

Balancing Thread Based Navigation for Targeted Video Search

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

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

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

ABSTRACT

Various query methods for video search exist. Because of the semantic gap each method has its limitations. We argue that for effective retrieval query methods need to be combined at retrieval time. However, switching query methods often involves a change in query and browsing interface, which puts a heavy burden on the user. In this paper, we propose a novel method for fast and effective search through large video collections by embedding multiple query methods into a single browsing environment. To that end we introduced the notion of query threads, which contain a shot-based ranking of the video collection according to some feature-based similarity measure. On top of these threads we define several thread-based visualizations, ranging from fast targeted search to very broad exploratory search, with the ForkBrowser as the balance between fast search and video space exploration. We compare the effectiveness and efficiency of the ForkBrowser with the CrossBrowser on the TRECVID 2007 interactive search task. Results show that different query methods are needed for different types of search topics, and that the ForkBrowser requires significantly less user interactions to achieve the same result as the CrossBrowser. In addition, both browsers rank among the best interactive retrieval systems currently available.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance

Keywords

video retrieval, interactive search, thread based browsing, conceptual similarity

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIVR'08, July 7-9, 2008, Niagara Falls, Ontario, Canada.
Copyright 2008 ACM 978-1-60558-070-8/08/07 ...\$5.00.

1. INTRODUCTION

Nowadays huge video archives are available from various domains, such as broadcasters, news agencies, YouTube, and home video archives. Users are accustomed to browsing, navigating and searching through these archives using text-based retrieval engines. The indices of these engines are based on filenames, speech-transcripts, or social tags provided by online users. However, it is well known [23] that textual descriptions of these sources do not capture the dynamic visual content of video adequately. Thus yielding questionable performance when a user wants to find something in the collection that has not been annotated as such.

To enable video content retrieval, a lot of research is being done in automated video content analysis techniques, e.g. [15][25][18][27]. In general, these techniques offer low-level access to the video content. Unfortunately specifying a search query using low-level features is a difficult task for a human interested in a specific fragment of video. As a consequence the video retrieval results are sub-optimal. Hence, the user has to look through many possible results, which slows down the retrieval process. Moreover it is not guaranteed that the user finds the fragment she was looking for.

One solution to the problem of having to browse through long lists of results, is to offer a rich, exploratory, user interface. In such an interface the user is not limited to browsing only through the initial results, but also through related items obtained through the use of various query methods. These will typically be offered in a rich user interface. Using this, the user is able to explore the collection, rather than just the initial list of results. A benefit of such a method is less dependence on the quality of automated techniques, at the cost of speed of retrieval.

In contrast with this solution, another solution is to further optimize the browsing technique employed so that it delivers fast, targeted search through the query results. This increases the dependence on automated query answering techniques, but because browsing is much more efficient such an interface is able to yield similar performance.

We distinguish two techniques for video retrieval. On the one hand we have *targeted search* with a focus on good automated retrieval techniques, and fast browsing through a single list of results. On the other hand we have *exploratory search* which allows the user to browse into related results based on what he sees fit. In this paper we will propose a balance between both techniques by introducing a browse method which optimizes speed of retrieval by embedding

multiple query methods into a single visual browsing environment.

1.1 Query methods for video retrieval

In multimedia retrieval literature a wide gamut of query methods have been proposed, applied, and evaluated, see [15][10] for an extensive overview. We focus here on the ones that are known to be effective for interactive video search, namely query-by-keyword, query-by-example, query-by-concept, and their combination.

One of the earliest and most successful methods for video retrieval to date is query-by-keyword. Query-by-keyword allows a user to enter textual keywords to find video content based on textual metadata that can be associated to a video. The implicit assumption is that the textual metadata accurately captures all the multimedia content of the video. Unfortunately this is not always the case. Imagine for example a search for a celebrity talking on a telephone. It is probable that the name of the celebrity will be tagged in the collection, however the fact that he or she talking on a telephone is not interesting, and is therefore not tagged, so this fragment cannot be found without watching all fragments of the celebrity. Moreover, social tags or textual transcripts of non-English languages may be error prone or simply missing. Thus for effective video search query-by-keyword needs to be extended with methods that incorporate analysis of the visual content.

For querying the visual content, the method of choice has long been query-by-example. These methods reduce a video shot, the fragment of video between camera cuts, to a single keyframe. This keyframe is then represented by a combination of color, texture or shape based features. A user is able to query a video collection represented by visual feature vectors by providing an example image [4][21], or sketch [8][24]. By comparing the provided visual feature representation to the database, the visual similarity between user query and the video collection is determined. An obvious downside of this query method is its dependence on dedicated user input in the form of existing example images or sketches. Moreover the low-level visual feature representation used for querying often does not correspond to the users intent. This problem is known as the semantic gap [23]. In order to reduce the semantic gap we need more than visual features alone.

In an effort to reduce the semantic gap, a recent alternative to query-by-example relies on so called concept detectors. Generic concept detection methods have emerged lately [18][27], which allow automatic labeling of people, objects, settings or events within the video content. In general these automatic methods are based on fusion of various invariant visual features in combination with supervised machine learning using large collections of labeled image examples. Collections of concept detectors form a lexicon. Examples from a representative lexicon such as LSCOM [17] may vary from people like *Hu Jintao*, objects like a *truck*, settings like *indoor* and events like *people marching*. A concept detector lexicon allows query-by-concept. This resembles query-by-keyword like search through the visual information of the video. Unfortunately due to varying performance of individual detectors and ever increasing sizes of lexicons this leads to new problems for the user: which detector should he use? Which one is good enough? And can they be combined? Query-by-concept delivers an interesting starting point for visual search but by itself it is not the ideal solution yet.

