ONLINE SELF-ORGANIZING HASHING

*Junxuan Chen, Yaoyi Li, Hongtao Lu**

Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China
chenjunxuan@sjtu.edu.cn, dsamuel@sjtu.edu.cn, htlu@sjtu.edu.cn

ABSTRACT

Hashing for similarity search in large scale data has become an increasingly popular technique. K-means Hashing (KMH) has been proven effective because of the benefits of adaptive k-means quantization. However, KMH is a batch-based learning model requiring high time and storage complexities, which makes it hard to load large scale data into memory to train and deal with streaming data. To address this problem, in this paper we propose an online hashing method using Self-Organizing Map (SOM) algorithm, named as Online Self-Organizing Hashing (SOH). Specifically, we map the training data to an affinity preserving hyper-cube with each vertex assigned a binary code using a SOM alike algorithm. After training, a new data point is quantized into a vertex of the hyper-cube and encoded into related binary code. Experimental results demonstrate that SOH has better or comparable retrieval performance to various state-of-the-art hashing methods while simultaneously requiring rather low computational complexity and storage space.

***Index Terms*—** Hashing, Self-Organizing Map, Unsupervised Learning

1. INTRODUCTION

The explosive growth of the vision data on the internet has posed a great challenge to many applications in terms of fast similarity search. To handle this problem, hashing based approximate nearest neighbors (ANN) search has attracted considerable attention because of their improvements in computational speed and storage reduction.

One of the most well-known data-independent hashing methods is Locality-Sensitive Hashing (LSH) [1]. It makes use of simple random hyper-planes for hash functions. Following LSH, many data-dependent hashing methods [2, 3] based on hyper-planes have been proposed. In these methods, several hyper-planes are generated in certain ways and partition the space into a number of non-overlapping regions in the first stage. Then in the second stage, the data points in each region will be quantized into same binary codes.

*Corresponding author

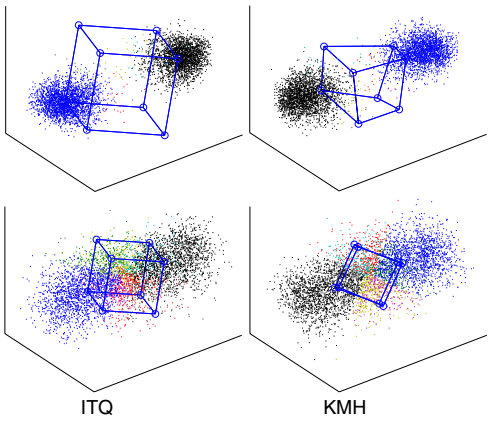

Fig. 1. A geometric view of the hyper-cubes of ITQ and KMH on two synthetic datasets. A circle denotes a codeword, and a line linking two circles means their Hamming distance is 1. Data points with same indices share a same color. This figure is best viewed in color version.

At the same time, hashing methods like Spherical Hashing [4] and K-means Hashing (KMH) [5] which do not depend on hyper-planes have been proposed. These methods provide stronger power in defining a tighter closed region in the data space [4] than those using hyper-planes and achieve a promising performance. In KMH, a geometric view is proposed that any orthogonal hyper-planes hashing method (*e.g.*, ITQ and PCAH [2]) can be considered as a vector quantization using the vertexes of a rotated hyper-cube as the codewords. KMH minimizes the sum of *quantization error* and *affinity error* by an Expectation-Maximization (EM) alike algorithm. Geometrically, it allows to *stretch* the hyper-cube while rotating it (Figure 1). Enjoying the benefits of adaptive k-means quantization and affinity preserving, KMH outperforms various hyper-planes based hashing methods [5].

Although these hashing methods obtain good performance, however, two critical problems have been mentioned [6]. First, for truly large scale datasets, data is usually s-

tored on a distributed disk group and is too large to be read into memory. For example, KMH uses PCAH to initialize the hyper-cube which needs an infeasible time complexity of $O(nd^2 + d^3)$ when n and d are too large. Second, data points come continuously in streaming fashion in many real-word applications, while KMH has to accumulate all the data and re-train a new model.

To overcome these problems, two online hashing methods have been developed named Online Hashing (OKH) [6] and Online Sketching Hashing (OSH) [7]. OKH adapts the hash function accommodate to new pair of data along the line of "Passive-Aggressive" method. OSH maintains a sketch of data that preserves the property of interest but with a smaller size and learns the hash functions based on the sketch. These methods both need one pass over data to train and allow a dramatic reduction in computation and storage. However, they are both hyper-planes based online hashing methods. As far as we know, there is no online hashing which does not depend on hyper-planes.

