Large-Scale Video Hashing via Structure Learning

Guangnan Ye[†], Dong Liu[†], Jun Wang[‡], Shih-Fu Chang[†]
[†] Dept. of Electrical Engineering, Columbia University, New York, NY, USA
[‡] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
{yegn,dongliu,sfchang}@ee.columbia.edu, wangjun@us.ibm.com

Abstract

Recently, learning based hashing methods have become popular for indexing large-scale media data. Hashing methods map high-dimensional features to compact binary codes that are efficient to match and robust in preserving original similarity. However, most of the existing hashing methods treat videos as a simple aggregation of independent frames and index each video through combining the indexes of frames. The structure information of videos, e.g., discriminative local visual commonality and temporal consistency, is often neglected in the design of hash functions. In this paper, we propose a supervised method that explores the structure learning techniques to design efficient hash functions. The proposed video hashing method formulates a minimization problem over a structure-regularized empirical loss. In particular, the structure regularization exploits the common local visual patterns occurring in video frames that are associated with the same semantic class, and simultaneously preserves the temporal consistency over successive frames from the same video. We show that the minimization objective can be efficiently solved by an Accelerated Proximal Gradient (APG) method. Extensive experiments on two large video benchmark datasets (up to around $150K$ *video clips with over* 12 *million frames) show that the proposed method significantly outperforms the state-of-the-art hashing methods.*

1. Introduction

Most of the current commercial video search engines rely on textual keyword matching rather than visual content-based indexing. Besides the well-known issue of semantic gap, the computational cost is another bottleneck for content-based video search since exhaustive comparisons of low-level visual features are practically prohibitive, when handling a large collection of video clips.

Fortunately, the emerging hash-based Approximate Nearest Neighbor (ANN) search methods provide efficient ways to large-scale video retrieval. Especially, the framework of *learning to hash* has been well studied, and many new hashing methods have been developed through incorporating various machine learning techniques, ranging from unsupervised to semi-supervised to supervised learning [22, 21, 11, 7, 12, 8, 6]. The key idea for learning-based hashing is to leverage data properties or human supervision to derive compact yet accurate hash codes. Most of the existing hashing methods can be directly applied to index video data, such as the recent multiple feature based video hashing [19] and submodular video hashing [3]. Despite of the promising results reported in the literature, the existing video hashing methods cannot explicitly encode the specific structure information in video clips, e.g., the commonly shared local visual patterns by videos associated with the same semantic labels and the temporal consistency between successive frames.

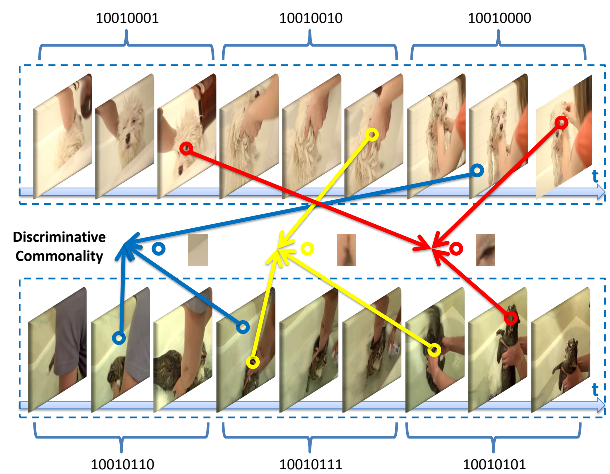


Figure 1. An illustration of the proposed hash code generation for videos within the event category "feeding an animal". Discriminative local commonality is automatically discovered, e.g., the eyes of animals, the edges of tubs, and the parts of human hands. Temporal consistency is preserved, and successive frames are grouped and put into the same hash bucket.

To address the above problems, we propose to explore the structure information to design novel video hashing methods. In particular, two important types of structure information are considered in the learning process. The first type is the spatial structure information, namely *Discriminative Local Visual Commonality*. Notably, although each video clip contains fruitful local visual patterns, only a limited number of discriminative local patterns are shared by videos within the same semantic category. For example, a video of event "feeding an animal" in Figure 1 can be characterized by a sparse subset of visual patterns (e.g.,"eyes" and "hands" in Figure 1). The idea is analogous to sparse coding, in which only a small subset of codewords or feature dimensions have non-zero weights. The second type is called *Temporal Consistency*. It is easy to expect successive frames to maintain similar visual appearance. Therefore, the hashing method should ensure that the hash codes for successive frames to be as similar as possible (Figure 1).

To incorporate these two types of structure characteristics of videos, we propose a supervised framework with structure learning to design efficient linear hash functions for video indexing. In particular, we formulate our objective as a minimization problem of a structure-regularized empirical loss function. To capture the common local patterns across all the video frames from the same category, the first regularization term imposes a $\ell_{2,1}$-norm over the hash functions so that only a small number of informative feature dimensions are selected. Since we apply the widely used Bag-of-Words (BoW) model with local SIFT [15] features for video representation in our formulation, such selected feature dimensions, i.e., visual words, correspond to discriminative local visual patterns. In this way, we obtain consistent patterns across different videos, and improve the discrimination ability of the learned hash functions. The second regularization term uses a ℓ_∞-norm on the hash codes of successive frames, which enforces successive frames to receive similar hash codes and essentially preserves the temporal consistency in Hamming space. Finally, we apply an APG method to efficiently solve the minimization problem. Extensive experiments over two large video benchmark datasets and comparisons with representative learning-based hashing methods demonstrate the superiority of the proposed video hashing method. Compared with our baseline method without structure learning, we also clearly demonstrate that leveraging video structure information helps improve the performance significantly.

