Compressed Hashing

Yue Lin* Rong Jin† Deng Cai* Shuicheng Yan‡ Xuelong Li§
*State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou, China
†Dept. of Computer Science & Eng., Michigan State University, East Lansing, MI, U.S.A.
‡Dept. of Electrical and Computer Engineering, National University of Singapore, Singapore
§OPTical IMagery Analysis and Learning, Chinese Academy of Sciences, China
linyue29@gmail.com, rongjin@cse.msu.edu, dengcai@cad.zju.edu.cn, eleyans@nus.edu.sg, xuelong_li@opt.ac.cn

Abstract

Recent studies have shown that hashing methods are effective for high dimensional nearest neighbor search. A common problem shared by many existing hashing methods is that in order to achieve a satisfied performance, a large number of hash tables (i.e., long codewords) are required. To address this challenge, in this paper we propose a novel approach called Compressed Hashing by exploring the techniques of sparse coding and compressed sensing. In particular, we introduce a sparse coding scheme, based on the approximation theory of integral operator, that generate sparse representation for high dimensional vectors. We then project sparse codes into a low dimensional space by effectively exploring the Restricted Isometry Property (RIP), a key property in compressed sensing theory. Both of the theoretical analysis and the empirical studies on two large data sets show that the proposed approach is more effective than the state-of-the-art hashing algorithms.

1. Introduction

Nearest Neighbor (NN) search is one of the most essential problem in machine learning and has found applications in many computer vision tasks [19, 14]. Given the intrinsic difficulty of exact nearest neighbor search, many hashing algorithms are proposed for Approximate Nearest Neighbor (ANN) search [5, 6, 21]. The key idea of these approaches is to generate binary codewords for high dimensional data points that preserve the similarity between data points.

Many hashing algorithms found their theoretic root in random projection, which is proved to be an effective method for preserving pairwise distances. One problem with most random projection theories is that, in order to achieve a small error in distance preservation, a large number of random projections are required. Even more disturbingly, the number of required random projections depends on the size of data set, making it less attractive for large databases. For example, according to Jonson Lindenstrauss Theorem [10], to preserve the pairwise distances for a database of n data points, the number of needed random projections is $O(\ln n/\epsilon^2)$, where ϵ is the relative error in distance approximation. Therefore, it is not a surprise the random projection based hashing methods do not perform well for short codes. To address this problem, several learning based hashing methods are proposed. Most of these algorithms learn the binary codes by exploiting the spectral properties of the data affinity matrix. In spite of the success for relative small codes, these learning based approaches often fail to make significant improvement as code length increases [11].

In this paper, we propose a novel approach called *Compressed Hashing* to address this challenge by exploring the techniques of sparse coding [8] and compressed sensing [7]. The main idea is based on the Restricted Isometry Property (RIP) [3], which states that for any *sparse* vector, the random projection is able to preserve the Euclidean distances between high dimensional vectors with an overwhelming probability. It is important to distinguish the RIP condition from Jonson Lindenstrauss (JL) theorem. The probability in JL theorem is related to each vector, namely for any vector, there is a chance for the property to be failed. As a result, to preserve the pairwise distance for *all* the data points, the number of random projections has to be dependent on the size of the database. In contrast, the probability in RIP condition is related to the random matrix, namely nearly all random matrix generated by iid Gaussian will be able to preserve Euclidean distance for *all*

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.64

data points. Because of this difference, the performance guarantee delivered by RIP applies to *all* vectors, making the number of random projections independent from the size of the database.

Note that the RIP only applies to the sparse vectors. In order to meet the sparse requirement in RIP, we use a sparse coding scheme similar to [14] that generates *compact sparse* codes for high dimensional vectors. The sparse codes are generated based on the approximation theory of integral operator and well preserve the relationship between vectors. Given the sparse codes, we then explore the theory of compressed sensing [7] to project the sparse codes into a low dimensional space, and generate the hash codes based on the low dimensional projection.

2. Related Work

Most hashing methods can be classified into two categories: the random projection based methods and the learning based methods.