In this paper we focus on learning binary codes through an online affinity preserving quantization which does not depend on hyper-planes. A naive solution to the issue is applying Stochastic Gradient Descent (SGD) to optimize the objective function of KMH. However, we find the method does not work well. This motivates us to propose a novel Self-Organizing Hashing (SOH) based on Self-Organizing Map [8] algorithm to optimize the joint of quantization and affinity error. Our method has two advantages:

- We suppose that train samples are given one at a time. When a new sample comes we update our model immediately. It is particularly suitable for large scale applications, where the number of data points and the problem dimensionality are both extremely large.
- The topological order can be kept during iterations so it is naturally suitable for constructing a hyper-cube in feature space. Besides the superiority over hyper-planes based methods, our approach empirically possesses lower quantization error and affinity error simultaneously comparing with applying SOM algorithm or SGD directly.

To the best of our knowledge, this is the first work that applies SOM in hashing methods. Experimental results show that our method has achieved promising performances on two large datasets.

2. RELATED WORK

2.1. K-means Hashing

K-means Hashing is a novel affinity-preserving hashing based on k-means algorithm. Given the dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^T \in R^{n\times d}$, KMH first constructs a set of vectors $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k]^T \in R^{k\times d}$. The set $\mathbf{W}$ is a codebook, $\mathbf{w}_i$ is a codeword, and each codeword has a binary indexing $\mathbf{y}_i \in \{0, 1\}^b$ with b denoting the code size and $k = 2^b$. Then KMH maps each data point $\mathbf{x}$ to a codeword $\mathbf{w}_{i(\mathbf{x})}$ and minimizes the objective function:

$$E = E_{\text{quan}} + \rho E_{\text{aff}}. \tag{1}$$

Where E_{quan} is the average *quantization error* of the classical k-means algorithm:

$$E_{\text{quan}} = \frac{1}{n}\sum_{\mathbf{x}\in\mathbf{X}} ||\mathbf{x} - \mathbf{w}_{i(\mathbf{x})}||^2, \tag{2}$$

and E_{aff} is the *affinity error* due to the distance approximation by binary code:

$$E_{\text{aff}} = \sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}(d(\mathbf{w}_i, \mathbf{w}_j) - d_h(i, j))^2. \tag{3}$$

Here $w_{ij} = n_i n_j / n^2$. n_i and n_j are the number of samples having index i and j respectively. d_h is a Hamming-based distance between two binary indices $\mathbf{y}_i$ and $\mathbf{y}_j$:

$$d_h(i, j) \triangleq s \cdot h^{\frac{1}{2}}(\mathbf{y}_i, \mathbf{y}_j), \tag{4}$$

where s is a scale constant and initialized by PCAH, h denotes the Hamming distance. KMH minimizes this function in an EM fashion like k-means.

Algorithm 1 Self-Organizing Map algorithm

repeat

1.At each time t, present an input $\mathbf{x}(t)$, and select the winner,

$$v(t) = \arg\min_{1\le i\le k} ||\mathbf{x}(t) - \mathbf{w}_i(t)|| \tag{5}$$

2.Update the weights of the winner and its neighbours,

$$\Delta\mathbf{w}_i(t) = \alpha(t)\eta(v, i, t)\big(\mathbf{x}(t) - \mathbf{w}_i(t)\big) \tag{6}$$

until converges

2.2. Self-Organizing Map

Self-Organizing Map (SOM) [8] is a powerful unsupervised neural network for high-dimensional data quantization and topology preservation.

The SOM uses a set of neurons, often arranged in a 2-D rectangular grid, to form a discrete topological mapping of the d dimensional input space $\mathbf{X}$. Here we assume that the grid dimension is b and there are k neurons in the gird. So we have a set of location vector $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_k]^T \in R^{k\times b}$ in the grid. At the start of the learning, all the weights $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k]^T \in R^{k\times d}$ are initialized to small random numbers, where $\mathbf{w}_i$ is the weight vector associated to

neuron i and is a vector of the same dimension d of the input $\mathbf{x}$. Then the algorithm repeats the steps shown in Algorithm 1, where $\eta(v, i, t)$ is the neighbourhood function. Although one can use any neighbourhood function, a Gaussian form is often used in practice:

