Efficient Targeted Search Using a Focus and Context Video Browser

ORK DE ROOIJ and MARCEL WORRING, University of Amsterdam

Currently there are several interactive content-based video retrieval techniques and systems available. However, retrieval performance depends heavily on the means of interaction. We argue that effective CBVR requires efficient, specialized user interfaces. In this article we propose guidelines for such an interface, and we propose an effective CBVR engine: the ForkBrowser, which builds upon the principle of focus and context. This browser is evaluated using a combination of user simulation and real user evaluation. Results indicate that the ideas have merit, and that the browser performs very well when compared to the state-of-the-art in video retrieval.

Categories and Subject Descriptors: H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Evaluation / methodology*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Multidimensional browsing, conceptual similarity, semantic threads, interactive search, information visualization

ACM Reference Format:

de Rooij, O. and Worring, M. 2012. Efficient targeted search using a facus and context video browser. ACM Trans. Multimedia Comput. Commun. Appl. 8, 4, Article 51 (November 2012), 19 pages.
DOI = 10.1145/2379790.2379793 http://doi.acm.org/10.1145/2379790.2379793

1. INTRODUCTION

The sizes of various video collections both online and offline are increasing rapidly. Also, the available techniques for analyzing and indexing such collections are extended every day. There is more to search through, and there are more techniques to support the search process.

In video retrieval, we distinguish two types of search based on the reason for searching. The first type is *exploratory search* where the goal is to get insight in the collection or browsing for unexpected results. The second type is *targeted search* where the goal is to find multiple relevant items for a clearly defined search need. This article focuses on making the latter as efficient as possible, though, in practice, both methods are often intermixed within one system. Targeted video search can be split into two separate stages. First, in the query stage the users employ a combination of access techniques that then yield a set of results. Second, in the browse stage users go through these results to select relevant items.

Author's addresses: O. de Rooij (corresponding author) and M. Worring, Intelligent Systems Lab Amsterdam, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam; email: orooij@uva.nl.

DOI 10.1145/2379790.2379793 http://doi.acm.org/10.1145/2379790.2379793

In early research systems, search through video collections was done based on filenames, speech transcripts, or existing user annotation. However, these do not necessarily reflect the actual visual content. Video content analysis techniques [Lew et al. 2006; Snoek et al. 2007; Natsev et al. 2005; Wang et al. 2007] resolve this by allowing users to query the video content itself using specialized input queries. Most algorithms then yield a list of videos that the system deems relevant. It is then up to the user to browse through this list to select relevant items. With many techniques available, and many ways to specify a query, users need means of selecting the right technique with the right parameters for a specific search need. Automatic selection systems do exist; see Natsev et al. [2007] for an overview. However, controlled benchmarks such as TRECVID [Smeaton et al. 2006] have shown that, compared to text search systems such as Google, accuracy is still quite limited.

Due to the limited accuracy, interaction is needed for almost all realistic video search tasks. Due to the limited accuracy in almost all realistic video search tasks, an interactive browse stage is needed after posing a query. Here, users have to determine whether the current query results are the best results obtainable for each individual technique, given the user search need. The relatively low performance of these individual techniques makes text domain tools, such as faceted search [Yee et al. 2003] where users iteratively narrow down search parameters in order to find the perfect result, not directly applicable. The relevant results from individual techniques disappear entirely when combined and filtered in such a way, long before the total list of results is narrowed down to a manageable level. New interaction techniques are therefore needed.

In literature, and at benchmarks like TRECVID, we see a variety of systems, each with unique approaches to interactive search. Some of these interfaces are specifically geared toward finding relevant results as fast as possible. For example, Hauptmann et al. [2006] use rapid serial visual presentation to allow categorization of automatic results organized in batches up to nine images. The VisionGo [Luan et al. 2007] system combines this with relevance feedback optimizing the stream of results the user is seeing. In both systems the role of the user is limited to visually understanding images and selecting correct results. Other systems allow for a more exploratory approach, and let the user decide what to find or select. For example, Christel and Yan [2007] and Zavesky et al. [2008] employ a grid-based structured visualization of results, with the ordering based on visual or storyboard-based and characteristics. IBM Marvel [Yan et al. 2007] allows users to find results for multiple queries at once allowing tagging of results with multiple labels. Whether a system is geared to finding specific results or to explore a collection, in both cases we see a tendency that interfaces which allow for fast and visually extensive user interaction tend to provide better results.

Typically, the browse stage first requires users to assess the current element, and the possibilities for navigation. To enhance assessment of both the current element and the navigation, we should leverage the remarkable capability of humans to quickly perceive, understand, and make decisions about shown image content. Similar issues have been thoroughly analyzed in the information visualization literature [Ware 2000], where focus and context techniques are proposed as an effective solution. In our case, the *focus* is provided by the current element, which can be a single shot, or a series of shots. The *context* is provided by the navigation options from this current element, that is, related shots. We propose to make targeted search more efficient by developing a focus and context technique for video retrieval.

We start of from our MediaMill CrossBrowser [Snoek et al. 2007] which has been shown to be good at targeted search when a good starting query is available, but is less effective at exploratory search. The RotorBrowser [de Rooij and Worring 2010] allows the user to explore a collection using many degrees of freedom, and is very usable for exploratory search in which users never backtrack to previous results and have no fixed target. However, this also requires the user to pay attention to more navigation directions, which slows down the speed in which users can navigate. The RotorBrowser is therefore less suited for targeted search.

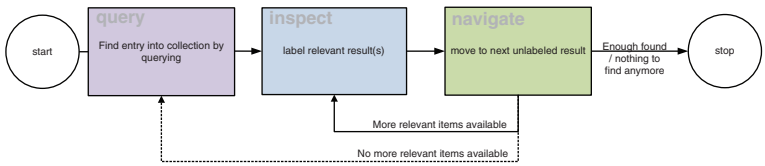

Fig. 1. Basic workflow for targeted video search. First, in the query phase users pose a query, which yields a list of results. In the browse phase users then start by inspecting the current set of results. This can be a single focal shot, or a set of shots, and mark these as relevant. Users then navigate to a new set of results by using any of the available navigation options, and the cycle repeats. More detailed implementations of the latter two phases will be presented in Figure 5.

Based on these insights and a careful analysis of the aforementioned systems we have developed a set of design rules that we will describe in Section 2 leading to our ForkBrowser to be described in Section 3. An evaluation of parameters for the browser is given in Section 4, followed by an evaluation of effectiveness using simulations and at TRECVID.

2. TARGETED VIDEO SEARCH

The goal of any targeted search system is to allow users to find results for a specific search need in the most efficient manner possible. In the query stage, users need to indicate their search need by configuring various query techniques, each yielding a list of results. The browse stage which follows is composed of two distinct actions. The user looks at some set of results and acts upon it, then the user choose what to view next. After this there will be a new set of results, and the cycle repeats until the user decides to stop the search process. See Figure 1 for an overview. In this section, we look at several existing systems based on these two actions, and extract existing common-sense guidelines from these systems. We will then use the combination of guidelines to design a browser for targeted video search.

Let us first define targeted search within a video collection more precisely.

Definition 1 (*Targeted Video Search*). Targeted video search is the act of finding multiple fragments of video within a collection which answer a specific search need.