The most notable approach for random projection based methods is Locality Sensitive Hashing [5, 1], which offers a sub-linear time search by trying to hash similar data points into the same entry of a hash table(s). One drawback of many LSH approaches is that in order to preserve the locality of the data points, they have to generate long codewords, leading to large storage space and high computational cost. Both entropy-based LSH approach [16] and Multi-Probe LSH [15] are proposed to reduce the storage limitation at the sacrifice of increasing the query time. Moreover, several studies extend LSH from the Euclidean space to the Reproducing Kernel Hilbert Space (RKHS) [12, 17].

A common problem of random projection based hashing algorithms is that in order to achieve a satisfied performance, they require a large number of hash tables and long codewords. To address this limitation, many learning based algorithms are proposed. Example algorithms in this category include Spectral Hashing [21], Unsupervised Sequential Projection Learning for Hashing [19], Spherical Hashing [9], Kernel-based Supervised Hashing [13], Anchor Graph Hashing [14] and so on. However, the learning based hashing algorithms often work well for short codewords but fail to make significant improvement as code length increases [11].

3. Compressed Hashing

In this section, we will first describe the sparse coding scheme, which is based on the kernel density estimation, followed by the approach of projecting sparse vectors into low dimensional space using compressed sensing theory.

3.1. Sparse Coding using RBF Kernel

Our goal is to create the sparse representations that preserve the relationship between the data points. Let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a large database, where $\mathbf{x}_i \in \mathbb{R}^d$ is a vector with $d \gg 1$ dimension. Assume x_a and x_b are two data points in $\mathcal{D}$. we consider the following measure

$$g(\mathbf{x}_a, \mathbf{x}_b) = \frac{1}{N} \sum_{i=1}^{N} \kappa(\mathbf{x}_a, \mathbf{x}_i)\kappa(\mathbf{x}_b, \mathbf{x}_i) \tag{1}$$

Here $\kappa(\mathbf{x}, \mathbf{x}')$ is a RBF kernel given by

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|_2^2}{2h^2}\right)$$

Since $g(\mathbf{x}_a, \mathbf{x}_b)$ defined in Eq.(1) can also be written as $g(\mathbf{x}_a, \mathbf{x}_b) = \mathbf{u}^\top(\mathbf{x}_a)\mathbf{u}(\mathbf{x}_b)$, where $\mathbf{u}(\mathbf{x}) = \frac{1}{\sqrt{N}}(\kappa(\mathbf{x}, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}, \mathbf{x}_N))^\top$, we can use $\mathbf{u}(\mathbf{x})$ as the representation of $\mathbf{x}$. The main problem of using the representation $\mathbf{u}(\mathbf{x})$ is its high dimensionality as the size of $\mathbf{u}(\mathbf{x})$ is N, where N is the size of the database $\mathcal{D}$. We address this problem by exploring the concentration inequality of integral operator [4].

Let $\mathcal{H}$ be the Reproducing Kernel Hilbert Space (RKHS) endowed with the kernel $\kappa(\cdot, \cdot)$, and let $|f|_{\mathcal{H}} = \sqrt{\langle f, f\rangle_{\mathcal{H}}}$ be the norm defined in $\mathcal{H}$. We define an integral operator $L : \mathcal{H} \mapsto \mathcal{H}$ as

$$L[f](\cdot) = \frac{1}{N} \sum_{i=1}^{N} \kappa(\mathbf{x}_i, \cdot) f(\mathbf{x}_i) \tag{2}$$

It is easy to see that

$$g(\mathbf{x}_a, \mathbf{x}_b) = \langle \kappa(\mathbf{x}_a, \cdot), L(\kappa(\mathbf{x}_b, \cdot))\rangle_{\mathcal{H}}$$

Based on the concentration inequality for integral operator, we can approximate L by a low rank operator, leading to a compact representation for data points in $\mathcal{D}$. More specifically, let $\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_m$ be m anchor points generated from $\mathcal{D}$. One can simply use random sampling or k-means [1] to generate the anchor points [14] and we will discuss this in our experiments.

Using the anchors, we approximate L by $\widehat{L}$

$$\widehat{L}[f](\cdot) = \frac{1}{m} \sum_{i=1}^{m} \kappa(\widehat{\mathbf{x}}_i, \cdot) f(\widehat{\mathbf{x}}_i) \tag{3}$$

[1] The anchor points can be set to the cluster centers returned by the k-means algorithm with cluster number m. There is no need to wait the k-means converges and we can stop it after p iterations, where p is a parameter (5 is usually enough).

