TZI Bremen - Trecvid 2006 high level feature extraction

A. Bruckmann B. Lerbs D. Gao J. Eidtmann L. Mozogovenko M. Buczilowski T. Jughardt Y. Xu A. Jacobs A. Lüdtke O. Herzog
TZI - Center for Computing Technologies, University of Bremen,
Am Fallturm 1, D-28359 Bremen, Germany
{andreasb|blerbs|gaodanxy|eidtmann|margo|ptmc|juggi|sinead|jarne|ludi|herzog}@tzi.de

Abstract

In this paper, the system developed by the University of Bremen for participation in the Trecvid 2006 high-level feature extraction task is presented. Six runs have been submitted, each of them incorporating a different combination of three classifiers based on image, sound, and text features. For the feature *Corporate Leader*, above-average results could be achieved. Results are shown and differences between the runs are discussed.

1 Introduction

The next pages will describe the techniques the University of Bremen has used to generate a video analysis for the Trecvid contest.
This paper is for people interested in picture, sound and video analyzing technologies. The documents starts with a quick overview of all runs, followed by a description of all used classifiers. Then, the methods used for fusion of the different classifiers are introduced. The paper closes with a discussion of the results.

2 Submitted runs

To get a good overview about how well our different classifiers perform we created six different runs. Every run has a different set of classifiers enabled. There are three classifiers based on image, sound, and text. You can find detailed information about every classifier later in this document in sec. 3.
The following table shows the different runs, their IDs and a small description:

- TZI_Text - Within this run only the text classification module was used
- TZI_RelaxText - Within this run the text classification module was used, taking into account temporal dependencies and dependencies between different high-level features using an approach based on probabilistic relaxation
- TZI_Avg - All available classifiers were enabled, combined with a performance-weighted average fusion
- TZI_RlxAll - Here every classifier combined with a fusion method based on probabilis-

tic relaxation, taking also into account temporal dependencies and dependencies between different features

- TZI_Image - This run incorporates the image-based classifier only
- TZI_RelaxImage - Within this run the image-based classifier was used, accounting for temporal dependencies and dependencies between different high-level features using the probabilistic-relaxation-based approach

In the following section the different classifiers will be described in detail. The above-mentioned fusion methods will be described in sec. 4.

3 Classifiers

We used more than one classifier to analyze the videos. In fact we used three different classifiers based on images, sound, and text. The results of these three classifier are combined later in a classifier fusion step.

3.1 Image features

The image feature classifier uses every 20th frame of a video to do its calculations. Within these calculations filters are used to calculate a probability for every high level feature.

3.1.1 Region of interest filter

The region of interest (ROI) filter is a special filter that can only be used in combination with other filters. We use it in combination with the color histogram (for 64 and 512 colors) and the text detection filter, for the detection of the *Map* high-level feature only. As a preprocessing filter it tries to cut out an important region of the current frame so that the next calculation step, e.g., a color histogram, will operate only on that region. It is intended to be used for TV screens built up with a picture inside picture method, e.g., a screen showing a weather map in the main area but also showing a temperature info box at the bottom and a "Weatherman" box on the left side. A map detection filter based upon a color histogram of the whole image would include all the colors of the info boxes and the weatherman, although these colors might be less important for finding maps. In this case the ROI filter is used to separate the map's main screen, so that the map filter will only build a histogram based upon the actual map. This operation isn't always clear, because it is a matter of interpretation which region of the screen will be important for a filter. We have parameterized the ROI filter in a way that minimizes the rate of falsely detected regions on an internal test set.

Filter implementation The ROI filter was the result of some experiments with edge detection mechanisms. In the beginning we planned to use an edge detection algorithm for finding country borders inside maps, but this turned out to be not very successful. The only noticeable schemes on the edge images of maps we examined were borders of the picture inside picture screens and the edges of information boxes. We thus changed our strategy to create a filter detecting different areas on screen. The ROI filter is searching for horizontal and vertical line segments and decides upon some definitions (i.e., percentage of line pixels, length of combined line pixels etc.) if these segments are parts of a long line. If a line has reached a parameterized size it is taken to the next processing step. In that step, the filter tries to build rectangles based upon the detected lines. The size and the position of these rectangles is used to determine the one rectangle which is returned as the region of interest.

