Bank of quantization models: A data-specific approach to learning binary codes for large-scale retrieval applications

Frederick Tung Julieta Martinez Holger H. Hoos James J. Little
University of British Columbia
{ftung,julm,hoos,little}@cs.ubc.ca

Abstract

We explore a novel paradigm in learning binary codes for large-scale image retrieval applications. Instead of learning a single globally optimal quantization model as in previous approaches, we encode the database points in a data-specific *manner using a bank of quantization models. Each individual database point selects the quantization model that minimizes its individual quantization error. We apply the idea of a bank of quantization models to data-independent and data-driven hashing methods for learning binary codes, obtaining state-of-the-art performance on three benchmark datasets.*

1. Introduction

The continuing growth of online image and video collections has motivated many advances in algorithms for large-scale recognition and retrieval. The goal of large-scale retrieval algorithms is to support efficient and accurate querying into the collection under memory and time constraints. Methods for large-scale retrieval may be divided into two main groups: hashing methods [3, 4, 6, 9, 12, 13, 15, 20, 21, 22, 23] and lookup-based methods [2, 7, 8, 10, 16]. Hashing methods learn a mapping from high-dimensional feature descriptors to compact binary codes such that the locality relationships in the original feature space are closely preserved in the reduced Hamming space. In large-scale applications, binary codes greatly reduce both memory requirements and query time. Lookup-based methods adaptively quantize the feature space and create a lookup table from cluster centroids to full-dimensional feature descriptors.

Which group of methods to choose is application dependent. State-of-the-art lookup-based methods tend to achieve higher recall at the cost of longer query times [6]. Hashing methods can take advantage of hardware-accelerated Hamming distance computation: the Hamming distance between two binary codes can be computed by performing an XOR and counting the ones, which is a fast operation

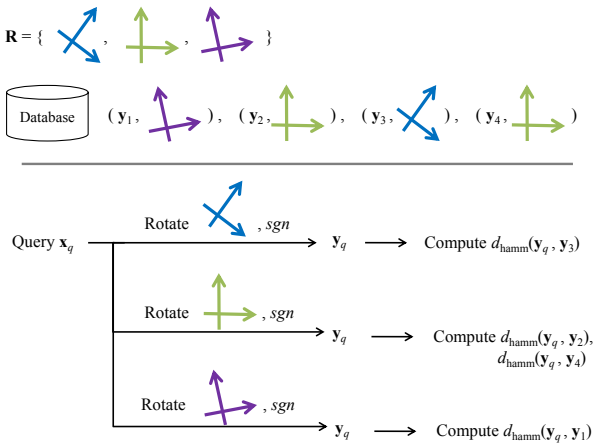

Figure 1. Toy example illustrating an online query with a bank of random rotations. Distances are computed adaptively, using each database point's selected rotation. Note that we perform K rotations but only the usual n Hamming distance computations (where n is the number of database points).

supported directly in modern hardware. Techniques for fast exact search in Hamming space [17] can further accelerate retrieval. Finally, hashing methods produce binary codes that can be used directly in downstream clustering or classification modules, which is not possible with lookup table methods [5]. In this paper, we propose a novel hashing paradigm in learning binary codes for large-scale retrieval applications. Previous hashing methods for learning binary codes optimize for a single global quantization model. In contrast, we employ a bank of multiple quantization models, and encode the database points in a *data-specific* manner. Each individual database point selects the quantization model that minimizes its individual quantization error. In other words, we take quantization error minimization to the extreme case and quantize the database in a data point specific manner. The framework can accommodate both data-independent (Section 3.1) and data-driven (Section 3.2) models. We show that the proposed approach yields state-of-the-art hashing performance on three standard retrieval benchmarks (Section 4).

2. Related work

Retrieval performance has been shown to be closely related to quantization error [2, 4, 6]. Therefore, it is useful to view retrieval approaches as lossy compression, whose objective is to reduce the quantization error. For a set of d-dimensional vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a set of m cluster centers (i.e., *codebook*) $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m\}$ the objective is to reduce the quantization error by solving

$$E = \min_{\mathcal{C}} \sum_{\mathbf{x} \in X} \|\mathbf{c}(\mathbf{x}) - \mathbf{x}\|^2 \tag{1}$$

where $\mathbf{c}$ is a quantizer function that maps $\mathbf{x}$ to its nearest cluster center in the codebook. Without any restrictions, the above function can be heuristically optimized by the k-means algorithm, which iteratively optimizes cluster locations and assignments. Although k-means exhibits better performance as the number of clusters increases, the algorithm is very hard to scale, as the required computation is quadratic in the number of clusters. Moreover, for k bits one would like to express 2^k clusters in order to achieve maximal compression, which makes the algorithm infeasible for large values of k.

