Video Content Browsing Based on Iterative Feature Clustering for Rushes Exploitation

Werner Bailer, Christian Schober, Georg Thallinger

Institute of Information Systems and Information Management
JOANNEUM RESEARCH
Steyrergasse 17, 8010 Graz, Austria

{firstname.lastname}@joanneum.at

Abstract

Rushes Exploitation

We have implemented a feature-independent framework for video content browsing, which is based on iterative clustering and filtering of the content set. Plug-ins for content clustering using the features camera motion, motion activity, audio volume, face occurrences, global color similarity and object similarity have been implemented. A light table view is used for visualization in the browsing tool. We have evaluated the tool on the TRECVID rushes data with four browsing tasks defined by a textual description of the material. We measured precision, recall and the number of items found during the working time.

1 Introduction

The description of the TRECVID 2006 rushes task does not define a specific use case. We thus start by describing the scenario at which our work is targeted. One of the goals of the IP-RACINE project (http://www.ipracine.org) is to improve the digital cinema production workflow. An important step in this workflow is the management of essence and metadata in the production and post-production phase. In post-production environments, users deal with large amounts of audiovisual material, such as newly shot scenes, archive material and computer generated sequences. A large portion of the material is unedited and often very redundant, e.g. containing *k* takes of the same scene shot by *n* different cameras. Typically only few metadata annotations are available (e.g. which production, which camera, which date). The goal is to support the user in navigating and organizing these material collections, so that unusable material can be discarded yielding a reduced set of material from one scene or location available for selection in the post-production steps.

In order to enable the user to deal with such large amounts of content, it has to be presented in a form which facilitates the comprehension of the content and allows to quickly judge the relevance of segments of the content. Media summarization techniques are proposed for that task and are strategies that aim at selecting parts of multimedia content which are expected to be relevant for the user. A multimedia summary shall

- support the user in quickly gaining an overview of an known or unknown set of material,
- organize content by similarity in terms of any feature or group of features,
- select representative content for subsets (e.g. clusters) of the material.

The main challenge is therefore to decide which parts of a video are relevant and to select a visual representation for these parts. In order to be flexible in terms of the features used for summarization, we have decided to design a framework for content browsing that is feature-independent and extensible. The feature specific parts are implemented as plug-ins.

The rest of this paper is organized as follows: Section 2 describes the framework for content browsing which we have defined and implemented. In section 3 we list the features we have used in the TRECVID 2006 rushes task and describe how they have been used in the context of the browsing framework. Section 4 presents the evaluation method and results, and section 5 concludes with a discussion of these results and future work.

2 A Framework for Content Browsing

This section describes the framework that we have designed and implemented for building a video browsing tool. First we describe the concept on which the framework is based and then its feature-independent components.

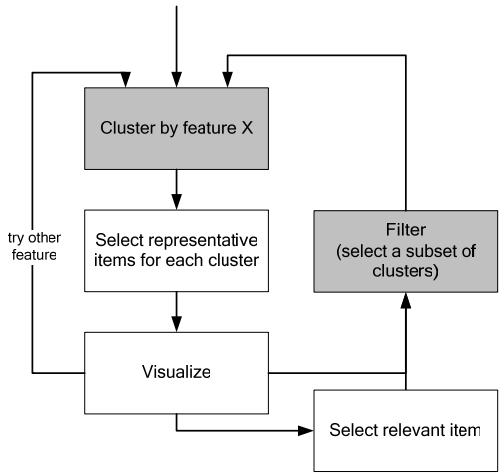

Figure 1: Conceptual approach to video content browsing.

2.1 Conceptual Approach

From the user's point of view, the basic difference between content browsing and search & retrieval is the limited knowledge of the user about how to formulate a query and about what to expect from the content set. Thus, the content browsing tool must support the user in building a query step by step, by trying to add new restrictions and reducing the content set when applying the chain of restrictions built up so far (cf. the ostensive model of developing information needs [3]).

Figure 1 shows the approach we are using for the system discussed in this paper. Starting from the full content set, the user restricts this set by the following iterative process:

Clustering. The content set is clustered in terms of a selected feature. Similar or even identical segments (with respect to the selected feature) are grouped in the same cluster. The clustering approach depends of course very much on the nature of the feature, i.e. whether it is has a discrete or continuous range of values, how the similarity between two instances can be measured etc.

Selecting representative items for each cluster. As the reduction of redundancy is the central goal of summarization, it does not make sense to show all items of a cluster. Thus, a set of representative items has to be selected. In our case we have only limited knowledge about the context (we only know that we clustered by feature X), thus this selection is not trivial (e.g. is it better to show only typical content of the cluster or are the outliers interesting?). Depending on the current size of the content set, often only a small fraction (a few percent or even less) of the segments can be selected to represent a cluster.