3.1.2 Color correlogram

This filter is an implementation of a standard color correlogram filter [3]. A color correlogram (henceforth correlogram) expresses how the spatial correlation of pairs of colors changes with distance.
We used the correlogram for the high-level features *Charts*, *Desert*, *Explosion/Fire*, and *Maps*.

3.1.3 Color histogram filter

The color histogram filter reduces the amount of colors by removing a parameterizable number of bits inside every RGB color channel. We use color histograms with sizes of 4 or 8 bins per dimension (R, G, and B), for a total of 64 or 512 bins, respectively. These histograms are used either with or without the ROI filter (see above).
We used the color histogram for the high-level features *Airplane*, *Charts*, *Corporate Leader*, *Entertainment*, *Military*, *Mountain* (64 bins respectively), *Animal*, *Car*, *Desert*, *Explosion/Fire*, *Maps*, *Police/Security*, *Truck*, *Waterscape/Waterfront*, and *Weather* (512 bins respectively).

3.1.4 Text detection filter

The text detection filter is based on an algorithm developed in a Diploma thesis at the University of Bremen [9]. It uses an edge detection filter tailored for overlayed text to find single characters and searches for regions where many characters appear on a line. It tries to find words and sentences by their typical spatial distribution and returns the positions of these inside an image.
We adapted the algorithm's parameters to values we found to work good for finding names of locations on maps on an internal test set.
We used the text detection filter only for the *Maps* high-level feature.

3.1.5 US-flag detector

A very important criterion to find US-flags in an image are its characteristic colors and their spatial arrangement. Our US-flag filter is therefore searching for neighbored red and white areas. We define a 5×5 area as white respectively red, if it contains mostly red or white pixels. Our definition of red and white is explained later in more detail. The blue stars of the US-flag are ignored, because on nearly every picture we examined containing a US-flag the blue parts were hidden or neglectably small.
For better color separation the picture is first converted to HSV color space. We mark a pixel as red if its hue (H) lies between 340 and 20 degrees. We mark a pixel as white if saturation (S) is low and luminance (V) is relatively high.
The next thing to look at is the distribution of the mostly red and mostly white areas. We count all red and white 5×5 areas that lie next to each other. The combination of red, white, and red and white areas is counted, too. The higher these counts, the more probable it is, that a given image contains an US-flag. The filter is used specifically to detect the *US-flag* high-level features.

3.1.6 Face and person detector

We use an implementation of the algorithm by Lienhart and Maydt [5] to detect persons and frontal faces in an image. We use the face detector for the high-level features *Entertainment*, *Face*, *Police/Security* and *Weather*. We use the person detector for the features *Entertainment*, *Military*, *Office*, *Police/Security*, and *Weather*.

3.1.7 Edge direction histogram

This filter is an implementation of a standard edge direction histogram by H. Tamura et al.[8]. We use it to analyze an image for directed textures. Tamura's directionality criterion is characterized by a histogram based on eight orientations. The edge direction histogram is used for the high-level features *Desert*, *US-flag*, *Mountain*, *Truck*, and *Waterscape/Waterfront*.

3.1.8 Image feature classification

The Trecvid 2005 common annotation set was separated into two different sets for every feature, an internal training set and an internal test set. The training set was used for building classification models, while the internal test set was used to validate these models.
We use a support vector machine (SVM) in the form of the SVM-light software [4] to train one model for each Trecvid high-level feature, based on our image filter results as described in the previous section. As a kernel we chose the radial basis function (RBF) kernel. We set the relative significance of positive vs. negative examples to $\frac{positive\_examples}{negative\_examples}$ to account for a low number of positive examples. In our validation process, we vary the variance parameter, which takes values 2^n with $n \in \{-4...4\}, n \in Z$. The trade-off between training error and margin is also varied and takes values 2^m with $m \in \{-1...4\}, n \in Z$. We validate a model created with given parameters on our internal test set, using the *F-measure* ($F{-}measure = \frac{2 \times recall \times precision}{recall + precision}$). We retain the model with the highest F-measure
The validation process is aware of how often a feature occurs. In case that the feature occurs frequently a big amount of positive and a small amount of negative examples are used, while in case that the feature occurs infrequently it is the opposite.

