Submodular Video Hashing: A Unified Framework Towards Video Pooling and Indexing

Liangliang Cao
Multimedia Group
IBM Watson Research Center
liangliang.cao@us.ibm.com

Zhenguo Li, Yadong Mu, Shih-Fu Chang
Dept. of Electrical Engineering
Columbia University
zgli,muyadong,sfchang@ee.columbia.edu

ABSTRACT

This paper develops a novel framework for efficient large-scale video retrieval. We aim to find video according to higher level similarities, which is beyond the scope of traditional near duplicate search. Following the popular hashing technique we employ compact binary codes to facilitate nearest neighbor search. Unlike the previous methods which capitalize on only one type of hash code for retrieval, this paper combines heterogeneous hash codes to effectively describe the diverse and multi-scale visual contents in videos. Our method integrates feature pooling and hashing in a single framework. In the pooling stage, we cast video frames into a set of pre-specified components, which capture a variety of semantics of video contents. In the hashing stage, we represent each video component as a compact hash code, and combine multiple hash codes into hash tables for effective search. To speed up the retrieval while retaining most informative codes, we propose a graph-based influence maximization method to bridge the pooling and hashing stages. We show that the influence maximization problem is submodular, which allows a greedy optimization method to achieve a nearly optimal solution. Our method works very efficiently, retrieving thousands of video clips from TRECVID dataset in about 0.001 second. For a larger scale synthetic dataset with 1M samples, it uses less than 1 second in response to 100 queries. Our method is extensively evaluated in both unsupervised and supervised scenarios, and the results on TRECVID Multimedia Event Detection and Columbia Consumer Video datasets demonstrate the success of our proposed technique.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

MM'12, October 29–November 2, 2012, Nara, Japan.

Keywords

video hashing, indexing, feature pooling, multiple feature hashing, submodular

1. INTRODUCTION

The power of word-based document indexing has been demonstrated by the success of modern text-based search engines like Yahoo, Google and Bing. Nowadays billions of users can easily enter search terms to query enormous repositories of text documents in a flexible way. In contrast, video retrieval is still in its early stage: most video search systems still rely on textual titles or comments rather than visual features. One reason is that it is still very expensive to compute distance metrics between video queries and tremendous video corpus on the Web. Consequently, personalized video search where users can impose individual preference of visual features is still far from being a reality. On the one hand, more video contents are being generated. On the other hand, it appears to be more difficult for effectively retrieve relevant contents.

The difficulty in large scale video indexing stems from the complex nature of video contents. Unlike text retrieval where a document can be retrieved by words in constant time using inverted index structure, a video is composed of multiple frames with dynamic backgrounds and evolving people activities. The semantic gap between low-level features and high-level concepts makes it difficult to model rich video semantics. Additionally, data structures based on inverted index files are not readily applicable for content-based retrieval of videos which are represented by high-dimensional visual features.

To capture the diversity of video contents, a straightforward approach is to employ multiple video features to describe different aspects of visual contents. However, there are two challenges of using multiple features. First, most multiple feature learning algorithms rely on supervised learning. While more features may capture sufficient video contents, non-discriminative or noisy features may degrade the performance substantially. Second, employing multiple features increases the computational complexity and makes large-scale visual retrieval slower. There is a trade-off between the quality of the retrieval and the corresponding efficiency. If the feature dimensionality is too high, nearest neighbor search will be expensive and undiscriminative due to the curse of high dimensionality [9].

In this paper, we propose a new framework, called *submodular video hashing*, for efficient video retrieval with large-scale feature selection. The approach achieves fast retrieval

speed and scales well to very large video data sets. The contributions of this new framework lie in three aspects:

- It captures the diversified contents of video frames using novel visual pooling methods in the temporal domain. By pooling the frames with similar visual features, it can distinguish different scenes in a video and group them into different video components. The temporal pooling method is especially useful in modeling long sequences or videos with multiple scenes.
- It develops a new selection strategy from multiple scene components and multiple features. To handle the trade-off between efficiency and accuracy, we prove that our feature selection formulation is submodular, and thus can be solved nearly optimally by greedy optimization.
- Finally, it uses multiple hash tables to combine the results from selected features and video components. Our hash tables work similarly to inverted indexes in text retrieval and support video retrieval with very low computational complexity.

Note that the objective of submodular video hashing is different from near-duplicate search and image hashing. Near-duplicate video search is a classical problem in multimedia field, which aims to find similar frames among videos, and is very useful in protecting copyright and intelligence property. In contrast, our submodular video hashing, can be used to find videos according to higher level similarities, such as video styles or topics. Image hashing has been a popular topic in recent years, but it does not consider the temporal pooling problem as in this paper. There is also another subtle but practically very important difference between our method and prior image hashing techniques. A number of prior efforts on image hashing [33] [34] require computation of the Hamming distances between the query and images in the datasets, which implies huge computational cost for large-scale video datasets. Following the paradigm in Indyk *et al.*'s seminal work [8], this paper utilizes hashing table as inverted indexes to enable very efficient retrieval and develops a novel submodular framework for combining multiple hash tables.

The rest of this paper is organized as follows: Section 2 first reviews related work on image pooling and submodular optimization. Section 3 generalizes previous studies by extending pooling from image domain to video domain. The proposed pooling technique comprises multiple components that can provide a comprehensive description of video contents. Based on these video components, Section 4 develops a novel submodular video hashing strategy. Section 5 reports experimental results using submodular video hashing techniques that show excellent performance. Finally we conclude this paper in Section 6 and report on future directions.

2. RELATED WORK

Our work is partly motivated by the recent progress in image hashing. Shaknarovich and Darrell [31] proposed BoostSSC to compute a weighted Hamming distance. However, the weighted Hamming distance makes it difficult to find the neighbors efficient in their approach. Torralba et al. used small hash codes to represent 80 million tiny images [33]. Weiss et al. developed spectral hashing [34] to improve the hashing quality using spectral graph theory. Liu et al. employed anchor nodes to speed up the hashing process on large-scale graphs [26]. Hashing with supervised information is also considered in recent studies [23], [25]. Zhang et al. [38] proposed to integrate information from several different sources by adjusting the weights on each individual source for maximizing the coding performance, and enabling fast conversion from query examples to their binary hash codes. However, all the above approaches adopt just a single hash code to represent an instance, which cannot capture the diversified content of video clips.

In contrast to previous studies in image hashing, this paper proposes to represent a video using multiple components produced by temporal video pooling. The pooling operation has been widely used in image recognition. Boureau et al. argued in [2] that many recent progresses in image recognition can be viewed as a combination of alternating series of coding and spatial pooling steps. Average pooling [20] tries to compute the mean of the feature vectors within a spatial neighborhood. Bag-of-words model can be viewed as a special case of average pooling. Max pooling is found to be useful for sparse coding features [37]. This idea is recently generalized in [1], by enlarging the spatial bins in feature space clustering. Video pooling is different from image pooling in the sense that the regularity in temporal domain is not as significant as that in the spatial domain. A video event can either happen at the beginning part of the video or the second half of the video, or it can last throughout the whole sequence.