It has been noticed, however, that the above formulation can be made tractable under certain restrictions. For example, product quantization (PQ) [8] looks for orthogonal subspaces that, taken together, can express a superlinear number of clusters. However, the method still requires a lookup table. On the other hand, the iterative quantization (ITQ) method [4] minimizes the objective

$$\begin{aligned} &E = \min_{\mathcal{C}} \sum_{\mathbf{x} \in X} \|\mathbf{c}(\mathbf{x}) - \mathbf{x}\|^2, \\ &\text{with } \mathbf{c} \in \mathcal{C} = \{\mathbf{c} \,|\, \mathbf{c} \cdot R = \{-a, a\}^d\},\ R^\top \cdot R = I. \end{aligned} \tag{2}$$

Since the distance between codewords remains constant, the resulting codes can be efficiently compared using Hamming distance. Intuitively, ITQ finds a rotation R of the PCA-projected, zero-centred data that minimizes the quantization error when mapping the data points to the closest corners of a binary hypercube, which represents 2^d cluster centers. This rotation is determined iteratively by alternating between updating the assignments given the current rotation, and updating the rotation to minimize the quantization error given the current assignments. The iterative optimization is initialized using a random rotation.

3. Proposed approach

Two observations motivate the bank of quantization models.

1. Most data-driven hashing methods try to solve for the quantization model that minimizes the global quantization error (Eq. 1). However, quantization error may still be high for *individual* data points.

2. For bit ranges of 32 to 128, several bits can be discarded with minimal effects on quantization performance.

The first observation motivates the search for multiple models to further reduce quantization error. The second observation leads us to allocate bits from the quantization model to index the best model *per data point*.

Formally, our objective function is

$$\begin{aligned} &E = \min_{\mathcal{C}} \sum_{\mathbf{x} \in X} \|\mathbf{c}(\mathbf{x}) - \mathbf{x}\|^2 \\ &\mathcal{C} = \{\mathcal{C}_i \,|\, i = 1 \ldots 2^k\} \\ &\mathbf{c} \in \mathcal{C}_i = \{\mathbf{c} \,|\, \mathbf{c} \cdot R_i = \{-a, a\}^{d-k}\},\ R_i^\top \cdot R_i = I \end{aligned} \tag{3}$$

where i indicates the index of the quantization model assigned to the data point $\mathbf{x}$. Note that we have 2^k models, use k bits to index them and therefore are still able to express $2^k(2^{d-k}) = 2^d$ cluster centers in total. However, the orthogonality constraint imposed by R_i does not span all the dimensions; rather, we have 2^k subspaces of $d - k$ dimensions each. This union of orthogonal models can be seen as a generalization of the orthogonality constraint of ITQ [4].

We now apply the simplest instantiation of our bank of quantization models idea to create an effective *data-independent* hashing method.

3.1. Data-independent hashing: Bank of random rotations (BRR)

The simplest instantiation of our idea is inspired by the observation that a random rotation of the cluster centers is a remarkably strong benchmark in both PQ and ITQ. Therefore, we propose to instantiate Eq. 3 by generating a collection of $K = 2^k$ random rotations $\mathcal{R} = \{R_1, R_2, ..., R_K\}$. We encode the database as follows. We first preprocess the database points by zero-centering and PCA embedding. Next, for each individual data point $\mathbf{x}$ we find $R^* \in \mathcal{R}$ that satisfies

$$R^* = \underset{R \in \mathcal{R}}{\operatorname{argmax}} \|\mathbf{x}R\|_1 \tag{4}$$

where $||\cdot||$ is the L1 norm (for space reasons we omit a proof showing that this has the equivalent effect as selecting based on individual quantization error). Encoding must be performed by assigning $\mathbf{x}$ to its nearest cluster center. Due to the orthogonality constraint, this is achieved by simply thresholding at zero: $\mathbf{y} = sgn(\mathbf{x}R^*)$. Finally, we store the pair $(\mathbf{y}, j)$, where j is the index of R^* in the bank of random rotations $\mathcal{R}$. That is, j identifies the rotation used to produce the binary code $\mathbf{y}$. Note that we allocate bits

from our budget to index the best quantization model (i.e. to store j) for each individual data point, and therefore incur no additional memory usage for the database. There is, however, a small memory overhead for storing the bank.

