UvA-DARE (Digital Academic Repository)

Active Bucket Categorization for High Recall Video Retrieval

de Rooij, O.; Worring, M.

DOI

[10.1109/TMM.2013.2237894](https://doi.org/10.1109/TMM.2013.2237894)

Publication date

2013

Document Version

Final published version

Published in

IEEE Transactions on Multimedia

[Link to publication](#)

Citation for published version (APA):

de Rooij, O., & Worring, M. (2013). Active Bucket Categorization for High Recall Video Retrieval. *IEEE Transactions on Multimedia*, 15(4), 898-907.
<https://doi.org/10.1109/TMM.2013.2237894>

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

Active Bucket Categorization for High Recall Video Retrieval

Ork de Rooij and Marcel Worring, *Member, IEEE*

Abstract—There are large amounts of digital video available. High recall retrieval of these requires going beyond the ranked results, which is the common target in high precision retrieval. To aid high recall retrieval, we propose Active Bucket Categorization, which is a multicategory interactive learning strategy which extends MediaTable [1], our multimedia categorization tool. MediaTable allows users to place video shots into buckets: user-assigned subsets of the collection. Our Active Bucket Categorization approach augments this by unobtrusively expanding these buckets with related footage from the whole collection. In this paper, we propose an architecture for active bucket-based video retrieval, evaluate two different learning strategies, and show its use in video retrieval with an evaluation using three groups of nonexpert users. One baseline group uses only the categorization features of MediaTable such as sorting and filtering on concepts and fast grid preview, but no online learning mechanisms. One group uses on-demand passive buckets. The last group uses fully automatic active buckets which autonomously add content to buckets. Results indicate a significant increase in the number of relevant items found for the two groups of users using bucket expansions, yielding the best results with fully automatic bucket expansions, thereby aiding high recall video retrieval significantly.

Index Terms—Active learning, interactive video retrieval, multi class categorization, relevance feedback, user evaluation, video retrieval.

I. INTRODUCTION

IN recent years the size of digital video collections containing for example home videos, surveillance footage, broadcast data or social science data, is growing at a staggering rate. In all these cases limited metadata are stored or available at the time of capture. For effective retrieval, manual labeling and categorization is required. Due to the large volumes this cannot be done at detailed granularity. Labels are limited to video or collection level, typically a title or short description of contents. Searching for specific items in these collections is difficult.

With limited metadata it is often still possible to get results with reasonable precision. Traditional search engines, such as Google or Bing, focus on providing such high precision, with the most relevant results returned first. This goes at the expense

of recall. In applications where high recall, finding all relevant results in a collection, is required, like in forensic or scientific studies, the level of recall obtained by traditional search engines is not enough. For high recall retrieval, we either need all entries in the collection to have detailed metadata, or a search engine that can cope with this lack of metadata.

To allow high recall retrieval of items in video collections on any granularity scale we need metadata at video level, shot level, and frame or event level. Adding this information manually by having the multimedia items tagged by a single user is often not feasible. Providing interfaces that allow for collaborative tagging helps, because many users add metadata to the collection. For example Flickr [2] and YouTube [3] allow users not only to tag their own material, but also content uploaded by others, which in turn enriches the entire collection. This works well for popular video content. However, not all types of content can be processed like this. For example, annotations on home video collections are usually wanted and needed only by the owner of the material. Security camera footage or video captured during a forensic investigation cannot be annotated online at all, because of privacy and confidentiality considerations. As a result, many collections do not have sufficient metadata to allow effective high recall retrieval.

If we assume that there will not be a complete metadata description of all items in a collection, we need alternative means for high recall retrieval. For this, we first look at content based video retrieval techniques, which allow the user to find items based on their (visual) content alone.

A. Video Content Analysis

In the early years content retrieval was based on low level features, and many of these systems required specialized forms of input from the user [4]. Examples include drawing sketches or providing example images. This is not always possible or practical. Moreover the low-level visual feature representation used for querying often does not correspond to the users intent, a problem known as the semantic gap [4]. Several solutions have been proposed and examined in recent literature to try to solve this problem [5]. The most effective solutions use generic concept detection strategies [6]–[11] which allow for automatic labeling of people, objects, settings or events within video content, albeit with varying performance. These methods first segment videos into individual shots: fragments of video from a single camera capture. These shots are then analyzed individually and low level visual features are extracted. Next, a supervised machine learning algorithm uses these features together with example images to determine the presence of a series of semantic concepts, such as “outdoor”, “face”, or “building”, for

Manuscript received December 16, 2010; revised April 20, 2011; accepted August 14, 2011. Date of publication January 04, 2013; date of current version May 13, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Shrikanth Narayanan.

The authors are with the Intelligent Systems Lab Amsterdam, University of Amsterdam, Amsterdam 1098 SJ, The Netherlands (e-mail: orooij@uva.nl; m.worring@uva.nl).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TMM.2013.2237894

each shot in the collection. The combined output of several semantic concept detectors yields an enriched set of metadata that can be used to yield a ranking of individual shots.