What this specific search need is depends on the user. This can vary between search for specific people or objects, or more generic like search for indoor or outdoor scenes, scenes containing people, animals, vehicles, or locations.

2.1 Structuring the Collection

Targeted video search relies on a combination of queries and on sebsequent navigation of the results as well as exploring new parts of the collection. To unify these into one framework we follow de Rooij and Worring [2010], that uses the notion of threads to define structure as follows.

Definition 2. A *thread* is a linked sequence of shots in a specified order, based upon an aspect of their content.

A thread is in essence a ranking of (a part of) the collection, based on a specific feature similarity space. As such, the output of any query interface in any search interface, which can have various names in various systems but is often represented by a list of shots, can be called a thread.

We distinguish two types of threads based on the extra input required to generate them. The first type is the static thread, defined as follows.

Definition 3. A *static thread* links shots in the collection without requiring extra input.

Static threads place a precomputed structure on top of a video collection. Each static thread can be seen as a list of shots in which the order is important. An example of a static thread is the timeline

of a video: the list of shots starting with the first shot of a video and ending with the last shot of that video.

The second type is the dynamic thread, that is defined on-the-fly based on some form of input. As an example, the resulting ranked list of shots from a query would typically be a dynamic thread.

The type of input required for a dynamic thread varies from a single shot of input, to a set of shots. In order to capture this difference we extend upon de Rooij and Worring [2010] and split dynamic threads into two definitions.

Definition 4. A *single shot dynamic thread* links shots in the collection based on a single shot as input.

Definition 5. A *multishot dynamic thread* links shots in the collection based on a series of shots as input.

Together these thread types define navigation options through a video collection. In order to actually navigate a collection we also need a position within the collection where users navigate from. For this we define the focus as being the following.

Definition 6. The *focal shot* s_f identifies the current position in the video collection.

Since each shot may be contained in various static threads, or can be the starting point for different dynamic threads, there will be a series of threads containing s_f. These threads span the context of s_f. In an interface typically only a part of this context will be visible, yielding the navigation context, defined as follwos.

Definition 7. The *navigation context* for s_f is the visible part of the set of threads which contain s_f.

As a typical example: consider the user looking for the word “bicycle” in a large collection with a system that allows text search and visual similarity matching. This word is present in speech recognition texts for several shots, some more than others, and a dynamic thread linking these shots would be generated, with the first shot in the thread being the one with the word “bicycle” being present most times. In an interface this thread would then be shown (in a list, a grid, or by some other means depending on the system) and the first shot would be the focus shot. The system would then use similarity matching on this focus shot to create another dynamic thread with, hopefully, more images of bicycles. Lastly, the first shot was also part of some video, and therefore part of a timeline, which is a static thread. All three threads could be displayed in the interface as the navigation context for the user. Users can then choose a new focus shot by selecting any shot from the navigation context. Subsequently, this action updates the navigation context, in which they can now find new results.

Using threads, we now look more closely at the inspection and navigation phases of Figure 1.

2.2 Inspection Phase

During targeted search users navigate through a collection, searching for specific items. When these are found, they mark, label, or otherwise select them. For the purpose of this article we shall use the shot as the unit of retrieval in video search, though other units, such as individual frames or entire videos, are also possible.

During the inspection phase users have to look at a shot or a sequence of shots to determine if it warrants any action. Many interactive video search engines depict a single keyframe from the shot, and let users base their actions on that alone. However, the content at the beginning of a shot can be quite different from the content at the end of that same shot. Also, video contains motion that cannot be seen by only looking at a keyframe. This means that a shot cannot be represented by just a single image.

Besides the need to display more than the keyframe, the information content and visual complexity of a shot vary [Sundaram and Chang 2001]. Also, certain search needs will require more attention to detail from the users than others. Intuitively, when a search need is simple, for example, *find shots of grass*, not much detail is needed to be able to see that a shot is relevant, and mark it as such, and many shots can be marked at once. When a search need is more complex, such as *find shots of a person walking up the stairs* users require more detail to accurately determine whether to mark shots.

The preceding observations lead to the following guidelines.

Guideline 1. The user has to be able to efficiently inspect the setting, scene, and all objects and their motions within a shot, with the level detailed adapted to the complexity.

The interpretation of a single shot can change when its surroundings are changed, the so-called Kuleshov effect. To ensure that users mark shots correctly they have to be able to access the timeline, which is indeed shown to be important for video retrieval [Rautiainen et al. 2006; Adcock et al. 2005; Christel et al. 2004; Snoek et al. 2007].

Guideline 2. The user has to be able to view the temporal context of video shots in order to determine the relevance of the current shot.

During the inspection phase users have to mark or label shots for relevance. This is often done by clicking on a shot, or by pressing a key. Depending on the search task the number of available relevant shots within the collection varies, and the local shot relevance density varies with it. Furthermore, as stated in Guideline 1 the level of detail varies. When limited detail is needed more shots can be visually inspected at once. The user therefore has to be able to choose between batch-labeling many results at once or mark individual results one by one. Therefore, we have the next guideline.

Guideline 3. The user should be able to label multiple shots at the same time, when this is more efficient.

2.3 Navigation Phase

When the current selection of shots has been mentally processed by the user, the user has the option to navigate to different results. In our thread model this implies selecting any thread, and getting results from there. The navigation context here, as given in Definition 7, yields the basis for further navigation from this set.

During each interaction step users mark additional relevant shots from those visible. The shots that are left unmarked are therefore irrelevant to the user search need, barring accidental misjudgments. To continue the retrieval process, users need to be able to navigate to shots that have not been shown before, therefore we have the following guideline.

Guideline 4. The system should show unexplored areas within the collection.

In a thread context this leads to threads containing the focal shot, but otherwise mostly unvisited shots, having priority over threads which contain high numbers of previously visited shots. Typical threads which yield unexplored areas are the time thread, since when there is one relevant shot in a video there is a high chance that there are more nearby in the timeline, and the visual similarity threads, which yield a set of visually similar results based on the focal shot, likely relevant to the search need. Even when no relevant shots are present in the interface, users should still have a clear idea of what to do next. The interface should therefore always present a viable default navigation option to the users, even when no relevant shots are present in the current navigation context.

Guideline 5. The system should always yield a default navigation path.

This covers where to navigate, but should a user browse through more retrieved results of the current search query, or should he browse through related shots based on the current focus shot? Such a decision needs to be made on the spot, and without too much repercussion when the wrong choice has been made. This leads to following.

Guideline 6. The user has to be able to efficiently inspect the potential relevance of the navigation options in the navigation context.

But, mistakes can and will be made, which should have limited impact on search performance. Furthermore, users might decide later on that they would have liked to follow a different retrieval path altogether. So, users should be able to go back to an earlier situation and choose again. To that end, we use the observation that people are known to be good at visually [Ware 2000] or spatially [Robertson et al. 1998] remembering earlier interface states, so this backtracking step should be made as visual as possible.

Guideline 7. The system should aid the user to return to earlier navigation options by visual or spatial means.

During this backtracking or any other navigation, the interface should remain consistent during use, and not switch through various panes. Even though the focus sometimes stays the same when this happens, this would remove or at least fragment the navigation context. This requires the users to reexamine the interface and all visual components before they can continue, which takes a lot of time. Therefore, we have the following.

Guideline 8. The interface should not switch between different panes.

