Sequential Spectral Learning to Hash with Multiple Representations

Saehoon Kim[1], Yoonseop Kang[1], and Seungjin Choi[1,2,3]

[1] Department of Computer Science and Engineering
[2] Division of IT Convergence Engineering
[3] Department of Creative IT Excellence Engineering,
Pohang University of Science and Technology,
77 Cheongam-ro, Nam-gu, Pohang 790-784, Korea
{kshkawa,e0en,seungjin}@postech.ac.kr

Abstract. Learning to hash involves learning hash functions from a set of images for embedding high-dimensional visual descriptors into a similarity-preserving low-dimensional Hamming space. Most of existing methods resort to a single representation of images, that is, only one type of visual descriptors is used to learn a hash function to assign binary codes to images. However, images are often described by multiple different visual descriptors (such as SIFT, GIST, HOG), so it is desirable to incorporate these multiple representations into learning a hash function, leading to *multi-view hashing*. In this paper we present a *sequential spectral learning* approach to multi-view hashing where a hash function is sequentially determined by solving the successive maximization of local variances subject to decorrelation constraints. We compute multi-view local variances by α-averaging view-specific distance matrices such that the best averaged distance matrix is determined by minimizing its α-divergence from view-specific distance matrices. We also present a scalable implementation, exploiting a fast approximate k-NN graph construction method, in which α-averaged distances computed in small partitions determined by recursive spectral bisection are gradually merged in conquer steps until whole examples are used. Numerical experiments on Caltech-256, CIFAR-20, and NUS-WIDE datasets confirm the high performance of our method, in comparison to single-view spectral hashing as well as existing multi-view hashing methods.

1 Introduction

Similarity search, which involves retrieving semantically relevant images given a query, is a core problem in computer vision and information retrieval community. Ever-increasing availability of image data on the Web entails the need of scalable search of relevant images. For example, content-based image retrieval (CBIR) takes an image as a query and returns its nearest neighbors, computing similarity between visual descriptors of the query and of images in database.

A naive solution to nearest neighbor search is linear scan where all items in database are sorted according to their similarity to the query, requiring linear complexity. In practical applications, however, linear scan is not scalable due to the size of examples in database. Approximate nearest neighbor search, which trades accuracy for scalability, becomes more important than ever. Earlier work [1] is a tree-based space partition approach that exploits spatial partitions of data space via various tree structures. While tree-based methods are successful for low-dimensional data, their performance is not satisfactory for high-dimensional data and does not guarantee faster search compared to linear scan [2]. Most of visual descriptors, such as SIFT [3], GIST [4], and HOG [5], constitute high-dimensional vectors, so tree-based space partition approach is not preferred in CBIR applications.

Hashing refers to methods for embedding high-dimensional data into a low-dimensional Hamming space such that similar objects are indexed by binary codes with small Hamming distances. It is categorized into data-independent and data-dependent methods. A notable data-independent method is locality sensitive hashing (LSH) [2, 6] where random projections followed by rounding are used to generate binary codes such that two similar objects in database are shown to have a higher probability of collision. The performance of LSH is not satisfactory when short binary codes are used [7]. Data-dependent hashing methods have drawn extensive attractions recently, where binary codes are learned from visual descriptors for indexing images, in unsupervised [8–10], supervised [11], or semi-supervised [12, 13] manner. Spectral hashing (SH) [8], which we base our method on, is a widely-used unsupervised hashing method, where a subset of eigenvectors of a graph Laplacian is rounded to determine binary codes.

Most of existing data-dependent hashing methods are categorized as *single-view hashing* since only one type of visual descriptors is used to learn a hash function. In practice, images are often described by several different types of visual descriptors such as GIST [4], HOG [5], SIFT [3], to name a few, and each of those descriptors has its own characteristics. Thus, it is desirable to incorporate these heterogenous visual descriptors into learning hash functions, leading to *multi-view hashing*, as shown in Fig. 1. Recently a few methods for multi-view hashing were developed, where spectral hashing was extended for cross-view similarity search [14] and a linear sum of view-specific similarity matrices was exploited [15]. In this paper we present a *sequential spectral learning* approach to multi-view hashing which contains the following contributions which were not explored in previous work:

- We determine a hash function sequentially by solving the successive maximization of local variances subject to decorrelation constraints, which requires only the largest eigenvector of a data-driven matrix at each step, in contrast to existing methods [15, 14] which require the repeated eigen-decomposition or the generalized eigen-decomposition.
- We compute multi-view local variances by α-averaging view-specific distances such that the best averaged distance matrix is determined by minimizing its α-divergence from view-specific distance matrices, whereas the arithmetic mean was only considered in [15].

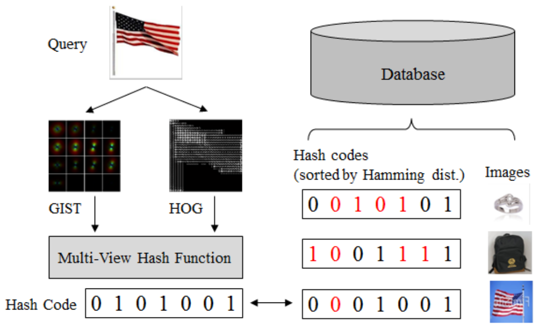

Fig. 1. Pictorial illustration for multi-view hashing. Two visual descriptors (GIST and HOG) are fed into a system where a hash function is determined to accommodate these two types of features, indexing images by integrated binary codes such that Hamming distance between the query and its relevant images is small.

