Universität Karlsruhe (TH) at TRECVID 2007

H.K. Ekenel, M. Fischer, H. Gao, K. Kilgour, J.S. Marcos, R. Stiefelhagen

interACT Research, Computer Science Department, Universität Karlsruhe (TH)
Am Fasanengarten 5, Karlsruhe 76131, Germany
{ekenel,mika.fischer,s_gao,kevin.kilgour,stiefel}@ira.uka.de, jordi@gps.tsc.upc.edu
Web page: http://isl.ira.uka.de/

***Abstract*—In this paper, we present the systems developed by the interACT Research Center at Universität Karlsruhe (TH) for the TRECVID 2007 evaluation. Our first participation in the TRECVID evaluation includes the shot boundary detection and high-level feature detection tasks. The shot boundary detection system contains four separate detectors, one for each type of shot boundary we try to detect: cuts, fast dissolves, fade in / fade outs and dissolves. High-level feature detection utilizes both visual and textual cues. It uses color, texture and edge-based features for visual processing and relative term frequency-based features for text processing. All the systems we have developed have been trained on the common development data.**

I. Introduction

This year, it was the first time that the Universität Karlsruhe (TH) participated in the TRECVID evaluations. The main focus of our first participation was on developing a common software framework for multimedia processing and to build baseline systems for the shot boundary detection and high-level feature detection tasks. The baseline systems have been developed by benefiting from the experiences gained from the previous TRECVID evaluations. They consist of state-of-the-art feature extractors and classifiers which have been shown to perform well. The shot boundary detection system is based on the ideas in [1]–[3]. For high-level feature detection, features similar to MPEG-7 visual descriptors, as well as geometric blur features [4] have been used as visual cues and relative term frequency-based features have been used as textual cues. The shot boundary detection system uses thresholding to detect shot-boundaries, whereas the high-level feature detection system relies on support vector machine classifiers. Many fusion schemes have also been tested for combining multiple low level features.

II. Shot Boundary Detection

The shot boundary detection system is built in a modular way and consists of an MPEG decoder [5], several feature extractors, several detectors for different types of shot boundaries, and a fusion module, that fuses the results of the different detection modules.

The detection modules declare which features they need and how many frames of history, so that only required features are computed, and features that are used by multiple detection modules are only computed once. This way it is also very easy to experiment with different inputs to the detection modules.

The detection modules are specialized for a specific type of shot boundary and are described in detail below. New modules can be easily integrated into this framework.

The fusion module has the purpose of resolving any conflicts that may arise, like overlap of shot boundaries detected by different modules.

After fusion the system outputs an XML file containing the detected shot boundaries.

A. Feature extraction

The following features were extracted from each frame:

1) RGB histogram difference: The RGB histogram of each frame, with 8 bins per color, was computed and – starting with the second frame – the histogram difference between the current and preceding frame was computed. First the 3D histograms were vectorized, yielding histogram vectors $\mathbf{x}$ and $\mathbf{y}$. The difference was then computed by:

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{A} (\mathbf{x} - \mathbf{y})$$

where a_{ij} is the similarity of the colors corresponding to the centers of the histogram bins i and j:

$$a_{ij} = 1 - \frac{\sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}}{N}$$

where N is a normalization constant, so that $0 \leq a_{ij} \leq 1$.

2) Edge map: For each frame an edge map was computed using the Sobel operator.

3) Intensity standard deviation: Each frame was converted to gray-scale and the standard deviation of the intensities was computed.

B. Shot boundary detectors

1) Cut detector: The cut detector uses the RGB histogram difference feature to find cut candidates. For this it considers a window of seven frames around the current frame. The cut detector searches for peaks, so if the current frame is not the largest value in the window, the candidate is discarded. Otherwise the following values are computed: the kurtosis of the values in the window, the ratio of the current value and the sum of the other values in the window, as well as the ratio of the current value and the second largest value in the window. The histogram difference itself and the computed values are then checked against thresholds. If any threshold is not reached, the candidate is passed to the fast dissolve detector. Otherwise a flash detector is run to ensure that the histogram difference is not the result of a flash. The flash detector works by computing the matching error between the edge maps of the current and preceding frames, as described in [1]. If the matching error is low, a flash is detected and the candidate is passed to the fast dissolve detector. Otherwise a cut is detected.

