Supervised Hashing via Uncorrelated Component Analysis

SungRyull Sohn
CG Research Team
Electronics and Telecommunications Research Institute
School of Electrical Engineering
Korea Advanced Institute of Science and Technology
sungluol@etri.re.kr

Hyunwoo Kim
Kakao Corp.
eugene.kim@kakaocorp.com

Junmo Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology
junmo@ee.kaist.ac.kr

Abstract

The Approximate Nearest Neighbor (ANN) search problem is important in applications such as information retrieval. Several hashing-based search methods that provide effective solutions to the ANN search problem have been proposed. However, most of these focus on similarity preservation and coding error minimization, and pay little attention to optimizing the precision-recall curve or receiver operating characteristic curve. In this paper, we propose a novel projection-based hashing method that attempts to maximize the precision and recall. We first introduce an uncorrelated component analysis (UCA) by examining the precision and recall, and then propose a UCA-based hashing method. The proposed method is evaluated with a variety of datasets. The results show that UCA-based hashing outperforms state-of-the-art methods, and has computationally efficient training and encoding processes.

Introduction

The nearest neighbor search problem is a central challenge in large-scale data processing and analysis. Its applications include information retrieval, data mining, and classification problems. The inherent challenges of the nearest neighbor search are now receiving more attention because of the growing availability of large databases. The inevitable difficulty of finding an exact nearest neighbor in large-scale datasets has led to the Approximate Nearest Neighbor (ANN) search problem, which aims to find the nearest neighbor point approximately but rapidly. Recent studies have shown that hashing is a promising approach to solving this problem. This is because the resulting binary hash code is highly compressed, allowing the main memory to be used to compute the distance, and because the distance between two binary hash codes can be calculated using the XOR operation, which is computationally efficient. The hashing method uses a hash function to encode the hash code, and the nearest neighbor is determined by thresholding the Hamming distance between hash codes. Hash functions can be designed in various ways, but a trade-off is generally required between the encoding complexity (T_{encode}) and performance in terms of quantization (or coding) error. There are two main types of hash functions: projection-based and codebook-based.

Projection-based hashing functions: Applications of the ANN search problem require a constant encoding complexity (T_{encode}) for scalability. One promising way to guarantee T_{encode}=O(k) complexity with a certain level of performance is known as projection-based hashing. This is characterized by the transform matrix $\mathbf{W}$ and quantization function $Q(\cdot)$ that produce a k-bit hash code. For an input vector (hash key) $\mathbf{x}$, the resulting output vector (hash code) $\mathbf{z}$ can be written as $\mathbf{z} = Q(\mathbf{W}^T\mathbf{x})$. Various researchers have developed projection-based methods, e.g., spectral hashing (SH) (Weiss, Torralba, and Fergus 2008), locality sensitive hashing (LSH) (Indyk and Motwani 1998), iterative-quantization (ITQ) (Gong and Lazebnik 2011), LDA-hash (Strecha et al. 2012), minimal loss hashing (MLH) (Norouzi and Fleet 2011), and locality preserving hashing (LPH) (Zhao, Lu, and Mei 2014).

Codebook-based hashing functions: Other researchers have developed codebook-based hashing functions, which assign a data point to one of the $N_{centroid}$ nearest codebook centroids in terms of the Euclidean distance. However, the complexity of this method becomes intractable when directly applied to a large-scale dataset (i.e., T_{encode} = O[$N_{centroid}$] = O[2^k]). To overcome this problem, product quantization (PQ)-based methods (Ge et al. 2013; He, Wen, and Sun 2013; Jegou, Douze, and Schmid 2011; Heo, Lin, and Yoon 2014) decompose the space of the data into low-dimensional subspaces, which are quantized separately.

More recently, a number of non-linear hash functions that exhibit much improved performance have been proposed. These include kernelized locality sensitive hashing (KLSH), supervised hashing with kernel (KSH), self-taught hashing (STH), two-step hashing (TSH), fast hashing (FastH), and supervised discrete hashing (SDH) (Kulis and Grauman 2012; Liu et al. 2012; Zhang et al. 2010; Lin et al. 2013;

2014; Shen et al. 2015).

Although many methods have enhanced the performance of the ANN search using hashing methods, the retrieval performance has not been directly tackled. In this paper, based on a theoretical analysis of the relation between hash functions and the precision-recall performance, we present a novel projection-based hash function. First, we formulate a family of candidate projection-based hash functions that produce efficient code (**Step 1**). We obtain these hash functions by an Uncorrelated Component Analysis (UCA) transformation. From the resulting hash functions, we then determine which one maximizes the precision and recall performance (**Step 2**). Specifically, based on an analysis of retrieval performance, we set a novel objective function to optimize the precision and recall. We then consider a special case of normally distributed data, and propose a refined subset of UCA-based transforms that maximize the sum of the true positive rates, which is equivalent to minimizing the expected Hamming distance between hash codes for similar data pairs.