Unfortunately, semantic concept extraction algorithms are not perfect. Furthermore, combining the output of several detectors to form a complex query yields even worse performance. To improve the quality, results need to be verified and/or the user posed query needs to be reformulated. An interactive video retrieval interface that 1) allows the user to adjust their search parameters interactively to see what yields the best results, and 2) allows the user to present relevance feedback on individual results, is essential.

B. Video Retrieval Interfaces

We highlight a few interactive interfaces based on ranked results. These methods fall into two categories. First, systems optimizing the visual appearance to allow users to see related and relevant videos faster. Second, systems which optimize the efficiency of judging relevant items.

In the first category Zavesky *et al.* [12] and Cao *et al.* [13] place query results in a structured grid, re-ordering the specific location of images in the grid so that they are optimized for fast recognition of similar results or categories. The MediaTable system [1] displays 20–50 ranked concept results in a table to quickly identify promising ones. Alternatively, results can also be organized on video story level. Christel *et al.* [14] display results using a storyboard. The FXPal MediaMagic system [15] allows story level search and displays results in a grid layout with individual image sizes varied to indicate relevance to the query. De Rooij *et al.* [8] show several navigational options inside a story.

The second category includes methods that improve the speed in which users can determine whether results are valid. For example the Extreme Video Retrieval system [16] uses rapid serial visual presentation to allow fast judgment of retrieval results, and the Vision Go system [17] allows rapid tagging of 3 items at once. MediaTable [1] employs several visualizations in which items can be categorized quickly.

All these methods provide the means to select relevant items quickly and provide good starting points for learning from those interactions.

C. Learning Algorithms to Aid During Retrieval

The previous section looked at the user interface side. On the computational side we also see solutions emerge. For example, video retrieval systems can employ explicit relevance feedback methods to iteratively increase the quality of results. At each iteration, the system decides which items the user needs to see and verify. For example [18] uses a Support Vector Machine [19], selecting the elements closest to the boundary between relevant and non-relevant items, because these are the most uncertain elements. Goh *et al.* [20] propose different sampling strategies for different types of search needs. The system in [17] uses a background recommender system that chooses between several types of sampling strategies based on user responses. This choice alters the method in which the relevance feedback system obtains results, which ultimately changes the stream of results

the user labels. All of the above systems use these algorithms to add functionality to their interfaces, enabling users to find results more effectively.

A downside of such explicit techniques is that they often are on-demand features. Users invoke the algorithm which then produces new potentially relevant items. This act of requesting leads the user to expect something of the answer: if you ask something explicitly you want to get a good answer. Furthermore, since users are now actively waiting for the results, the processing needs to be instantaneous. This limits the amount of items that can be processed. Typically only a reranking of the top-N items is possible.

D. Our Proposed Solution for High Recall Retrieval

To provide users with a system for high recall video retrieval we need to overcome the limitations of the search interfaces and learning engines. In this paper, we propose Active Bucket Categorization, a method and framework for high recall video retrieval. This method combines state of the art video analysis algorithms with an easy to use user interface build upon MediaTable.

Using the MediaTable interface, users sort and categorize parts of videos from the collection into buckets, a metaphor for a set. These buckets are user defined, and can contain anything the user wants to categorize or retrieve. A learning algorithm then takes the current contents of individual buckets, and extends these by looking for related footage in the entire collection. All this is done unobtrusively and in the background. The categorization ends when all footage in the collection is placed into buckets.

This paper is organized as follows. In Section II we present this method, and we experimentally validate the method in Section III, followed by a conclusion in Section IV.

II. ACTIVE BUCKETS

In this section we describe a complete framework consisting of several components, all focused on processing buckets of multimedia items. We first introduce what these buckets are, and then elaborate on our active bucket categorization method.

A. Bucket Based Categorization

Video retrieval is defined as the act of finding relevant items specific to a search query for a given collection of video fragments:

$$X = \{x_1, \dots, x_n\}$$

where each element x_n in the collection is composed of 3 elements: *the original video fragment, its accompanying metadata* such as filenames, user tags and titles, and computer generated *concept descriptors* indicating presence/absence of semantic concepts. As both accompanying metadata and concept descriptors will be used in a similar way in the system later, we will refer to them simply as metadata.

As indicated, when considering video retrieval we make a distinction between retrieval aiming at precision and retrieval aiming at high recall. High precision is what most current search engines are optimized for, finding a ranking of the collection

such that there are as many relevant results as possible in the top N results, with N typically 20 to 100. In contrast, high recall targets much larger parts of the collection. In fact, high recall retrieval can be viewed as a categorization process, marking all items R which conform to a user posed query, and all items $\neg R$ which don't:

$$R = \{x_1, \dots, x_r\}$$

$$\neg R = \{x_{r+1}, \dots, x_n\}.$$

To aid in making this distinction, our bucket based categorization approach has two underlying principles. First, we focus on minimizing the search space of results, instead of finding all results at once. This is done by iteratively selecting easy-to-discard subsets from the collection. Second, each subset found is expanded automatically using a binary active learning strategy.

