Query On Demand Video Browsing

Ork de Rooij
Intelligent Systems Lab Amsterdam, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
orooij@science.uva.nl

Cees G. M. Snoek
Intelligent Systems Lab Amsterdam, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
cgmsnoek@science.uva.nl

Marcel Worring
Intelligent Systems Lab Amsterdam, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
worring@science.uva.nl

ABSTRACT

This paper describes a novel method for browsing a large collection of news video by linking various forms of related video fragments together as threads. Each thread contains a sequence of shots with high feature-based similarity. Two interfaces are designed which use threads as the basis for browsing. One interface shows a minimal set of threads, and the other as many as possible. Both interfaces are evaluated in the TRECVID interactive retrieval task, where they ranked among the best interactive retrieval systems currently available. The results indicate that the use of threads in interactive video search is very beneficial. We have found that in general the query result and the timeline are the most important threads. However, having several additional threads allow a user to find unique results which cannot easily be found by using query results and time alone.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Performance

Keywords

multi dimensional browsing, video retrieval, conceptual similarity, semantic threads, interactive search

1. INTRODUCTION

Nowadays there is an ever increasing amount of video material available about various subjects. Video fragments can be retrieved quickly if you know the exact title of the fragment. If this is not known a content based video search system can be used to find the fragment. However, current content based video retrieval results are far from optimal. In order to improve on this the set of results from such a system should be taken as the start for further exploration.

MM'07, September 23–28, 2007, Augsburg, Bavaria, Germany.

Starting from a given result set we can explore in many different directions depending on the type of query and the intention of the user. The visualization of the result determines the navigation possibilities. In [6] the collection is visualized as a cloud of images by mapping a high dimensional feature space to the screen. This allows for very unconstrained navigation. Informedia [3] proposes to visualize the result set in a regular grid and uses rapid serial visual presentation to efficiently explore this set. Both methods put a high cognitive load on the user. A graph like structure such as [1] allows for a more constrained navigation, but navigating large dynamic graphs of video material is still difficult because of the size of such a graph. The above methods use one modality. In order to give the user more navigation options, a multimodal representation should be considered. The system of [4] combines several modalities, including results, time and a visualization of semantically related shots in a graph. Each modality is however shown in a separate visualization. Separate windows lead to more visual clutter, which makes user response slower. In [7] similarity and time are combined into one visualization. In all of the above systems navigation is either too unconstrained, or too separated. We propose a system where all modalities are combined into one consistent representation. This representation allows advanced explorative browsing through the collection without cluttering the interface.

The organization of this paper is as follows. The browsing method is explained in section 2, with a prototype visualization method described in section 3. An evaluation of the system at NIST TRECVID is described in section 4, followed by a conclusion in section 5.

2. VIDEO THREADS

In our system, the basis for navigation is the thread τ, defined as a linked sequence of camera shots from various videos in some specified order. Many different threads can be computed. We define the following threads: A textual thread τ_t containing shots with similar textual annotation. A visual thread τ_v with visually similar shots constructed from low-level visual features. A timeline thread τ_T containing all shots ordered by their original timeline. A semantic thread τ_s containing semantically equivalent shots constructed from high-level textual and visual features. And finally an initial query thread τ_q containing a list of results formed by a standard query interface.

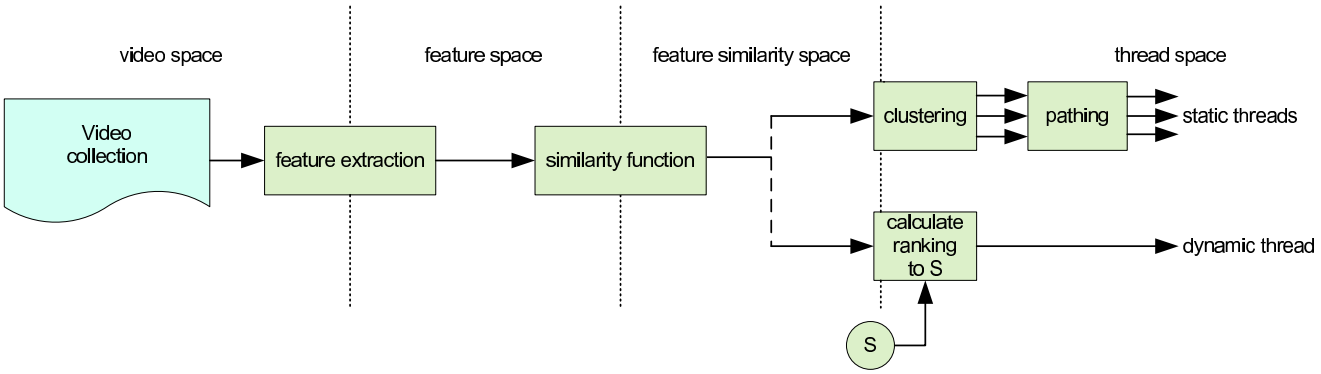

Figure 1: Generic framework for thread computation. Feature extraction is used to compute the feature space, which is transformed into feature similarity space. From here the framework computes dynamic threads by using the distance to the focal shot as ranking measure. The framework computes static threads by clustering the similarity space, subsequently creating a path through each cluster.

