

TRECVID 2005

Experiments at MediaTeam Oulu

Mika Rautiainen[†], Matti Varanka[†], Ilkka Hanski[†], Matti Hosio[†],
Anu Pramila[†], Jialin Liu[†], Timo Ojala[†] and Tapio Seppänen[†]
[†] MediaTeam Oulu

P.O.BOX 4500, FIN-90014 University of Oulu, Finland
{firstname.lastname@ee.oulu.fi} discard umlauts

Abstract

University of Oulu's MediaTeam research group participated to the manual and interactive search tasks with our video retrieval system, which is constructed out of three sub-applications: Content-based Query Tool, Cluster-Temporal Browser and Result Container with relevance feedback mechanism. The search engine in the heart of our search system creates result sets using three distinct search sub-engines that take care of the text, semantic concept and example based queries. The sub-applications and sub-engines result in varying search strategies. Therefore their effect on overall search performance has been studied in our experiments. The principal contribution in our search system is the Cluster-Temporal Browser, the effect of which we tested this year in conjunction with the traditional content-based paradigm: recurrent queries with relevance feedback.

Seven search runs were submitted for the TRECVID 2005 search task:

```
I_A_2_OUMT_I1Q_1: interactive with browsing disabled, expert users
I_A_2_OUMT_I2B_2: interactive with browsing enabled, expert users
I_A_2_OUMT_I3Q_3: interactive with browsing disabled, novice users
I_A_2_OUMT_I4B_4: interactive with browsing enabled, novice users
M_A_1_OUMT_M5T_5: manual with official text transcripts
M_A_2_OUMT_M6TS_6: manual with official text transcripts and selected semantic concepts
M_A_2_OUMT_M7TE_7: manual with official text transcripts and selected topic examples
```

For interactive experiments, we have used eight test users: four novice and four expert users. In a carefully designed experimental setup, all users have used two system configurations: one with and one without Cluster-Temporal Browser. In our experiments we found that by incorporating Cluster-Temporal Browser the performance gain is more than 12% over the conventional content-based tools with novice users. We also found that on average expert users gained more than 18% better search performance over novice users, which shows that the test design has a significant effect to the outcome of the interactive test.

Our manual experiment is a comparison of three query configurations: a baseline text only search, text combined with visual example search and text combined with semantic concept search. To facilitate a concept search, we trained a set of concept detectors (to name a few: newsroom, entertainment, maps and charts, news footage, weather, sports, outdoors). Our example search was based on low level descriptions (color and structure). The goal of the experiment was to find out how much more visual search examples and user defined semantic concepts contribute to the search performance over a text baseline. Our experimental results showed that the combined text and semantic concept search gives about 19% better performance over text baseline whereas text combined with example based search gives about 25% performance gain over the baseline. The results show that specific visual search examples accumulate better overall precision than the queries defined with our detected set of semantic concepts.

1 Introduction

This paper starts with describing the system components (search engines) of the Video Browsing and Retrieval System (VIRE) as they were implemented for 2005 experiments. Next, the different search interfaces are being discussed. Following sections give details about the experiments in manual and interactive search tasks.

2 Manual and Interactive Search

2.1 Video Browsing and Retrieval System

The video browsing and retrieval system VIRE was used in manual and interactive search experiments. System is constructed using J2SE, QuickTime for Java and MySQL JDBC software components. Principally, the system consists of a search server component and a client application; the server delivers queries down to the three search engines and formulates the final result set by fusing the sub-results from the independent engines.

For TRECVID 2005, the entire client application was re-designed from ground up. Previously, different client interfaces were developed as separate applications. This led to problems in system usability since, according to the questionnaires, switching between different interface windows was considered as problematic for the novice test users. The re-designed client incorporated all user interface elements into a single master application view for streamlined switching between them. Client software was constructed of three search interfaces; first for creating video queries manually, second for facilitating cluster-temporal browsing in the video database and third for collecting and organizing results and viewing the relevance feedback based on collected results. References to media data and additional meta-information are stored in MySQL-database. Actual video data is stored on a network file system. The playback of video files and shots does not utilize any intermediary streaming service. The principal contribution in our search system is the browsing interface (cluster-temporal browser) [9] which display relevant database content dynamically to the user.

VIRE server creates a search using three search engines (see Fig.1). Lowest-level engine computes visual similarity between two video objects and is typically used in a content-based example search. Concept search engine uses higher-level features extracted from low-level features by pre-trained semantic concept detectors. Text search is based on automatic speech recognition transcripts and machine translation transcripts, which makes the text search engine useful tool especially in news video material where spoken content is a significant part of the news media structure.