Within this single interface pane we can further optimize the time between a user decision on a visible shot and the response of the interface. This should allow users to make the next decision sooner. An indirect means of interaction, such as using a mouse for controlling the navigation, yields extra interface actions for users: they need to move the pointer to the correct shot, and then click on it. If we eliminate this, users can respond faster. For example, the usage of efficiently placed keyboard shortcuts would already allow experienced users to navigate through the collection without having to resort to mouse movements, and browse and select more items in less time. A more direct mapping between user action and interface response is therefore beneficial for browsing efficiency.

Guideline 9. The interface should use a clear mapping between navigation and visualization.

We have based the preceding nine guidelines on related work, which is summarized in Table II together with their defining characteristics for the query, inspection, and navigation phases. The implementation of individual guidelines varies between systems, and we have summarized the guidelines and their implementations in various related works in Figure 3.

These nine guidelines both help with analyzing the current result, and in determining where to navigate next. As such, they are similar to the focus and context techniques. The guidelines yield focus- and context-based browsing through a collection, where users focus on an active result, and the context around it is determined by both user actions and the collection itself, and optimized to show the most relevant results first.

3. TECHNIQUES FOR TARGETED VIDEO SEARCH

The guidelines in the previous section do not yet yield specific details. In this section we list a series of techniques for the individual elements that help build an efficient targeted search video browser. We then combine these techniques into a prototype video search interface: the ForkBrowser.

Reference	#	Query	Inspect	Navigate
VisionGo [Luan et al. 2007]	1	concepts or relevance feedback based on earlier results	single result list displayed in grid 3 items wide	browse through list of results, mark 3 at a time
Informedia [Christel and Yan 2007]	2	by text, concepts or example	storyboard grid with shots clustered by story	browse through results from multiple query methods
CuZero [Zavesky et al. 2008]	3	by concepts, using query permutation for related searches	(unobtrusive) retrieval of results, displayed in grid	allows instant-switch between depth-search and breadth-search
FXPal MediaMagic [Adcock et al. 2007]	4	by text, concepts or example	story based gridlike interface with visual cues	allows user to find similar based on text, visual content or concepts
Tagging and Browsing [Yan et al. 2007]	5	no single user query, but tags results for multiple queries at once	tag results for multiple queries at once	navigate by tagging or by browsing
Extreme Video Retrieval [Hauptmann et al. 2008]	6	no user query, started from automatic search results on topic	images are briefly shown on screen in increasing grid sizes	browse through single list of results extremely fast
UvA CrossBrowser [Snoek et al. 2007]	7	by text, concepts or example	cross shaped series of shots	browse through results and time
UvA RotorBrowser [de Rooij and Worring 2010]	8	by text, concepts or example	circle shaped series of shots with single focus shot in center	browse through results from single focus shot
UvA ForkBrowser	9	by text, concepts or example	fork shaped series of shots, with single focus shot in center	browse through results, time and similar from focus

Fig. 2. An overview of selected related work with their defining characteristics listed for each stage in the retrieval process.

	1 INSPECT The user has to be able to efficiently inspect the scene & objects and motions within a shot with the level detailed adapted to the complexity.	2 TEMPORAL CONTEXT The user has to be able to view the temporal context of video shots in order to determine the relevance of the current shot.	3 MULTIPLE SHOTS The user should be able to label multiple shots at the same time, when this is more efficient.	4 SHOW UNEXPLORED The system should show unexplored areas within the collection.	5 DEFAULT NAVIGATION The system should always yield a default navigation path.	6 INSPECT NAVIGATION OPTIONS The user has to be able to efficiently inspect the potential relevance of the navigation options in the navigation context.	7 RETURN TO PREVIOUS STATES The system should aid the user to return to earlier navigation options by visual or spatial means.	8 NO PANES The interface should not switch between different panes.	9 MAPPING BETWEEN NAV AND VIS The interface should use a clear mapping between navigation and visualization.
VisionGo [Luan et al. 2007]	✓ video preview		✓ up to three items	✓ relevance feedback mining	✓ down for more			✓ one pane	✓ down/up, one nav direction
Informedia [Christel and Yan 2007]	✓ video preview						✓ from capture panel		
CuZero [Zavesky et al. 2008]	✓ animated icon	✓ using player		✓ query-cell exploration	✓ browse pages of results	✓ click-free inspection of results		~ both panes visible at same time	✓ grid, but linked
FXPal MediaMagic [Adcock et al. 2007]		✓ timeline display	✓ batch select	✓ find similar					
Tagging and Browsing [Yan et al. 2007]			✓ multiple tags/queries		✓ system asks for annotations				
Extreme Video Retrieval [Hauptmann et al. 2008]			✓ first one, then multiple		✓ single direction			✓ one pane	✓ keyboard mapping
UvA CrossBrowser [Snoek et al. 2007]	✓ video preview	✓ timeline display			✓ query thread	✓ threads displayed for navigation	✓ backward in query thread	✓ one pane	✓ keyboard mapping
UvA RotorBrowser [de Rooij and Worring 2010]		✓ timeline display		✓ multiple threads from focus shot		✓ threads displayed for navigation		✓ one pane	✓ keyboard mapping
UvA ForkBrowser	✓ video preview	✓ timeline display	✓ grid-select single thread	✓ similarity threads + feedback mining	✓ query thread	✓ threads displayed for navigation	✓ history thread	✓ one pane	✓ keyboard mapping

Fig. 3. The nine guidelines and their implementations within selected related works (see Figure 2 for the index). For each guideline we follow the definition as given in Section 2 to determine whether systems follow that guideline or not.

3.1 Threads

The set of threads should be as diverse as possible to allow interesting navigation options to users (Guideline 6), and to show as much diversity of the collection to users as possible (Guideline 4). We specify the following types of threads.

—The *default navigation thread* is a multishot dynamic thread that initially represents a list of results obtained from a query interface used to seed the initial browser. This thread yields the means of entry into a collection, and allows the user to find initial results. Furthermore, it also acts as a fallback navigation option when no other relevant results are visible.

—*Time threads* are static threads containing all shots in their original sequence. This allows users to inspect a shot depending on context, and it allows users to browse through nearby relevant results.

—The *history thread* is a multishot dynamic thread that contains the list of previous focal shots. This allows the user to go back to earlier results.

—Multiple *visual similarity threads* are single-shot dynamic threads that show shots relevant to s_f based on visual similarity using Algorithm a. This yields a set of perceptually similar results based on the current focal shot.

To make inspecting a single shot efficient, as required by Guideline 1, we propose an interface displaying individual threads as a sequence of individual shots. These are as a default depicted as keyframes. On-demand, users are able to view in-place motion icons [Luan et al. 2007] when they need more detail. These show up to 16 frames of the entire shot at high speed. This provides a compromise between storage capacity and the video itself. Given typical shot lengths showing 16 frames yields more than enough information about objects present in the shot, and camera motions used, while the storage capacity needed to store 16 frames per shot for each shot for all videos is still manageable.