2.1 Features and Similarity

The generic framework for creating a thread is as follows. Given a set of features for each shot and a similarity function, we create a feature-based similarity space. From this space we can create both static and dynamic threads. See figure 1 for an overview.

To determine visual threads τ_v we need a notion of visual similarity between shots. We propose the following method. The visual indexing starts with computing a high dimensional feature vector for each shot. In our system we use Wiccest features with related similarity function from [2], and Gabor features. This gives us the *visual similarity space*. This space forms the basis for visual exploration of the dataset.

A dynamic textual thread τ_t is obtained by performing a text search through textual meta-data of shots, which can include on screen text and the output of automated speech recognition systems. The set of shots which is returned from such a query forms a dynamic textual thread.

To determine the semantic threads τ_s we first need a way of extracting the semantic content from a video. We use the method described in [9] which provides us with a measure of presence P_j for a given concept j in each shot. Given a large enough lexicon of concepts we can approximate the semantic content of a shot with a single semantic concept vector. We use the lexicons as defined by LSCOM[5] and [10], which when combined result in 491 concepts. These concepts range from generic to specific classes of people, settings, objects or events.

Given two semantic concept vectors and a similarity function S_C shots can now be compared based on their semantics. The semantic similarity space induced by S_C is complex as shots can be related to several concepts. Typically only a few concepts will be available in a shot, so S_C should be chosen such that it explicitly values shots that share the same concepts. Here S_C is defined as in eq.1, which resembles histogram intersection, adding the minimum probability of the two shots for concept j over the whole vector P. This yields the *semantic similarity space*, which forms the basis for semantic exploration of the dataset.

$$S_{pq} = \frac{\sum_i min(p_i, q_i)}{\sum_i max(p_i, q_i)} \qquad p, q\text{: the feature vectors} \qquad (1)$$

2.2 Static and Dynamic threads

To allow the user to recognize his location in the collection within a search session, or between sessions it is beneficial if there is some rigid structure in this collection, so that the user can use this structure to navigate. However, to allow the user to search through the collection optimally on demand querying should also be possible. Because of this we make a distinction between threads that are precomputed in advance - the so called static threads, and threads that are shown on demand - the dynamic threads.

We propose the following method for static thread calculation. The similarity space of a large video collection can be too large to consider at once. Because of this we first cluster the space into groups of shots with a high similarity among each other. To find such groups we perform k-means clustering in the similarity space. In our experiments we choose $k = 200$. Ordering of these elements is done by applying a shortest path algorithm inside the cluster. So, shots with similar content are close to each other in the thread. This results in a static thread for every cluster. Every shot in the collection is part of one such thread. This gives the user a neighbourhood of similar shots in that modality.

Dynamic threads are always based on a shot S specified by the user. They are created by calculating the similarity distance for all shots to S, and ordering the top-ranked shots in a list. This list can then be used as a new dynamic thread with S as starting point.

3. THREAD VISUALIZATION

Now we know what threads are and how to create them, we define a method for visualizing them on a regular screen so they can actually be used in explorative browsing.

We form navigation paths from a focal shot S, which identifies the current position in the video collection. In order to highlight S, we chose a visualization where S is always the largest most centered shot on the screen. All threads which contain S are added as navigation possibilities. These are shown in a star formation around S. During the search process only two navigation options are possible: accept a shot as relevant, or navigate away from S into any connecting thread. This can be done by selecting any visible shot as the new S.

An interface that shows all possible navigation paths to

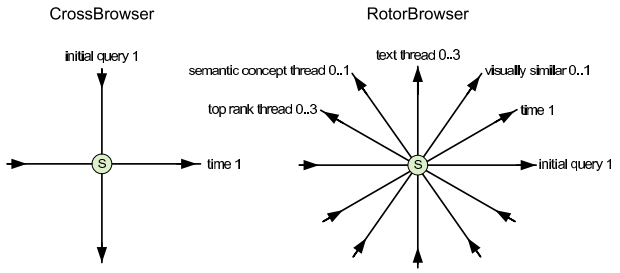

Figure 2: Multi thread dimensions for each browser, where the number indicates the possible times this dimension can be present for any shot