It is worth noting that this paper is also different from classical near-duplicate video search problem that are chiefly motivated by the problem of copyright detection. Many of the proposed solutions decompose videos into multiple key-frames and look for similar key-frames across pairs of videos [16], [35], [15], [12], [32]. Some recent work considers pair-wise distances of key-frames [24], which can further improve the detection accuracy but suffers from heavy computational burden. This prior work does not generalize for content-based video retrieval because: (1) General video retrieval requires support for content-based search based on high-level semantics rather than visual duplication. (2) Traditional near-duplicate search requires a distance measure between key-frames, and the performance is determined by the quality of low-level features of key-frames in unsupervised learning scenario. In contrast, a single low-level feature may not work well for general video retrieval, which often requires supervised learning. (3) Video retrieval systems are desired to support personalized retrieval, where user profile or retrieval history improves retrieval performance. This cannot be accomplished by classical near-duplicate retrieval.

This paper employs hash tables as the basic data structure and develops a novel method to construct multiple hash tables. The contribution in this paper is in developing a unified submodular framework that integrates video pooling and hashing, and provides an efficient strategy exploring the rich contents in video retrieval. In recent years, submodular optimization has emerged as a powerful optimization tool in a variety of machine learning problems including active learning, structure learning, clustering, and ranking. A submodular function is essentially characterized by a diminishing return property which states that, the marginal gain of adding a new element to a subset B is higher than that of adding it to any larger subset A, where $B \subset A$. Formally, let P be the power set of a set $\mathbb{V}$. A set function $h : P \rightarrow \mathbb{R}$

is said to be submodular [18] if

$$h(\mathbb{A}\cup\{v\})-h(\mathbb{A}) \leq h(\mathbb{B}\cup\{v\})-h(\mathbb{B}) \quad \forall \mathbb{B} \subset \mathbb{A} \subset \mathbb{V}, v \in \mathbb{V}\backslash\mathbb{A}. \tag{1}$$

The very desirable consequence of submodularity is that provably (near-)optimal solutions can be found via efficient greedy optimization [18]. Some typical submodular functions explored in machine learning include set cover, influence in social network [21], electricity network [18], anisotropic diffusion [17], and sparse representation [6]. In this paper, we will prove that our hash code selection is also a submodular problem and can be solved by an efficient greedy approach.

3. VIDEO POOLING

The basic idea of our video hashing algorithm is to exploit the rich semantics of videos and construct binary codes corresponding to different characteristics of video contents. Our methods contrast with previous image based hashing methods by considering video feature pooling in frame-level.

Suppose there is a video with T frames, where each frame is represented by a feature vector x_t. We generate temporal pooling component k as

$$z^k = \sum_{t=1}^{T} p_t^k x_t. \tag{2}$$

Here p_t^k stands for the probability $Pr(k|x_t)$ that captures the amount of contribution of frame t in forming the scene component k. We approximate the probability using soft vector quantization on GIST features.

From (2), we can see that to obtain different scene component $z^k = \sum_{t=1}^{T} p_t^k x_t$, in fact is equivalent to decompose each frame feature x_t into different scenes

$$x_t \rightarrow [p_t^1 x_t, p_t^2 x_t, \cdots], \tag{3}$$

From eqs (2) and (3) one can see that our new model is a feature pooling method [1]. The unique characteristic of our method is that our pooling weights p_t^k are on video frames instead of spatial domain. Here we force the pooling weights p_t^k to follow the constraint $\sum_k p_t^k = 1$. We will explain how to compute p_t^k in the following.

Following the previous studies [29] [30] [36], we use GIST features to represent the scenes of video frames and then to compute p_t^k. Note that the pooling weights are computed based on Euclidean distance, which is not reliable for sparse histogram features like SIFT. On the other hand, GIST is easy to compute and gives a reliable measure on how different two frames look holistically [29]. With all the training data, we compute the GIST features for all frames and cluster them into C centers using the K-means algorithm. The C center is represented as $g_c^1, g_c^2, \cdots, g_c^C$. To compute the pooling weight p_t^k for frame t in a training or test video clip, we extract it GIST feature vector g_t . The pooling weight is derived by comparing $\{g_c^k\}$ and g_t.

One simple way to compute pooling weights is by vector quantization (VQ), which forces all but the nearest one p_t^k to be zero. However, VQ is well-known to be sensitive to noises. In this paper, we consider a soft voting pooling weights as

$$p_t^k = \frac{1/(d_t^k)^3}{\sum_{l=1}^{C} 1/(d_t^l)^3}, \tag{4}$$

Algorithm 1 Video Pooling Algorithm.

Input: A video with the set of frame feature vectors $\{x_t\}$, $1 \leq t \leq T$. A codebook with C centers of GIST feature g_c^k, $1 \leq k \leq C$.

Extract GIST descriptor g_t for every frame t.
Compute the pooling weight p_t^k using eq. (4).
Compute scene component z^k using eq. (2).
Normalize z^k and encode each component with a hash code.

where d_t^k is the Euclidean distance between centers g_c^k and frame feature g_t. It is not difficult to see that our pooling weights satisfy the constraint $\sum_k p_t^k = 1$. Also our pooling weights rely on GIST feature, regardless of the event label y. This means we will use the same pooling strategy independent with events. We call the method in eq. 4 as concurrent VQ. Compared with traditional VQ, our new pooling method penalizes the impact of large outliers, and assigns more weights to the centers closer to g_t.

Note that this paper chooses an unsupervised way to model pooling weights. The reasons why unsupervised learning is preferred are as follows: First, unsupervised learning can save the extra labeling efforts; On the other hand, in our model, the goal is not to recognize the exact scene category but to do pooling according to scene context, and thus supervised learning is not necessary. Algorithm 1 summarizes the workflow of our video pooling algorithm.

Our recent experiments show [4] that Support Vector Machine (SVM) classifiers with video pooling components can significantly improve the recognition accuracy for complex video events. However, this paper is different from [4] in the fact that it considers the video hashing problem (as elaborated in the next section) instead of SVM classification. After temporal pooling components $\{z^k\}$ are computed, we further learn a compact hash code (less than 32 bits) for each pooling component. We generate a number of hash codes using different temporal pooling. We use both random projection [8] and spectral hashing [34] in this paper, but note that our system is also flexible to be combined with other hashing functions.

