JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 499-515 (2017)
DOI: 10.6688/JISE.2018.33.2.13

The Normalized Distance Preserving Binary Codes and Distance Table*

HONGWEI ZHAO[1,2], ZHEN WANG[1], PINGPING LIU[1,2,+] AND BIN WU[1]
[1]*School of Computer Science and Technology*
[2]*Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education*
Jilin University
ChangChun, 130012 P.R. China
E-mail: {zhaohw; liupp}@jlu.edu.cn; {wangzhen14; wubin14}@mails.jlu.edu.cn

In the Euclidean space, the approximate nearest neighbors (ANN) search measures the similarity degree through computing the Euclidean distances, which owns high time complexity and large memory overhead. To address these problems, this paper maps the data from the Euclidean space into the Hamming space, and the normalized distance similarity restriction and the quantization error are required to satisfy. Firstly, the encoding centers and their binary labels are obtained through a lookup-based mechanism. Then, the candidate hashing functions are learnt under supervision of the binary labels, and the ones which satisfy the entropy criterion are selected to boost the distinctiveness of the learnt binary codes. During the training procedure, multiple groups of the hashing functions are generated based on different kinds of centers, which can weaken the inferior influence of the initial centers. The data with minimal average Hamming distances are returned as the nearest neighbors. In the Hamming space, different Euclidean distances may be substituted by one identical value, thus a distance table is predefined to distinguish the similarity degrees among the data pairs with the same Hamming distance. The final experimental results show that our algorithm is superior to many state-of-the-art methods.

Keywords: approximate nearest neighbors search, binary codes, hashing algorithm, semi-supervised learning, entropy

1. INTRODUCTION

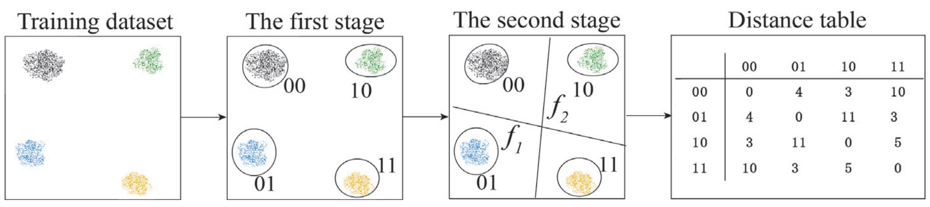
The approximate nearest neighbors (ANN) search aims to retrieve the samples with minimal distances to the query samples. Currently, ANN search is applied in applications such as image/video retrieval [1, 2], recognition [3], image classification [4, 5], and pose estimation [6]. Traditional ANN search is achieved through comparing Euclidean distances, which causes large memory overhead and high time complexity. To simplify the ANN search procedure, the interest way is mapping data into low dimensional binary vectors, where ANN search can be done in the Hamming space [7-10]. In computer vision and multimedia retrieval tasks, the bag-of-words (BoW) model captures the correlations between local features and visual objects, where two different, naturally associated entities correspond to its two dimensions or views. To fully explore the duality between the two views, collaborative hashing [11] is intended to preserve both the entity similari-

Received September 19, 2015; revised June 19 & September 3, 2016; accepted October 8, 2016.
Communicated by Chia-Feng Juang.
* This work was supported in part by the grants of the National Natural Science Foundation of China (No. 61101155), the Natural Science Foundation of Jilin Province (Nos. 20140101184JC and 20150520063JH), Postdoctoral Science Foundation of China (No. 2015M571363) and Project 2015051 Supported by Graduate Innovation Fund of Jilin University.
+ Corresponding author.

ties in each view and the interrelationship between views. The classical encoding mechanism computes the projection results of the data with hyper planes [7, 23, 24]. The random hashing functions in locality sensitivity hashing (LSH) [7] are independent from training data. To achieve reasonable search accuracy, the length of the binary bits in the random methods should be long enough. This will cause reduced search speed and large memory overhead. Liu *et al*. [12] propose a learning based framework to learn the bilinear hashing functions. To improve the collision probability of the binary codes of the similar entities, Liu *et al*. [13] design multiple linear projections for generating hash bits. The binary codes in spectral hashing (SH) [15] are obtained through partitioning spectral graph. The data distribution needs to be uniform in SH, which is often unrealistic. Terasawa *et al*. [16] map data into binary codes according to their relative position with hyper spherical. Gong *et al*. [17] define the restriction of the similarity error and find binary codes through aligning rotated data to the vertices of hyper binary cube. The restriction of the quantization error in *k*-means clustering method [18] is employed in *k*-means hashing (KMH) [20]. He *et al*. [20] adopt an iterative mechanism to minimize the similarity error and the quantization error, which makes the binary codes adaptive to data distribution. To learn hashing functions from the data with complex inherent structure, Liu *et al*. [19] propose a structure sensitive hashing based on cluster prototypes, which explicitly exploits both global and local structures.

Based on the encoding mechanism, existing hashing algorithms are roughly divided into two categories [20]. The first type is referred to as lookup-based methods, such as KMH method [20], product quantization (PQ) algorithm [21] and optimized product quantization (OPQ) approach [22]. Lookup-based methods establish encoding centers table, and the out of samples are mapped into the same binary codes as their nearest centers. The second type of hashing algorithm is the Hamming-based methods, which include LSH method [7], spectral hashing [15], minimal loss hashing [23] and semi-supervised hashing [24], and they map data into binary codes according to the projection signs with linear hashing functions. The binary codes in lookup-based methods are more distribution adaptive than those in Hamming-based methods [20]. The encoding time complexity of lookup-based methods is higher. For obtaining *m*-bit binary codes, lookup-based methods should compute the Euclidean distances between each unseen data and 2^m encoding centers. In contrast, Hamming-based methods just calculate the projection signs of the data with *m* linear hashing functions. The encoding time complexity of lookup-based methods is $O(2^m)$, and that of Hamming-based methods is only $O(m)$.