Figure 3: Screenshot of the multi thread visualization with the CrossBrowser (left) and the RotorBrowser (right)

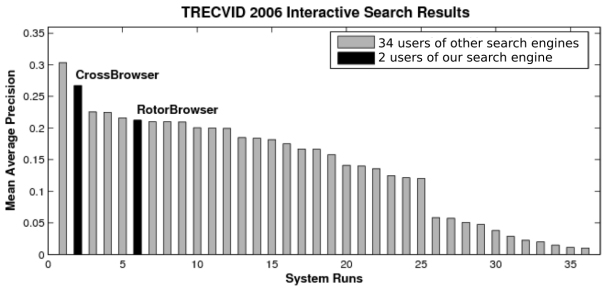

Figure 4: Overview of all interactive search runs submitted to TRECVID 2006, ranked according to Mean Average Precision

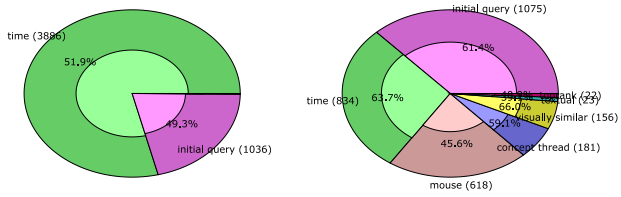

Figure 5: These graphs shows how many results were obtained from each thread, and which percentage of them was judged relevant. Left: CrossBrowser results. Right: RotorBrowser results.

the user might be too overwhelming. An interface that only shows a minimal set of navigation paths might not show the optimal path. In order to determine the maximum number of paths which can be shown to the user, we have implemented both variants of this browser. One variant - the CrossBrowser - limits the number of threads that are shown. It shows the user the results from their initial query and only allows to browse through the time thread which is known to be important [7]. The other variant - the RotorBrowser - shows all relevant threads, including time, for the query shot S. This difference is visible in figure 2.

The RotorBrowser also allows the user to leave the initial query result set. This enables the user to browse through anything that catches his interest, though it also increases the chance that the user reaches the end of a thread. To prevent this, static threads are available as a guideline to the RotorBrowser. The contents of a static thread are precomputed and do not change during the browsing session, only the part that is displayed of a static thread changes. This helps the user to recognize his location within the dataset. Because the underlying semantics of video material are more important than the looks of the video we chose to use semantic features for calculating a series of static threads for guiding the user in this browser, and keep the other threads dynamic. To allow for quick navigation, hotkeys were added to allow jumps back to last known relevant shots. A screenshot of this visualization is visible in figure 3.

4. RESULTS

Both browsers were evaluated by participating in the TRECVID Interactive Retrieval effort (see [8]). This allows for a comparison between the entire system and the current state of the art in video retrieval systems. The experimental setup is as follows. Each participant is given a list of topics for which they need to find results. For each topic the participant has 15 minutes to interactively use their system to find results within a collection of 160 hours of news video from English, Arabic and Chinese broadcast channels. This dataset is not known beforehand. The results are judged using a pooling mechanism by TRECVID. For every topic the Average Precision, a one valued measure related to precision and recall, is then determined. Because this score does not penalize incorrect shots at the end of the list we instructed our participants to find as many shots as possible, even though some will be invalid. The last 2 minutes for each topic were reserved for sorting the end result so that the best results are ranked higher.

The Average Precision per topic is finally averaged over all the topics, resulting in one Mean Average Precision score per system.

Overall our browsers perform well in the TRECVID evaluation. The CrossBrowser placed 2nd and the RotorBrowser placed 6th in the overall results. See figure 4. When we compare the RotorBrowser to the CrossBrowser a significant gap in results can be seen. However because both runs were performed by expert users with varying levels of experience in performing TRECVID retrieval tasks, we cannot compare the results directly. The experience and capabilities of each user influences the result also. We can however deduce some interesting facts from the results.

During the interactive retrieval effort we have kept detailed log files of each user action. If we look at the originating thread of each shot in the results from both browsers (figure 5) we can see the following. Results gathered from

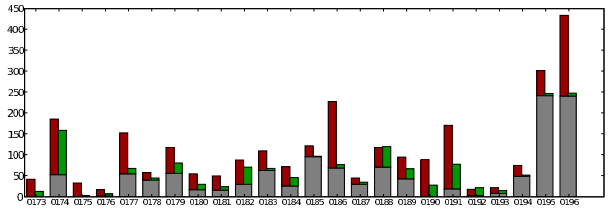

Figure 6: This graphs shows the number of unique results found per topic for each browser. Grey bars: number of shots both browsers were able to find. Red bars: unique CrossBrowser results. Green bars: unique RotorBrowser results.