Next, we explain how online queries work in a bank of quantization models framework. Given a (zero-centered, PCA-embedded) query point $\mathbf{x}_q$, we compute distances to the database points *adaptively*, using each database point's selected quantization model (rotation). More specifically, for each rotation R in the bank, we quantize $\mathbf{x}_q$ with respect to the rotation R via $\mathbf{y}_q = sgn(\mathbf{x}_q R)$, and compute the Hamming distance of $\mathbf{y}_q$ to all database binary points that selected R as their rotation. Figure 1 illustrates with a toy example. For greater clarity, we emphasize that for a single query, we perform K rotations (typical overhead of fractions of a millisecond) and then compute only n Hamming distances – not Kn distances. The number of Hamming-space comparisons is unchanged by the bank of rotations.

Despite the simplicity of the bank of random rotations, we show in the experiments that it provides competitive retrieval accuracy with state-of-the-art hashing methods, without requiring any complex optimization or training. In addition, large-scale applications often involve databases that grow over time as users continually add images or video. In the bank of random rotations approach, when a new point is added to the database it selects a rotation from the bank independently from all other database points. In contrast to data-driven hashing methods, no global optimization problem over the database needs to be re-solved. The proposed bank of random rotations approach therefore offers a particularly scalable solution in large-scale applications where the database is expected to grow over time.

A natural question to ask is whether the rotations can be learned from the data. We next describe a *data-driven* instantiation of our bank of quantization models idea in which the rotations are learned.

3.2. Data-driven hashing: Bank of iterative quantizations with hypercube stretch (BITQS)

A natural extension is to replace the bank of random rotations with a bank of quantization models that are (a) learned (i.e., data-driven) and (b) different from each other (i.e. complementary). The intuition is that even if the quantization models are locally optimal, the union of them will have a better chance at approximating a globally optimal solution.

ITQ satisifies both requirements: it iteratively adapts to the distribution of the data, so as to minimize its quantization error, and thanks to the use of a random initialization and subsequent coordinate descent, the rotations learned via ITQ are locally optimal and different from each other. However, having only one degree of freedom (R) limits the complementarity of the models. Since we are minimizing quantization error, it makes sense to look for models with more degrees of freedom, in the hope that they will better complement each other.

It has been recently shown that better retrieval performance can be obtained by exploring the tradeoff between quantization error and Hamming-to-Euclidean distance (*affinity*) error. K-means hashing [6] jointly minimizes both terms by constraining the relative positions of k-means clusters. On the other hand, orthogonal k-means (ok-means) [16] maintains the orthogonality constraint of ITQ, but also adds translation and non-uniform scaling to the model. While translation does not impact the affinity error, non-isotropic scaling of the quantization hypercube does, and the metric between the learned codes becomes a *scaled* Hamming distance. In practice, however, we observe that the non-scaled metric maintains a very competitive performance, therefore being directly comparable to other Hamming approaches. ok-means also offers other advantages, such as the possibility to learn the mapping directly in the feature space, and the inclusion of an out-of-projection error, which can reduce quantization error by projecting out large parts of the feature space. In practice, though, we observe that simply keeping the non-uniform scaling is responsible for almost all the improvement in retrieval performance, and therefore only incorporate that into our objective.

Formally, our objective function becomes the search for a rotation and a scaling which satisfy

$$(R^*, S^*) = \underset{(R,S)}{\operatorname{argmin}} \|YS - XR\|_F^2, \tag{5}$$

where $S \in \mathbb{R}^{c \times c}$ is a diagonal matrix whose entries $s_1, \ldots, s_c$ contain the scaling per dimension of the quantization hypercube. We also use a random initialization for R and coordinate descent to optimize Eq. 5, alternatively solving for R (solving a Procrustes problem), S (via 1-dimensional k-means), and the cluster assignments. In practice, we find that updating S with the mean of the data is comparable to using k-means, and use that in all our experiments. This results in a two-line modification to ITQ.

Our data-driven extension replaces the random rotations in the previous section with rotation-scaling pairs found by solving Eq. 5 with different random initializations. Given the bank of rotation-scaling pairs, the database encoding and online query processes are analogous to the data-independent instantiation in the previous section: we encode the rotation-scaling pair that minimizes the individual quantization error for each database point, and at query time compute distances to the database points adaptively using each database point's selected rotation-scaling pair.