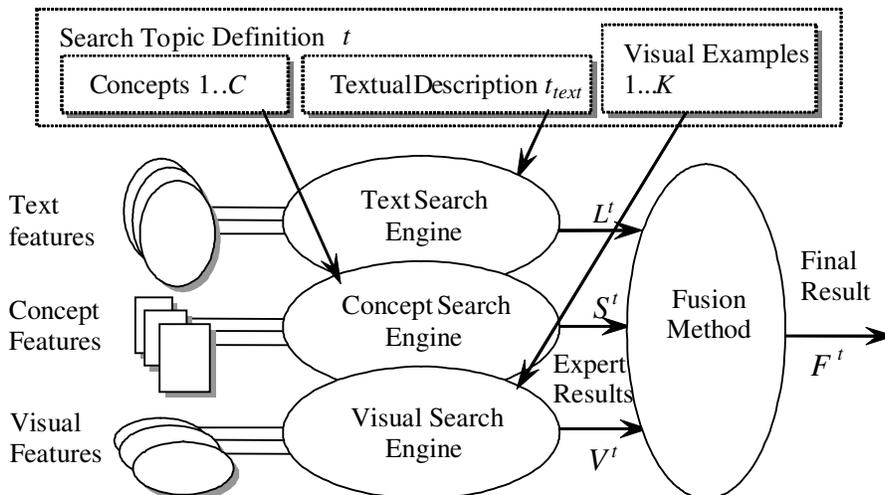


Figure 1. The VIRE test system used in TRECVID experiments

Visual Search Engine

Unlike previous years, this year features were not extracted in temporal fashion. The rationale underlying the decision was preserving the computational efficiency. The ~170 hours of TRECVID 2005 video data was constructed from two level segmentation of video: subshot and master shot. Whereas the subshots were based on actual detection of visual shot transitions, master shot segmentation was instead based on simple aggregation rules for creating shots of meaningful length e.g. having at least two seconds of duration [11]. For this reason our system extracted a visual feature vector in subshot level using key frames that were provided for each subshot.

Generic features are measured from the physical properties of the visual video data. Generic feature vectors are compared using a geometric distance that measures dissimilarities between two vectors. To compute the geometric dissimilarities of the described feature vectors, we have used L1 norm (city-block distance). Each feature vector is normalized prior computing the dissimilarity. Several features can be used simultaneously in search to compute overall similarity of two objects. The results of individual feature queries (ranked lists of database shots) are combined with late fusion technique to provide a final ranked set of most similar video shots. The fusion of different visual features is realized with a straightforward sum of ranks with equal weights. The benefit of using rank combination is fair when two features have very different dimensionalities. [10]

The visual similarity is constructed from the two physical properties of a shot: (I) *Color* is the most widely used content property in CBVR research. Similarity by color gives initially very good perceptual correspondence between two color images that are small or short of details. After visual content is reviewed in detail, other properties for similarity emerge from the details. (II) *Structure* of the edges in a visual imagery is a strong cue for many computer vision applications, such as classification of city and landscape images or segregating natural and non-natural objects. This can also provide invaluable in queries where statistical color information is insufficient in describing the relevant properties of an image. Following features have been used to describe color and structure: Color Correlogram (CC) and Gradient Correlogram (GC). These features are computed from every sub-shot key frame in the database. A closer description of the features follows.

Color Correlogram (CC). Color properties of a video shot were obtained into a Color Correlogram (CC) feature vector. Its efficiency against traditional static color descriptors has been indicated in [1]. At this point it is important to emphasize that our previous study [2] has revealed Temporal Color Correlogram to be more efficient in visual video retrieval scenarios. CC captures the correlation of HSV color pixel values in spatial neighborhoods. The details about the computational parameters such as color space quantization and spatial constraints are described in [1][2].

Gradient Correlogram (GC). Gradient Correlogram (GC) feature computes local correlations of specific edge orientations producing an autocorrelogram, whose elements correspond to probabilities of edge directions occurring at particular spatial distances. Unlike TGC, feature vector is computed only from a single video key frame for the entire video sequence, in similar fashion as the CC feature. Due to lack of sampling of the sequence, CC auto-correlogram is not affected by the temporal changes in spatial edge orientations. Initially, the edge orientations (obtained with Prewitt kernels) are quantized into four segments depending on their orientation being horizontal, vertical or either of the diagonal directions. The details of the temporal version of this feature and the parameters can be found from the study utilizing it in the detection of city/landscape scenes and people from video shots [5].

Concept Search Engine

In our retrieval system, semantic features are lexically defined concepts that exist in a shot with a certain confidence which is described as a floating value between 0 and 1. Trained semantic concept detectors are used to create a series of confidences for each video shot. Our concept search engine utilizes a defined set of concept confidences which are being compared against other shots' confidence values. One problem with floating confidence values is that they are generated with an algorithm that attempts to find out the degree in which the concept in question truly exists in a shot. Therefore at a certain range, the confidence value starts to become unstable indicator resulting in non-relevant results returned for a concept query. However, successful concept detectors can work efficiently as filters for removing conceptually non relevant shots from the results, thus improving overall search efficiency.

