Projected Residual Vector Quantization for ANN Search

Benchang Wei, Tao Guan, and Junqing Yu
Huazhong University of Science & Technology

A method of projected residual vector quantization for approximate nearest neighbor (ANN) search that considers the projection errors in the quantization process can address the problem of large-scale ANN search in a high-dimensional space.

Nearest neighbor (NN) search is of great importance for many computer vision applications, such as feature matching,[1] content-based video retrieval (CBVR),[2] object recognition,[3] social image retrieval,[4] and mobile landmark recognition.[5–7] For example, to obtain accurate on-device mobile landmark recognition, we need a fast and effective NN search algorithm to perform landmark recognition by searching the results from a million-scale image dataset.

Exact NN search is computationally inefficient in dealing with high-dimensional similarity search problems, however, because of the curse of dimensionality.[8,9] Approximate nearest neighborhood (ANN) search algorithms have been proposed to solve this problem. The main goal shared by different ANN algorithms is to find the NN with a high probability "only," instead of a probability of 1.[10]

Traditional ANN algorithms such as tree-based (such as KD-tree,[11] vocabulary tree,[12] and FLANN[13]) and hashing-based (such as Exact Euclidean Locality Sensitive Hashing, or E2LSH[14]) methods commonly use indexing structures to accelerate the search process. Although promising, these methods need to store the original vectors in the main memory for final reranking use, which seriously limits the scale of the database used.

Recently, researchers have proposed several ANN search methods based on quantization and asymmetric distance to deal with the large-scale high-dimensional vector-searching problem.[10,15,16] Compared with tree- or hashing-based methods, these methods can compress high-dimensional vectors into short codes while providing rational searching accuracy. For example, Hervé Jégou and his colleagues proposed a product quantization (PQ) method that decomposes the vector space into a Cartesian product of low-dimensional subspaces.[10] A vector is represented by a short code that consists of its subspace quantization indices. The distance between two vectors can be estimated efficiently from their codes and a lookup table. However, PQ requires prior knowledge about the structures of the input vector; otherwise, the searching accuracy substantially declines. To overcome this problem, Yongjian Chen and his colleagues designed an efficient ANN algorithm that uses residual vector quantization.[16] Unlike PQ, RVQ treats the vector space as a whole and does not rely on prior knowledge of the input vector's structural information.

Existing quantization and asymmetric distance-based ANN methods commonly use principal component analysis (PCA) to reduce the dimensionality of input vectors. Although promising, existing algorithms often ignore an important problem: the projection errors generated in the PCA process will be completely discarded during the quantization and encoding process, which will inevitably lower the search accuracy. In view of that, we propose a projected residual vector quantization (PRVQ) method that incorporates the projection errors into the vector quantization process. We also designed three simple and effective optimization strategies to improve the performance of our PRVQ algorithm by minimizing the projection and quantization errors. Our method differs from the optimization method in earlier work,[17] in which the dimensionality reduction and vector quantization are jointly optimized to obtain a precise vector comparison and a compact representation and the projection errors are discarded in the quantization process. The effectiveness of our PRVQ algorithm is validated on two kinds of high-dimensional vectors: namely, GIST and vector of locally aggregated descriptors (VLAD) descriptors. A comparison with the state of the art shows that our approach outperforms existing techniques, in particular PQ,[10] transform coding (TC),[15] and RVQ.[16] We also prove that the proposed PRVQ algorithm can be used to

