Kernel-Based Supervised Discrete Hashing for Image Retrieval

Xiaoshuang Shi, Fuyong Xing, Jinzheng Cai, Zizhao Zhang,
Yuanpu Xie, and Lin Yang

University of Florida, Gainesville, FL, 32611, USA
xsshi2015@ufl.edu

Abstract. Recently hashing has become an important tool to tackle the problem of large-scale nearest neighbor searching in computer vision. However, learning discrete hashing codes is a very challenging task due to the NP hard optimization problem. In this paper, we propose a novel yet simple kernel-based supervised discrete hashing method via an asymmetric relaxation strategy. Specifically, we present an optimization model with preserving the hashing function and the relaxed linear function simultaneously to reduce the accumulated quantization error between hashing and linear functions. Furthermore, we improve the hashing model by relaxing the hashing function into a general binary code matrix and introducing an additional regularization term. Then we solve these two optimization models via an alternative strategy, which can effectively and stably preserve the similarity of neighbors in a low-dimensional Hamming space. The proposed hashing method can produce informative short binary codes that require less storage volume and lower optimization time cost. Extensive experiments on multiple benchmark databases demonstrate the effectiveness of the proposed hashing method with short binary codes and its superior performance over the state of the arts.

Keywords: Supervised kernel hashing, discrete constraint, accumulated quantization error reduction

1 Introduction

Over the past decade, hashing has attracted considerable attentions in computer vision [1] [2] [3] [4] and machine learning [5] [6] communities. With the increasing of visual data including images and videos, it is favorable to apply compact hashing codes to data storing and content searching. Basically hashing encodes each high-dimensional data into a set of binary codes and meanwhile preserves the similarity between neighbors. Recent literature reports that when each image is encoded into several tens of binary bits, the storage of one hundred million images requires only less than 1.5 GB [7] and searching in a collection of millions of images costs a constant time [8] [9].

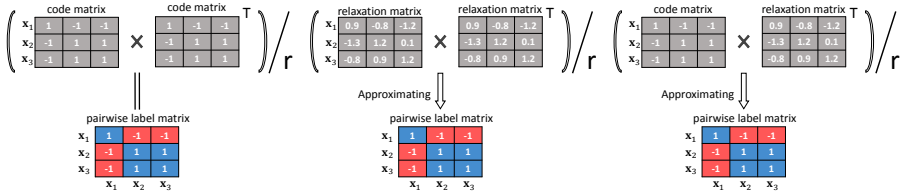
Nowadays many hashing methods have been proposed. Based on whether semantic information is considered, they can be grouped into two major categories: unsupervised and supervised. Unsupervised hashing methods aim to

explore the intrinsic structure of data to preserve the similarity of neighbors without any supervision. Local sensitive hashing (LSH)[10] is one of the most popular unsupervised hashing approaches and has been applied to tackling many large-scale data problems. However, LSH is a data-independent method that uses random projection to generate binary codes, thereby requiring long binary codes to achieve satisfactory retrieval accuracy. On the other hand, many data-dependent methods, such as spectral hashing (SH) [6], anchor graph hashing (AGH) [11] and iterative quantization (ITQ) [12], have been proposed to learn compact codes and achieve promising retrieval performance.

Due to the semantic gap [13], however, unsupervised hashing methods are not able to guarantee good retrieval accuracy via semantic distances. Therefore, supervised hashing methods, which utilize semantic information to map high-dimensional data into a low-dimensional Hamming space [14], are developed to improve the performance. Supervised hashing methods can be divided into linear and nonlinear categories. Several popular supervised linear hashing methods are: linear discriminant analysis hashing (LDAHash) [15] that projects descriptor vectors into the Hamming space; the minimal loss hashing (MLH) [16] utilizes structured prediction with latent variables; the semi-supervised hashing (SSH) [9] leverages the Hamming distance between pairs, etc. Compared to linear hashing methods, nonlinear methods like binary reconstruction embedding (BRE) [14] and kernel-based supervised hashing (KSH) [17] often generate more effective binary codes because of the usage of the nonlinear structure hidden in the data.

KSH is a very popular supervised nonlinear method, which can achieve stable and encouraging retrieval accuracy in various applications. However, it has two major issues: (i) Its objective function is NP-hard so that it is difficult to be directly solved; (ii) The relaxation used might result in a large accumulated quantization error between the hashing and linear projection functions such that the retrieval accuracy can deteriorate rapidly with increasing number of training data [18]. Recently, discrete graph hashing (DGH) [7], asymmetric inner-product binary coding (AIBC) [19] and supervised discrete hashing (SDH) [18] have demonstrate that with discrete constraints preserved, hashing methods can directly work in the discrete code space so that their retrieval accuracy can be boosted. Furthermore, although the greedy algorithm can reduce the accumulated quantization error, it is computationally expensive and usually requires relatively long binary codes to obtain the desired retrieval accuracy.

In this paper, we propose a novel hashing framework, kernel-based supervised discrete hashing (KSDH), that can provides competitive retrieval accuracy with short binary codes and low training time cost (The core idea and the difference to KSH are shown in Fig. 1). Specifically, in order to reduce the accumulated quantization error between hashing and linear projection functions, we replace the element-wise product of hashing functions with the element-wise product between hashing and linear projection functions. Furthermore, we improve the model by relaxing the hashing function into a general binary code matrix and introducing an additional regularization term, to attain stable and better re-



(a) Hashing (b) KSH (c) Ours

Fig. 1. The core idea of the proposed hashing method and its difference to KSH. (a) The standard hashing uses the element-wise product of the ideal code matrix to fit the pairwise label matrix. (b) The KSH uses the symmetric relaxation, the element-wise product of the relaxation matrix, to approximate the pairwise label matrix. (c) Our algorithm utilizes an asymmetric relaxation, the element-wise product between the code and relaxation matrices, to approximate the pairwise label matrix.

trieval accuracy. We apply an alternative and efficient strategy to the model optimization with a very low time cost. Our contributions are summarized as follows:

- We propose a novel yet simple kernel-based supervised discrete hashing framework via the asymmetric relaxation strategy that can preserve the discrete constraint and reduce the accumulated quantization error between binary code matrix and linear projection functions. In addition, to the best of our knowledge, there exist no method using the asymmetric relaxation strategy to improve the retrieval accuracy of KSH.
- We solve the optimization models in an alternative and efficient manner, and analyze the convergence and time complexity of the optimization procedure.
- We evaluate the proposed framework on four popular large-scale image and video databases, and achieve superior performance over the state of the arts, especially with short binary codes.