Formulation

Problem Formulation

Let $p(\mathbf{x})$ denote the probability density function of the data $\mathbf{x} \in \mathbb{R}^d$. Without loss of generality, we assume the mean of $\mathbf{x}$ to be the zero vector. Then, we are given a set of N *i.i.d.* training data $\{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^d$ drawn from $p(\mathbf{x})$. The pairwise similarity of two data points $(\mathbf{x}_i, \mathbf{x}_j)$ denoted by $s_{i,j}$ is 1 when $(\mathbf{x}_i, \mathbf{x}_j)$ are similar, and $s_{i,j} = 0$ when dissimilar. We define $\mathcal{M} = \{(\mathbf{x}_i, \mathbf{x}_j) | s_{i,j} = 1\}$ and $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{x}_j) | s_{i,j} = 0\}$, and seek a hash function that preserves this similarity information. Let $\mathbf{z} = \Phi(\mathbf{x})$ be the k-bit hash code, which is the output of the hash function $\Phi(\cdot)$ for data $\mathbf{x}$. Then, the canonical projection-based hash functions, characterized by the transformation matrix $\mathbf{W} \in \mathbb{R}^{d\times k}$ and bias $\mathbf{b} \in \mathbb{R}^k$, can be written as:

$$\Phi(\mathbf{x}) = u(\mathbf{W}^T\mathbf{x} + \mathbf{b}) = u(\mathbf{y} + \mathbf{b}), \tag{1}$$

where the quantization function $Q(\cdot)$ is replaced by the unit step function $u(\cdot)$ with a bias $\mathbf{b}$, and $\mathbf{y} = \mathbf{W}^T\mathbf{x} \in \mathbb{R}^k$ is the transformed data point for $\mathbf{x}$. The n^{th} bit of the vector $u(\mathbf{y} + \mathbf{b})$ is 1 *iff* the n^{th} element of $\mathbf{y} + \mathbf{b}$ is positive, and is 0 otherwise. The performance of the hash function is evaluated in terms of its precision and recall values by retrieving data for given queries from the database.

Similar to (Gong and Lazebnik 2011; Weiss, Torralba, and Fergus 2008; Wang, Kumar, and Chang 2012), we first require the hash code to be efficient. Among all the hash functions that generate hash codes with this property, we seek the one that maximizes the precision and recall performance. A hash code is considered to be efficient if it satisfies the following conditions:

- **Condition 1**: Each bit of the hash code has a 50% chance of being 1 or 0.
- **Condition 2**: Different bits of the hash code are independent of one another.

Note that **Condition 1** is satisfied for any $\mathbf{W}$ if $\mathbf{b}$ is set as the negative of the median vector of $\mathbf{y}$. Thus, we have $\{W | \exists b, s.t.$ **Condition 1**, **Condition 2** satisfied$\}$ $=$ $\{W|$**Condition 2** satisfied$\} \triangleq \mathcal{W}$. Then, we can drop **Condition 1**, and our problem can be formulated as

$$\max_{\mathbf{W}\in\mathcal{W}} L(\mathbf{W}), \tag{2}$$

where $L(\mathbf{W})$ is the objective function that measures the precision and recall performance. In the following sections, we first formulate the precision and recall, and determine the form of $L(\mathbf{W})$. We then tackle problem (2) in two steps.

Formulation of Precision and Recall

In a traditional information retrieval scenario with a binary hash code, a data point $\mathbf{x}_i$ from a database is matched with an input data query $\mathbf{q}$ and retrieved when the Hamming distance between $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{q})$ is not greater than some threshold γ (i.e., $d_H(\Phi(\mathbf{q}), \Phi(\mathbf{x}_i)) \leq \gamma$). In the training stage, we consider all of the points in the database as potential queries, and the precision and recall for the training data pair $(\mathbf{x}_i, \mathbf{x}_j)$ and threshold γ can be written as

$$\text{precision}(\mathbf{W}, \gamma) = \frac{\text{TPR}(\mathbf{W}, \gamma)}{\text{RR}(\mathbf{W}, \gamma)} \tag{3}$$

$$= \frac{P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma, (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}]}{P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma]} \tag{4}$$

$$\text{recall}(\mathbf{W}, \gamma) = \frac{\text{TPR}(\mathbf{W}, \gamma)}{Pr(\text{similar})} \tag{5}$$

$$= \frac{P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma, (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}]}{P[(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}]}, \tag{6}$$

where $\mathbf{z}_i = \Phi(\mathbf{x}_i), \mathbf{z}_j = \Phi(\mathbf{x}_j)$, and the true positive rate (TPR) and retrieval rate (RR) change with γ, whereas $Pr(\text{similar})$ remains constant. Considering that the denominator of Eq. 6 does not change with respect to γ and $\Phi(\cdot)$, we have two variables to deal with:

$$\text{RR} = P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma] \tag{7}$$

$$\text{TPR} = P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma, (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}]. \tag{8}$$

Setting Objective Function

In this section, we first prove an interesting relation between RR and the efficient code condition in **Theorem** 1. Based on this relation, we set the objective function $L(\mathbf{W})$.