2) Fast dissolve detector: The fast dissolve detector is employed on cut candidates that fail to pass the strict requirements for a cut. It considers the histogram differences within a window of nine frames around the current candidate and computes the ratio of the three center frames and the sum of the rest of the frames in the window. If this value does not exceed a threshold, the frame is discarded. Otherwise it is checked if some frames can be found, so that the candidate is between them and they can be successfully modeled as a dissolve. If X is the start frame and Y is the end frame, then all frames between X and Y are modeled by $Z = \alpha X + (1-\alpha)Y$. The α's are found by applying the least squares method. For the dissolve to be accepted the α must increase from start to end frame and the squared error must not be too large. If a dissolve is detected, the start and end frames are checked for a flash and if none is detected, the candidate is declared a fast dissolve.

3) Fade out / fade in detector: For the FOI detector we followed [2]. The FOI detector considers the standard deviation of pixel intensities. It is triggered on monochrome frames, which have a very low standard deviation. Sequences of monochrome frames are reported to the fusion module. Additionally the start and end of the monochrome frames are checked for a fade out or fade in respectively. This is done by checking for a linear decrease/increase in the standard deviation by linear regression. If a fade is found it is also reported to the fusion module. The fusion is done late because this way we also capture cases where there is a cut to black, then some black frames and a a fade in to the next scene. Such transitions would not be detected by a pure FOI detector as described in [2] because there is no fade out in the beginning of the transition.

Optionally, the system can ignore monochrome frames that are not black or gray but contain a significant amount of color.

4) Dissolve detector: For the dissolve detector several heuristic approaches were tried: [2], [3], [6]. Unfortunately none of them reached acceptable performance on the TRECVID 2006 data set. So in the end we implemented a method that's similar to the one used in the fast dissolve detector.

Dissolve candidates are detected similarly to [1]. The (smoothed) intensity standard deviation is checked for a decrease followed by an increase. Each of those cases is a dissolve candidate.

For each candidate the point of minimal standard deviation is found and it is checked that the start or the end of the dissolve has a significantly larger standard deviation than the minimum. If this is the case, the sequence of frames with a length of nine frames around the frame with minimum standard deviation is checked for a dissolve using the same technique as described in section II-B2. If a dissolve is detected, the complete candidate is declared a dissolve.

Run	Comment
base	Baseline system
mod1	Discard FOIs with color
mod2	lower thresholds for cuts
mod3	higher thresholds for cuts
mod4	lower thresholds for fast dissolves
mod5	higher thresholds for fast dissolves
mod6	dissolves have precedence over other transitions
mod7	lower thresholds for dissolves
mod8	higher thresholds for dissolves
mod9	no dissolve detection at all

TABLE I
RUNS SUBMITTED IN SHOT BOUNDARY DETECTION TASK

C. Fusion module

The fusion module is responsible for fusing the monochrome frames returned by the FOI detector with corresponding fades, cuts or dissolves. The other function is to resolve overlapping transitions. Especially the dissolve detector produces many false positives, so if a dissolve is overlapping with any other transition, the dissolve is discarded in all runs except one.

D. Evaluation Results

For the TRECVID evaluations 2007 we prepared ten system configurations, which are shown and described in Table I. The results of the evaluation can be seen in Table II.

E. Preliminary analysis

It can be seen clearly that our system works quite well for cuts (i.e. transitions spanning five frames or less), but the performance is not so good for longer transitions.

Because there were fewer long transitions than in previous evaluations the total results are still acceptable.

We have performed a preliminary analysis of the results:

1) Cuts and fast dissolves: There are very few false detections or misses that cannot be explained easily. The most common problems were:

- Our thresholds were not tuned for black and white videos, so there were some misses in those segments. Similarly for segments with washed-out colors.
- Some segments have an image distortion in the last frame before a cut, which caused some misses as well.
- Some false detections really have a sudden change in image content. To establish that it is not a cut semantically, more detailed analysis of the image content should be performed.