2. Related Works

The rapid growth of massive databases in various applications has promoted the research and study of hash-based indexing algorithms. Recently, *learning to hash* framework has been extensively investigated and various machine learning algorithms are incorporated for designing efficient hash functions. In the following, we briefly introduce several representative learning-based hashing methods and the recent applications on video indexing and retrieval.

Unsupervised hashing methods often utilize the data properties such as distribution or manifold structure to design effective indexing schemes. For example, spectral hashing assumes that the data are sampled from a uniform distribution and partitions the data along their principal directions with the consideration of spatial frequencies [22]. Graph hashing explores the low-dimensional manifold structure of data to design compact hash codes [13]. Supervised hashing learning can be mainly categorized as pointwise and pairwise methods. Pointwise methods, such as boosted similarity sensitive coding [18] and deep neural network-based method [20], often treat the design of hash functions as a special classification problem and use samples' labels in the training procedure. Pairwise methods take into account the pairwise relationship of samples in the hash function learning. As a popular formulation, many recent works, including binary reconstructive embedding [11], complementary hashing [23], supervised hashing with kernels [14], and iterative quantization [7] fall into the category of pairwise methods. Finally, semi-supervised hashing method plays a tradeoff between supervised information and data properties to design robust hash functions, which aims at alleviating the defects from overfitting or insufficient training [21].

Note that all the aforementioned methods treat the data samples separately and generate binary codes for each sample independently. To our best knowledge, there are very limited studies on developing specific hash functions to index structured data like videos. More recently, Douze et al. [4] proposed a method to handle the temporal burstiness for video copy detection. Cao et al. [3] proposed a submodular hashing framework to index videos. Song et al. [19] proposed a multiple feature based hashing for video near-duplicate detection. However, in those video hashing methods, conventional hashing methods like locality sensitive hashing [6] and spectral hashing [22] are often applied to generate binary codes. None of the existing methods really consider the special structure information like visual commonality and temporal consistency of videos to design structure-specific hash functions for video indexing. In contrast, our proposed video hashing method leverages video structure information in a supervised learning paradigm to derive optimal binary codes for large-scale video retrieval.

3. Structure Learning Based Video Hashing

In this section, we will introduce our structure learning method for video hashing. We first present the notation and definition, and then describe the problem formulation.

3.1. Notation and Definition

Suppose we have a training video collection $\mathcal{X} = \{X_i, y_i\}_{i=1}^N$ with N videos, where $X_i = [\mathbf{x}_{i1}, ..., \mathbf{x}_{ij}, ..., \mathbf{x}_{i,n_i}]$ is a video consisting of

n_i successive frames, $y_i \in \{0,1\}$ is the label of video X_i[1]. $\mathbf{x}_{ij} \in \mathbb{R}^d$ is the feature vector of the j-th frame of video X_i with d being the feature dimensionality. Without loss of generality, we assume all frame features in $\mathcal{X}$ have been normalized with zero mean.

Given a video frame $\mathbf{x}$, we want to learn K-bit binary codes $\mathbf{c} \in \{0,1\}^K$, which needs to design K binary hash functions. In this work, we consider the linear hash functions for their simplicity and efficiency. Specifically, the k-th hash function ($k = 1, \dots, K$) can be defined as:

$$h_k(\mathbf{x}) = \operatorname{sgn}(\mathbf{w}_k^\top \mathbf{x} + b_k), \tag{1}$$

where $\mathbf{w}_k \in \mathbb{R}^d$ is a hash hyperplane and b_k is the intercept. With $\mathbf{x}$ being zero-mean, b_k will be 0. The resulting $h_k(\mathbf{x}) \in \{-1,1\}$, and the corresponding binary hash bit can be simply calculated as $c_k(\mathbf{x}) = (1 + h_k(\mathbf{x}))/2$.

The Hamming distance between the hash codes of two frames $\mathbf{x}_{ia}$ and $\mathbf{x}_{jb}$ from two videos X_i and X_j can be defined as:

$$\begin{aligned} d(\mathbf{x}_{ia}, \mathbf{x}_{jb}) &= \sum_{k=1}^{K} (c_k(\mathbf{x}_{ia}) - c_k(\mathbf{x}_{jb}))^2 \\ &= \frac{1}{4} \sum_{k=1}^{K} (h_k(\mathbf{x}_{ia}) - h_k(\mathbf{x}_{jb}))^2. \end{aligned} \tag{2}$$

Based on the above definition, a straightforward way to define the Hamming distance between videos X_i and X_j is:

$$D(X_i, X_j) = \frac{1}{n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} d(\mathbf{x}_{ia}, \mathbf{x}_{jb}), \tag{3}$$

which means that the Hamming distance between two videos equals to the average Hamming distance of each pair of frames. The basic assumption behind this definition is that most of frames within a video should be useful for measuring the distance between videos, and similar pairwise metric methodology has been proven to be effective in [3].