Theorem 1. *For any given γ, RR$(= P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma])$ is constant regardless of $\mathbf{W}$ **if** the hash code is efficient, that is, each bit of the hash code has a 50% chance of being* 1 *or* 0*, and different bits of the hash code are independent of one another.*

Proof. See the supplementary material. □

From **Theorem** 1, RR is constant for $\mathbf{W} \in \mathcal{W}$. Note that, in Eqs. 4 and 6, the denominators of both the precision and recall are constant if RR is constant, and the numerators are the TPRs. This means that both precision and recall can be maximized by maximizing TPR when **Condition 2** is true. Thus, our goal becomes simply to maximize TPR. Inspired by this relation, we set our objective function as $L(\mathbf{W}) = \sum_{\gamma=0}^{k} \text{TPR}(\mathbf{W}, \gamma)$ to summarize the TPR for all values of the threshold γ, and write problem (2) as

$$\max_{\mathbf{W} \in \mathcal{W}} \sum_{\gamma=0}^{k} \text{TPR}(\mathbf{W}, \gamma). \tag{9}$$

Note that if there exists a single $\mathbf{W}$ that maximizes TPR for all γ, it should be the solution to problem (9). Additionally, it is very interesting that maximizing the objective function can also be interpreted as minimizing the expected Hamming distance between similar data pairs:

$$\sum_{\gamma=0}^{k} \text{TPR}(\mathbf{W}, \gamma) \propto \sum_{\gamma=0}^{k} \text{recall}(\mathbf{W}, \gamma) \tag{10}$$

$$= \sum_{\gamma=0}^{k} P[d_H(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma | (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}] \tag{11}$$

$$= \sum_{\gamma=0}^{k} \{(k+1-\gamma) P[d_H(\mathbf{z}_i, \mathbf{z}_j) = \gamma | (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}]\} \tag{12}$$

$$= (k+1) \cdot 1 - E[d_H(\mathbf{z}_i, \mathbf{z}_j) | (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}]. \tag{13}$$

Proposed Method

Step 1: Uncorrelated Component Analysis

As mentioned earlier, we aim to maximize $L(\mathbf{W})$ subject to **Condition 1&2**. However, we cannot tackle this problem directly, because it is NP-hard.

Lemma 2. *The problem of maximizing $\sum_{\gamma=0}^{k} TPR(\mathbf{W}, \gamma)$ subject to the condition that the bits of the hash code are independent and each bit has a 50% chance of being* 1 *or* 0 *is NP-hard.*

Proof. See the supplementary material. □

As in (Gong and Lazebnik 2011; Weiss, Torralba, and Fergus 2008; Wang, Kumar, and Chang 2012), we first relax the independence assumption of **Condition 2**, and require the bits of the hash code to be uncorrelated. This relaxation simplifies the problem, but **Lemma** 3 implies that it is still NP-hard.

Lemma 3. *The problem of maximizing $\sum_{\gamma=0}^{k} TPR(\mathbf{W}, \gamma)$ subject to the condition that the bits of the hash code are uncorrelated and each bit of hash code has a 50% chance of being* 1 *or* 0 *is NP-hard.*

Proof. See the supplementary material. □

Since the NP hardness of our problem is mainly a result of the combinatorial nature of the uncorrelatedness constraint on $\mathbf{z}$, we instead require the components of $\mathbf{y}$ to be uncorrelated. Our problem can now be formulated as maximizing $L(\mathbf{W})$ subject to the constraint:

$$E[\mathbf{y}\mathbf{y}^T] = E[\mathbf{W}^T \mathbf{x}\mathbf{x}^T \mathbf{W}] = \mathbf{D}, \tag{14}$$

for some diagonal matrix $\mathbf{D} \in \mathbb{R}^{k \times k}$.

This relaxed problem turns out to be tractable. **Theorem** 4 describes a UCA method to find $\mathbf{W}$ such that Eq. 14 holds.

Theorem 4. *For $\mathbf{y} = \mathbf{W}^T\mathbf{x}, E[\mathbf{y}\mathbf{y}^T] = \mathbf{D}$ if and only if $\mathbf{W} = \mathbf{W}_{PCA}\mathbf{K}_{PCA}^{-\frac{1}{2}}\mathbf{Q}\mathbf{D}^{\frac{1}{2}}$, where $\mathbf{W}_{PCA}$ is the principal component analysis (PCA) transformation matrix, $\mathbf{K}_{PCA} = E[\mathbf{W}_{PCA}^T \mathbf{x}\mathbf{x}^T \mathbf{W}_{PCA}] \in \mathbb{R}^{d \times d}$, $\mathbf{Q} \in \left\{\widetilde{\mathbf{Q}} | \widetilde{\mathbf{Q}}^T \widetilde{\mathbf{Q}} = \mathbf{I}, \widetilde{\mathbf{Q}} \in \mathbb{R}^{d \times k}\right\}$, and $\mathbf{D}$ is a diagonal matrix.*

Proof. Considering that the PCA transformation matrix ($\mathbf{W}_{PCA} \in \mathbb{R}^{d \times k}$) satisfies Eq. 14, we start from PCA transformation matrix. As $\mathbf{W}_{PCA}$ is invertible, an arbitrary transform matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$ can be represented as

$$\mathbf{W} = \mathbf{W}_{PCA}\mathbf{A} \text{ where } \mathbf{A} \in \mathbb{R}^{d \times k}. \tag{15}$$