4. SUBMODULAR VIDEO HASHING

4.1 Model

Suppose we have n_f features indexed by $f = 1, \ldots, n_f$, where each (f) is captured by multiple (C) components $(v_f^1, \cdots, v_f^C)$ and each component (v_f^j) is expressed by a compact binary hash code[1]. Our first observation is that not every code is equally informative and there can be significant redundancy among the codes. On the other hand, it is critical to use compact representation for large-scale video retrieval. Inspired by these observations, we aim to select a *small* number of "informative" codes which can well "represent" the entire set of codes. We found these concepts can be effectively modeled using a graph over the codes whose edge weights capture the degree of overlapping or similarity between pairwise codes. In this section, we assume that the mechanism to compute pairwise code similarity is known (which will be addressed in the next subsection).

[1]Our model applies to any other feature representation.

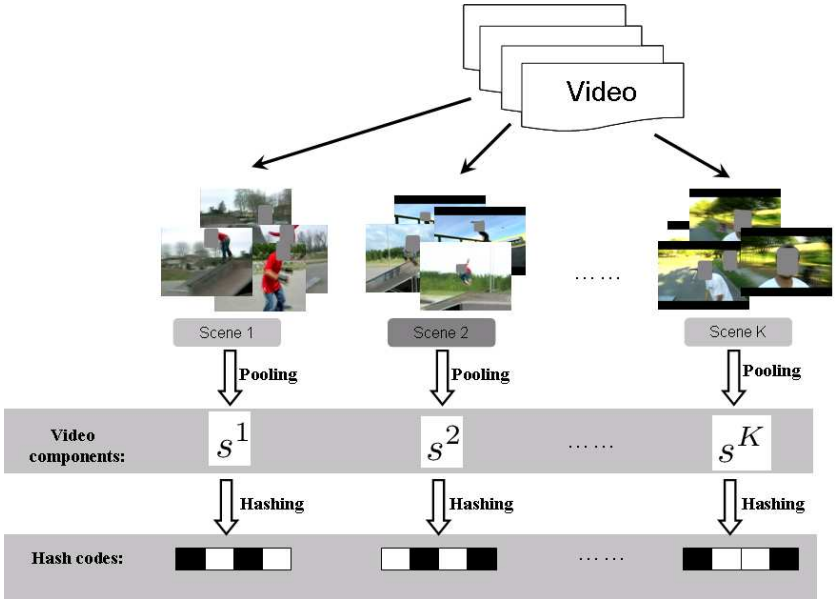

Figure 1: Temporal pooling for video hashing.

Let $\mathcal{G} = \{\mathbb{V}, W\}$ be a graph with node set $\mathbb{V}$ consisting of the hash codes considered, i.e., $\mathbb{V} = \{v_f^k | k = 1, \ldots, C; f = 1, \ldots, n_f\}$, and similarity matrix $W = [w_{ij}]$ where w_{ij} captures the similarity between codes i and j. Our goal is to select m representative nodes from the graph, where m can be manually specified or determined automatically. We cast this node selection problem as an influence maximization problem where the influence of the selected m nodes can be *maximally* propagated to the rest of the nodes in the graph.

Denote $\mathbb{A}$ as the set of selected nodes whose influences are assigned and fixed to 1's. We use a score $u_{\mathbb{A}}(i) \in [0, 1]$ to quantify the influence node i received from $\mathbb{A}$. For technical reason that will be clear soon, we introduce a sink node s to the graph that is connected to each node with a small constant weight. The sink node s is very "cool" in that it is never influenced by others or tries to influence others. So its influence is fixed to 0. For simplicity and a little abuse of notations, we still denote the graph as $\mathcal{G} = \{\mathbb{V}, W\}$ but keep in mind that $s \in \mathbb{V}$ and W is expanded accordingly. We also denote $\hat{\mathbb{A}} = \mathbb{A} \cup s$ and $\mathbb{N}$ the set of remaining nodes, i.e., $\hat{\mathbb{A}} \cap \mathbb{N} = \emptyset$ and $\hat{\mathbb{A}} \cup \mathbb{N} = \mathbb{V}$. Formally, we propose the following influence maximization framework:

$$\max_{\mathbb{A} \subset \mathbb{V}} \Omega(\mathbb{A}) := \sum_{i \in \mathbb{N}} u_{\mathbb{A}}(i). \tag{5}$$

To instantiate the above framework, a concrete model for influence propagation is needed. Since label propagation on graph is a well studied problem in machine learning [39], we adapt it for influence, though other propagation techniques can be employed [3] [13]. Specifically, we use the concept of Harmonic fields from Zhu et al. [40] which has been shown to be a highly effective label propagation model.

Denote $u_{\mathbb{A}}(i) = 1$ if $i \in \mathbb{A}$ and $u_{\mathbb{A}}(s) = 0$. Our influence model enjoys a "harmonic" interaction among the nodes as follows:

$$u_{\mathbb{A}}(i) = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} w_{ij} u_{\mathbb{A}}(j), \ i \in \mathbb{N} \tag{6}$$

where $\mathcal{N}(i)$ denotes the set of neighbors of node i and $d_i = \sum_{j \in \mathcal{N}(i)} w_{ij}$ is the degree of node i. In other words, *the influence on node i is the weighted sum of the influences on its neighbors.*

Recall that our objective is to select m nodes $\mathbb{A}$ to maximize the overall influence $\Omega(\mathbb{A})$, which appears to be a difficult discrete optimization problem. However, we will show below that set function $\Omega(\mathbb{A})$ is *submodular* (see eq. (1)), whose optimization can be done in a greedy manner which is guaranteed to converge to a solution that is nearly optimal [28]. Our strategy is partially inspired by previous work in semi-supervised learning [40] and image co-segmentation [17]. Our work differs in a new formulation for hash function selection as well as a novel and easier way for showing the submodularity of the influence maximization problem.

We introduce some notations first. Denote

$$W = \begin{bmatrix} W_{\hat{\mathbb{A}}\hat{\mathbb{A}}} & W_{\hat{\mathbb{A}}\mathbb{N}} \\ W_{\mathbb{N}\hat{\mathbb{A}}} & W_{\mathbb{N}\mathbb{N}} \end{bmatrix}.$$

Let $\mathbf{u}_{\hat{\mathbb{A}}}$ and $\mathbf{u}_{\mathbb{N}}$ be the two vectors corresponding to the influences on $\hat{\mathbb{A}}$ and $\mathbb{N}$, respectively. Then (6) becomes

$$D_{\mathbb{N}\mathbb{N}} \mathbf{u}_{\mathbb{N}} = W_{\mathbb{N}\hat{\mathbb{A}}} \mathbf{u}_{\hat{\mathbb{A}}} + W_{\mathbb{N}\mathbb{N}} \mathbf{u}_{\mathbb{N}}, \tag{7}$$

where $D_{\mathbb{N}\mathbb{N}}$ is the diagonal matrix corresponding to degrees of nodes in $\mathbb{N}$.

Below we will show that $\Omega(\mathbb{A}) = \sum_{i \in \mathbb{N}} u_{\mathbb{A}}(i)$ is submodular. Since any non-negative linear combination of submodular functions is also submodular, it suffices to show $u_{\mathbb{A}}(i)$ is submodular w.r.t. $\mathbb{A}$ for any $i \in \mathbb{N}$.