As a result, we obtain a compact representation $\widehat{\mathbf{u}}(\mathbf{x}) = \frac{1}{\sqrt{m}}(\kappa(\widehat{\mathbf{x}}_1, \cdot), \ldots, \kappa(\widehat{\mathbf{x}}_m, \cdot))^\top$. The following theorem shows that with a high probability, for any $\mathbf{x}_a$ and $\mathbf{x}_b$, $|\mathbf{u}^\top(\mathbf{x}_a)\mathbf{u}(\mathbf{x}_b) - \widehat{\mathbf{u}}^\top(\mathbf{x}_a)\widehat{\mathbf{u}}(\mathbf{x}_b)|$ is small if m is sufficiently large.

Theorem 1. *Let $\widehat{L}$ be the integral operator constructed based on m anchor points, as shown in Eq.(3). With a probability $1-\delta$, for any $\mathbf{x}_a$ and $\mathbf{x}_b$, we have*

$$\left|\mathbf{u}^\top(\mathbf{x}_a)\mathbf{u}(\mathbf{x}_b) - \widehat{\mathbf{u}}^\top(\mathbf{x}_a)\widehat{\mathbf{u}}(\mathbf{x}_b)\right| \leq \frac{2\ln(2/\delta)}{m} + \sqrt{\frac{2\ln(2/\delta)}{m}}$$

The Theorem 1 can be directly derived from the Lemma 2 in [18]. As indicated by Theorem 1, with appropriate m, it is sufficient to represent each data point $\mathbf{x}$ by

$$\widehat{\mathbf{u}}(x) = \frac{1}{\sqrt{m}}\left(\kappa(\widehat{\mathbf{x}}_1, \mathbf{x}), \ldots, \kappa(\widehat{\mathbf{x}}_m, \mathbf{x})\right)^\top$$

Subsequently, we normalize $\widehat{\mathbf{u}}(\mathbf{x})$ by $|\widehat{\mathbf{u}}(\mathbf{x})|_1$ so that the sum of entries in the vector representation is equal to 1, which leads to the representation of $\mathbf{x}$

$$\mathbf{z}(\mathbf{x}) = \frac{1}{\sum_{i=1}^m \kappa(\widehat{\mathbf{x}}_i, \mathbf{x})}\left(\kappa(\widehat{\mathbf{x}}_1, \mathbf{x}), \ldots, \kappa(\widehat{\mathbf{x}}_m, \mathbf{x})\right)^\top$$

This normalization step makes all the vectors $\mathbf{z}$ more comparable. More importantly, based on the observation that all vectors lying in a ℓ_1 ball, to preserve pairwise distance, it is sufficient to keep the largest entries in vectors. More specifically, let $h_s : \mathbb{R}^m \mapsto \mathbb{R}^m$ be a vector function, where the output of $h_s(\mathbf{z})$ is the vector $\mathbf{z}$ with everything set to zero except for its first s entries with the largest absolute values. Then, according to [7], for any vectors $\mathbf{z} \in \mathbb{R}^m$ in a ℓ_1 ball, i.e., $|\mathbf{z}_1|_1 \leq 1$ and $|\mathbf{z}_2|_1 \leq 1$, there exists a constant C, independent from $\mathbf{z}_1$, $\mathbf{z}_2$ and m, such that for any integer $s > 0$, we have

$$|(\mathbf{z}_1 - \mathbf{z}_2) - h_s(\mathbf{z}_1 - \mathbf{z}_2)|_2 \leq \frac{C}{\sqrt{s+1}}|\mathbf{z}_1 - \mathbf{z}_2|_1 \quad (4)$$

Note that we only need to keep a part of information for each point, which leads to the sparse representation. It is this reduced requirement that makes it possible to generate a small number of projections to accurately preserve the distances between the vectors.