Let $\mathbf{W}_{UCA} \in \mathbb{R}^{d \times k}$ denote any $\mathbf{W}$ that satisfies Eq. 14. Then, $\mathbf{W}_{UCA}$ can also be represented in the form of Eq. 15, and $\mathbf{W}_{UCA}$ should satisfy Eq. 14:

$$E[\mathbf{y}_{UCA}\mathbf{y}_{UCA}^T] = E[\mathbf{A}_{UCA}^T \mathbf{W}_{PCA}^T \mathbf{x}\mathbf{x}^T \mathbf{W}_{PCA}\mathbf{A}_{UCA}] \tag{16}$$

$$= \mathbf{A}_{UCA}^T \mathbf{K}_{PCA} \mathbf{A}_{UCA} = \mathbf{D}, \tag{17}$$

where $\mathbf{K}_{PCA} = E[\mathbf{W}_{PCA}^T \mathbf{x}\mathbf{x}^T \mathbf{W}_{PCA}]$, $\mathbf{y}_{UCA} = \mathbf{W}_{UCA}^T \mathbf{x}$, $\mathbf{W}_{UCA} = \mathbf{W}_{PCA}\mathbf{A}_{UCA}$. Multiplying both sides of Eq. 17 by $\mathbf{D}^{-\frac{1}{2}}$, we have

$$\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{UCA}^T \mathbf{K}_{PCA}\mathbf{A}_{UCA}\mathbf{D}^{-\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{D}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I}, \tag{18}$$

$$\mathbf{I} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{UCA}^T \mathbf{K}_{PCA}\mathbf{A}_{UCA}\mathbf{D}^{-\frac{1}{2}}$$
$$= (\mathbf{K}_{PCA}^{\frac{1}{2}}\mathbf{A}_{UCA}\mathbf{D}^{-\frac{1}{2}})^T \mathbf{K}_{PCA}^{\frac{1}{2}}\mathbf{A}_{UCA}\mathbf{D}^{-\frac{1}{2}} \tag{19}$$

where, $\mathbf{K}_{PCA}^{\frac{1}{2}}$ denote one $\mathbf{F}$ $s.t.$ $\mathbf{K}_{PCA} = \mathbf{F}^T\mathbf{F}$.

Eq. 19 holds *if and only if* $\mathbf{K}_{PCA}^{\frac{1}{2}}\mathbf{A}_{UCA}\mathbf{D}^{-\frac{1}{2}} \in \left\{\widetilde{\mathbf{Q}} | \widetilde{\mathbf{Q}}^T \widetilde{\mathbf{Q}} = \mathbf{I}, \widetilde{\mathbf{Q}} \in \mathbb{R}^{d \times k}\right\}$. Then, $\mathbf{A}_{UCA}$ can be written as

$$\mathbf{A}_{UCA} = \mathbf{K}_{PCA}^{-\frac{1}{2}}\mathbf{Q}\mathbf{D}^{\frac{1}{2}}. \tag{20}$$

where $\mathbf{Q} \in \left\{\widetilde{\mathbf{Q}} | \widetilde{\mathbf{Q}}^T \widetilde{\mathbf{Q}} = \mathbf{I}, \widetilde{\mathbf{Q}} \in \mathbb{R}^{d \times k}\right\}$. Finally, combining Eqs. 15 and 20,

$$\mathbf{W}_{UCA} = \mathbf{W}_{PCA}\mathbf{A}_{UCA} = \mathbf{W}_{PCA}\mathbf{K}_{PCA}^{-\frac{1}{2}}\mathbf{Q}\mathbf{D}^{\frac{1}{2}}. \tag{21}$$

□

From Eq. 21, we can obtain infinitely many transformation matrices $\mathbf{W}_{UCA}$ that satisfy Eq. 14, because there are infinitely many $\mathbf{Q}$ and $\mathbf{D}$. Among them, in **Step 2**, we seek the best one $\mathbf{W}_{UCA}^*$ by choosing $\mathbf{Q}^*$ and $\mathbf{D}^*$ to maximize $\sum_{\gamma=0}^{k} \text{TPR}(\mathbf{W}, \gamma)$.

Step 2: Determining $\mathbf{D}^*$

In this section, we determine $\mathbf{D}^*$ so as to maximize $\sum_{\gamma=0}^{k}$ TPR$(\mathbf{W},\gamma)$. The first step is to prove the following lemma.

Lemma 5. *The TPR does not depend on* $\mathbf{D}$.

Proof. See the supplementary material. □

From **Lemma 5**, we are now free to use any diagonal matrix as $\mathbf{D}^*$. For simplicity, we will set $\mathbf{D}^*$ to be the identity matrix ($\mathbf{I}$):

$$\mathbf{W}^*_{UCA} = \mathbf{W}_{PCA}\mathbf{K}_{PCA}^{-\frac{1}{2}}\mathbf{Q}^*. \quad (22)$$