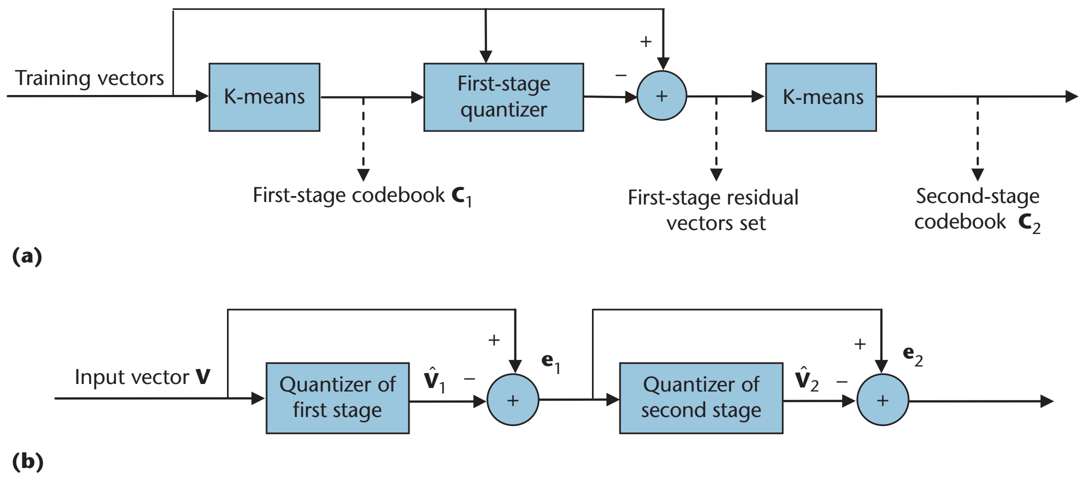


Figure 1. Illustration of residual vector quantization.[16] *(a) Learning codebooks. (b) Quantizing a vector.*

improve the recognition accuracy of mobile landmark search systems.

Residual Vector Quantization

RVQ contains several stage codebooks (quantizers),[16] each of which is used to generate the quantization errors that will be simulated by the next-stage codebook.

Figure 1a illustrates the RVQ quantizer generation process. To obtain the first-stage codebook $C_1 = \{cp_1^1, cp_1^2, \ldots, cp_1^K\}$, a K-means process is carried out on the set of input vectors to generate a set of K cluster centroids. Each input vector $\mathbf{V}$ can then be quantized by using the first-stage quantizer defined by C_1 as follows:

$$\hat{\mathbf{V}}_1 = \arg\min_{c_1^k \in C_1} d(\mathbf{V}, c_1^k) \quad (1)$$

where $d(\mathbf{V}, c_1^k)$ is the Euclidean distance between $\mathbf{V}$ and c_1^k.

To obtain the second-stage codebook, we need to first compute the first-stage quantization error of each input vector $\mathbf{V}$:

$$\mathbf{e}_1 = \mathbf{V} - \hat{\mathbf{V}}_1 \quad (2)$$

The second-stage codebook $C_2 = \{c_2^1, c_2^2, \ldots, c_2^K\}$ can then be generated by carrying out K-means on the obtained residual vector set. To get an N-stage RVQ quantizer that contains N-stage codebooks, we need to repeat this process N times. The residual vectors obtained in the final stage will be discarded.

As Figure 1b shows, to encode an input vector $\mathbf{V}$, we need to quantize $\mathbf{V}$ sequentially by using the built-stage codebooks and store the indices of quantization outputs. For an N-stage quantizer that contains N-stage codebooks, the corresponding code string of $\mathbf{V}$ is $B = (b_1, b_2, \ldots, b_N\}$, where $b_n (1 \le n \le N)$ is the serial number of the centroid that belongs to codebook C_l and has the least distance to the $\mathbf{V}$'s $(n-1)$th-stage residual vector $\mathbf{e}_{n-1}$.

For each vector $\mathbf{V}$, we can reconstruct its approximate vector $\hat{\mathbf{V}}$ by using the corresponding code $B = (b_1, b_2, \ldots, b_N\}$ as follows:

$$\hat{\mathbf{V}} = \sum_{n=1}^{N} \hat{\mathbf{V}}_n = \sum_{n=1}^{N} c_n^{b_n}, c_n^{b_n} \in C_n, 1 \le b_n \le K \quad (3)$$

where $c_n^{b_n}$ is the b_n-th centroid of codebook C_n.