In this paper, a two-step mechanism [25-28] is adopted to integrate the lookup-based mechanism and the Hamming-based trick. The training flowchart of our algorithm is shown in Fig. 1. The first stage shows the encoding centers and their binary labels are



	00	01	10	11
00	0	4	3	10
01	4	0	11	3
10	3	11	0	5
11	10	3	5	0

Fig. 1. The training flowchart of our algorithm. In the first stage, the distribution adaptive binary labels are learnt to approximate their Euclidean distance relationship. After that, hashing functions are learnt under the supervision of the binary labels to improve encoding efficiency. In the final stage, the distance table is established to resort the retrieval results.

learnt through a lookup-based mechanism. In the second stage, aiming to reduce the time complexity of the encoding procedure, linear hashing functions are learnt under the supervision of the binary labels. To distinguish the data pairs with the same Hamming distance, the distance table is established.

As the gradient descent algorithm is adopted in the first stage, the convergence results would be impacted by the initial encoding centers. To weaken such an adverse effect, different kinds of encoding centers are initialized, and multiple groups of hashing functions are generated. In ANN search stage, the data with minimal average Hamming distances are returned as retrieval results.

The rest of this paper is organized as follows. In section 2, the normalized distance similarity restriction and the quantization constraint are introduced. The processes of learning binary labels and linear hashing functions are described in section 3. In section 4, multiple hashing functions are built to return the data with minimal average Hamming distance as nearest neighbors. To further improve ANN search performance, a distance table is established to distinguish the data pairs with the same Hamming distance. The comparative experiments are given in section 5. The conclusions are drawn in section 6.

2. THE RESTRICTIONS

2.1 The Normalized Distance Similarity Restriction

To ensure the Euclidean nearest neighbors can be retrieved in the Hamming space, the distance relationship computed on the basis of binary codes should be consistent with that in the Euclidean space [15, 23, 29, 30].

In this paper, the above goal is achieved through two steps. Firstly, the Euclidean and Hamming distances are normalized through Eq. (1). nd_{ij} represents the normalized Euclidean distance between x_i and x_j, nh_{ij} denotes the corresponding normalized Hamming distance. d_{ij} and h_{ij} are the original Euclidean and Hamming distances. $min(\cdot)$ returns the minimal distance value in each distance set, and $max(\cdot)$ computes the maximum value. d and h separately represent the Euclidean and Hamming distance set. The normalization process can help measure the performance of the obtained binary codes. For example, three data pairs' Euclidean distances are 1, 3 and 5, and their Hamming distances are 0, 1 and 2. If not adopting the normalization process, it's hard to judge the ANN search performance of the binary labels. After normalizing the distance values, the two kinds of distances become 0, 0.5 and 1. It's obvious to know the distance relationship in the Euclidean space is well preserved in the Hamming space.

$$\begin{aligned} nd_{ij} &= \frac{d_{ij} - min(d)}{max(d) - min(d)} \\ nh_{ij} &= \frac{h_{ij} - min(h)}{max(h) - min(h)} \end{aligned} \tag{1}$$

$$SE = \sum_{i,j=1}^{n} |1 - \frac{nh_{ij} + \varepsilon}{nd_{ij} + \varepsilon}| \tag{2}$$

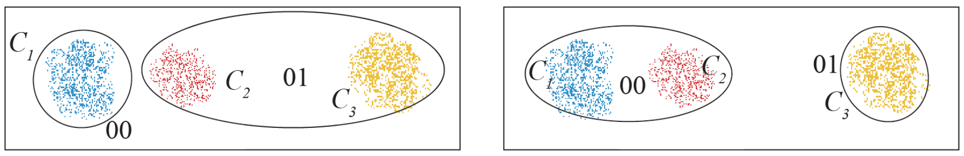
Secondly, the value of SE in Eq. (2) is minimized to ensure the similarity degrees in both spaces are consistent. ε is a minimal constant value, in case nd_{ij} and nh_{ij} equal to zero. n is the amount of training data.

2.2 The Quantization Restriction

The data with m-bit binary labels can be divided into 2^m categories, but the number of the data is larger than 2^m. Many data are mapped into identical binary codes. If not taken data distribution into consideration, the data with an identical binary code may not be near neighbors. As shown in Fig. 2 (a), data groups C_2 and C_3 belong to the same clustering category, but the distance between C_2 and C_1 is smaller than that between C_2 and C_3. To avoid such a situation in Fig. 2 (a), the quantization restriction [18] defined in Eq. (3) needs to be satisfied.

$$QE = \sum_{i=1}^{n} (x_i - C(x_i)) \tag{3}$$

$C(x_i)$ denotes the encoding center of data x_i. The near neighbors are mapped into the same binary codes to minimize the value of QE. As shown in Fig. 2 (b), the data groups C_1 and C_2 are mapped into the same binary code '00', and the binary code of C_3 is '01'.



(a) (b)

Fig. 2. The effect of the quantization restriction in generating binary labels. The quantization restriction requires the data with the same binary code should be near neighbors. In (a), data groups C_2 and C_3 are mapped into the same binary code, but they have larger distance. In (b), the quantization restriction is employed, which makes the near neighbors have the same binary code.