We have developed two types of concept detectors for our experiments in TRECVID 2005 search task:

1: Classifier based detectors. We trained Support Vector Machines (SVMs) to detect concepts from the subshots. Following concepts were trained: *entertainment, faces, newsroom, outdoor, desert, natural-disaster, and snow*. The training data was obtained from the collaborative annotation effort, organized by IBM [8]. No refinement to the annotation data was made by a human. Training

sets for each concept were randomly selected from the positive/negative labelled development data using 1:2 proportion between the positive and negative set size. Half of the available positive labels were used for training each concept. The classifier implementation was SVM light [13]. In order to create confidences from the classifier output, we have used probabilistic output for SVM [14].

Experimental validation of detectors was executed in the annotated development set to obtain SVM parameters and suitable low level feature sets. Experiments consisted of cross-validation training for SVM classifiers and performance measurement with F-measure [12] metric. Two primary configurations were selected for the SVM concept detectors as a result of experimental validation.

- Configuration A consisted of audio and face detector features (mel-cepstral coefficients, Root Mean Square and Zero Crossings Ratio, localized number of detected faces) with linear SVM kernel.
- Configuration B consisted of visual and face detector features (Color Correlogram, Gradient Correlogram, localized number of detected faces) with second order polynomial kernel.

Configuration A was used for detecting *entertainment* whereas B was assigned to *newsroom*, *outdoor*, *desert*, and *snow* detectors. Additionally, *natural-disaster* classifier consisted of all previously described visual, audio and text features with second order polynomial kernel, unfortunately resulting in unsatisfactory overall performance. *Faces* concept was derived from the output of a face detector that was based on cascade learning algorithm of Wu *et al.*[15].

2: Propagated labelling based on small example sets. We also created a few concept detectors using simple training scheme as described in [6]. The ‘training’ of the detectors was based on selection of small example sets from development data set and confidence values were generated by conducting nearest neighbour queries from these example sets. The queries from the example sets were based on low level Color Correlogram feature. Therefore the confidence value for a concept was based on visual similarities in the color structure between the images in example sets and subshot key frames in the database. The detectors were (number of example images in parenthesis): *fire-explosion-smoke*(17), *maps-charts*(30), *meeting-footage*(33), *nature-footage*(80), *sports*(80), *water*(32), and *weather*(23).

All of the above mentioned detectors were used to create concept feature vectors for the concept search engine. Concept search engine acts as the provider of final results for concept queries. Manual concept query is based on user-defined set of query concepts that have been attributed as relevant for the query. In order to achieve the overall concept similarity, confidences to every query concept are obtained as a rank value in the concept list. The dissimilarity of a query concept and a database item equals to item’s scaled confidence rank on that concept. Finally, after computing dissimilarities (scaled confidence ranks) for every query concept, overall dissimilarity between the query concept list and a database item is computed as the sum of individual dissimilarity values. Final result list is sorted by the overall dissimilarities and is outputted as the result for concept query.

Semantic search engine performance was estimated in the TRECVID 2005 manual search experiments, specifically in run OUMT_M6TS. In our prior interactive experiments we have incorporated concept features in cluster-temporal browser application. However, this year we left out the concept features from the browser application since, according to the feedback from previous user questionnaires, they have been conceived less useful than visual and text features in cluster-temporal browsing.

Text Search Engine

The text indexes consisted of automatic speech recognition (abbreviated as ASR) and machine translation (MT) transcripts. The sources for the transcripts were NIST and Carnegie Mellon University. Transcript texts were produced by running audio through a program, which automatically converted spoken audio to text. In TRECVID 2005 the video data contained three languages: English, Arabic and Chinese. When necessary, a program had been used to create language translation to English. The quality of the machine translations was not consistent. Since the formats for the provided transcript types varied greatly, generating consistent text search index was much more difficult than in the previous TRECVID experiments. Our approach was first to convert each transcript format into our system compatible format to simplify the index creation. Speech segments were especially problematic since not all transcript types provided such information, but it was required for our indexing algorithms nevertheless. Our solution was to use transcript dependent rules to group words into approximated speaker segments.

The transcripts were indexed into a database treating each white-space delimited token as a word. A stop word list was used to exclude grammatical and otherwise non-discriminating words that would have led to poor resolution. Remaining words were then stemmed using the Porter stemming algorithm [4].