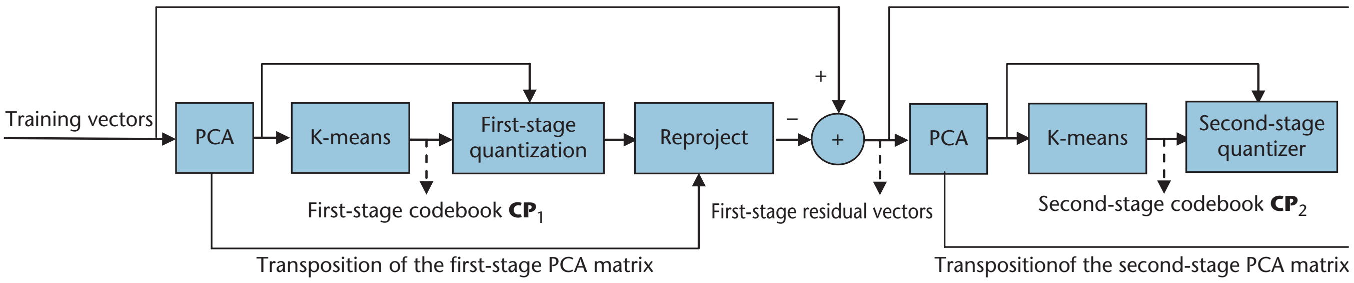
When a query vector $\mathbf{V}^q$ is submitted, we can compute the similarity between $\mathbf{V}^q$ and each vector $\mathbf{V}$ by using the asymmetric distance $\hat{D}$:

$$\begin{aligned}\hat{D}(\mathbf{V}^q, \mathbf{V})^2 &= D(\mathbf{V}^q, \hat{\mathbf{V}})^2 \\ &= \left\|\mathbf{V}^q - \hat{\mathbf{V}}\right\|^2 = \|\mathbf{V}^q\|^2 + \left\|\hat{\mathbf{V}}\right\|^2 \\ &\quad - 2\left\langle \mathbf{V}^q \cdot \hat{\mathbf{V}} \right\rangle \\ &= \|\mathbf{V}^q\|^2 + \left\|\hat{\mathbf{V}}\right\|^2 - 2\sum_{n=1}^{N} \left\langle \mathbf{V}^q \cdot c_n^{b_n} \right\rangle \end{aligned} \quad (4)$$

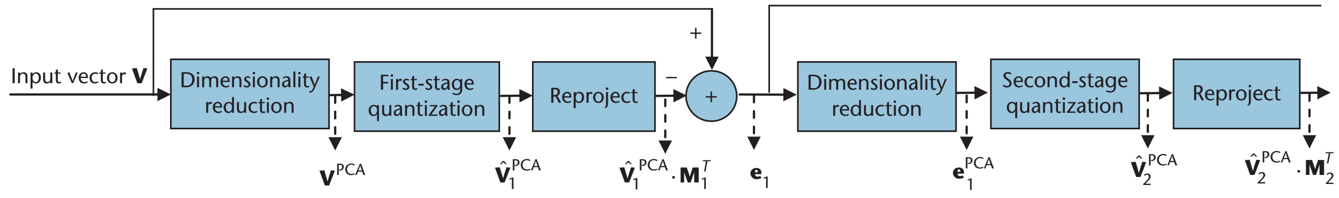
where $\langle \cdot \rangle$ is a dot product, $\|\mathbf{V}^q\|^2$ can be computed in advance, and $\left\langle \mathbf{V}^q \cdot c_n^{b_n} \right\rangle$ can be computed and stored in a lookup table when $\mathbf{V}^q$ is submitted.