- We also present a scalable implementation, exploiting a fast approximate k-NN graph construction method, in which α-averaged distances computed in small partitions determined by recursive spectral bisection are gradually merged in conquer steps until whole examples are used.

2 Related Work

We briefly review spectral hashing [8] and its two multi-view extension [14, 15]. Suppose that $\{\boldsymbol{x}_i\}_{i=1}^N$ is a set of N objects. We denote object i with K different views by $\{\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(K)}\}$, where each view-specific example is given by $\boldsymbol{x}_i^{(k)} \in \mathbb{R}^{d_k}$ where d_k is the dimension associated with corresponding visual descriptors. Then, the view-specific data matrix is defined as $\boldsymbol{X}^{(k)} = [\boldsymbol{x}_1^{(k)}, \ldots, \boldsymbol{x}_N^{(k)}]$. We also denote by $\boldsymbol{y}_i^{(k)} \in \{-1, +1\}^M$ a binary code of length M associated with $\boldsymbol{x}_i^{(k)}$. Then the binary code matrix is given by $\boldsymbol{Y}^{(k)} = [\boldsymbol{y}_i^{(k)}, \ldots, \boldsymbol{y}_N^{(k)}]$. In the case of single-view hashing, the binary code matrix is represented by $\boldsymbol{Y}$ (without the superscript). Multi-view hashing seeks an integrated binary code matrix $\boldsymbol{Y}^* = \{\boldsymbol{y}_i^*\}_{i=1}^N \in \mathbb{R}^{M \times N}$ which well captures the average similarities between objects across views.

2.1 Spectral Hashing

Spectral hashing [8] counts on a subset of thresholded eigenvectors of the graph Laplacian to seek similarity-preserving compact binary codes. Spectral hashing requires the average Hamming distance between similar neighbors to be minimized. In addition, the codes of length M are also required to be balanced and uncorrelated. Thus, spectral hashing involves the following optimization:

$$\begin{aligned} &\underset{\boldsymbol{Y}}{\arg\min} \sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2, \\ &\text{subject to } \boldsymbol{Y} \in \{-1,+1\}^{M\times N},\ \boldsymbol{Y}\boldsymbol{1}_N = \boldsymbol{0},\ \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^\top = \boldsymbol{I}_M, \end{aligned} \tag{1}$$

where S_{ij} is the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $\|\boldsymbol{y}_i\|_2$ is the Euclidean norm of $\boldsymbol{y}_i$, $\boldsymbol{I}_M \in \mathbb{R}^{M\times M}$ denotes the identity matrix, and $\boldsymbol{1}_N \in \mathbb{R}^N$ is the vector of all ones. The last two constraints represent the balancing condition and pairwise decorrelation condition.

The formulation in spectral hashing is equivalent to a particular form of graph partitioning, which is known to NP-hard. The problem is relaxed by discarding binary constraints $\boldsymbol{y}_i \in \{+1,-1\}^M$. Then rounding a subset of eigenvectors of the graph Laplacian of the similarity graph leads to binary codes for spectral hashing. For out of sample extension, data are assumed to be generated from separable multi-dimensional uniform distribution and eigenfunctions of the weighted Laplace-Beltrami operators defined on manifold are used to determine binary codes of unseen data points.

2.2 Existing Work on Multi-view Hashing

Two recent methods for multi-view hashing are briefly reviewed, including 'composite hashing with multiple information sources' (CHMIS) [15] and 'multi-view hashing for cross-view similarity search' (MVH-CS) [14]. These methods share a similar idea with ours, but it will be emphasized what are differences later.

Suppose that for each view ($k = 1,\ldots,K$), a hash function is parameterized by an embedding matrix $\boldsymbol{W}^{(k)}$ and a bias $\boldsymbol{b}^{(k)}$, so that view-specific binary code is given by

$$\boldsymbol{y}_i^{(k)} = \operatorname{sgn}\left(\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)} + \boldsymbol{b}^{(k)}\right),$$

where sgn$(\cdot)$ is the sign function. We assume that view-specific data are centered, i.e., $\boldsymbol{X}^{(k)}\boldsymbol{1}_N = \boldsymbol{0}$, then the bias $\boldsymbol{b}^{(k)} = -\frac{1}{N}\boldsymbol{W}^{(k)\top}\boldsymbol{X}^{(k)}\boldsymbol{1}_N = \boldsymbol{0}$ is neglected throughout this paper.