3. LINEAR HASHING FUNCTIONS

3.1 Learning Binary Labels

During the process of learning binary labels, the normalized distance similarity error and the quantization error are jointly minimized through the stochastic gradient descent algorithm [18]. The objective function E is defined in Eq. (4).

$$E = SE + QE \tag{4}$$

The time complexity of directly learning binary bits on the basis of the training data's integer dimensions is higher. To accelerate the convergence speed, the mechanism which divides the dimensions of training data into different sub-spaces [21, 22] is adopted. In each sub-space, the sub-labels are learnt based on only part of the data's dimensions. After that, the sub-labels in all sub-spaces are concatenated to form the final long binary label. Given mapping 128-dimension SIFT descriptors into 64-bit binary labels in 16 sub-spaces, 4-bit sub-labels are learnt according to 8 dimensions of the original data in each sub-space. Finally, all 4-bit sub-labels in 16 sub-spaces are concatenated to form the final 64-bit binary labels.

In each sub-space, the iterative mechanism for learning b-bit binary labels contains three basic steps:

Step 1: 2^b data are chosen as initial encoding centers, and they are randomly mapped into b-bit unique binary labels.

Step 2: Compute the distances between each data and all encoding centers, and the data are assigned to their nearest encoding centers.

Step 3: Assisted with stochastic gradient descent algorithm (*matlab* offers *fminunc* tool), the objective function E is minimized to find the encoding centers and their binary labels.

In each sub-space, the binary labels are generated through repeatedly executing steps 2 and 3. After convergence, the out of training samples are mapped into the same binary labels as their nearest encoding centers.

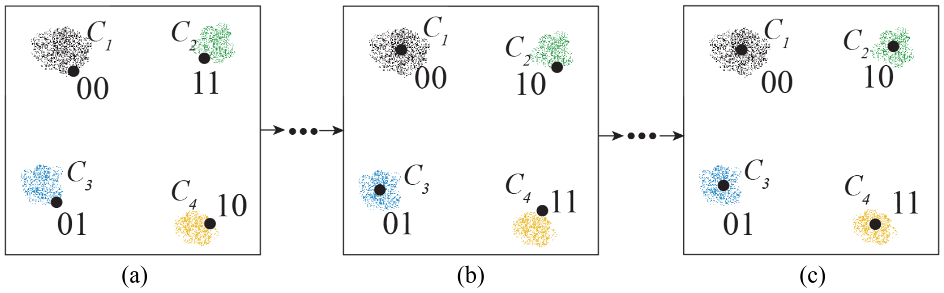


Fig. 3. Iterative steps for obtaining binary codes. In order to minimize the value of the objective function E, C_1 and C_3 are moving towards the centers of their own groups, C_2 and C_4 have exchanged their binary labels.

The iterative procedure for learning 2-bit binary labels is demonstrated in Fig. 3. In (a), four encoding centers are initialized, and they are randomly mapped into 2-bit binary labels. The quantization constraint is violated by all four encoding centers, and only the centers with binary labels '00' and '01' obey the normalized distance similarity restriction. During the following iterative steps as shown in (b), the encoding centers with binary labels '00' and '01' are moving towards their respective group centers to minimize the quantization error, while C_2 and C_4 have exchanged their binary labels to satisfy the normalized distance similarity restriction. When the objective function convergences in (c), the distance relationship computed on the basis of the obtained binary labels can approximate that in the Euclidean space.

3.2 Supervised Learning Hashing Functions

In section 3.1, the out of samples are encoded through the mechanism of lookup-based method, thus the time complexity is relative higher. Given obtaining *m*-bit binary codes, the encoding time complexity in section 3.1 is $O(2^m)$. In contrast, the encoding mechanism with linear hashing functions just computes *m* projection results, and the time complexity is only $O(m)$. Aiming to reduce the encoding time complexity, we should learn linear hashing functions.

Linear hashing functions map data into different signs, and the binary values are computed according to the projection signs as defined in Eq. (5).

$$b_j(x_i) = \begin{cases} 1 & f_j(x_i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$b_j(x_i)$ represents the binary value on the *j*th bit of data x_i. $f_j(x_i)$ denotes the hashing function for computing the binary value on the *j*th bit. As known from Eq. (5), the *j*th hashing function maps the data with different binary values on the *j*th bit into contrary signs. Therefore, the *j*th hashing function can be computed according to the distribution of the binary values on the *j*th bit. According to this rule, the data with *m*-bit binary labels are divided into 2^{m-1} groups, and SVM is employed to compute the hashing function which maps the data with different binary labels into different binary codes in each group.

As described above, 2^{m-1} hashing functions are obtained for computing the binary value on each bit. For the encoding stage, only one candidate hashing function is chosen through comparing their entropy values. The entropy criterion is defined as in Eq. (6), which demands the distribution of the encoded data should be uniform.

$$entropy(f_j) = -P(1)\log(P(1)) - P(0)\log(P(0)) \tag{6}$$

$entropy(f_j)$ denotes the entropy value of the hashing function f_j. $P(\cdot)$ represents the probability value. Entropy criterion implicitly requires that the encoded data should be uniformly distributed, which is similar to the bucket balance criterion. Large entropy value means that the amount of each kind of encoded data is alike. The distinctiveness of the binary codes computed through the selected linear hashing function is improved.

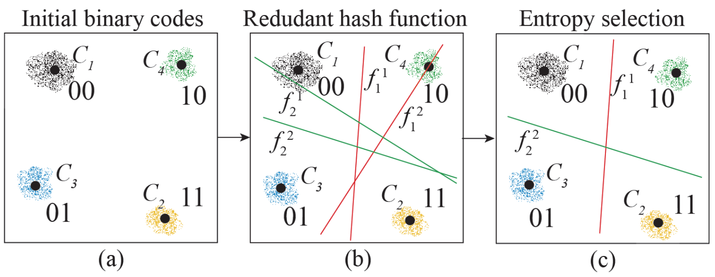


Fig. 4. The flowchart of obtaining linear hashing functions. In (b), the candidate hashing functions are learnt according to the data distribution in different groups. In (c), only one hashing function is selected through entropy criterion.