PRVQ-Based ANN Search

PCA is commonly used in RVQ to reduce the dimensionality of vectors in order to accelerate the codebook training and lookup table


Training vectors
PCA
K-means
First-stage quantization
Reproject
+
−
+
PCA
K-means
Second-stage quantizer
First-stage codebook CP$_1$
First-stage residual vectors
Second-stage codebook CP$_2$
Transposition of the first-stage PCA matrix
Transpositionof the second-stage PCA matrix

(a)


Input vector V
Dimensionality reduction
First-stage quantization
Reproject
+
−
+
Dimensionality reduction
Second-stage quantization
Reproject
$\mathbf{V}^{PCA}$
$\hat{\mathbf{V}}_1^{PCA}$
$\hat{\mathbf{V}}_1^{PCA} \cdot \mathbf{M}_1^T$
$\mathbf{e}_1$
$\mathbf{e}_1^{PCA}$
$\hat{\mathbf{V}}_2^{PCA}$
$\hat{\mathbf{V}}_2^{PCA} \cdot \mathbf{M}_2^T$

(b)

Figure 2. Illustration of projected residual vector quantization. (a) Learning codebooks. (b) Quantizing a vector.

generation processes. For each original vector **V**, a PCA matrix $\boldsymbol{M}$ is used to obtain the corresponding dimensionality-reduced vector $\mathbf{V}^{\mathrm{PCA}}$:

$$\mathbf{V}^{\mathrm{PCA}} = \mathbf{V} \cdot \boldsymbol{M} \tag{5}$$

Instead of the original vector **V,** the PCA-compressed vector $\mathbf{V}^{\mathrm{PCA}}$ will be used in the RVQ quantizer generation and vector-searching processes, and the PCA projection error given in the following equation will be discarded absolutely:

$$\mathbf{e}^p = \mathbf{V} - \mathbf{V}^{\mathrm{PCA}} \cdot \boldsymbol{M}^T \tag{6}$$

where $\boldsymbol{M}^T$ is the transposition of the PCA matrix $\boldsymbol{M}$.

Abandoning the projection error will inevitably deteriorate the quantization quality, which will then reduce the search accuracy. To address that problem, PRVQ considers both the projection error and quantization error in the quantizer generation process. The detailed algorithm is described as follows.

Figure 2a shows the PRVQ quantizer building method. To generate the first-stage codebook, we use PCA to compress each P-dimensional input vector **V** into a T-dimensional $(T \ll P)$ vector $\mathbf{V}^{\mathrm{PCA}}$. We then carry out a K-means process on the set of PCA-compressed vectors to generate the first-stage codebook $CP_1 = \{cp_1^1, cp_1^2, \ldots, cp_1^K\}$. The first-stage quantization result $\hat{\mathbf{V}}_1^{\mathrm{PCA}}$ of each vector $\mathbf{V}^{\mathrm{PCA}}$ can then be obtained by

$$\hat{\mathbf{V}}_1^{\mathrm{PCA}} = \underset{cp_1^k \in CP_1}{\arg\min}\, d(\mathbf{V}^{\mathrm{PCA}}, cp_1^k) \tag{7}$$

where $d(\mathbf{V}^{\mathrm{PCA}}, cp_1^k)$ is the Euclidean distance between $\mathbf{V}^{\mathrm{PCA}}$ and cp_1^K.

The residual vector of each vector **V** generated in the first projection and quantization stage can then be computed:

$$\begin{aligned}\mathbf{e}_1 &= \mathbf{e}_1^p + \mathbf{e}_1^q \\ &= \left(\mathbf{V} - \mathbf{V}^{\mathrm{PCA}} \cdot \boldsymbol{M}_1^T\right) \\ &\quad + \left(\mathbf{V}^{\mathrm{PCA}} \cdot \boldsymbol{M}_1^T - \hat{\mathbf{V}}_1^{\mathrm{PCA}} \cdot \boldsymbol{M}_1^T\right) \\ &= \mathbf{V} - \hat{\mathbf{V}}_1^{\mathrm{PCA}} \cdot \boldsymbol{M}\end{aligned} \tag{8}$$

where $\mathbf{e}_1^p$ and $\mathbf{e}_1^q$ are the first-stage projection and quantization errors, respectively, and M_1^T is the transposition of the first-stage PCA matrix $\boldsymbol{M}_1$.