CHMIS [15] considers a linear sum of view-specific similarities as an average similarity $S_{ij}^* = \sum_{k=1}^K S_{ij}^{(k)}$ that is plugged into the spectral hashing framework (1):

$$\begin{aligned} &\underset{\boldsymbol{Y}^*}{\arg\min} \sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}^*\|\boldsymbol{y}_i^* - \boldsymbol{y}_j^*\|_2^2, \\ &\text{subject to } \boldsymbol{Y}^*\boldsymbol{1}_N = \boldsymbol{0},\ \frac{1}{N}\boldsymbol{Y}^*\boldsymbol{Y}^{*\top} = \boldsymbol{I}_M,\ \boldsymbol{Y}^* \in \{-1,+1\}^{M\times N}. \end{aligned} \tag{2}$$

For out of sample extension in CHMIS, binary codes of unseen examples are determined by a convex combination of linear hash functions, i.e., $\sum_{k=1}^K \beta_k \boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}$

with $\sum_{k=1}^{K} \beta_k = 1$. Embedding matrices $\boldsymbol{W}^{(k)}$ and coefficients β_k for $k = 1, \ldots, K$ are estimated by solving the following regularized regression problem

$$\sum_{i=1}^{N} \|\boldsymbol{y}_i^* - \sum_{k=1}^{K} \beta_k \boldsymbol{W}^{(k)\top} \boldsymbol{x}_i^{(k)}\|_2^2 + \eta \sum_{k=1}^{K} \|\boldsymbol{W}^{(k)}\|_2^2, \tag{3}$$

where $\boldsymbol{y}_i^*$ computed in (2) are treated as response variables and η is the trade-off parameter for regularizer.

Since $\boldsymbol{y}_i^*$ are coupled with β_k when (2) is solved, spectral hashing (2) and consistency requirements (3) are alternatively optimized, dropping binary constraints for $\boldsymbol{y}_i^*$. CHMIS involves the eigen-decomposition of an N-by-N dense matrix, which is not scalable for a large scale problem. There is no systematic study on how to integrate similarities involving multiple representations of images to estimate the integrated binary codes. This motivates us to investigate various averages to find better average similarity matrix in our method, which is explained in Section 3.2.

MVH-CS [14] is another recently-developed multi-view hashing method which was mainly applied to the problem of cross-view similarity search. The main idea in MVH-CS is to find view-specific binary codes such that similar objects are mapped to similar binary codes across all the views. Once again, assuming a linear hash function $\boldsymbol{W}^{(k)\top} \boldsymbol{x}_i^{(k)}$ for each view, embedding matrices $\boldsymbol{W}^{(k)}$ are determined by minimizing the Hamming distance between the binary codes summed over all the views $d_{ij} = \sum_{k=1}^{K} \sum_{k' \geq k}^{K} \|\boldsymbol{y}_i^{(k)} - \boldsymbol{y}_j^{(k')}\|_2^2$. MVH-CS finds view-specific binary codes instead of integrated binary codes, so the integrated binary code is constructed by concatenating view-specific binary codes, i.e., $\boldsymbol{y}_i^* = [\boldsymbol{y}_i^{(1)}; \boldsymbol{y}_i^{(2)}]$. See [14] for more details.

3 Multi-view Spectral Hashing

This section presents the main contribution of this paper, referred to as *multi-view spectral hashing* (MVSH), where embedding matrices are determined by the successive maximization of local variances with decorrelation constraints satisfied (see Section 3.1) and the α-average is employed to integrate K view-specific distance matrices (see Section 3.2).

3.1 Algorithms for MVSH

Given the α-averaged similarity matrix $\boldsymbol{S}^*$ (which will be described in Section 3.2) which combines K view-specific similarity matrices $\{\boldsymbol{S}^{(k)}\}_{k=1}^{K}$, we begin with the spectral hashing framework:

$$\begin{aligned} &\underset{\boldsymbol{Y}^*}{\arg\min} \ \sum_{i=1}^{N} \sum_{j=1}^{N} S_{ij}^* \|\boldsymbol{y}_i^* - \boldsymbol{y}_j^*\|_2^2, \\ &\text{subject to } \boldsymbol{Y}^* \boldsymbol{1}_N = \boldsymbol{0}, \ \frac{1}{N} \boldsymbol{Y}^* \boldsymbol{Y}^{*\top} = \boldsymbol{I}_M, \ \boldsymbol{Y}^* \in \{-1, +1\}^{M \times N}, \end{aligned} \tag{4}$$

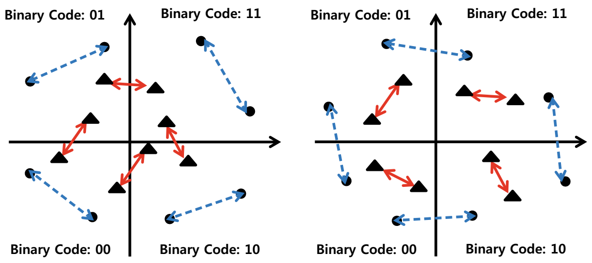

Fig. 2. Pairs of triangles connected by red solid lines are similar ones and pairs connected by blue dotted lines are dissimilar ones in the original space. Both embeddings well preserve locality, however, similarities in the Hamming space are not preserved after binary quantization (left panel), while similarities are still preserved even after binary quantization (right panel). A proper rotation by minimizing the quantization error improves the performance, as demonstrated in [9].

where $\|\boldsymbol{y}_i^* - \boldsymbol{y}_j^*\|_2^2 = \sum_{k=1}^{K} \sum_{k'=1}^{K} \|\boldsymbol{y}_i^{(k)} - \boldsymbol{y}_j^{(k')}\|_2^2$ and $\boldsymbol{y}_i^{(k)} = \mathrm{sgn}\left(\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}\right)$.