Speaker segmentation was utilized to create better contextual organization for the index and whip up the temporal search capabilities. This was achieved by expanding the search from shots containing exact hit into neighboring shots until the boundaries of given

amount of speaker segments were reached. Due to this, every shot became topically connected to its neighbors, presuming that the speaker would not change his/her semantic context during speech. Expansion was done by assigning each surrounding shot a score which was dependent on how far away the shot was from the exact hit and whether the shot was in the future or past. This was done by prioritizing shots, sorting them globally and then scoring them using an exponential decay function (highest priority shot got highest score). Priority value of zero, which is considered as the highest, is given to the shot containing the searched word. The next shot in the coming timeline would be given a priority value of 1, the shot after that value of 3 (then 4,5,6...) until the given amount of speech segment boundaries were met. The priority of 2 was always assigned to the first shot preceding the exact hit. Finally, the remaining shots before the exact hit were prioritized with lower priorities than then shots in the future. If one shot were ranked more than once (ie. there were two exact hits in its vicinity), shot-to-be-ranked would be given the highest priority from the available value range.

To compute the matching score between database shots and a query term, we used prioritised ranking combined with weighed term frequency score.

$$L(queryterm, s) = 0.2 \cdot \frac{\log(t+1)}{\log(dl+1)} * \log\left(\frac{N}{m}\right) + e^{-\frac{B \cdot j}{J}} \quad (1)$$

where s is the database shot to be scored, t is the number of matching words in the shot s , dl is the amount of all words in the text index (stop words are excluded), N is the number of shots in the test database and m is the amount of matching shots containing the searched word, B time constant, j the index (position) of s in the list of matching shots that is ordered by priority values, J is the size of the priority ordered list of shots. The given query words were stemmed to remove any suffixes. The scores were computed independently for every query term. The final scoring for the shot s was the sum of individual scores. Finally, the result list was sorted based on the final scores.

For the baseline run and rest of the manual runs, the files required by NIST were used when creating the indexes. For interactive search, the text indexes were partially patched with the optional file set. Since text similarity was also a search feature in the cluster-temporal browser, we constructed an example-based text search that used spoken text from the example shot as a source for the query. Text was prepared using stop word removal and stemming. A list of lexically similar shots was then created using previously described scoring scheme.

Feature Fusion

In order to construct a final ranked list of most similar shots, VIRE system has to formulate sub-queries for the search engines. User can select whether he/she wants to retrieve shots based on all or some of the semantic search engines. For example, user can define manually following type of query description t :

- Text: 'Find shots of a ship or boat'
- Concepts: OUTDOOR, WATER
- Visual: example shots of boats and ships

The sub-results can be considered as votes of individual feature 'experts' e . To make a fusion of these feature lists, we use a Borda count [7] variant. Following formulas describe the fusion procedure

$$f^t(n) = \text{sum}\left(\frac{w_v \cdot v^t(n)}{V_{\max}^t}, \frac{w_s \cdot s^t(n)}{S_{\max}^t}, \frac{w_l \cdot l^t(n)}{L_{\max}^t}\right) \quad (2)$$

$$F^t = \left[\text{sort}\{f^t(1), \dots, f^t(N)\} \right]_X \quad (3)$$

where $f^t(n)$ = overall rank of a result shot n to the search topic definition t

$v^t(n), s^t(n), l^t(n)$ = rank of a result n by independent search engines

w_v, w_s, w_l = weights of the search engines, we used values 1 (vis), 1 (sem), 2 (text) in our experiments

$V_{\max}^t, S_{\max}^t, L_{\max}^t$ = last rank of the independent result lists

F^t = Final ranked set of results for the search topic t
 $\left[\right]_X$ = X top-ranked items in a sorted list, where X equals to 1000 for TRECVID 2005 search experiments

Content-based Query Tool

The interface of Content-based Query Tool allows defining query attributes manually for the search engines. Changing a topic from the menu initializes a timer and new set of example shots/images from the topic description files. User can select any combination of the topic examples for visual search and select the low-level features individually for each example. From the list of semantic concepts, user can construct a configuration for semantic query. Textual query terms are created by typing words to a text box. After user has enabled the desired engines for the search and selected the appropriate attributes for every selected engine a search is submitted to the VIRE server. Server distributes the query definitions to the respective search engines. When the result shots for the query have arrived, user can select any interesting shot from the result set as a start point for browsing with the browsing interface.

