ITEC-UNIKLU
Known-Item Search Submission

M. Lux K. Schoeffmann M. del Fabro M. Kogler M. Taschwer *

Feb 28, 2011

Abstract

In this article we describe our approach to the known-item search task for TRECVID 2010. We describe how we index available metadata and speech transcriptions and how we cope with three different content-based search modules, which take global video motion, local video motion, color, and edges into account.

1 Introduction

For the TRECVID 2010 known-item search task we were provided with about 8,500 videos including annotations and a set of search topics, each of which defining a single known item (video) within the data set. Our task was to maximize the inverted rank of the correct result. For development purposes we obtained a set of 122 sample topics along with correct answers.

The availability of metadata for nearly all of the video files motivated the use of text-based search. However, some of the videos had no annotation at all and the quality and extent of annotation varied significantly within the video data set. So our approach was to perform a multi-step search. First, possible candidates were found using text-based search (see Section 2.1). Under the assumption that these candidates represented the search topic visually, we used the highest ranked results from text search to create queries for content-based retrieval in different feature spaces. Using these queries we obtained multiple ranked lists of results, which were then merged to a single result list. With this approach we hoped to increase the recall of our search engine by making videos with low quality metadata searchable.

After this introduction we present a short overview on the general architecture of our search engine and then describe the text- and content-based search modules. We then outline the approach of creating queries for the different content-based feature spaces and the merging of result lists. Finally we present the results of our approach and conclude our work.

2 Known-Item Search Approach

For our approach we identified the combination of different feature spaces as main challenge. Based on prior research and development we selected best candidates for content-based retrieval (see Section 2.2) and designed a search engine architecture that integrated available metadata and content-based search.

An overview of the architecture is given in Fig. 1. First step is to obtain n results from the text search module. Based on these results queries for each of the content-based search modules (see Sections 2.2 and 2.3) are generated. Finally, the obtained result lists are merged into a single ranked list.

*ITEC, Klagenfurt University, Austria, contact: mlux@itec.uni-klu.ac.at

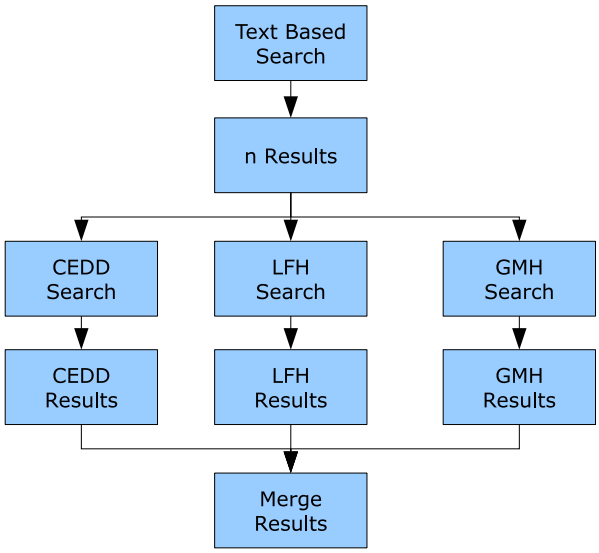


Figure 1: Architecture of our approach for the known-item search task. The results of the text search are used to create queries for the content-based search. The content-based search modules (CEDD, LFH, and GMH) are explained in the respective subsections of Section 2.2.

Event	**Added terms**
If there is a speech transcript	talk talking speech speaking speak
Male speaker recognized	male guy man
Female speaker recognized	female woman girl lady
Spanish language recognized	spanish, spain
Dutch language recognized	dutch, netherlands, holland
Chinese language recognized	chinese, china
Russian language recognized	russian, russia
French language recognized	french, france
Arabic language recognized	arabic, arabian, arab

Table 1: Terms added to the index in case of recognized events in the provided speech transcripts.