We use a linear hash function such that the binary code is computed by $\boldsymbol{y}_i^{(k)} = \mathrm{sgn}\left(\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}\right)$, where $\boldsymbol{W}^{(k)} = [\boldsymbol{w}_1^{(k)}, \dots, \boldsymbol{w}_M^{(k)}] \in \mathbb{R}^{d_k \times M}$. Relaxing binary constraints $\boldsymbol{Y}^* \in \{+1, -1\}^{M\times N}$ by discarding the sgn$(\cdot)$ function, as in [16, 14], the objection function in (4) becomes

$$\sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}^* \left\{ \sum_{k=1}^{K}\sum_{k'=1}^{K} \|\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)} - \boldsymbol{W}^{(k')\top}\boldsymbol{x}_j^{(k')}\|_2^2 \right\}. \tag{5}$$

However, such relaxation might cause an undesirable embedding, as shown in the left panel in Fig. 2.

In order to avoid an undesirable embedding, we now consider the following expansion without dropping the sign function:

$$\|\boldsymbol{y}_i^{(k)} - \boldsymbol{y}_j^{(k')}\|_2^2 = 2M - 2\sum_{m=1}^{M} \mathrm{sgn}\left(\boldsymbol{w}_m^{(k)\top}\boldsymbol{x}_i^{(k)}\right) \mathrm{sgn}\left(\boldsymbol{w}_m^{(k')\top}\boldsymbol{x}_j^{(k')}\right). \tag{6}$$

Note that $\boldsymbol{y}_i^{(k)\top}\boldsymbol{y}_i^{(k)}$ is always M. We now discard the sign function because the terms with the same sign are more favored than the terms with the different sign, regardless of their magnitudes. Neglecting the constant M in (6), we have

$$\begin{aligned} \|\boldsymbol{y}_i^* - \boldsymbol{y}_j^*\|_2^2 &\approx -\sum_{k=1}^{K}\sum_{k'=1}^{K} \mathrm{tr}\left(\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}\boldsymbol{x}_j^{(k')\top}\boldsymbol{W}^{(k')}\right), \\ &= -\mathrm{tr}\left(\left(\sum_{k=1}^{K}\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}\right)\left(\sum_{k'=1}^{K}\boldsymbol{x}_j^{(k')\top}\boldsymbol{W}^{(k')}\right)\right). \end{aligned} \tag{7}$$

This justifies that the integrated binary codes are approximately calculated as $\boldsymbol{y}_i^* \approx \sum_{k=1}^{K}\boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}$.

We define concatenated matrices as $\widetilde{\boldsymbol{W}} = [\boldsymbol{W}^{(1)}; \cdots ; \boldsymbol{W}^{(K)}] \in \mathbb{R}^{(d_1+\cdots+d_K)\times M}$ and $\widetilde{\boldsymbol{X}} = [\boldsymbol{X}^{(1)}; \cdots ; \boldsymbol{X}^{(K)}] \in \mathbb{R}^{(d_1+\cdots+d_K)\times N}$. With these definitions and invoking the approximation (7), we convert the minimization problem (4) to the following maximization problem:

$$\begin{aligned} &\underset{\widetilde{\boldsymbol{W}}}{\arg\max}\ \operatorname{tr}\left(\widetilde{\boldsymbol{W}}^\top \widetilde{\boldsymbol{X}} \boldsymbol{S}^* \widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{W}}\right), \\ &\text{subject to } \ \widetilde{\boldsymbol{W}}^\top \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{W}} = \boldsymbol{I}_M, \end{aligned} \tag{8}$$

where the solution $\widetilde{\boldsymbol{W}}$ is determined by solving the generalized eigenvalue problem for the matrix pencil $(\widetilde{\boldsymbol{X}} \boldsymbol{S}^* \widetilde{\boldsymbol{X}}^\top, \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top)$. This method is referred to as GE-MVSH.

GE-MVSH does not work well in the case where binary code length is large, because the relaxed decorrelation constraint $\widetilde{\boldsymbol{W}}^\top \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{W}} = \boldsymbol{I}_M$ deteriorates the performance. Good hash functions are obtained from the directions that have large variance [12], but GE-MVSH minimizes the global variance, so that the performance decreases as the code size increases. When naively discarding the sign function of the decorrelation condition, the diagonal entries lead to minimize the global variances. Considering the sign function, the diagonal entries in the decorrelation constraints are always satisfied, since $\operatorname{sgn}(\widetilde{\boldsymbol{w}}_m^\top \widetilde{\boldsymbol{x}}_i) \operatorname{sgn}(\widetilde{\boldsymbol{w}}_m^\top \widetilde{\boldsymbol{x}}_i) = 1$ for $m = 1, \ldots, M$. Thus we now consider only off-diagonal entries in the constraints $\frac{1}{N}\boldsymbol{Y}^*\boldsymbol{Y}^{*\top} = \boldsymbol{I}$, which require

$$\operatorname{sgn}(\widetilde{\boldsymbol{w}}_i^\top \widetilde{\boldsymbol{X}}) \operatorname{sgn}(\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{w}}_j) = 0, \ \text{for } i = 2, ..., M \text{ and } j = 1, ..., i-1. \tag{9}$$