2) Fade outs / fade ins: A problem with our approach to fade detection was that our threshold for the standard deviation of the pixel intensities was set too high, leading to some false detections in dark scenes with very low contrast. In the past TREDVID data, which was used for development, such scenes were very rare.

3) Dissolves: Our dissolve detector did not work very well and the thresholds were set too low in all runs except mod8, leading to lots of false detections.

In the future we will not pursue the presented approach further and instead use an approach based on SVMs [1], [3].

	Total		Cuts		Graduals		Gradual frames	
Run	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)
base	91.3	59.8	93.6	94.1	65.5	8.9	74.8	71.2
mod1	91.4	60.1	93.9	93.9	63.1	8.7	74.6	71.3
mod2	93.1	58.8	95.6	89.8	65.5	9.0	74.8	71.2
mod3	90.0	59.8	92.3	95.1	65.5	8.9	74.9	71.3
mod4	92.7	51.3	95.5	70.2	61.7	9.2	73.5	72.0
mod5	90.9	60.1	93.2	95.7	65.5	8.9	74.6	71.2
mod6	81.3	53.4	82.6	94.4	67.0	7.8	76.2	69.9
mod7	91.8	40.1	93.5	94.1	72.8	4.4	73.8	70.5
mod8	90.4	81.7	93.7	94.1	54.4	23.3	74.5	73.4
mod9	87.6	92.0	93.8	94.0	20.4	44.2	77.6	87.2

TABLE II
SHOT BOUNDARY DETECTION RESULTS

III. HIGH-LEVEL FEATURE EXTRACTION

As can be seen in Fig. 1, the high-level feature extraction system is composed of several successive processing steps. In the first step, videos are described by means of feature vectors. In the second step, a set of Support Vector Machines classifies the features in each vector as belonging or not to each category. In the third step, previous results are combined to generate the final score and the shots are ranked using this score.

A. Low-level descriptors

1) Color descriptors: Color features are the most popular visual features in the area of image retrieval, because colors are usually related to objects and scenes in images. Furthermore, color features are less dependent on the size, direction and view point of images compared to other visual features, which leads to high robustness. In our case, we use three different types of color descriptors.

a) Histogram: Color histograms [7] are applied in many image retrieval systems as a color feature. They describe the distribution of different colors in an image, while ignoring each color's spatial location.

We chose the HSV color space to extract histogram bins because the “V (Value)” value corresponding to the brightness component can be down-sampled more, so that a compact histogram can be obtained while the “H (Hue)” values, which represent the color information, can be more precisely quantized.

Digital image displays usually use the RGB color space to describe each pixel, so we need to convert the images into HSV space. We use 18 bins for the “H” channel, 3 bin for the “S (Saturation)” channel and 3 bins for the “V” channel. So totally we get 162 histogram bins.

b) Color moments: Color moments [8] have been successfully used in many retrieval systems, especially when the image contains just the object. The first order (mean), the second (variance) and the third order (skewness) color moments have been proved to be efficient and effective in representing color distributions of images. To be able to extract some local color features such as the sky, which usually lies in the upper part of an image, we divide an image into $k \times k$ blocks, and extract color moments from each image block. The final feature vector is obtained by concatenating the color moments extracted from the blocks, which results in a $9 \times k \times k$ feature vector. Here we set $k = 3$.

c) Color correlogram: The color correlogram [9] was proposed to characterize not only the color distributions of pixels, but also the spatial correlation of pairs of colors. The first and the second dimension of the three-dimensional histogram are the colors of pixel pairs and the third dimension is their spatial distance.

A color correlogram is a table indexed by color pairs, where the k-th entry for (i, j) specifies the probability of finding a pixel of color j at a distance k from a pixel of color i in the image. Let I represent the entire set of image pixels and $Ic(i)$ represent the set of pixels whose colors are $c(i)$.

If we consider all the possible combinations of color pairs the size of the color correlogram will be very large. Therefore a simplified version of the feature called the color autocorrelogram is often used instead. The color autocorrelogram only captures the spatial correlation between identical colors and thus reduces the dimension to $O(Nd)$.