For later understanding, let us consider the retrieval task from the data point of view. When there is a semantic concept available as metadata which is directly related to the search query we have a good starting point. Depending on whether the concept relates to data that shares visually similar characteristics or not defines whether from this initial starting point the search is easy (one cluster in feature space) or difficult (various clusters in feature space). When there is no directly related metadata we can still find results indirectly if the data is visually similar, but it is very difficult otherwise. This brings us to the four types of retrieval conditions:

- none A *metadata available, visually similar*
- none B *no direct metadata, visually similar*
- none C *metadata available, visually diverse*
- none D *no direct metadata, visually diverse*

The basis for our categorization approach is formed by MediaTable [1]. MediaTable provides different views on a collection, with a spreadsheet-like table interface, showing the entire collection. This table shows per row a shot of video material, together with a preview image, and all associated metadata. On top of this, several visualization techniques enhance the effectiveness of MediaTable for video categorization and video retrieval. Basic user interactions are defined by *sort* and *select* operations. First, users sort the table, and thereby the collection, on any of its columns. Secondly, they select rows from the sorted results by visually examining the associated shots, and place these in a bucket. The process of iteratively selecting, sorting and placing items in buckets yields effective categorization.

The bucket list B is the set of buckets containing items from the multimedia collection. The user can view the content of the buckets and categorize it further and place it in other buckets. We make a distinction between *system buckets* and *user defined buckets*. System buckets automatically update their contents, whereas user defined buckets are filled by the user. To be precise, we have:

$$B = B_{system} \cup B_{user}$$

$$B_{user} = \{B_1, \dots, B_N\}$$

$$B_{system} = \{B_{everything}, B_{unknown}, B_{seen}, B_{selected}\}$$

where:

- $B_{everything}$: all media items in the collection;

- $B_{unknown}$: all media items in the collection which haven't been categorized yet;
- B_{seen} : all media items that have ever been shown to the user;
- $B_{selected}$: the current selection;

Initially, all elements of the collection are placed in the $B_{everything}$ and $B_{unknown}$ buckets. Categorization is done by moving items from $B_{unknown}$ into any user bucket $B_1..B_N$, typically by selecting elements in some visualization and then select a bucket through a keyboard shortcut, pop-up bucket menu, or by dragging. As soon as $B_{unknown} = \emptyset$ all elements are placed in buckets, and the categorization is complete. Note that system buckets are handled by the system based on the interactions users have with the interface. See Fig. 1 to see how bucket based retrieval is different from traditional video retrieval.

B. System Overview

Active buckets define a method allowing the computer to suggest new relevant items for any bucket, based on the contents of that bucket. These suggestions are based on low-level visual feature characteristics of individual items placed inside a bucket, and those that are currently not inside. Active buckets leverage anything the user has categorized, i.e., anything placed in one of the user buckets. This differs from traditional search systems with embedded learning or relevance feedback components. These often only find materials related to the user query specified. The idea behind active buckets is not to do just that, but optimize every bucket the user is using. For binary classification problems where there is typically a "relevant" and a "non-relevant" bucket this allows users to discard non-relevant results quicker, because once they have been placed in the non-relevant bucket they will not appear as candidates during a search for relevant items. This allows the user to reduce the search space faster.

The primary components of active buckets, and the main contribution of this paper, are shown in Fig. 2. The active buckets extend MediaTable by unobtrusively observing changes in user selections. Based on this the active bucket engine decides which bucket is the most prominent candidate to extend with more suggestions. This bucket is then passed through to a sampling and learning engine which selects items from several buckets and uses these to first dynamically train a new model on the whole dataset, and then return any new suggestions to the user. We discuss each of these components in more detail in the following sections.

C. Active Bucket Algorithm

The active bucket algorithm continuously monitors user interactions with the user buckets B_{user} . As soon as the user interacts with any bucket, either by placing items into the bucket, or removing items from the bucket, this bucket is added to a bucket processing queue. Each time the learning engine is ready for another cycle, one of these buckets is promoted and removed from the queue based on the time of their last alteration. The next bucket to be processed is therefore always the bucket with the most recent user updates. The system buckets B_{system} cannot be placed into the queue.

Fig. 1. Traditional single-category search vs our active bucket based retrieval approach. This figure shows how the collection could be iteratively split into buckets, shown here as color coded bars. The approach depicted on the left uses traditional search. The approach depicted in the center uses buckets to aid in splitting the collection into known-but-irrelevant and potentially relevant parts to speed up retrieval. The rightmost approach adds active buckets, which aids in reducing the search space as quickly as possible.

Fig. 2. An overview of active bucket based categorization. The graph is divided into user actions, and system actions resulting from this. MediaTable provides options for sorting on any type of metadata in the collection, selecting results using the sort result using various visualizations, and adding these to buckets. The Active Bucket Engine monitors the buckets for changes, and processes these one by one in the background, alerting the user when interesting results are found. To keep the user in control the user may also manually trigger passive bucket expansion which requests related results for a specific bucket and shows these.