PROPOSITION 1. *The matrix $L_{\mathbb{N}\mathbb{N}} := D_{\mathbb{N}\mathbb{N}} - W_{\mathbb{N}\mathbb{N}}$ is positive definite and $L_{\mathbb{N}\mathbb{N}}^{-1}$ is non-negative.*

PROOF. Denote $D_{\mathbb{N}} = \text{diag}(W_{\mathbb{N}\mathbb{N}}\mathbf{1})$ where $\mathbf{1}$ is a vector of one of appropriate size. Then $\hat{D} = D_{\mathbb{N}\mathbb{N}} - D_{\mathbb{N}}$ is a diagonal matrix whose every diagonal element is no less than a. Here a denotes the weight between the sink node and each of the other nodes. So $L_{\mathbb{N}\mathbb{N}} = \hat{D} + L_{\mathbb{N}}$ where $L_{\mathbb{N}} := D_{\mathbb{N}} - W_{\mathbb{N}\mathbb{N}}$ is

known to be positive semi-definite [5]. Thus $L_{\bar{\mathbb{N}}\bar{\mathbb{N}}}$ is positive definite.

Note that to show $L^{-1}_{\bar{\mathbb{N}}\bar{\mathbb{N}}}$ is non-negative is equivalent to showing $L := D^{1/2}_{\bar{\mathbb{N}}\bar{\mathbb{N}}} L^{-1}_{\bar{\mathbb{N}}\bar{\mathbb{N}}} D^{1/2}_{\bar{\mathbb{N}}\bar{\mathbb{N}}}$ is non-negative. However, $L = (I-C)^{-1}$ where $C = D^{-1/2}_{\bar{\mathbb{N}}\bar{\mathbb{N}}} W_{\bar{\mathbb{N}}\bar{\mathbb{N}}} D^{-1/2}_{\bar{\mathbb{N}}\bar{\mathbb{N}}}$ is a non-negative matrix whose eigenvalues are of magnitude less than 1[2]. Thus $L = \sum_{\ell=0}^{\infty} C^{\ell}$ is non-negative since each component in the summation is non-negative. □

PROPOSITION 2. *$u_{\mathbb{A}}(i) = 0$ if $\mathbb{A} = \emptyset$.*

PROOF. Since $\mathbb{A} = \emptyset$, we have $\hat{\mathbb{A}} = \mathbb{A} \cup s = \{s\}$. So $\mathbf{u}_{\hat{\mathbb{A}}} = 0$. By (7), $L_{\bar{\mathbb{N}}\bar{\mathbb{N}}}\mathbf{u}_{\bar{\mathbb{N}}} = \mathbf{0}$ which yields $\mathbf{u}_{\bar{\mathbb{N}}} = \mathbf{0}$ since $L_{\bar{\mathbb{N}}\bar{\mathbb{N}}}$ is invertible (Proposition 1). □

PROPOSITION 3. *$u_{\mathbb{A}}(i)$ is non-decreasing.*

PROOF. To show $u_{\mathbb{A}}(i)$ is non-decreasing is to show $u_{\mathbb{A}}(i) \leq u_{\mathbb{A}\cup e}(i)$ where $e \in \mathbb{N}$. Denote $\hat{N} = \mathbb{N} \setminus \{e\}$, and write $\mathbf{u} = [\mathbf{u}^{\top}_{\hat{\mathbb{A}}}, u_e, \mathbf{u}^{\top}_{\hat{\mathbb{N}}}]^{\top}$ and

$$W = \begin{bmatrix} W_{\hat{\mathbb{A}}\hat{\mathbb{A}}} & W_{\hat{\mathbb{A}}e} & W_{\hat{\mathbb{A}}\hat{\mathbb{N}}} \\ W_{e\hat{\mathbb{A}}} & W_{ee} & W_{e\hat{\mathbb{N}}} \\ W_{\hat{\mathbb{N}}\hat{\mathbb{A}}} & W_{\hat{\mathbb{N}}e} & W_{\hat{\mathbb{N}}\hat{\mathbb{N}}} \end{bmatrix}$$

When only A is selected, we have

$$D_{\hat{\mathbb{N}}\hat{\mathbb{N}}}\mathbf{u}_{\hat{\mathbb{N}}} = W_{\hat{\mathbb{N}}\hat{\mathbb{A}}}\mathbf{u}_{\hat{\mathbb{A}}} + W_{\hat{\mathbb{N}}e}u_e + W_{\hat{\mathbb{N}}\hat{\mathbb{N}}}\mathbf{u}_{\hat{\mathbb{N}}}.$$

When both $\mathbb{A}$ and e are selected, we have

$$D_{\hat{\mathbb{N}}\hat{\mathbb{N}}}\mathbf{u}'_{\hat{\mathbb{N}}} = W_{\hat{\mathbb{N}}\hat{\mathbb{A}}}\mathbf{u}_{\hat{\mathbb{A}}} + W_{\hat{\mathbb{N}}e}1_e + W_{\hat{\mathbb{N}}\hat{\mathbb{N}}}\mathbf{u}'_{\hat{\mathbb{N}}}.$$

Thus

$$(D_{\hat{\mathbb{N}}\hat{\mathbb{N}}} - W_{\hat{\mathbb{N}}\hat{\mathbb{N}}})(\mathbf{u}'_{\hat{\mathbb{N}}\hat{\mathbb{N}}} - \mathbf{u}_{\hat{\mathbb{N}}}) = W_{\hat{\mathbb{N}}e}(1_e - u_e),$$

yielding

$$(\mathbf{u}'_{\hat{\mathbb{N}}\hat{\mathbb{N}}} - \mathbf{u}_{\hat{\mathbb{N}}}) = (D_{\hat{\mathbb{N}}\hat{\mathbb{N}}} - W_{\hat{\mathbb{N}}\hat{\mathbb{N}}})^{-1}W_{\hat{\mathbb{N}}e}(1_e - u_e), \quad (8)$$

which is non-negative since $(D_{\hat{\mathbb{N}}\hat{\mathbb{N}}} - W_{\hat{\mathbb{N}}\hat{\mathbb{N}}})^{-1}$ is non-negative (Proposition 1) and $u_e \leq 1$ [40]. □

PROPOSITION 4. *$u_{\mathbb{A}}(i)$ is submodular.*

PROOF. To show $u_{\mathbb{A}}(i)$ is submodular is to show that for any $\hat{\mathbb{B}} \subseteq \mathbb{B} \subseteq \mathbb{V}$, the following inequality holds

$$u_{\hat{\mathbb{B}}\cup\{e\}}(i) - u_{\hat{\mathbb{B}}}(i) \geq u_{\mathbb{B}\cup\{e\}}(i) - u_{\mathbb{B}}(i). \quad (9)$$