Visualization. The clusters are visualized using the selected media items. There may be a number of different visualizations, and the choice of the visualization may also depend on the features used for clustering.

At this point, it is the user's turn to do one of the following (the last two are exclusive):

Select relevant item. If the user finds relevant items, she probably wants to see more details about it in order to verify its relevance (e.g. watch a short clip of a video) and then select it (e.g. to keep it in a result list).

Repeat clustering. If the clustering result turns out to be not discriminative for the browsing task, clustering is redone using another feature. The process starts again with the first step.

Filter. In the case, that the user is not satisfied with the items found (which is the usual case in the first iterations), further restriction of the content set is necessary. The user will thus select clusters that seem to be relevant and discard the others. The remaining content set is the input to the clustering step in the next iteration.

2.2 Components

The framework consists of the following main components (as shown in Figure 2):

Data store. Essence and derived essence (such as key frames) are stored in the file system. The metadata needed for clustering are stored in a relational database when feasible, or in the file system, if specific index structures that cannot be easily integrated into a relational database system are needed.

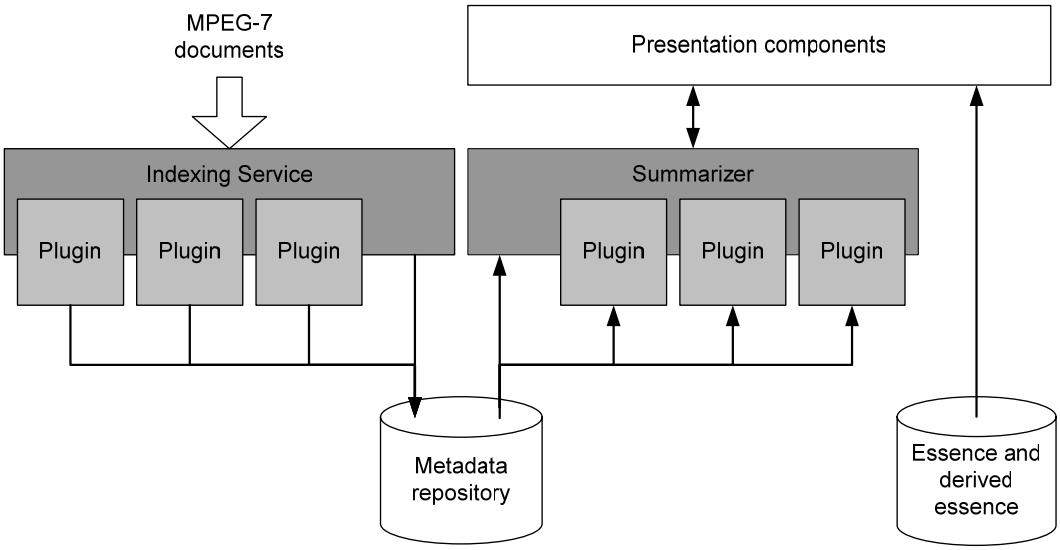

Figure 2: Components of the content browsing framework.

Indexing service. The indexing service is a background process that takes MPEG-7 descriptions (conforming to the MPEG-7 Detailed Audiovisual Profile [7]) as input (currently it watches a directory on the file system and processes the files inserted there) and indexes them in the relational database or other index structures. The indexing component does not process the essence itself. It is assumed, that a description of the content has been created before, either using automatic content analysis tools, by importing legacy metadata or by manual annotation. The indexing service itself only performs feature-independent tasks such as registering new content. The feature specific indexing is performed by a set of indexing plug-ins. Each plug-in handles a certain feature by reading the required information from the MPEG-7 description and creating the necessary database entries. New indexing plug-ins can be registered with the indexing service and will be executed when the next document is processed. In order to increase the response time of the summarizer, data structures that speed up clustering are directly created and updated by the indexing service.

Summarizer. The summarizer is the component handling clustering, filtering and selection of representative media items. Like in the indexing service, all feature-specific tasks (such as clustering itself) are delegated to a set of plug-ins. Each plug-in provides the clustering (and in some cases also item selection) algorithms for one feature. A plug-in takes a reference to the current data set as input and returns a cluster partition and optionally a list of items to be visualized for each of the clusters. The filter and item selection can be performed in the summarizer, as they are feature-independent. The summarizer has a defined interface towards the GUI layer, in order to allow the use of different visualizations resp. interaction paradigms.

User interface. The user interface visualizes the current content set and cluster structure and allows selecting items as input for filtering or as relevant results. Currently only a key frame based visualisation (light table) is implemented. Figure 3 shows the user interface of the video browsing tool. The central component is the light table view, which visualizes the clusters by a colored frame around the images, with the cluster label written above the first two images of the cluster. The controls around the light table view allow to scroll and to change the size of the key frames. The controls for selecting the cluster criterion and for filtering are placed in the tool bar. Relevant items can be dragged into the result list on the right.