However, the above function is not tractable due to the discrete nature of sgn function in each hash function. Hence, a typical practice is to relax the distance function by replacing sgn with its signed magnitude and rewrite the above distance as

$$\begin{aligned} D(X_i, X_j) &= \frac{1}{4 n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} \sum_{k=1}^{K} (\mathbf{w}_k^\top \mathbf{x}_{ia} - \mathbf{w}_k^\top \mathbf{x}_{jb})^2 \\ &= \frac{1}{4 n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} (\mathbf{x}_{ia} - \mathbf{x}_{jb})^\top W W^\top (\mathbf{x}_{ia} - \mathbf{x}_{jb}), \end{aligned} \tag{4}$$

where $W = [\mathbf{w}_1, \dots, \mathbf{w}_K] \in \mathbb{R}^{d \times K}$ is a matrix consisting of the coefficients of all hash functions.

[1] The proposed method can be easily extended from binary class to multi-class case.

3.2. Problem Formulation

Our goal is to learn a coefficient matrix W which not only conveys discriminative information but also incorporates the temporal information of videos. We formulate the following objective which learns the hash functions by minimizing a structure-regularized cost function:

$$\begin{aligned} \min_{W} \quad & \sum_{i,j=1}^{N} \ell\big(\hat{y}_{ij}, D(X_i, X_j)\big) + \lambda \|W\|_{2,1} \\ & + \gamma \sum_{i=1}^{N} \sum_{t=1}^{n_i - 1} \|W^\top \mathbf{x}_{i,t} - W^\top \mathbf{x}_{i,t+1}\|_\infty, \end{aligned} \tag{5}$$

where $\lambda, \gamma > 0$ are two parameters balancing the three competing terms, $\|W\|_{2,1} = \sum_{i=1}^{d} \sqrt{\sum_{j=1}^{K} W_{ij}^2}$ is the $\ell_{2,1}$-norm, and $\|W^\top \mathbf{x}_{i,t} - W^\top \mathbf{x}_{i,t+1}\|_\infty = \max_{j=1,\dots,K}\{|s_{i,t}^j|\}$ is the ℓ_∞-norm, in which $\mathbf{s}_{i,t} = W^\top \mathbf{x}_{i,t} - W^\top \mathbf{x}_{i,t+1}$ and $s_{i,t}^j$ denotes the j-th entry of vector $\mathbf{s}_{i,t}$, $\ell(\cdot,\cdot)$ is an empirical loss function with $\hat{y}_{ij} = 1$ iff $y_i = y_j$ and $\hat{y}_{ij} = -1$ otherwise.

Now we explain the rationality of our objective function. The minimization of $\|W\|_{2,1}$ ensures only a small number of rows in matrix W are non-zero. Since each row of W will be multiplied with one specific dimension of the features, making it zero will discard the influence of the feature dimension from hash function learning. As a result, only a subset of the feature dimensions are selected through the non-zero rows of matrix W. The selected features correspond to the local common patterns in the video frames related to a certain video category, which convey discriminative information. The minimization of $\|W^\top \mathbf{x}_{i,t} - W^\top \mathbf{x}_{i,t+1}\|_\infty$ ensures the hash vectors of two successive frames as similar as possible [2], i.e., encouraging the maximum absolute value of the entry-wise differences between two hash vectors to be zero. This accounts for the preservation of the temporal structure in the hash codes generated for all frames of a video. Notably, we apply ℓ_∞-norm here instead of ℓ_1-norm or ℓ_2-norm since ℓ_∞-norm will ensure stronger constraint that the two hash vectors of successive frames are close to each other. Through the above two regularizers, the structure information of videos can be comprehensively encoded into the generated Hamming space.

We define the loss function in Eq. (5) based on ℓ_1-loss:

$$\ell(\hat{y}_{ij}, D(X_i, X_j)) = \max\big(0, \hat{y}_{ij}(D(X_i, X_j) - \delta)\big), \tag{6}$$

where δ is a threshold. When minimizing Eq.(6), $\hat{y}_{ij} = 1$ leads to $D(X_i, X_j) \le \delta$ while $\hat{y}_{ij} = -1$ leads to $D(X_i, X_j) > \delta$. In this way, it enforces that videos with the same category label should be close to each other while videos with different category labels should be far apart.

With this loss term, we incorporate the supervision information into the hash function learning, leading to discriminative binary codes. Without loss of generality, we simply set $\delta = 1$. Note that other loss functions such as hinge loss or least square loss can be used as alternatives.

The objective function in Eq. (5) is non-convex, and thus is expected to achieve a local optimum. In the next section, we will introduce an efficient procedure for the optimization.