The above query methods each yield a ranking of the video collection. From there it is left to the user to select relevant keyframes from this ranking. The quality of these rankings are dependent on both the precision of the used query method and the skill of the user in specifying query parameters. Therefore, the quality of the results is not clearly defined and often inadequate. This requires the user to view hundreds of results before he retrieves a few good ones. Furthermore, over the years the number of available video query methods has expanded enormously. Every time a new query method was introduced query interfaces had to cope with yet another set of parameters which the user needed to configure. This yielded highly complex search interfaces. To assist the user with the problem of which query method to use, a lot of work has been done in visualization methods for video search.

1.2 Visualization methods for video search

As stated earlier there are two distinct techniques for video retrieval. *Targeted search* focuses on exploiting automated retrieval techniques, and uses fast browsing techniques to browse through the results obtained by these techniques. *Exploratory search* allows the user to have various ways to control the browsing process. It allows the user to browse into related shots based on what he sees. We therefore categorize related work into visualization methods which primarily focus on targeted search, and in visualization methods which primarily focus on exploration.

In the first category systems typically use automated querying techniques to generate a list of results, which is then rapidly explored by the user. For example, the Informedia XVR interface[11] uses rapid serial visual presentation taken to the extreme to explore results at high speed. The idea behind this is the following. The system obtains initial results by using automated techniques only. Subsequently, the user interactively filters these at very high speeds to weed out the bad results. Inspired by Informedia, the VisionGo video search engine [16] is primarily based on targeted search. It combines a single list RSVP technique with several relevance feedback methods to create new lists of results from the set of selected results. The MediaMill CrossBrowser[26] also falls under this category. It is optimized to browse results from a query in a rapid fashion together with the time-line of a video.

All systems which combine multiple forms of similarity into the browse representation in order to allow the user interact with the dataset fall into the second category. For example the system in [20] combines both the time-line of videos with content-based similarity in one view. The Informedia storyboard-based interface[6][5] was one of the first search systems which combined multiple modalities into one interface. The FxPal MediaMagic[1] uses an innovative storyboard based interface for displaying results, and allows selected results as starting points for “find similar” queries. The uBase browser[12] combines even more forms of similarity based browsing. It combines hierarchical, temporal and lateral browsing together. The NN^k browser used for lateral browsing shows the user a graph network of related results. The MediaMill RotorBrowser[7] is also an exploratory browser. This browser is designed for topics that require a combination of query methods. It allows to integrate query results with time, visual similarity, semantic similarity and various other shot based similarity metrics.

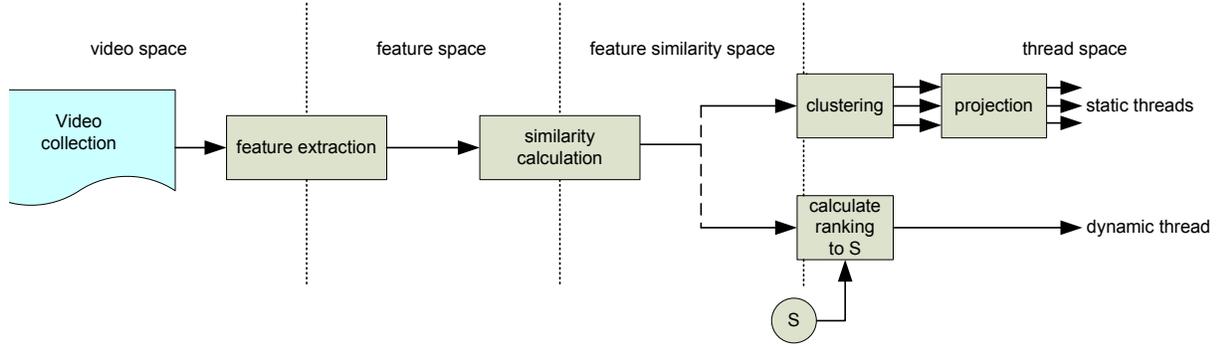


Figure 1: Overview of thread generation from a video collection. Feature extraction techniques are applied to every shot in every video, which yields the feature space. Subsequently, every shot is compared to every other shot using a similarity function appropriate to the features used. We generate two types of threads from this feature similarity space: static threads and dynamic threads. Static threads are calculated by 1) clustering the space into groups of related shots, and 2) creating a thread by projecting the cluster to a ranked list of results. Dynamic threads are calculated on the fly based on a given query shot S , which ranks the feature similarity space according to the distance to S .

Combinations of both categories are also possible, and most systems do not explicitly fall into just one category. To give just one example, the FXPAL collaborative search system [2] combines their MediaMagic system for query exploration with an RSVP like single list browsing technique into a multi user collaborative search system where each user fulfills one of both tasks.

The above list of systems is by no means complete, but should give an indication of the recent advances in the field of visualization methods for video search.

1.3 Contribution

To leverage the benefits of having multiple query methods without slowing down the interface we elaborate on the notion of threads [7]. Using these threads, we introduce a browser which combines techniques from targeted search browsers and exploratory search browsers into a single browsing environment: the ForkBrowser. We demonstrate that this novel browser balances effectiveness of results with high efficiency. In section 2 we introduce thread-based visualization framework on which the ForkBrowser is founded. In section 3 we present the experimental setup in which we compare the ForkBrowser against the CrossBrowser. We highlight results in section 5.

2. THREAD-BASED VISUALIZATION

To structure the definition of methods for browsing through a video collection we use the notion of threads [7]. Threads are defined as follows:

- A **thread** is a linked sequence of shots in a specified order, based upon an aspect of their content.

Threads are in essence ranked lists of shots, based on a specific feature similarity space, which is in itself based on a specific video query method. Let's look at *query-by-example*: this query method extracts low-level features, such as Wiccest features [9], from an example image, and compares these with precomputed features from the collection. These pre-computed features define the *feature space* of the collection.