Step 2: Determining $\mathbf{Q}^*$

From Eq. 13, maximizing $\sum_{\gamma=0}^{k}$ TPR$(\mathbf{W},\gamma)$ is equivalent to minimizing $E[d_H(\mathbf{z}_i,\mathbf{z}_j)|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}]$. The Hamming distance property then leads to:

$$E[d_H(\mathbf{z}_i,\mathbf{z_j})|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] \quad (23)$$

$$= E[\sum_{n=1}^{k} d_H(z_{i,n},z_{j,n})|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] \quad (24)$$

$$= E[\sum_{n=1}^{k}\mathbf{d_n}|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] = \sum_{n=1}^{k} E[\mathbf{d_n}|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] \quad (25)$$

$$= \sum_{n=1}^{k} Pr(\mathbf{d_n}=1|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}) \quad (26)$$

$$= \sum_{n=1}^{k}\{1 - Pr(\mathbf{d_n}=0|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M})\}, \quad (27)$$

where $\mathbf{d_n} = d_H(\mathbf{z}_{i,n},\mathbf{z}_{j,n})$, $\mathbf{z}_{i,n}$ is the n^{th} bit of $\mathbf{z}_i$ and $\mathbf{z}_{j,n}$ is the n^{th}bit of $\mathbf{z}_j$. $Pr(\mathbf{d_n}=0|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M})$ can then be interpreted as the probability of the n^{th} bits of $\mathbf{z}_i$ and $\mathbf{z}_j$ being the same. Let us consider the special case of the data points being distributed as below:

$$\mathbf{x}_i \overset{i.i.d.}{\sim} Normal(0,\mathbf{K}_{org}), \quad (28)$$

$$\mathbf{x}_i|[\mathbf{x}_j,(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] \overset{i.i.d.}{\sim} Normal(\mathbf{x}_j,\mathbf{K}_{s,org}). \quad (29)$$

where $\mathbf{K}_{org}$ denote the covariance matrix of data, and $\mathbf{K}_{s,org}$ denote the covariance matrix of similar data pairs and can be calculated as $\mathbf{K}_{s,org} = E[(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}]$. The projected data $\mathbf{y}=\mathbf{W}^T\mathbf{x}$ then have the following distribution:

$$\mathbf{y}_i \overset{i.i.d.}{\sim} Normal(0,\mathbf{K}), \quad (30)$$

$$\mathbf{y}_i|[\mathbf{y}_j,(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] \overset{i.i.d.}{\sim} Normal(\mathbf{y}_j,\mathbf{K}_s), \quad (31)$$

$$\text{where } \mathbf{K}=\mathbf{W}^T\mathbf{K}_{org}\mathbf{W},\ \ \mathbf{K}_s=\mathbf{W}^T\mathbf{K}_{s,org}\mathbf{W}. \quad (32)$$

In addition, by marginalizing, the distribution of the n^{th} components of $\mathbf{y}_i$ and $\mathbf{y}_j$ can be represented as:

$$\mathbf{y}_{i,n} \overset{i.i.d.}{\sim} Normal(0,\sigma_n^2), \quad (33)$$

$$\mathbf{y}_{i,n}|[\mathbf{y}_{j,n},(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}] \overset{i.i.d.}{\sim} Normal(\mathbf{y}_{j,n},\sigma_{s_n}^2), \quad (34)$$

where σ_{s_n} is the $(n,n)^{th}$ element of $\mathbf{K}_s$ and σ_n is the $(n,n)^{th}$ element of $\mathbf{K}$. We can now express $Pr(\mathbf{d}_n|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M})$ in terms of sc_n (where $sc_n = \frac{\sigma_n}{\sigma_{s_n}}$):

Lemma 6. $Pr(\mathbf{d}_n=0|(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}) = \frac{1}{\pi}tan^{-1}(sc_n)+\frac{1}{2}$

Proof. See the supplementary material □

Lemma 6 indicates that the probability of the nth bit of a similar data pair being the same increases as σ_n increases and σ_{s_n} decreases. Using the result of **Lemma 6**, **Theorem 7** concludes that the optimal $\mathbf{Q}^*$ is $\mathbf{V}$.

Theorem 7. $\sum_{\gamma=0}^{k}$ *TPR*$(\mathbf{W},\gamma)$ *is maximized when* $\mathbf{Q}=\mathbf{V}$, *where the columns of* $\mathbf{V}$ *are the eigen vectors of* $\mathbf{K}'_s$ *with the smallest* k *eigen values, and* $\mathbf{K}'_s = \mathbf{K}_{PCA}^{-\frac{T}{2}}\mathbf{W}_{PCA}^T\mathbf{K}_{s,org}\mathbf{W}_{PCA}\mathbf{K}_{PCA}^{-\frac{1}{2}}$.

Proof. See the supplementary material. □

Observing that the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{K}'_s$ with the smallest k eigenvalues, we can obtain $\mathbf{V}$ by performing an eigenvalue decomposition of $\mathbf{K}'_s$. In conclusion, the final version of the transformation matrix $\mathbf{W}^*_{UCA}$ for the proposed hash function can be written as follows:

$$\mathbf{W}^*_{UCA} = \mathbf{W}_{PCA}\mathbf{K}_{PCA}^{-\frac{1}{2}}\mathbf{V}, \quad (35)$$

where the columns of $\mathbf{V}$ are the eigenvectors with the smallest k eigenvalues and $\mathbf{W}_{PCA}$ is a PCA transformation matrix of the data $\mathbf{x}$.