Based on Eq.(4), we are determined to keep the information of *a small number* of nearest anchors for each point and we get the final sparse representation of $\mathbf{x}$

$$\mathbf{z}(\mathbf{x}) = \frac{1}{\sum_{i=1}^m \psi(\widehat{\mathbf{x}}_i, \mathbf{x})}\left(\psi(\widehat{\mathbf{x}}_1, \mathbf{x}), \ldots, \psi(\widehat{\mathbf{x}}_m, \mathbf{x})\right)^\top \quad (5)$$

where

$$\psi(\widehat{\mathbf{x}}_i, \mathbf{x}) = \begin{cases} \kappa(\widehat{\mathbf{x}}_i, \mathbf{x}) & \mathbf{x}_i \in \mathcal{S}(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \quad i = 1, 2, ..., m$$

$\mathcal{S}(\mathbf{x})$ stands for the s nearest anchors of $\mathbf{x}$ among anchor points. It is important to note that the above sparse coding scheme is also used by Anchor Graph Hashing (AGH) [14]. AGH uses this coding strategy to speed up the spectral analysis of the data while our motivation is to generate sparse codes to meet the sparse requirement in RIP. In the next subsection, we show how the approximate nearest neighbor search can be effectively performed using these sparse vectors by exploring the RIP condition.

3.2. Approximate Sparse Representation by Random Projection

Given the sparse representation of the data points, our next goal is to create binary codes that approximate the distances between the sparse vectors. For any data point $\mathbf{x}$ and its sparse representation $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^m$ generated by the proposed sparse coding scheme, we create a K-bits code $\mathbf{b}(\mathbf{x}) = (b_1(\mathbf{x}), \ldots, b_K(\mathbf{x}))^\top$ by a linear hashing method: it first projects $\mathbf{z}$ to a low dimensional space by K linear operators $\{\phi_i \in \mathbb{R}^m, i \in [K]\}$, i.e., $y_i(\mathbf{z}) = \phi_i^\top \mathbf{z}, i \in [K]$; it then creates a binary codeword $\mathbf{b}(\mathbf{x})$ for $\mathbf{z}(\mathbf{x})$ by thresholding each dimension by its median, i.e.,

$$b_i(\mathbf{x}) = \frac{1}{2}(1 + \text{sgn}(y_i(\mathbf{z}(\mathbf{x})) - \bar{y}_i)), i \in [K] \quad (6)$$

where $\bar{y}_i$ is the median value in the vector $\phi_i^\top \mathbf{z}(\mathbf{x}_i)$, and $\text{sgn}(a) = 1$ if $a > 0$ and -1 otherwise. We choose $\{\phi_i\}_{i=1}^K$ randomly by drawing $\{\phi_{ij}, i \in [K], j \in [m]\}$ independently from a Gaussian distribution $\mathcal{N}(0, 1/K)$. The detailed theoretical analysis is showed as follows.

We first present the Restricted Isometry Property of random matrices as below.

Theorem 2. *(Restricted Isometry Property)(inferred from Lemma 3.1 in [2]) Assume $K < m$ and let Φ be a random matrix of size $m \times K$ whose entries are i.i.d. Gaussian with mean zero and variance $1/K$. If s/m is small enough and $K = cs\log(m/s)$, where c is a constant independent of s, there exists a positive constant $\delta_s < 1$ such that with an overwhelming probability, we have the following inequality hold for any $\mathbf{z} \in \mathbb{R}^m$ with at most s non-zero entries*

$$(1-\delta_s)|\mathbf{z}|_2^2 \leq \frac{m}{K}|\Phi^\top \mathbf{z}|_2^2 \leq (1+\delta_s)|\mathbf{z}|_2^2$$

Below, we present the property of the random projection for sparse vectors, which follows directly from the Theorem 2 and the inequality in Eq.(4). We now state our main result.

Theorem 3. *Assume $K < m$ and let $\Phi \in \mathbb{R}^{m\times K}$ be a random matrix whose entries are i.i.d. Gaussian with mean zero and variance $1/K$. Choose $s \ll m$ and $K = cs\log(m/s)$, where c is a constant independent of s. Let $h_s : \mathbb{R}^m \mapsto \mathbb{R}^m$ be a vector function, where the output of $h_s(\mathbf{z})$ is the vector $\mathbf{z}$ with everything set to zero except for its first s entries with the largest absolute values. Let $L = |\Phi^\top h_s(\mathbf{z}_1 - \mathbf{z}_2)|_2$, there exists a positive constant $\delta_s < 1$ such that with an overwhelming probability, for any two vectors $|\mathbf{z}_1|_1, |\mathbf{z}_2|_1 \leq 1$, we have*

$$L \geq \sqrt{\frac{K(1-\delta_s)}{m}}(|\mathbf{z}_1 - \mathbf{z}_2|_2 - \frac{C}{\sqrt{s+1}}|\mathbf{z}_1 - \mathbf{z}_2|_1)$$

$$L \leq \sqrt{\frac{K(1+\delta_s)}{m}}(|\mathbf{z}_1 - \mathbf{z}_2|_2 + \frac{C}{\sqrt{s+1}}|\mathbf{z}_1 - \mathbf{z}_2|_1)$$