With the residual vector $\mathbf{e}_1$ of each input vector **V** computed, we can generate the second-stage codebook $CP_2 = \{cp_2^1, cp_2^2, \ldots, cp_2^K\}$ by performing K-means on the set of PCA-compressed residual vectors. This process needs to be performed N times to get an N-stage PRVQ quantizer, and the residual vectors obtained in the final stage will be discarded.

Figure 2b illustrates the quantization and encoding process. Each input vector **V** can be encoded into a short code $BP = \{bp_1,$

$bp_2, \ldots, bp_N\}$, where $bp_n (1 \le n \le N)$ is the serial number of the centroid that belongs to the codebook CP_n and has the least distance to $\mathbf{V}$'s (n – 1)th-stage PCA-compressed residual vector $\mathbf{e}_{n-1} \cdot \boldsymbol{M}_{n-1}$.

The approximate vector $\hat{\mathbf{V}}$ of vector $\mathbf{V}$ can be computed as follows:

$$\begin{aligned}\hat{\mathbf{V}} &= \sum_{n=1}^{N} \hat{\mathbf{V}}_n^{\mathrm{PCA}} \cdot \boldsymbol{M}_n^T \\ &= \sum_{n=1}^{N} cp_n^{bp_n} \cdot M_n^T, \quad cp_n^{bp_n} \in CP_n, 1 \le bp_n \le K\end{aligned} \tag{9}$$

where $cp_n^{bp_n}$ is the bp_n-th centroid of codebook CP_n.

In the query stage, the similarity between query vector $\mathbf{V}^q$ and each database vector $\mathbf{V}$ can be computed as

$$\begin{aligned}\hat{D}(\mathbf{V}^q, \mathbf{V})^2 &= \left\|\mathbf{V}^q - \hat{\mathbf{V}}\right\|^2 = \|\mathbf{V}^q\|^2 + \left\|\hat{\mathbf{V}}\right\|^2 \\ &\quad - 2\left\langle \mathbf{V}^q, \hat{\mathbf{V}} \right\rangle \\ &= \|\mathbf{V}^q\|^2 + \left\|\hat{\mathbf{V}}\right\|^2 \\ &\quad - 2\left\langle \mathbf{V}^q, \sum_{n=1}^{N} cp_n^{bp_n} \cdot \boldsymbol{M}_n^T \right\rangle \\ &= \|\mathbf{V}^q\|^2 + \left\|\hat{\mathbf{V}}\right\|^2 \\ &\quad - 2\sum_{n=1}^{N} \left\langle \mathbf{V}^q,\ cp_n^{bp_n} \cdot \boldsymbol{M}_n^T \right\rangle \\ &= \|\mathbf{V}^q\|^2 + \left\|\hat{\mathbf{V}}\right\|^2 \\ &\quad - 2\sum_{n=1}^{N} \left\langle \mathbf{V}^q \cdot \boldsymbol{M}_n,\ cp_n^{bp_n} \right\rangle\end{aligned} \tag{10}$$

where $\left\langle \mathbf{V}^q \cdot \boldsymbol{M}_n, cp_n^{bp_n} \right\rangle$ can be computed and stored in a look-up table when query vector $\mathbf{V}^q$ is submitted.

PRVQ Optimization

This section deals with optimizing the proposed PRVQ algorithm to improve search by minimizing the Nth-stage residual $\mathbf{e}_N$ of each training vector. The rationale of our approach is that the residual $\mathbf{e}_N$ is an important indicator when evaluating the quantization accuracy. The smaller the value of $\mathbf{e}_N$, the more the encoded vector will be similar to the original vector.

We use three strategies to optimize our PRVQ algorithm. First, we optimize the parameter T to minimize $\mathbf{e}_N$ by setting a constraint on the stage number N and codebook size K—for instance, $N = 8$ and $K = 256$. Second, we optimize the quantizer CP_n and stage PCA matrix $\boldsymbol{M}_n$ to minimize $\mathbf{e}_N$ by using fixed K and N. Third, we optimize the quantization and encoding process to further reduce $\mathbf{e}_N$ by using fixed CP_n and $\boldsymbol{M}_n$. These three optimization methods are performed in an offline training and encoding stage and will not increase the computational complexity of the online search stage at all.