The user starts from the complete content set. By selecting a cluster criterion the content will be grouped according to this feature. The user can then decide to try another cluster criterion on the same level, or select a subset of clusters, which are then the input to future clustering steps. At any time the user can select relevant items and drag them into the result list. The user can also always choose to view all segments that are in the current content set (practically this only makes sense after sufficient restrictions have been defined).

3 Features

This section describes the features that are used in our video browsing tool. The feature extraction is performed before ingestion into the system. The result of feature extraction is one metadata description per video, conforming to the MPEG-7 Detailed Audiovisual Profile [7]. The MPEG-7 descriptions are then ingested into the system as described above. In each plug-in of the summarizer, a specific clustering algorithm is implemented for each of the features. These clustering algorithms are also described in this section.

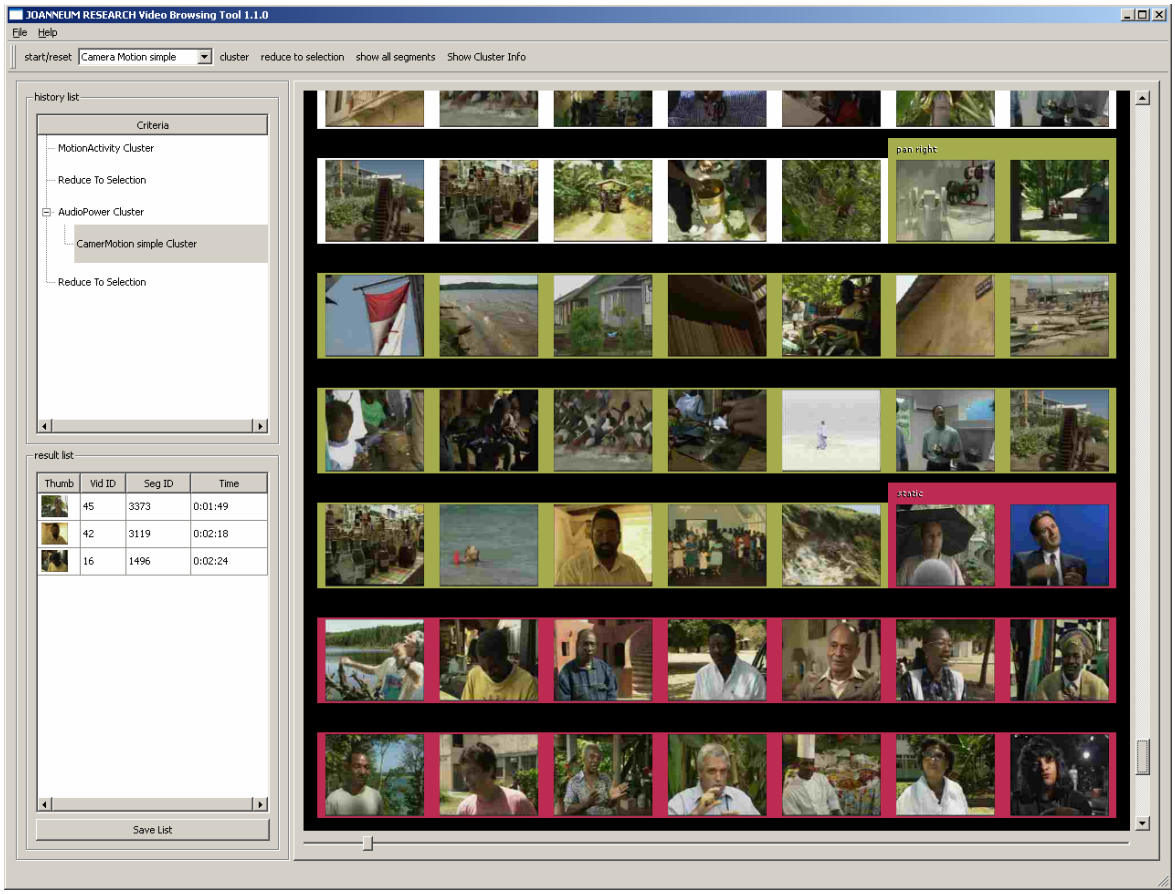

Figure 3: Screenshot of the video browsing tool.

3.1 Shot Boundary Detection and Key Frame Extraction

These features are not directly used for browsing, but for the extraction of other features discussed below. As shot boundaries are natural limits of the occurrence of most visual features (such as camera movements, objects), they are an important prerequisite for further visual feature extraction algorithms.