2.1 Text-based Search

More than 99% of the videos in the test data set featured usable structured text annotations. For the remaining small number of videos the annotations just contained an HTML error message. Metadata fields were heterogeneous as were the values. Most of the videos had a subject metadata field. Nearly half of them had creator, source and color fields and only a few of the video annotations featured notes, keywords or country fields. Due to the heterogeneous data we chose to merge all of the available text into one single field for indexing.

Besides the metadata from the provided annotations we also indexed the provided speech transcripts. To focus on the more reliable portions of the speech transcripts we chose to index only terms with a recognition score $r > 50\%$. Furthermore we utilized the speaker and language recognition and added terms based on the results of speech recognition as defined by Table 1.

Based on the collected text from annotation, speech transcripts and speaker and language recognition we created a merged document to be put into an inverted index. Pre-processing of text was reduced to a minimum as we just converted all the text to lower case and tokenized the documents based on punctuation

and whitespace. The resulting tokens (terms) were stored in an inverted index to allow for fast and efficient retrieval based on term frequency and inverse document frequency (tf*idf).

Search was implemented using the Lucene search engine[1]. For indexing the Lucene 3.0 standard analyzer was used and for scoring the built-in relevance function was utilized without modifications as we noticed that modification of analyzer and scoring lead to worse results during testing.

2.2 Content-based Search

2.2.1 Color and Edge Directivity Descriptor (CEDD)

The color and edge directivity descriptor is a joint histogram utilizing a fuzzy color scheme and an edge direction histogram [2]. Due to its small footprint it can be used very efficiently for retrieval in large image databases. For our approach we used the provided shot detection results. We extracted a median keyframe from each shot of each video and computed the CEDD feature. The Tanimoto coefficient was then used to rank keyframes according to an input keyframe. We employed the Lucene Image Retrieval library[2] (LIRe) to provide efficient indexing and search of the extracted keyframes based on the CEDD descriptor [3].

2.2.2 Local Feature Histogram (LFH)

The creation of *Local Feature Histograms* consists of three steps:

Step-1: Computation of forward-predicted motion vectors on a per-frame basis

We use a motion estimation algorithm similar to that of x264 [1]. The main difference is, however, that (1) we compute motion vectors for every 16x16 block of pixels[3] and (2) compute forward-predicted motion vectors only. These forward-predicted motion vectors are always computed on a per-frame basis, i.e. from frame $n-1$ to frame n. As Su et al. noted [5], motion vectors represent the sum of camera motion, object motion, and noise introduced by the motion estimation algorithm of the encoder. Consequently, our approach considers both camera motion and object motion, but will suffer from noisy motion vectors.

Step-2: Local motion histograms for shots

In contrast to other motion-based approaches, we do not use motion histograms that only express the motion within single frames. For our local feature histograms we use a motion representation that shows how the motion is distributed within single shots. Also the spatial region where the motion occurs is considered. Every frame is divided into blocks of 16x16 pixels size. A sample image that illustrates that spatial segmentation is shown by figure 2. For every block we estimate how the motion evolves over the duration of a shot.

Every macroblock or partition from the H.264/AVC stream is assigned to the corresponding block in our local motion feature representation. For each 16 pixel block we calculate a 13-bin motion histogram that indicates how the motion for that block is distributed during a shot. A schematic illustration for the motion histogram is given in figure 3. The same type of histogram has already been used successfully for other motion-related tasks [4].

For every macroblock or partition from the H.264/AVC stream we calculate the direction of the motion and add the partition area (number of covered pixels) to the corresponding bin of the motion histogram of the related block in our representation. The assignment of directions to bins is also indicated in figure 3. This assignment is done for all macroblocks of a frame and all frames of one shot. Finally the pixel number for each block is normalized by dividing it by the number of frames of the corresponding shot.

[1]URI: http://lucene.apache.org/

[2]URI: http://www.semanticmetadata.net/lire/

[3]x264 uses a threshold to decide whether intra-coding should be used instead.