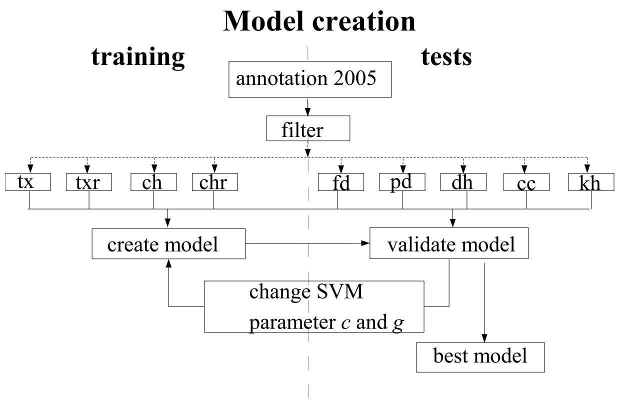

Figure 1: Model building process

Figure 1 shows a schema of the model building process. The abbreviations in the figure have the following meaning:

- tx - Text detection
- txr - Text detection with ROI filter
- ch - Color histogram
- chr - Color histogram with ROI filter
- fd - Face detector
- pd - Person detector
- dh - Edge direction histogram
- cc - Color correlogram

The best validating models are then applied to the Trecvid 2006 test set. A schematic representation of the process can be seen in fig. 2. On

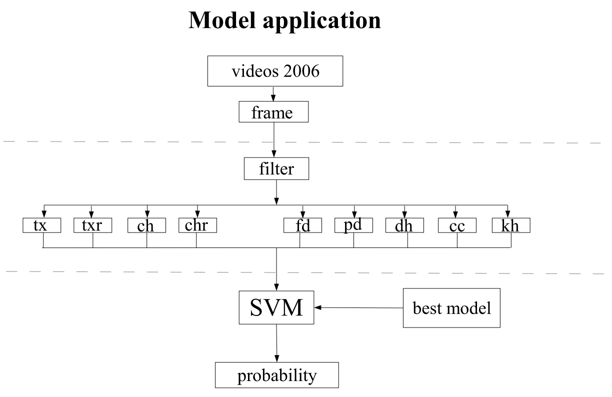

Figure 2: Analyzing the Trecvid 2006 test set

the 2006 test set we analyzed every 20th frame and returned the best positive result in a shot as the result for that shot.

3.2 Audio-based classifier

The audio classifier searches the audio tracks of the Trecvid collection for a number of previously learned sounds. We trained models for gunshots, outdoor, applause and person on a manually selected subset of the Trecvid development data and applied the resulting classifiers to the Trecvid 2006 test set. We then applied a simple mapping from detected sounds to a subset of the LSCOM-lite high-level features: If a gunshot was detected in a shot, the feature *Military* was assumed to be present. The same goes for applause, which was used to detect the feature *Crowd*. The features *Person* and *Outdoor* are directly mapped from the corresponding detected sounds.
The classifier is built up of two stages. In the first stage we extract spectral features from the audio tracks. In the second step we use a supervised learning algorithm for training and prediction. Our approach is based on the algorithm proposed by Hoiem et al. [2] but it differs in classifying. While Hoiem et al. suggest a decision tree classifier, we chose to use a support vector machine (SVM) in the classifier stage, in the form of the Libsvm library [1].
We will first describe the feature extraction step, followed by a description of our selection of training data and the training of a support vector machine.