For each shot, a number of key frames is selected. The selection of key frame positions is based on the visual activity in the visual essence, i.e. the more object and/or camera motion, the shorter the time interval between two key frame positions. At least one key frame is extracted near the beginning of a shot. The key frames are not only needed for feature extraction, but also for visualisation in the browsing tool.

3.2 Camera Motion

The use of camera motion as a browsing feature is twofold: It is often used to guide the user's attention and express relevance of certain parts of the scene, for example, zooming on an object or person is an indicator of relevance, and in field sports, pans indicate the direction of the game. Secondly, it is an important selection criterion during editing, as visual grammar imposes constraints on the camera motion of sequences to be combined.

3.2.1 Extraction and Description

The detection of camera motion in an image sequence involves two basic problems. The dominant motion in the sequence is not necessarily the camera motion, e.g. if a large objects moves in front of a static camera, the dominant motion will be estimated as the object's motion. Different types of camera motion causing the same visual effect cannot be discriminated against one another, e.g. pan left and track left if target is distant and amount of motion is small.

Unlike other approaches, which ignore the fact that camera motion can only be determined reliably over a larger time range and which consider the most dominant motion between a frame pair as the camera motion, our approach is to estimate a number of

dominant motions. We assume that the camera motion is consistent and smooth over the time range and that it is the most dominant one (e.g. the one with the largest region of support).

The extraction algorithm is the same that we used for the TRECVID 2005 camera motion task [2]. It is based on feature tracking which is a compromise between spatially detailed motion description and performance. The feature trajectories are then clustered by similarity in terms of a motion model and the cluster representing the global motion is selected.

In contrast to the TRECVID 2005 camera motion task, we describe camera motion on a sub-shot level. A new camera motion segment is created, when there is a significant change of the camera motion pattern (e.g. a pan stops, in addition to a tilt a zoom starts). For each of these segments, the types of motion present in the segment and a roughly quantized amount of motion are described.

3.2.2 Clustering

We have implemented two options for clustering by camera motion. The first creates a fixed number of clusters, one for each type of camera motion (pan left, pan right, tilt up, tilt down, zoom in, zoom out, static). A camera motion segment is assigned to a cluster, if that type of camera motion is present in the segment, e.g. if a segment contains a pan left and a zoom in, it is assigned to both clusters.

The second clustering method tries to better model the actual data. Using the amounts of each type of motion, each camera motion segment is described by a vector in a three-dimensional feature space. The feature vectors are then clustered using the Mean Shift algorithm. The algorithm determines the number of clusters and assigns each camera motion segment to one of them. Due to the use of the optimized implementation described in [1] the clustering is fast. Depending on the data, the clusters contain single camera motions or combinations. From the strengths of the motion, a textual label for the cluster is created (e.g. “moderate pan and strong zoom”).

3.3 Motion Activity

Motion activity is a measure of the visual dynamics in a scene. Together with camera motion information, it is a measure for the local motion in a scene and can thus be used to discriminate quiet scenes from those with object motion.

3.3.1 Extraction and Description

In this setup, we do not use a detailed description of the motion activity, but we just measure the amplitude of visual change. A list of motion activity samples (we describe one value every five frames) is created for each shot. The list is then median filtered to be robust against short term distortions and split into homogenous segments. Each of these sub-shot segments is described by its average activity value.

3.3.2 Clustering

The motion activity segments are clustered using the k-means algorithm.

3.4 Audio Volume

Audio volume can for example be used to discriminate shots without any sound, calm shots of inanimate objects, interviews with a constant volume level and loud outdoor shots in city streets.

3.4.1 Extraction and Description

As we do not have an algorithm for audio segmentation (e.g. by the audio class) at hand, we start with segments of a fixed length of 30 seconds. A list of audio volume samples is extracted for each of these segments by calculating the average volume of a 0.5 seconds time window. The list is then median filtered to be robust against short term distortions and split into homogenous segments. Each of these sub-segments is described by its average volume value.

3.4.2 Clustering

The audio volume segments are clustered using the k-means algorithm.

3.5 Face Occurrence

The occurrence of faces is a salient feature in video content, as it allows inferring the presence of humans in the scene. The size of a face is also a hint for the role of the person, i.e. a large face indicates that this person is in the center of attention.

3.5.1 Extraction and Description

Our extractor is based on the face detection algorithm from OpenCV [8]. In order to make the description more reliable and to eliminate false positives, which mostly occur for a single or a few frames, we only accept face occurrences that are stable over a longer time (we use a time window of about a second to check this). As a result we get a continuous segmentation into sub-shot segments with and without faces.

3.5.2 Clustering

There is no real clustering done here, as there are only the two groups of segments: the ones containing faces and the ones not containing faces.

