British Journal of Mathematics & Computer Science
18(1): 1-11, 2016, Article no.BJMCS.27865
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

A Rapid Quantization-based Image Retrieval Algorithm

Tingjia Shan[1], Qiang Ling[1*], Kaikai Song[1], Binbin Du[1], Feng Li[1] and Song Wang[1]

[1] *Department of Automation, University of Science and Technology of China, Hefei 230027, China.*

Authors' contributions:

This work was carried out in collaboration by all authors. Author TS designed the study, implemented the algorithm, and wrote the first draft of the manuscript. Author QL proposed the concerned problem, and finished the final manuscript. Authors KS and BD optimized the method and guided the experiments. Authors FL and SW analyzed the experimental results. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/27865
Editor(s):
(1) Kai-Long Hsiao, Taiwan Shoufu University, Taiwan.
Reviewers:
(1) Bikesh Kumar Singh, National Institute of Technology, Raipur, India.
(2) Noorhaniza Wahid, Information Technology of Tun Hussein Onn University of Malaysia, Malaysia.
(3) Anonymous, University of Florence, Italy.
Complete Peer review History: http://www.sciencedomain.org/review-history/15658

Original Research Article

Received: 23rd June 2016
Accepted: 30th July 2016
Published: 4th August 2016

Abstract

Fast image retrieval has been a fundamental problem in the area of image processing for a long time. This paper proposes a rapid image retrieval algorithm by improving the conventional nearest neighbor search through the implementation of vector product quantization and inverted indexing structure. Vector product quantization can efficiently accomplish the fast nearest neighbor search task, and has many great advantages in terms of storage requirements, retrieval speed and accuracy. In order to further reduce the search time, an approximate threshold-based distance estimation technique is introduced into the retrieval algorithm. Moreover, the quick sort method is implemented to reorder the image search results, which can significantly improve the performance of our retrieval algorithm.

**Corresponding author: E-mail: qling@ustc.edu.cn*

Keywords: Product quantization; nearest neighbor search; image retrieval; inverted indexing structure.

2010 Mathematics Subject Classification: 06A11, 06B35, 54C35, 54D45.

1 Introduction

Fast image retrieval is a basic task in image processing. One way to realize fast image retrieval is the approximate nearest neighbor(ANN) search, which plays a critical role in image retrieval, image classification [1] and object recognition [2]. There are many classical algorithms to solve the nearest neighbor (NN) search problem, like K-D Tree [3][4][5], spectral hashing [6], Locality Sensitive Hashing (LSH) [7], Fast Library for Approximate Nearest Neighbors (FLANN) [8][9], Product Quantization (PQ) [10][11] and so on [12][13][14][15][16]. Among these algorithms, PQ stands out for its faster computational speed, less storage requirement and higher retrieval accuracy especially in the large scale database.

By combining the inverted indexing structure, PQ is a very efficient method for ANN search. It consists of two major steps,

1. building inverted indexing structure [17];
2. encoding vectors into compact codes [10][18].

The inverted indexing structure is a set of lists, each of which contains encoded vectors assigned to the relative cluster centers, named *coarse centroids*. These cluster centers represent the distribution of database. When w nearest results are expected for a query, we first find some cluster centers which are near the query and then traversal those lists associated with these centers. In the above second step, data are quantized into codewords using Product Quantizer [19]. Product Quantizer divides the original vector to M sub-vectors and quantize each sub-vectors to K codewords which are learned from training data. The original vector is represented by a cascade of codewords and referred to as *compact vector*. We can utilize the the distance between the query and the *compact vector* to approximate the Euclidean distance between the query and vectors in database. By combining the compact encoding method with inverted index structure, both a rapid speed and high-quality search results can be achieved in millions of vectors, which is exactly the main task of the present paper.

1.1 Related works

In the literature, there are some optimized PQ methods to solve the ANN search problem in image search. In [12], an improved method was proposed in computing cluster centers, and its experiments demonstrated that the obtained search accuracy is a bit higher than the original PQ method [10], at the cost of possible longer search time. In [14], the multi inverted index [20] method was implemented to reduce the search time, but the parameters of that method need be set manually with different scale databases. Recently Zhou *et. al.* [21] proposed a new quantization method and used k-d tree and to build the inverted index structure. Specially bit manipulation was implemented to accelerate the search speed in [21]. But the method in [21] is poor with high dimension features extracted by some Convolutional Neural Network(CNN), like GoogleNet [22]. In summary, the above mentioned methods do not consider that the inverted indexing structure may not always perform well for different databases. Experiments show that their performance strongly depends on the concerned databases. Moreover, their parameters have to be tuned manually based on the scale of database, which prevents their applications in reality.