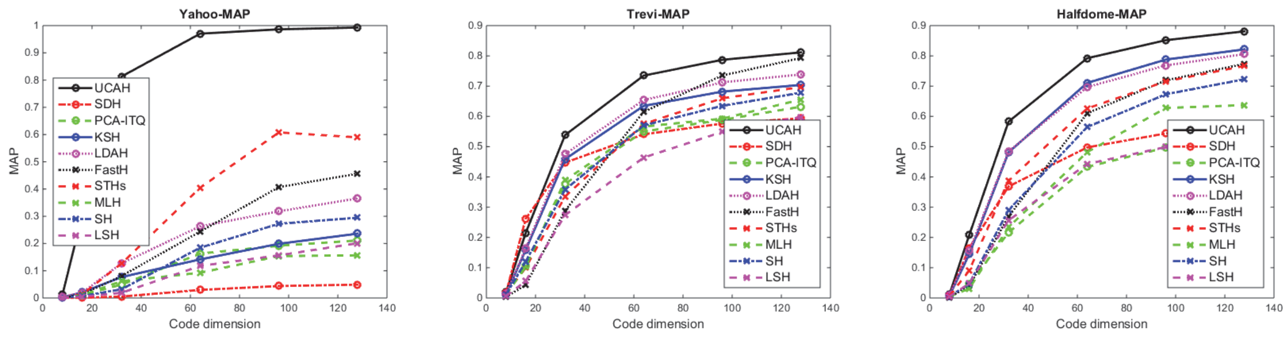
Experiment

We performed image retrieval experiments on three image benchmarks: Yahoo! (Thomee et al. 2013), Trevi and Halfdome photo-tourism (Snavely, Seitz, and Szeliski 2006). The Yahoo! duplicate image collection consists of 360K images, each of which has been subjected to 60 transformations. The photo-tourism project dataset consists of local image patches taken from Flickr photos of various landmarks. For the Trevi and Halfdome dataset, we only used data points from classes containing at least seven data points. We also performed the scalability test, by adding 1M distractor images from the Flickr-1M (Huiskes and Lew 2008) dataset to the database of Yahoo! as the false positive samples. The images of each dataset were divided into query and database. 1000 images were chosen at random for the query, and the rest formed the database. 5K images in the database were randomly sampled and used for training. Each channel of the color image was represented as the 320-dimensional GIST descriptor (Lin et al. 2013; 2014), and the gray image was represented as the 384-dimensional GIST descriptor.

We evaluated the proposed UCAH, and compared it against six state-of-the-art supervised methods (MLH (Norouzi and Fleet 2011), KSH (Liu et al. 2012), supervised STH (STHs) (Zhang et al. 2010), LDAH (Strecha et al. 2012), SDH (Shen et al. 2015), and FastH (Lin et al. 2014)) and three unsupervised methods, LSH (Indyk and Motwani 1998), PCA-ITQ (Gong and Lazebnik 2011), and SH (Weiss, Torralba, and Fergus 2008). We used the MATLAB code and parameters provided by the authors of each

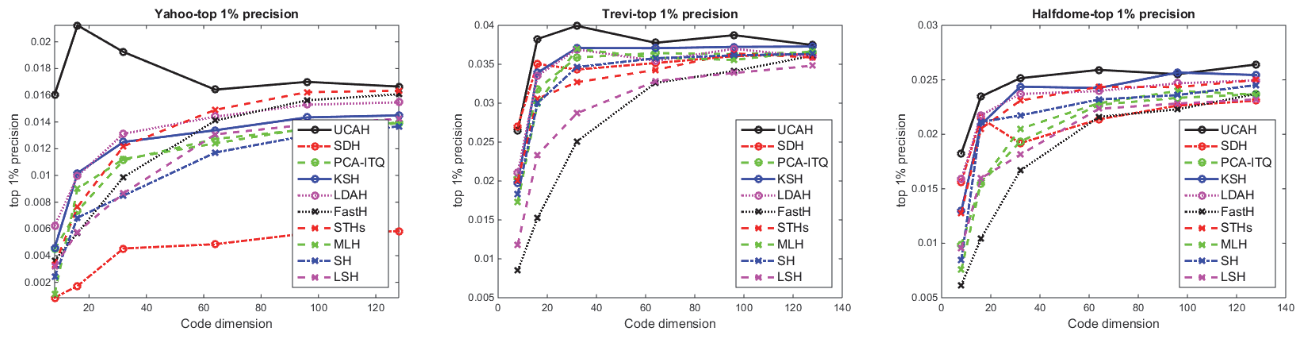
Table 1: Training and encoding time of each method on the Halfdome dataset.

	UCAH	SDH	PCA-ITQ	KSH	LDAH	FastH	STHs	MLH	SH	LSH
T_{train}(s)	1.1	485.3	1.9	843.2	50.1	440.0	14.2	554.2	0.3	-
T_{encode}(s)	0.06	0.27	0.06	0.27	0.06	21.78	93.97	0.06	3.78	0.06