4. Optimization Procedure

In Eq. (5), the gradient of W cannot be calculated due to the non-smoothness of the $\ell_{2,1}$-norm and ℓ_∞-norm regularizers. In this section, we first show that by using the dual norm and smoothing approximation, the gradient of $\ell_{2,1}$-norm and ℓ_∞-norm can be calculated. After that we employ APG method to solve the optimization problem.

4.1. Smoothing Approximation

First, we concatenate all pairwise frame differences of the training video collection into a matrix $\hat{X} \in \mathbb{R}^{d\times T}$ with the total number of successive frame pairs $T = \sum_{i=1}^{N}(n_i - 1)$, in which each column denotes the difference of two successive frames and all columns are placed in the original temporal order in the videos. Then the original objective function in Eq. (5) can be rewritten as:

$$\begin{aligned} \min_W \quad & \sum_{i,j=1}^{N} \ell\big(\hat{y}_{ij}, D(X_i, X_j)\big) + \lambda \|W\|_{2,1} \\ & + \gamma \sum_{i=1}^{T} \|(W^\top \hat{X})_i\|_\infty, \end{aligned} \tag{7}$$

where $(W^\top \hat{X})_i \in \mathbb{R}^{K\times 1}$ denotes the i-th column in matrix $(W^\top \hat{X})$.

We resort to the smoothing approximation to the solve problem in Eq. (7). First, we decompose the objective function as the sum of the following two terms:

$$f(W) \quad := \quad \sum_{i,j=1}^{N} \ell\big(\hat{y}_{ij}, D(X_i, X_j)\big), \tag{8}$$

$$r(W) \quad := \quad \lambda \|W\|_{2,1} + \gamma \sum_{i=1}^{T} \|(W^\top \hat{X})_i\|_\infty. \tag{9}$$

Herein, the subgradient of $f(W)$ can be easily calculated based on its formula in Eq. (6). However, the gradient of $r(W)$ cannot be directly calculated due to its non-smoothness nature. Therefore, we need to give a smooth approximation so that its gradient can be computed.

Based on Nesterov's smoothing approximation method [17], $r(W)$ can be approximated by the following smooth function:

$$r_\mu(W) = \lambda \sum_{i=1}^{d} h_{i,\mu}(W) + \gamma \sum_{i=1}^{T} g_{i,\mu}(W), \tag{10}$$

with the definitions:

$$h_{i,\mu}(W) := \max_{\|\mathbf{v}\|_2 \le 1} \langle (\mathbf{w}^i)^\top, \mathbf{v}\rangle - \frac{\mu}{2}\|\mathbf{v}\|_2^2, \tag{11}$$

$$g_{i,\mu}(W) := \max_{\|\mathbf{u}\|_1 \le 1} \langle (W^\top \hat{X})_i, \mathbf{u}\rangle - \frac{\mu}{2}\|\mathbf{u}\|_2^2, \tag{12}$$

where μ is a positive smoothness parameter to control the accuracy of the approximate, $\langle\cdot,\cdot\rangle$ denotes the inner product operator, $\mathbf{w}^i$ denotes the i-th row of matrix W. $\mathbf{v}$ and $\mathbf{u}$ are respectively a vector of auxiliary variables associated with $\mathbf{w}^i$ and $(W^\top \hat{X})_i$. In our work, we set μ to be 10^{-4}.

For a fixed $\mathbf{w}^i$, assume that $\mathbf{v}(\mathbf{w}^i)$ is the unique minimizer of Eq. (11). It is standard that $\mathbf{v}(\mathbf{w}^i) = \frac{\mathbf{w}^i}{\mu\|\mathbf{w}^i\|_2}\min\{\mu, \|\mathbf{w}^i\|_2\}$. $h_{i,\mu}(W)$ is differentiable and its gradient $\nabla h_{i,\mu} = \mathbf{v}(\mathbf{w}^i)$ is Lipschitz continuous with the constant $L_{h,i} = 1/\mu$. In addition, $\mathbf{u}((W^\top \hat{X})_i)$ is the unique minimizer of Eq. (12), and it can be calculated by the ℓ_1-ball projection algorithm in [5]. $g_{i,\mu}(W)$ is differentiable and its gradient $\nabla g_{i,\mu} = \mathbf{u}((W^\top \hat{X})_i)$ is Lipschitz continuous with the constant $L_{g,i} = 1/\mu$.

4.2. Optimization Using APG

For a fixed μ, we minimize the following objective:

$$F_\mu(W) = f(W) + r_\mu(W). \tag{13}$$

It is known that F_μ is a μ-accurate approximation of F , and it is differentiable with gradient:

$$\nabla F_\mu(W) = \nabla f(W) + \nabla r_\mu(W), \tag{14}$$

where we have

$$\nabla f(W) \quad = \quad \sum_{i,j=1}^{N} \nabla \ell\big(\hat{y}_{ij}, D(X_i, X_j)\big), \tag{15}$$

$$\nabla r_\mu(W) \quad = \quad \lambda \mathbf{v}(W) + \gamma \sum_{i=1}^{T} \hat{X}_i \mathbf{u}^\top((W^\top \hat{X})_i), \tag{16}$$