2) Texture descriptors: Texture extractors allow us to describe images even in those cases where there exists no color information. In our system, we use three kinds of texture descriptors.

a) Co-occurrence texture: The implemented algorithm is based on the description given in [10]. In addition to this paper, [11] and [12] report the exact equations for the chosen five types of features extracted from the gray level co-occurrence matrix (GLCM): Entropy, Energy, Contrast, Correlation and Local homogeneity (also called homogeneity or inverse difference moment (IDM)).

Those features, as in [10], are extracted from 24 different GLCMs, in our case with 8 gray level bins (matrix with $8 \times 8 = 64$ bins), at different orientations and distances, although this number is adjustable to 12, 8, 6, 4, 3, 2 or 1 directions/distances. The set of offsets is as depicted in Fig. 2 and the resulting vector is $24 \times 5 = 120$-dimensional.

b) Wavelet texture grid: The implementation follows the description in [10], obtaining the variances of the high-frequency sub-bands of the Wavelet transform of each grid region. As an example, they use 12 sub-bands (4-level analysis), and so we do in our implementation. The used wavelet base function is the simple Haar wavelet while the grid has $4 \times 4 = 16$ regions. Thus, the resulting vector is $16 \times 12 = 192$-dimensional.

c) Edge histogram: For the edge histogram, 5 filters as proposed in the MPEG-7 standard are used to extract the kind

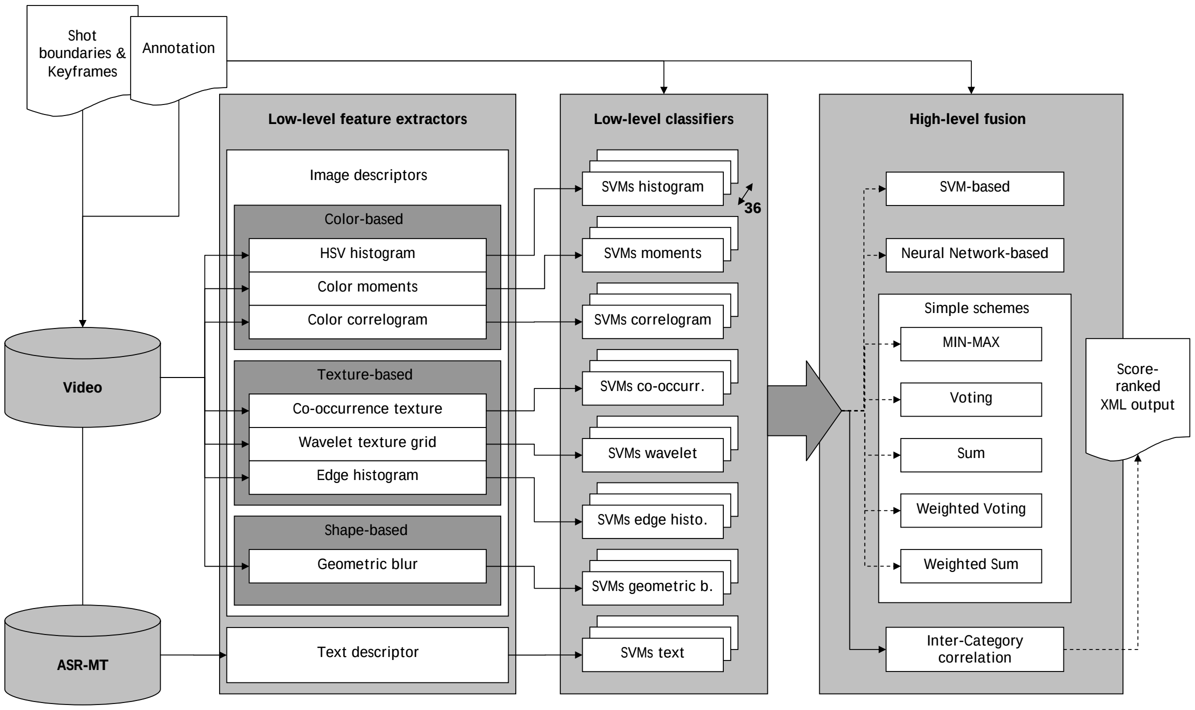

Fig. 1. Our high-level feature extraction system. Three processing steps are involved in the presentation of the results: low-level description of the shot, low-level individual descriptor classification and high-level fusion.