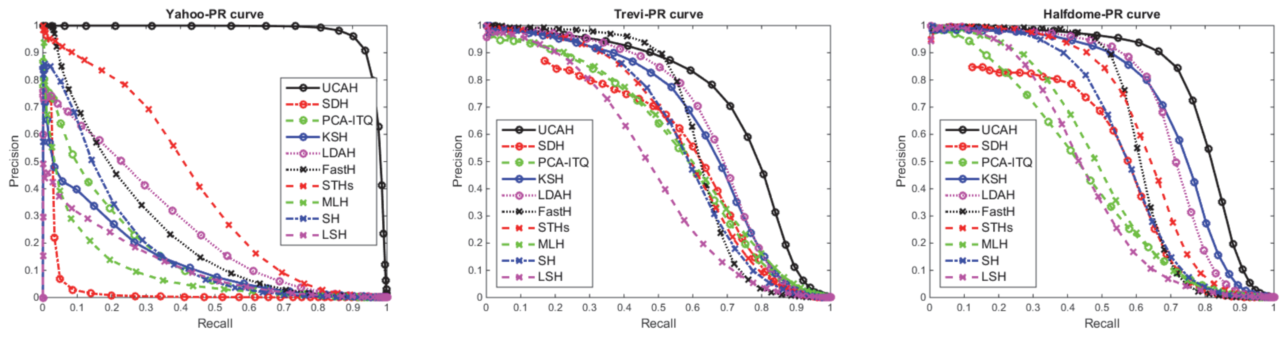
(a) (b) (c)

Figure 1: MAP curve on (a) Yahoo! (b) Trevi (c) Halfdome



(a) (b) (c)

Figure 2: Top1% precision curve on (a) Yahoo!, (b) Trevi, (c) Halfdome



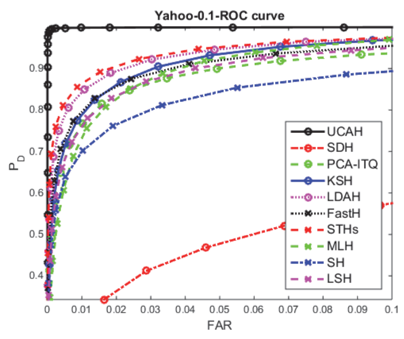
(a) (b) (c)

Figure 3: Precision-recall curve for k = 64 on (a) Yahoo!, (b) Trevi, (c) Halfdome

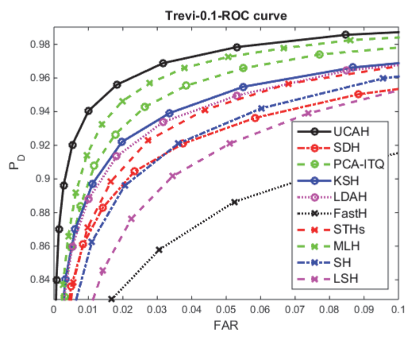
method. All the experiments were implemented in MATLAB and performed on a desktop with an Intel Core i7 CPU at 3.30 GHz with 32 GB RAM.

Results

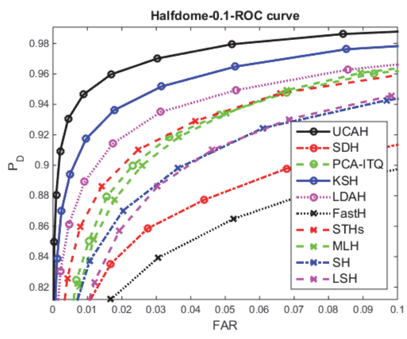
Evaluation results Figure 1 and 2 show the Hamming ranking performance evaluated by the mean average precision (MAP) and Top1%-precision (Top1) where Top1 is



(a)

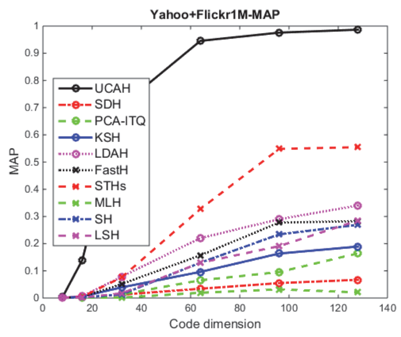


(b)

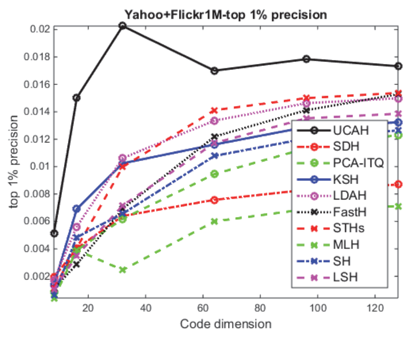


(c)

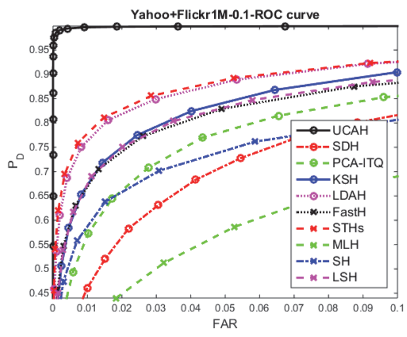
Figure 4: Scaled ROC curve for k=64: (a) Yahoo!, (b) Trevi, (c) Halfdome



(a)



(b)



(c)

Figure 5: Scalability test result on Yahoo!+Flickr1M (a) MAP (b) Top1% precision (c) ROC curve for k=64

evaluated by the accuracy at the top 1% ranked data. We can see that UCAH outperforms compared methods for most cases in terms of Top1 and MAP. SDH achieves the highest value of MAP with $k \leq 16$ and Top1 with $k = 8$ on Trevi dataset, but its performance diminishes rapidly with increasing code lengths.