The focal shot is in the center and is the largest shot shown. This lets users focus on the focal shot first, while the related information from the rest of the thread is observed in the periphery. To allow users to get more detail when the search need requires this, the system allows users to zoom in on the focal shot. Zooming also determines the amount of context that can be shown in multiple threads. The number of shots shown in each thread is dependent on the zoom level, which influences the size of each image, and the available amount of screen real estate. Also, since users focus on the focal shot only shots shown within a limited cone of vision from this focal shot are accurately seen. The number of visible shots in each thread is therefore determined by the complexity of the search topic. During practical use we found that on average users are able to perceive six shots per thread.

This covers the length of threads, but not the amount. From any focus shot a number of relevant threads can be shown, based on the defined types given earlier. The number of displayed threads is dependent on both the interface and whether there are related threads for any focus shot. Intuitively, visualizing too many extra threads inhibits search efficiency, since users have to process much more information from the screen before a decision can be made. But too few threads leads to no benefit from unseen parts of the dataset. So, in practice we need to find a balance between these two conflicting criteria.

3.2 Direct Mapping

To make video retrieval as efficient as possible, we need an interface which is controllable as directly as possible (see Guideline 9). We therefore design the spatial layout of the interface around its navigation. This is a two-step process. First, we need to select an controller interface. Second, we need to adapt the visual interface to this configuration. Since this controller interface by itself should allow for fast interaction this removes devices such as the mouse and voice control, since these are either indirect and/or slow. Modern touch-screen interfaces would allow for fast and direct interaction with the graphical interface. However, when manipulating the visualization directly the hand will obscure the underlying visual information, requiring to remove the hand after each iteration.

We have chosen to use a traditional keyboard since this is inexpensive, well known, offers tactile feedback when buttons are pressed so the user does not have to look at the keyboard itself, and is fast to use. Next, we borrow from the field of modern computer games which evolved to use the so-called “W A S D” configuration for movement of player characters. These buttons have been chosen because they allow the character to move, while still having plenty of extra keyboard buttons nearby the left hand that directly control other features of the game. We take this same configuration, and place all displayed threads in directions such that they correspond to the directions on a keyboard. This provides a direct spatial mapping between depicted direction and keyboard button. See Figure 4. All other functions are located on buttons near the directional buttons, so that they are within reach of the users’ hand. The interface is designed to be controlled with one (left) hand only. This leaves the other hand free to click on relevant shots with the mouse. Our direct mapping leads to the layout depicted in Figure 4.

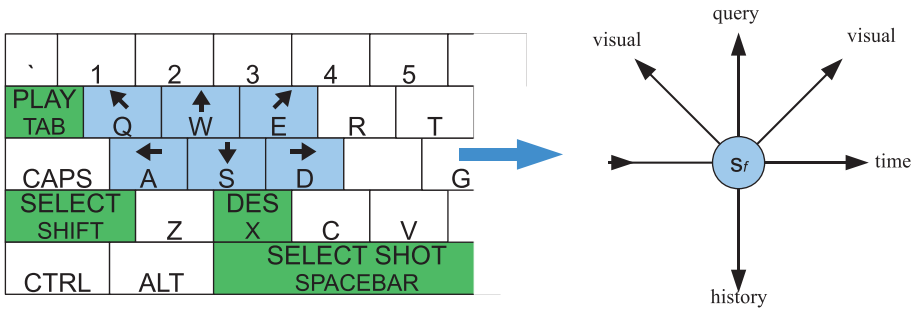

Fig. 4. The layout of the interface is based on the required keyboard interactions. The user navigates from center-placed focal shot s_f, to shots in related threads using several directional keyboard commands. The shots that are increasingly further away from s_f are more and more peripheral to the user, and require more actions to reach. Other interface actions, such as playing a shot (tab), selecting (space), or deselecting (x) shots, or selecting while browsing (shift + direction) are also directly controlled by the keyboard. All keyboard commands are placed such that all actions are accessible with the left hand only, leaving the right hand free to use the mouse if needed.

3.3 Active Zooming

To enable users to inspect a thread more closely, or to label many shots at once as required in Guideline 3, we propose active zooming, which smoothly rearranges the shots in a chosen thread as a grid on the screen, while hiding all other threads. This allows the user to "zoom in" into a thread in such a way that it does not violate Guideline 8.

Navigation options change depending on whether the chosen thread is static or dynamic. For static threads the number of initial shots on a grid is fixed, and users can navigate between pages of results, or increase/decrease the number of shots in a grid. This allows users to rapidly browse through a single static thread.

For dynamic threads, the contents are determined by some user-given origin, and the initial number of shots depends on their similarity to the focus shot. The number of items shown is determined by the measure of similarity. The system automatically derives an optimal threshold based on the similarity decay curve, and users have the option to interactively increase or decrease the cutoff threshold which is initially set to show the top 20% results when looking at the similarity scores. Changing T allows a user to adjust the context and thereby show more images with the chance that they will not be similar anymore, or less images with the chance that images that were similar are missed. Independent of the type of thread, users always have the choice to label single shots, label all shots at once, or return to the previous visualization by stopping active zooming.

3.4 Relevance Feedback

The default navigation thread initially starts with the results from a user query, meant as a starting point into the collection only. We expect these results to be exhausted after a number of iterations. To be able to continue searching through the collection we use a relevance feedback method that uses the entire collection to gather new results.

During the search session users hopefully find relevant shots within the collection. As soon as there are enough results or when users indicate that the default navigation thread is no longer suitable, the system initiates a background learner process. The specific implementation of the learner process can be altered. We chose to take this to the largest extent possible, and train a Support Vector Machine classifier [Chang and Lin 2001] on the entire dataset each time users find potentially relevant results.

The required positive and negative examples for this are automatically derived from user actions in the browser interface. Positives are selected from the set of user-marked relevant results. Retrieving negative examples is more difficult, as we do not wish to burden users with extra steps of marking irrelevant items. Negatives are therefore based on unobtrusive user monitoring of the shown focus and context. When something is relevant, users will navigate toward it and thereby placing it in the focus. During this, potential relevant shots will appear in other threads in the context, with each individual shot often appearing multiple times during the navigation process. When users do not explicitly undertake action to mark these as relevant, the system will eventually mark these as negative examples for the search task. To do so, the system keeps track of how many times a shot was shown in the interface. Each shot x accumulates a chance p_x^t that the shot is seen over time t, which can be 1 maximum. This is determined by the summation of a weighted neighborhood distance to the focal shot, where a shot actually being the focal shot has the highest chance to be seen.

$$p_x^t = min\left(1, p_x^{t-1} + w \times D_{s_f,s_x}\right)$$

As soon as p_n reaches a certain threshold the shot is marked as a negative result. This whole process is unobtrusive, though users have an option to explicitly mark shots as negative as well, which explicitly sets $p_n = 1$. See also Figure 4.

Next, the system trains an SVM classifier, as commonly used in image/video retrieval [Tong and Chang 2001; Gosselin and Cord 2004; Chen et al. 2005; Cord et al. 2007]. Again, there are several kinds of possible solutions here, though the core consideration here is interactive processing time. In our specific implementation positive and negative examples are transported to a compute cluster and the relevant and irrelevant examples are used to train a Support Vector Machine model [Chang and Lin 2001], using the procedure as described in Snoek et al. [2008] on 4000 dimensional feature vectors. Because speed is essential here, we have adapted the SVM kernel to use a precomputed kernel matrix of intershot distances, which is kept in the distributed memory of the cluster [Snoek et al. 2008]. This allows new models to be trained at near-interactive speeds. A typical run with 200 shots selected takes about 2 to 6 seconds on a dataset of 200 hours of video consisting of 35,766 individual elements. For our purposes this is fast enough, so we did not need to consider further optimizations such as iterative retraining of SVM models or using other classifiers.