$$\eta(v, i, t) = exp\Big(- \frac{Dist(\mathbf{r}_v, \mathbf{r}_i)}{2\sigma(t)^2}\Big), \tag{7}$$

$$\sigma(t) = \sigma_0 exp(-\frac{t}{\lambda}), \tag{8}$$

The learning rate, $\alpha(t)$ in Eq.(6), is also an exponential decay function and ensures that the SOM will converge:

$$\alpha(t) = \alpha_0 exp(-\frac{t}{\lambda}), \tag{9}$$

It has been shown that while SOMs with a small number of nodes behave like k-means, larger SOMs rearrange data in a way that is fundamentally topological in character. Hence, our SOM based hashing is expected to preserve affinity better than using SGD to minimize Eq.(1).

3. THE PROPOSED APPROACH

3.1. Online Self-Organizing Hashing

Motivated by SOM, our SOH is to quantize the feature space in an *online self-organizing* fashion rather than an *off-line EM* fashion. Instead of using traditional 2-D or 1-D map, we construct b dimensional hyper-cube in the grid space, that is, a hyper-cube of $k = 2^b$ points for indexing so that each point has a b-bit unique binary code $\mathbf{y}_i$. We minimize the following affinity-preserving sample function at each step:

$$E(t) = \sum_{i=1}^{k} \eta(v, i, t)\big(||\mathbf{x}(t) - \mathbf{w}_i||^2 + \beta E_{\text{aff}}(\mathbf{w}_i)\big) \tag{10}$$

Here β is a fixed weight and we denoting

$$E_{\text{aff}}(\mathbf{w}_i) = \sum_{j=1}^{k} (d(\mathbf{w}_i, \mathbf{w}_j) - d_h(i, j))^2 \tag{11}$$

This leads to a new update formula at step 2 in Algorithm 1:

$$\begin{aligned}\Delta\mathbf{w}_i(t) =& \alpha(t)\eta(v, i, t)\big(\mathbf{x}(t) \\ & - \mathbf{w}_i(t) - \beta\frac{\partial E_{\text{aff}}(\mathbf{w}_i)}{\partial \mathbf{w}_i}\big)\end{aligned} \tag{12}$$

We can derive from Eq.(11) that:

$$\frac{\partial E_{\text{aff}}}{\partial \mathbf{w}_i} = \sum_{j=1}^{k} 4\Big(1 - \frac{d_h(i, j)}{d(\mathbf{w}_i, \mathbf{w}_j)}\Big)(\mathbf{w}_i - \mathbf{w}_j) \tag{13}$$

In each iteration we adjust the index points according to Eq.(12) and Eq.(13). We compute the distance of two index points in Eq.(7) in grid using the square of Hamming distance of their binary indices:

$$Dist(\mathbf{r}_v, \mathbf{r}_i) = h(\mathbf{y}_v, \mathbf{y}_i)^2, \tag{14}$$

After learning, we assign each sample $\mathbf{x} \in \mathbf{X}$ a binary code as the binary indices of the nearest index point.

Algorithm 2 Online Self-Organizing Hashing

Initialize the index points hyper-cube and s.
for t = 1,2,... **do**
1.Update s according to Eq.(15).
2.Present an input $\mathbf{x}(t)$, and select the winner according to Eq.(5).
3.Update the weights of the winner and its neighbours according to Eq.(12) and Eq.(13).
end for

3.2. Initialization and choosing s

KMH initializes the indices using the binary codes learned by PCAH, while our SOH can get equivalent performance when we initialize the hyper-cube arbitrarily because SOM can follow the distribution of the data. We initialize a regular hyper-cube with each edge parallel to the axis (Figure 2).

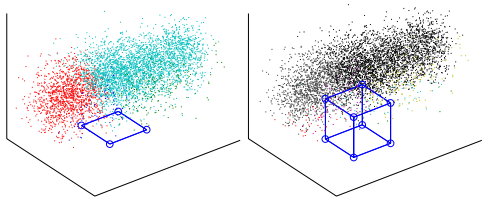

Fig. 2. A geometric view of the initialization of SOH when feature space is 3-D, and grid space are 2-D and 3-D. Data points are synthetic and not processed with centralization.