2 Kernel-based Supervised Hashing

In this section, we briefly review the related work KSH [17] by which we are inspired. Given a set of N data points and randomly selected n points $\mathbf{X} \in \mathbb{R}^{n \times d}$, $n << N$, the similar pairs (neighbors in terms of a metric distance or sharing the same label) are collected in the set $\mathcal{M}$ and the dissimilar pairs (non-neighbors or with different labels) are collected in the set $\mathcal{C}$. Let $\phi : \mathbb{R}^d \mapsto T$ be a kernel mapping from the original space to the kernel space, where T is a Reproducing Kernel Hilbert Space (RKHS) with a kernel function $\kappa(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$. In order to obtain the compact representation of each data point and preserve the similarity of pairs, KSH aims to look for r hashing functions to project the data $\mathbf{X}$ into a Hamming space. With m points selected from $\mathbf{X}$ and a projection

matrix $\mathbf{A} \in \mathbb{R}^{r\times m}$, the k-th ($1 \le k \le r$) hashing function of KSH is defined as:

$$h_k(\mathbf{x}) = sgn(\sum_{j=1}^{m} \kappa(\mathbf{x}_j, \mathbf{x})a_{jk} - b_k) = sgn(\mathbf{a}_k\bar{\kappa}(\mathbf{x})), \tag{1}$$

where $\mathbf{x} \in \mathbf{X}$ and $b_k = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\kappa(\mathbf{x}_j, \mathbf{x}_i)a_{jk}$. Eq. (1) implies a balanced hashing function constraint that is $\sum_{i=1}^{n} h_k(\mathbf{x}_i) = 0$.

Based on Eq. (1), $h_k(\mathbf{x}) \in \{-1, 1\}$, KSH attempts to learn the projection matrix $\mathbf{A} \in \mathbb{R}^{r\times m}$ such that $h_k(\mathbf{x}_i) = h_k(\mathbf{x}_j)$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$, and $h_k(x_i) \neq h_k(x_j)$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$. Let the r-bit hash code of each point $\mathbf{x}$ be $code_r(\mathbf{x}) = [h_1, h_2, \cdots, h_r]$. Then, if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}$, $code_r(\mathbf{x}_i) \circ code_r(\mathbf{x}_j) = r$; otherwise, $code_r(\mathbf{x}_i) \circ code_r(\mathbf{x}_j) = -r$, where $\circ$ represents the code inner product. In order to obtain the hashing function, the weight (pairwise label) matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ is defined as:

$$s_{ij} = \begin{cases} 1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \\ -1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \\ 0 & otherwise \end{cases} \tag{2}$$

Since $code_r(\mathbf{x}_i) \circ code_r(\mathbf{x}_j) \in [-r, r]$ and $s_{ij} \in [-1, 1]$, KSH learns the projection matrix by solving the following optimization model:

$$\min_{\mathbf{H}\in\{-1,1\}} \left\|\mathbf{H}^T\mathbf{H} - r\mathbf{S}\right\|_F^2 = \min_{\mathbf{A}} \left\|sgn(\mathbf{A}\bar{\mathbf{K}})^T sgn(\mathbf{A}\bar{\mathbf{K}}) - r\mathbf{S}\right\|_F^2, \tag{3}$$

where $\mathbf{H} = sgn(\mathbf{A}\bar{\mathbf{K}}) = [code_r(x_1), \cdots, code_r(x_n)] \in \mathbf{R}^{r\times n}$ denotes the code matrix produced by hashing functions, and $\bar{\mathbf{K}} \in \mathbb{R}^{m\times n}$ is a kernel matrix with zero-mean. There is one implied condition: $\mathbf{H}\mathbf{1}_n = 0$ in Eq. (3), where $\mathbf{1}_n \in \mathbb{R}^n$ is a column vector with all elements equal to one. This condition maximizes the information from each bit.

3 Kernel-based Supervised Discrete Hashing (KSDH)

3.1 KSDH with Hashing Function Preserved

Since the optimization problem in Eq. (3) is non-differential and NP-hard, KSH [17] adopts the symmetric relaxation and greedy strategy to approximate the weight matrix rS, but it might produce a large accumulated quantization error between hashing $sgn(\mathbf{A}\bar{\mathbf{K}})$ and linear projection $\mathbf{A}\bar{\mathbf{K}}$ functions, which can significantly affect the effectiveness of hashing functions, especially for the large number of training data [17]. To learn a discrete matrix and reduce the accumulated quantization error, based on the asymmetric relaxation strategy, we propose a novel optimization model as follows:

$$\begin{aligned} &\min_{\mathbf{A}} \left\|\mathbf{H}^T\mathbf{A}\bar{\mathbf{K}} - r\mathbf{S}\right\|_F^2, \\ &s.t.\ \mathbf{A}\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}^T = n\mathbf{I}_r,\ \mathbf{H} = sgn(\mathbf{A}\bar{\mathbf{K}}). \end{aligned} \tag{4}$$

Note that the hashing function $\mathbf{H}$ is preserved in the objective function. Usually, the smaller quantization error between $\mathbf{A}\bar{\mathbf{K}}$ and $sgn(\mathbf{A}\bar{\mathbf{K}})$, the smaller

reconstruction error of the objective function. We add one more constraint, $\mathbf{A}\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}^T = n\mathbf{I}_r$ that is derived from the constraint $\mathbf{H}\mathbf{H}^T = n\mathbf{I}_r$, which enforces r bit hashing codes to be mutually uncorrelated such that the redundancy among these bits is minimized [6] [7]. In addition, the constraint $\mathbf{A}\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}^T = n\mathbf{I}_r$ can also reduce the redundancy among data points [20].