4. Experiments

In this section, we evaluate the retrieval performance of both our data-independent (BRR) and data-driven (BITQS) instantations of the bank of quantization models framework. We explore suitable parameters for our approach and compare to state-of-the-art hashing methods on three benchmarks.

Methodology. Experiments were conducted on three standard retrieval benchmarks: SIFT1M and GIST1M, both from [8], and CIFAR-10 from [11]. The SIFT1M dataset consists of 100,000 128-dimensional SIFT [14] descriptors for training, 1 million descriptors for the database and 10,000 descriptors for queries. GIST1M consists of 500,000 960-dimensional Gist [18] descriptors for training, 1 million for the database and 1,000 for queries. The CIFAR-10 dataset is a 60,000-image, 10-class subset of the Tiny Images dataset [19]. We computed 384-dimensional Gist descriptors for each CIFAR-10 image and randomly split 6,000 descriptors for train, 600 for queries and the rest for database. We show retrieval performance using recall@N curves [2, 6, 8, 16], which plot the proportion of true neighbors retrieved in the first N Hamming neighbors. We followed the convention of He et al. [6] and considered the ground truth to be the 10 nearest Euclidean neighbors of the query. We used public implementations of spectral hashing, iterative quantization, and k-means hashing.

Recall that, for each database point, we allocate bits from our budget to store the identifier (index) of the quantization model selected from the bank. In our comparisons to the state-of-the-art methods, we adjusted the number of bits in our binary codes accordingly. For example, for 64-bit codes and a bank of 256 random rotations, we computed 56-bit codes since 8 bits are needed to index into the bank.

Choosing the number of models. Our method requires only one parameter to be set, that of k in Eq. 3. There is a natural trade-off between the number of quantization models and the information loss due to quantizing on fewer bits. We explore this trade-off in Figure 2. We observe that retrieval performance increases as more models are added, saturating at around $K = 256$. Eventually, performance decreases when too many bits are allocated for indexing; this is easier to see when the budget is more constrained (e.g., 32 bits). We note that using 256 models (i.e., using $k = 8$ bits for indexing), we obtain good performance across all datasets and bit budgets, and in what follows, we use this parameter for all our experiments.

Comparison with the state of the art. We compare the two instantiations of our bank of quantization models idea – the data-independent BRR and the data-driven BITQS – with three state-of-the-art hashing methods: spectral hashing [22], iterative quantization [4], and k-means hashing [6]. We include also two classical baselines: PCA hashing and locality sensitive hashing. K-means hashing requires a product-space decomposition [8] to be tractable. We performed this split into product subspaces following the split used by He et al. [6] in their experiments: for SIFT descriptors, 2 bits per subspace for 32-bit codes and 4 bits per subspace for 64-bit and 128-bit codes; and for Gist descriptors, 8 bits per subspace. In these experiments, we ran ITQ for 50 iterations following Gong and Lazebnik [4], and give the same number of iterations to our models in BITQS.

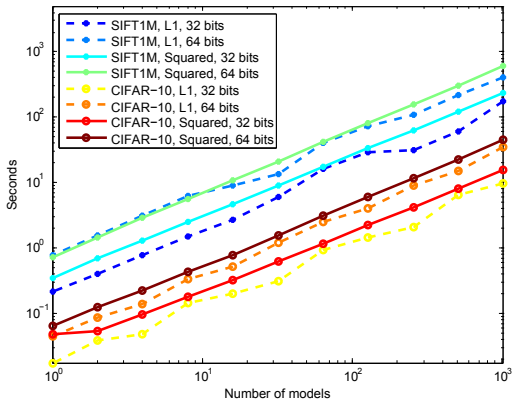

Figure 4. Running times for building an index offline using the L1 norm criterion (Eq. 4) and the squared loss (Eq. 5). Measurements were obtained on a desktop with a 3.40GHz processor, using a single core.

Figure 3 shows the retrieval performance of 32-bit, 64-bit, and 128-bit codes on the SIFT1M, GIST1M, and CIFAR-10 benchmarks. We observed that, despite its simplicity, the data-independent bank of random rotations (BRR) outperforms the state of the art on GIST1M and CIFAR-10. On SIFT1M, at 64 bits and 128 bits the bank of random rotations is comparable to k-means hashing, despite not requiring any training or complex sequential optimization. The data-driven bank of iterative quantizations with hypercube stretching (BITQS) outperforms the state of the art on SIFT1M for 32 bits, 64 bits, and 128 bits.