Optimizing Parameter *T*

The parameter T influences the performance of our method. For small values of T, large projection errors will be introduced; for large values of T, large quantization errors will be introduced. In fact, both projection and quantization errors will deteriorate the search accuracy. So, when the parameters N and K are fixed, we determine the values of T by minimizing the average value of residual errors obtained from all the training vectors:

$$E = \sum_{s=1}^{S} \left\|\mathbf{e}_N^s\right\| \Big/ S \tag{11}$$

where S is the number of training vectors and $\mathbf{e}_N^s$ is the Sth training vector's residual vector generated in stage N.

Optimizing the Stage Quantizer and PCA Matrix

With the value of T determined, we then turn to the second strategy to optimize the stage quantizer and stage PCA matrix. Because of the uneven distribution of the training data, we cannot be sure we can obtain the optimal quantizer and PCA projection matrix with one training. So, we integrate the obtained residuals into the projection and training processes to obtain more stable quantization results. We do this by refining the quantizer and PCA matrix stage by stage.

To refine the first-stage PCA matrix and quantizer, we need to compute the sum of each training vector's first-stage quantization result's reprojection $\hat{\mathbf{V}}_1^{\mathrm{PCA}} \cdot \boldsymbol{M}_1^T$ and its Nth stage residual vector $\mathbf{e}_N$ as follows:

$$R_1 = \hat{\mathbf{V}}_1^{\mathrm{PCA}} \cdot \boldsymbol{M}_1^T + \mathbf{e}_N \tag{12}$$

With the R_1 of each training vector obtained, we can then recompute the first-stage PCA matrix $\boldsymbol{M}_1$ and retrain the first-stage quantizer.

Each training vector can then be requantized to generate a new residual vector.

To refine the second-stage PCA matrix and quantizer, we need to compute the R_2 of each training vector by using the newly obtained residual vector $\mathbf{e}_N$:

$$R_2 = \hat{\mathbf{V}}_2^{\text{PCA}} \cdot \boldsymbol{M}_2^T + \mathbf{e}_N \tag{13}$$

Then, we recompute the second-stage PCA matrix $\boldsymbol{M}_2$ and retrain the second-stage quantizer CP_2. This process is repeated in each of the remaining stages to fulfill one round of optimization. In our method, we perform several rounds of this optimization process until the change in the value of E between two rounds is less than 0.1 percent. The optimized-stage PCA matrices and quantizers will be used to generate the code of each database vector for use in the search.

Optimizing the Encoding Process

The third strategy is to optimize the encoding process to further minimize the residual $\mathbf{e}_N$ of each database vector. In the encoding process discussed in the "PRVQ-Based ANN Search" section, the greedy algorithm is used in each stage to quantize an input vector. Although it is computationally efficient, the greedy algorithm may produce suboptimal quantization results. On the other hand, it is impractical both in memory consumption and computational complexity to get the optimal encoding result by comparing the input vector with all possible vectors that can be reconstructed by using the built-stage codebooks and Equation 9. In view of that, we designed a simple and effective encoding strategy to increase the probability of obtaining a near-optimal encoding result.

Our strategy uses the soft quantization technique and generates Q candidate codes for each input vector. The code that minimizes the residual $\mathbf{e}_N$ will be selected from the Q candidate codes as the final code of the input vector. Specifically, in the first-stage quantization process, we quantize the PCA-compressed input vector to the first Q nearest the cluster centroids of the first-stage codebook CP_1. Then, we generate Q residual vectors by using Equation 8 for the second-stage quantization.

In the second stage, each PCA-compressed residual vector generated in the first stage is quantized to the first Q nearest cluster centroids of the second-stage codebook CP_2. We can then obtain residual vectors that will be used in the third-stage quantization process. However, to reduce the computational complexity, we do not use all the Q^2 residual vectors. Only the top Q smallest residual vectors will be quantized in the third-stage quantization process.