By (8), $\delta_{\hat{\mathbb{N}}} := \mathbf{u}'_{\hat{\mathbb{N}}} - \mathbf{u}_{\hat{\mathbb{N}}}$ decreases (component-wisely) as u_e is larger (i.e., more nodes are selected), which confirms (9). □

PROPOSITION 5. *$\Omega(\mathbb{A}) = \sum_{i\in\mathbb{N}} u_{\mathbb{A}}(i)$ is submodular.*

PROOF. This is true because any non-negative linear combination of submodular functions is also submodular [18]. □

Based on the above development, we are in a position to employ Nemhauser et al.'s celebrated result on the maximization of submodular functions [28], which is stated as follows:

PROPOSITION 6. *If h is submodular, nondecreasing, and $h(\emptyset) = 0$, then the greedy algorithm finds a set $\mathbb{A}$ such that $h(\mathbb{A}) \geq (1 - \frac{1}{c}) \max_{|\mathbb{B}|\leq m} h(\mathbb{B})$ where $c \sim 2.718$ is the natural logarithm base.*

[2]This can be seen by noting C is similar to $\hat{C} := D^{-1/2}_{\bar{\mathbb{N}}\bar{\mathbb{N}}} C D^{1/2}_{\bar{\mathbb{N}}\bar{\mathbb{N}}} = D^{-1}_{\bar{\mathbb{N}}\bar{\mathbb{N}}} W_{\bar{\mathbb{N}}\bar{\mathbb{N}}}$ while the sum of each row in $\hat{C}$ is less than 1 [11].

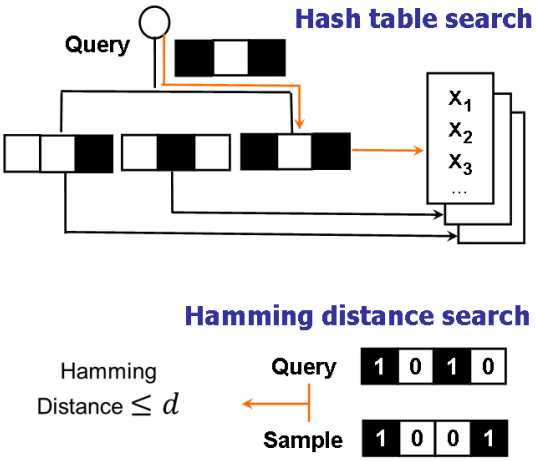

Figure 2: Comparing the difference between the searching methods using hash table and Hamming distance.

Since our objective $\Omega(\mathbb{A})$ for the hash function selection problem is submodular, we can obtain a nearly optimal combination of hash codes by a greedy algorithm. We start from an empty set $\mathbb{A}_0 := \emptyset$ and iteratively adds a code y to $\mathbb{A}_i$ that maximizes the increase of influence of Ω. Specifically, we select the node y_{i+1} in step $i+1$ using the following criterion

$$y_{i+1} = \arg\max_{y\in\mathbb{N}_i} \Omega(\mathbb{A}_i \cup \{y\}) - \Omega(\mathbb{A}_i) \quad (10)$$

where $\mathbb{A}_i$ denotes the set of nodes selected in the previous i steps and $\mathbb{N}_i$ denotes the remaining nodes (the sink node s is excluded).

4.2 Implementation

In this subsection, we discuss several important issues in implementing the submodular video hashing: how to build hash tables and how to measure the similarity between two hash codes.

Hash Tables: A lot of previous studies [34] [31] employ the Hamming distance of binary codes to approximate the neighborhood structure in Euclidean space. However, there are two shortcomings of computing Hamming distance. First, when the dataset is large, computing the pairwise Hamming distance is still expensive and requires a lot of storage. Moreover, when the length of hashing codes grows high, the Hamming approximation will become less precise and the Hamming ball algorithm will be ineffective [10]. These problems will become more serious when multiple hash codes are involved for video indexing. In this paper, we do not compute the hamming distance between query and samples in the datasets. On the contrary, we use hash tables to find nearest neighbors. Hash tables, which are in principle similar with inverted index [41], build a series of bucket containing the indices of the documents with the same hash code. Given a query, we can find the bucket corresponding hash codes in near constant time, and return all the documents in the bucket as the retrieval results. Figure 2 compares the different searching methods using hash table and hamming distance. It is easy to see that searching based on

Hamming distance means computing the pairwise distance between queries and samples in the database, and hence the computational complexity increases with the growing number of samples and length of the hash codes. On the other hand, searching with hash tables does not compute the pairwise distance and is scalable to large-scale data, which is especially useful for video retrieval. The cost of our efficient search lies in the efforts of building hash tables and overlooking those hash codes which are different from the query. However, such cost is acceptable for our retrieval systems. The hash tables can be built once and used for ever, and we employ multiple hash codes to catch the similarity in different feature descriptions.

Hash Functions: The structure of hash table is flexible enough to accommodate various hash codes. There have been a lot of studies of designing hash function and generating hash codes. This paper tries two types of popular hash functions: random projection [8] and spectral hashing [34]. The random projection method generates some random vectors, and computes the inner project of a random vector and the feature vectors which are thresholding to binary variables. To remove the affects of unorganized data, we choose the threshold as randomly sampling from the data. The other method, spectral hashing, develops the hashing function by minimizing the weighted sum of Hamming distance on graph Laplacian. Weiss et al. showed that spectral hashing can be obtained by evaluating small eigenvalues in PCA direction [34]. In the original spectral hashing algorithm, the authors compute the Hamming distance between query and other hash codes. In this paper, we employ hash tables for the retrieval task without computing the Hamming distance.

Similarity between Hash Functions: How to measure the similarity between hash codes is a crucial question. We first consider the scenario of unsupervised learning. Suppose there are two hash function h_i and h_j. Given a query q, we can retrieve its similar videos $\mathbb{R}_i(q)$ and $\mathbb{R}_j(q)$ which share the same hash code using i and j, respectively. We can explore whether the two hash functions are similar by examining how much overlapping $\mathbb{R}_i(q)$ and $\mathbb{R}_j(q)$ enjoy. If $\mathbb{R}_i(q)$ and $\mathbb{R}_j(q)$ are similar, we can expect the two hash functions are similar. Otherwise we expect the hash functions are different from each other. To measure the overlapping of retrieval results we employ the Jaccard index

$$\gamma_{ij}(q) = \frac{|\mathbb{R}_i(q) \cap \mathbb{R}_j(q)|}{|\mathbb{R}_i(q) \cup \mathbb{R}_j(q)|}.$$

By averaging the Jaccard index scores over a number of queries, we get the similarity between two codes as

$$w_{ij} = E_q \left[\gamma_{ij}(q)\right], \tag{11}$$

where E_q denotes the expectation over all queries.