3.2.1 Audio feature extraction

The first step of the sound feature extraction module is to create an abstract feature representation of the audio signal using an FFT on a temporally shifted window of the audio track. From the spectral representation, a set of 63 descriptive features is computed [2] to serve as input to the classifier. The size of the window is dependent of the type of sound that should be detected. The longer the sound, the bigger the window. We use 800 milliseconds for gunshots, 1200 milliseconds for applause. When applying the final classifiers to the test set, the window is shifted in steps of 100 milliseconds. For training, we manually cut a set of training sounds for each type of sound to be detected.

3.2.2 SVM classification

We found that the Trecvid 2005 common annotation data was not detailed enough to be able to automatically generate an appropriate training set. Thus, we chose to manually create a training set for the sound we wanted to detect. We cut a small number of short example sounds between 0.5 and 2.5 seconds from the Trecvid 2005 development set, including all the disturbing sounds that might be in the background. The manual searching and cutting of sample sounds

takes a long time, but is in the end the only way to ensure that the system learns the right type of sounds. It turned out that the selection type and also the number of training sounds has a great effect on the prediction quality of the SVM. During the testing of the system for various videos, the prediction sometimes returned very few or no results, even if the news contained plenty of the regarded sound events. The reason for this was probably that the analyzed sounds were too different from the training examples. However, finding good training examples that cover the whole variance of guns, explosions in video material is hard to manage, and it is very hard to cut the sounds from the test material.
Our solution to this problem was to lower the threshold on the prediction values yielded by the SVM classifier, such that not only positive predictions are counted, but also negative results down to -0.4 or lower. That way, we reached a much bigger number of positives.

As a third modality we integrated a classifier based on spoken text, which will be described in the following section.

3.3 Text-based classifier

Our text-based classifier consists of two stages. First, a fixed-dimensional feature vector is computed based on the words spoken in a shot and its temporal vicinity. Then, for each of the 39 LSCOM-lite high-level features, a two-class SVM classifier is applied to the shot's feature vector. Thus, our text-based classifier is based on the shot as atomic unit, using the TRECVID common shot boundary reference [7].
In this section, we will first describe the generation of the feature vector, followed by a description of the training process used to setup the 39 SVM classifiers.

3.3.1 Text features

The basis for our text features are the results of the automatic speech recognition (ASR) given by the *Linguistic Data Consortium.* In the training process, we used the data from the TRECVID 2005 development set. For classification of the 2006 test set, we used the corresponding 2006 ASR output.
All three languages are processed separately. For english videos, we directly use ASR output. For chinese and arabic videos, we used the corresponding machine translation (MT) output instead. Thus, we use english text as the basis for all videos, regardless of their original language. We use, however, different vocabularies and thus a slightly different feature vector generation process for different languages. This influences the subsequent classification stage, for each high-level feature has to have a classifier trained for each of the three languages.
The text feature vector for a given shot and a given language is built up in four steps. First, we remove common english stop words. Then, an english stemming algorithm is applied to the remaining words. The number of different english stems occuring in all 2005 videos of the given language determines the vocabulary size and thus the feature vector dimensionality for this language. In the next step, we count the stems occuring in the given shot and its temporal vicinity (within four seconds before the start and four seconds after the end of the shot). This yields a (sparse) *word frequency* vector. In the last step, the word frequency vector is binarized. The binary word frequency vector is used as input for the SVM classifier described in the next section.

The text feature vector generation for TRECVID 2006 shots is done in the same way as for the 2005 data. If word stems occur in the ASR/MT that are not in the vocabulary of the corresponding language (which is based on 2005 data), they are simply discarded.

3.3.2 SVM classification

Classification of the text feature vectors is done using Support Vector Machines (SVM), using the LIBSVM library [1]. We built a total of three times 39 classifiers, one for each LSCOM-lite feature for each of the three languages. For each classifier, we partitioned the TRECVID 2005 data of the corresponding language into a training set and a validation set. Using a linear SVM kernel we then did a grid search over the SVM cost parameter C, training on our training set, and validating on our validation set. We picked the classifier with the highest *average precision* on the validation set to be used with the TRECVID 2006 test data. The TRECVID 2005 common annotation data was used for training and validation.

In the training process, only shots with a minimum of three spoken words were taken into account. The SVM class weights were set according to the distribution of present vs. not present high-level features in our training set.