In order to efficiently resolve the aforementioned issues, this paper proposes a new retrieval algorithm based on the inverted indexing structure and Product Quantization. According to the triangle

inequality, we use the approximate distance [10] and the distance between the original vector and its responding *compact vector* to estimate the upper threshold of the real distance. When the distance between the query and *coarse centroids* are by far more than that upper threshold, we will stop traversing the rest lists. Compared with the methods based on the original inverted index structure, our method needs to visit less *coarse centroids*. The original inverted index structure requires to visit a fixed number of *coarse centroids*. That number has to be manually set according to the scale of the concerned database, which is not easy. Our method only needs to visit much less number of *coarse centroids*, especially in databases whose distributions are asymmetric. Our method is robust against the data distribution and can greatly reduce the computational time. To further improve performance, our algorithm implements the divide-and-conquer technique to find the k nearest results and reorder results by the quickly sort method [23].

The rest of this paper is organized as follows. Section 2 introduces the basic knowledge of the PQ method. In Section 3, we present our retrieval algorithm. Experimental results confirm the effectiveness and efficiency of our algorithm in Section 4. Finally, some concluding remarks are placed in Section 5.

2 Preliminaries

This section first introduces vector quantization and then describes the vector product quantization.

2.1 Vector quantization

For a D-dimensional vector $x \in R^D$, quantization maps a vector x to a D-dimensional vector $f_q(x)$ which is a cluster centroid got by K-Means [24] or other method [25]. A cluster centroid set C of size K and $f_q(x)$ are related as

$$f_q(x) \in C = \{c_i \mid i \in I\}, \tag{2.1}$$

where the centroid set C is referred to as the codebook, c_i a codeword, and I is a finite index set, i. e. , $I = \{1, 2, ..., K\}$. The quantizer is represented by the function $f_q(\cdot)$. For vector quantization, the quantization distortion is an objective function and will be minimized. To reach this goal, many methods were proposed [14][12] Due to this minimization criterion, a vector should be quantized to its nearest codeword. So the function $f_q(\cdot)$ takes the following form [10],

$$f_q(x) = \text{argmin}_{c_i \in C} \ d(x, c_i), \tag{2.2}$$

where $d(x, c_i)$ stands for the distance between x and c_i.

2.2 Product quantization

Product quantization is an effective way to represent a high-dimensional vector codewords with smaller storage space. Take a D-dimensional vector x as an example. We split a D-dimensional vector x to M sub-vectors,

$$x = [x_1, x_2, \cdots, x_M], \tag{2.3}$$

where each sub-vector x_i has the same dimension $D^* = D/M$. Then using M distinct quantizers, $f_q^1(x), \cdots, f_q^m(x), \cdots, f_q^M(x)$, to quantize the sub-vectors. The the quantizer $f_q^m(x)$ is associated with the m-th subvector x_m. There would be M relative sub-codebooks $C_1, C_2, \cdots, C_M$. For the sake of simplicity, we assume that each sub-codebook has the same size k. According to the quantizer in equation (2.2) we know

$$\begin{aligned} c &= f_q(x) = \left[\ f_q^1(x_1) \quad f_q^2(x_2) \quad \cdots \quad f_q^M(x_M) \ \right] \\ c &\in C_1 \times C_2 \times ... \times C_M \end{aligned} \tag{2.4}$$

where $\times$ stands for the Cartesian product. Given the sub-codebook size k and the sub-vector number M, from equation 2.4, we can easily know that their Cartesian products can generate k^M codewords and need to store $k * M$ subvectors. So it has obvious advantages over the results by the conventional k-means. For the ANN search, the PQ method is implemented by approximating the distance $d(y, x)$ with the distance $d(y, f_q(x))$,

$$d(y, x) \approx d(y, f_q(x)) \tag{2.5}$$

The approximate distance can be computed rapidly using lookup tables.