Besides automated active bucket expansions the active bucket engine also allows for passive bucket expansions. In this case a user can manually and immediately request an update based on the contents of any bucket, at the cost of waiting for the results. When this happens the automated bucket expansion queue is interrupted, and the chosen bucket is immediately sent to the sampling engine. The system waits for the results to be available, and shows them to the user in their current visualization.

Based on the suggestions obtained from either active or passive bucket expansion, the user is given a set of new possibly relevant results, and is given the option to add these to existing buckets. See Fig. 3 for a screenshot of how this is embedded within MediaTable.

D. Sampling and Learning

The sampling engine selects samples from subsets of the collection, based on the current contents of the user defined buckets. In order to use the bucket list to the fullest, users are encouraged to split difficult queries into simpler ones. By dividing any concept into several subconcepts, the content of each bucket becomes visually more coherent, thus on-line learning will be more successful. The union of the results provides the final answer.

As an example, consider the query “vehicle visible”. There are many types of vehicles, which are visually distinctive. By creating buckets for “cars”, “bicycles”, “buses”, and “other”, and then categorizing the collection into these subsets we get

Fig. 3. An overview of the MediaTable interface, currently showing approximately 40 out of 36000 shots, which totals approximately 200 hours of video. Each filled cell indicates the estimated presence or absence for each particular detected concept in each video shot. The colors in the table and in the overview on the right indicate in which bucket(s) each particular shot is currently categorized. The bucket list shows the current categorization. The active buckets here form only a minor detail in this interface, just a number and a button at the top of each bucket when new results are available. The impact however is large as this allows users to iteratively refine and extend the contents of each bucket.

buckets for which the contents is visually more related, and bucket expansion will improve.

Next to splitting queries into subclasses, putting obviously irrelevant material into a bucket is also useful, as this removes these from the to be searched collection and reduces the search space for the learning engine. When active bucket learning is triggered the sampling engine takes the contents of bucket B_Q as positive examples, and a random subset of the $B_{everything}$ bucket is taken as unrelated. Because there usually are significantly less relevant items than there are items in total there is a low probability that this subset contains items relevant to the selected bucket. Note that we could take the explicitly defined non-relevant as negative examples, but in many cases these would not form a representative sample of the whole collection.

For our approach we shall see the learning algorithm as a "black box", which can be replaced by other learning algorithms depending on the chosen dataset or available compute power. As such, we assume no knowledge of the underlying visual features of each item in the collection, and intelligent sampling of buckets is not possible. If the descriptors are known better sampling strategies can be selected which yield up to 50% error reduction [21], but such techniques are classifier dependent.

To keep speed of processing within the requirements of the active buckets algorithm, a sample ρ of B_Q is taken to ensure that only a limited number of items is being used:

$$\begin{aligned} i+ &= \rho_l(B_Q) \\ i- &= \rho_l(B_{everything}) \end{aligned}$$

with $i+$ and $i-$ denoting the selected sets of positive and negative samples, and l being the limit on samples taken.

Next, we use a learning algorithm to train a model for these specific sets. This model is then applied to the collection to obtain a re-ranking of the whole collection, together with confidence scores for each item. The scoring function is denoted by:

$$S_{a,b}(x) \in [0, 1]$$

corresponding to a model trained on a relevant set of examples a and irrelevant examples b . The score function gives a measure for the estimated presence/absence of a concept in video fragment x .

These shots and their scores are then sent back to the active bucket engine, where the list is first pruned for any shots already categorized. Next, the similarity score for each item as obtained from the learning engine is compared to a preset threshold t . Thus, the set of additional potentially relevant elements R_i is given by:

$$R_i = \{x \in X | S_{i+,i-}(x) \geq t\}.$$

Elements in R_i are marked as candidate results for bucket B_i , which flags them as "new and relevant" in the interface. The user is then notified with a visual marker at the top of the processed bucket, indicating the number of new candidates found.

Fig. 4. The four tasks for the interactive user experiment, together with a few typical examples of relevant shots within the collection and a description of the total set.

III. EVALUATION

A. Dataset and Tasks

To provide insight into whether bucket expansions yield a benefit for users, we set up an experiment using 21 non-expert participants. We used a dataset of 200 hours of video provided by TRECVID [22], split into 35766 individual shots, with 57 associated metadata aspects containing semantic annotations ranging from *wildlife* to *face* to *people marching*, all automatically extracted using the MediaMill Semantic Video Search Engine content analysis framework [23]. As we focus on high recall we do not use the TRECVID topics but define our own based on the four types of retrieval tasks as specified in Section II. See Fig. 4 for an overview. For each task the user has five minutes to split the dataset into two sets, one containing all shots related to the topic, and one containing the rest. This is in contrast to existing TRECVID interactive search tasks which require the user to find a ranked list of results pertaining to a specific topic. As such, we skipped retrieval task type A, since this type can also be easily performed using existing high precision search engines. Each participant was asked to perform all these four search tasks on one variant of the system. The tasks are specified such that we have one task where the ground truth results contains visually similar items (1), two tasks with items that are somewhat more visually diverse, but have a good entry point within the collection by use of a specific concept detector (2 and 3), and one task which requires combination of several detectors (4).