the CrossBrowser indicate that more than 75% of the results are selected using the time thread. This can be explained by the fact that selection is done at shot level, while many of the TRECVID topics can be answered at story level: when a valid shot is found there is a good chance that the shots near that shot are also valid. For the RotorBrowser the largest portion of results is also generated from the initial query results and the time thread. From the other threads the semantic concept thread and the visual thread were most used. The text thread was seldom used because these were only visible when the user explicitly performed a text search. Interestingly, the top-rank threads are rarely used, even though they take up the most screen real estate. This means that showing these threads is not wise at all. The user has to process all visible top-rank threads mentally even though they are probably not usable. A quarter of the selected shots were generated by using mouse interaction. However from this part only 45.6% of the shots were judged correct. This stands in contrast when compared to the initial query results and the time thread, where approx. 64% was judged correct. After questioning the participant about this we discovered that the mouse was only used when the participant saw a possibly valid shot on the visible edge of some thread on the screen. If the shot was closer to the center he used the keyboard, since this allowed for quicker navigation. The fixed layout of the RotorBrowser also resulted in these shots being the smallest on screen, which could account for the user making the most mistakes in determining if a shot was correct.

When we look at which shots were selected by each browser we see the following. For each topic there are shots which were found by both browsers, and unique shots which were only found by one browser. This difference is visible in figure 6. The number of unique results per browser is relatively large, indicating that both browsers find different results. This can be explained as follows. The CrossBrowser displays only two threads. As a consequence a user has to browse these threads for a longer time to find results. We call this deep browsing. The RotorBrowser displays many threads. The user is able to select results from all these threads, but because of increased navigation options the individual threads will be navigated less extensively. We call this shallow browsing. The large number of unique results for each browser seems to indicate that both techniques have merit. Ideally a browser which displays "just enough" threads might be the optimal solution. More research is required to design such a hybrid between browsers, so both shallow and deep browsing is combined.

5. CONCLUSIONS

Overall our results indicate that using multiple modalities in one browser is beneficial. Both browsers score high marks in the TRECVID benchmark. Results from the CrossBrowser indicate that time is a very important factor, and that using this in conjunction with results from a query screen is very important. Both browsers are able to find different sets of results. The results from the RotorBrowser indicate that both visual similarity and conceptual similarity are beneficial for finding unique results. More tuning is required to allow the RotorBrowser to show the most relevant threads automatically. In order to further evaluate the differences between both browsers we are currently performing a large non-expert user study.

6. ACKNOWLEDGMENTS

This research is sponsored by the BSIK MultimediaN project.

7. REFERENCES

[1] C. Chen. Bridging the gap: The use of pathfinder networks in visual navigation. *JVLC*, 9(3):267–286, 1998.

[2] J. Gemert, J.-M. Geusebroek, C. J. Veenman, C. G. Snoek, and A. W. Smeulders. Robust scene categorization by learning image statistics in context. In *SLAM - CVPR*, NY, USA, June 2006.

[3] A. G. Hauptmann, W.-H. Lin, R. Yan, J. Yang, and M.-Y. Chen. Extreme video retrieval: joint maximization of human and computer performance. In *ACM Multimedia*, pp. 385–394, NY, USA, 2006.

[4] D. Heesch, P. Howarth, J. Magalhães, A. May, M. Pickering, A. Yavlinksy, and S. Rüger. Video retrieval using search and browsing. In *Proc. TRECVid*, Gaithersburg, MD, 2004.

[5] M. Naphade, J. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE Multimedia*, 13(3):86–91, 2006.

[6] G. P. Nguyen and M. Worring. Similarity based visualization of image collections. In *Seventh DELOS on audio-visual content and information visualization in digital libraries*, May 2005.

[7] M. Rautianen, T. Ojala, and T. Seppänen. Cluster-temporal browsing of large news video databases. In *IEEE ICME*, pp. 751–754, Taipei, Taiwan, 2004.

[8] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR*, pp. 321–330, NY, USA, 2006.

[9] C. G. M. Snoek, M. Worring, J.-M. Geusebroek, D. C. Koelma, F. J. Seinstra, and A. W. M. Smeulders. The semantic pathfinder: Using an authoring metaphor for generic multimedia indexing. *IEEE TPAMI*, 28(10):1678–1689, 2006.

[10] C. G. M. Snoek, M. Worring, J. van Gemert, J.-M. Geusebroek, D. Koelma, G. P. Nguyen, O. de Rooij, and F. Seinstra. Mediamill: exploring news video archives based on learned semantics. In *ACM Multimedia*, pp. 225–226, NY, USA, 2005.