In addition to unsupervised modeling, it is often interesting to consider the semi-supervised scenario where part of the videos are labeled. Suppose $\mathbb{T}_i(q)$ and $\mathbb{T}_j(q)$ are two sets with labeled samples, we consider the Jaccard index of labeled samples

$$\delta_{ij}(q) = \frac{|\mathbb{T}_i(q) \cap \mathbb{T}_j(q)|}{|\mathbb{T}_i(q) \cup \mathbb{T}_j(q)|}, \tag{12}$$

and the similarity is measured by

$$w_{ij} = E_q \left[\alpha\gamma_{ij}(q) + (1-\alpha)\delta_{ij}(q)\right]. \tag{13}$$

Algorithm 2 Submodular Video Hashing

Stage I: Video Pooling
Set the set of hash code $\mathbb{V} = \emptyset$.
for every feature f **do**
  create C video pooling component using feature f.
  for every component $k = 1$ to C **do**
    generate a hashing function using component f^k.
    build a hashing table where each bucket store video names corresponding to the same hash code.
    add the hash code to the set $\mathbb{V} = \mathbb{V} \cup \{f^k\}$.
  end for
end for
Stage II: Submodular Code Selection
Build a graph with $|\mathbb{V}|$ nodes corresponding to all possible hash codes.
Measure the similarity between each pair of the codes.
Select m hash codes in a greedy way using eq. (10). The set of selected hash codes is denoted as $\mathbb{A}$.
Stage III: Video Retrieval
for every query q **do**
  Initialize the retrieve set as $\mathbb{R} = \emptyset$.
  for code m in $\mathbb{A}$ **do**
    $\mathbb{R} = \mathbb{R} \cup \{$samples in the same bucket of q using code $m\}$.
  end for
end for
Output: Return $\mathbb{R}$ as the retrieved videos.

Note here α is the weighting factor to balance the labeled and unlabeled information.

So far we have discussed the important issues in implementing the submodular video hashing. The overall video indexing algorithm is described in Algorithm 2.

4.3 Analysis

Diversified Ranking of Hash Functions: The goal of our hash function selection is to select representative hash codes to reduce redundancy while maintaining their diversity. Intuitively, in order to maximize the code influence, the selected codes should be densely connected to other codes with good similarity. Simultaneously, the selected codes should be sufficiently distant from each other to have a broad and balanced coverage of the search. Once a node is selected, the marginal gains of including its neighbors largely drop because they have been well represented by the selected node. To illustrate this, we employ a toy example to illustrate our code selection method. As shown in Figure 3, there are five hash codes, and each code corresponds to a set of retrieval result for a given query. For the ease of clearness, in this toy example we measure hash code similarity based on only a single query. In practice we randomly select 100 queries and compute the average of the Jaccard scores. We compute similarity between these codes using Jaccard index score, and apply our submodular code selection method on this toy data. The results show that our greedy method first selects hash code 1 and then code 4. We also plot the corresponding nodes with their similarity relationship for intuitive observation.

Computational Complexity: By combining multiple hash tables, our system can retrieve large-scale video dataset very efficiently. Suppose there are N videos and $|\mathbb{A}|$ hash codes, each hash code corresponds to b bits hash function. Given a query, the computational complexity of our sys-

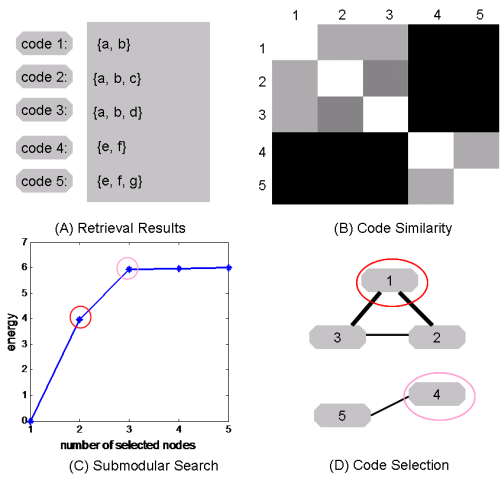

Figure 3: Toy data: representative hash function selection and diversified ranking.

tem is $O(|\mathbb{A}|)$, while the complexity of Hamming distance search is $O(bN|\mathbb{A}|)$. To validate the analysis, we generate a synthetic dataset with 1,000,000 randomly generated binary hash codes, and another 100 codes as queries. We compare different code length: 8 bits, 16 bits and 32 bits. For each code length, we compare the speeds using Hamming distance search and hash tables. Note that the hamming distance is computed efficiently using fast bit XOR operation with the help of look up table so that it has been used in quite a number of studies [33] [32]. However, as Figure 4 shows, computing Hamming distances even with the fast XOR operation is still much less efficient than our hash table method.

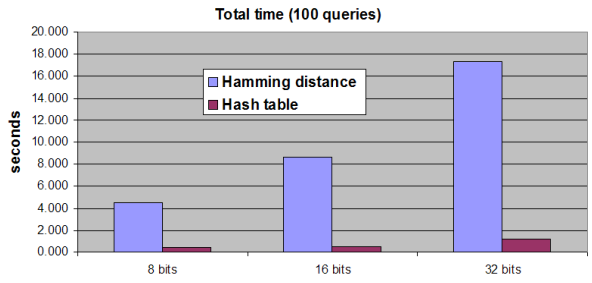

Figure 4: Comparing the speed of Hamming distance search and hash table search.

5. EXPERIMENTS

In recent years, there has been a proliferation of Web-shared videos, with about 48 hours of video uploaded on Youtube every minute, and over 700 billion videos watched in 2010. The large amount of online videos calls for efficient ways to organize and retrieve methods for online video sharing communities. However, those online videos are difficult to model since most of them are free of editing, accompanied with non-professional recording and variety of illumination, camera motion, and cluttered background.

In our experiments, we select two datasets most similar to real world online videos:

- TRECVID Multimedia Event Detection (MED 2011) dataset is the largest annotated dataset specifically designed to model complex video events. There are about 1570 hours of clips in the Event-Kit and Development collections. The video duration is similar to YouTube videos, and the average duration is about 2 to 3 minutes. The evaluation of MED is separated in two test sets: the mid size dryrun evaluation and the large final evaluation set. We report results on both dry run and final evaluation sets respectively.

- Columbia Consumer Video (CCV) dataset [14] contains 9,317 YouTube videos over 20 semantic categories. Jiang et al. [14] used Amazon MTurk platform to perform manual annotation. The database was collected with extra care to ensure relevance to consumers' interest and originality of video content without post-editing. Such videos typically have very little textual annotation and thus can benefit from the development of automatic content analysis techniques.

Experiments on both datasets are conducted on a computer with quad-core 2.33 Intel Xeon CPU and 16G RAM. All the experiments use only one thread and have the potential of further speeding up using multiple threads.