Memory and time requirements. Memory requirements are the usual for hashing approaches (i.e., d bits per data point). Since we allocate bits from our budget to store the indexing of each point, we incur no additional memory usage for the database. There is, however, a linear overhead on the number of models, as our approaches need to store K $c \times c$ rotations. For 256 random rotations and $d = 128$ bits per data point, this amounts to about 14 MB, which can be reduced by slightly more than half using the exponential map of $SO(c)$ [1].

For the data-driven BITQS only, we first have to train K models, and then evaluate them on each point of the database offline. We report running times for this step in Figure 4. The quantization models in BITQS are trained using iterative optimization approaches, and there is a linear overhead on the number of models. For K models, each

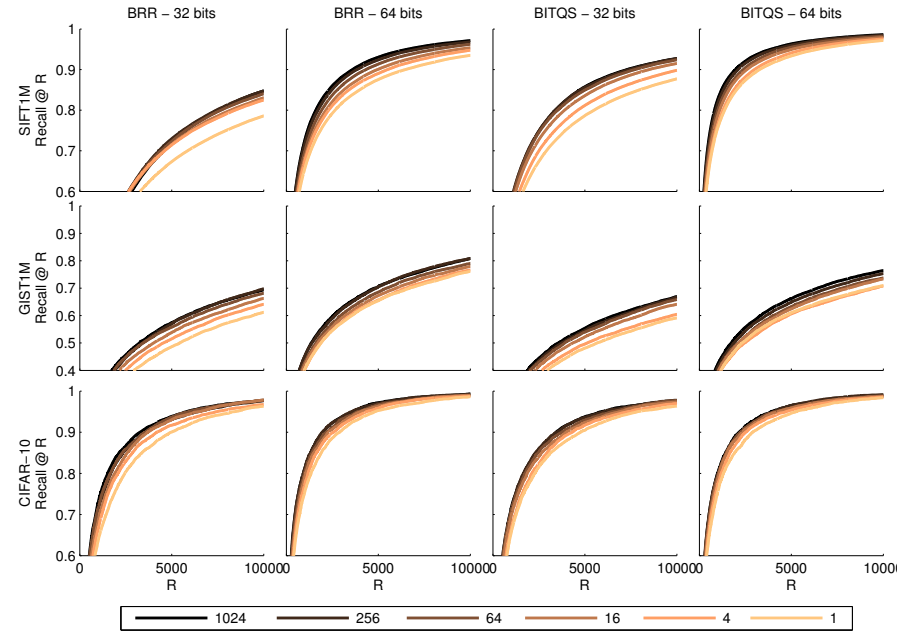

Figure 2. Trade-off between the number of models ($K = 2^k$, indicated in the label) and retrieval performance for BRR and BITQS. We find similar results for 128 bits.