We randomly initialize the scale parameter s in $d_h(i, j)$ of Eq.(4). At each iteration, before we update index points, we need to update s. Since Eq.(11) is a quadratic function of s ,we have a closed form solution $\hat{s}$:

$$\hat{s} = \frac{\sum_i \sum_j d(\mathbf{w}_i, \mathbf{w}_j) h^{\frac{1}{2}}(i, j)}{\sum_i \sum_j h(i, j)} \tag{15}$$

The procedure of Online Self-Organizing Hashing is summarized in Algorithm 2.

3.3. Generalization to a Product Space

Like KMH, our method can be generalized to a product space in a similar way when bit number b is large. We can apply random rotations to decompose the product space and

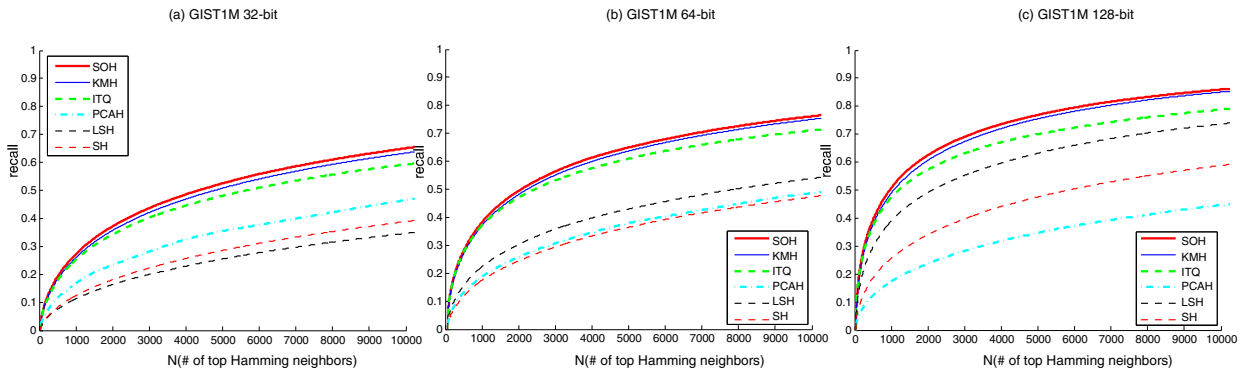

Fig. 3. (a)(b)(c)Recall curves on GIST1M with 32,64,128 bits. Our method and KMH uses b=4 in the 32/64-bit case, and b=8 in the 128-bit case. In this figure, K=10 Euclidean nearest neighbors are considered as the ground truth. This figure is best viewed in color version.

get multi-table lookup as in LSH. However the retrieval performance of single hash table can not be guaranteed. To get best single hash table, we can decompose the product space using the state-of-the-art technique Eigenvalue Allocation [9]. If we use Eigenvalue Allocation we need perform PCA on the dataset. We can use data sketching method [7] to perform PCA in an online fashion. In practice we find that performing online PCA on the whole dataset is unnecessary. Instead, a small subset of data is enough to generate a satisfactory decomposition before we perform our SOH.

4. EXPERIMENTAL RESULTS

In this section, we first compare our method with applying SGD directly to optimize Eq.(1) (we call this method KMH-SGD) to manifest the good property of SOH. Then we compare with state-of-the-art hashing methods. We conduct experiments under batch setting and online setting to verify the effectiveness and efficiency. We decompose the space to M subspaces, and each sub-codebook has b bits. These M sub-indices are concatenated into a binary code with $M \cdot b$ bits. We control the parameter λ to ensure the algorithm converges until each sample is presented once. We choose $\beta = 0.5$ when $b = 2$, $\beta = 0.1$ when $b = 4$ and $\beta = 0.01$ when $b = 8$. Other parameters are given as $\sigma_0 = 1$ and $\alpha_0 = 0.05$.

4.1. Datasets and Evaluation Protocols

We evaluate the retrieval performances on two public datasets: SIFT1M[1] and GIST1M[2]. SIFT1M contains 1 million 128-d SIFT features [10] and 10,000 queries. GIST1M contains 1 million 384-d GIST features [11] and 10,000 independent queries randomly sampled. For both datasets, We randomly select 100,000 data points to train under batch setting and use all data points to train under online setting. We evaluate the performance by using each query's K Euclidean neighbors in the full set of data as ground truth. In most experiments we set $K = 10$ and we also evaluate the performance w.r.t a wide range of K. Under batch setting, we adopt *Hamming ranking* to report the recall at the first N Hamming neighbors. Under online setting the retrieval performance is evaluated with mean average precision (MAP). Our experiments are run on an Intel i5-4430 @ 3.0GHz CPU with 16GB RAM.