in which $\hat{X}_i$ denotes the i-th column of matrix $\hat{X}$, $\mathbf{v}(W) = [\mathbf{v}(\mathbf{w}^1), \ldots, \mathbf{v}(\mathbf{w}^d)]^\top \in \mathbb{R}^{d\times K}$, and $\nabla \ell\big(\hat{y}_{ij}, D(X_i, X_j)\big)$ can be calculated as:

$$\nabla \ell\big(\hat{y}_{ij}, D(X_i, X_j)\big) = \left\{ \begin{array}{l} 0, \ \text{if } \hat{y}_{ij}\big(D(X_i, X_j) - \delta\big) \le 0, \\ \hat{y}_{ij}\Omega_{ij}W, \ \text{otherwise.} \end{array} \right. \tag{17}$$

Algorithm 1 Solving Problem of Eq. (13) by APG

Input: $X_i \in \mathbb{R}^{d \times n_i}$, $y_i \in \{0, 1\}$, $i = 1, \ldots, N$, $\hat{X}$, λ, γ, δ and μ.
Initialize: Calculate L_{F_μ} based on Eq. (18), randomly initialize $W^{(0)}$, $Z^{(0)} \in \mathbb{R}^{d \times K}$, and $\eta^{(0)} \leftarrow 0$, $t \leftarrow 0$.
repeat
  $\alpha^{(t)} = (1 - \eta^{(t)})W^{(t)} + \eta^{(t)} Z^{(t)}$.
  Calculate $\nabla F_\mu(\alpha^{(t)})$ based on Eq. (14).
  $Z^{(t+1)} = Z^{(t)} - \frac{1}{\eta^{(t)} L_{F_\mu}} \nabla F_\mu(\alpha^{(t)})$.
  $W^{(t+1)} = (1 - \eta^{(t)})W^{(t)} + \eta^{(t)} Z^{(t+1)}$.
  $\eta^{(t+1)} = \frac{2}{t+1}$, $t \leftarrow t + 1$.
until Converges.
Output: $W^{(t)}$.

Herein, $\Omega_{ij} = \frac{1}{2n_i n_j} \sum_{a=1}^{n_i} \sum_{b=1}^{n_j} (\mathbf{x}_{ia} - \mathbf{x}_{jb})(\mathbf{x}_{ia} - \mathbf{x}_{jb})^\top$.

It is straightforward to verify that $\nabla f(W)$ is Lipschitz continuous with constant $L_f = \| \sum_{i,j=1}^{N} \Omega_{ij} \|_2$ where $\| \cdot \|_2$ denotes the spectral norm of a matrix. Combining with the Lipschitz constants of $h_{i,\mu}(W)$ and $g_{i,\mu}(W)$, we get that $\nabla F_\mu(W)$ is Lipschitz continuous with constant

$$L_{F_\mu} = \| \sum_{i,j=1}^{N} \Omega_{ij} \|_2 + \frac{1}{\mu} \Big(\lambda \times d + \gamma \times T \Big). \tag{18}$$

We are now ready to employ APG to optimize $F_\mu(W)$. The optimization procedure is described in Algorithm 1.

5. Experiments

In this section, we provide extensive experiments and comparison studies on two large video collections, i.e., the Columbia Consumer Video (CCV) [9] and the TRECVID Multimedia Event Detection (MED) 2012 video dataset [1]. The details of these two video datasets will be described later. To demonstrate the strength of using both spatial and temporal structure information of videos, we test several variants of the proposed structure learning based hashing methods, and also compare with several representative hashing methods in our experiments. The following six methods will be compared: (1) Spectral Hashing (SH) [22]; (2) Sequential Projection Learning based Hashing (SPLH) [21]; (3) Unstructured Video Hashing (UVH), where we ignore the structure information of videos and learn the hash functions based on frame features only by setting λ and γ in Eq. (5) to be 0; (4) Video Hashing with Discriminative commonality (VHD) only, which can be realized by setting γ in Eq. (5) to be 0; (5) Video Hashing with Temporal consistency (VHT) only, which can be realized by setting λ in Eq. (5) to be 0; (6) Video Hashing with both Discriminative commonality and Temporal consistency (VHDT). Note that both SH and SPLH treat each video as a composite of independent frames and index the video by combining the

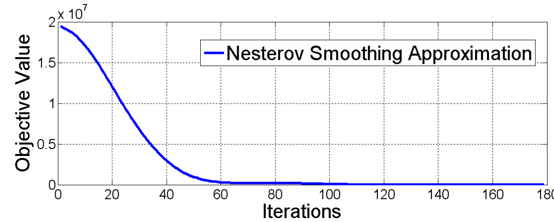


Figure 2. Convergence curve on the CCV dataset experiment.