Plugging (9) into a penalty term of objective function in (8), we determine $\widetilde{\boldsymbol{w}}_i$ one by one by successively solving

$$\underset{\widetilde{\boldsymbol{w}}_i}{\arg\max}\, \widetilde{\boldsymbol{w}}_i^\top \widetilde{\boldsymbol{X}} \boldsymbol{S}^* \widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{w}}_i - \mu \sum_{j=1}^{i-1} \lambda^{i-j} (\widetilde{\boldsymbol{w}}_i^\top \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{w}}_j)^2, \tag{10}$$

subject to $\widetilde{\boldsymbol{w}}_i^\top \widetilde{\boldsymbol{w}}_i = 1$, where μ is a trade-off parameter and λ is a decaying parameter to de-emphasize the decorrelation requirements for the current weight vector and earlier-computed weight vectors. The sequential maximization (10) is solved by computing the largest eigenvector of the adjusted local covariance matrix:

$$\widetilde{\boldsymbol{X}} \boldsymbol{S}^* \widetilde{\boldsymbol{X}}^\top - \mu \sum_{j=1}^{i-1} \lambda^{i-j} \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{w}}_j \widetilde{\boldsymbol{w}}_j^\top \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top. \tag{11}$$

This method is referred to as SU-MVSH, which is summarized in Algorithm 1. Note that our sequential formulation is different from [17], even if both methods are designed to compute binary codes in a sequential manner. In [17], each new hash function is learned to correct the errors made by the previous hash functions, using (pseudo-)label information. However, our formulation does not require label information, because it directly approximates the decorrelation condition.

Algorithm 1. Sequential Update for Multi-View Spectral Hashing (SU-MVSH)

```
Input: training data as $\widetilde{\boldsymbol{X}} = [\boldsymbol{X}^{(1)}; \ldots; \boldsymbol{X}^{(K)}]$, test data point $\widehat{\boldsymbol{x}} = [\widehat{\boldsymbol{x}}^{(1)}; \ldots; \widehat{\boldsymbol{x}}^{(K)}]$,
    $\alpha$-value and $M$ (binary code length).
Output: binary code $\widehat{\boldsymbol{y}}$ associated with $\widehat{\boldsymbol{x}}$
1: Construct multi-view local covariance: $\widetilde{\boldsymbol{X}}\boldsymbol{S}^*\widetilde{\boldsymbol{X}}^\top$, where $S^*_{ij} = \exp(-\frac{1}{\sigma^{*2}} D^{*2}_{ij})$, $D^*_{ij}$
   is $\alpha$-average distance (see Section 3.2) and $\sigma$ is set by local-scaling [18].
2: $\widetilde{\boldsymbol{w}}_1 \leftarrow$ the largest eigenvector of $\widetilde{\boldsymbol{X}}\boldsymbol{S}^*\widetilde{\boldsymbol{X}}^\top$.
3: for $i = 2, \ldots, M$ do
4:     $\widetilde{\boldsymbol{w}}_i \leftarrow$ the largest eigenvector of the adjusted local covariance: $\widetilde{\boldsymbol{X}}\boldsymbol{S}^*\widetilde{\boldsymbol{X}}^\top -$
       $\mu \sum_{j=1}^{i-1} \lambda^{i-j} \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{w}}_j \widetilde{\boldsymbol{w}}_j^\top \widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{X}}^\top$.
5: end for
6: Return $M$-bit integrated binary code: $\widehat{\boldsymbol{y}} = \mathrm{sgn}(\widetilde{\boldsymbol{W}}^\top \widehat{\boldsymbol{x}})$, where $\widetilde{\boldsymbol{W}} = [\widetilde{\boldsymbol{w}}_1, \ldots, \widetilde{\boldsymbol{w}}_M]$.
```

3.2 α-Average Similarity

One of core components in MVSH is to construct an average similarity matrix which well integrates K view-specific similarity matrices. To this end, we adopt the α-average [19] which includes various widely-used averages (such as arithmetic, geometric, harmonic means) as its special cases.

Given K view-specific distance matrices, $\boldsymbol{D}^{(1)}, \ldots, \boldsymbol{D}^{(K)}$, we compute the *average distance matrix* $\boldsymbol{D}^*$ by minimizing the α-divergence from view-specific distance matrices,

$$\arg\min_{\boldsymbol{D}^*} \mathcal{J}[\boldsymbol{D}^*] = \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} w_k D_\alpha[D^{(k)}_{ij} \| D^*_{ij}], \tag{12}$$

where w_k is the scaling factor for k-th view ($w_k \geq 0$), $D^{(k)}_{ij} = \|\boldsymbol{x}^{(k)}_i - \boldsymbol{x}^{(k)}_j\|_2$, and $D_\alpha[a\|b]$ is the α-divergence between two positive numbers a and b, defined as

$$D_\alpha[a\|b] = \begin{cases} b - a + a\log\frac{a}{b}, & \alpha = -1, \\ a - b + b\log\frac{b}{a}, & \alpha = 1, \\ \frac{2}{1+\alpha}a + \frac{2}{1-\alpha}b - \frac{4}{1-\alpha^2} a^{\frac{1-\alpha}{2}} b^{\frac{1+\alpha}{2}}, & \alpha \neq \pm 1. \end{cases}$$

As shown in [19], the solution $\boldsymbol{D}^*$ to the problem (12) is given by

$$D^*_{ij} = f_\alpha^{-1}\left(\sum_{k=1}^{K} w_i f_\alpha\left(D^{(k)}_{ij}\right)\right), \tag{13}$$

where $f_\alpha(x)$ is a differentiable monotonic function given by

$$f_\alpha(x) = \begin{cases} x^{\frac{1-\alpha}{2}}, \text{ for } \alpha \neq 1, \\ \log x, \text{ for } \alpha = 1. \end{cases} \tag{14}$$

In practice, the weight factor, w_k, is chosen as $1/\sigma_k$, where σ_k is the median of pairwise distances of k-th view (due to the different scale of distance for k-th view). This α-average distance is used for SU-MVSH (algorithm 1).