	24		20	
	23	21		18
	22		17	
	19	16	15	14
X	13			
1	7	10	11	12
	4		9	
	3	5		8
	2		6	

Fig. 2. Set of offsets used for computing the 24 GLCMs. X is a pixel in an image.

of edge in each region of 2×2 pixels. Then, those small regions are grouped in a certain number of areas (4 rows $\times$ 4 columns in our case) and the number of edges matched by each filter (vertical, horizontal, diagonal $45°$, diagonal $135°$ and non-directional) are counted in the region's histogram. Thus, the resulting vector is $4 \times 4 \times 5 = 80$-dimensional.

3) Shape descriptor: Color features are not sufficient for an image retrieval task, especially for object detection and, as mentioned with texture descriptors, in the case of retrieving monochrome images. Features based on shape information can be used in these cases.

a) Geometric blur: Geometric blur based features attempt to capture the local shape cues in images. The geometric blur of a feature signal is simply a convolution with a spatially varying kernel. The motivation is to provide robustness to variations in the position of features due to intra-class variation and small changes in pose.

To extract the geometric blur feature, we first extract four oriented edge features from a given image and randomly select 50 points where the edge energy is high. After geometric blur is applied on each of the four channels, we sampled 60 points on the concentric orbit of the 50 selected base points. The largest radius of the concentric circles is 50 pixels. Then a geometric blur descriptor is obtained by concatenating the 4 sub-sampled features on each channel. The similarity of two images is calculated by finding average minimal distances between the descriptors using the L2 distance.

With this method, we expect to gain confidence in those categories that represent objects. These categories are: Airplane, Animal, Boat-Ship, Building, Bus, Car, Face, Mountain and Person. We select 338 key-frames as the samples from these categories and the distances between a given image to all these samples to build a 338 vector for SVM training and testing.

4) Text descriptor: Classifying video shots based on the text spoken **??** is done in multiple steps. The required data for this stage is a list of shot-boundaries and a list of time stamped sentences. Research has shown [15] that often the critical words describing a shot are not spoken during the shot, so the recall rate can be increased by taking a bigger window than the shot size into consideration. The window size combined with this data produces a list of sentences associated with each shot. This can be turned into a list of shot feature vectors by calculating the relative word frequency of each word. Stemming and stop word filtering are used to reduce the dimension. An SVM is trained for each concept and then used for classification. All this processing is depicted in Fig. 3