3 Main Results

This section will introduce our retrieval algorithm, which consists of the invert indexing structure and a searching method based on that structure. In order to improve the robustness and reduce the computational complexity further, we introduce a threshold when we approximate the distance between the query and the vectors in database, which is tested to work well n some typical SIFT databases [10][12][14]. In the following, the method of distance threshold estimation is introduced firstly and our retrieval algorithm is presented in details afterwards.

3.1 Distance threshold estimation

By (2.5), we can get

$$d(y, x) \leq d(y, f_q(x)) + d(x, f_q(x)) \tag{3.1}$$

We define a variable $T_H = d(y, f_q(x)) + d(x, f_q(x))$ as an upper threshold. By taking the query y to compute the approximate distance with different vector x, T_H is updated. After a finite number of iterations, it will reach a minimum value. In searching the K nearest neighbors in the inverted indexing structure, if the distance between the query and the centroid is much larger than T_H, we will believe that the vector assigned in this centroid is far from the query and stop searching. More specifically, when $d(y, f_q(x)) > \lambda T_H$, we give up traversing the vectors assigned in $f_q(x)$; otherwise, we will traverse the vectors. Here $\lambda \geq 1$ is a constant.

Fig. 1 illustrates the above algorithm. Suppose a vector x has the nearest distance with the query y. Then the distance estimation threshold $T_H = d(y, f_q(x)) + d(x, f_q(x))$. When y finishes traveling the vectors assigned in $q(x)$, it will compare λT_H and $q(y, q(z))$. If $q(y, q(z))$ is bigger than λT_H, we believe $q(z)$ has a long distance to the query y and will not travel the vectors assigned in $q(z)$.

3.2 Inverted indexing structure

Inverted indexing structure is an array of lists $\{L_1, L_2, \cdots, L_K\}$. Each list is associated with one centroid among $\{c_1, c_2, \cdots, c_k\}$. Fig. 2 shows the sample structure. For each vector $x \in L_K$, it satisfies the following condition,

$$c_k = f_q(x), \;\; x \in L_K \tag{3.2}$$

To get a better codebook $c \in C_1 \times C_2 \times ... \times C_M$, we pre-process the sample database by subtracting each vector's quantizer $f_q(x)$. Constructing inverted indexing structure is composed of the following steps.

1. Train codebook C with the sample database.
2. For every vector x in the sample database, quantize x to $f_q(x)$ and compute the residual $r(x) = x - f_q(x)$.

3. Train C_i in every m-th subvector using the residual vector $r(x)$.
4. For each vector x to be quantified, firstly compute its $r(x)$ with the codebook C above.
5. Split $r(x)$ into subvectors and quantize each subvector. Record the corresponding codewords index number and assign these indices to the centroid $f_q(x)$