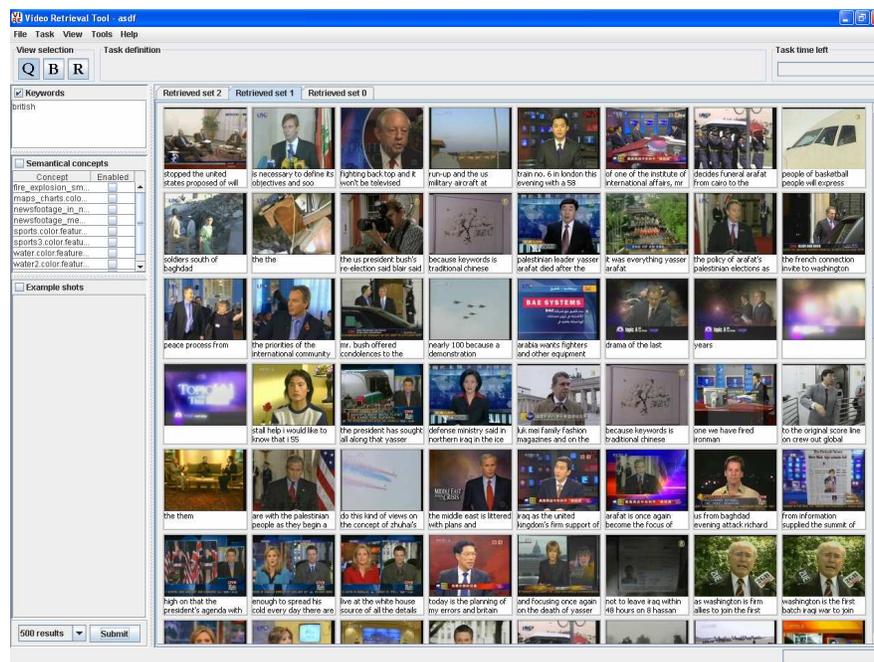


Figure 2. The Content-based Query Tool

Cluster-temporal Browsing Interface

The role of the video browser is to provide complementary cues for users to navigate through the vast search space towards the relevant objects. Our approach in the interactive search task is based on *cluster-temporal browsing* [9]. The motivation is to reduce the ambiguousness that is typically present in a traditional content-based example search by incorporating timeline navigation and parallel content-based queries. Cluster-temporal browsing shows both inter-video similarities and local temporal relations of intra-video shots in a single interface. Users can perform browsing that combines timeline presentation of videos dynamically with content-based retrieval. The name 'cluster-temporal browsing' implies that the content-based feature clusters are presented jointly with the temporal video structure.

Figure 3 (left) illustrates the browsing interface. In the figure, the topmost panel showing the first row of the key frame images displays a timeline of chronologically adjacent shots from a video file. At any time, user can scroll through the rest of the key frames in the video file to get a fast overview of the content. The panel right below the video timeline gives user another view of key frames, but this time they are shots from other videos in the database. This panel shows the results of multiple content-based queries created using the shots at the top row as query examples. The query results are organized into columns and are shown in parallel order to

create a similarity matrix. The columns show the most similar matches organized in top-down rank-order. Therefore the most similar shots are organized at the top of the matrix. This similarity matrix provides a view to find other shots that are similar to the shots in the video timeline. Therefore a searcher who is browsing through the timeline of a video file can instantly see large numbers of additional shots that have content similar to the query video sequence at the top. The similarity criterion is defined by the user-selected features. The engines that are available for browsing are visual and text search engines. User can select any combination depending on what conditions he/she wants to browse the database with.

Navigation with the browser is straightforward. When user locates interesting shot in the similarity view, he can replace the current broadcast video (on top row) with the source of the interesting shot so that the shot of interest is located at the top of the middle column. After the video shots have changed at the top row, system re-computes the similarity view from the new set of shots. At any time, user can update the current view by changing to other feature combinations. Each transition caused by browsing the timeline in the current video brings new shots to the view. Because of the new content on the screen, the similarity view updates itself immediately.

Each shot key frame shows also the spoken content as text under the key frames. The requirements to update the similarity view are heavy, since the browsing speed should be close to real-time. To update a view, system must process parallel queries to get results for several simultaneous example-based queries. Multi-threaded index queries with efficient query cache provide reasonable access times even for the most complex feature configurations. Lower left panel in Fig. 3 (left) shows the Browsing history panel. It collects the sequence of shots that have been selected for browsing. Above history panel are similarity settings that govern the features used to compute content-based similarity. User can select visual, text or both as a similarity criterion. Figure 3 (right) shows buttons that are superimposed on the key frames when mouse cursor is dragged over. These buttons can be used to fast playback the shot, open master video to the timeline row, display more information about the shot and add the shot to the Result container view.