Since $Tr\left\{\bar{\mathbf{K}}^T\mathbf{A}^T\mathbf{H}\mathbf{H}^T\mathbf{A}\bar{\mathbf{K}}\right\}$ and $Tr\left\{\mathbf{S}^T\mathbf{S}\right\}$ are constant, the optimization problem in Eq. (4) is equivalent to the following optimization problem:

$$\begin{aligned} &\underset{\mathbf{A}}{max}\ Tr\left\{\mathbf{H}\mathbf{S}\bar{\mathbf{K}}^T\mathbf{A}^T\right\}, \\ &s.t.\ \mathbf{A}\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}^T = n\mathbf{I}_r, \mathbf{H} = sgn(\mathbf{A}\bar{\mathbf{K}}). \end{aligned} \tag{5}$$

Eq. (5) is clearly different from the objective function in [7] with two major differences: (i) Eq. (5) aims to learn the projection matrix $\mathbf{A}$ for supervised hashing, while [7] is an unsupervised method; (ii) The discrete constraint in Eq. (5) is asymmetric, while the objective function of [7] adopts symmetric discrete constraints. Eq. (5) is also different from the objective function in [21], which relaxes the hashing functions into a continuous set $[-1, 1]$. In the following, we will explain our proposed procedure to solve this optimization problem.

To solve Eq. (5), we introduce an auxiliary variable $\mathbf{C}$ and let $\mathbf{C} = \mathbf{A}\bar{\mathbf{K}}$, and then $\mathbf{C}\mathbf{1}_n = 0$ due to $\bar{\mathbf{K}}\mathbf{1}_n = 0$ (the implicit constraint). The model in Eq. (5) can be rewritten as:

$$\begin{aligned} &\underset{\mathbf{C}}{max}\ Tr\left\{\mathbf{H}\mathbf{S}\mathbf{C}^T\right\}, \\ &s.t.\ \mathbf{C}\mathbf{C}^T = n\mathbf{I}_r, \mathbf{C}\mathbf{1}_n = 0, \mathbf{H} = sgn(\mathbf{C}). \end{aligned} \tag{6}$$

After obtaining $\mathbf{C}$, the projection matrix $\mathbf{A}$ is calculated by $\mathbf{A} = \mathbf{C}\bar{\mathbf{K}}^T(\bar{\mathbf{K}}\bar{\mathbf{K}}^T)^{-1}$. In practice, we obtain $\mathbf{A}$ by $\mathbf{A} = \mathbf{C}\bar{\mathbf{K}}^T(\bar{\mathbf{K}}\bar{\mathbf{K}}^T + \epsilon\mathbf{I}_m)^{-1}$ to attain a stable solution. An alternative optimization strategy is used to solve Eq. (6). The detailed steps are shown in the following.

Fix H and update C: With $\mathbf{H}$ fixed, the optimization problem in Eq. (6) becomes:

$$\begin{aligned} &\underset{\mathbf{C}}{max}\ Tr\left\{\mathbf{H}\mathbf{S}\mathbf{C}^T\right\}, \\ &s.t.\ \mathbf{C}\mathbf{C}^T = n\mathbf{I}_r, \mathbf{C}\mathbf{1}_n = 0. \end{aligned} \tag{7}$$

whose solution is shown in Proposition 1.

Proposition 1: Suppose the rank of the matrix $\mathbf{S}$ is c, $\mathbf{C} = \sqrt{n}\mathbf{V}\left[\mathbf{U}, \bar{\mathbf{U}}\right]^T$ is an optimal solution of Eq. (7). $\mathbf{V} \in \mathbb{R}^{r\times r}$ can be obtained by applying singular value decomposition (SVD) to $\mathbf{H}\mathbf{S}\mathbf{J}\mathbf{S}\mathbf{H}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$, $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$. If $r \le c$, $\mathbf{U} = \mathbf{J}\mathbf{S}\mathbf{H}^T\mathbf{V}\mathbf{\Sigma}^{-1}$ and $\bar{\mathbf{U}} = \varnothing$; otherwise, $\mathbf{U} = \mathbf{J}\mathbf{S}\mathbf{H}^T\mathbf{V}(:, 1:c)\mathbf{\Sigma}(1:c, 1:c)^{-1}$, $\bar{\mathbf{U}}^T\bar{\mathbf{U}} = n\mathbf{I}_{r-c}$ and $\left[\mathbf{U}, \mathbf{1}_n\right]^T\bar{\mathbf{U}} = 0$.

Proposition 1 is derived from the Lemma 2 in [7]. If $r \le c$, Eq. (7) has a unique global solution; otherwise, Eq. (7) has numerous optimal solutions.

Fix C and update H: With $\mathbf{C}$ fixed, $\mathbf{H} = sgn(\mathbf{C})$.

In summary, we present the detailed optimization procedure of Eq. (5) in Algorithm 1. Because this algorithm preserves the hashing function $\mathbf{H}$ in the objective function, we name it as **KSDH_H**.

Algorithm 1: KSDH_H

Input: Kernelized data matrix $\bar{\mathbf{K}} \in \mathbb{R}^{m\times n}$, weight matrix $\mathbf{S}$, number of bits r, and $\epsilon = 0.01$.

Output: Projection matrix $\mathbf{A} \in \mathbb{R}^{r\times m}$.

Initialize: t=0, let $\mathbf{A}_0$ be the eigenvectors of $\bar{\mathbf{K}}\mathbf{S}\bar{\mathbf{K}}^T$ and $\mathbf{H}_0 = sgn(\mathbf{A}\bar{\mathbf{K}})$;
Calculate $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$, $\mathbf{P} = \mathbf{JS}$, and $\mathbf{M} = \bar{\mathbf{K}}^T(\bar{\mathbf{K}}\bar{\mathbf{K}}^T + \epsilon\mathbf{I}_m)^{-1}$;
while not converge or reach the maximum iterations
  Calculate $\mathbf{U}_{t+1}$, $\bar{\mathbf{U}}_{t+1}$ and $\mathbf{V}_{t+1}$ by Proposition 1;
  Update $\mathbf{C}_{t+1} = \sqrt{n}\mathbf{V}_{t+1}\left[\mathbf{U}_{t+1}, \bar{\mathbf{U}}_{t+1}\right]^T$;
  Update $\mathbf{H}_{t+1} = sgn(\mathbf{C}_{t+1})$;
end while
Calculate $\mathbf{A} = \mathbf{CM}$.