The flowchart of learning hashing functions on the basis of binary labels is shown in Fig. 4. When obtaining the hashing functions for computing the left first binary value, the data are divided into two groups: (C_1, C_4) and (C_2, C_3). In each group, the binary labels are different only on the left first bit. According to the distribution of the binary labels in the two groups, the candidate hashing functions f_1^1 and f_1^2 are obtained. f_i^j represents the ith candidate hashing function generated in the jth group. In (b), the data group C_4 is separated by f_1^2. The distribution of the data with binary values computed through f_1^2 is not uniform. Through comparing their entropy values, f_1^1 is selected. Similarly, f_2^2 is chosen for computing the left second binary value.

4. BOOSTING ANN SEARCH PERFORMANCE

4.1 Multiple Hashing Functions

In section 3.1, the convergence results are influenced by initial encoding centers. Aiming to weaken such an inferior impact, many different kinds of encoding centers are initialized to generate multiple groups of hashing functions.

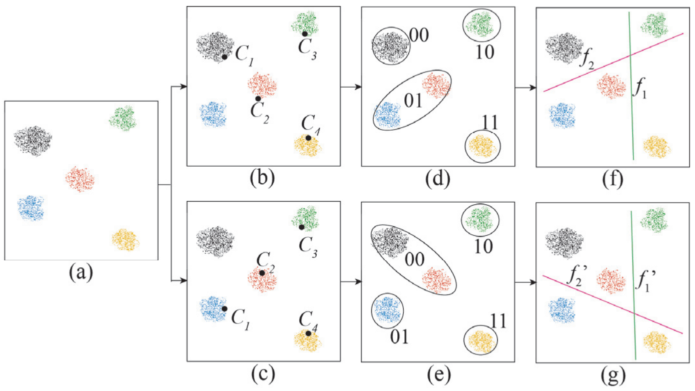


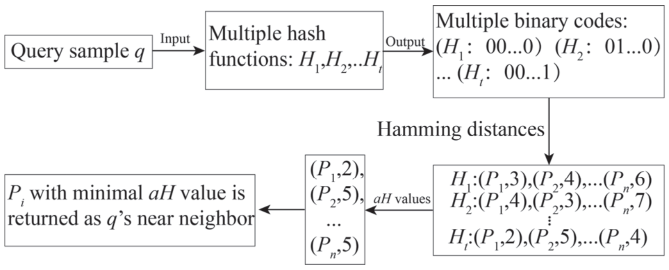
Fig. 5. The framework of training multiple groups of hashing functions. In (b) and (c), different kinds of data are initialized as encoding centers. As a result, multiple binary labels are generated in (d) and (e). According to these binary labels, multiple groups of hashing functions are learnt in (f) and (g).

The process of generating multiple groups of hashing functions is illustrated in Fig. 5. As shown in (b), (c), (d) and (e), multiple binary labels are learnt on the basis of different kinds of initial encoding centers. Then, multiple groups of hashing functions are generated under the supervision of multiple binary labels as in (f) and (g). Both (f_1, f_2) and (f_1', f_2') can independently generate 2-bit binary codes. The distance relation between the red and black data is identical to that between the red and blue data. If adopting one hashing functions group, only the black or blue data are considered as the nearest neighbors of the red data. To avoid such a situation, both groups of hashing functions are employed to return the data with minimal average Hamming distance as the nearest neighbors.

The retrieval mechanism based on multiple groups of hashing functions is established as in Fig. 6. Firstly, the query sample q is mapped into multiple binary codes according to hashing functions (H_1, H_2, …, H_t). H_i represents the i-th hashing functions group, which can independently calculate the binary codes. Secondly, the multiple Hamming distances between q and all data points (P_1, P_2, …, P_n) are calculated. Finally, the average Hamming distances are calculated, and the data with minimal values are returned as the nearest neighbors.

The average Hamming distance is defined in Eq. (7). $aH(x, y)$ returns the average Hamming distance between the samples x and y. d computes the Hamming distance. t is the number of the hashing functions groups.

$$aH(x,y) = \frac{1}{t}\sum_{i=1}^{t} d(H_i(x), H_i(y)) \tag{7}$$

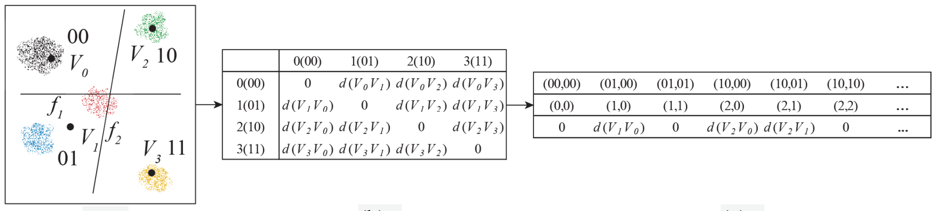

Query sample q
Input
Multiple hash functions: $H_1,H_2,..H_t$
Output
Multiple binary codes: (H_1: 00...0) (H_2: 01...0) ... (H_t: 00...1)
Hamming distances
H_1:(P_1,3),(P_2,4),...(P_n,6)
H_2:(P_1,4),(P_2,3),...(P_n,7)
H_t:(P_1,2),(P_2,5),...(P_n,4)
aH values
(P_1,2), (P_2,5), ... (P_n,5)
P_i with minimal aH value is returned as q's near neighbor
Fig. 6. The ANN search procedure based on multiple hashing functions.

4.2 Distance Table

m-bit binary codes can represent only m kinds of Hamming distances, thus different Euclidean distances are merged into the same Hamming value. For example, the data with binary code '01' and '10' are considered as the same nearest neighbors of the query sample with binary code '00'. The Hamming distances of both data pairs are 1, but their Euclidean distances may not be identical. To distinguish the difference among the data pairs with the same Hamming distance, a distance table is established.