A practical question in submodular video hashing is to select the number of video components and the number of features. We use C to represent the number of components as well as the number of scenes. In traditional scene recognition, Fei-Fei and Perona [7] employed 13 scenes in their experiment, which is later enlarged to 15 scenes by Lazebnik *et al.* [19]. Following their research, we choose the number of video component as $C = 16$, although our method can deal with any number of components.

Another important question is to model the correlation between hash codes. In Section 4.2 we describe retrieval similarity measures in both unsupervised and supervised learning. We apply the unsupervised measure for the MED dataset and the semi-supervised measure for CCV dataset. Since each video in the CCV dataset is assigned a label from 20 semantic categories, we evaluate retrieval performance for all categories. Note that since many of the videos in MED dataset have no labels, further effort is needed to assess the retrieval performance.

5.1 Results on MED datasets

The dataset of MED 2011 is composed of dryrun evaluation stage and final evaluation stage. In the dryrun evaluation, 5 events are considered: *attempting a board trick*, *feeding an animal*, *landing a fish*, *wedding ceremony*, *working on a woodworking project*. The final evaluation stage, on the other hand, considers 10 new events: *birthday party*, *changing a vehicle tire*, *flash mob gathering*, *getting a vehicle unstuck*, *grooming an animal*, *making a sandwich*, *parade*, *parkour*, *repairing an appliance*, and *working on a sewing project*. In addition to these 15 events, there are a lot of videos which are not associated with any event. The public dataset contains labels for more than 13K video clips.

Since the dryrun evaluation is of a relatively small scale, we utilize several low-level features to test their effectiveness. We explore the following features: edge histogram (edge-

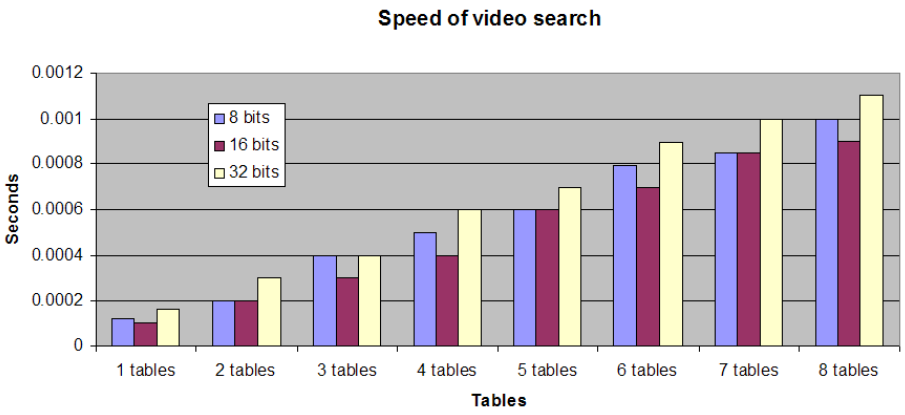

Figure 5: Speed of video retrieval on MED final evaluation dataset.

hist), linear binary pattern (LBP), color histogram (colorhist) and SIFT histogram (SIFT). We extract video frame feature and compute 16 video components for every feature. Our method represents every video component with a hashing function and select a number of hash tables using the submodular search approach in Section 4. We use two types of hashing functions: random projection (RP) and spectral hashing (SH).

To measure the performance of our hashing method, we use every video in the five event categories as query and check how many videos in the retrieved results with the same event category as the recall for the query. Since more hash tables call for more retrieve results, we compare the number of recall subject to different number of hash tables. We retrieve the videos for every query and compute the average of the recalls.

To validate the success of our submodular hashing, we compare its performance of classical hashing methods. Following the previous work [35], we compute a video signature by accumulating all the key-frames in the video using different features (edgehist, LBP, colorhist, dense SIFT). To make a fair comparison, we employ random projections to construct the same number of hash tables, and compare the recalled samples with our submodular hashing techniques. The results in Figure 6 show that our submodular hashing method is significantly better, This confirms the crucial benefit of employing multiple scene components for video representation and the submodular selection strategy for hash table construction The results also show that the performance of low-level features such as LBP and color histogram are inferior than SIFT histogram, which motivate us to focus on powerful features like SIFT and its variants in the final run.

After we finish the dryrun evaluation, we also apply the proposed method on the final evaluation dataset. Since the final submission is larger, it contains more diversified contents and poses harder challenging for video hashing. We select several powerful features: dense SIFT, sparse SIFT, and a new semantic concept feature named 780 dimensional model vectors. Our semantic model vector is an intermediate level semantic representation, by evaluating 780 concept classifiers for each frames. The 780 classifiers are trained separately using thousands of labeled web photos. A number of recent research studies [27] [22] show that similar seman-

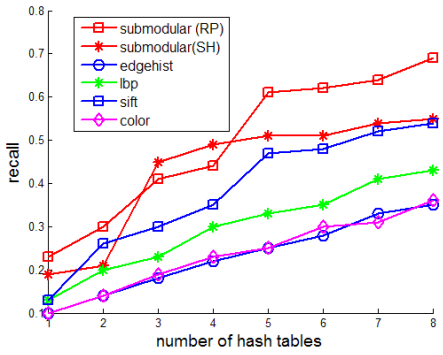

Figure 6: Comparing the performance on MED Dryrun dataset.

tic features are complementary to low-level features and can be useful in many retrieval and annotation tasks. As shown in Figure 7, semantic feature vectors work better than SIFT features for video retrieval, however, our submodular hash methods still clearly boost the performance over these features. The results also show that random projection works better than spectral hashing, which is probably because the objective function of spectral hashing applies to the hamming distances between all the samples, of which a large proportion has been ignored in our hash table structure. Despite that, the contributions of the video component and submodular selection are still consistent in both cases.

In practice, a big advantage of our submodular video hashing lies in the speed even in large-scale dataset. Figure 5 compares the speed of our hash table based method and hamming distance based searching methods on MED final evaluation dataset. These experiments are evaluated on a windows 64bit workstation with 2.33G CPU. The results show that although the search speed decreases with more hash tables, we can still finish the retrieval in about 0.001 seconds for each query even with 8 hash tables.

5.2 Results on CCV dataset

The CCV dataset hosts 9.2K video clips, where a video corresponds one of the 20 events: *basketball*, *baseball*, *soccer*, *ice skating*, *skiing*, *swimming*, *biking*, *cate*, *dog*, *bird*,

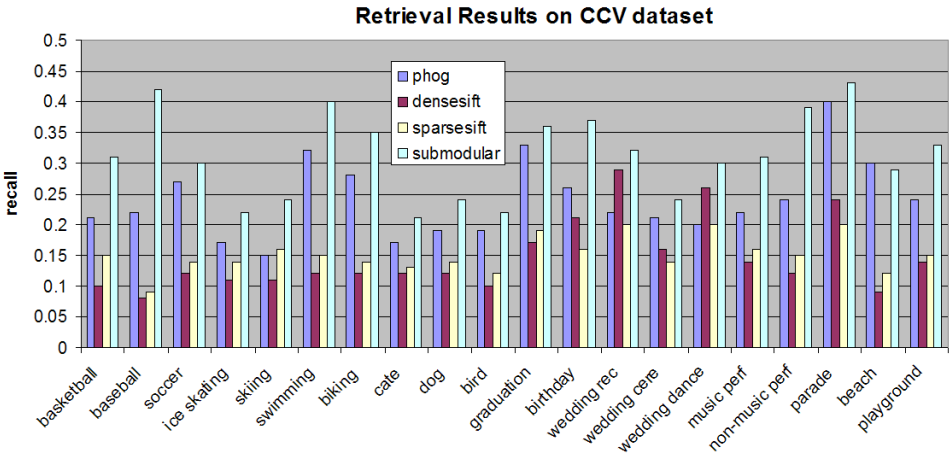

Figure 8: Video retrieval performance on CCV dataset.