3.2 KSDH with A Relaxed Binary Code Matrix

KSDH_H is very effective for binary encoding, but sometimes the objective value might fluctuate during the optimization procedure and thus would affect the binary code generation. For example, Fig. 2(a) and (b) show the objective value and the retrieval accuracy, mean average precision (MAP), with respect to the number of optimization iterations, respectively. In order to address this problem, we further improve the model Eq. (4) as:

$$\begin{aligned} \underset{\mathbf{B}\in\{-1,1\},\mathbf{A}}{min} & \left\|\mathbf{B}^T\mathbf{A}\bar{\mathbf{K}} - r\mathbf{S}\right\|_F^2 + \lambda\left\|\mathbf{B} - \mathbf{A}\bar{\mathbf{K}}\right\|_F^2, \\ & s.t.\ \mathbf{A}\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}^T = n\mathbf{I}_r, \end{aligned} \tag{8}$$

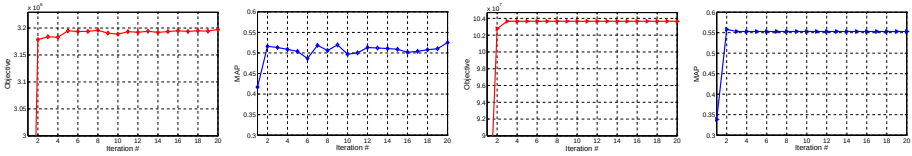
where $\mathbf{B} \in \mathbb{R}^{r\times n}$ represents binary codes of training data, the term $\left\|\mathbf{B} - \mathbf{A}\bar{\mathbf{K}}\right\|_F^2$ aims to reduce the accumulated quantization error between binary code matrix $\mathbf{B}$ and linear functions $\mathbf{A}\bar{\mathbf{K}}$, and the parameter λ is to balance the semantic information and the accumulated quantization error. The major differences between Eq. (4) and Eq. (8) are: (i) The binary code matrix $\mathbf{B}$ in Eq. (8) is not required to be equivalent to $sgn(\mathbf{A}\bar{\mathbf{K}})$, and $\mathbf{B} = sgn(\mathbf{A}\bar{\mathbf{K}})$ can be viewed as one particular case of Eq. (8); (ii) The regularization term can guarantee Eq. (8) to have a stable optimal solution (see Proposition 2 and 3). Similar to Eq. (4), the optimization problem in Eq. (8) is equivalent to the following optimization problem:

$$\begin{aligned} \underset{\mathbf{B}\in\{-1,1\},\mathbf{A}}{max} & Tr\left\{\mathbf{B}(r\mathbf{S} + \lambda\mathbf{I}_n)\bar{\mathbf{K}}^T\mathbf{A}^T\right\}, \\ & s.t.\ \mathbf{A}\bar{\mathbf{K}}\bar{\mathbf{K}}^T\mathbf{A}^T = n\mathbf{I}_r. \end{aligned} \tag{9}$$

Let $\mathbf{C} = \mathbf{A}\bar{\mathbf{K}}$, Eq. (9) becomes:

$$\begin{aligned} \underset{\mathbf{B}\in\{-1,1\},\mathbf{C}}{max} & Tr\left\{\mathbf{B}(r\mathbf{S} + \lambda\mathbf{I}_n)\mathbf{C}^T\right\}, \\ & s.t.\ \mathbf{C}\mathbf{C}^T = n\mathbf{I}_r, \mathbf{C}\mathbf{1}_n = 0. \end{aligned} \tag{10}$$

We can obtain a stable projection matrix $\mathbf{A}$ by $\mathbf{A} = \mathbf{C}\bar{\mathbf{K}}^T(\bar{\mathbf{K}}\bar{\mathbf{K}}^T + \epsilon\mathbf{I}_m)^{-1}$ after obtaining $\mathbf{C}$. Similar to Algorithm 1, we also adopt an alternative strategy to solve the optimization problem in Eq. (10), and the two alternate steps are



(a) (b) (c) (d)

Fig. 2. Retrieval accuracy (32-bit) of KSDH_H and KSHD_B with respect to the number of iterations. (a) The objective value of Eq. (5) vs. Iteration #. (b) MAP of KSDH_H vs. Iteration #. (c) The objective value of Eq. (9) vs. Iteration #. (d) MAP of KSDH_B vs. Iteration #. (In total 1K training data are selected from CIFAR-10 database, and 100 training data are chosen as anchors to construct kernels)

presented in details below.

Fix B and update C: The matrix $\mathbf{C}$ in Eq. (10) can be obtained by Proposition 2, which can be viewed as a particular case of Proposition 1.

Proposition 2: With $\mathbf{B}$ fixed, the unique global optimal solution of Eq. (10) is $\mathbf{C} = \sqrt{n}\mathbf{V}\mathbf{U}^T$, where $\mathbf{V} \in \mathbb{R}^{r\times r}$ is obtained based on the SVD of $\mathbf{B}(r\mathbf{S} + \lambda\mathbf{I}_n)\mathbf{J}(r\mathbf{S} + \lambda\mathbf{I}_n)\mathbf{B}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$, and $\mathbf{U} = \mathbf{J}(r\mathbf{S} + \lambda\mathbf{I}_n)\mathbf{B}^T\mathbf{V}\mathbf{\Sigma}^{-1}$.

Fix C and update B: The optimization problem in Eq. (10) becomes:

$$\underset{\mathbf{B}\in\{-1,1\}}{max}\ Tr\left\{\mathbf{B}(r\mathbf{S} + \lambda\mathbf{I}_n)\mathbf{C}^T\right\}, \tag{11}$$

whose optimal solution is $\mathbf{B} = sgn(\mathbf{C}(r\mathbf{S} + \lambda\mathbf{I}_n))$.