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

[2]http://groups.csail.mit.edu/vision/TinyImages/

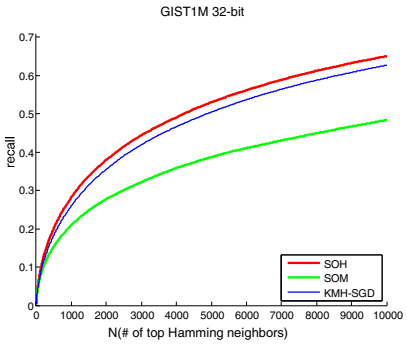

Fig. 4. Recall curves with three online methods on GIST1M. In this figure, K=10 Euclidean nearest neighbors are considered as the ground truth. Our method use b=4.

4.2. Results and Discussions

We first compare our method with KMH-SGD and basic SOM algorithm without affinity preserving (Algorithm 1). Figure 4 shows the comparison between these methods on SIFT1M with 64 bits (M = 16 subspaces). We can see that the basic SOM algorithm performs poorly. The KMH-SGD is better than SOM but still inferior to SOH. To understand why these two methods is inferior, we generate 5000 synthetic da-

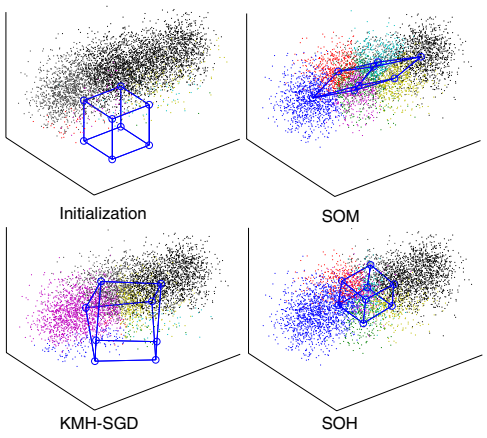

Fig. 5. A geometric view of the hyper-cubes of SOM, KMH-SGD and SOH when feature space and grid are both 3-D. Data points are synthetic and not processed with centralization. This figure is best viewed in color version.

ta points with non-zero mean and visualize the hyper-cube of SOM, KMH-SGD and SOH when b=3 in Figure 5. We can observe that the hyper-cube of SOM embedded into data well but collapsed when the distribution of data is not uniform. In this case, two neurons which have small Hamming distance in grid space may have large Euclidean distance in feature space, which leads to large *affinity error*. Different form SOM, KMH-SGD has considered affinity-preserving, however, it is easier to fall into local optimum because quite a few indices will not be updated if no data point is closest to them. This situation becomes more serious when data has non-zero mean. From Figure 5, we can find in this method, hyper-cube is not embedded into data properly which leads to large *quantization error*. SOH has the good property of SOM and also considers affinity-preserving, which makes it better to fit the data and has low affinity error.

Next, we compare our method with other state-of-the-art batch based hashing methods including LSH [1], Spectral Hashing (SH) [12], KMH [5], PCAH and ITQ [2]. We vary hash bits from 32 to 128, and K from 1 to 100 to see the performances of all methods. We present the recall curves when $K = 10$ in Figure 3. It is clear that our SOH outperforms other methods on each bit, proving its efficiency and stability. In Figure 6, we show the evaluation in a wide range of K (1 to 100) comparing with KMH and ITQ. It can be seen again, our method is consistently superior.

Third, for other state-of-the-art online methods, we choose OKH [6], OSH [7] and LSH. For both datasets, the training data is evenly divided into 100 chunks. We evaluate the MAP scores after selected rounds. For SOH, we use 1 round (10000 points) to perform online PCA based on data sketching to decompose the space in advance. Figure 7 shows the comparison between these methods on SIFT1M with 16,

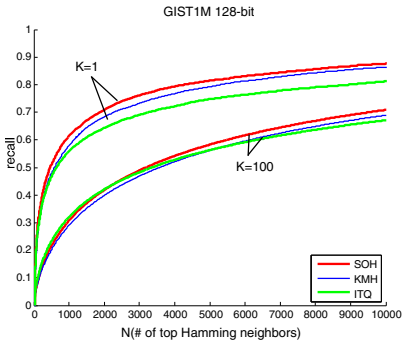

Fig. 6. Comparison in GIST1M with 128 bits under different K(1, 100).