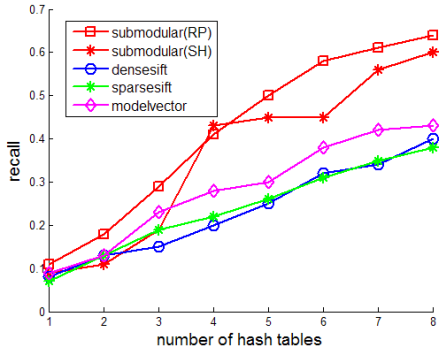

Figure 7: Comparing the performance on MED Final dataset.

graduation, *birthday*, *wedding reception*, *wedding ceremony*, *wedding dance*, *music performance*, *non-music performance*, *parade*, *beach*, *playground*. Every category contains 200-800 positive videos. Since there are rich amount of positive samples, we can employ eq. (13) to measure the code similarity for each category with the parameter $\alpha = 0.8$. In this dataset, we employ three features including sparse sift, dense sift, and pyramid histogram of oriented gradients (PHOG). Since the similarity of different hash codes depends on the category label, we also compare the retrieval performance for each category. To make a fair comparison, we report the results with four hash tables. The average recall is used to measure retrieval accuracy as we did on the MED dataset. As shown in Figure 8, our submodularity based video hashing is consistently better than hashing accumulated video features.

6. CONCLUSIONS

This paper develops a novel framework for large-scale video retrieval. We view a video as a collections of scene components and represent them with multiple hash codes. We cast the code selection problem as an influence maximization problem where the influence of the selected nodes can be maximally propagated to the rest of the nodes in the graph. We prove that the code selection is submodular and can be solved by a greedy method with guaranteed lowerbound. Benefiting from the three key components (video pooling, submodular selection, and multiple hash tables), our video hashing framework works efficiently for large-scale video retrieval and obtains superior performance on TRECVID MED and Columbia CCV datasets.

We plan to extend our work in the following directions:

- Heterogenous feature fusion for video retrieval. One advantage of our submodular video hashing method is that it can easily accommodate various features. We plan to generalize this approach to multi-modal features, including not only visual but also audio features and tag features.
- Personalized video retrieval. It is also attractive to leverage users' interests at individual levels. Our method provides a flexible way to capture interesting scenes or contents according to user interests and has the potential to be combined with more complex user profile information for improved retrieval.

Acknowledgement

This research is supported by Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Busyness Center contract number D11PC20070. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

7. REFERENCES

[1] Y. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2011.

[2] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[4] L. Cao, Y. Mu, S.-F. Chang, A. Natsev, G. Hua, and J. R. Smith. Scene aligned pooling for complex video recognition. In *ECCV*, 2012.

[5] F. Chung. *Spectral graph theory*. Number 92. Amer Mathematical Society, 1997.

[6] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *ICML*, 2011.

[7] L. Fei-Fei and P. Perona. A bayesian hierarchy model for learning natural scene categories. In *CVPR*, 2005.

[8] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[9] J. He, S. Kumar, and S.-F. Chang. On the difficulty of nearest neighbor search. In *ICML*, 2012.

[10] J. He, W. Liu, and S. Chang. Scalable similarity search with optimized kernel hashing. In *SIGKDD*, pages 1129–1138, 2010.

[11] R. Horn and C. Johnson. *Matrix analysis*. Cambridge Univ Pr, 1990.

[12] Z. Huang, H. Shen, J. Shao, X. Zhou, and B. Cui. Bounded coordinate system indexing for real-time video clip search. *ACM Transactions on Information Systems*, 27(3):17, 2009.

[13] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *ACM SIGKDD*, pages 538–543, 2002.

[14] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *ICMR*, 2011.

[15] A. Karpenko and P. Aarabi. Tiny videos: A large dataset for nonparametric video retrieval and frame classification. *TPAMI*, 33(3):618–630, 2011.

[16] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *ACM Multimedia*, pages 869–876, 2004.

[17] G. Kim, E. Xing, L. Fei-Fei, and T. Kanade. Distributed cosegmentation via submodular optimization on anisotropic diffusion. In *ICCV*, pages 169–176, 2011.

[18] A. Krause and C. Guestrin. Beyond convexity: Submodularity in machine learning. *ICML Tutorials*, 2008.

[19] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178, 2006.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. *KDD*, 2007.

[22] L.-J. Li, H. Su, E. P. Xing, and L. Fei-Fei. Object bank: A high-level image representation for scene classification and semantic feature sparsification. In *NIPS*, 2010.

[23] Z. Li, H. Ning, L. Cao, T. Zhang, Y. Gong, and T. Huang. Learning to search efficiently in high dimensions. *NIPS*, 24, 2011.

[24] J. Liu, Z. Huang, H. Shen, and B. Cui. Correlation-based retrieval for heavily changed near-duplicate videos. *ACM Transactions on Information Systems*, 29(4):21, 2011.

[25] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. CVPR, 2012.

[26] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. ICML, 2011.

[27] M. Merler, L. X. Bert Huang, G. Hua, and A. Natsev. Semantic model vectors for complex video event recognition. *IEEE Transactions on Multimedia*, 2011.

[28] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[29] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001.

[30] B. Russell, A. Torralba, C. Liu, R. Fergus, and W. T. Freeman. Object recognition by scene alignment. In *NIPS*, 2007.

[31] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–757, 2003.

[32] J. Song, Y. Yang, Z. Huang, H. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Multimedia*, pages 423–432, 2011.

[33] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[34] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. NIPS, 2008.

[35] X. Wu, A. Hauptmann, and C. Ngo. Practical elimination of near-duplicates from web video search. In *ACM Multimedia*, pages 218–227, 2007.

[36] J. Xiao, J. Haysy, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[37] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[38] D. Zhang, F. Wang, and L. Si. Composite hashing with multiple information sources. In *SIGIR*, pages 225–234, 2011.

[39] X. Zhu. Semi-supervised learning literature survey. 2005.

[40] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

[41] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, 23(4):453–490, 1998.