The detailed optimization procedure of solving the model Eq. (8) is shown in Algorithm 2. Since Eq. (8) maintains the discrete constraint using binary code matrix $\mathbf{B}$, we name it as **KSDH_B**. In Algorithm 2, since each iterative optimization step maximizes the objective function in Eq. (9) and the objective value in each iteration is always non-decreasing and bounded, Algorithm 2 will converge to an optimal solution. Therefore, we have the following proposition:

Proposition 3: The loop step in Algorithm 2 will monotonously increase in each iteration, and thus Algorithm 2 will converge to an optima.

Fig. 2(c) and (d) show the iteration process of the loop steps and the objective value of Eq. (9) does not exhibit significant variations after three iterations, which implies that the loop step converges rapidly. This is attributed to the fact that each iteration has a closed-form solution. Usually we set the maximum iteration number l to be three.

3.3 Time Complexity Analysis

Before analyzing the time complexity of two proposed algorithms, we first investigate the time cost of calculating the kernel data matrix $\bar{\mathbf{K}}$. Given a data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, the time complexity of constructing kernel matrix $\mathbf{K}$ is $\mathcal{O}(mdn)$ with m selected points, and normalizing $\mathbf{K}$ to have zero-mean needs at most $\mathcal{O}(mn)$

Algorithm 2: KSDH_B

Input: Kernelized data matrix $\bar{\mathbf{K}} \in \mathbb{R}^{m \times n}$, weight matrix $\mathbf{S}$, number of bits r, the parameter λ and $\epsilon = 0.01$.

Output: Binary code matrix $\mathbf{B} \in \mathbb{R}^{r \times n}$, projection matrix $\mathbf{A} \in \mathbb{R}^{r \times m}$.

Initialize: t=0, let $\mathbf{A}_0$ be the eigenvectors of $\bar{\mathbf{K}}\mathbf{S}\bar{\mathbf{K}}^T$ and $\mathbf{B}_0 = sgn(\mathbf{A}\bar{\mathbf{K}})$;
Calculate $\mathbf{J} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$, $\hat{\mathbf{S}} = r\mathbf{S} + \lambda\mathbf{I}_n$, $\mathbf{P} = \mathbf{J}\hat{\mathbf{S}}$, and $\mathbf{M} = \bar{\mathbf{K}}^T(\bar{\mathbf{K}}\bar{\mathbf{K}}^T + \epsilon\mathbf{I}_m)^{-1}$;
Repeat
  Update $\mathbf{V}_{t+1}$ and $\mathbf{\Sigma}_{t+1}$ with the SVD of $\mathbf{B}_t\hat{\mathbf{S}}\mathbf{P}\mathbf{B}_t^T$;
  Update $\mathbf{U}_{t+1} = \mathbf{P}\mathbf{B}_t^T\mathbf{V}_{t+1}\mathbf{\Sigma}_{t+1}^{-1}$;
  Update $\mathbf{C}_{t+1} = \sqrt{n}\mathbf{V}_{t+1}\mathbf{U}_{t+1}^T$;
  Update $\mathbf{B}_{t+1} = sgn(\mathbf{C}_{t+1}\hat{\mathbf{S}})$;
Until convergence
Calculate $\mathbf{A} = \mathbf{C}\mathbf{M}$.

operations. Therefore, the time complexity of calculating the kernel data matrix $\bar{\mathbf{K}}$ is $\mathcal{O}(mdn)$. Usually, $m << n$ and $d < n$.

KSDH_H contains initialization, the loop step, and projection matrix computation. In the initialization step, the time complexity of calculating $\mathbf{A}_0$ and $\mathbf{H}_0$ is $\mathcal{O}(mn^2)$ and $\mathcal{O}(rmn)$, respectively. Calculating the matrices $\mathbf{J}$, $\mathbf{P}$ and $\mathbf{M}$ requires at most $\mathcal{O}(n^2)$, $\mathcal{O}(n^2)$, $\mathcal{O}(m^2n)$ operations, respectively. Hence, the initialization step takes $\mathcal{O}(mn^2)$. In the loop step, computing matrices $\mathbf{U}_{t+1}$ and $\mathbf{V}_{t+1}$ needs $\mathcal{O}(rn^2)$ operations. Updating matrices $\mathbf{C}_{t+1}$ and $\mathbf{H}_{t+1}$ requires $\mathcal{O}(rmn)$ and $\mathcal{O}(rn)$, respectively. Thus, the time complexity of the loop step is $\mathcal{O}(lrn^2)$, where l is the number of iterations, and it is empirically set to be three. Calculating the projections matrix $\mathbf{A}$ takes at most $\mathcal{O}(rmn)$ operations. Therefore, the total time complexity of KSDH_H is $max(\mathcal{O}(mn^2), \mathcal{O}(lrn^2))$.

Similarly, KSDH_B also consists of initialization, the loop step, and projection matrix calculation. Initialization and calculating the projection matrix $\mathbf{A}$ takes $\mathcal{O}(mn^2)$ and $\mathcal{O}(rmn)$, respectively. In the loop step, calculating the matrix $\mathbf{B}_t\hat{\mathbf{S}}\mathbf{P}\mathbf{B}_t^T$ costs $\mathcal{O}(rn^2)$, and its SVD spends at most $\mathcal{O}(r^3)$ operations. Updating matrices $\mathbf{U}_{t+1}$, $\mathbf{C}_{t+1}$ and $\mathbf{B}_{t+1}$ requires $\mathcal{O}(rn^2)$, $\mathcal{O}(rmn)$, and $\mathcal{O}(rn^2)$ operations, respectively. Thus, the time complexity of the loop step is $\mathcal{O}(lrn^2)$. The total time complexity of KSDH_B is also $max(\mathcal{O}(mn^2), \mathcal{O}(lrn^2))$.

In summary, the time complexity of the training stage of both KSDH_H and KSDH_B is $max(\mathcal{O}(mn^2), \mathcal{O}(lrn^2))$ determined by the weight matrix $\mathbf{S}$ with size $n \times n$. Note that both algorithms can be easily paralleled to handle large-scale datasets, since the weight matrix $\mathbf{S}$ is simply used for matrix multiplication. In the test stage, the time complexity of encoding one test sample into r-bit binary codes is $\mathcal{O}(md + mr)$.