3.6 Global Color Similarity

Global color similarity allows to group shots that depict visually similar content, e.g. several takes of the same scene or different shots taken at the same location (if the foreground objects are not too dominant).

3.6.1 Extraction and Description

To describe the color properties of a shot, the MPEG-7 ColorLayout descriptor [5] is extracted from each key frame. The ColorLayout descriptor has the advantage of also taking the spatial color distribution of the image into account. In order to reduce the number of color descriptions to be processed, similar descriptors extracted from key frames of the same shot are eliminated. Then the pair-wise similarities between all remaining descriptors of the content set are calculated and stored in a matrix.

3.6.2 Clustering

The similarity matrix is used as input for hierarchical clustering using the single linkage algorithm [4]. The cutoff value for the resulting tree is determined from the number of desired clusters.

3.7 Object Similarity

To overcome the limitations of global color similarity, we additionally use an interest point based approach to detect similar objects in the content. This approach does not only rely on color information and does not require global visual similarity. As we do not perform segmentation, an “object” means in this context a set of matching interest points that are detected in an image. Thus, the approach will also respond to partially similar objects or similar parts of the background.

3.7.1 Extraction and Description

Our algorithm uses a combination of the SIFT [6] and MPEG-7 color descriptors. The interest points are extracted using the difference of Gaussian pyramid originally proposed for SIFT. The MPEG-7 color descriptors are extracted from a window around the interested points. The descriptor for an image contains the SIFT and MPEG-7 descriptors for the interest points detected in the image. The details of the algorithm are described in [9].

The extraction is performed on the key frames. Like for the global color descriptors, similar descriptors within a shot are eliminated and a matrix of pair-wise descriptor similarities across the whole content set is calculated.

3.7.2 Clustering

The similarity matrix is used as input for hierarchical clustering using the single linkage algorithm [4]. The cutoff value for the resulting tree is determined from the number of desired clusters.

4 Evaluation

There are a number of problems related to the evaluation of the video browsing tool. In the following, we list some of them and the choice we have made for our evaluation method.

Formulation of tasks. Using a formulation that is very close to the features used by the system reduces the room for interpretation by the user and facilitates evaluation. On the other hand, a less precise formulation is closer to a real-world scenario, especially in browsing, where the user will not have a clearly specified query in mind. We have thus used textual descriptions of the content to be retrieved, containing some references to features, but not precisely specifying the shot to be found. In addition, some of the queries only have very few matches, while others are more general. The drawback of our decision is that some of the tasks require interpretation by the user (e.g. “how can I decide if this is a tourist?”).

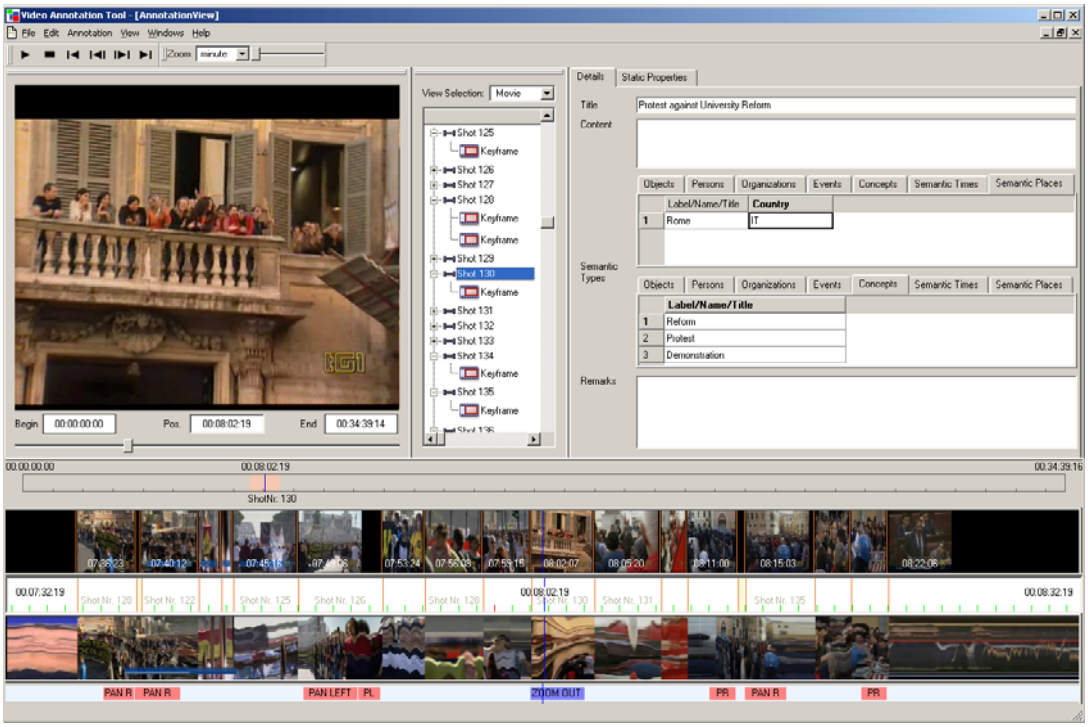

Figure 4: Video annotation tool used for ground truth annotation.

