000 cuts and 540,000 non-cut samples in the training data.

D.4. Results

The submitted runs are results of three individual classifiers which differ in the number of weak classifiers, amount of non-cut training samples and the feature set. Three or four runs with different threshold setting were generated for each of the three classifiers to trade-off precision and recall. Description of the individual runs and achieved results can be seen in table 1 and the classification results can be compared with results of other participants in figure 2. When comparing HMNR and HMWR classifiers, the temporal ranks do not seem to improve the results. On the other hand, results of the SMWR classifier suggest that longer classifiers provide better results, although the training samples are ideally classified after first few weak classifiers.

Run	Precision	Recall	F-measure	Training set size	Temporal rank features	Classifier length
HMNR_0	0,984	0,866	0,921	180000	NO	15
HMNR_1	0,967	0,955	0,961	180000	NO	15
HMNR_2	0,922	0,981	0,951	180000	NO	15
HMWR_0	0,985	0,847	0,911	170000	YES	15
HMWR_1	0,975	0,942	0,958	170000	YES	15
HMWR_2	0,944	0,972	0,958	170000	YES	15
SMWR_0	0,987	0,723	0,835	120000	YES	30
SMWR_1	0,987	0,901	0,942	120000	YES	30
SMWR_2	0,978	0,957	0,967	120000	YES	30
SMWR_3	**0,96**	**0,976**	**0,968**	**120000**	**YES**	**30**

Table 1. Our cut detection results in TRECVID 2007 evaluation and description of the classifiers.

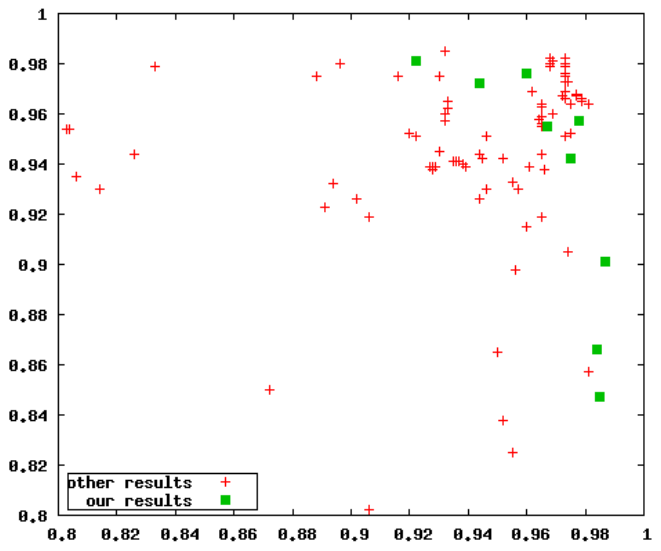


Figure 2. Our cut detection results in TRECVID 2007 evaluation compared to other participants.

D.5. Conclusion and Future Work

Classifiers created by the AdaBoost algorithm using simple weak learners and a large set of features proved to be very suitable for cut detection. Without any previous experience and experiments we were able to achieve results which are close to the overall top scores. In fact, only three other participants submitted runs which achieved higher F-measure then our top run.

This approach to cut detection is very promising as there is lot of possible ways how to improve the results. In fact, we already created a classifier that achieved recall 0.973 and precision 0.982 on the TRECVID 2007 SBD data which was the top score in the evaluations. This classifier was trained using WaldBoost [6] algorithm on the same data and with the same features as the submitted runs. In the future we plan to add more complex frame distance measures (e.g. based on motion estimation, KLT-tracking, …) and experiment with cascades of classifiers. We also plan to explore possibilities of semi-automatic annotation which will be necessary to annotate enough video data needed for bootstrapping.

E. References

[1] Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119--139, August 1997.

[2] Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. In: Machine Learning, 37(3), 1999, s. 297-336.

[3] Freund, Y., Schapire, R..: A short introduction to boosting. J. Japan. Soc. for Artif. Intel. 14(5), 1999, s. 771-780.

[4] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In CVPR, 2001.

[5] M.Cooper, J.Adcock, R.Chen and H.Zhou. FXPAL at TRECVID 2005. Proceedings of TRECVID 2005, 2005.

[6] Sochman, J., Matas, J.: WaldBoost - Learning for Time Constrained Sequential Detection. CVPR, (2), 2005, s. 150-156.