The α-average (13) reduces to the maximum, arithmetic mean, geometric mean, harmonic mean, and minimum when $\alpha = -\infty, -1, 1, 3, \infty$, respectively. Note that as α increases, the smaller values become more important, while as α decreases, the larger values are taken more seriously. Note also that when $\alpha = -1$ with uniform scaling factors and Euclidean distance is used for view-specific distance, the α-average distance is equivalent to the Euclidean distance between concatenated data over all views. The α-average distance matrix was originally used in manifold integration [20].

We observe that the performance of proposed method (SU-MVSH) is changed with respect to various α values on three datasets (Caltech-256, CIFAR-20, and NUS-WIDE) in Fig. 3. We observe that $\alpha \geq 1$ achieves the best performance, leading that higher α value is proper in this case since it is natural to put more weight on small distance. Note that we have the better averaged distance rather than simple feature concatenation ($\alpha = -1$).

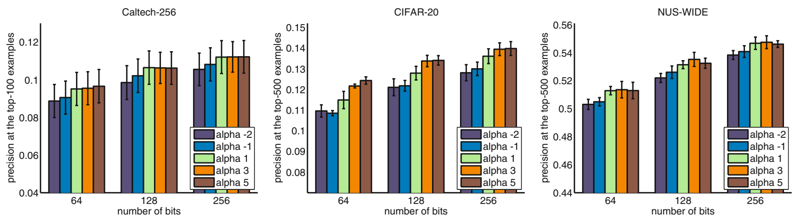

Fig. 3. Performance of SU-MVSH with respect to various α values. Detailed experimental settings are found in section 4.

3.3 Scalable Implementation

The direct application for α-average to large-scale distance matrices is infeasible. Since full distance matrix over all pairs of examples requires $O(N^2)$ of space and $O(dN^2)$ of computation time (d is the data dimension), we use approximate k-NN graph based on recursive spectral bisection [21], which runs in divide-and conquer fashion as follows. The data is split into two overlapping subsets recursively until the size of subset is sufficiently small to compute the exact k-NN graph, then conquer the results (k-NN graphs of the subsets) to a final approximate k-NN graph.

The straightforward way to construct α-average k-NN graph is to compute approximate k-NN graphs for each view and to apply α-average process into the k-NN graphs. This naïve implementation leads the α-average k-NN graph of which most elements are zero, for $\alpha = 1$, since we perform element-wise product of highly sparse k-NN graphs. To avoid the undesirable α-average k-NN graph, we propose small modifications (yet, effective) in the divide step to incorporate α-average process into approximate k-NN based on spectral bisection.

In the divide step, data is split into two overlapping subsets using the principal direction of the data matrix concatenating all views, which can be computed linearly with respect to N and d [1]. Given the concatenated data points, $\widetilde{\boldsymbol{X}} = [\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_N]$, we compute the largest singular triplet $(\sigma, \boldsymbol{u}, \boldsymbol{v})$ of $\widetilde{\boldsymbol{X}}$ with $\boldsymbol{u}^\top \widetilde{\boldsymbol{X}} = \sigma \boldsymbol{v}^\top$. The Lanzos method is used to compute an well-approximated largest singular vector with $O(dN)$ time complexity [21]. Then the data is split into two overlapped subsets:

$$\widetilde{\boldsymbol{X}}_+ = \{\widetilde{\boldsymbol{x}}_i | \boldsymbol{u}^\top \widetilde{\boldsymbol{x}}_i \geq -c(r)\}, \quad \widetilde{\boldsymbol{X}}_- = \{\widetilde{\boldsymbol{x}}_i | \boldsymbol{u}^\top \widetilde{\boldsymbol{x}}_i < c(r)\}, \tag{15}$$

where $c(r)$ is a constant to retain that $(100r)\%$ of the elements to be overlapped in both $\widetilde{\boldsymbol{X}}_+$ and $\widetilde{\boldsymbol{X}}_-$. When the subset is small enough to compute the exact k-NN in brute-force way, we apply α-average into the pair-wise distance matrices of view-specific data, which leads to the α-average k-NN graph of the subset.

In the conquer step, we use the same conquer step as in [21]. Two k-NN graphs of each subset is merged into a larger k-NN graph in the following way: if a data point is overlapped in both subsets $\boldsymbol{X}_+$ and $\boldsymbol{X}_-$, then its k-nearest neighbors are chosen from its neighbors in each subset (a further refinement step can be applied) [21]. We gradually merge the two k-NN graphs into a larger one until a final approximate k-NN graph is constructed.

By this simple modification, we can avoid having too many zeros in the final approximate α-average k-NN graph. The time complexity remains the same with the original approximate k-NN construction, which is $O(dN^t)$, where $t \in (1, 2)$ is governed by a parameter r. The parameter r controls how many points are overlapped in the divide step. Although the algorithm gets slower as r increases, it runs fast enough with moderate values of r ($r = 0.2$, in our experiment). We use α-average approximate k-NN graph for the α-average distance matrix $\boldsymbol{D}^*$ in algorithm 1.