For linear hashing functions, the mean values of the data with the same binary codes are considered as the centers, and the Euclidean distances among these centers are calculated as the elements of the distance table. The element $e(i, j)$ is computed through Eq. (8). V_i represents the center of $Bin(i)$ which is the binary representation of i. The process of establishing the distance table for 2-bit binary codes is demonstrated in Fig. 7. In (a), V_0, V_1, V_2 and V_3 represent the centers. In (b), the Euclidean distances among centers are computed as the elements of the distance table. $d(V_i, V_j)$ represents the Euclidean distance between V_i and V_j.

$$e(i,j) = d(V_i, V_j) = \left\| V_i - V_j \right\|^2 \tag{8}$$



	0(00)	1(01)	2(10)	3(11)
0(00)	0	$d(V_0V_1)$	$d(V_0V_2)$	$d(V_0V_3)$
1(01)	$d(V_1V_0)$	0	$d(V_1V_2)$	$d(V_1V_3)$
2(10)	$d(V_2V_0)$	$d(V_2V_1)$	0	$d(V_2V_3)$
3(11)	$d(V_3V_0)$	$d(V_3V_1)$	$d(V_3V_2)$	0

(00,00)	(01,00)	(01,01)	(10,00)	(10,01)	(10,10)	...
(0,0)	(1,0)	(1,1)	(2,0)	(2,1)	(2,2)	...
0	$d(V_1V_0)$	0	$d(V_2V_0)$	$d(V_2V_1)$	0	...

(a) (b) (c)

Fig. 7. The distance table for 2-bit binary codes.

The elements in the distance table are symmetry, and we can linearly store only half of the elements to reduce memory occupancy as in Fig. 7 (c). The element $e(i, j)$ can be found from the linear distance table through Eq. (9). $L(k)$ is the kth element in linear distance table.

$$e(i,j)=\begin{cases} L(\frac{i\cdot(i+1)}{2}+j) & i>j \\ L(\frac{j\cdot(j+1)}{2}+i) & \text{otherwise} \end{cases} \tag{9}$$

In section 4.1, multiple binary codes are obtained, and one distance table can be established according to each kind of binary codes. In ANN search task, the average values of the elements located in the corresponding position of all distance tables are calculated to re-sort the data with the same average Hamming distance.

5. EXPERIEMTNAL SETUP

5.1 Evaluation Standards

In this paper, the criterions of *recall* and *mAP* are used to evaluate the ANN search performance.

recall in Eq. (10) represents the fraction of the query samples' relevant data that are successfully retrieved. #(*retrieved relevant points*) is the number of retrieved relevant points, and #(*all relevant points*) returns the number of all relevant points.

$$recall=\frac{\#(retrieved\ relevant\ points)}{\#(all\ relevant\ points)} \tag{10}$$

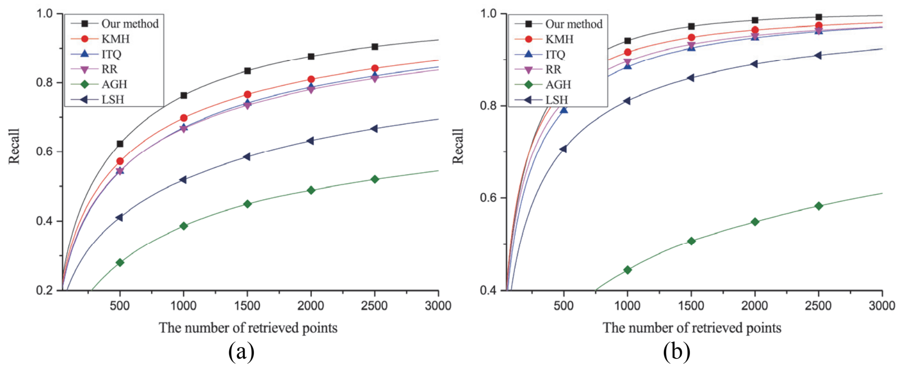
The value of *mAP* [31] in Eq. (11) expresses which position the ith positive data locates. $|Q|$ represents the number of testing samples and n_j returns the number of true positive data of the jth testing sample. $rank(i)$ is the ranking value of the ith positive data in final results.

$$mAP=\frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{n_j}\sum_{i=1}^{n_j}\frac{i}{rank(i)} \tag{11}$$

5.2 Experimental Results Compared with Other Methods

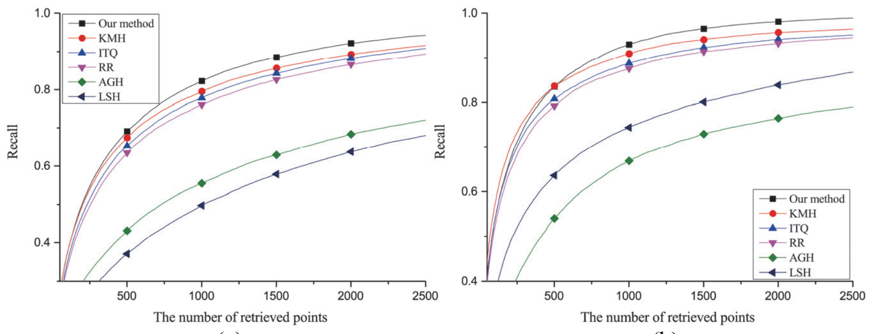
Three datasets are used to compare the ANN search performances of our method and the other five state-of-the-arts. *Labelme* [32] contains the 128-dimension SIFT obtained through the VL_SIFT tool [33], and the data in *SIFT*1*M* [20] are also SIFT descriptors. *Cifar*10 [34] stores the GIST descriptors. For *Labelme* and *SIFT*1*M*, 100000 samples are randomly selected as a training data set, and 10000 data are considered as testing samples. In *Cifar*10 database, 50000 training samples and 10000 testing data are used. Five comparative methods are *k*-means hashing (KMH) [20], iterative quantization hashing (ITQ) [17], RR [2], anchor graph hashing (AGH) [35] and locality sensitivity hashing (LSH) [7].