Measuring precision. Recall can be easily derived from the number of items found by the test user and those available in the ground truth. The precision measure depends very much on the chance of the user to check each potentially relevant item before selecting it, i.e. if the user can view a short clip of video for each of those items, precision will be close to 1. We thus decided not to allow the use of a video player in order to measure the precision that can be achieved when using only the browsing tool.

Unit of retrieval. The first aspect of this problem is the question which is the unit retrieved when the user selects a relevant item. Without the possibility of having a video player and selecting in and out point, this decision will be taken by the tool for the user. The second aspect is that of evaluation: which part of a segment must be found to accept it as correct? As shots are much too long in unedited material, there is no "natural unit" in the content. In our framework, there are specific sub-shot segmentations for each feature, depending on the value of the feature. A user will choose a certain combination of features to locate the desired content, but we cannot decide which segmentation is appropriate. We also believe that this is not so much a practical problem, as – once a matching segment is found – it is quite simple to determine which time span before and after it is relevant. Thus, we decided to count each overlap between ground truth and retrieved segment as correct.

4.1 Ground Truth

In order to measure recall, it is crucial that all of the test data is investigated during ground truth annotation. As it was not feasible to manually annotate the complete test set, we made a thematic restriction. The task for the annotator was to do annotation on all shots that are in a village/town/city environment, showing streets, people, buildings, cars or monuments. For these segments, a description of the location, salient objects and persons and concepts was made.

The video annotation tool shown in Figure 4, which we have recently developed within PrestoSpace (http://www.prestospace.org), has been used for ground truth annotation. In order to support the user, automatic content analysis was performed before, so that shot boundaries, key frames and stripe images are available as navigation support. Using the tree structure, the user can create a segmentation of the content into any kind of segments. In the right part of the window, location, objects, persons, etc. can be annotated.

4.2 Test Procedure

For the test, seven colleagues and students from our institute have served as test users. None of them has been involved in the development of the software, all of them have been using the tool for the first time and none of them has seen the TRECVID rushes data before. Each user got a short introduction of about 10-15 minutes about the functionality and the use of the tool and

then a list of browsing tasks. The time for the completion of each of the tasks was 10 minutes. The four tasks done by all users and used for evaluation were:

T1. Find segments showing an interview with an African man with a yellow shirt.

T4. Find segments showing tourists visiting a town/city.

T6. Find a zoom in on an interviewee.

T7. Find a segment showing a street in a town/city, with cars (parking or driving) and pedestrians, containing a pan along the street.

4.3 Measures

The measures used for evaluation are the following:

- Recall: the fraction of the correctly selected segments in relation to all segments in the ground truth matching the task.
- Precision: the fraction of the correct ones in relation to all selected segments for this task.
- The total time needed for completing the task (max. 10 minutes).
- The number of items selected (both correct and false) within the working time, measured every 30 seconds.

4.4 Results

Figures 5 and 6 show the results of the evaluation of the four browsing tasks. Precision and recall are plotted in Figure 5. As there are cases, where it is not always clearly decidable whether a segment should count as a correct result for a task description, we have decided to perform two evaluations. In the strict one, all the questionable segments are classified as false positives, in the less strict evaluation they are counted as correct. As a result, the precision values are typically higher for the less strict case, while the recall values are comparable. The results of the less strict evaluation are marked with “a” in Figure 5. Table 1 shows the number of ground truth segments for each of the tasks for both the strict and less strict evaluation.

It can be observed that the recall is always much lower than the precision. This means that the segments that are found are relevant to the query, but that there is always a number of relevant segments in the content that have not been discovered

We have also measured the time needed for completion of the task and the number of items found within 30 seconds intervals during the working time. The average time for completion of the task was 7.6 minutes, ranging from 5.4 (task 6) to 9.5 (task 7). Figure 6 shows the number of items put into the result list during the working time. Although these values are very different and non-linear for the individual users, this is leveled out in the average for one task and the overall average is surprisingly close to linear.

There are some task specific results that are noteworthy. Task 6 (“zoom in on interviewee”) is the one that yields the worst results, and at the same time the one with the shortest average working time. This means that the users were convinced that they had found everything they could find for this task quite soon. There is also a rather low precision, as the task was often interpreted as “zoom in on a person in front of the camera”, so that the result contained many segments that were not from an interview situation.