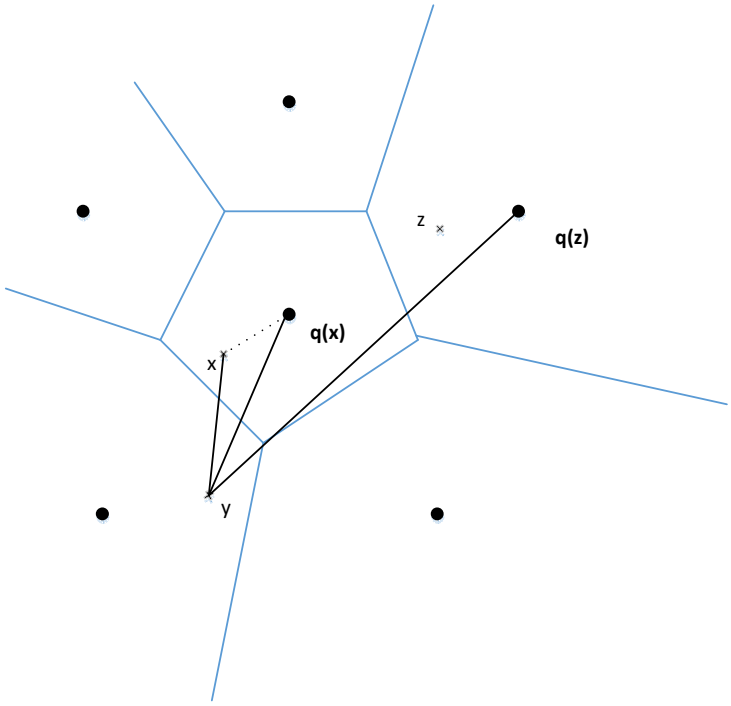

Fig. 1. Distance estimation

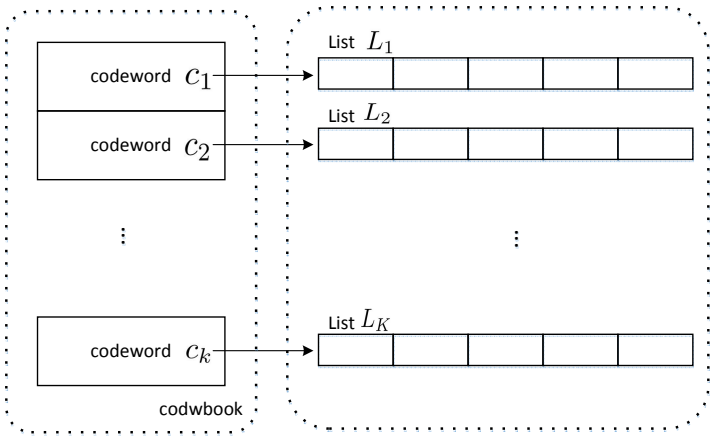

Fig. 2. Inverted indexing structure

3.3 Search the nearest neighbor

Given a query y, its k nearest neighbors search follows the following procedure.

1. Compute the distance between the query and each codeword in the codebook C which is used to assign vector. Sorting the distance to find the w nearest neighbors $c_{y1}, c_{y2}, \cdots, c_{yw}$ and save the corresponding distance $d_{y1}, d_{y2}, \cdots, d_{yw}$.

2. Compute the approximate distance $d(y, [f_q^1(x_1); f_q^2(x_2); \cdots; f_q^M(x_M)])$, where x_i is the vector assigned to the codebook c_{yi}. Then update the upper threshold T_H with the minimal value if the computed number is smaller than a finite number. Otherwise, the updating is stopped.

3. If T_H is still updated and the program has not finished traveling all w neighbor vectors, we will compute the distance between the query and vectors in the next codebook in order. Otherwise we will compare λT_H and d_{yw}. If d_{yw} is larger than λT_H, we will stop traveling the rest vectors.

4. Reorder the result with the Quick Sort method.

4 Experimental Results

We evaluate the performance of our algorithm on the 1M and 10MSIFT descriptors [26] of [10]. All the experiments have been conducted on a server with Intel(R) Xeon(R) CPU E5620 @2.4GHZ and 32G RAM. We use the search accuracy under different recallR@ numbers to evaluate the search quality as [10]. As shown in [10], with given M and k, we can generate k^M codewords. Obviously the search result's accuracy is better if the value of M is big. However with an increased M, the computational complexity also increases and the search time is longer. In the inverted indexing structure, the number of lists also affects λ. In the following we will show the impact of concerned parameters on the performance of our algorithm.

As mentioned in Section 3, our algorithm is an improvement of the algorithm in [10]. Specifically, the concerned algorithm in [10] takes the following parameters as a baseline for performance comparison,

- $M = 16$.
- 128 coarse centroids are traversed for each query.
- The number of centroids and the nearest neighbors' number used in the inverted indexing structure are set as 8192 and 128, respectively.
- The sub-vector's codebook size is set as 256. The size of query set is 100.

In our algorithm, $\lambda = 1.5$ and the number of iterations N ranges from 0 to 30000 for the 1M sift database and 0 to 300000 for the 10M sift database. We use the average visited coarse centroids to evaluate our algorithm's efficiency and refer to it as AVCC. If AVCC is smaller than 128, we only need traverse a part of the 128 coarse centroids, which means shorter search time than [10]. The experimental AVCC results under different N are shown in Fig. 3. The corresponding accuracy is showed in Fig. 4.