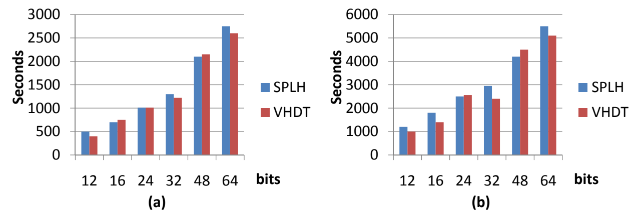
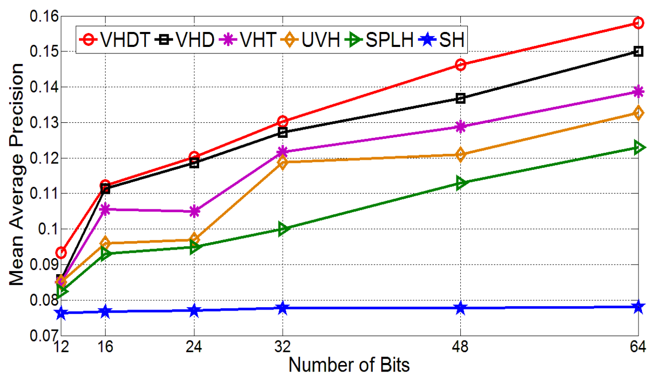


Figure 3. Comparisons of training time between SPLH and our proposed VHDT method on (a) CCV dataset and (b) TRECVID MED 2012 dataset.

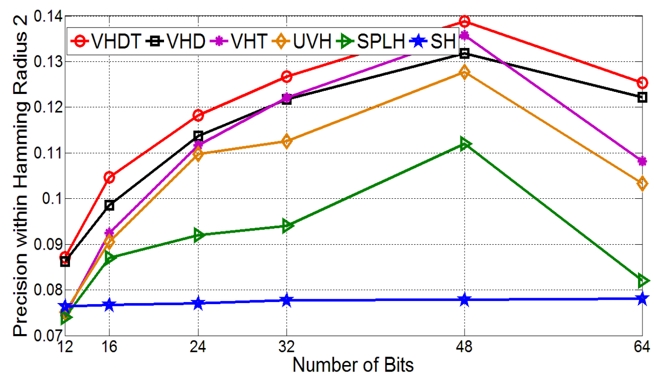
hash codes of frames, similar as that described in [3]. Moreover, SH and SPLH are the representative hashing methods with publicly available codes, and hence are chosen for fair study. Although there are some recent methods for video hashing [3, 19], they are built on standard hashing techniques like LSH, and require specific settings, like submodular or multiple-feature representations, hence not serving as standard comparable methods in our experiments.

5.1. Evaluations and Settings

We follow the evaluation protocols used in [21] and adopt the following two criteria: (1) Hamming ranking: All the video clips are ranked according to their Hamming distance to the query video. (2) Hash lookup: A hash lookup table is constructed and all the samples fall within a Hamming ball with radius r ($r = 2$ in our setting) to the query sample are returned. Since each query video is represented as a set of binary codes corresponding to the frames in the video, here we adopt the following query strategy to return the nearest neighbor (NN) video clips. For Hamming ranking based evaluation, it is fairly straightforward to return the nearest video clips by exhaustively computing and ranking the video Hamming distance between the query and the database samples, as defined in Eq. (3). For hash lookup based evaluation, we first retrieve the nearest neighbor frames within Hamming radius 2 for each individual hash code vector of the query video frames. Then all the videos whose composite frames are successfully hit by any query frame will be regarded as the candidate NN videos. An intuitive way to rank all these candidate NN videos is by counting the hit frequency of each video (normalized by the number of total frames in that video). Note that these two evaluations focus on different aspects of hashing techniques.



(a) MAP



(b) Precision within Hamming radius 2

Figure 4. MAP/precision within Hamming radius 2 comparisons of different methods on the CCV dataset.

Hamming ranking provides better quality assessment but its complexity is linear. Hash lookup emphasizes the search speed since the query complexity is often constant time, but the search quality could be unjustified when using very long hash codes, resulting in failed queries due to empty return within Hamming radius r.

In order to get the quantitative comparison, for Hamming ranking criterion, we employ Average Precision (AP) as the evaluation metric. Specifically, given the returned ranking list with length m, AP is computed as $AP = \frac{1}{R}\sum_{j=1}^{m}\frac{R_j}{j}G_j$, where R is the number of positive samples in the ranking list, and R_j is the number of relevant samples among the top j samples, $G_j = 1$ if the jth sample is positive and 0 otherwise. For hash lookup criterion we compute retrieval precision which measures the percentage of true neighbors within Hamming radius r [21]. For each of the above two metrics, we calculate the result for each query and then report the mean value across all queries as the final evaluation metric.

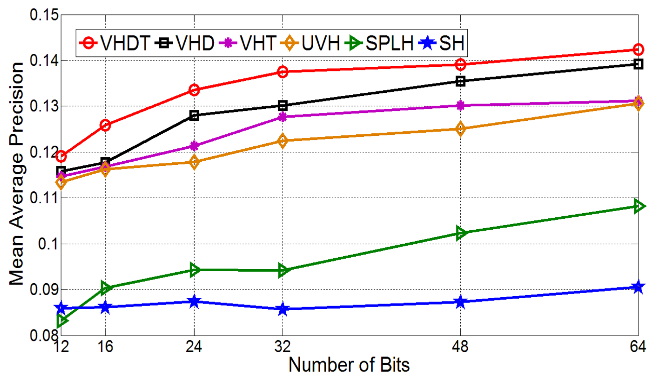
For SH and SPLH, we use the best settings reported in literatures [22, 21]. For our proposed method, we use cross validation to determine the appropriate parameters, i.e., the weights λ and γ. In addition, for all the supervised and semi-supervised methods, a small set of labeled samples are randomly selected as training data on both CCV dataset and TRECVID dataset.