Theorem 3 provides both lower and upper bounds for the ℓ_2 norm of the difference between two sparse vectors, which justify the random projection approach for approximating the difference between sparse vectors in a ℓ_1 ball. Algorithm 1 shows the steps of the proposed algorithm.

3.3. Computational Complexity Analysis

Given N data points with the dimensionality d, the computational complexity of Compressed Hashing in the training stage is as follows:

1. $O(pNmd)$: k-means with p iterations to generate m groups (Step 1 in Alg. 1).
2. $O(Nm(d+s))$: Generate sparse representation for data points (Step 2 in Alg. 1).
3. $O(NmK)$: Compute the embedding of the data (Step 3 in Alg. 1).
4. $O(NK)$: Compute the hashing codes with respect to the median values (Step 4 in Alg. 1).

As we can see, the overall computational complexity of Compressed Hashing training is dominated by the k-means clustering step, which is $O(pNmd)$. In the testing stage, given a query point, Compressed Hashing needs $O(m(d+s))$ to compress the query point into a sparse representation and needs $O(mK)$ to obtain the binary codes.

4. Experiments

In this section, we evaluate our Compressed Hashing (CH) algorithm on the high dimensional nearest neighbor search problem. Two large scale real-world data sets **SIFT1M** and **GIST1M** are used in our experiments.

Algorithm 1 Compressed Hashing

Input:

$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$: the database;
K: the number of bits for hashing codes;
m: the number of anchor points;
$h > 0$: the kernel width used by RBF;
s: the number of nearest anchors in sparse coding;

1: Apply k-means to compute m cluster centers from the data points in $\mathcal{D}$, and use them as the anchor points $V \in \mathbb{R}^{m\times d}$.
2: Generate sparse representation $Z \in \mathbb{R}^{n\times m}$ for data points in $\mathcal{D}$, based on the anchor points in V, using Eq.(5).
3: Generate linear projections $\Phi \in \mathbb{R}^{m\times K}$ by drawing $\Phi_{j,k}$ from $\mathcal{N}(0, 1/K)$ independently. Compute the embedding of data by $Y' = Z\Phi$.
4: Compute the hashing code Y by thresholding $Y'_{i,k}$ with respect to the median $\bar{y}_k$.

Output:

The model:
The anchor points: $\{\widehat{\mathbf{x}}_i\}_{i=1}^m, \widehat{\mathbf{x}}_i \in \mathbb{R}^d$;
The random projection matrix: $\Phi \in \mathbb{R}^{m\times K}$; Binary hashing codes for the training samples: $Y \in \{0,1\}^{N\times K}$

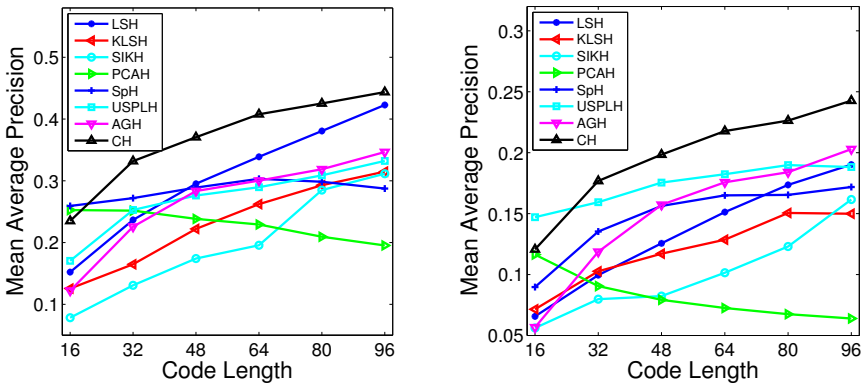
Both data sets contain one million image features. SIFT feature is represented by a 128-dim vector and GIST feature is represented by a 960-dim vector. The data sets are publicly available[2].