Instead of providing an example image, we define a *similarity space* for this collection by defining a similarity function which compares every shot to every other shot. From this, we derive two types of threads based on the search need: static threads and dynamic threads.

Static threads help in creating a “web” of related shots through the collection beforehand. These are not changed during search, and help the user with locating himself within the collection. They are created by clustering the similarity space into k clusters. Subsequently, these clusters are projected into one dimensional lists where individual shots next to each other are also close to each other in similarity space. In a static thread there is no overall ranking, only a measure of relatedness between shots. So, the first shot in a static thread is not the most important one.

Dynamic threads are obtained on the fly based on any user provided *query shot* which is already in the collection itself. The rest of the collection is then re-ranked with the query shot as the first result. Compare a dynamic thread with the generated ranking of a query method.

An overview of the thread generation process is visualized in figure 1. There is a one to one relationship between thread types and query methods. More specifically, every thread type is based on a query method. We define the following thread types.

- **query result thread**: thread based on the results of a user constructed query.

In a query result thread the shots are linked because they all originate as results from the same user specified query. This thread is used as the starting point for navigation.

- **visual thread**: thread based on visual similarity.

The visual thread links shots together which share the same visual characteristics, so that shots next to each other are also visually similar.

- **semantic thread**: thread based on semantic similarity.

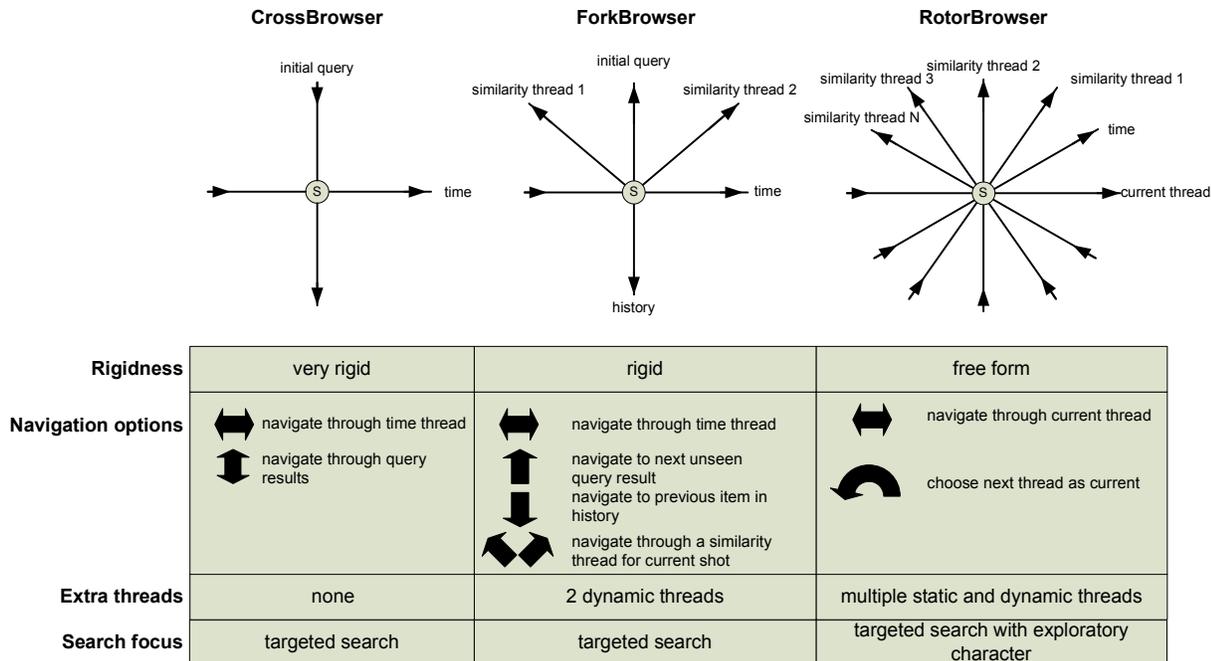


Figure 2: Overview of thread-based browsers and their navigation options. The rigidity indicates the dependence on initial query results. The navigation options indicate possible browsing directions for the user. The CrossBrowser and ForkBrowser use a direct mapping between user interaction and the interface. The CrossBrowser displays results and time only. The RotorBrowser displays all possible threads, with the active thread mapped to user action and a special navigation command to switch threads by rotation. Though all three browsers focus on targeted search, the RotorBrowser is the only one which can leave the original search and browse through anything the user sees at that moment.

The semantic thread links shots together based on their detected semantic concept scores, so that shots containing the same set of detected semantic concepts are close to each other in the ranked list.

- **time thread:** thread based on the time-line.

This thread is always static, since the time-line is predetermined for a video.

- **textual thread:** thread based on textual similarity.

The textual thread links shots to each other which contain similar ASR text.

All of the above threads, with the exception of the time thread which is always static, are available as either pre-computed static or on-the-fly dynamic threads. When a thread is used as a dynamic thread the difference is that the ranking is defined by a given query shot.

Every thread consists of a ranking of the entire collection. This implies that every shot is present in every thread. This means that given any shot, the user is able to switch to another thread by selecting one that catches their interest. But if every shot is present in all threads, this creates a problem: which threads are relevant for the user?

Since any query method, on which threads are based, does not know whether it is providing valid results or not. It only knows the similarity between shots within a thread. And these similarities are based on the quality of the used features, which varies from thread type to thread type. Therefore these similarities are only valid within a thread, they

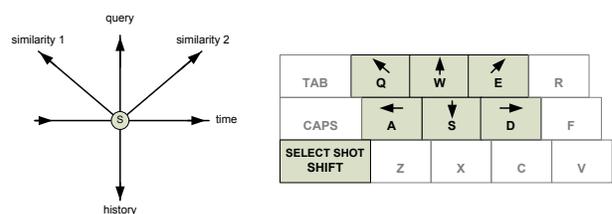


Figure 3: Mapping between the ForkBrowser interface and the keyboard. The entire interface is controlled by using the left hand only. This leaves the right hand free to deal with the mouse. The mapping allows for rapid interface control by the user.

cannot be used to compare one thread to another. It is therefore not possible to automatically discard bad threads based on these similarities alone. A human is however able to see in a glance whether results are satisfactory or not. Hence, the decision to use or reject threads should be mostly left to the user. The browser interface needs to support the user as much as possible in this role.