B. Learning Algorithms

As indicated, in our implementation, we keep the specific learning algorithm a "black box". However, a core consideration for the choice of algorithm is its processing time. Processing does not have to be interactive, e.g., less than two seconds, since the user is not actively waiting for results, but the longer the process takes the higher the chance that the user will no longer be interested in additional results. Furthermore, because the user is not actively waiting for results he will not expect any, and high precision for any iterative loop is not crucial. So we consider two methods, one expensive but accurate, one simple and fast but less accurate.

As a fast method we use a Nearest Mean Classifier which takes the mean position of elements in B_Q , and takes the distance from that point to the rest of the collection. These are then ranked, and the resulting ranking is returned to the user. If we use a pre-computed matrix of inter-shot distances stored on disk this can be done on the fly on a single machine with typical speeds of less than 0.5 seconds on the dataset specified above. This algorithm does not take negative examples into account. When the set of relevant items is visually diverse, we expect the algorithm to have limited accuracy.

A more expensive method is the Support Vector Machine learner. This learner uses the relevant and irrelevant examples to train a Support Vector Machine model [19], using the procedure as described in [23] on 4000 dimensional feature vectors. Such techniques are often used for active learning or relevance feedback systems within the Content Based Image Retrieval domain [18], [24]–[26]. One of the major noted downsides of using standard SVMs in interactive learning systems is its lack of speed or scalability [24]–[26]. Because speed is essential here, we follow [23] and have adapted the SVM kernel to use the same pre-computed kernel matrix of inter-shot distances as with the Nearest Mean classifier, but now kept in the distributed memory of the cluster. This allows new models to be trained at near interactive speeds. A typical training run on a cluster with 4 nodes, a model with 200 examples ($l = 100$) and applying it to the whole dataset specified above, takes about 2 to 6 seconds. Based on the worst case learning time, we estimate an upper limit of 10 bucket updates per minute, though in practice we noticed users averaging between 4 to 6 bucket updates per minute, mostly because the bucket queue was sometimes empty.

Algorithm Comparison

We compared the SVM kernel and the Nearest Mean classifier using automated evaluation based on available ground truth for the dataset. We are interested in seeing how many new relevant results each learner can find given a preset number of relevant items for a specific search task. We define "new" here as the number of items which the user has not previously categorized as relevant, and which are easily accessible to the user in the

Fig. 5. Boxplot results of the learning algorithm experiment. Each graph shows the average number of results found given a specific number of example images for a variety of topics. The input images were chosen randomly, and each experiment was repeated 10 times. Results indicate that a larger number of input images yield a more consistent number of new relevant images for the SVM kernel, and topics which contain visually related materials (marked with a V) usually yield more new relevant items for both engines.

interface. We have set up the following experiment on a set of search topics based on TRECVID [22]:

loop T through each search topic :

let N loop through 2, 5, 10, 15, 20, 25 :

repeat 10 times :

take N random relevant items for T

submit these to both engines

rank results

compute # of relevant items in top K

average results for T, N combination

For our experiment we set $K = 50$ as the number of items a user would review in a glance, typically this is one or two screenfuls of items. We show results of this experiment in Fig. 5. Results indicate that when more relevant items are submitted, SVM will consistently obtain more new relevant items. However, although less accurate the Nearest Mean classifier does allow the user to find new results, so the user does not get stuck during search. In situations where there is limited computing power available, this can be enough. We also observe that when only a limited number of relevant items is available, the engines vary in result quality. For the active bucket engine this does not matter, since this processing was done in the background without knowledge of the user. If there

are new relevant results the user will select them, if there aren't, there isn't much harm done.

For the following interactive user experiments, we use the SVM kernel because of its better results. To keep response times at interactive levels, we needed a small cluster to provide 16 GB of memory in total. For automatic active buckets, we determined the notification threshold as defined in Section II-D by analyzing results at various thresholds, and set this at $t = 0.8$ for the user experiments.

C. Interactive Experiment Setup

We split the users randomly into 3 groups. Each group worked with a different variant of the system, specified as follows: (see Fig. 2)

- **Baseline:** this system did not include any form of bucket expansions. All categorization had to be performed by manually placing relevant shots into the result bucket.
- **Passive bucket expansion:** this variant did allow users to manually initiate bucket expansions on any bucket, but did not have any automation suggesting other possibly relevant expansions.
- **Automated bucket expansion:** for this variant the entire active bucket engine was enabled, but without the ability to manually initiate bucket expansions. The system notified the users when new potentially relevant fragments were available for any bucket. Users were free to use these as they saw fit or simply ignore them.

Fig. 6. Scatter clouds of all results per task. Each dot indicates the number of results found versus the steepest increase with which users were able to find these results. Blue indicates users using no bucket expansion techniques. Green indicates users that have fully automated bucket expansions, and red indicates users that were able to use passive bucket expansions. We have highlighted areas of interest with ellipses.