For each data set, we randomly select 10k data points as the queries and use the remaining to form the gallery database. We use the same criterion as in [19], that a returned point is considered to be a true neighbor if it lies in the top 2 percentile points closest (measured by the Euclidian distance in the original space) to the query. For each query, all the data points in the database are ranked according to their Hamming distances to the query. We evaluate the retrieval results by the Mean Average Precision (MAP) and the precision-recall curve. In addition, we also report the training time and the testing time (the average time used for each query) for all the methods.

4.1. Compared Algorithms and Settings

We evaluate our proposed Compressed Hashing method by comparing with seven state-of-the-art methods for high dimensional nearest neighbor search: Locality Sensitive Hashing (**LSH**) [5], Kernelized Locality Sensitive Hashing (**KLSH**) [12], Shift-Invariant Kernel Hashing (**SIKH**) [17], Principal Component Analysis Hashing (**PCAH**) [20], Spectral Hashing (**SpH**) [21], Unsupervised Sequential Projection Learning for Hashing (**USPLH**) [19], Anchor Graph Hashing (**AGH**) [14].

[2] http://corpus-texmex.irisa.fr



(a) SIFT1M (b) GIST1M

Figure 1. The Mean Average Precision of all the algorithms on SIFT1M and GIST1M data sets.

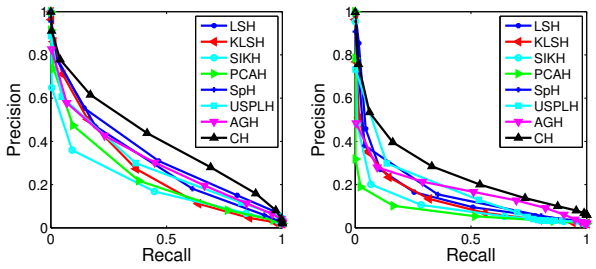
Table 1. Training and testing time of all algorithms on SIFT1M and GIST1M data sets.

Method	SIFT1M				GIST1M			
	Training Time (s)		Test Time (s)		Training Time (s)		Test Time (s)	
	$K = 32$	$K = 64$	$K = 32$	$K = 64$	$K = 32$	$K = 64$	$K = 32$	$K = 64$
LSH	0.3	0.6	1.1×10^{-6}	1.9×10^{-6}	1.4	2.1	2.7×10^{-6}	3.0×10^{-6}
KLSH	10.5	10.7	14.6×10^{-6}	16.2×10^{-6}	29.5	30.7	27.2×10^{-6}	38.0×10^{-6}
SIKH	1.1	2.3	3.4×10^{-6}	3.9×10^{-6}	1.8	3.4	13.9×10^{-6}	27.5×10^{-6}
PCAH	6.5	7.4	1.1×10^{-6}	2.0×10^{-6}	49.2	52.3	2.7×10^{-6}	3.0×10^{-6}
SpH	25.8	88.2	28.0×10^{-6}	101.9×10^{-6}	65.3	130.8	40.2×10^{-6}	116.3×10^{-6}
USPLH	234.1	484.8	3.1×10^{-6}	3.7×10^{-6}	1357.2	2732.0	5.3×10^{-6}	9.2×10^{-6}
AGH	144.7	184.2	55.7×10^{-6}	72.0×10^{-6}	242.5	279.4	83.7×10^{-6}	95.6×10^{-6}
CH	93.4	98.2	53.5×10^{-6}	54.4×10^{-6}	194.3	210.5	64.1×10^{-6}	71.5×10^{-6}

We implement LSH, PCAH by ourselves, and use the codes provided by the authors for the algorithms KLSH, SIKH, SpH, USPLH and AGH. To run KLSH, we use the Gaussian kernel and sample 300 training points to form the empirical kernel map. The bandwidth of Gaussian kernel is set to 0.3. AGH with two-layer is used in our comparison for its superior performance over AGH with one-layer [14]. Both our CH method and the AGH need an anchor-based sparse coding step and we use the exactly same strategy. There are three parameters: the number of anchor points (m), the number of iterations (p) in k-means and the number of nearest anchors in sparse coding (s). We empirically set $m = 200$, $p = 5$ and $s = 50$ for both algorithms. For both methods, the Gaussian kernel width parameter h is empirically[3] set to be 0.3 on SIFT1M and 0.5 on GIST1M.