Results are fed back into the interface as an updated default navigation thread in order to keep navigation clear, as per Guideline 5.

3.5 Putting It All Together

The aforeside techniques have been incorporated into the generalized workflow for targeted video search. See Figure 5 for an update with respect to Figure 1. The preceding interface rules have subsequently been used to define the MediaMill ForkBrowser targeted video search interface, which displays a single navigation context with the focal shot in the center of the visualization. Each thread is represented as a list of images in a specific direction, so that users can spatially map them out and follow a path through the dataset. The focal shot and its context is displayed using a fisheye [Furnas 1986]-like method to further guide user attention to the focal shot. When a single thread shows significantly more relevant results than others, users have the option to enable active zooming. This enables a seamless zoom into the selected thread, which then provides users with a means of inspecting and selecting large sets of relevant items in less user interaction steps for that particular thread only. See Figure 4 for a mapping between threads and tines, and Figure 11 for a screenshot depicting the ForkBrowser together with active zooming.

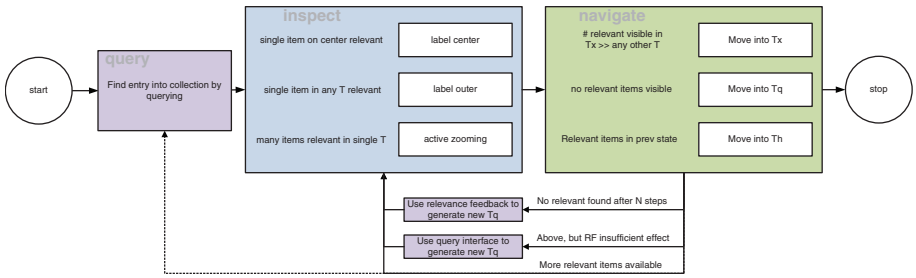

Fig. 5. Updated workflow which shows how ForkBrowser is used to perform targeted video search. See Section 3 for details on the individual methods used.

4. EXPERIMENTAL SETUP AND EVALUATION

Each individual component of the ForkBrowser has one or several parameters to be configured which we can tune for optimal efficiency. Of course, it is not feasible to optimize each parameter with a separate user study. We therefore opt for parameter optimization using simulated users whereas the real users evaluate the system as a whole.

4.1 Dataset and Task

We use the same dataset for all our experiments, which is the TRECVID benchmark dataset of 2007 and 2008. This is a video collection totaling 300 hours of video material, provided by the Netherlands Institute for Sound and Vision. The video collection contains news magazines, science news, news reports, documentaries, and educational programs. Shot boundary information for the collection is provided by Petersohn [2004], and automatic speech recognition on these videos in Dutch is provided by Huijbregts et al. [2007]. We added a series of semantic concept detector results, as explained in Snoek et al. [2008] to allow users to perform query by concept searches into this collection.

On top of this dataset NIST provides a series of search topics. 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.” Topics for 2008 were more specific, and contained, for example, *“Find shots of a person looking through a microscope.”* For easy reference we use the official TRECVID topic numbering scheme.

4.2 Experimental Setup

There are several parameters in the components of Figure 5 that can be tuned in order to provide the most possible relevant results in each phase of interactive search. Specifically, we want to know the following.

—Q1: The optimal number of displayed threads,
—Q2: The optimal display length of individual threads,
—Q3: When to use active zooming,
—Q4: When to use relevance feedback.

To answer these questions we have implemented the workflow described in Figure 5 as a deterministic simulated user experiment by viewing this workflow as a deterministic state machine. For each step in the query and browse stages, there is a best choice of action, based on which shots are relevant to a query and which aren’t. By explicitly expanding this state machine we get a deterministic user simulator. The parameters of each component can be varied, and the effects of the parameter changes measured.

Each action performed is denoted as a User Interaction Step, or UIS. In order to have consistency between the various simulated user experiments we chose for a fixed limit of User Interaction Steps rather than a limit in search time. Each real-world user would require a different time to evaluate information, and this is difficult to translate to simulated users. We measured approximately 1.9 ± 0.8 UIS per second on average over all topics for all expert users, which translates to 574 UIS in a five-minute timeframe. The UIS/second also varies according to the difficulty per topic.

We chose 500 UIS as a limit, which approximates to slightly less than 5 minutes for expert users. After 500 UIS the search session is stopped, and results are tabulated. We assume that all topics are equally complex, and use a default tine length of 6 shots in each direction. As soon as a relevant shot is positioned as the focal shot s_f, it is marked as "retrieved", and will not be considered anymore if it appears elsewhere in the interface.

The experiments Q1 to Q4 above shall aid with finding optimal parameter settings for the ForkBrowser, and as such do not try to optimize implementations for all the guidelines. Specifically, these experiments help to optimize performance in Guidelines 4 (show unexplored), 6 (inspect navigation options), and partially 9 (mapping between nav and vis).

4.3 Simulating the Query Stage

In interactive search, we need an entry point into the data, defined as the initial query thread. Users create a query thread by selecting (a combination of) semantic concepts, keywords, or example images to be used in query by example. For the user simulation we need something similar without having to resort to manual intervention. However, which query method to use is nontrivial for a computer. We therefore perform all simulated user experiments for all topics with the set of semantic concepts defined in Snoek et al. [2008] as the set of possible initial queries. The concept that yields the best results for a topic is taken as the query thread for that topic.

4.4 Simulating the Browsing Stage

Given a specific s_f, the system generates a set of shots for each of the visible threads during the browsing stage. The first s_f is the first result in the query thread. The simulated user follows the same diagram as given in Figure 5. Specifically, for each UIS the simulated user performs one of the following actions.

(1) Judge a shot as "relevant" when this is labeled as such in the ground truth.
(2) If there are relevant shots visible in any visual similarity or time thread, follow the thread with the most results visible.
(3) If there is no relevant shot visible, follow the query thread.

This is the basic browsing scenario for simulated users. Each of the experiments listed in the following sections alters this basic ruleset slightly for the purpose of the experiment.

4.5 Q1: Estimate Benefit of Multiple Relevant Threads

In this experiment we measure the benefit of displaying extra threads for an individual focal shot by using simulated users for various combinations of enabled/disabled threads. Specifically:

—only the query thread enabled,
—query and time threads enabled,
—query thread and one or more dynamic visual similarity threads enabled,
—query, time, and dynamic visual similarity threads enabled.