2.1 Visualizing threads

Any search engine typically has an interface for entering queries into the system, and an interface for retrieving results. In a text retrieval engine the query interface is often limited to text entry, with the interface for results placed

just below the query interface. The results are typically displayed in a long list. For video search engines a single interface screen is not sufficient. This is caused by the fact that video search engines typically require a much larger number of query methods, which all demand screen real estate. So the query interface and result interface are typically divided into multiple panes on the screen.

A search action for a multi pane search engine is defined as follows:

1. the user enters parameters into the query screen
2. the system retrieves results and displays them on a result screen
3. the user mentally adapts to the new interface
4. the user starts browsing through results
5. the user evaluates the results. If these are unsatisfactory, he goes back to step 1.

For our system we try to address a number of problems with this approach. We begin by defining the required design criteria which the visualization should adhere to. The primary goal of any search engine is to help the user to decide quickly if a retrieved result is relevant. This means that the interface should be fast and effective, both in response time and in visual design.

The interface should always show unexplored parts of the collection to the user, in the form of unvisited threads. If the interface shows not enough possible directions, there is a danger that the user has to restart her search from an earlier point each time a thread has no more valid results. To avoid this the system should provide multiple paths for finding results.

DESIGN CRITERION 1. *The interface should always indicate to the user where she can go.*

The opposite side of this criterion is showing too many directions. This overwhelms the user with choice. If an interface is too overwhelming the user requires more time to process all visual information before a decision can be made. The interface should therefore be as minimalistic as possible.

DESIGN CRITERION 2. *The interface should not overwhelm the user.*

A benefit of an interface which always allows more options is that it reduces the number of times a search has to be restarted. Since restarting means switching back to a query interface, a restart requires extra time from the user to mentally adapt to the different screens. This is to be avoided.

DESIGN CRITERION 3. *The interface should avoid switching between different interfaces.*

Also, the faster the user is able to select a relevant thread the faster he can choose to follow or discard it. A direct mapping between user action and interface response is therefore beneficial for browsing efficiency.

DESIGN CRITERION 4. *The interface must use a clear mapping between navigation and visualization.*

We use the shot as the elementary element in video, this implies that when a part of a shot is relevant to the query, the entire shot is relevant. Depending on the genre of video, such as broadcast news, or home video, the average length of a shot varies, and there is no preset maximum length of a shot. Therefore, the content at the beginning of a shot can be something different from the content at the end of that same shot. The system should be able to handle this.

DESIGN CRITERION 5. *The interface must be able to browse within a shot.*

Finally, video contains motion. By only looking at a keyframe this motion cannot be seen. This means that a shot cannot be represented by just one keyframe, so a requirement is that the system should be able to visualize motion within a shot.

DESIGN CRITERION 6. *The interface must support queries demanding explicit motion.*

Given these design requirements several thread-based browsers are possible. We define three types of browsers which vary in the decision of which threads to show and which threads to hide. See figure 2 for an overview of these three browsers. As introduced earlier, the CrossBrowser [26] primarily focuses on targeted search. It displays only initial query results and time. On the opposite side is the RotorBrowser [7], which uses exploratory search to find results. It displays as many threads as possible. We introduce a new browser which balances in-between the CrossBrowser and the RotorBrowser: the ForkBrowser.

2.2 The ForkBrowser

The ForkBrowser visualizes results by displaying the shots based on the shape of a fork, see figure 4. This shape was chosen to allow the user to view multiple threads at once. Using a grid based representation would allow more shots to be viewed from a single thread, but this limits the navigation into related threads. Also, by placing the current shot in the center of the visualization the interface forces the user to focus attention to the center of the screen. From here she can look into the various tines of the fork. These represent the different threads the user can visit. Any navigation action into one of the adjoining thread moves the first shot of that thread into the center of the display. The display will then update to show threads for that shot. This addresses design criterion 1. To address design criterion 2 the remainder of the screen is deliberately left empty. This helps the user to focus her attention only on the video.

The center tine shows unseen query results. In order to give the user the context of the shot in the center the leftmost and rightmost tines show the time-line. In order to allow the user to have more than one direction the two diagonal directions are used to show assignable similarity results. This addresses design criterion 3.

Finally, to help the user to know where she is, and to easily backtrack to earlier junctions, the stem of the fork displays browse history. This addresses design criterion 1 again. To ensure that quick navigation is possible all browse directions, each tine and the stem, are accessible by both the mouse and directly mapped keyboard shortcuts, see figure 3. This provides a direct mapping between the interface and the user, as stated in criterion 4.