4.2. Experimental Results

Figure 1 shows the MAP curves of all the algorithms on the SIFT1M and GIST1M data sets. We can see that the random projection based algorithms (LSH, SIKH and KLSH) have a low MAP when the code length is short. As the code length increases, the performances of both the two methods consistently increases. On the other hand, the learning based algorithms, such as SpH and PCAH, have a high MAP when the code length is short. However, they fail to make significant improvements as the code length increases. Particularly, the performance of PCAH decreases as the code length increases. This is consistent with previous work [19] and is probably because that most of the data variance is contained in the top few principal directions so that the later bits are calculated using the low-variance projections, leading to the poorly discriminative codes [19]. By combining the techniques of sparse coding and com-



(a) SIFT1M (b) GIST1M

Figure 2. The precision-recall curves of all algorithms on SIFT1M and GIST1M data sets for the codes of 64 bits.

[3]We estimate h by randomly choose 3000 samples and let h equal to the average of the pairwise distances.

pressed sensing, we successfully preserve the information in the low dimensional space. As a result, the proposed CH method achieves satisfied performances on both data sets and almost outperforms its competitors for all code lengths. Figure 2 presents the precision-recall curves of all the algorithms on two data sets with the code of 64 bits.

Table 1 shows both the training time and test time for different algorithms on two data sets. Considering the training time, USPLH is the most expensive to train. The random projection based algorithms are relatively efficient, especially the LSH. The proposed CH algorithm uses similar but less training time than AGH due to its fast process in converting the sparse vectors to binary codes. Both of them spend most of the training time on k-means step. In terms of the test time, LSH and PCAH are the most efficient methods. Both of them simply need a matrix multiplication and a thresholding to obtain the binary codes. SpH consumes much longer time than other methods since it needs to compute the analytical eigenfunctions involving the calculation of trigonometric functions. The proposed CH method takes similar test time with AGH. The most expensive part in CH is to obtain the sparse representation for the query point.

5. Conclusion

In this paper, we have developed a hashing algorithm for high dimensional nearest neighbor search by combining the techniques of sparse coding and compressed sensing. The key idea is to first generate compact sparse codes based on the theory of density function estimation for high dimensional vectors that preserve the relationship between the data points, and then project sparse vectors into low dimensional space to preserve pairwise distances by exploring the RIP condition. Empirical studies on the large data sets show that the proposed algorithm scales well to data size and significantly outperforms the state-of-the-art hashing methods in retrieval accuracy. In the future, we plan to further explore anchors selection methods that are both effective and computationally efficient for large data sets.

Acknowledgments

This work was supported by the National Basic Research Program of China(973 Program) under Grant 2011CB302206, Army Research Office (ARO Award W911NF-11-1-0383) and National Nature Science Foundation of China (Grant Nos: 61125106, 61222207, 91120302).

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[2] E. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346:589–592, 2008.

[3] E. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51:4203 – 4215, 2005.

[4] E. J. Candès, L. Demanet, and L. Ying. Fast computation of fourier integral operators. *SIAM J. Scientific Computing*, 29(6):2464–2493, 2007.

[5] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253–262, 2004.

[7] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

[8] D. L. Donoho and M. Elad. Optimally sparse representation in general (non-orthogonal) dictionaries via ℓ_1 minimization. In *PNAS*, pages 2197–2202, 2003.

[9] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, pages 2957–2964, 2012.

[10] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26:189–206, 1984.

[11] A. Joly and O. Buisson. Random maximum margin hashing. In *CVPR*, pages 873–880, 2011.

[12] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[13] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

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

[15] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[16] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA*, pages 1186–1195, 2006.

[17] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009.

[18] S. Smale and D.-X. Zhou. Geometry on probability spaces. *Constr Approx*, 30:311–323, 2009.

[19] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *ICML*, 2010.

[20] X. Wang, L. Zhang, F. Jing, and W. Ma. Annosearch: Image auto-annotation by search. *CVPR*, 2, 2006.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.