Fig. 7. Plot of all individual results from the experiment above. Each graph shows how many results were selected (on a scale 0..80) during the entire task (0..5 minutes). Each row indicates all results for a single task. Each column indicates all tasks for a single user, ordered by the average number of results found. The color of a graph indicates which variant of the system the user used. The grey part in each graph shows amount of user activity during each task, and the small stripes on top of each graph indicate when bucket expansions occurred. This graph shows that users using bucket expansions find more results than users without. Also, users using automatic bucket expansions on average find more than users with only passive bucket expansions, and are more consistent with the number of items found.

In order to measure the benefit of active buckets for users, we measured user interactions for one search task over a period of 5 minutes. We asked users to select only results relevant to the task, and place these in user defined buckets. We verified this after each experiment, and filtered out erroneous results. Our evaluation criterium was to measure the number of correct items placed in the buckets as a function of time. During this process, the history of each shot was tracked: when and where was it first seen, when was it placed in a bucket, etc. This gives us an indication of how users found the results. By then aggregating this for all users and showing this over time for the various system variants, we can determine how active buckets influenced this procedure.

D. Interactive Experiment Results

We have summarized all results in Fig. 6, which plots the number of results found together with its steepest increase in number of results found, a measure which summarizes whether

sudden increases in number of results found occurred during the search process. From this, we see the following patterns emerge:

- **Task 1:** For topics with a high visual similarity we find a clear and definite advantage for users using active buckets: most of the users were able to select available results in one go, which cause most of the vertical jumps in Fig. 7. This shows a typical pattern for retrieval of clusters with high visual similarity: as soon as a few visual examples are found, the active bucket strategy provides the rest of the cluster.
- **Task 2:** Active buckets allow users to find more results at once, but on average the same amount of total results. For both task 2 and 3 there was a direct link to a concept possible, which aided the users using no bucket expansions.
- **Task 3:** Clear difference between manually triggered passive bucket expansions and automatic expansions. Automated bucket expansions on average found a much lower number of shots at once. However, on average they did find more shots than users using manual bucket expansions.

Task	Baseline	Manual	Automatic
1	13 ± 7	36 ± 19 *	46 ± 8 * °
2	24 ± 14	23 ± 19	32 ± 8 * °
3	27 ± 24	31 ± 18	53 ± 18 * °
4	13 ± 8	19 ± 7 *	28 ± 5 * °
Average	20 ± 16	27 ± 18 *	40 ± 15 * °
Participants	8	7	6

Fig. 8. Average number of items found together with its standard deviation for each task for each of the 3 system variants; no active buckets, passive bucket expansion and automated bucket expansion. * = statistically significant change at $p = 0.01$ with Welch's t-test compared to the baseline, ° = statistically significant compared to passive buckets.

- **Task 4:** When there is no clear starting point active buckets show a clear benefit. Users without bucket expansions needed to manually combine and filter from two different concepts, *black and white* and *airplane*. Users using fully automated bucket expansions had the most gain here, they only got notified when the system found relevant results whereas some passive bucket expansion users made a couple of wrong guesses about what to expand.

Fig. 7 shows these results in more detail, and Fig. 8 shows a numerical summary. If we look at the results from Fig. 7, we see three distinctly different types of result gathering:

- **No results found.** Appearing in the graph as a horizontal line. This is seen for many users without any bucket expansions, and some users using passive buckets. This indicates that users were unable to find any results during the task.
- **Gradual addition of results.** Appearing in the graph as approximately straight lines which have an upward slope. This is mostly seen for users using automated active buckets. This indicates a steady increase of relevant results over time.
- **Sudden fast increases in results found.** Seen as vertical jumps in the graphs, which are visible at many users using passive bucket expansions, and some using active buckets. This pattern typically occurs when users were able to find sets of 50–60 relevant results at a time. This pattern occurs most often in task 1, which had a series of highly similar results.

The graphs indicate that more results are retrieved when any form of bucket expansion was available: most of the red and green graphs are to the right in the sorted list.

When looking at individual users, i.e., the columns, we see a number of patterns.

Users which had no active buckets (the blue entries in the graphs of Fig. 7) generally tended to retrieve fewer or no results, showing horizontal lines (users 0 and 1) and slope patterns, including a few jumpy slope patterns (users 9 and 10), indicating users selecting whole subsets at once. Note that all users in this experiment were non-expert users, with limited experience in performing video retrieval tasks, so being unable to find anything in a five minute time frame is not uncommon.