Figure 3 and 4 show that UCAH is also competitive in terms of the precision-recall and ROC curves. Note that all the ROC graphs are appropriately scaled for clarity; the full-scale ROC curves are given in the supplementary material. The precision-recall and ROC curves for other choices of k are given in the supplementary material. Again, UCAH consistently produced superior results to the comparative methods on all of the test benchmarks for other choices of k.

Figure 5 shows the result of scalability test on Yahoo! and Flickr1M dataset. Compared to the results on Yahoo! dataset in Figure 1, 2, and 4, UCAH outperforms all other methods with a larger margin indicating that UCAH scales well to the large number (>1M) of data samples.

Computation time We used the Halfdome dataset, and recorded the time spent on training and encoding for $k = 128$ with 1K query, 39K database, and 3K training data. The result in Table 1 shows that UCAH is the most efficient for training among the supervised methods. UCAH is also one of the most efficient methods for encoding hash code, because it adopts the simplest form of hash function. SH adopts the same form of hash function, but requires more time for complex thresholding step. KSH, SDH, STHs, and FastH require more time for the kernel computation and decision tree evaluation steps.

Discussion

Comparison with Other PCA-based Methods

PCA is widely used in many hashing schemes. Like SH, ITQ, QS-rank (Zhang, Zhang, and Shum 2012), and semi-supervised hashing (SSH), UCAH uses PCA transformation. We will contrast these other PCA-based methods with UCAH. SH uses d PCA-transformed components, but further evaluates the k smallest eigenvalues for each component. It then chooses the k smallest eigenvalues out of dk eigenvalues. ITQ and Qs-rank use a truncated PCA-transformation matrix for dimension reduction, while UCAH uses a full-PCA transformation matrix to search the solution $\mathbf{W}_{UCA}$ among all $\mathbb{R}^{d\times k}$ matrix $\mathbf{W}$ because the full-PCA projection matrix has full rank and is invertible. ITQ multiplies an orthogonal matrix after the PCA-transformation to minimize the quantization error, restricting $\mathbf{A}$ to be an orthogonal matrix, while UCAH does not restrict the matrix $\mathbf{A}$ to be orthogonal. QS-rank uses PCA for dimension reduction and adopts a new metric instead of the Hamming distance. SSH relaxes the orthogonality condition of PCA by revising the original objective function, and aims to optimize the hash code in terms of empirical error.

Conclusion

In this paper, we have proposed a novel scalable hashing method for the large-scale ANN problem. We first devised a family of transformation matrices $\mathbf{W}_{UCA}$ that produce efficient codes, and then selected the $\mathbf{W}^*_{UCA}$ that optimized the retrieval performance. Experiments were conducted on three publicly available datasets, and the resulting ROC and precision-recall curves demonstrated that the proposed method outperforms other state-of-the-art methods in most cases.

Acknowledgments

This work was supported by the IT R&D program of MSIP/KEIT. [15501-15-1016, Instant 3D object based Join&Joy content technology supporting simultaneous participation of users in remote places and enabling realistic experience]

References

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2013. Optimized product quantization for approximate nearest neighbor search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 817–824. IEEE.

He, K.; Wen, F.; and Sun, J. 2013. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heo, J.-P.; Lin, Z.; and Yoon, S.-E. 2014. Distance encoded product quantization. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2139–2146. IEEE.

Huiskes, M. J., and Lew, M. S. 2008. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*. New York, NY, USA: ACM.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, 604–613. ACM.

Jegou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33(1):117–128.

Kulis, B., and Grauman, K. 2012. Kernelized locality-sensitive hashing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34(6):1092–1104.

Lin, G.; Shen, C.; Suter, D.; and Hengel, A. v. d. 2013. A general two-step approach to learning-based hashing. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2552–2559. IEEE.

Lin, G.; Shen, C.; Shi, Q.; van den Hengel, A.; and Suter, D. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1971–1978. IEEE.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.-G.; and Chang, S.-F. 2012. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2074–2081. IEEE.

Norouzi, M., and Fleet, D. J. 2011. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 353–360.

Shen, F.; Shen, C.; Liu, W.; and Tao Shen, H. 2015. Supervised discrete hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 37–45.

Snavely, N.; Seitz, S. M.; and Szeliski, R. 2006. Photo tourism: exploring photo collections in 3d. *ACM transactions on graphics (TOG)* 25(3):835–846.

Strecha, C.; Bronstein, A. M.; Bronstein, M. M.; and Fua, P. 2012. Ldahash: Improved matching with smaller descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34(1):66–78.

Thomee, B.; Huiskes, M.; Bakker, E.; and Lew, M. 2013. An evaluation of content-based duplicate image detection methods for web search. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, 1–6.

Wang, J.; Kumar, S.; and Chang, S.-F. 2012. Semi-supervised hashing for large-scale search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34(12):2393–2406.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *Advances in neural information processing systems*, 1753–1760.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 18–25. ACM.

Zhang, X.; Zhang, L.; and Shum, H.-Y. 2012. Qsrank: Query-sensitive hash code ranking for efficient-neighbor search. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2058–2065. IEEE.

Zhao, K.; Lu, H.; and Mei, J. 2014. Locality preserving hashing. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada*, 2874–2881.