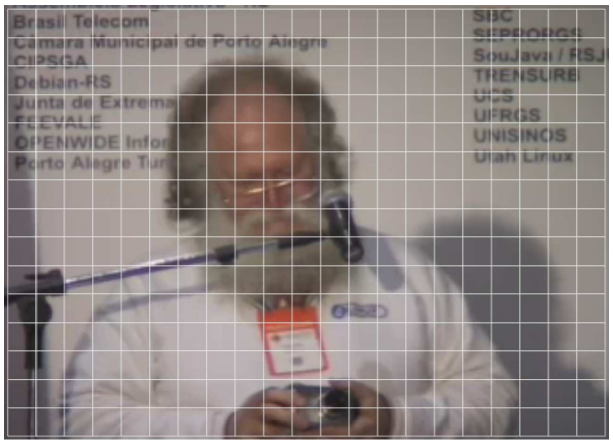


Figure 2: Every frame is divided into 16x16 pixel blocks.

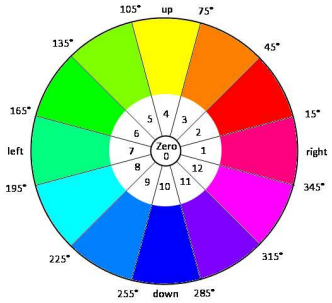


Figure 3: 13-bin motion histogram calculated for each 16x16 pixel macroblock

Step-3: Clustering of motion histograms

Each video shot contains several 13-bin motion histograms, hereby describing the local motion of objects. The histograms, stored as vectors, of all shots of every video get clustered using K-Means. The computed cluster centers constitute a motion codebook, which is required later on for local feature histogram creation. For local feature histogram creation every 13-bin vector within a shot is assigned to the most similar cluster center. The number of assignments to each cluster center is summed up, which provides a histogram describing the particular shot. During content based search videos are compared on shot bases, hereby leveraging the pre-computed local feature histograms.

2.2.3 Global Motion Histogram (GMH)

The creation of *Global Motion Histograms* (GMH) is based on the same first step as used for Local Feature Histograms. However, the second step and the third step are different.

Step-1: Computation of forward-predicted motion vectors on a per-frame basis

See Section 2.2.2.

Step-2: Classification of motion vectors by motion vector direction

The one-frame-distance motion vectors of a particular frame are classified into a motion histogram with $K+1$ bins, modelling K equidistant motion direction intervals (bins $b \in \{1,\ldots,K\}$) and a separate bin ($b=0$) for zero-length motion vectors. We define the *direction* of a motion vector $\mu=(x,y)\neq(0,0)$ with length $|\mu|$ as:

$$\omega(\mu)=\begin{cases}\arccos\frac{x}{|\mu|} & \text{if } y\geq 0\\ 2\pi-\arccos\frac{x}{|\mu|} & \text{if } y<0\end{cases} \tag{1}$$

Note that $0\leq\omega(\mu)<2\pi$ and that $\omega(\mu)=0$ corresponds to direction *right*. A motion vector μ is assigned to a bin of a $(K+1)$-bin motion histogram by the following equation:

$$b(\mu)=\begin{cases}0 & \text{if } \mu=(0,0)\\ 1+\left(\left\lfloor\omega(\mu)\frac{K}{2\pi}+\frac{1}{2}\right\rfloor \bmod K\right) & \text{otherwise}\end{cases} \tag{2}$$

This formula has been chosen such that all motion directions differing from direction *right* ($\omega=0$) by less than π/K (modulo 2π) are assigned to the same bin (bin 1). The same condition holds for the other main directions *left*, *up*, and *down*, too, if and only if K is a multiple of 4.