The recall curves of all methods in *Lableme*, *Cifar*10 and *SIFT*1*M* are separately shown in Figs. 8-10. The *mAP* values are demonstrated in Table 1.



(a) (b)

Fig. 8. The recall curves in *Labelme* database. The data are mapped into 32-bit binary codes in (a), and the number of binary bits in (b) is 64.



(a) (b)

Fig. 9. The recall curves in *Cifar*10 database. The data are mapped into 32-bit binary codes in (a), and the number of binary bits in (b) is 64.

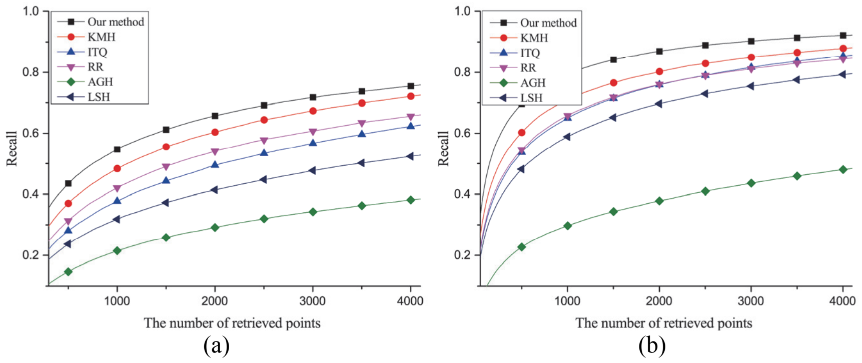


Fig. 10. The recall curves in *SIFT*1*M* database. The data are mapped into 32-bit binary codes in (a), and the number of binary bits in (b) is 64.

Table 1. The *mAP* values(%) of all methods. The data in *Labelme*, *Cifar*10 and *SIFT*1*M* are mapped into 32-bit and 64-bit binary codes respectively.

		Our method	KMH	ITQ	RR	AGH	LSH
Labelme	32-bit	**23.62**	22.02	19.87	21.11	3.45	16.92
	64-bit	**41.58**	40.46	37.25	39.81	4.69	32.54
*Cifar*10	32-bit	**9.02**	8.81	8.27	7.88	3.88	3.65
	64-bit	**19.32**	18.97	17.63	17.14	5.25	10.69
*SIFT*1*M*	32-bit	**4.97**	3.59	2.19	2.77	0.55	2.06
	64-bit	**13.12**	12.3	7.83	8.90	0.90	6.88

The experimental results show the algorithm proposed in this paper is superior to other state-of-the-art methods. The principle of ITQ [17] and RR [2] are similar. Both methods rotate the data to align to the vertices of one hyper binary cube, and the data are assigned the same binary codes as their nearest vertices. However, the restriction of quantization error, which makes encoding results adaptive to data distribution, is ignored by ITQ and RR. In AGH [14], the binary codes are obtained through partitioning the spectral graph, and the graph's nodes are learnt through *k*-means algorithm. As described in spectral hashing [20], both AGH and SH demand that the data distribution should be uniform. The data sets in our experiments do not follow this unrealistic requirement. The random hashing functions in LSH [3] are data independent, so the performance of LSH method does not obviously improve as the number of binary bits increasing. The normalized distance similarity restriction ensures the distance relationship in the Hamming space can approximate that in the Euclidean space, and the quantization restriction makes the hashing functions adaptive to data distribution. In this paper, both restrictions are required to satisfy, so the ANN search performances of the obtained binary codes are excellent.

Another key contribution of our method is greatly reducing the time complexity of encoding procedure. Table 2 shows the amount of the time used by all methods to map one million 128-dimension SIFT descriptors into binary codes. Our method, ITQ [21], RR [32] and LSH [3] belong to Hamming-based method, and the time complexity of mapping data into *m*-bit binary codes is only $O(m)$. In contrast, KMH [5] belongs to lookup-based method, and 2^m Euclidean distances between the query sample and encoding centers need to be computed and compared. The encoding time complexity of KMH is $O(2^m)$. As AGH [14] should establish spectral graph, it takes the longest time.

Table 2. The time(s.) consumed for mapping one million data into binary codes.

	Our method	KMH	ITQ	RR	AGH	LSH
32-bit	0.69	7.25	2.01	0.73	10.05	**0.42**
64-bit	1.09	10.37	2.83	1.16	23.17	**0.91**

All hashing methods can effectively reduce the memory occupancy through mapping data into binary codes. If 128-dimension SIFT descriptors are mapped into 64-bit binary codes, only 7.63M bytes is needed for storing 10^6 SIFT descriptors. Considering float point data are represented by 4-bit single-precision float point data, the compression ratio would be 64:1.

5.3 The Number of Binary Bits in Each Sub-Space

In this paper, the dimensions of training data are divided into sub-spaces to reduce the training time. Given obtaining B-bit binary codes, only b-bit binary codes need to be obtained in each sub-space, and the number of the sub-spaces is B/b. The training times and the ANN search performances with different b values are shown in Table 3 and Fig. 11. A large b makes a small number of sub-spaces, and the neighbor relationship can be kept. In contrast, a small b with a large amount of sub-spaces would lose neighbor information. For the training time complexity, a large b means lots of encoding centers and binary labels need to be learnt in each sub-space. As a result, the convergence procedure is slow, and plenty of time is spent. However, the training procedure can be done in real time with a small b. In this paper, b is set 4 to balance the training time complexity and the ANN search performance.