In addition to the 39 LSCOM-lite features we also created classifiers for the two LSCOM [6] features *Commercial Advertisement* and *Politics*, based on the LSCOM annotation on TRECVID 2005 development data.

4 Classifier fusion

To combine the results of our different classifiers (based on image, sound, and text features), we used different classifier fusion techniques. The first fusion method we employ is performance-weighted average. Knowing that only a subset of the 39 LSCOM-lite features would be evaluated by NIST, we employed another technique to take into account possible dependencies between features, based on *probabilistic relaxation labelling.* This technique also takes into account temporal dependencies. This is motivated by the observation that most of the LSCOM-lite features are related to each other in some way. E.g., the presence of a *face* strongly coincides with the presence of a *person.* By analyzing the 2005 development annotation data, we could confirm that observation. Our hope was to improve the results of other classifiers if we would have a good performing classifier for a related feature.

In the fusion step, only classifiers that address the same high-level feature are fused. For each TRECVID feature, we have up to three classifiers (image-, sound-, and text-based). The maximum number of classifiers to be fused is therefore three.

4.1 Performance-weighted average

Each classifier in the previous section yields a probability estimate between zero and one for a given shot. We also have access to several performance criteria for each classifier, based on the internal test/validation set. We use a classifier's *F-measure* on the validation set to weight its probability estimate in relation to other classifiers for the same high-level feature.

4.2 Probabilistic relaxation labelling

Relaxation labelling is the process of assigning labels to different objects according to a priori knowledge about the compatibility of different

objects' labels and certain observed object features.
We use relaxation labelling as a means to fuse different classifiers for the same high-level feature, to take into account dependencies between different high-level features in the same shot (e.g., presence of a face and presence of a person), and to consider temporal dependencies between consecutive shots.

4.2.1 Fusion of different classifiers

To fuse n different classifiers for the same high-level feature, we use $n + 1$ objects in the relaxation process, with two labels each (present and not present). One object represents the fusion result. It is linked to each of the other n objects, which represent the classifiers' results. The labelling probabilities of the classifier objects are initialized with the classifiers' results. The fusion result object's labelling probabilities are initialized with a constant, 0.5 in our case. The compatibility coefficients between classifier results and fusion result are computed according to Yamamoto [10], based on the conditional probabilities of the high-level feature being present given the classifier says it is present. These can be directly computed from the performance of the respective classifier on our validation set (e.g., probability for a feature being present if a classifier says it is present equals the classifier's *precision*).

4.2.2 Feature dependencies

To model interdependencies between different high-level features in the same shot, we assign each high-level feature to one object in the relaxation process. The labelling probabilities of these objects are computed through classifier fusion as described in the previous section. In the case of only one available classifier result for a high-level feature (as is in runs TZI_RelaxText and TZI_RelaxImage), the labelling probabilities are taken directly from that classifier's result.
Every object is linked to every other object, with a compatibility coefficient computed according to Yamamoto [10], based on the conditional probabilities of the two high-level features computed using the TRECVID 2005 common annotation data.
For the runs including text classifier results, there are 41 objects (LSCOM-lite features plus *Politics* and *Commercial Advertisement*), in the run only including image classifiers (TZI_RelaxImage), there are 39 objects, one for each LSCOM-lite concept.

4.2.3 Temporal dependencies

Temporal dependencies are modelled in the same way as dependencies between different features, with a difference in computing the compatibility coefficients. Here, the conditional probability is computed on subsequent shots. To model temporal dependencies in the relaxation process, a shot (represented by its 39/41 objects) is linked to its predecessor and its successor (represented by their 39/41 objects).

5 Results

Figure 3 shows the TZI results compared to the median performance and the best performance among all Trecvid 2006 participants. The performance of the different TZI runs will be discussed in the following sections.