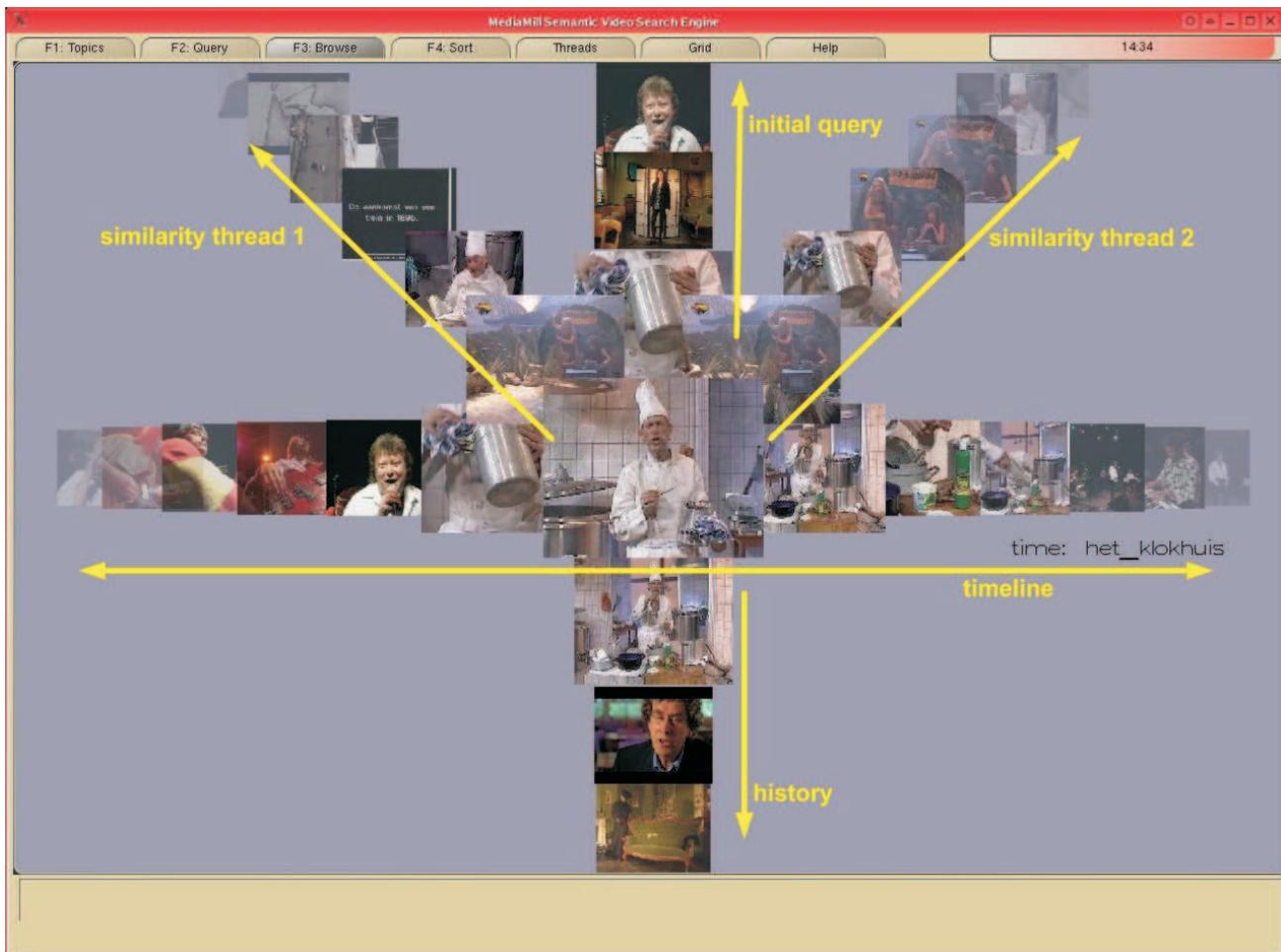


Figure 4: Thread-based visualization of the ForkBrowser. In this example from the TRECVID 2007 corpus, the top tine displays automated query-by-keyword results, the horizontal tines display the time-line of a “Klokhuis” episode, the diagonal tines display dynamic threads related to the center image, and finally the stem of the fork displays the current browse history.

To provide a solution for design criteria 5 and 6, shots are displayed as an animated sequence of up to 16 frames from the originating shot. This helps answering queries where the shot is relevant but this information was not shown in a single keyframe. This resolves design criterion 5. It also helps solving queries containing explicit motion, which was design criterion 6.

3. EXPERIMENTAL SETUP

In order to evaluate thread based visualization using the ForkBrowser we participated in the 2007 NIST TRECVID Interactive Search task[22]. This is an independent international benchmark for video retrieval systems. The goal of TRECVID is to promote progress in content-based retrieval from digital video material, by using open, metrics-based evaluation. Numerous universities, research institutes and companies participate in TRECVID, hence it can be considered the *de facto* standard for video retrieval evaluation. A general overview of successful approaches for TRECVID is given in [10]. TRECVID participation gives us a frame-

work for a comparison between the ForkBrowser and the CrossBrowser with respect to effectiveness and efficiency. It also gives us an indication how the ForkBrowser measures up to the current state of the art in video retrieval.

3.1 Data set

The video collection of the TRECVID 2007 benchmark consists of 100 hours of video material, provided by the Netherlands Institute for Sound and Vision. The collection contains videos of news magazines, science news, news reports, documentaries, educational programming, and archival video. Shot boundary information for the collection is provided by [19], automated speech recognition on these videos in Dutch is provided by [13], and a machine translation of the Dutch text to English is available.

TRECVID splits the collection into two parts. 50 hours are available for training and development of the system. The other 50 hours is used for evaluation. The contents of the evaluation set is not known beforehand, but is comparable to the first collection in terms of content.