We implement our method on an Intel XeonX5660 workstation with 2.8GHz CPU and 8GB memory, and observe very good convergence property. Figure 2 shows the convergence process of the iterative optimization which is captured in our later experiment. As seen, the objective function converges to the local minimum after about 60 iterations, and thus the convergence speed is fast. For example, in the experiments on the CCV dataset, it takes around 6.3 seconds in average to run one iteration from step 3 to step 9 in Algorithm 1. Figure 3 demonstrates the training time of SPLH and our proposed VHDT method, where we can observe that these two methods have similar computational time. This indicates that our method is able to achieve comparable time complexity as the existing hashing method. Moreover, we also empirically discover that our method is not sensitive to the initialization, which shows very tiny performance differences with different initializations.

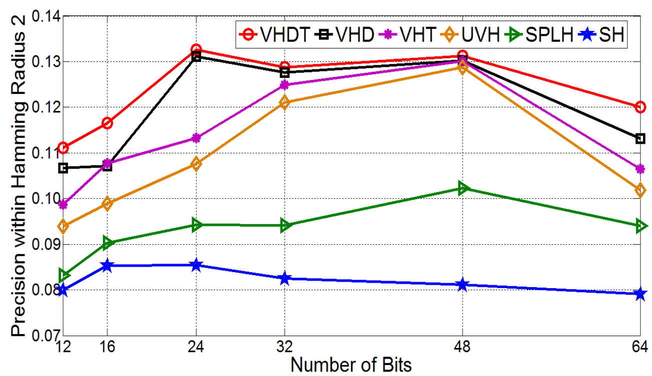
5.2. Columbia Consumer Video (CCV) Dataset

The Columbia Consumer Video (CCV) dataset [9] contains $9,317$ YouTube videos with over 20 diverse semantic categories. In our experiments, we randomly select 5 videos from each semantic category as labeled data for training, and choose another 25 videos in each category as the query videos for testing hashing performance. This results in 100 training videos and 500 query videos. The remaining $8,717$ videos are considered as the samples in the database. The key frames are evenly sampled every 2 seconds and each video has at least 30 key frames. For each key frame, we extract 128-dimensional SIFT features [15] over key points and perform BoW quantization to derive the image representations [16]. In particular, we utilize two different sparse key point detectors, i.e., Different of Gaussian and Hessian Affine. Finally, each video key frame is represented as a $5,000$-dimensional BoW feature.

In the experiments, we evaluate the performance using hash codes with different length, ranging from 12 to 64 bits. Figure 4(a) shows performance curves of the Mean Average Precision (MAP) averaging over 500 queries of different methods. From the results, we have the following observations : (1) The proposed VHDT method consistently outperforms the other baseline methods by a large margin, which demonstrates its effectiveness for video hashing; (2) All structure learning based hashing methods, including VHDT, VHD and VHT, produce significantly higher MAPs than UVH. This is due to the fact that the former methods take advantages of structure information of video data (either discriminative local patterns or temporal consistency



(a) MAP



(b) Precision within Hamming radius 2

Figure 5. MAP/precision within Hamming radius 2 comparisons of different methods on the TRECVID MED 2012 dataset.

across successive frames) while the latter one only blindly generates hash codes without accounting any structure information; (3) Our proposed VHDT clearly beats the conventional hashing methods like SH and SPLH. The reason is that these methods only try to learn hash functions for simple samples such as images and hence are not appropriate for video data; (4) Our VHDT method performs better than VHD and VHT, since the latter ones merely consider one aspect of the structure information in videos. In contrast, our VHDT method fully exploits both structure information and produces the best performance in the experiments.

We also show the precision curves within Hamming radius 2 in Figure 4(b), where similar performance gains of our method can be observed. However, the precisions of all methods begin to drop when using longer hash codes. This is because that, with the increasing of the number of hash bits, the number of samples falling in a bucket decreases exponentially, resulting in empty returns within the Hamming radius 2. Similar performance droppings have also been observed in the previous work [21]. Again, the VHDT method achieves the best performance among all methods in comparison.

5.3. TRECVID MED 2012 Dataset

TRECVID MED is the benchmark dataset for the evaluation of semantic event detection in videos. The dataset used in our experiments is TRECVID MED 2012 [1]. The entire dataset has around $150K$ videos falling into 25 semantic event categories. There are around $10K$ video clips with the ground truth semantic labels. For each video, we extract key frames every 2 seconds and obtain the final set containing over 12 million video key frames. To our best knowledge, TRECVID MED 2012 is among one of the largest video collections with manual annotation in the public research community. Since we merely have partial ground truth labels for the $150K$ videos, we evaluate the performance based on two different protocols. The first is to evaluate on the $10K$ videos with ground truth labels while the second is to evaluate on the entire $150K$ videos based on the top returned videos labeled by ourselves. In each protocol, we follow exactly the same feature extraction process as in the CCV dataset experiment. In the training process, 5 labeled videos from each category (in total 125 videos) are randomly selected.