We also measured the number of user interactions during each completed task. Each mouse click, keyboard press, drag select, or continuous mouse scroll wheel action counts as one action. Fig. 9 shows the average number of actions for each task per system type, and Fig. 6 shows the individual scores per user. We found that users with fully automated bucket expansions were

Task	Baseline	Passive	Automatic
1	181 ± 68	171 ± 59	115 ± 49
2	143 ± 42	125 ± 45	143 ± 50
3	139 ± 42	113 ± 36	225 ± 33
4	151 ± 65	127 ± 57	227 ± 66
Average	154 ± 58	134 ± 55	177 ± 71

Fig. 9. Average number of user actions (keyboard presses + mouse clicks) for each task of the 3 system variants; no active buckets, passive bucket expansion and automated bucket expansion.

able to do the most actions during each task, 177 actions/task on average. This also explains the upward slope in the graphs of Fig. 7: users continuously added new relevant items, which in turn allowed the algorithm to find more relevant items. Users with passive bucket expansions performed the least amount of actions in 5 minutes, 134 actions on average. Though the system allowed the users to continue looking for results after starting bucket expansions, we found that most of them choose to do nothing during that time, and wait for the results. Furthermore, these users tended to use drag selects more often to select all relevant results of a bucket expansion at once.

Users using passive bucket expansion (the red entries) typically scored high on 2 tasks, and low on two other tasks, see for example users 6, 7, 8 and 12. Note here that these weren't the same tasks. The cause of this changing behavior was often due to wrong guesses from the users on when to use passive bucket expansion. For example, when buckets contained a subset which did not yield further results within the collection and users tried to expand on it anyway. This indicates that manual activation in the hands of non-expert users alone is not good enough, and has the risk of leading users into bad decisions. Only 2 passive bucket expansion users (16 and 17) yielded consistently high scores for all topics. Users using automated bucket expansion (the green entries) typically scored high on all tasks, with most users showing a steep slope pattern, and some jumps.

Overall results indicate a clear benefit for users using active buckets.

IV. CONCLUSION

In this paper, we have proposed a complete framework for high recall video retrieval using bucket based categorization as an intermediate step. Our Active Buckets Categorization approach automatically finds and suggests potentially relevant shots for individual buckets.

In our evaluation we investigated two learning engines. When sufficient computational resources are available SVM is much more accurate, however the Nearest Mean is still very useful as it does find relevant new results. We evaluated active buckets using 3 groups of non-expert users with various types of active buckets. Results indicate a statistically significant advantage in terms of results retrieved when using active buckets. Furthermore, we found that using unobtrusive automated bucket expansions let the user continue searching while using passive bucket expansions would let the user wait for results, which hampered the overall performance. Automated bucket expansions yielded the highest number of results retrieved, with the user being able to continuously keep interacting with the system to find more results.

The Active Bucket categorization approach could also be combined with high precision interfaces to cover a large part of the spectrum of interactive video retrieval. For example, given a difficult to answer search query, Active Buckets can be used to first segment the dataset into several subsets based on individual easy to answer high recall searches. This allows the user to rule out subsets that clearly have nothing to do with the posed search query. This reduces the search space, and the much smaller remainder can then be searched through with a fast high precision search system such as the XVR system [16], VisionGo [17] or the ForkBrowser [27].

We conclude that unobtrusive observation of user interactions is an effective means of retrieval especially when users can employ intermediate categorization and although we demonstrate this for high recall video retrieval only, it could be equally suitable as a collection categorization or annotation tool.