Finally, the *motion histogram* of a video frame consisting of R pixels is used to express the relative amount of pixels being predicted to a specific direction D_b, for each bin b. Therefore, D_b contains the relative amount of pixels of the frame being predicted by a motion vector of bin b. More precisely, if P_b denotes the set of macroblock partitions predicted by motion vectors of bin b, and if $|p|$ denotes the size of partition p in pixels, then

$$D_b=\frac{1}{R}\sum_{p\in P_b}|p| \tag{3}$$

Step-3: Creation of Global Motion Histogram (GMH)

An additional histogram (the GMH) is created for a video file that is used to count the number of frames with specific characteristics in their motion histograms. As characteristic we use the combination of the two most representative bins R_{b1} and R_{b2} in the motion histogram of a frame. Therefore, if $maxbin()$ returns the bin with the maximum value of D_b of a set of bins, the two most representative bins are defined as:

$$R_{b1}=maxbin(b),\forall b\in\{1,...,K\} \tag{4}$$

$$R_{b2}=maxbin(b),\forall b\in\{1,...,K\}\neq R_{b1} \tag{5}$$

As we use $K=13$ there are 13x12 possible combinations of R_{b1} and R_{b2}. Thus, the global motion histogram contains 156 bins for every video file.

2.3 Content-based Queries and Combining Results

For each query we start with the text-based search and use the first Z results in order to execute a separate content-based similarity search for each result (from 1...Z). This process is summarized in Figure 4. The content-based similarity search consists of up to three different search tasks (CEDD,LFH,GMH), depending on the chosen configuration.

While CEDD and LFH search is based on shots, GMH search is based on videos. Therefore, for CEDD and LFH we use the corresponding feature vector of the center shot of video z ($\in$ 1...Z) as input, while a feature vector taken from the entire video z is used as input for GMH search. Consequently, CEDD and LFH will return a ranked list of similar shots and GMH will return a ranked list of similar videos for each query. For each of the three types of content based search we use a rank-based interlacing scheme over all Z queries: the first result for video $z=1$ is ranked before the first result for video $z=2$,..., the first result of video $z=Z$ is ranked before the second result for video $z=1$ a.s.o. Because the results of the different

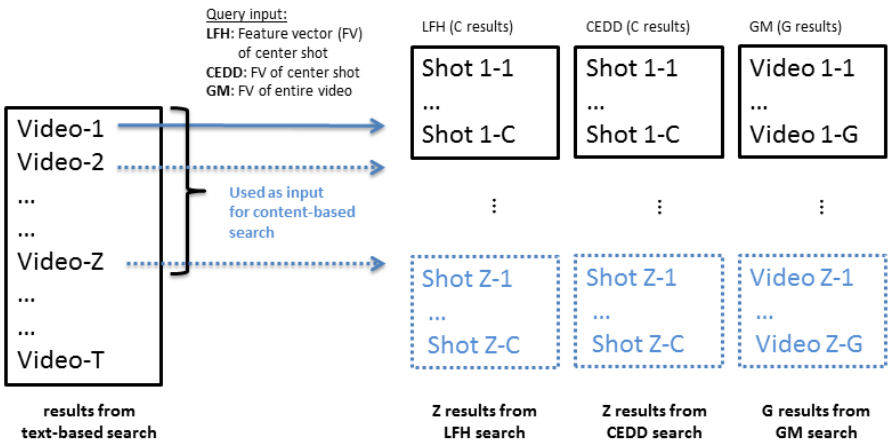


Figure 4: Content-based search with input from text-based search.