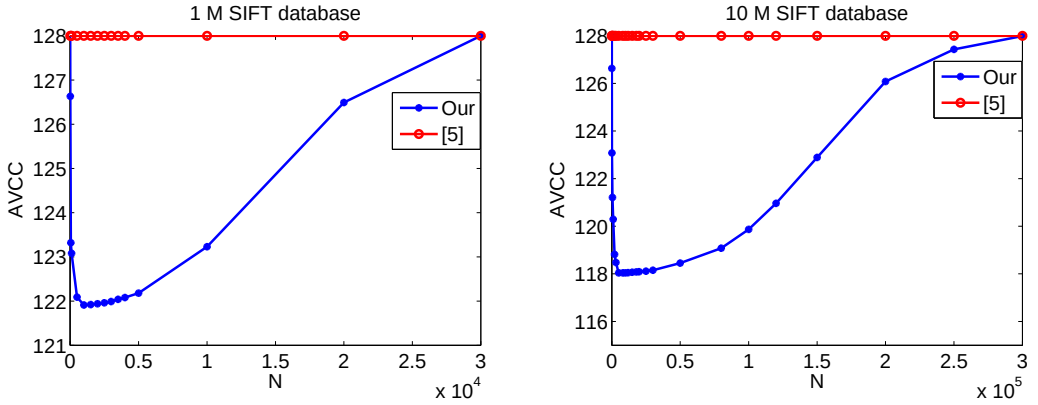

Fig. 3. The average visited coarse centroids (AVCC) in 1M and 10M SIFT databases

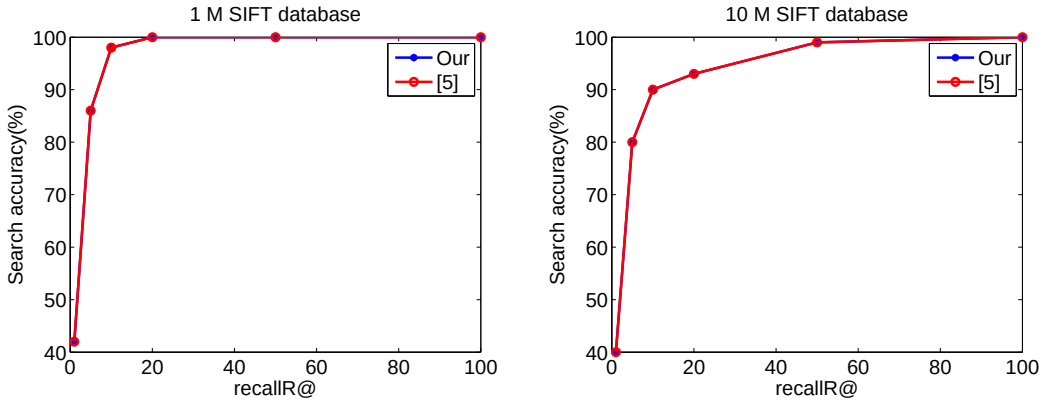

Fig. 4. Search accuracy (the same search accuracy is achieved by different N)

Under different N, the search accuracy keeps invariant, which is shown in Fig. 4. The reason lies in that PQ can guarantee the accuracy in large scale database with traversing parts of the database as proved in [10]. In our experiments, we choose a large λ. So N do not affect the accuracy. Compared with the algorithm in [10], our algorithm does not sacrifice any search accuracy for the improvement of computational efficiency as shown in Fig. 3.

We see that the AVCC of our algorithm is smaller than that of [10] by Fig. 3. As the computational complexity is roughly proportional to the AVCC, our algorithm owns a higher efficiency than [10]. We can measure the computational efficiency advantage of our algorithm as

$$\eta = 1 - \frac{AVCC_{our}}{AVCC_{[5]}},$$

where $AVCC_{our}$ and $AVCC_{[5]}$ represent the AVCCs of our algorithm and [10], respectively. η under different N is shown in Fig. 5.