During the benchmark itself the user receives a question

Experiment 1: Comparing the effectiveness of the CrossBrowser and the ForkBrowser

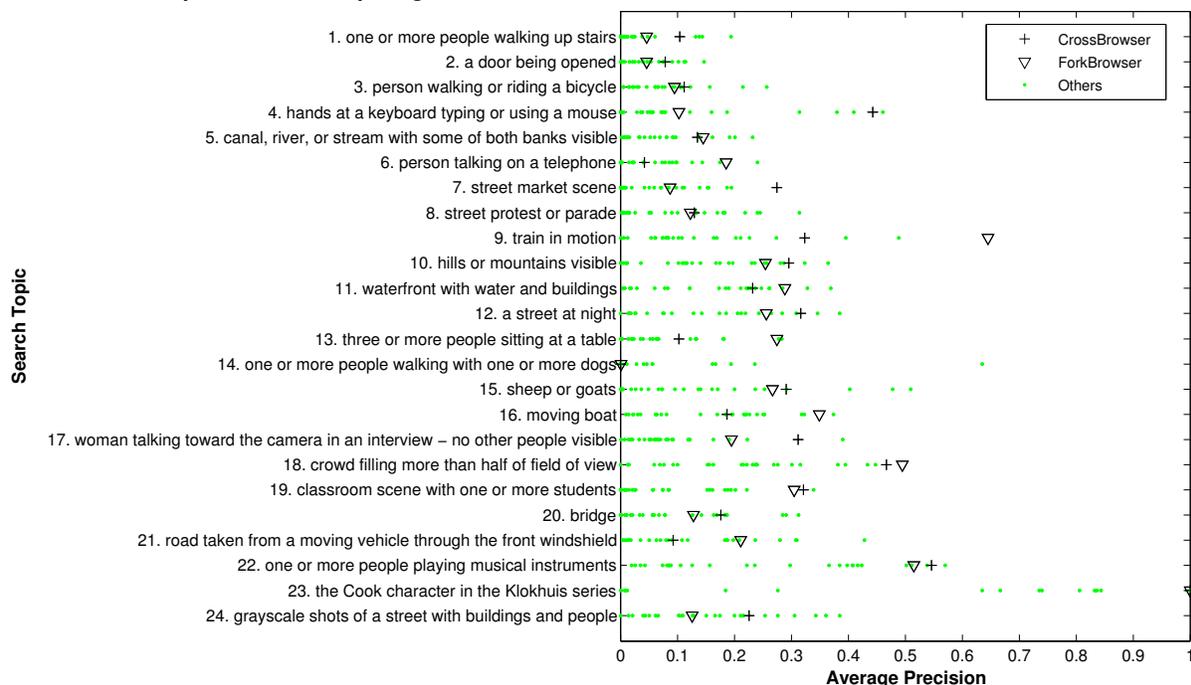


Figure 5: Comparison of effectiveness per topic of the CrossBrowser and ForkBrowser on the TRECVID 2007 interactive retrieval benchmark. The green dots indicate browsers of the 28 users of 10 other systems which used the same evaluation method, i.e. one user, one topic at a time.

in English, together with a couple of example images and videos. He then uses the system to come up with as many relevant shots on the test set as possible within a 15 minute timespan. The resulting shots are evaluated and judged by NIST. There are in total 24 topics to complete. Typical topics of 2007 include “Find shots of people walking or riding a bicycle.”, “Find shots of a train in motion.” and “Find shots of a woman talking toward the camera in an interview - no other people visible.”.

3.2 Implementation

We have implemented both automatic and manual querying possibilities. For every new topic the system automatically suggests 1000 results based on the ASR text [14]. The searcher is free to use the automatically generated results or not.

In order to supplement the automatic results, the following querying systems were also available for searchers.

- **query by keyword:** keyword search through the ASR results of the TRECVID 2007 dataset. Both Dutch and English search engines were available.
- **query by example:** NIST provided several example images for each topic. The searcher is able to use these as a starting point for search through precomputed Wiccest [9] and Gabor [3] features.
- **query by concept:** concept selection from a lexicon of 572 concepts. These were trained beforehand using the 50 hours of development data and a labelled corpus of example images.

The CrossBrowser used the above querying systems in combination with the time-line of each video. The ForkBrowser has two extra similarity threads. We chose to use visual similarity based on Wiccest features [9] for the first thread. The second thread uses visual similarity based on Gabor features [3].

During the search the interface times and logs all user actions. When the user submits the final search result, the resulting list is automatically appended to up to 1000 results based on a user selected query.

3.3 Experiments and evaluation

We want to evaluate both effectiveness and efficiency of the browsers. However, since any interactive browser needs to have a user controlling it, we cannot measure the performance of both browsers directly. The benchmark in essence compares the two expert users, one using the CrossBrowser and the other the ForkBrowser, on their performance during the TRECVID 2007 interactive search task. For this reason we have to be cautious in the conclusions we can draw from this experiment. We define the following two experiments:

- **Experiment 1: Comparing the effectiveness of the CrossBrowser and the ForkBrowser** This experiment compares how both browsers perform, based on their Average Precision scores for individual topics. Average Precision is a one-valued, bounded, metric which combines both precision and recall. It favors good results at the beginning of the list. We also measure the Mean Average Precision, which is the average precision scores averaged over all topics.

- **Experiment 2: Comparing the efficiency of the CrossBrowser and the ForkBrowser** We measure how both users interact with their browser during the 15 minute timeframe. This allows us to see if there is a significant difference in browser usage, and whether this varies between topics or sets an overall trend. For this we compare the evaluation results as provided by NIST with our user logs, which are automatically parsed to provide browse statistics. These statistics include time spent adjusting a query, time spent searching for results, number user interaction steps made during searching, number of selected results, the thread from which results were selected, etc.

4. RESULTS

4.1 Experiment 1: Comparing the effectiveness of the CrossBrowser and the ForkBrowser

The results for this experiment are plotted in figure 5. When we look at the mean average precision, we see that both browsers achieve nearly the same score, 0.259 for the CrossBrowser and 0.256 for the ForkBrowser, though average precision scores for individual topics vary greatly.

For example: the CrossBrowser performs significantly better than the ForkBrowser for topics *4: hands at a keyboard typing* and *7: street market scene*. The ForkBrowser performs significantly better for topic *9: train in motion* and *16: moving boat* which both contain explicit motion, and *13: people sitting at a table*. This indicates that the browser effectiveness is dependent on the topic. Note again that when looking at individual results the expert user performing the task influences the results greatly. Therefore though it is interesting to see that the ForkBrowser and CrossBrowser yield nearly equal Mean Average Precision statistics, a larger scale user study is required to determine which browser performs better.