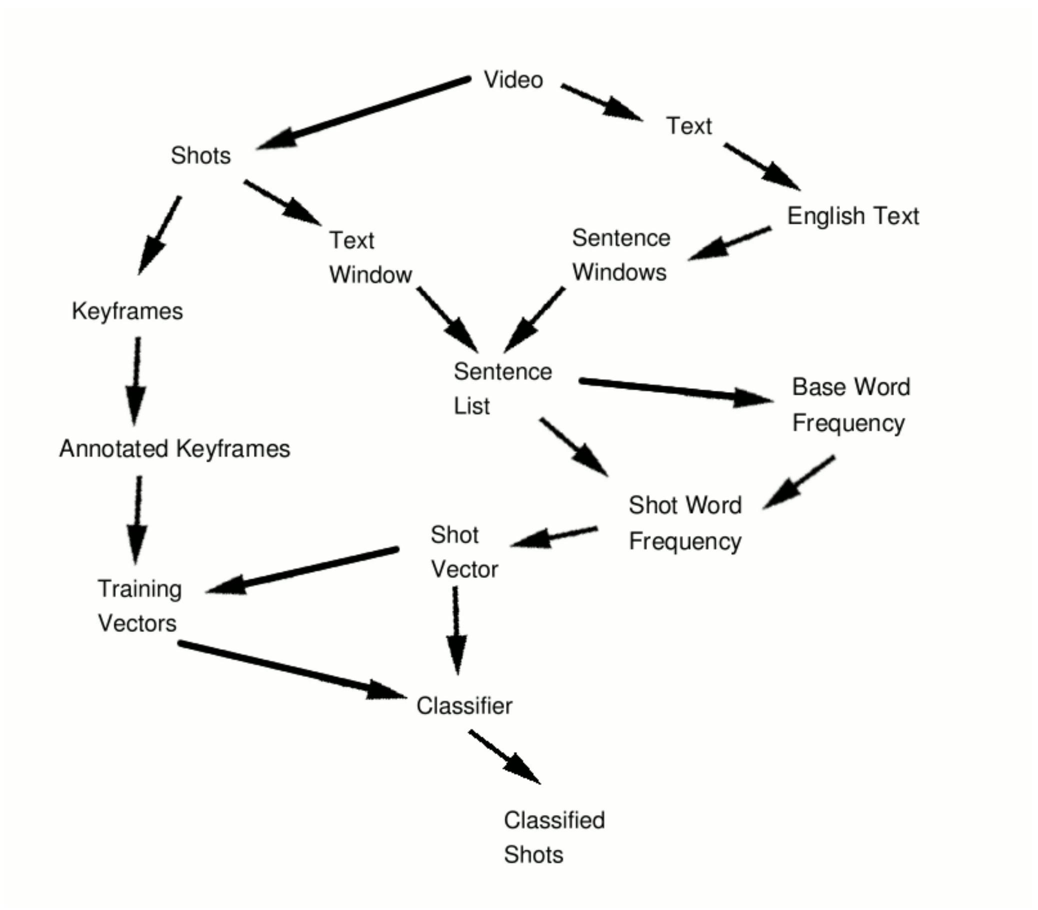

Fig. 3. Text descriptor processing chain.

a) Window size: The English translation of the text data for each video is supplied as a list of sentences with their respective starting and ending times. Each shot has a starting time x and finishing time y. The easiest way to generate a list of sentences associated with the shot is to include every sentence ending after x and starting before y. Because some important keywords are just outside the shot boundary the recall rate can be improved by including every sentence ending after $x - t$ and starting before $y + t$. If t is too small then important keywords will be missed, if t is too large, too much unassociated text will be included. Tested values of t were 0, 15, 20 and 60 seconds with 15 or 20 performing best.

b) Stemming: A stemmer finds the root or stem of a given word. For example "runs", "running", ... are forms of the word "run". Stemming is performed to reduce the size of the feature vector and to neutralize the grammar. For our purpose a "big gun" and"the biggest gun" will be counted as the same; "someone is shot" and "someone is shooting" both describe a similar scene. Two different methods of stemming were tested, rule based and dictionary lookup:

- *Rule Based Stemmer*. A rule based stemmer like the Porter Stemmer applies a series of rules to the word, like removing the 's' or 'ed' at the end.
- *Dictionary Lookup*. A giant lookup table is generated by processing a dictionary. This type of stemmer is only as good as the base dictionary. The wikimedia dictionary en.wiktionary.org is cc licensed and can be downloaded and easily mined.

c) Stop words: Stop words like 'and' or 'maybe' are frequently occurring words that mostly only have grammatical or syntactic meaning and do not contain any relevant information. These stop words are filtered out to reduce the feature vector size.

d) Shot feature vectors: Because a lot of words naturally occur frequently the important information that should be used to train our SVMs is not an absolute word frequency but the factor by how much it changes compared to our base word count.

For each of the n words w_i we can count its total occurrences $\#w_i$ and its occurrences in each document d_i $\#w_{i,j}$.

$GTF(w_i) = \#w_i$ per 10^6 words in the base language

$TF(w_i, d_j) = \#w_{i,j}$ per 10^6 words in document d_i

This allows us to compute the feature vector $\overrightarrow{f_j} = (f_{1,j}, f_{2,j}, ..., f_{n,j})$.

$$f_{i,j} = \frac{TF(w_i, d_j)}{GTF(w_i)} (= RTF) \tag{1}$$

We discard $f_{i,j} < t$, so that our vector is sparse.