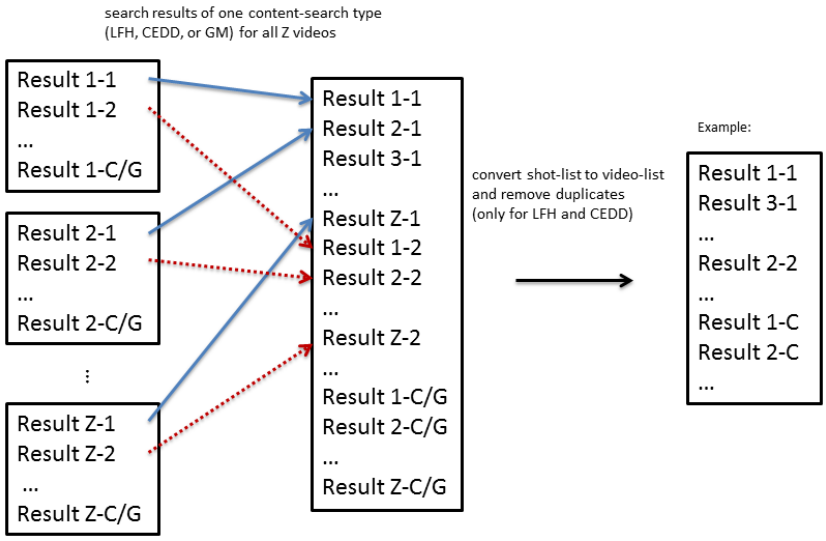


Figure 5: Combining results of content-based search

content-based search tasks may contain equal entries, we further remove duplicates from the merged lists before we combine these lists with the result list of text-based search. The whole combination process is summarized in Figure 5.

In order to combine the results of the content-based search with the T results of the text-based search, we use the averaged inverted rank of a specific video in all obtained result lists (see Figure 6). More precisely, for each different video file contained in the results, we compute an averaged inverted rank over all result lists (one from text-search, up to three from content-based search). This averaged inverted rank is used to create the final result list. Please note that if a video file is not contained in a certain result list, we do not consider this list when computing the average value (i.e. the sum of the inverted ranks is divided by 3 rather than 4 if one list does not contain the video file). The reason is that we want to avoid results of a specific type of search to be penalized if this particular result ist not found by another type of search.

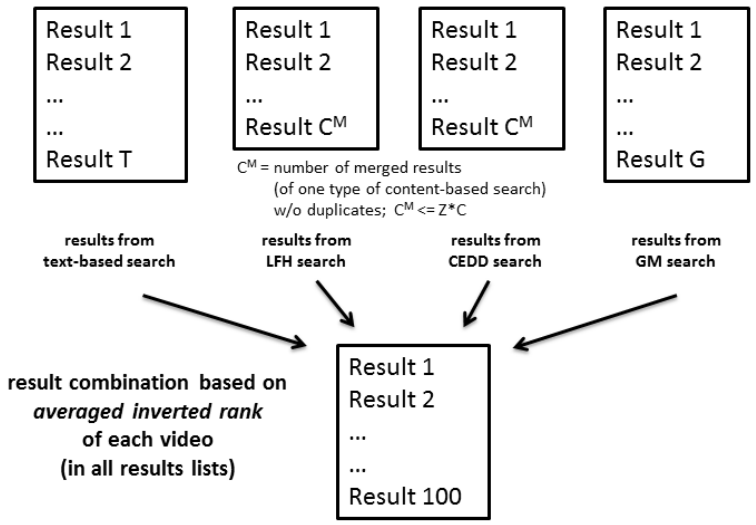


Figure 6: Combining results of content-based search and text-based search

Metric	Run 1	Run 2	Run 3	Run 4
mean inverted rank	0.266	0.260	0.257	0.263
topics not found (298 total)	143 (48%)	168 (56%)	149 (50%)	151 (51%)
topics with rank 1	65 (22%)	64 (22%)	62 (21%)	65 (22%)
topics with rank ≤ 10	103 (35%)	101 (34%)	100 (34%)	98 (33%)
C1 topics	–	3	3	2
C2 topics	–	28	9	10
mean rank of C2 topics in run 1	–	52	71	70
C3 topics	155 (100%)	127 (82%)	146 (94%)	145 (94%)
C3a topics	–	91	84	96
C3b topics	–	36	58	48
C3c topics	–	0	4	1

Table 2: Evaluation results of our known-item search runs. There were 298 search topics in total, and each run gave 100 results. The topic sets C1–C3 are defined in Section 3.2.