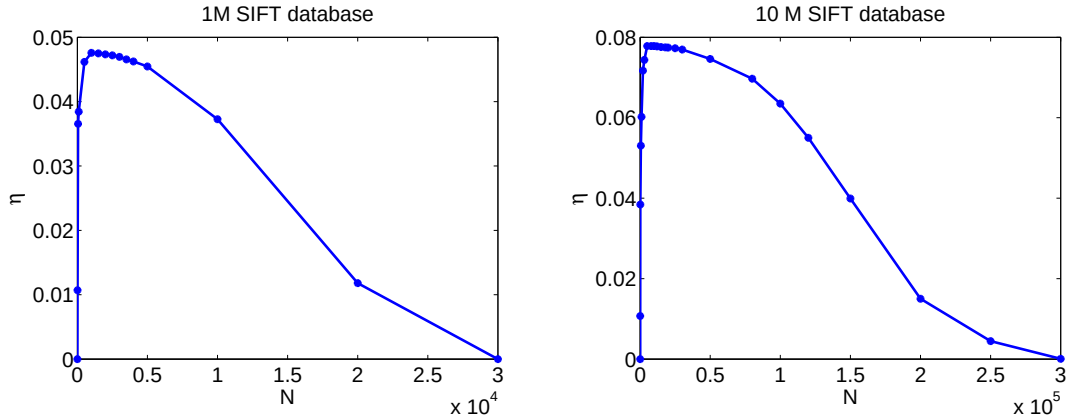

Fig. 5. The computational efficiency advantage of our algorithm

From Fig. 3 and 5, we can see that when N is too small, T_H may not reach the minimum value. But when N is too large, we may traverse lots of lists in Inverted Index structure and AVCC is large too. Our algorithm shows the best performance when N is about 1% of the size of database. Note that such performance improvement, 5% in the 1M database and 8% in the 10M database, is achieved without any search accuracy loss as shown in Fig. 4.

Now we will further test the influence of λ on AVCC and the search accuracy. The AVCC results are shown in Fig. 6. while the search accuracy results are shown in Fig. 7. By Fig. 7, we see that our algorithm achieves almost the same search accuracy as the algorithm in [10]. Fig. 6. shows that the AVCC of our algorithm is smaller than that of the algorithm in [10], which means lower computational burden.

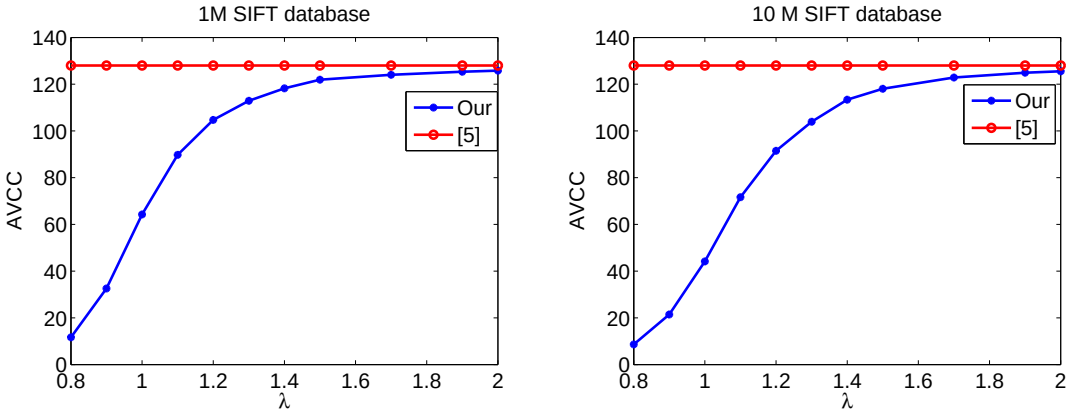

Fig. 6. Search accuracy (AVVC with different λ.)

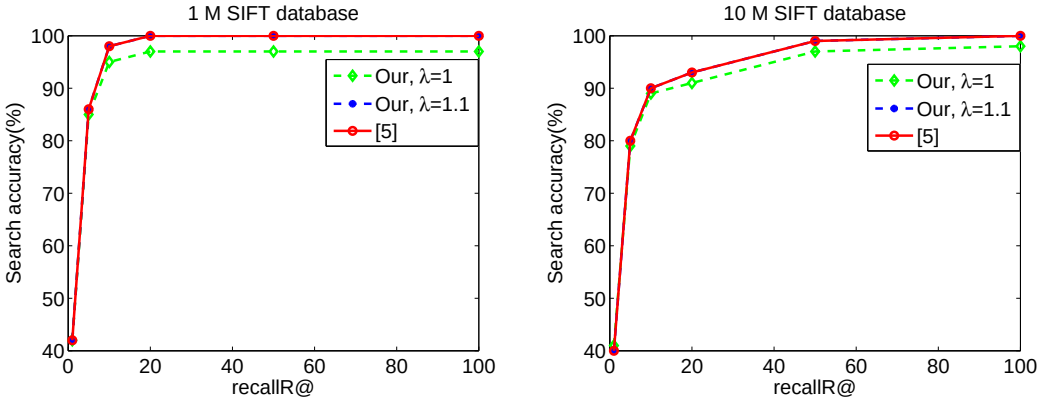

Fig. 6. The search accuracy (when $\lambda \geq 1.1$, the search accuracy is invariant with respect to λ)

Fig. 6 reveals that under our algorithm, AVVC increases as λ increases. We also see that when AVVC is small, the search accuracy is lower in Fig. 7. The search accuracy is almost the same when $\lambda \geq 1.1$, which means that our algorithm can achieve the same accuracy with the algorithm in [10].