Both browsers achieve a 1.0 average precision score for topic *23: Cook in Klokhuis*, as did a lot of other systems. This is because there was only a very low number of positive results in the collection, which were all found.

To analyze whether thread usage is independent of topic types we have measured the thread usage per topic for the ForkBrowser. See figure 7. The graph indicates that each topic required a different combination of threads in order to find good results. For example topics *18: crowd* and *10: mountain* used the initial query thread more than others. This was because there was a direct concept mapping available for these topics. For topics *13: people sitting at a table* and *17: woman talking toward the camera* the visual similarity threads were really helpful. During the search these threads allowed a great number of similar shots to be selected. For most topics the time thread was very important. We explain the success of the time thread as follows. When a shot is relevant, the shots immediately before and immediately after are often also relevant. The user sees this and selects these items. This accounts for the large portion of results obtained from the time-line. The results from this graph strengthen the hypothesis that we cannot beforehand merge the different threads into one, since the optimal thread is not known beforehand.

4.2 Experiment 2: Comparing the efficiency of the CrossBrowser and the ForkBrowser

We plot results for experiment 2 in figure 6. Results indicate that the ForkBrowser required significantly less user interaction steps for all topics within the same 15 minute timeframe. Also, the high performance on topic *7: street market scene* for the CrossBrowser can be explained. The expert user went through a great number of results using more than 6000 interaction steps. Also interesting is that though both browsers were able to achieve an 1.0 score on topic *23: Cook in Klokhuis*, the CrossBrowser user used more than double the amount of user interactions.

5. CONCLUSIONS

In this paper, we combined aspects of exploratory browsing and targeted browsing into an effective thread-based video search system: the ForkBrowser.

For this we have made a distinction between exploratory search systems and targeted search systems. Targeted search focuses on fast retrieval based on an initial set of results from a query system. Exploratory search focuses on exploration of the video collection by combining multiple query methods into one visualization, to allow the user to browse to related shots. It is less dependent on an initial set of results.

To leverage the benefit of exploratory search without slowing down the interface we used the notion of threads. This allows for a direct mapping between query methods and threads. Using these threads, we have introduced the ForkBrowser. A browser which combines techniques from targeted search browsers and exploratory search browsers into a single browsing environment.

We investigate effectiveness and efficiency of the ForkBrowser with two experiments in the NIST TRECVID 2007 Interactive search task. In these experiments we compare the ForkBrowser with the CrossBrowser, a browser based primarily on targeted search. The TRECVID interactive search task consists of finding relevant shots for a given topic within a 15 minute time span from a collection of 50 hours of videos of news magazines, science news, news reports, documentaries, educational programming, and archival video.

On average, both browsers were equally effective in this task. However, if we look at individual topic results, we see that these vary greatly. This indicates that both browsers have different strengths depending on the topic. A key insight is that ForkBrowser requires significantly less user interaction steps to achieve this same result. Results also indicate that different combinations of threads are required for different types of topics. Furthermore, both browsers rank among the best interactive retrieval systems currently available.

6. REFERENCES

- [1] J. Adcock, M. Cooper, and F. Chen. Fxpal mediamagic video search system. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 644–644, New York, NY, USA, 2007. ACM.
- [2] J. Adcock, J. Pickens, M. Cooper, L. Anthony, F. Chen, and P. Qvarfordt. Fxpal interactive search experiments for trecvid 2007. In *Proceedings of the NIST TRECVID workshop 2007*.

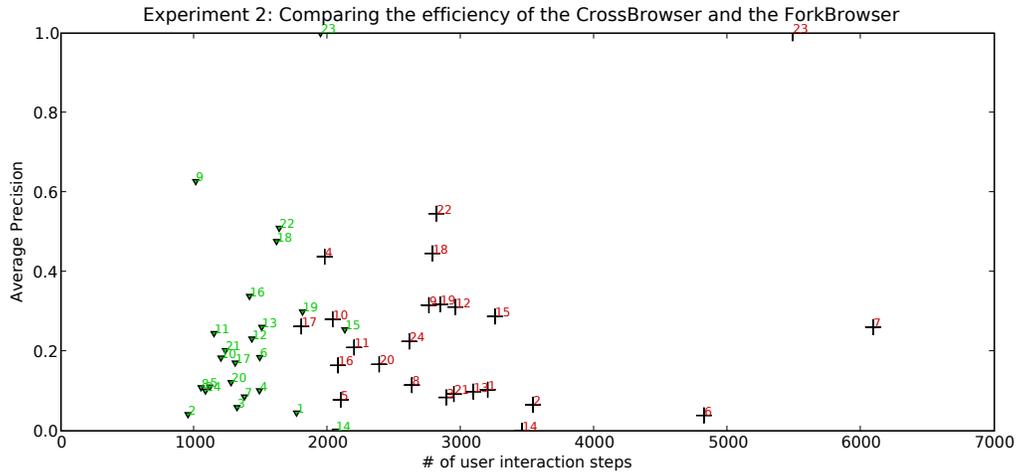


Figure 6: Average Precision for each topic compared to the number of user interaction steps. The interaction steps include keyboard presses and mouse clicks for both browsers. Note that the user using the ForkBrowser (▽) used significantly less interactions than the user of the CrossBrowser (+).