4 Experiments

We evaluate our methods on three different datasets: Caltech-256 [22], CIFAR-20 [23], and NUS-WIDE [24]. Caltech-256 consists of 29,780 images, each of which is associated with one of 256 object categories. CIFAR-20, which is a subset of 80-million tiny images [25], contains 60,000 images with 20 coarse labels. In each of these two datasets, we form a query set by randomly choosing 1,000 images and construct a training set using the rest of images. We use GIST [4] (512-dimension for Caltech, 384-dimension for CIFAR) and HOG [5] (2048-dimension for Caltech, 1152-dimension for CIFAR) descriptors to produce two views of each image.

- GIST: For Caltech-256, we apply Gabor filters with 8 different orientations and 4 scales. Each Gabor-filtered image is averaged over 4-by-4 grid [7], which results in a 512-dimensional vector ($8 \times 4 \times 16 = 512$). For CIFAR-20, we use Gabor filters with 8 different orientations and 3 scales, which results in 384-dimensional vector.

[1] CCA cannot be used in this case, because the time complexity is not linear with respect to N and d.

– HOG: For both Caltech and CIFAR datasets, we compute the image gradients of non-overlapping windows, where the orientation of gradients is quantized into 8 bins and normalized with 4 different metrics, as in [5]. For Caltech-256, we resize each image into 64-by-64, leading to 64 6-by-6 non-overlapping windows, which results in a 2048-dimensional vector ($8 \times 4 \times 64 = 2048$). For CIFAR-20, 36 4-by-4 non-overlapping windows yield 1152-dimensional vectors.

NUS-WIDE contains 270,000 images with 81 concept classes. As in [10], we only use the most frequent 21 concept classes. In such a case, there exist unlabeled data, but we put them into the training set (even if the unlabeled data are not used, the trends in all figures remain the same). We use 2,100 images (100 images per class) as the query set, and the rest of images are used as the training set. NUS-WIDE provides 6 different features (bag-of-words, wavelet coefficient, color histogram, and etc.), so each of which is treated as a single view, leading to 6 different views.

We compare our methods (SU-MVSH and GE-MVSH) with three existing multi-view hashing algorithms (MVH-CS, MVH-CCA and CHMIS) as well as spectral hashing with features concatenated over multiple views. MVH-CS [14] and CHMIS [15] were briefly reviewed in Section 2.2 and note that MVH-CCA is a special case of MVH-CS when the averaged similarity matrix fixed as the identity matrix. For SU-MVSH, we set $\lambda = 0.9$ and choose μ as one of values in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ which yields the best performance by 5-fold cross-validation. For SU-MVSH, GE-MVSH and MVH-CS, we employ scalable implementation to calculate the α-average (with $k = 10$), described in Section 3.3, where the value of $\alpha \in \{-2, -1, 1, 3, 5\}$ is determined by 5-fold cross-validation. For CHMIS, we use an approximate k-NN graph method ($k = 10$) [21]. Since CHMIS is not scalable for large-scale data, we use a subset of training set (10,000 examples) to learn hash functions. All experiments are repeated five times to produce error bars in Fig. 4 and 5.

As a performance measure, we use Hamming ranking [12, 10] where rankings are measured by Hamming distance between query and data points. We calculate the precision at the top 100 examples for Caltech-256, and at the top 500 examples for other datasets, as shown in Fig. 5. A tie breaks at random,

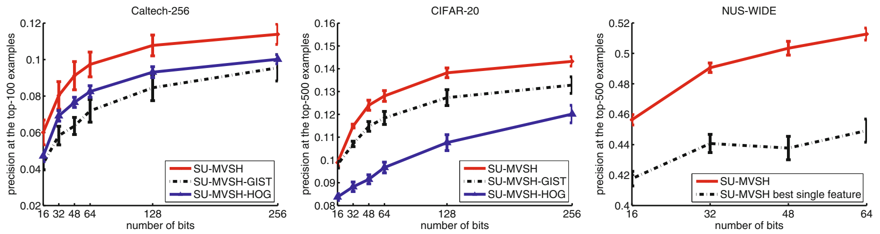

Fig. 4. Performance of SU-MVSH when a single descriptor is used, or both descriptors are used together via the α-average