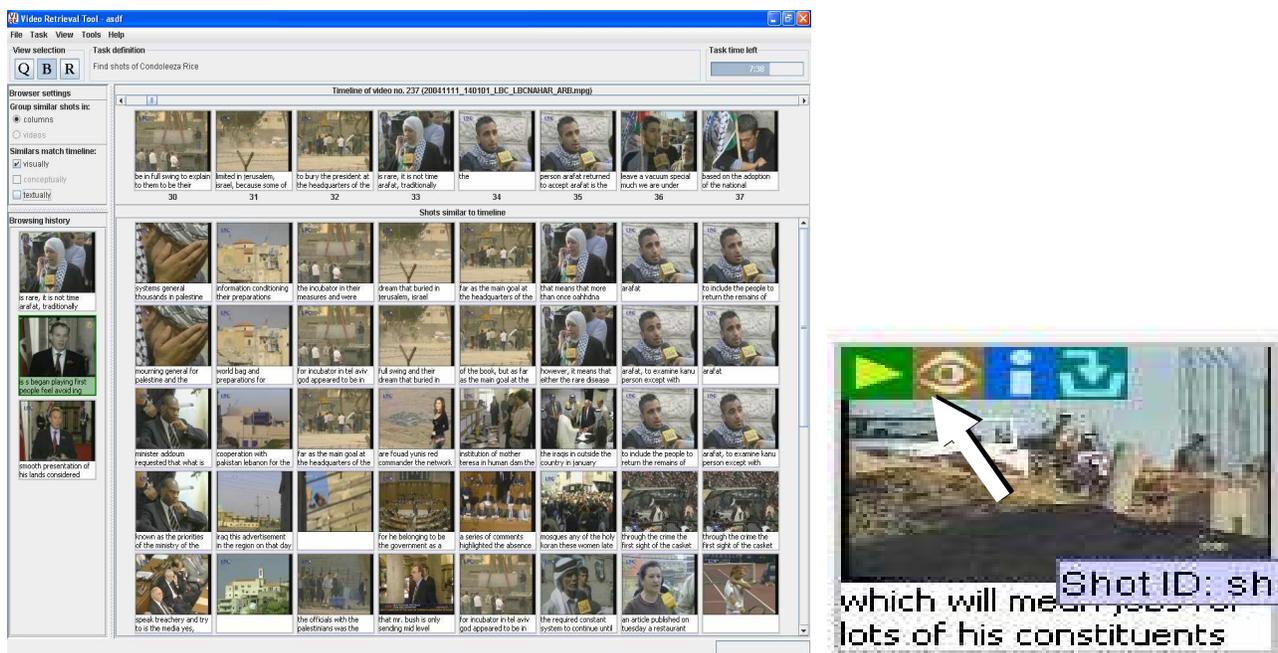


Figure 3. (Left) Cluster-temporal browsing interface. The similarity view organizes result shots column-wise. The results are displayed using shot key frames. Also speech transcript of the shot is visible under the key frame. (Right) Quick buttons appear over the key frame when mouse cursor is dragged over.

Result Container

Figure 4 shows the container for relevant shots. When user finds a relevant shot, he adds it to Result container which keeps a list of selected shots. Following relevance feedback mechanism is built into Result container: new content-based query is constructed every time a new shot is added to the container. Updated relevant shots are used as query examples. Returned results are given as an additional resource to the user in order to help finding more relevant shots from the other parts of the database. The relevance feedback query is directed to visual and text search engines and returned results are displayed under the user's selection of shots.