B. High-level fusion

We get a score from each low level feature classifier; this score indicates the confidence of the corresponding category.

We combine these low level scores in order to improve the results. The following fusion approaches are considered:

1) *SVM*. The normalized scores resulting from the low level feature classifiers for each frame are concatenated to a multi-confidence vector. These score vectors are then used to train a high level SVM classifier. The final confidence scores result from this high level classifier.
2) *Neural network*. Similarly to fusion with an SVM classifier, we also build a neural network for each category with one hidden layer to learn and evaluate the low level scores. The number of hidden units in the hidden layer of each network is tuned by a grid search.
3) *MIN-MAX*. The MIN-MAX rule is a simple fusion strategy. We only consider the score s, which has the largest distance to the border of judgment b. If $s - b$ is positive, then judgment will be positive, too. Otherwise we consider it a negative sample.
4) *Voting*. This simple rule judges through voting the scores in the low level score vector. The scores of the majority determine the final judgment.
5) *Sum*. All scores in a score vector are added together and the result is the combined confidence score.
6) *Weighted voting and weighted sum*. Assign a weight for each score in a score vector. This weight can be the normalized *InfAP* (inferred average precision) for each low level feature classifier (each feature extractor and each category). This was inspired by the fact that some low level features are more suitable and descriptive for some categories. For example, the category "Sky" should be represented more precisely by color information, etc.
7) *Correlation*. Among the 39 categories, there is a correlation between semantically correlated categories. A frame that contains "Boats" will most probably contain the category "Water", for instance. We represent these correlation relations as a correlation matrix. The elements in this matrix are the conditional probabilities $P(A|B)$ of a category A given a certain category B. This correlation matrix helps us to the analyze the confidence scores in a semantic way.

C. Experiments and results

1) Development: The main problem we had to deal with was the training of the SVMs, both for low-level classification and SVM-based high-level fusion.

We divided the development data set in three parts. The first part is for low-level feature learning, the second part is for low-level feature evaluation and fusion score learning, and finally the third part is for fusion evaluation. Through permutation, we made a 6-fold cross validation on these 3 parts of dataset. We optimized the parameters for fusion classifiers through this 6-fold cross validation and selected the classifier from the fold that resulted in the best mean *InfAP* score as the final classifier.

The used kernel function in all cases is an RBF (Gaussian function) with default parameter $\gamma = 1$. All the parameters in the learning process are the default ones in *SVM-light*, except for a parameter responsible of controlling the cost margin for positive examples. This is set to the number of negative samples in the training set divided by the number of positive samples. The idea was extracted from [13] and it aims at achieving better decision thresholds with unbalanced training sets.

Run	Mean InfAP	Description
UKA1	2.1%	Neural networks without ASR-MT
UKA2	2.2%	Sum with ASR-MT
UKA3	2.3%	Sum without ASR-MT
UKA4	0.6%	Voting with ASR-MT
UKA5	1.2%	SVM fusion with ASR-MT
UKA6	2.2%	Neural networks with correlation, w/o ASR-MT

TABLE III
HIGH-LEVEL FEATURE EXTRACTION RESULTS

2) Submission for TRECVID: Our submission for TRECVID consists of six runs, each of them featuring a different high-level fusion scheme or applying some further combination of the low-level classification. The list is:

1) *UKA1*. Neural networks-based high-level fusion, using only image information
2) *UKA2*. Simple fusion by sum of low-level scores, using both image and text information
3) *UKA3*. Simple fusion by sum of low-level scores, using only image information
4) *UKA4*. Simple fusion by voting of low-level scores, using both image and text information
5) *UKA5*. SVM-based high-level fusion using both image and text information
6) *UKA6*. Neural networks-based high-level fusion with simple correlation module, using only image information

The first five runs were ordered by the mean *InfAP* obtained with the mentioned division of data. The sixth run, unfortunately, could not be evaluated and that's why it appeared in the 6th place. The obtained results are grouped in Table III.