4 Experiments and Analysis

We evaluate the proposed KSDH_H and KSDH_B on four publicly available benchmark databases: CIFAR-10, MNIST, Youtube, and ImageNet. CIFAR-10 database is a labeled subset of 80M tiny images [22], containing 60K color images

of ten object categories. Each of which is constituted of 6K images. Every image is aligned and cropped to 32×32 pixels and then represented by a 512-dimensional GIST feature vector [23]. MNIST database [24] consists of 70K images each of which is represented by a 784-dimensional vector, with handwritten digits from '0' to '9' contained. Youtube face database [25] contains 1,595 individuals, from which we choose 400 people to form a set with 136,118 face images and then randomly select 50 individuals that each one has at least 300 images to form a subset. We use the LBP feature vector [26] with 1,770-dimensional to represent each face image. ImageNet database [27] contains over 14 million labeled data, and we adopt the ILSVRC 2012 subset, which has more than 1.2 million images of totally 1000 object categories. We use GIST to extract a 2048-dimensional feature vector for each image.

We compare KSDH_H and KSDH_B against four start-of-the-art supervised hashing methods including semi-supervised hashing (SSH) [9], binary reconstructive embedding (BRE) [14], kernel supervised hashing (KSH) [17], and supervised discrete hashing (SDH) [18]. In addition, we also show the results of the baseline method nearest neighbors (NN). In KSDH_H and KSDH_B, we choose the same kernel as KSH for fair comparison and set the regularization parameter $\lambda = r$ in experiments. For SSH, we kernelize the labeled data using the same kernel as KSH and apply the non-orthogonal method to its relaxed objective function; for BRE, we assign label 1 to similar pairs and 0 to dissimilar pairs. Since all these methods refer to kernels, we choose ten percent of training data as anchors to construct the kernels.

Two standard main criterions: mean average precision (MAP) and precision-recall (PR) curve, are used to evaluate the above hashing methods. Note that since KSDH_H, KSDH_B, SSH, KSH and SDH use the same type of kernels, they have almost the same test time. However, different methods have significantly different training speeds. In the experimental part, we will provide the training time of each hashing method for comparison. All experiments are conducted using Matlab on a 3.60GHz Intel Core i7-4790 CPU with 32GB memory.

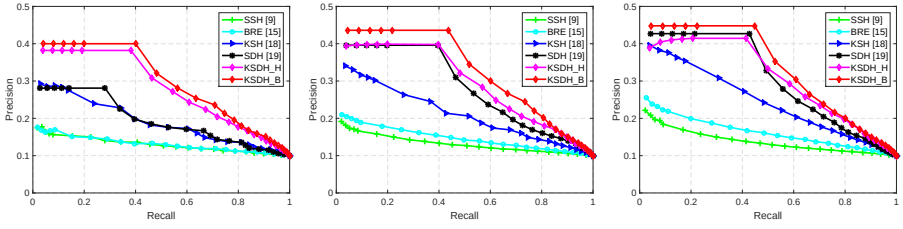
4.1 CIFAR-10

We partition this database into two parts: a training subset of 59K images and a test query set of 1K images, which contains ten categories with each consisting of 100 images. We uniformly select 100 and 500 images from each category to form two training sets, respectively. The weight matrix $\mathbf{S}$ is constituted by the true semantic neighbors. We encode each image into 8-,16- and 32-bit binary codes by SSH, BRE, KSH, SDH and our proposed KSDH_H and KSDH_B. The ranking performance is shown in term of MAP together with their training time (seconds) in Table 1. In addition, we also provide the PR curves with 8-, 16- and 32-bit codes in Fig. 3.

As shown in Table 1 and Fig. 3, KSDH_B achieves the highest retrieval accuracy (MAP and PR curve) among all hashing methods, and KSDH_H is the second best at 8- and 16-bit. More importantly, both KSDH_H and KSDH_B significantly outperform the other hashing methods at 8-bit, which is smaller

Table 1. Ranking performance and training time (seconds) on the CIFAR-10 database.

Method	n=1000				n=5000			
	MAP			Time	MAP			Time
	8-bit	16-bit	32-bit	32-bit	8-bit	16-bit	32-bit	32-bit
NN	0.1755			-	0.1713			-
SSH [9]	0.1544	0.1517	0.1684	0.02	0.1540	0.1529	0.1598	0.66
BRE [14]	0.1587	0.1715	0.1852	156.36	0.1589	0.1723	0.1905	1989.20
KSH [17]	0.2474	0.2782	0.3135	21.75	0.2787	0.3317	0.3746	692.78
SDH [18]	0.3580	0.4937	0.5314	0.05	0.5224	0.5889	0.6218	0.70
KSDH_H	0.5102	0.5109	0.4633	0.50	0.5661	0.5974	0.6131	3.10
KSDH_B	**0.5185**	**0.5481**	**0.5495**	0.05	**0.5995**	**0.6208**	**0.6317**	1.65



(a) @ 8-bit (b) @ 16-bit (c) @ 32-bit

Fig. 3. PR curves of different hashing methods using 8, 16, 32-bit codes on the CIFAR-10 database with 1000 labeled training data.

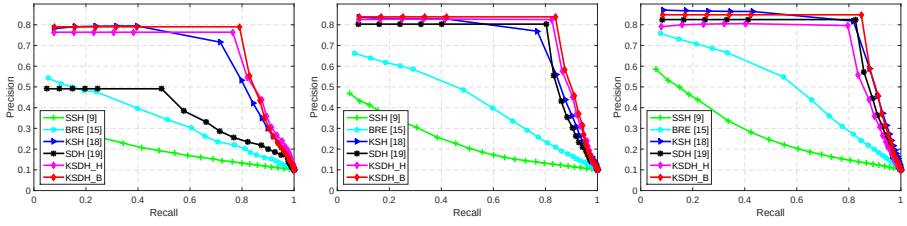
than the number of categories, and the gain in MAP ranges from 8.3% to 44.8% over the best competitor SDH. It is clear that KSDH_H and KSDH_B are very effective with short binary codes, which are often favorable due to their low requirement of storage. Compared to KSH and BRE, KSDH_H and KSDH_B are much faster to learn a training model, and in contrast with SSH and SDH, the training time cost is also acceptably low. In addition, KSDH_H requires more training cost than KSDH_B due to the construction of $\bar{\mathbf{U}}$ (see Proposition 1). As we show in Fig. 3, we want to emphasize that our proposed KSDH_H and KSDH_B are significantly better than other state of the arts especially when the number of bits is low in the hashing code (Fig. 3).