The above experimental results demonstrate that our algorithm reduces the searching time with the same search accuracy. Moreover, it performs well for different databases.

5 Conclusion

In this paper, we present an optimized retrieval algorithm through adding an upper threshold to reduce the computational cost. The experimental results show that our algorithm can improve the retrieval speed with high search performance. In the future, we will try to find a better algorithm to estimate the real distance and improve the search performance further.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under Grant 61273112, the National Key Research and Development Project under Grant 2016YFC0201000 and the Young Innovation Association of CAS.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Boiman O, Shechtman E, Irani M. In defense of nearest-neighbor based image classification [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2008;1-8.

[2] Torralba A, Fergus R, Freeman WT. 80 million tiny images: A large data set for nonparametric object and scene recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2008;30(11):1958-1970.

[3] Friedman JH, Bentley JL, Finkel RA. An algorithm for finding best matches in logarithmic expected time [J]. ACM Transactions on Mathematical Software (TOMS). 1977;3(3):209-226.

[4] Silpa-Anan C, Hartley R. Optimised KD-trees for fast image descriptor matching [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2008;1-8.

[5] Moore AW. An intoductory tutorial on kd-trees [J]; 1991.

[6] Weiss Y, Torralba A, Fergus R. Spectral hashing [C]. Proceedings of the Advances in Neural Information Processing Systems. 2009;1753-1760.

[7] Datar M, Immorlica N, Indyk P, et al. Locality-sensitive hashing scheme based on p-stable distributions [C]. Proceedings of the twentieth annual symposium on Computational geometry, ACM. 2004;253-262.

[8] Muja M, Lowe DG. Fast approximate nearest neighbors with automatic algorithm configuration [J]. VISAPP. 2009;2(1).

[9] Muja M, Lowe DG. Scalable nearest neighbor algorithms for high dimensional data [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014;36(11):2227-2240.

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

[11] Jgou H, Douze M, Schmid C, et al. Aggregating local descriptors into a compact image representation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2010;3304-3311.

[12] Zhang T, Du C, Wang J. Composite quantization for approximate nearest neighbor search [C]. Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014;838-846.

[13] Norouzi M, Fleet DJ. Cartesian k-means [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2013;3017-3024.

[14] Ge T, He K, Ke Q, et al. Optimized product quantization for approximate nearest neighbor search [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2013;2946-2953.

[15] Özkan S, Ateş T, Tola E, et al. Feature encoding models for geographic image retrieval and categorization [C]. Proceedings of the 22nd IEEE Signal Processing and Communications Applications Conference (SIU). 2014;83-86.

[16] Tang B, Yiu ML, Hua KA. Exploit every bit: Effective caching for high-dimensional nearest neighbor search [J]. IEEE Transactions on Knowledge and Data Engineering. 2016;28(5):1175-1188.

[17] Sivic J, Zisserman A. Video Google: A text retrieval approach to object matching in videos [C]. Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV). 2003;1470-1477.

[18] Brandt J. Transform coding for fast approximate nearest neighbor search in high dimensions[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2010;1815-1822.

[19] Gray RM, Neuhoff DL. Quantization [J]. IEEE Transactions on Information Theory. 1998;44(6):2325-2383.

[20] Babenko A, Lempitsky V. The inverted multi-index [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2012;3069-3076.

[21] Zhou W, Yang M, Wang X, et al. Scalable feature matching by dual cascaded scalar quantization for image retrieval [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2016;38(1):159-171.

[22] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015;1-9.

[23] Hoare CAR. Quicksort [J]. The Computer Journal. 1962;5(1):10-16.

[24] MacQueen J. Some methods for classification and analysis of multivariate observations[C]. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1967;1(14):281-297.

[25] Zhou W, Lu Y, Li H, et al. Scalar quantization for large scale image search [C]. Proceedings of the 20th ACM International Conference on Multimedia. 2012;169-178.

[26] Lowe DG. Distinctive image features from scale-invariant keypoints [J]. International Journal of Computer Vision. 2004;60(2):91-110.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/15658