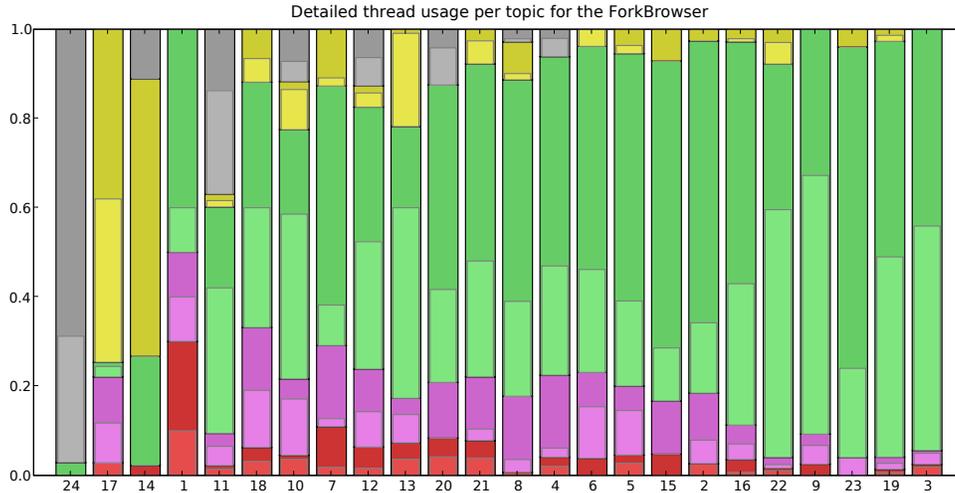


Figure 7: Normalized thread usage per topic (indicated by stacked bar). Each color denotes a different type of thread. Green represents time, purple represents the initial query, yellow represents visually similarity, red represents history and gray represents other. The lighter colored inlay represents the % of shots that was judged correct by NIST for that thread. Note that the overall trend in thread usage is browsing through 1) time, 2) initial query method followed by 3) visual similarity. Note however that different topics use different thread combinations for optimal performance.

[3] A. Bovik, M. Clark, and W. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, 1990.

[4] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. *Third International Conference on Visual Information Systems*, pages 509–516, 1999.

[5] M. G. Christel and N. Moraveji. Finding the right shots: assessing usability and performance of a digital video library interface. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 732–739, New York, NY, USA, 2004. ACM.

[6] M. G. Christel and R. Yan. Merging storyboard strategies and automatic retrieval for improving interactive video search. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 486–493, New York, NY, USA, 2007. ACM.

- [7] O. de Rooij, C. G. M. Snoek, and M. Worring. Query on demand video browsing. In *Proceedings of the ACM International Conference on Multimedia*, pages 811–814, Augsburg, Germany, September 2007.
- [8] M. S. Flickner, H. Niblack, W. Ashley, J. Q. H. Dom, B. Gorkani, M. Hafner, J. Lee, D. Petkovic, D. Steele, D. Yanker, et al. Query by image and video content: the QBIC system. *Computer*, 28(9):23–32, 1995.
- [9] J. M. Geusebroek. Compact object descriptors from local colour invariant histograms. In *British Machine Vision Conference*, volume 3, pages 1029–1038, 2006.
- [10] A. G. Hauptmann and M. G. Christel. Successful approaches in the trec video retrieval evaluations. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 668–675, New York, NY, USA, 2004. ACM.
- [11] 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 *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 385–394, New York, NY, USA, 2006. ACM Press.
- [12] D. Heesch, A. Yavlinsky, and S. Rüger. Nnk networks and automated annotation for browsing large image collections from the world wide web. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 493–494, New York, NY, USA, 2006. ACM.
- [13] M. Huijbregts, R. Ordelman, and F. de Jong. Annotation of heterogeneous multimedia content using automatic speech recognition. In *Proceedings of the international conference on Semantics And digital Media Technologies*, LNCS, Berlin, 2007. Springer Verlag.
- [14] B. Huurnink and M. de Rijke. Exploiting redundancy in cross-channel video retrieval. In *Proceedings of the 9th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR 2007)*, pages 177–186. ACM Press, September 2007.
- [15] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, 2006.
- [16] H.-B. Luan, S.-Y. Neo, H.-K. Goh, Y.-D. Zhang, S.-X. Lin, and T.-S. Chua. Segregated feedback with performance-based adaptive sampling for interactive news video retrieval. In *Proceedings of the 15th international conference on Multimedia*, pages 293–296, New York, NY, USA, 2007. ACM.
- [17] M. Naphade, J. R. Smith, J. Tešić, 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.
- [18] A. P. Natsev, M. R. Naphade, and J. Tešić. Learning the semantics of multimedia queries and concepts from a small number of examples. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 598–607, New York, NY, USA, 2005. ACM.
- [19] C. Petersohn. Fraunhofer HHI at TRECVID 2004: Shot boundary detection system. In *Proceedings of the TRECVID Workshop*, NIST Special Publication, Gaithersburg, USA, 2004.
- [20] M. Rautianen, T. Ojala, and T. Seppänen. Cluster-temporal browsing of large news video databases. In *IEEE ICME*, pages 2:751–754, Taipei, Taiwan, 2004.
- [21] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003.
- [22] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.
- [23] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content based image retrieval at the end of the early years. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1349–1380, December 2000.
- [24] J. R. Smith and S. F. Chang. VisualSEEK: a fully automated content-based image query system. *Proceedings of the fourth ACM international conference on Multimedia*, pages 87–98, 1997.
- [25] C. G. M. Snoek, M. Worring, D. C. Koelma, and A. W. M. Smeulders. A learned lexicon-driven paradigm for interactive video retrieval. *IEEE Transactions on Multimedia*, 9(2):280–292, 2007.
- [26] C. G. M. Snoek, M. Worring, D. C. Koelma, and A. W. M. Smeulders. A learned lexicon-driven paradigm for interactive video retrieval. *IEEE Transactions on Multimedia*, 9(2):280–292, February 2007.
- [27] D. Wang, X. Liu, L. Luo, J. Li, and B. Zhang. Video diver: generic video indexing with diverse features. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 61–70, New York, NY, USA, 2007. ACM.