4.2 MNIST

Similar to CIFAR-10, we also partition the MNIST handwritten digit database into two subsets. Specifically, we select 6.9K images from each digit to constitute a training set with the remaining 1K images as a test query set, and then we uniformly select 100 and 500 images from each digit for training. Table 2 and Fig. 4 present the retrieval accuracy in term of MAP and PR curves at 8-, 16- and 32-bit, respectively. As we can see, KSDH_B outperforms the other hashing methods in most of cases, especially at 8-bit, at which its MAP is 10.82% and 3.08% higher than the best competitor except KSDH_H on two different training sets,

Table 2. Ranking performance and training time (seconds) on the MNIST database.

Method	n=1000				n=5000			
	MAP			Time	MAP			Time
	8-bit	16-bit	32-bit	32-bit	8-bit	16-bit	32-bit	32-bit
NN	0.4466			-	0.4394			-
SSH [9]	0.2301	0.3117	0.3642	0.02	0.2432	0.3406	0.3338	0.67
BRE [14]	0.4306	0.5352	0.6247	138.20	0.4389	0.5515	0.6662	2283.90
KSH [17]	0.7381	0.8160	0.8478	20.48	0.8169	0.8803	0.9185	717.46
SDH [18]	0.6039	0.8485	0.8680	0.06	0.9031	0.9410	0.9427	0.90
KSDH_H	**0.8591**	0.8621	0.8626	0.50	0.9330	0.9320	0.9379	3.43
KSDH_B	0.8463	**0.8757**	**0.8792**	0.04	**0.9339**	**0.9415**	**0.9482**	1.69



(a) @ 8-bit (b) @ 16-bit (c) @ 32-bit

Fig. 4. PR curves of different hashing methods using 8, 16, 32-bit codes on the MNIST database with 1000 labeled training data.

respectively. In addition, KSDH_H and KSDH_B exhibit similar high retrieval accuracy (MAP) at all three types of bit numbers. This implies that they can produce very effective and compact binary codes.

4.3 Youtube

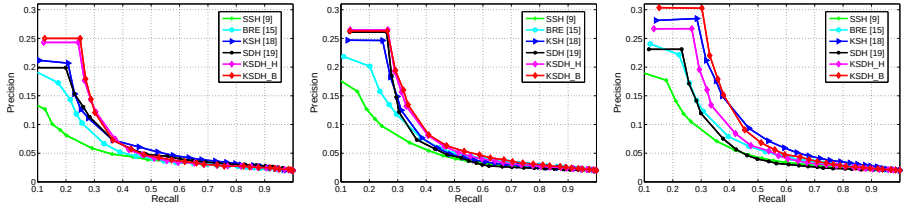
We split the selected set constituted by 50 people into a training set and a test query set, and then uniformly choose 20 and 100 images from each individual in the training set for training, and 20 images from each individual in the query set for test. Evaluation results in term of MAP and PR curves are shown in Table 3 and Fig. 5. Table 3 shows that KSDH_H achieves the best retrieval accuracy (MAP) at 8- and 16-bit, respectively, and when $n = 1000$, the MAP is 7.02% and 3.52% higher than the best competitor except KSDH_B, respectively. In addition, KSDH_B consistently has superior retrieval accuracy to the other hashing methods except KSDH_H. It also costs the least training time among all comparative methods as well.

4.4 ImageNet

We randomly pick around 65K images with 10 categories from the ILSVRC 2012 database, and then partition this set into two subsets: a training set about

Table 3. Ranking performance and training time (seconds) on the Youtube database.

Method	n=1000				n=5000			
	MAP			Time	MAP			Time
	16-bit	32-bit	64-bit	64-bit	16-bit	32-bit	64-bit	64-bit
NN	0.2965			-	0.3067			-
SSH [9]	0.1631	0.1678	0.2241	0.03	0.1697	0.2134	0.2293	0.75
BRE [14]	0.2198	0.2347	0.2361	3113.38	0.2428	0.2510	0.2909	26888.55
KSH [17]	0.2520	0.3009	0.3106	34.90	0.3390	0.3946	0.4246	1356.09
SDH [18]	0.2449	0.2314	0.2764	0.64	0.3572	0.3693	0.3942	3.83
KSDH_H	**0.3222**	**0.3351**	0.3060	2.14	0.3580	0.4014	0.4402	10.51
KSDH_B	0.2727	0.3140	**0.3492**	0.08	**0.3790**	**0.4420**	**0.4475**	1.93



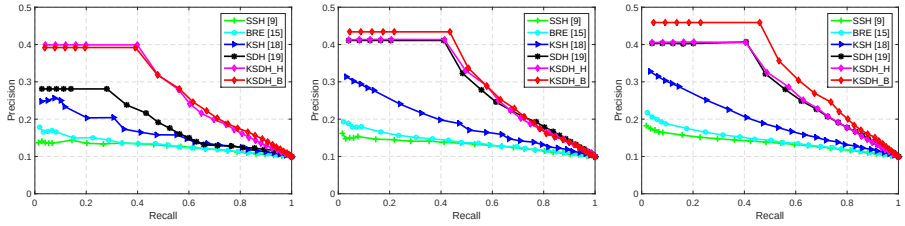
(a) @ 16-bit (b) @ 32-bit (c) @ 64-bit

Fig. 5. PR curves of different hashing methods using 16, 32, 64-bit codes on the Youtube database with 1000 labeled training data.

Table 4. Ranking performance and training time (seconds) on the ImageNet database.