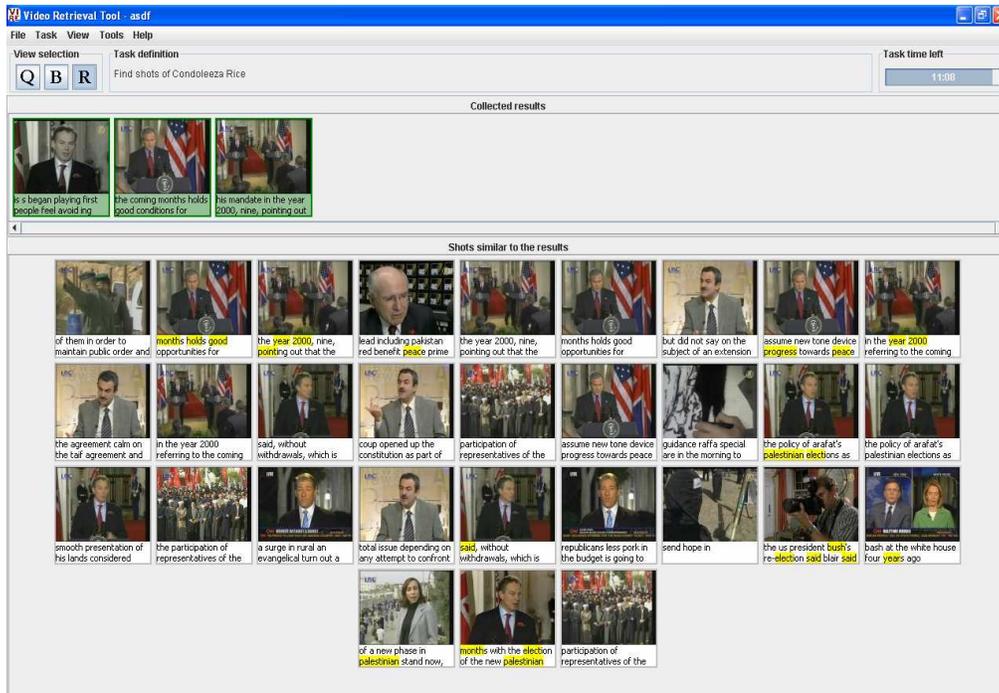


Figure 4. Result container stores all the relevant selected shots and suggests more similar shots based on the selected ones. Both visual and text similarities are utilized in relevance feedback query.

2.2 TRECVID 2005 Experiments

MediaTeam research group submitted full seven result runs, four for interactive and three for manual search tasks. NIST provided 24 search topics that were used in the experiments. A topic contained one or more example clips of video or images and textual description to aid the search process. Fraunhofer HHI provided common segmentation for the video data [11][11].

Experimental Setup at MediaTeam

We configured our search system to use subshots, which were converted into master shots prior final result submission. Visual, semantic and text search engines were used with weights of 1, 1 and 2 respectively. Semantic concept search was available in manual search experiments, but not in the cluster-temporal browser. The visual similarity search utilised CC and GC features as described in 2.1. Provided video examples of the search topics were used in visual search configurations throughout the manual experiments. Semantic concept search parameters were selected from the list of concepts described in 2.1. Text search baseline run in manual experiments was based on original transcripts from the NIST. In manual experiments a test user defined queries for all 24 topics. Content-based Query Tool was used to generate three different configurations: [features] (weights) (run ID)

- [text baseline] (run M5T)
- [text & semantic concept features] (weights 2 & 1) (run M6TS)
- [text & visual examples] (weights 2 & 1) (run M7TE)

The interactive search task was carried out by a group of 8 test users, from which four users were experts in searching with the system, but had not seen the given search topics or any content from the test database. Novice test users were mainly information engineering undergraduate or post-graduate students, having good skills in using computers but little experience in searching video databases. Experienced users had participated to the development of the system and had been testing it with development set for a significant time period (several hours). One of the test people was a native Chinese speaker, rest were speaking Finnish as their mother language. All of the test users were used to web searching. 8 users, 24 topics and two variants of VIRE system were divided into following configurations

- I1Q: Variant A: S1[149-154], S3[155-160], S2[161-166], S4[167-172]
- I2B: Variant B: S2[149-154], S4[155-160], S1[161-166], S3[167-172]
- I3Q: Variant A: S7[149-154], S5[155-160], S6[161-166], S8[167-172]
- I4B: Variant B: S8[149-154], S6[155-160], S5[161-166], S7[167-172]

Cluster-temporal browser was disabled in System variant A. Therefore queries were made with Content-based Query Tool and Result Container with relevance feedback. System variant B enabled cluster-temporal browser. Each user completed first six topics with one system configuration and then another six topics using another configuration. By setting the experiment into this configuration, the effect of learning was reduced between the system variants. The effect of fatigue was minimized with break and refreshments between system configuration change. The effect of learning within the topic sets was not controlled, most of the users processed the topics in numerical order. However, in runs I1Q and I2B, users started the experiment with topic set having bigger numbers. All users were given half an hour introduction to the system, with emphasis on search and browsing interface functions demonstrated with a couple of example searches. Users were constrained to twelve minutes for searching per topic, during which they selected shots that were relevant to the topic description. The result submissions of 1000 shots were created using selected results as examples to retrieve more shots for the result set. Additional shots were retrieved using combined text and visual search with text having weight of two and visual weight of one. The additional search time was added to the original time. During the change of system configurations (halfway of the experiments), users were given refreshments and a break. The machines that the search client was running on were 0,8-2GHz PCs with Windows XP operating system installed. Total duration of the experiment was slightly less than three hours.

2.3 Search Results TRECVID 2005

Mean average precisions (MAP) of the 24 search topics for the 7 search runs are shown in Table 1. The descriptions of the runs are found from Chapter 2.2.

Search Run ID	MAP	Total Relevant Shots Returned
I1Q (interactive, expert users)	0.264	2284
I2B (interactiv, expert users)	0.242	1916
I3Q (interactive, novice users)	0.202	1907
I4B (interactive, novice users)	0.226	1998
Mean (interactive)	0.218	1618
Max (interactive)	0.414	3044
M5T (baseline)	0.081	1836
M6TS (txt+semantic)	0.097	2003
M7TE (txt+examples)	0.102	1972
Mean (manual)	0.067	1510
Max (manual)	0.169	2278

Table 1. Results for search runs in 24 search topics

Some examples of the most successful topics in the interactive tests were 153 (Tony Blair), 156 (tennis players on the court) and 171 (tall buildings).