REFERENCES

- [1] O. de Rooij, M. Worrington, and J. van Wijk, "MediaTable: Interactive categorization of multimedia collections," *IEEE Comput. Graphics Appl.*, vol. 30, no. 5, pp. 42–51, Sep. 2010.
- [2] Flickr [Online]. Available: <http://flickr.com/>
- [3] Youtube [Online]. Available: <http://youtube.com/>
- [4] A. W. M. Smeulders, M. Worrington, S. Santini, A. Gupta, and R. Jain, "Content based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.
- [5] C. G. M. Snoek and M. Worrington, "Concept-based video retrieval," *Found. Trends Inf. Retrieval*, vol. 4, no. 2, pp. 215–322, 2009.
- [6] 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.*, vol. 2, no. 1, pp. 1–19, 2006.
- [7] C. G. M. Snoek, M. Worrington, D. C. Koelma, and A. W. M. Smeulders, "A learned lexicon-driven paradigm for interactive video retrieval," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 280–292, Mar. 2007.
- [8] O. de Rooij and M. Worrington, "Browsing video along multiple threads," *IEEE Trans. Multimedia*, vol. 12, no. 2, pp. 121–130, Mar. 2010.
- [9] A. P. Natsev, M. R. Naphade, and J. Tešić, "Learning the semantics of multimedia queries and concepts from a small number of examples," in *Proc. 13th Annu. ACM Int. Conf. Multimedia*, ACM, New York, NY, USA, 2005, pp. 598–607.
- [10] D. Wang, X. Liu, L. Luo, J. Li, and B. Zhang, "Video diver: Generic video indexing with diverse features," in *Proc. ACM Int. Workshop on Workshop on Multimedia Inform. Retrieval*, New York, NY, USA, 2007, pp. 61–70.
- [11] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Comput. Survey*, vol. 40, no. 2, pp. 1–60, 2008.
- [12] E. Zavesky, S.-F. Chang, and C.-C. Yang, "Visual islands: Intuitive browsing of visual search results," in *Proc. ACM Int. Conf. Content-Based Image Video Retrieval*, New York, NY, USA, 2008, pp. 617–626.
- [13] J. Cao, Y.-D. Zhang, J.-B. Guo, L. Bao, and J.-T. Li, "Videomap: An interactive video retrieval system of mcg-ict-cas," in *Proc. ACM Int. Conf. Image Video Retrieval*, New York, NY, USA, 2009, pp. 1–1.
- [14] M. G. Christel and R. Yan, "Merging storyboard strategies and automatic retrieval for improving interactive video search," in *Proc. 6th ACM Int. Conf. Image Video Retrieval*, New York, NY, USA, 2007, pp. 486–493.
- [15] J. Adcock, M. Cooper, and F. Chen, "Fxpall MediaMagic video search system," in *Proc. of the ACM Int. Conf. Image Video Retrieval*, 2007, pp. 644–644.
- [16] 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 *Proc. 14th Annu. ACM Int. Conf. Multimedia*, New York, NY, USA, 2006, pp. 385–394.
- [17] 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 *Proc. 15th Int. Conf. Multimedia*, New York, NY, USA, 2007, pp. 293–296.
- [18] M.-Y. Chen, M. Christel, A. Hauptmann, and H. Wactlar, "Putting active learning into multimedia applications: Dynamic definition and refinement of concept classifiers," in *Proc. 13th Annu. ACM Int. Conf. Multimedia*, New York, NY, USA, 2005, pp. 902–911.
- [19] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Inf. Syst.*, vol. 2, no. 3, Apr. 2011 [Online]. Available: <http://dl.acm.org/citation.cfm?id=1961199>
- [20] K.-S. Gob, E. Y. Chang, and W.-C. Lai, "Multimodal concept-dependent active learning for image retrieval," in *Proc. 12th Annu. ACM Int. Conf. Multimedia*, New York, NY, USA, 2004, pp. 564–571.
- [21] R. Yan, J. Yang, and A. Hauptmann, "Automatically labeling video data using multi-class active learning," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, 2003, vol. 2003, pp. 516–523.
- [22] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVID," in *Proc. 8th ACM Int. Workshop Multimedia Inf. Retrieval*, New York, NY, USA, 2006, pp. 321–330.
- [23] C. G. M. Snoek, K. E. A. van de Sande, O. de Rooij, B. Huurnink, J. C. van Gemert, J. R. R. Uijlings, J. He, X. Li, I. Everts, V. Nedovic, M. van Liempt, R. van Balen, F. Yan, M. A. Tahir, K. Mikolajczyk, J. Kittler, M. de Rijke, J.-M. Geusebroek, T. Gevers, M. Worrington, A. W. M. Smeulders, and D. C. Koelma, "The mediamill TRECVID 2008 semantic video search engine," in *Proc. 6th TRECVID Workshop*, Gaithersburg, MD, USA, Nov. 2008.
- [24] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in *Proc. 9th ACM Int. Conf. Multimedia*, New York, NY, USA, 2001, pp. 107–118.
- [25] P. H. Gosselin and M. Cord, "A comparison of active classification methods for content-based image retrieval," in *Proc. 1st Int. Workshop Comput. Vision Meets Databases*, New York, NY, USA, 2004, pp. 51–58.
- [26] M. Cord, P. H. Gosselin, and S. Philipp-Foliguet, "Stochastic exploration and active learning for image retrieval," *Image Vis. Computing*, vol. 25, no. 1, pp. 14–23, 2007.
- [27] O. de Rooij, C. G. M. Snoek, and M. Worrington, "Balancing thread based navigation for targeted video search," in *Proc. ACM Int. Conf. Image Video Retrieval*, Niagara Falls, ON, Canada, Jul. 2008, pp. 485–494.

Ork de Rooij received his M.Sc. degree in Artificial Intelligence from the University of Amsterdam, and is pursuing the Ph.D. degree in computer science at the University of Amsterdam, The Netherlands. He is currently employed as a researcher at the same university. His research interests cover text mining, information visualization, user computer interaction, and large-scale content-based retrieval systems.

Marcel Worrington received the M.Sc. degree (honors) in computer science from the VU Amsterdam, The Netherlands, in 1988 and the Ph.D. degree in computer science from the University of Amsterdam in 1993. He is currently an Associate Professor in the Informatics Institute of the University of Amsterdam. His research focus is multimedia analytics, the integration of multimedia analysis, multimedia mining, information visualization, and multimedia interaction into a coherent framework yielding more than its constituent components. He

has published over 150 scientific papers covering a broad range of topics from low-level image and video analysis up to multimedia analytics. Dr. Worrington was co-chair of the 2007 ACM International Conference on Image and Video Retrieval in Amsterdam, co-initiator and organizer of the VideOlympics, and program chair for both ICMR 2013 and ACM Multimedia 2013. He was an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA and the Pattern Analysis and Applications journal.