Method	n=5000				n=10000			
	MAP			Time	MAP			Time
	8-bit	16-bit	32-bit	32-bit	8-bit	16-bit	32-bit	32-bit
NN	0.1427			-	0.1356			-
SSH [9]	0.1593	0.1485	0.1507	0.61	0.1556	0.1472	0.1514	4.09
BRE [14]	0.1522	0.1515	0.1620	3153.42	0.1492	0.1506	0.1577	15453.53
KSH [17]	0.2268	0.2321	0.2454	1292.50	0.2147	0.2466	0.2616	4663.91
SDH [18]	0.3882	0.5007	0.5063	1.03	0.4860	0.5316	0.5610	2.98
KSDH_H	**0.5042**	0.5170	0.5199	4.92	**0.5429**	0.5461	0.5501	12.54
KSDH_B	0.4987	**0.5368**	**0.5509**	1.63	0.5255	**0.5734**	**0.5714**	7.06

64K images and a test query set of 1K images evenly sampled from these ten categories. Next, we uniformly select 500 and 1000 images of each category from the training set for training. Evaluation results in term of MAP and PR curve are presented in Table 4 and Fig. 6, respectively, which show that KSDH_H and KSDH_B outperform the other hashing methods, and the gain in MAP ranges from 1.9% to 29.9% over the best competitor. Meanwhile, KSDH_H has almost the same retrieval accuracy (MAP) at all 8, 16 and 32 bits.



(a) @ 8-bit (b) @ 16-bit (c) @ 32-bit

Fig. 6. PR curves of different hashing methods using 8, 16, 32-bit codes on the ImageNet database with 5000 labeled training data.

4.5 Discussion

Based on the experiments on the four benchmark databases, we can observe that KSDH_H usually achieves the best or the second to the best retrieval accuracy with short r-bit ($r \leq c$) binary codes, while generally KSDH_B exhibits more stable and better retrieval accuracy than KSDH_H. Moreover, KSDH_B has superior retrieval accuracy in term of MAP and PR curve to SSH, BRE, KSH and SDH in most cases. The main possible reasons are summarized as follows:

- KSDH_B performs better than KSDH_H in most cases, probably because KSDH_B relaxes the hashing function constraints into a binary code matrix and adds a regularization term, which can help us to obtain optimal and stable solutions of Eq. (9) at different number of bits.
- Unlike SSH that directly relaxes its objective function and KSH that adopts the relaxation as well as the greedy algorithm to solve its non-differential objective function, KSDH_H and KSDH_B preserve the discrete constraint in their optimization models and reduce the accumulated quantization error between the binary code matrix and the linear projection functions. Meanwhile, the effective optimization algorithms that we used can effectively capture and preserve the semantic information in a low-dimensional Hamming space.
- Compared with the discrete hashing methods such as BRE and SDH, KSDH_H and KSDH_B can obtain significantly better retrieval accuracy (MAP and PR curve) with shorter binary codes ($r \leq c$), because their objective functions can better capture the low-rank semantic information determined by the weight matrix $\mathbf{S}$. Usually, SDH can achieve the best retrieval accuracy themselves when $r > c$, because it aims to reduce the dimension of the discrete matrix to c.

5 Conclusions

In this paper, we propose a novel yet simple kernel-based supervised discrete hashing algorithm, including two effective and efficient models: KSDH_H and

KSDH_B, via the asymmetric relaxation strategy. To reduce the accumulated quantization error between hashing and linear projection functions, KSDH_H adopts the element-wise product between the hashing and linear projection functions to approximate the weight matrix; KSDH_B relaxes the hashing function into a general binary code matrix and introduces a regularization term, which can guarantee optimal and stable solutions to the objective function. In addition, we adopt an alterative strategy to efficiently solve the corresponding optimization models, and the semantic information are captured and preserved into a low-dimensional Hamming space. Experiments on four benchmark databases demonstrate that the proposed hashing framework has superior retrieval performance to the state of the arts, and can achieve very good retrieval accuracy with short binary codes. Since the time complexity and the memory consumption are determined by the weight matrix, in the future we plan to investigate the weight matrix to further reduce the time complexity without sacrificing the retrieval accuracy.

References

1. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features (2010) ECCV.
2. Shakhnarovich, G., Viola, P., Darrell, T.: Fast pose estimation with parameter-sensitive hashing (2003) ICCV.
3. Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint (2012) CVPR.
4. Zhang, S., Yang, M., Cour, T., Yu, K., Metaxas, D.N.: Query specific rank fusion for image retrieval. TPAMI **37**(4) (2015) 803–815
5. Li, X., Lin, G., Shen, C., Hengel, A.V.D., Dick, A.: Learning hash functions using column generation (2013) arXiv preprint arXiv:1303.0339.
6. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing (2009) NIPS.
7. Liu, W., Cun, M., Kumar, S., Chang, S.F.: Discrete graph hashing (2014) NIPS.
8. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large databases for recognition (2008) CVPR.
9. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for large scale search. TPAMI **34**(12) (2012) 2393–2406
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality (1998) ACM STOC.
11. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs (2011) ICML.
12. Gong, Y., , Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. TPAMI **35**(12) (2013) 2916–2929
13. Smeulders, A.W., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. TPAMI **22**(12) (2000) 1349–1380
14. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings (2009) NIPS.
15. Strecha, C., Bronstein, A.M., Bronstein, M.M., Fua, P.: Ldahash: Improved matching with smaller descriptors. TPAMI **34**(1) (2012) 66–78
16. Norouzi, M., Blei, D.M.: Minimal loss hashing for compact binary codes (2011) ICML.

17. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels (2012) CVPR.
18. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing (2015) CVPR.
19. Shen, F., Liu, W., Zhang, S., Yang, Y., Shen, H.: Learning binary codes for maximum inner product search (2015) ICCV.
20. Shi, X., Guo, Z., Nie, F., Yang, L., You, J., Tao, D.: Two-dimensional whitening reconstruction for enhancing robustness of principal component analysis. TPAMI (2015)
21. Lin, G., Shen, C., Suter, D., Hengel, A.: A general two-step approach to learning-based hashing (2013) ICCV.
22. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. TPAMI **30**(11) (2008) 1958–1970
23. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV **42**(3) (2001) 145–175
24. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
25. Wolf, L., Hassner, T., Maoz, I.: Face recognition in unconstrained videos with matched background similarity (2011) CVPR.
26. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. TPAMI **28**(12) (2006) 2037–2041
27. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: Imagenet: A large-scale hierarchical image database (2009) CVPR.