This process is repeated in each of the remaining stages and only the top Q smallest residual vectors generated from the previous stage will be quantized in each stage. The code corresponding to the smallest residual vector generated in the final stage will be selected as the final code of the input vector.

Results

After introducing the datasets used in our experiments, we can analyze the performance of the proposed PRVQ algorithm. We compared our approach with three state-of-the-art methods: product quantization (PQ),[10] transform coding (TC),[15] and residual vector quantization (RVQ).[16]

Table 1. Dataset parameters.

	Dataset	
Parameter	**GIST**	**VLAD**
Descriptor dimensionality	960	4,096
Number of database vectors	1,000,000	1,295,000
Number of training vectors	500,000	200,000
Number of query vectors	1,000	849

Datasets

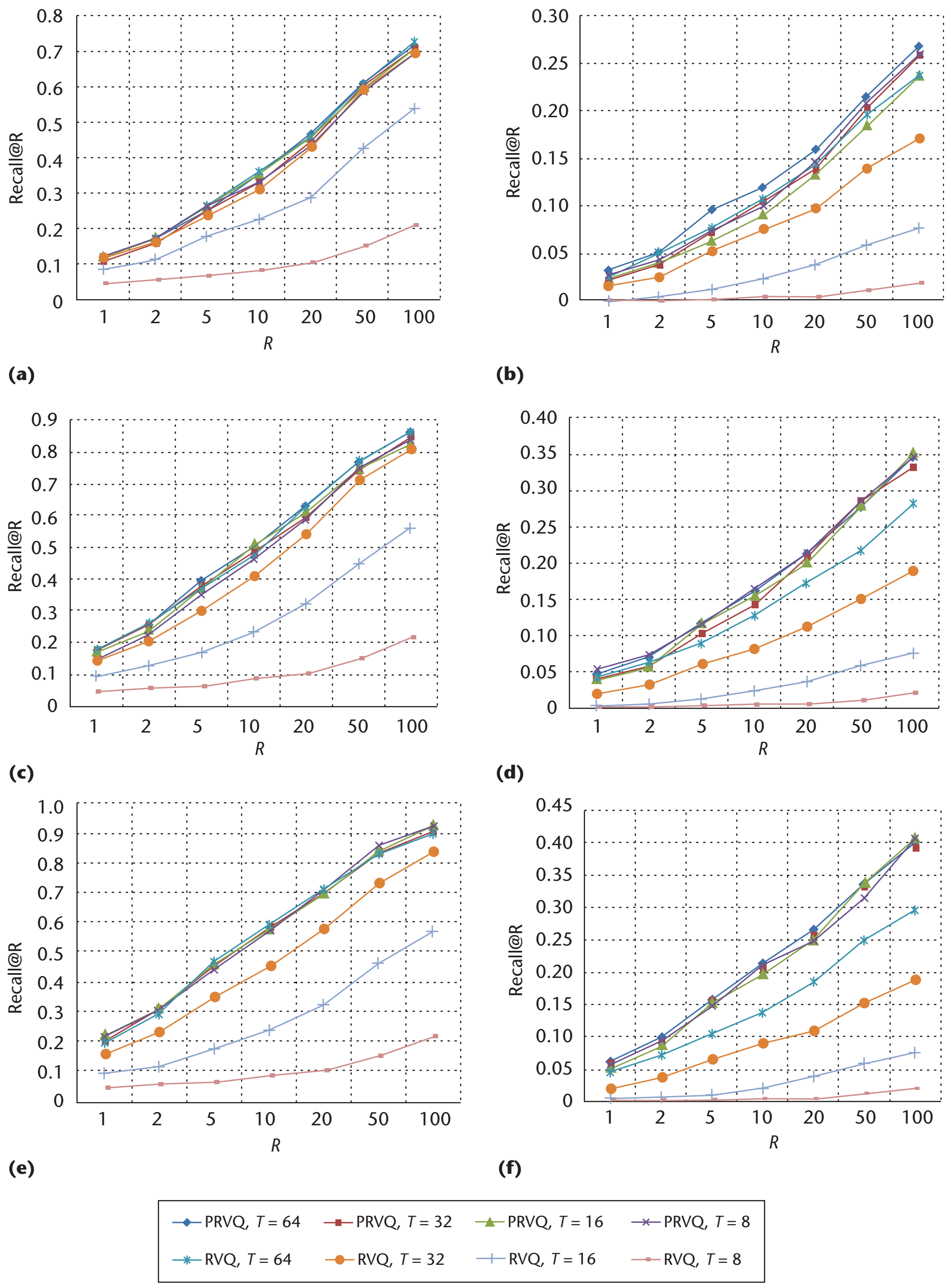
To evaluate the performance of our method, we used the GIST and VLAD datasets. Table 1 gives the parameters of each dataset. We used the training sets for codebook learning and the query set for performance evaluation. (More details about the GIST datasets are available elsewhere.[10])