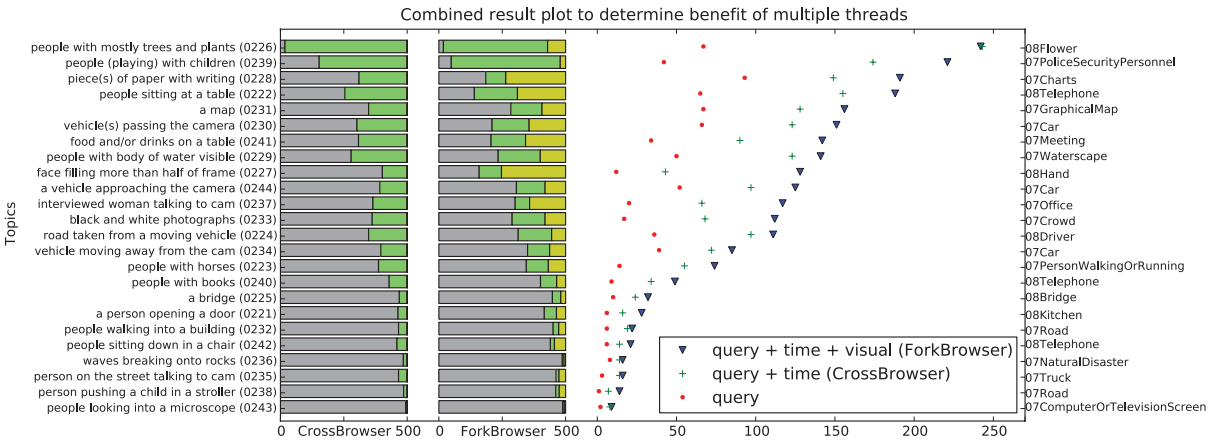

Fig. 6. This graph shows a combination of results. Each row shows the result of an individual TRECVID 2008 topic, with the number indicating the TRECVID topic ID. The topics are sorted by best performing ForkBrowser result for easy reading. The left two bar plots indicate which kind of actions simulated users performed during their 500 User Interaction Steps. Move actions through query thread are gray, through time are green, and visually similar are yellow. The scatter plot on the right displays the resulting number of relevant results found after these interactions. The rightmost text indicates the chosen starting concept for each topic. Results indicate that having time and visual similarity threads is beneficial, since they provide a large boost to the found number of results.

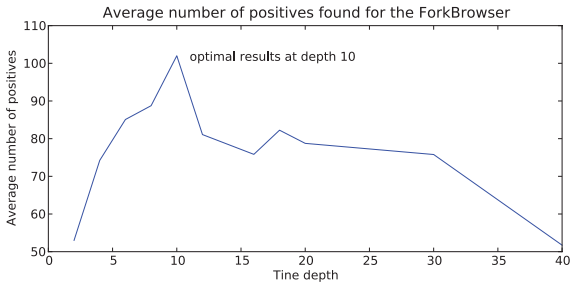

Fig. 7. Average number of positives found after 500 User Interaction Steps for all topics at varying tine lengths for the ForkBrowser.

The two visual similarity threads here are based on Gabor and Wiccest [van Gemert et al. 2010] similarity features, respectively.

The results, depicted in Figure 6, indicate that having more threads yields better performance in almost all cases. Also, the results indicate that the extra similarity threads do allow users to find some specific results faster. For example, in topic 227: face filling more than half the frame a large number of results were found because of the visual similarity between faces.

4.6 Q2: Estimate Optimal Tine Display Length

We performed a simulated user experiment to determine the length of tines on screen. Real-world users are limited in what they can understand within a reasonable timeframe on screen, while simulated users do not have this drawback. We are interested to see what would have been the optimal length for simulated users, and performed experiment Q1, but now with all threads enabled, and varying tine lengths. The results of this experiment are depicted in Figure 7. Simulated user results indicate that the tine length should not go beyond 10. After that the number of interaction steps needed to reach

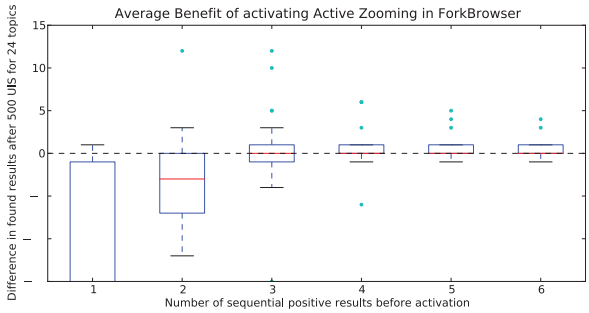

Fig. 8. These boxplots show the difference between the number of relevant results retrieved after 500 UIS when active zooming is enabled after the user spots N positive results in a single thread. Postive scores indicate benefit when active zooming is used. The results indicate that using active zooming when too few positive results are visible on average deteriorates the number of found results, though several individual topics exist which benefit from AZ. On average, when the user always activates AZ when more than 4 positive results are visible the end result will yield more relevant shots.

that individual relevant result from a single thread takes so much UIS that it would have been better not to see that result at all, and discover it elsewhere instead. Analysis with real users showed that the number of shots shown in each direction was topic dependent, with more difficult topics requiring the user to zoom-in more, which reduced tine length. On average a tine length of 6 was used. For the ForkBrowser interface, which shows 6 threads (2 similarity threads, time thread in both directions, query thread and history thread) this implies 36 shots are visible. In the worst-case scenario where all shots are different except the ones of the last navigation direction this implies $5 \times 6 + 1 = 31$ previously unknown shots are presented to the user.

4.7 Q3: Determine when Active Zooming Should Be Used

For this experiment we want to determine at which number of relevant shots in a single thread it is most efficient to enable active zooming. Also, we want to determine whether the number of currently displayed relevant shots in other threads affects performance, and if so, when this would influence results. The experiment is set up in such a way that a simulated user browses through a series of 4 related threads with 6 shots visible at a time from each thread. When active zooming is enabled the user either sees 30 shots from that thread, or, for dynamic threads, a number determined by the visual similarity between the results and the focus shot. The act of enabling or disabling active zooming costs one UIS, the act of selecting all visible items costs another UIS, and deselecting mistakes also costs one UIS per shot. The simulated users will determine whether to proceed with selecting results or switching back to the browser just after enabling active zooming, so a mistaken activation still costs 2 UIS. The choice to switch to active zooming mode is dependent on the number of shots relevant in a single thread versus the number of relevant shots seen in other threads. We measure active zooming performance by altering the threshold at which the simulated user activates active zooming.

The results are shown in Figure 8. As expected, these results indicate that AZ can best be used when multiple positive results are visible in a single thread. As soon as there are 4 or more positive results the number of UIS required to select positive results is lower than without AZ, though there are several individual topics which still benefit even when less consequitive relevant shots are shown. In a real scenario users will typically have a better indication of when to use active zooming because of the known search need, so users typically know when to use AZ even when less than 4 shots are relevant.

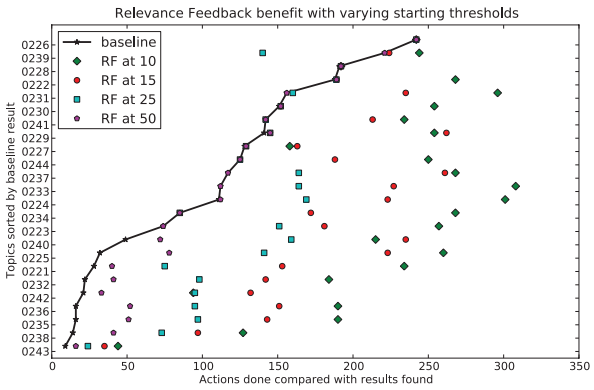

Fig. 9. This graph shows the benefit of activating relevance feedback after different numbers of UIS without finding any results for all topics, sorted by baseline result. The baseline here is based on the results obtained by the simulation run with the ForkBrowser, as depicted in Figure 6. The graph shows that activating RF after only a few nonrelevant items found is already highly beneficial, and enables the user to greatly increase the number of found results.