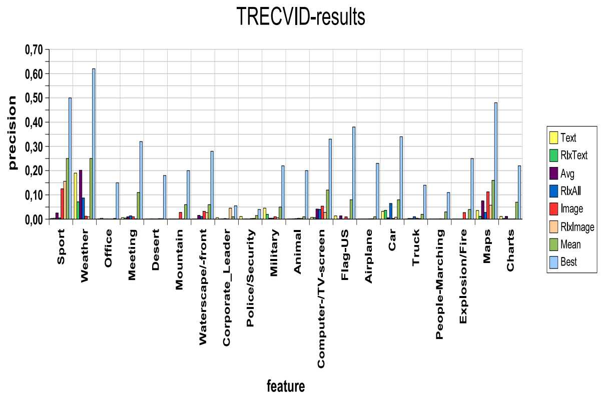

Figure 3: Results for all evaluated features: The six TZI runs, the median performance, and the best performance among all participants

5.1 Text classifier

The text classifier worked best for the features *Weather* and *Map*, which is what we expected based on our experiments with the Trecvid 2005 development set. It is easy to imagine that special (trained) words occur frequently in Weather reports, and maps are often linked to weather news. The next best features are *Military* and *Car*. The letter is unexpected and might be by chance. The overall results are not as good as we expected after experiments with the english Trecvid 2005 data. We suspect that the results for arabic and chinese videos, which were simply merged with the english results, but were based on different classifiers, may have worsen the results, due to the low number of positive examples in arabic and chinese development data in comparison to the english data.

5.2 Image classifier

In developing the image classifiers, we first focussed on the features *Waterscape/waterfront*, *US-Flag*, and *Map*. Due to the change in the Trecvid high-level feature extraction task, we had to extend our focus to the other 36 feature and create a more general system. The *Map* feature is among our best results, which is what we expected due to our above-mentioned focus in the beginning of the development. The best performing image-based classifier is the one for the *Sport* feature, which we expect to be due to the characteristic coloring of sport scenes.

5.3 Performance-weighted average fusion

The weighted average fusion classifier was only in one case significantly better than the best single classifier, in the case of the *Weather* feature. In all other cases, the weighted average performance was between our best performance and the worst for that feature. This suggests, that the positive results returned by the different classifier were disjunctive. This would suggest the use of another fusion algorithm, e.g., a *Maximum* fusion.

5.4 Relaxation-based fusion

The results with relaxation are a bit disappointing. There are situations where relaxation helped to achieve higher precisions for some features but there are also situations where the precision was less because of relaxation. In some small experiments we did with the temporal relaxation, however, we came to the conclusion that it can help a lot for features that appear in temporal blocks, like, e.g., *Commercial advertisement*. In the future, it might help to modify the relaxation process in a way that only relates features that are dependent in a statistically significant manner.

5.5 Overall comparison

For most features, our results are below the median and for some features, especially those we have not concentrated on, the results are not satisfying, so that we come to the conclusion that here are still a lot of things to do to reach the current state of the art.

References

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] D. Hoiem, Y. Ke, and R. Sukthankar. Solar: Sound object localization and retrieval in complex audio environments. March 2005.

[3] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 762, Washington, DC, USA, 1997. IEEE Computer Society.

[4] T. Joachims. *Making large-Scale SVM Learning Practical.* MIT-Press, 1999.

[5] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *ICIP (1)*, pages 900–903, 2002.

[6] Milind Naphade, John R. Smith, Jelena Tesic, Shih-Fu Chang, Winston Hsu, Lyndon Kennedy, Alexander Hauptmann, and Jon Curtis. Large-scale concept ontology for multimedia. *IEEE MultiMedia*, 13(3):86–91, 2006.

[7] C. Petersohn. Fraunhofer hhi at trecvid 2004: Shot boundary detection system. In *TREC Video Retrieval Evaluation Online Proceedings*, 2004.

[8] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Textural features corresponding to visual perception. *IEEE Trans. Syst., Man, Cyb.*, 8(6):460–473, 1978.

[9] N. Wilkens. Detektion von videoframes mit texteinblendungen in echtzeit. Master's thesis, Universität Bremen, 2003.

[10] K. Yamamoto. A method for deriving compatibility coefficients for relaxation operators. 10:256–271, 1979.