From the table, it can be seen that text features didn't enhance the results achieved by using just image features. The basic correlation module slightly enhances the performance, although not dramatically. These results encourage us to further exploit inter-category relationships in order to achieve better improvements in the future.

IV. CONCLUSIONS

In our first participation in the TRECVID evalualtions, our goals were to develop a common software infrastructure for doing multimedia retrieval and to develop baseline systems for shot boundary detection and high-level feature extraction. These goals were achieved: our shot boundary system is performing well in most areas and our system for high-level feature extraction performs around the average level of all participants.

There is still room for improvement in both systems. The shot boundary detection system would benefit from a better dissolve detector and detailed analysis of the TRECVID results will help us to find improvements for the other detectors.

For the high-level feature extraction system, an area for improvement is the optimization of the SVM parameters, which is the main reason for the comparatively low performance of our low-level feature classifiers. The fusion of the visual and textual cues could also be weighted according to the corresponding performance in each category, while in our current system they are always weighted equally.

References

[1] Z. Liu, D. Gibbon, E. Zavesky, B. Shahraray, P. Haffner, "AT&T Research at TRECVID 2006," *NIST TRECVID Workshop*, Gaithersburg, USA, Nov. 2006.

[2] R. Lienhart, "Comparison of Automatic Shot Boundary Detection Algorithms," *Proc. SPIE Storage and Retrieval for Image and Video Databases*, vol. 3656, pp. 290–301, Jan. 1999.

[3] R. Lienhart, "Reliable transition detection in videos: A survey and practitioners guide," *Int. Journal Image and Graphics*, vol. 1, no. 3, pp. 469–486, Aug. 2001.

[4] A.C. Berg, T.L. Berg, J. Malik, "Shape matching and object recognition using low distortion correspondences," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[5] F. Bellard, M. Niedermayer et al., "The FFMpeg Project," `http://ffmpeg.sf.net/`.

[6] S. Ayache, J. Gensel, G.M. Quénot, "CLIPS-LSR Experiments at TRECVID 2006," *NIST TRECVID Workshop*, Gaithersburg, USA, Nov. 2006.

[7] M.J. Swain, D.H. Ballard, "Color Indexing," *Int. Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[8] M. Stricker, M. Orengo, "Similarity of color images," *Proc. SPIE Storage and Retrieval for Image and Video Databases*, vol. 2420, pp. 381–392, San Jose, USA, Feb. 1995.

[9] J. Huang, S.R. Kumar, M. Mitra, W.-J. Zhu, R. Zabih, "Image indexing using color correlograms," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 762–768, San Juan, 1997.

[10] M. Campbell, A. Haubold, S. Ebadollahi et al., "IBM Research TRECVID-2006 Video Retrieval System," *NIST TRECVID Workshop*, Gaithersburg, USA, Nov. 2006.

[11] M. Partio, B. Cramariuc, M. Gabbouj, A. Visa, "Rock texture retrieval using gray level co-occurrence matrix," *Proc. 5th Nordic Signal Processing Symposium*, Oct. 2002.

[12] D. Gadkari, "Image quality analysis using GLCM," M.Sc. Thesis, B.S.E.E. University of Pune, 2000.

[13] K. Morik, P. Brockhausen, T. Joachims, "Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring," *Proc. 16th Int'l Conf. on Machine Learning*, 1999.

[14] J. Yang, M. Y. Chen, A. G. Hauptmann, "Finding person X: Correlating names with visual appearances," *Proc. Conf. Image and Video Retrieval (CIVR04)*, Ireland, 2004.

[15] M. Fuller, J. Zobel, "Conflation-Based Comparison of Stemming Algorithms," *Proc. 3rd Australian Document Computing Symposium*, Sydney, Australia, 1998.

[16] T. Joachims, "Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms," Kluwer/Springer, 2002.

[17] M. Huijbregts, R. Ordelman, F. de Jong, "Annotation of Heterogeneous Multimedia Content Using Automatic Speech Recognition," *Proc. 2nd Int. Conf. Semantics And Digital Media Technologies (SAMT)*, Dec. 2007.