4.8 Q4: Determine Benefit of Relevance Feedback

As indicated in Section 3.4, when users cannot find new relevant results within the collection an option is to use relevance feedback to rerank the collection based on past browsing behavior. However, using relevance feedback when there are not enough results gathered might yield a suboptimal ranking. Also, it can happen that there are still relevant results in T_q which are not yet visible, and browsing a little longer would have shown these results.

To determine this, we measure the number of found results within 500 UIS for various numbers of user interaction steps done without selecting any relevant result. After this limit is reached, the system performs reranking and this is then loaded as the new T_q and browsing continues from that point until 500 UIS total have been reached. This implies that a single retrieval session can feature multiple relevance feedback rerankings.

Results are displayed in Figure 9. Results indicate that the benefit is greatest when the system activates RF after only 10 negative results have been found. Note that the relevance feedback algorithm does require moderate computing requirements, and in our case each relevance feedback reranking typically takes two to five seconds to calculate. With this timeframe taken into account it might be beneficial to let the user browse more than the optimal number of 10 results.

When the RF activation threshold is set at 25 UIS, the benefit is only significant with topics that yielded poor results in the baseline. In almost all cases performances increase or stay the same. The one exception is 239: “people playing with children”, where results significantly deteriorate when using relevance feedback. This is caused by the fact that most of the relevant shots were found using the time thread of a single video, and the relevance feedback mechanism inhibited finding this specific video.

4.9 ForkBrowser Performance at TRECVID

In order to evaluate targeted video search using the ForkBrowser we participated in the 2007, 2008, and 2009 NIST TRECVID Interactive Search tasks [Smeaton et al. 2006]. This gives us a framework 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 state-of-the-art in video retrieval [Hauptmann and Christel 2004]. See Figure 10 for an overview of results for the topics of 2008. The graph shows that the ForkBrowser and CrossBrowser consistently received high marks.

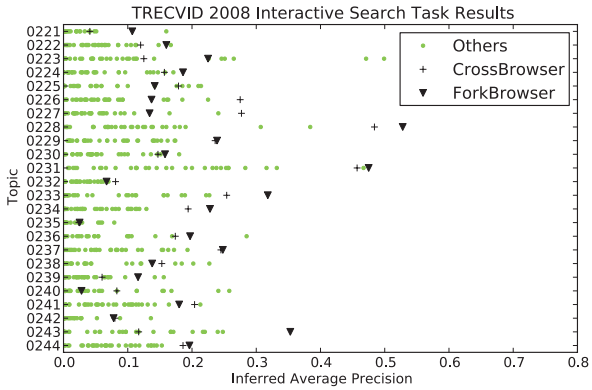

Fig. 10. Trecvid 2008 interactive search results. The graph shows that the CrossBrowser and ForkBrowser consistently yield high scores on a variety of different topics.

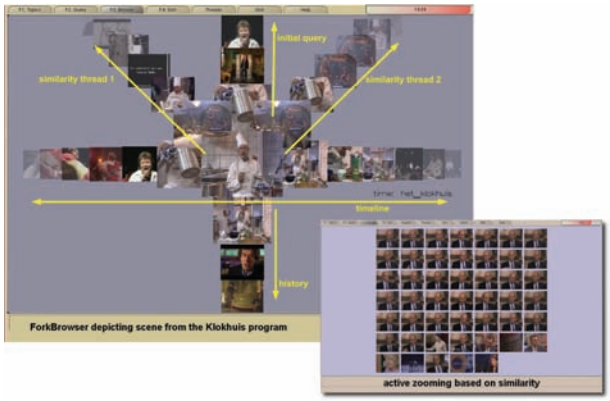

Fig. 11. Thread-based visualization in the ForkBrowser. In this example from the TRECVID 2007 corpus, the top tine displays query-by-keyword results, the horizontal tines display the timeline of a "Klokhuis" episode, the diagonal tines display dynamic threads related to the center image, and finally the stem of the fork displays the current browse history. The picture on the right shows the ForkBrowser with active zooming enabled.

In 2009 similar results were achieved, even though the dataset increased threefold, with an average precision of 0.246 for the CrossBrowser and 0.241 for the ForkBrowser, both browsers using active zooming and relevance feedback strategies, compared to 0.186 of the nearest other system. Also, the capability to view details on demand of individual shots by either looking at motion icons or by looking at the context in time, yielded many relevant results. For example, most of the results obtained for 2009 topic "Find dogs walking, running or jumping" were only found thanks to the capability of playing shots at rapid speeds.

5. DISCUSSION

The results in the previous section indicate that the ForkBrowser has excellent performance in the hands of expert users. In this section we discuss some of the unexpected results and some limitations in terms of efficiency and usability.

When we look at efficiency, we found that the quality of results obtained with the ForkBrowser are less dependent on the initial query than we expected. For example, the topic 239: "Find people standing,

Initial query	top 100	total found
people walking or running	18	206
crowd	10	135
sports game	9	118
people marching	6	89
vegetation	5	116
military personel	5	74

Fig. 12. A list of the top 6 initial queries for *"Find people standing, walking or playing with children"*. Depicted is the number of relevant results top 100 in the initial query, and number of relevant results found after 500 steps using that query as a starting point.

walking or playing with children" yields 206 positive results after 500 interaction steps when the user starts with the concept person walking or running. This concept was determined as optimal by the system, as it contains 18 positive results in the top 100. When we look at several other concepts, however, see Figure 12, we see that this number of found results, though lower than 206 still yield relatively high numbers of results. The number of relevant results in the top 100, however, lowers drastically, and the conceptual link between some initial queries and the topic is unlikely to be found by real-world users. We found similar results for many other topics. This seems to indicate that interfaces that allow users to navigate through multiple threads help alleviate the effect of poorly chosen initial queries.

The guidelines proposed in Section 2 give us a framework for designing an interactive targeted video search system. The ForkBrowser implements all of these guidelines, but also many other systems in literature do already implement many of the guidelines, and these systems might be further improved by implementing all of them. We also note here that our evaluations didn't take into account the ease of learning the interface. The ForkBrowser interface is quite different from other user interfaces, which requires users to spend time with the interface before they are able to master it. However, the TRECVID results do show that the ForkBrowser in the hands of an expert user yields very good results. Other systems based on more traditional layouts might be easier to learn. Examples from recent TRECVID benchmarks include the Informedia Storyboard interface [Christel and Yan 2007] and the FXPal MediaMagic [Adcock et al. 2007], which use a recognizable grid layout to browse through results, while the VisionGo video search engine [Luan et al. 2007] uses a layout where the user browses through a linear list at only three images per step.

6. CONCLUSION

In this article, we proposed a technique for targeted search through large video collections. Targeted search focuses on fast retrieval based on an initial set of results from a query system. Based on a focus + context-based browsing methodology, we have proposed a set of guidelines to which efficient targeted video search interfaces should adhere. We compare these guidelines with existing interfaces in literature, which do already implement many of these guidelines.

Subsequently, we follow these guidelines in designing the ForkBrowser, a browser that allows users to navigate through a collection by following related threads linked to a single focal shot.