3 Conclusions

For interactive experiments, we have used eight test users: four novice and four expert users. In a carefully designed experimental setup, all users have used two system configurations: one with and one without Cluster-temporal Browser. In our experiments we found that by incorporating Cluster-Temporal browsing to the system, novice users were able to achieve about 12% performance gain (in MAP) over the conventional content-based tools. This result is in line with previous reported experiments using novice test users [5]. However, the situation was opposite with expert users: higher MAP was achieved without Cluster-temporal Browser. Perhaps expert users' broad knowledge about system capabilities and limitations makes them perform well with every configuration. Also personal skills may vary. Finally, it can be seen that on average expert users gained more than 18% better search performance over novice users, which shows that the test design has a significant effect to the outcome of the interactive test.

Our manual experiment is a comparison of three query configurations: a baseline text only search, text combined with visual example search and text combined with semantic concept search. To facilitate a concept search, we trained several concept detectors (to name a few: newsroom, entertainment, maps and charts, news footage, weather, sports, outdoors). Our example search was based on low level descriptions of color and structure. The weight ratio between text, visual and concept search was set to 2:1:1 respectively. The goal of the experiment was to find out how much more visual search examples and user defined semantic concepts contribute to the search performance over a text baseline. Our experimental results showed that the combined text and semantic concept search gives about 19% better performance over text baseline whereas text combined with example based search gives approximately 25% performance gain over the baseline. The results show that specific visual search examples accumulate better overall precision than the queries defined with our detected set of semantic concepts.

Acknowledgments

The financial support of the National Technology Agency of Finland, Academy of Finland and Nokia Foundation is gratefully acknowledged.

References

- [1] Ojala T, Rautiainen M, Matinmikko E & Aittola M (2001) Semantic image retrieval with HSV correlograms. Proc. 12th Scandinavian Conference on Image Analysis, Bergen, Norway, pp.621-627.
- [2] Rautiainen M & Doermann D (2002) Temporal color correlograms for video retrieval. Proc. 16th International Conference on Pattern Recognition, Quebec, Canada.
- [3] MPEG-7 standard: ISO/IEC FDIS 15938-3 Information Technology - Multimedia Content Description Interface - Part 3: Visual.
- [4] Porter M (1980) An Algorithm for Suffix Stripping Program. *Program*, 14(3):130—137.
- [5] Rautiainen M, Ojala T, Seppänen T (2005) Content-based browsing in large news video databases. 5th IASTED International Conference on Visualization, Imaging and Image Processing, Benidorm, Spain.
- [6] Rautiainen M, Seppänen T, Penttilä J & Peltola J (2003) Detecting semantic concepts from video using temporal gradients and audio classification. International Conference on Image and Video Retrieval, Urbana, IL.
- [7] Ho T, Hull J & Srihari S (1994) Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(1): 66–75.
- [8] Volkmer T., Smith J.R., Natsev A., Campbell M., Naphade M. (2005) A web-based system for collaborative annotation of large image and video collections, In Proceedings of the 13th ACM international Conference on Multimedia, Singapore.
- [9] Rautiainen M, Ojala T & Seppänen T (2004) Cluster-temporal browsing of large news video databases. 2004 IEEE International Conference on Multimedia and Expo, Taipei, Taiwan.
- [10] Rautiainen M, Ojala T & Seppänen T (2004) Analysing the performance of visual, concept and text features in content-based video retrieval. 6th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2004, New York, NY.
- [11] Petersohn C. (2004) Fraunhofer HHI at TRECVID 2004: Shot Boundary Detection System, TREC Video Retrieval Evaluation Online Proceedings, TRECVID 2004, URL: www-nlpir.nist.gov/projects/tvpubs/tvpapers04/fraunhofer.pdf
- [12] van Rijsbergen C.J. (1979) *Information Retrieval* (2 ed.) Butterworths.
- [13] Joachims T. (1999) Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press.
- [14] Lin H.T., Lin C.J., Weng R.C. (2003) A note on Platt's probabilistic outputs for support vector machines. Technical report, Department of Computer Science, National Taiwan University, URL: <http://www.csie.ntu.edu.tw/~cjlin/papers/plattprob.ps>.
- [15] Wu J., Rehg J.M., Mullin M.D. (2003) Learning a Rare Event Detection Cascade by Direct Feature Selection. *Advances in Neural Information Processing Systems*, NIPS 16, MIT Press, Cambridge, MA.