3 Known-Item Search Results

3.1 System Configuration

We used the following configuration for our runs:

Run 1: $T = 100$, text-based search only ($C = 0, G = 0$)

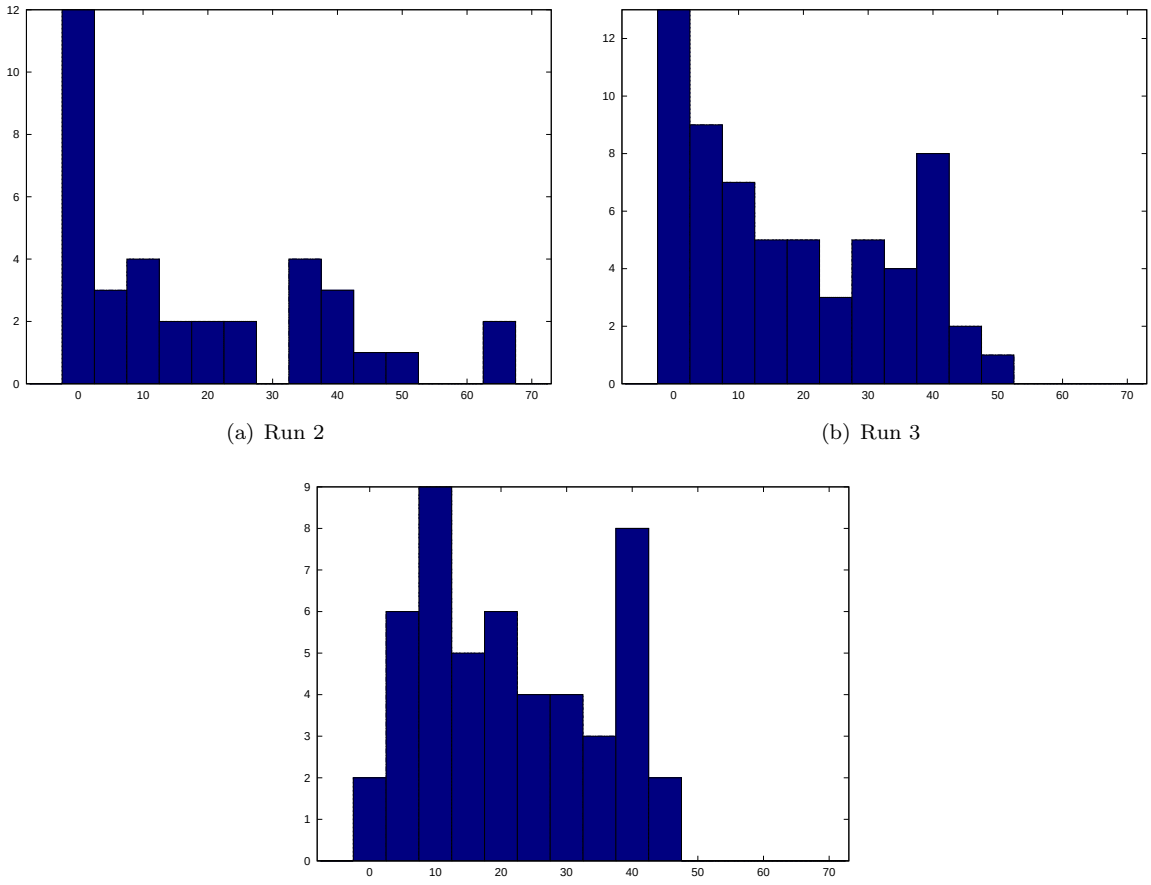
Run 2: $T = 60, Z = 10, C = 40, G = 40$, text-based search with CEDD,LFH,GMH search

Run 3: $T = 60, Z = 5, C = 60$, text-based search with CEDD search

Run 4: $T = 60, Z = 5, G = 60$, text-based search with GMH search

3.2 Run Results

The evaluation results of our 4 submitted known-item search runs are summarized in Table 2. Since runs 2, 3, and 4 combined the text-based search of run 1 with content-based search methods, we analyzed the



(a) Run 2

(b) Run 3

(c) Run 4

Figure 7: Histograms of non-zero rank differences (C3b and C3c topics) of combined search runs with respect to text-only search run 1 for items found by both run X and run 1 ($X \in \{2, 3, 4\}$). The counts for rank difference 0 (C3a topics) have been removed due to their high values. A positive rank difference means a deterioration. Note the different normalizations of the y-axes.

evaluation results with respect to the following sets of search topics: (X denotes one of 2, 3, or 4)

C1 set of topics *not* found by run 1, but *found* by run X.