Task 7 (“street in city, cars and pedestrians, pan”) is also interesting, as it is the only task for which there were users who submitted an empty result set (3 out of 7). In addition, users found first results not before minute 3 of the working time. If we only analyze the four non-empty result sets, we get better precision and recall values than for all other tasks (T7 non-empty and T7a non-empty in Figure 5). It seems that some users failed to figure out how to use the browsing tool for this task while those who did, got very good results.

We have also evaluated the agreement between the users. The results are shown in Figure 7. None of the segments has been found by all users. Interestingly the task with the worst overall performance has the highest agreement among the different users.

5 Conclusion and Future Work

We have developed a tool for browsing redundant and un-annotated video collections. The tool is based on a generic and feature-independent framework, which supports iterative clustering and filtering of the content set. Plug-ins for a number of features have been implemented. A light table view is used for the visualization of the summarized content.

The evaluation has shown that the improvement of recall is the most important task for the future. This is also the greater challenge than the improvement of precision. If some more context information from the video can be viewed and a video player is available, the user can easily and efficiently judge the relevance of the few selected items, which will significantly increase precision. The reasons for low recall are more complex: The user chooses a chain of features to restrict the content set. Relevant

segments may not be included because there are not always appropriate features available which are discriminative for a browsing task. Segments may not be included in a set because feature extraction has not been reliable enough. A user sometimes discards a cluster or does not follow a path because the visual representation of the clusters suggests this, although relevant segments are contained therein.

Task	T1	T4	T6	T7
Number of segments (strict)	5	29	9	7
Number of segments (less strict)	6	33	9	10

Table 1: Number of segments in the ground truth for each of the tasks.

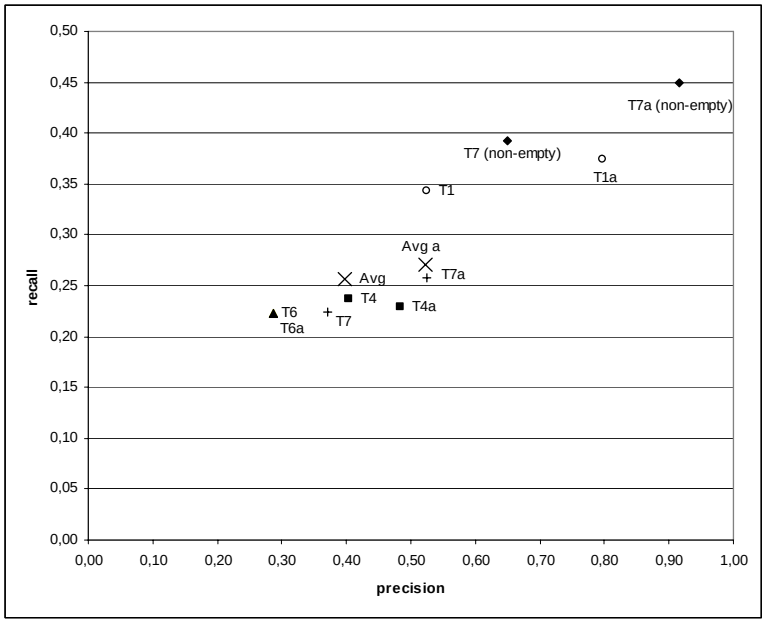

Figure 5: Precision/recall plot for the browsing tasks. The values marked with "a" are the results of the less strict evaluation. The "non-empty" values for task 7 are calculated only from the submissions with non-empty result sets.

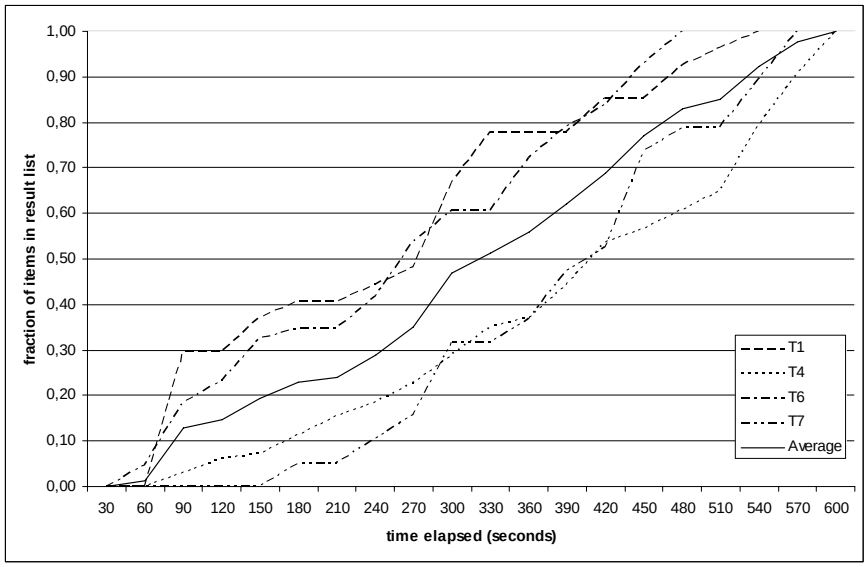

Figure 6: The number of items put into the result list during the working time for the browsing task.