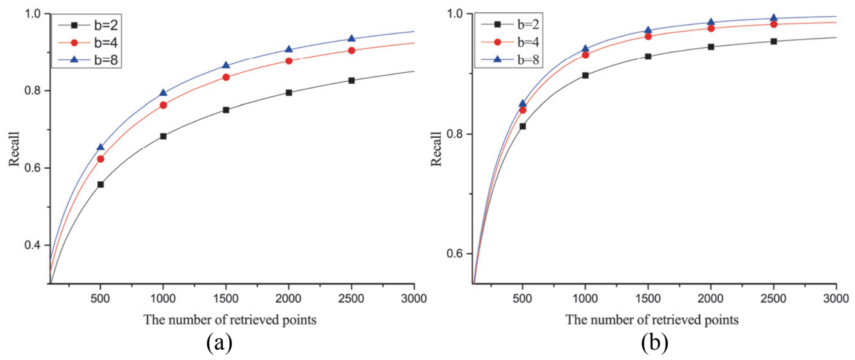


Fig. 11. The recall curves of our method with different b values; The data are mapped into 32-bit binary codes in (a), and the number of binary bits in (b) is 64.

Table 3. The training time with different *b* values.

	b=2	*b*=4	*b*=8
32-bit	1 minutes	2 minutes	752 minutes
64-bit	2 minutes	3 minutes	801 minutes

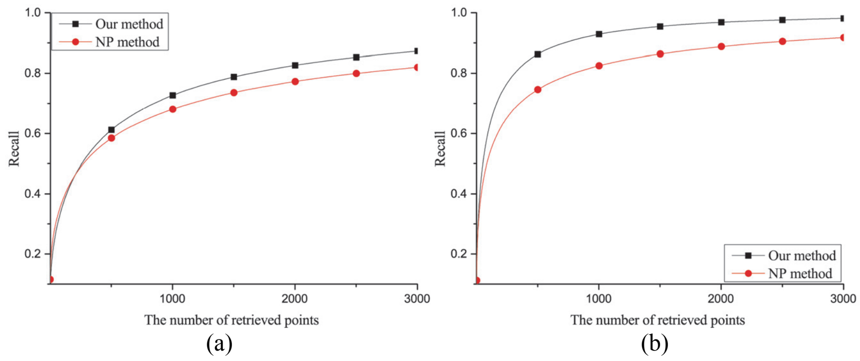


Fig. 12. The recall curves of our method and NP method in *Labelme* dataset. The data are mapped into 32-bit binary codes in (a), and the number of binary bits in (b) is 64.

5.4 Hyper Planes

Hyper planes are learnt to embed in the encoding mechanism, which can greatly reduce the encoding time complexity. In section 3.1, no hyper planes are used to encode data and this mechanism is called NP method in this paper. The ANN search performances of our method and NP method are shown in Fig. 12. As discussed in section 3.2, the hyper planes selected through entropy criterion can improve the distinctiveness of the obtained binary codes. The distribution of the obtained binary codes is uniform, and the number of each kind of binary code is similar. Without the requirement of bucket balance criterion in section 3.2, the number of the data with specified binary codes may be very large, so a lot number of data would be returned as the nearest neighbors during ANN search. Thanks to the entropy selection procedure, the ANN search performances are improved.

5.5 Distance Table

As described in section 4.2, many different Euclidean distances are assigned the same Hamming value. In this paper, the distance table is established to distinguish the data pairs with the same Hamming distance. The nearest neighbors are firstly returned according to their Hamming distances, and then the returned data with the same Hamming distance are re-sorted according to the predefined distance table. The recall curves of our method with and without distance table (DT) are shown in Fig. 13. The experimental results indicate that the utilization of the distance table is valid.

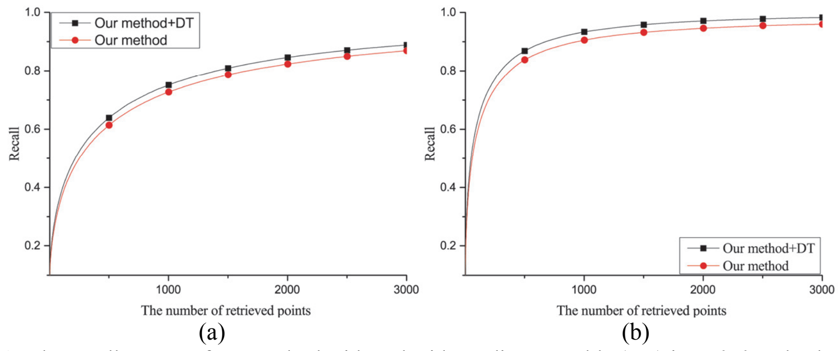


Fig. 13. The recall curves of our method with and without distance table (DT) in *Labelme* database. The data are mapped into 32-bit binary codes in (a), and the number of binary bits in (b) is 64.

6. CONCLUSIONS

In this paper, both the process of generating binary codes and the ANN search procedure are taken into consideration. During the training procedure, the normalized distance similarity restriction and the quantization error are proposed to ensure that the distance relationship in the Hamming space can well approximate that in the Euclidean space. To weaken the inferior influence of the initial centers, different kinds of centers are initialized to generate multiple groups of the hashing functions, and the data with minimal average Hamming distance are returned as the nearest neighbors. As many different Euclidean distances are instead by the same Hamming value, the distance table is established to distinguish the data pairs with the same Hamming distance, and the ANN search performances are boosted through re-sorting the data with the same Hamming distance. Thanks to these effective measures, our method is superior to many state-of-the-arts. In this paper, the length of the binary bits is identical. In the future work, asymmetric hashing will be studied to generate the binary codes with diverse length, and the memory occupancy can be further reduced.