32, 64 bits codes. We can observe that SOH decisively outperforms other methods. In addition, the performance of OSH is not stable because it uses a random rotation to the PCA projection. By contrast, SOH achieves a stably increasing accuracy in most experiments.

Finally we present the average training time of 1 round of samples (10000 points) on SIFT1M. Although KMH performs closely to our method, it requires much more time due to the heavy iteration procedure over all data points. The comparison results with different code lengths and subspace bits are shown in Table 1. It is clear our SOH method is about 5x faster than KMH.

Table 1. Training time comparison between SOH and KMH with various code length on SIFT1M (10000 points).

CODE LENGTH	SUBSPACE BITS	KMH(S)	SOH(S)
32	2	20.04	5.94
32	4	76.10	7.07
64	4	74.46	14.42
64	8	944.16	152.71
128	8	989.27	241.21

5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a novel online hashing method named Online Self-Organizing Hashing. We use classical SOM algorithm to optimize the joint of quantization error and affinity error. Experimental results have demonstrated our method is better or comparable to various state-of-the-art methods. Recently, SOMs with multi-layers structures [13, 14] have been developed for image retrieval to overcome some limitations of traditional SOM. How to integrate these works into our method seems an interesting future work.

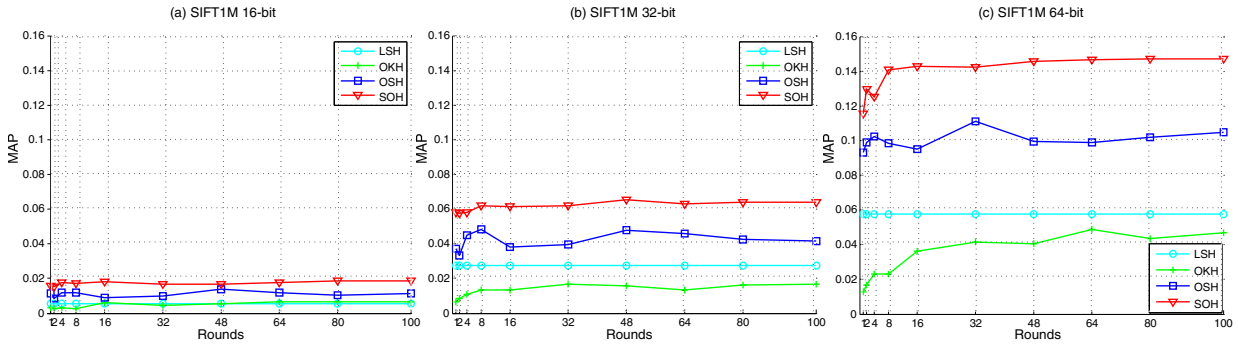

Fig. 7. (a)(b)(c) MAP on SIFT1M at selected rounds with 16, 32, 64 bits. Our method uses b=2 in the 16/32-bit case, and b=4 in the 64-bit case. This figure is best viewed in color version.

6. ACKNOWLEDGEMENT

This paper is supported by NSFC (No. 61272247, 61533012, 61472075), the 863 National High Technology Research and Development Program of China (SS2015AA020501) and the Major Basic Research Program of Shanghai Science and Technology Committee (15JC1400103).

7. REFERENCES

[1] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *VLDB*, 1999, vol. 99, pp. 518–529.

[2] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 12, pp. 2916–2929, 2013.

[3] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Semi-supervised hashing for large-scale search," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 12, pp. 2393–2406, 2012.

[4] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon, "Spherical hashing," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2957–2964.

[5] Kaiming He, Fang Wen, and Jian Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2938–2945.

[6] Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng, "Online hashing," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 1422–1428.

[7] Cong Leng, Jiaxiang Wu, Jian Cheng, Xiao Bai, and Hanqing Lu, "Online sketching hashing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2503–2511.

[8] Teuvo Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.

[9] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun, "Optimized product quantization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 4, pp. 744–755, 2014.

[10] David G Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] Aude Oliva and Antonio Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[12] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2009, pp. 1753–1760.

[13] Andreas Rauber, Dieter Merkl, and Michael Dittenbach, "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data," *Neural Networks, IEEE Transactions on*, vol. 13, no. 6, pp. 1331–1341, 2002.

[14] Tommy WS Chow, MKM Rahman, and Sitao Wu, "Content-based image retrieval by using tree-structured features and multi-layer self-organizing map," *Pattern Analysis and Applications*, vol. 9, no. 1, pp. 1–20, 2006.