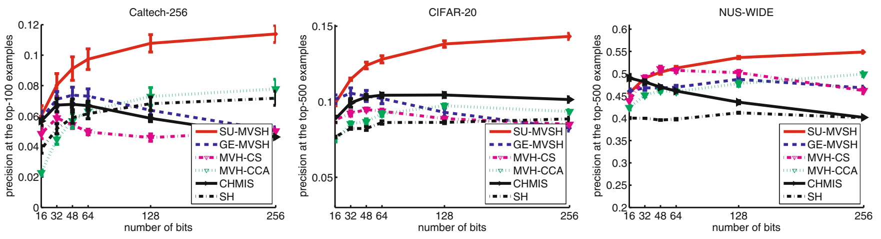

Fig. 5. Performance comparison with respect to different number of bits

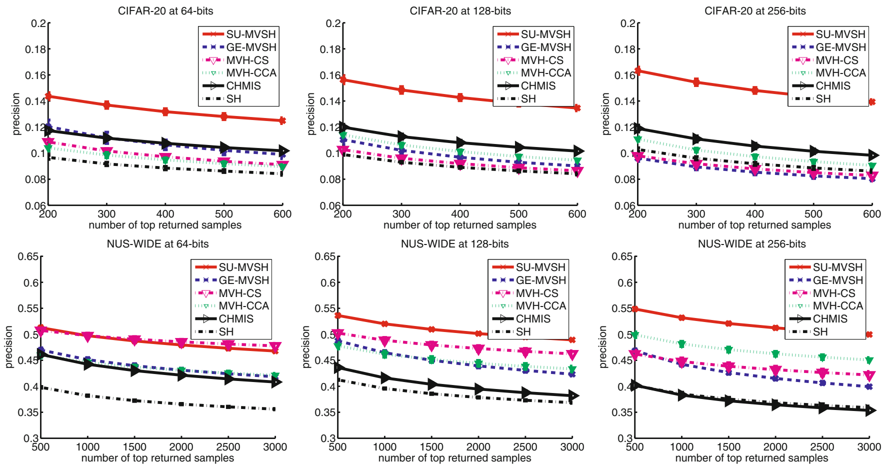

Fig. 6. Precision when the number of top returned images varies from 200 to 600 (or 500 to 3000), for various code sizes (64, 128, and 256 bits)

in the Hamming distance. Fig. 4 shows the performance of SU-MVSH when a single descriptor is used, or both descriptors are used together via the α-average, demonstrating that integrating multiple descriptors improves the performance. As shown in Fig. 5, our method SU-MVSH outperforms GE-MVSH as well as existing multi-view hashing methods (MVH-CS, MVH-CCA and CHMIS) and spectral hashing with concatenated descriptors, especially when the code size is large, since the decorrelation condition is better considered in SU-MVSH, compared to any other methods in comparison. Fig. 6 shows the precision when the number of top returned images varies from 200 to 600 (or 500 to 3000), for various code sizes (64, 128, and 256 bits). In most of cases, SU-MVSH shows the best performance. More experiments can be found in the longer version [26].

5 Conclusions

We have presented sequential updating algorithm for multi-view spectral hashing (SU-MVSH), where hash functions are sequentially determined by the maximization of local variance subject to the decorrelation condition. We incorporated multiple visual descriptors into the best average distance matrix determined by minimizing its α-divergence from view-specific distance matrices. We also presented a scalable implementation to construct α-average k-NN graph, exploiting spectral bisection. Numerical experiments validated the usefulness of the proposed method, compared to the multi-view hashing methods as well as single-view spectral hashing.

Acknowledgments. This work was supported by NIPA-MSRA Creative IT/SW Research Project, MEST Converging Research Center Program (2012K001343), NIPA ITRC Support Program (NIPA-2012-H0301-12-3002), MKE and NIPA "IT Consilience Creative Program" (C1515-1121-0003), and NRF WCU Program (R31-10100).

References

1. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Softwares 3, 209–226 (1977)
2. Gionis, A., Indyk, P., Motawani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the International Conference on Very Large Data Bases, VLDB (1999)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60, 91–110 (2004)
4. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision 42, 145–175 (2001)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA (2005)
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.: Locality sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Annual ACM Symposium on Computational Geometry, SoCG (2004)
7. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large image databases for recognition. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, Alaska (2008)
8. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems (NIPS), vol. 20. MIT Press (2008)
9. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO (2011)
10. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: Proceedings of the International Conference on Machine Learning (ICML), Bellevue, WA (2011)
11. Salakhutdinov, R., Hinton, G.: Semantic hashing. In: Proceeding of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models (2007)
12. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA (2010)

13. Kim, S., Choi, S.: Semi-supervised discriminant hashing. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), Vancouver, Canada (2011)
14. Kumar, S., Udupa, R.: Learning hash functions for cross-view similarity search. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Spain (2011)
15. Zhang, D., Wang, F., Si, L.: Composite hashing with multiple information sources. In: Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Beijing, China (2011)
16. He, J., Radhakrishnan, R., Chang, S.F., Bauer, C.: Compact hashing with joint optimization of search accuracy and time. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO (2011)
17. Wang, J., Kumar, S., Chang, S.F.: Sequential projection learning for hashing with compact codes. In: Proceedings of the International Conference on Machine Learning (ICML), Haifa, Israel (2010)
18. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: Advances in Neural Information Processing Systems (NIPS), vol. 17, pp. 1601–1608. MIT Press (2005)
19. Amari, S.: Integration of stochastic models by minimizing α-divergence. Neural Computation 19, 2780–2796 (2007)
20. Choi, H., Choi, S., Katake, A., Kang, Y., Choe, Y.: Manifold Alpha-Integration. In: Zhang, B.-T., Orgun, M.A. (eds.) PRICAI 2010. LNCS, vol. 6230, pp. 397–408. Springer, Heidelberg (2010)
21. Chen, J., Fang, H.R., Saad, Y.: Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection. Journal of Machine Learning Research 10, 1989–2012 (2009)
22. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical report, Caltech (2007)
23. Krizhevsky, A., Hinton, G.E.: Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto (2009)
24. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: NUS-WIDE: a real-world web image database from national university of singapore. In: Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR), Santorini, Greece (2009)
25. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large dataset for non-parametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 1958–1970 (2008)
26. Kim, S., Kang, Y., Choi, S.: Sequential spectral learning to hash with multiple representations. Technical Report POSTECH-MLG-2012-005, Machine Learning Group, Department of Computer Science and Engineering, POSTECH (2012)