C2 set of topics *found* by run 1, but *not* found by run X.

C3 set of topics *found* by both run 1 and run X.

C3a subset of C3 where each topic has the *same* rank in both run 1 and run X.

C3b subset of C3 where each topic has a *worse* rank in run X than in run 1.

C3c subset of C3 where each topic has a *better* rank in run X than in run 1.

According to mean inverted rank, the text-only search (run 1) performed best. For 1 of 5 search topics the correct answer appeared at rank 1 of the result list. The combined search methods (runs 2–4) deteriorated the rank of the searched item more often (C2 and C3b topics) than it improved the rank (C3c topics) with respect to text-only search. However, content-based search was able to find items not found by text search

(C1 topics) in some cases. For the majority of search topics the combination of result lists (see Section 2.3) was stable in the sense that it preserved the text-based result rank of the searched item (C3a topics, including items with rank less or equal to 10).

The bad performance of run 3 (text search combined with search in CEDD feature space) seems to be due to a relatively high number of C3b topics, where the mean rank of the searched item deteriorated more than with other runs. A more detailed view of the rank differences of searched items for runs 2, 3, and 4 with respect to run 1 is given in Fig. 7. The relatively good overall performance of run 4 (text search combined with search in GMH feature space) is due to the relatively high number of C3a topics, which are not included in the histograms of Fig. 7. Although run 2 (text search combined with search in CEDD, LFH, and GMH feature spaces) performs well with respect to C3 topics, the relatively high number of missed search topics (in particular, C2 topics) lead to a modest overall performance.

4 Conclusion

According to the official TRECVID result lists our approach was relatively fast in terms of execution time. That helped a lot for testing as one single run of the benchmarking suite developed for our search engine just took about 8 minutes. Therefore, we could test a lot of different parameter settings and possible extensions. Some of our ideas did not contribute to a reasonable inverted rank score and were not utilized for generating the final results. Our audio classification, for instance, turned out to be too erroneous and our query expansion module, which was based on a co-occurrence analysis of metadata, just improved recall at the cost of reduced precision.

Interestingly enough our simple text search approach was the best performing part. While it worked quite well on annotations only, results were improved significantly by adding terms from speech transcripts. We think that text search performance can be improved further by applying more advanced methods.

References

[1] Aimar, L. and Merritt, L. and Petit, E. and Chen M. and Clay, J. and Rullgard, M. and Czyz, R. and Heine, C. and Izvorski, A. and Wright, A., *x264 - a free H264/AVC encoder*, 2010, Online (last accessed on: 10/16/10): `http://www.videolan.org/developers/x264.html`.

[2] Savvas A. Chatzichristofis and Yiannis S. Boutalis, *Cedd: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval*, ICVS'08: Proceedings of the 6th international conference on Computer vision systems (Berlin, Heidelberg), Springer-Verlag, 2008, pp. 312–322.

[3] Mathias Lux and Savvas A. Chatzichristofis, *Lire: lucene image retrieval: an extensible java cbir library*, MM '08: Proceeding of the 16th ACM international conference on Multimedia (New York, NY, USA), ACM, 2008, pp. 1085–1088.

[4] K. Schoeffmann, M. Lux, M. Taschwer, and L. Böszörmenyi, *Visualization of video motion in context of video browsing*, Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on, August 2009, pp. 658–661.

[5] C.W. Su, H.Y.M. Liao, and K.C. Fan, *A motion-flow-based fast video retrieval system*, Proc. of the 7th ACM SIGMM Int. Workshop on Multimedia information retrieval, ACM New York, NY, USA, 2005, pp. 105–112.