REFERENCES

1. J. Sivic and A. Zisserman, “Video google: a text retrieval approach to object matching in videos,” in *Proceedings of IEEE International Conference on Computer Vision*, 2003, pp. 1470-1477.
2. H. Jegou, M. Douze, C. Schmid, and P. Perez. “Aggregating local descriptors into a compact image representation,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3304-3311.
3. A. Torralba, R. Fergus, and W. T. Freeman, “80 million tiny images: a large data set for nonparametric object and scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, 2008, pp. 1958-1970.
4. F. Perronnin and C. Dance, “Fisher kernels on visual vocabularies for image categorization,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1-8.

5. O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1-8.
6. G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-neighbor methods in learning and vision*: *theory and practice*, The MIT Press, London, England, 2006.
7. A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, Vol. 51, 2008, pp. 117-122.
8. J. He, J. Feng, X. Liu, T. Cheng, T. H. Lin, H. Chung, and S. Chang. "Mobile product search with bag of hash bits and boundary reranking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3005-3012.
9. K. Lin, H. F. Yang, J. H. Hsiao and C. S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 27-35.
10. K. Lin, H. F. Yang, K. H. Liu, J. H. Hsiao, and C. S. Chen, "Rapid clothing retrieval via deep learning of binary codes and hierarchical search," in *Proceedings of ACM International Conference on Multimedia Retrieval*, 2015, pp. 499-502.
11. X. Liu, J. He, C. Deng, and B. Lang, "Collaborative hashing," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2147-2154.
12. W. Liu, J. Wang, Y. Mu, S. Kumar and S. Chang. "Compact hyperplane hashing with bilinear functions," in *Proceedings of International Conference on Machine Learning*, 2012.
13. X. Liu, X. Fan, C. Deng, Z. Li, H. Su, and D. Tao, "Multilinear hyperplane hashing," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5119-5127.
14. Y. Weiss, R. Fergus, and A. Torralba. "Multidimensional spectral hashing," in *Proceedings of European Conference on Computer Vision*, 2012, pp. 340-353.
15. Y. Weiss, A. Torralba, and R. Fergus. "Spectral hashing," in *Proceedings of the Advances in Neural Information Processing Systems*, 2009, pp. 1753-1760.
16. K. Terasawa and Y. Tanaka, "Spherical LSH for approximate nearest neighbor search on unit hypersphere," in *Proceedings of International Workshop on Algorithms and Data Structures*, 2007, pp. 27-38.
17. Y. Gong and S. Lazebnik, "Iterative quantization: a procrustean approach to learning binary codes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 817-824.
18. J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281-297.
19. X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Transactions on Cybernetics*, Vol. 46, 2016, pp. 2252-2264.
20. K. He, F. Wen, and J. Sun, "*K*-means hashing: an affinity-preserving quantization method for learning binary compact codes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2938-2945.
21. H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor

search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, 2011, pp. 117-128.

22. T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, 2014, pp. 744-755.
23. M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proceedings of International Conference on Machine Learning*, 2011, pp. 353-360.
24. J. Wang, S. Kumar, and S. F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3424-3431.
25. G. Lin, C. Shen, Q. Shi, A. V. D. Hengel, and D. Suter, "fast supervised hashing with decision trees for high-dimensional data," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1971-1978.
26. G. Lin, C. Shen, D. Suter, and A. Hengel, "A general two-step approach to learning-based hashing," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2552-2559.
27. X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen, "Sparse hashing for fast multimedia search," *ACM Transactions on Information Systems*, Vol. 31, 2013, p. 9.
28. D. Zhang, J. Wang, D. Cal and J. Lu. "Self-taught hashing for fast similarity search," in *Proceedings of Annual International ACM SIGIR Conference on Research Development in Information Retrieval*, 2010, pp. 18-25.
29. B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proceedings of the Advances in Neural Information Processing Systems*, 2009, pp. 1042-1050.
30. J. Wang, J. Wang, N. Yu, and S. Li, "Order preserving hashing for approximate nearest neighbor search," in *Proceedings of ACM International Conference on Multimedia*, 2013, pp. 133-142.
31. L. Zhang, Y. Zhang, J. Tang, X. Gu, J. Lin, and Q. Tian, "Topology preserving hashing for similarity search" in *Proceedings of ACM International Conference on Multimedia*, 2013, pp. 123-132.
32. B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: a database and web-based tool for image annotation," *International Journal of Computer Vision*, Vol. 77, 2008, pp. 157-73.
33. A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *Proceedings of International Conference on Multimedia*, 2010, 1469-1472.
34. A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical Report, Computer Science Department, University of Toronto, Rep, 2009.
35. W. Liu, J. Wang, S. Kumar and S. F. Chang. "Hashing with graphs," in *Proceedings of International Conference on Machine Learning*, 2011, pp. 1-8.

Hongwei Zhao (趙宏偉) received the B.S. and M.S. degrees in Jilin University of Technology, China, in 1985 and 1993 respectively, and then the Ph.D. degree in Jilin University, China, in 2001. He is currently a Professor and Ph.D. supervisor in School of Computer Science and Technology College, Jilin University. His main research interests include intelligent information system, embedded technology and computer image processing and visualization.

Zhen Wang (王振) received the B.S. degree in Qufu Normal University, China, in 2012. He is currently a Ph.D. candidate in School of Computer Science and Technology College, Jilin University. His main research interests include machine vision and pattern recognition.

Pingping Liu (劉萍萍) received the B.S., M.S. and Ph.D. degrees in Jilin University, China, in 2001, 2004, 2009 respectively. She is currently an Associate Professor in School of Comter Science and Technology College, Jilin University. Her main research interests include machine vision, pattern recognition and embedded system.

Bin Wu (吳彬) received the B.S. degree in Beihua University, China, in 2014. He is currently a Master candidate in School of Computer Science and Technology College, Jilin University. His main research interests include machine vision and pattern recognition.