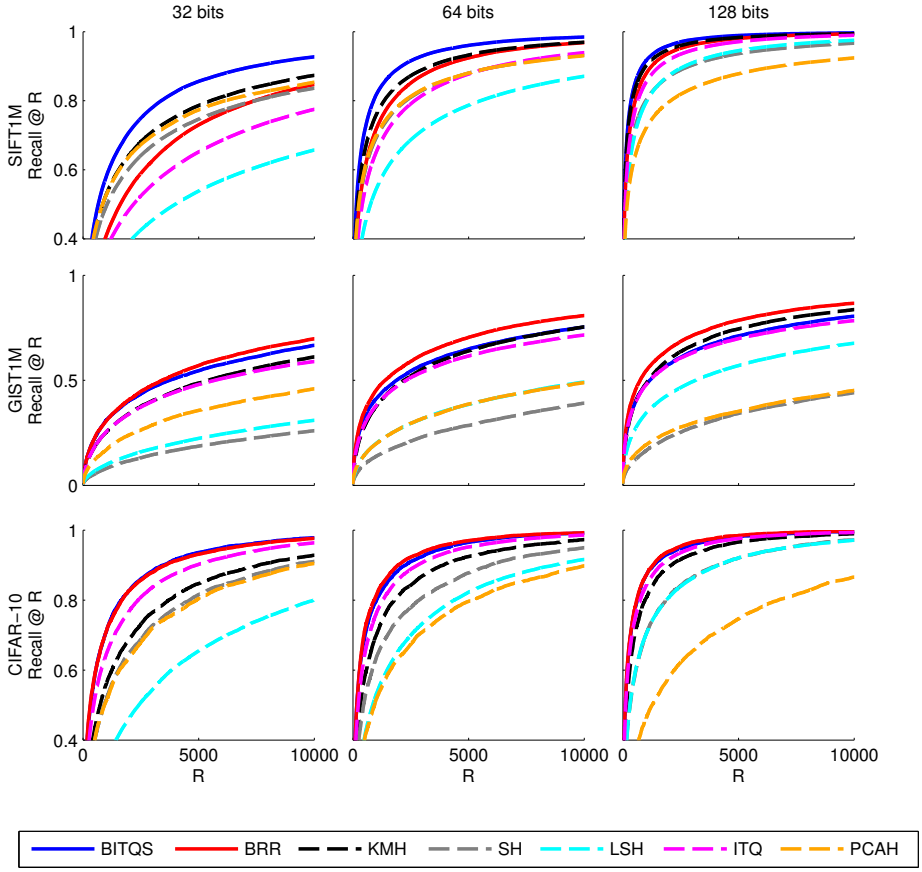

Figure 3. Comparison with three state-of-the-art hashing methods (SH: spectral hashing, ITQ: iterative quantization, KMH: k-means hashing) and two classical methods (LSH: locality sensitive hashing, PCAH: PCA hashing) on SIFT1M, GIST1M, and CIFAR-10 benchmarks.

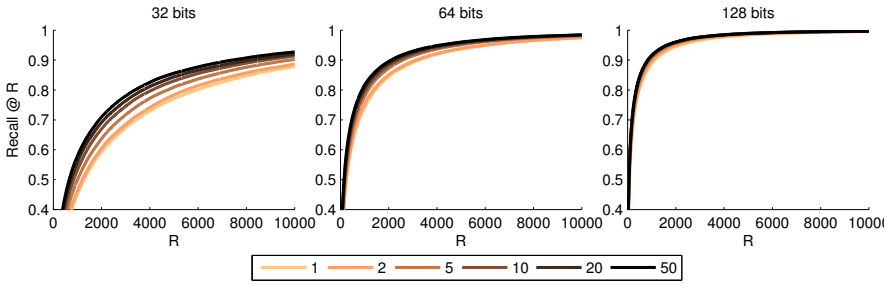

Figure 5. Retrieval performance as a function of per-model iterations in BITQS (using 256 models) on the SIFT1M dataset.

running for m iterations, this amounts to mK iterations in total. In our experiments we set $m = 50$ following [4]. Since BITQS shows the best performance in SIFT1M, we explore different values of m in Figure 5. We observe that competitive performance can be obtained with around 20 iterations, reducing training times by slightly more than half.

When performing a query online, we incur a small overhead from applying K rotations, prior to the usual n Hamming distance computations. Typically this overhead is on the order of fractions of a millisecond ($\sim$0.05 ms for $K = 256$), and can be easily parallelized.

5. Conclusion

We have presented a novel paradigm in learning binary codes for large-scale retrieval applications, in which we quantize the database in a data point specific manner. This idea is implemented by indexing into a bank of quantization models instead of optimizing a single quantization model as in previous approaches. We presented simple instantiations of our framework using data-independent random rotations (BRR) and data-driven iterative quantization with hypercube stretching (BITQS). We showed that BRR already obtains state-of-the-art hashing performance, without any complex optimization or training as required by other state-of-the-art approaches. Moreover, BRR offers a scalable solution in large-scale applications where the database is expected to grow over time, as no re-optimization is needed when new database points are added.

Acknowledgments. This research is sponsored in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] M. do Carmo. *Riemannian Geometry*. Springer, 1992.

[2] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2013.

[3] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 2013.

[4] Y. Gong and S. Lazebnik. Iterative quantization: a Procrustean approach to learning binary codes. In *CVPR*, 2011.

[5] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: a Procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35(12):2916–2929, 2013.

[6] K. He, F. Wen, and J. Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[7] J.-P. Heo, Z. Lin, and S.-E. Yoon. Distance encoded product quantization. In *CVPR*, 2014.

[8] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, 2011.

[9] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian. Super-bit locality-sensitive hashing. In *NIPS*, 2012.

[10] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2014.

[11] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[12] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

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

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[15] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[16] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, 2013.

[17] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in Hamming space with multi-index hashing. In *CVPR*, 2012.

[18] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[19] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *TPAMI*, 30(11):1958–1970, 2008.

[20] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[21] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[23] G. Ye, D. Liu, J. Wang, and S.-F. Chang. Large-scale video hashing via structure learning. In *ICCV*, 2013.