Results on $10K$ videos with ground truth labels. In this scenario, 25 labeled videos in each category are chosen as the query video clips for testing hashing performance. Figure 5(a) and Figure 5(b) show the performance curves of different methods in terms of MAP and precision. As can be seen, our method achieves the best performance over all the other methods in comparison. The performance improvements are consistent as the number of hash bits varies. Once again the experiment results demonstrate the effectiveness of our method. It is worth noting that, for the TRECVID MED task, most of the state-of-the-art systems [10] use SVM classifier as the basic framework since the focus is the prediction performance, while the efficiency is mostly neglected. In contrast, our proposed hashing method focuses on real time retrieval on large-scale video data. For example, it will take hours to just compute the non-linear SVM kernels for $10,000$ videos, while for hash-based methods it only needs 10 seconds to find similar video clips from the entire database. Therefore, it is inappropriate to directly compare the performance of our methods with the official results of TRECVID MED 2012 due to the completely different technical purposes.

Results on $150K$ videos. Since we do not have all labels of the $150K$ videos, the AP and precision within Hamming radius 2 cannot be calculated. Therefore, we only report the precision within the top 100 returned videos for each method. We randomly select 5 videos with ground truth from each of the 25 categories, and consider the 125 videos as queries. 64-bit binary code is generated to search videos within the $150K$ video dataset. For each query, we

Figure 6. Qualitative evaluation over the $150K$ video dataset using 64-bit hash code. Queries from top to bottom belongs to category "woodworking", "rock climbing", and "vehicle unstuck". Top 6 retrieval results are presented. Incorrect results are shown with red border.

	SH	SPLH	UVH	VHT	VHD	VHDT
ACC	0.12	0.15	0.17	0.20	0.21	**0.24**

Table 1. Accuracy of top 100 retrieval videos using 64 bits over the $150K$ video dataset.

retrieve the NN videos within Hamming radius 2 and then pick up the top 100 videos ranked by the normalized frame hit frequencies. Finally, the category label of each unlabeled video within the top 100 videos is manually annotated. We calculate the accuracy of top 100 videos of each query and report the average performance across 125 queries in Table 1. As seen, our method achieves the best performance among all methods. Figure 6 shows the key frames of some exemplar query videos as well as the top 6 returned key frames, which shows that our structure learning based video hashing consistently generates better visual retrieval results.

6. Conclusion

We have introduced a structure learning method for large-scale video hashing. The proposed method works in a supervised setting with a ℓ_1-norm based empirical loss regularized by the video structure related terms. Specifically, we use a $\ell_{2,1}$-norm to select certain feature dimensions in the training videos to capture the discriminative local visual patterns, and employ a ℓ_∞-norm in the binary codes of successive frames to preserve the temporal consistency in the learned Hamming space. The final objective is formulated as a $\ell_{2,1}$-norm and ℓ_∞-norm regularized minimization problem and the APG method is applied for the optimization. Extensive experiments on two large video datasets validate the effectiveness of our method. In the future, we will investigate the proposed method in the kernel space [14].

7. Acknowledgment

We thank Wei Family Private Foundation for their support for the first author.

References

[1] http://www.nist.gov/itl/iad/mig/med12.cfm/.

[2] H. Bondell and B. Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 2008.

[3] L. Cao, Z. Li, Y. Mu, and S.-F. Chang. Submodular video hashing: A unified framework towards video pooling and indexing. In *ACM MM*, 2012.

[4] M. Douze, H. Jegou, and C. Schmid. An image-based approach to video copy detection with spatio-temporal post-filtering. *TMM*, 2010.

[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *ICML*, 2008.

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[7] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[8] J. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, 2012.

[9] 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.

[10] Y.-G. Jiang, X. Zeng, G. Ye, D. Ellis, S.-F. Chang, S. Bhattacharya, and M. Shah. Columbia-UCF TRECVID 2010 multimedia event detection: Combining multiple modalities, contextual concepts, and temporal matching. In *TRECVID workshop*, 2010.

[11] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[12] P. Li, A. Shrivastava, J. Moore, and C. Konig. Hashing algorithms for large-scale learning. In *NIPS*, 2011.

[13] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with Graphs. In *ICML*, 2011.

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

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[16] K Mikolajczyk and C Schmid. Scale and affine invariant interest point detectors. *IJCV*, 2004.

[17] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program*, 2005.

[18] G. Shakhnarovich. Learning task-specific similarity. *PhD thesis, Massachusetts Institute of Technology*, 2005.

[19] 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 MM*, 2011.

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

[21] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large scale search. *PAMI*, 2012.

[22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[23] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu. Complementary hashing for approximate nearest neighbor search. In *ICCV*, 2011.