We set up a user simulation framework to validate and optimize parameters of the ForkBrowser in a series of user simulation experiments. Results show a clear benefit of having a multithread browser versus a single-thread browser. Also, the experiments show a clear benefit of using relevance feedback and indicate under which conditions it should be applied. Furthermore they indicate that active zooming is very helpful for certain topics, and never deteriorates results. Besides automated experiments we also performed real user experiments by participation in the international TRECVID 2007, 2008, and

2009 benchmarks. The results show that the ForkBrowser, together with active zooming and relevance feedback, consistently performs very well.

REFERENCES

ADCOCK, J., COOPER, M., AND CHEN, F. 2007. Fxpal mediamagic video search system. In *Proceedings of the ACM International Conference on Image and Video Retrieval*. 644–644.

ADCOCK, J., COOPER, M., GIRGENSOHN, A., AND WILCOX, L. 2005. Interactive video search using multilevel indexing. In *Proceedings of the ACM International Conference on Image and Video Retrieval*. Lecture Notes in Computer Science, vol. 3568. Springer. 205–214.

CHANG, C.-C. AND LIN, C.-J. 2001. Libsvm: A library for support vector machines. http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

CHEN, M.-Y., CHRISTEL, M., HAUPTMANN, A., AND WACTLAR, H. 2005. Putting active learning into multimedia applications: Dynamic Definition and refinement of concept classifiers. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*. ACM, New York, 902–911.

CHRISTEL, M. G., HUANG, C., MORAVEJI, N., AND PAPERNICK, N. 2004. Exploiting multiple modalities for interactive video retrieval. In *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing*. Vol. 3. 1032–1035.

CHRISTEL, M. G. AND YAN, R. 2007. Merging storyboard strategies and automatic retrieval for improving interactive video search. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*. ACM, New York, 486–493.

CORD, M., GOSSELIN, P. H., AND PHILIPP-FOLIGUET, S. 2007. Stochastic exploration and active learning for image retrieval. *Image Vis. Comput. 25,* 1, 14–23.

DE ROOIJ, O. AND WORRING, M. 2010. Browsing video along multiple threads. *IEEE Trans. Multimedia 12,* 2, 121–130.

FURNAS, G. 1986. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 16–23.

GOSSELIN, P. H. AND CORD, M. 2004. A comparison of active classification methods for content-based image retrieval. In *Proceedings of the 1st International Workshop on Computer Vision Meets Databases*. ACM, New York, 51–58.

HAUPTMANN, A. G. AND CHRISTEL, M. G. 2004. Successful approaches in the trec video retrieval evaluations. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*. ACM, New York, 668–675.

HAUPTMANN, A. G., LIN, W.-H., YAN, R., YANG, J., AND CHEN, M.-Y. 2006. Extreme video retrieval: Joint maximization of human and computer performance. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*. ACM Press, New York, 385–394.

HUIJBREGTS, M., ORDELMAN, R., AND DE JONG, F. 2007. Annotation of heterogeneous multimedia content using automatic speech recognition. In *Proceedings of the International Conference on Semantics and Digital Media Technologies*. Lecture Notes in Computer Science.

LEW, M. S., SEBE, N., DJERABA, C., AND JAIN, R. 2006. Content-Based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl. 2,* 1, 1–19.

LUAN, H.-B., NEO, S.-Y., GOH, H.-K., ZHANG, Y.-D., LIN, S.-X., AND CHUA, T.-S. 2007. Segregated feedback with performance-based adaptive sampling for interactive news video retrieval. In *Proceedings of the 15th International Conference on Multimedia*. ACM, New York, 293–296.

NATSEV, A. P., HAUBOLD, A., TEŠIĆ, J., XIE, L., AND YAN, R. 2007. Semantic concept-based query expansion and re-ranking for multimedia retrieval. In *Proceedings of the 15th International Conference on Multimedia*. ACM, New York, 991–1000.

NATSEV, A. P., NAPHADE, M. R., AND TEŠIĆ, J. 2005. 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*. ACM, New York, 598–607.

PETERSOHN, C. 2004. Fraunhofer HHI at TRECVID 2004: Shot boundary detection system. In *Proceedings of the TRECVID Workshop*.

RAUTIAINEN, M., SEPPÄNEN, T., AND OJALA, T. 2006. On the significance of cluster-temporal browsing for generic video retrieval: A statistical analysis. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*. ACM, New York, 125–128.

ROBERTSON, G., CZERWINSKI, M., LARSON, K., ROBBINS, D. C., THIEL, D., AND VAN DANTZICH, M. 1998. Data mountain: Using spatial memory for document management. In *Proceedings of the 11th Annual ACM Symposium on User interface Software and Technology*. ACM Press, New York, 153–162.

SMEATON, A. F., OVER, P., AND KRAAIJ, W. 2006. Evaluation campaigns and trecvid. In *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. ACM Press, New York, 321–330.

SNOEK, C. G. M., VAN DE SANDE, K. E. A., DE ROOIJ, O., HUURNINK, B., VAN GEMERT, J. C., UIJLINGS, J. R. R., HE, J., LI, X., EVERTS, I., NEDOVIĆ, V., VAN LIEMPT, M., VAN BALEN, R., YAN, F., TAHIR, M. A., MIKOLAJCZYK, K., KITTLER, J., DE RIJKE, M., GEUSEBROEK, J.-M., GEVERS, T., WORRING, M., SMEULDERS, A. W. M., AND KOELMA, D. C. 2008. The MediaMill TRECVID 2008 semantic video search engine. In *Proceedings of the 6th TRECVID Workshop*.

SNOEK, C. G. M., WORRING, M., KOELMA, D. C., AND SMEULDERS, A. W. M. 2007. A learned lexicon-driven paradigm for interactive video retrieval. *IEEE Trans. Multimedia 9,* 2, 280–292.

SUNDARAM, H. AND CHANG, S.-F. 2001. Condensing computable scenes using visual complexity and film syntax analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. 70.

TONG, S. AND CHANG, E. 2001. Support vector machine active learning for image retrieval. In *Proceedings of the 9th ACM International Conference on Multimedia*. ACM, New York, 107–118.

VAN GEMERT, J. C., SNOEK, C. G. M., VEENMAN, C. J., SMEULDERS, A. W. M., AND GEUSEBROEK, J. M. 2010. Comparing compact codebooks for visual categorization. *Computer Vision and Image Understanding* in press.

WANG, D., LIU, X., LUO, L., LI, J., AND ZHANG, B. 2007. Video diver: Generic video indexing with diverse features. In *Proceedings of the International Workshop on Multimedia Information Retrieval*. ACM, New York, 61–70.

WARE, C. 2000. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

YAN, R., NATSEV, A., AND CAMPBELL, M. 2007. An efficient manual image annotation approach based on tagging and browsing. In *Workshop on Multimedia Information Retrieval on the Many Faces of Multimedia Semantics*. ACM, New York, 13–20.

YEE, K.-P., SWEARINGEN, K., LI, K., AND HEARST, M. 2003. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, 401–408.

ZAVESKY, E., CHANG, S.-F., AND YANG, C.-C. 2008. Visual islands: Intuitive browsing of visual search results. In *Proceedings of the International Conference on Content-Based Image and Video Retrieval*. ACM, New York, 617–626.

Received June 2010; revised December 2010; accepted May 2011