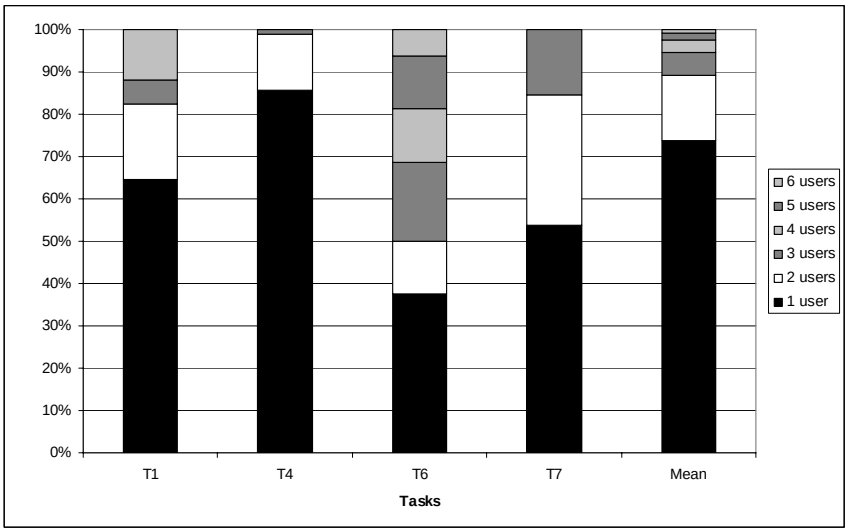

Figure 7: The bars show the relative amount of segments found by n users for each of the tasks and the average across the tasks. None of the segments has been found by all users.

Apart from the measures presented above the evaluation has brought a lot of useful feedback from the users concerning the functionality and the usability of the tool. Among the ones mentioned most was the wish for a history of the clustering and filtering operations and the ability to move forwards and backwards in the history like in a web browser.

6 Acknowledgements

The authors would like to thank their colleagues at the Institute of Information Systems of JOANNEUM RESEARCH for their support and feedback, especially Philipp Schügerl, Andreas Kriechbaum and Roland Mörzinger for providing some of the feature extractors, Jürgen Oberngruber, Harald Csaszar and Xavier Chisvert Valero for their contributions to the browsing tool and Martin Lang for doing the ground truth annotation. Special thanks also go to the colleagues and students who volunteered to be test users for the evaluation.

The work presented here has been partially funded under the 6th Framework Programme of the European Union within the IST project "IP-RACINE – Integrated Project Research Area Cinema" (IST-2-511316-IP, http://www.ipracine.org). This work contributes to the activities of the IST research network of excellence K-SPACE (FP6-027026, http://www.k-space.eu) funded by the 6th Framework Programme of the European Union.

7 References

[1] W. Bailer, P. Schallauer, H. Bergur Haraldsson and H. Rehatschek, "Optimized Mean Shift Algorithm for Color Segmentation in Image Sequences". In: *Image and Video Communications and Processing*, San Jose, CA, USA, Jan. 2005, pp. 522—529.

[2] W. Bailer, P. Schallauer and G. Thallinger, "Joanneum Research at TRECVID 2005 – Camera Motion Detection". In: *Proceedings of TRECVID Workshop*, Gaithersburg, MD, USA, Nov. 2005.

[3] I. Campbell and K. van Rijsbergen, "The ostensive model of developing information needs". In: Proceedings of CoLIS 2, Copenhagen, DK, 1996. pp. 251-268.

[4] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*. 2nd ed., Wiley-Interscience, 2001.

[5] Information Technology—Multimedia Content Description Interface, Part 3: Visual. ISO/IEC 15938-3, 2001.

[6] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the International Conference on Computer Vision*, 1999, pp. 1150—1157.

[7] MPEG-7 Detailed Audiovisual Profile (DAVP). URL: http://mpeg-7.joanneum.at

[8] Open Source Computer Vision Library (OpenCV). URL: http://sourceforge.net/projects/opencvlibrary

[9] P. Schügerl, *Object Recognition for Video Content Analysis and Surveillance*. Diploma Thesis, Upper Austria University of Applied Sciences, Dept. of Software Engineering, Hagenberg, 2006.