The VLAD dataset is generated from the Wuhan street view database,[5] which contains 1.295 million street view images and 849 test queries. We used 1 million 64-dimensional SURF (speeded up robust features)[18] features detected from the database images to train (by K-means) 64 visual words for VLAD generation purposes. We randomly selected 0.2 million VLAD descriptors to form the training set. The ground truth was obtained using exact NN search. (Our VLAD dataset is freely available upon request.)

PRVQ Performance

Our first experiment tested the performance of our PRVQ algorithm. We changed the parameter T when using PRVQ to search different datasets. (We did not use the optimization methods

Figure 3. Performance of the proposed PRVQ algorithm. Results for the GIST dataset when (a) 8, (b) 12, and (c) 16 bytes are used. Results for the VLAD dataset when (d) 8, (e) 12, and (f) 16 bytes are used.



discussed earlier in this experiment.) For comparison, we also tested the performance of RVQ on the PCA-compressed vectors.

The size of stage codebook was set to 256 (K). The K-means algorithm provided in the Yael package (https://gforge.inria.fr/projects/yael/) was used to train the codebooks needed in different methods. We followed the exhaustive search strategy[10,16] and measured the performance of different algorithms by recall@R, which is defined as the ratio of query vectors for which the correct match is ranked within the top R returned results.

For the two databases used, Figure 3 shows the results, from which we can draw several conclusions. First, compared with the RVQ

*Table 2. Experiments used to test the effectiveness of the proposed optimization methods.**

Experiment name	Method	Description
MAP-PQ	Use PQ to test MAP	Divide the original vector space on average into 8 subspaces and use the same number of bytes to encode different subvectors
MAP-RVQ	Use RVQ to test MAP	Encode original vectors and assign 1 byte to each stage for quantization and encoding use
MAP-TC	Use TC to test MAP	Encode the PCA-compressed vectors and assign the number of bits for each principal component according to its variance
MAP-PRVQ-O1	Use PRVQ to test MAP	Optimize the parameter *T* with fixed *K* and *N*
MAP-PRVQ-O2	Use PRVQ to test MAP	Optimize the stage quantizer and PCA matrix with fixed *T*, *K*, and *N*
MAP-PRVQ-O3-2	Use PRVQ to test MAP	Optimize the encoding process by using the optimized stage quantizer and PCA matrix, *Q* is set to 2
MAP-PRVQ-O3-4	Use PRVQ to test MAP	Optimize the encoding process by using the optimized stage quantizer and PCA matrix, *Q* is set to 4
MAP-PRVQ-O3-8	Use PRVQ to test MAP	Optimize the encoding process by using the optimized stage quantizer and PCA matrix, *Q* is set to 8
Error-PRVQ-O1	Record the error of PRVQ	Record *E* (obtained from database vectors) generated in the experiment MAP-PRVQ-O1
Error-PRVQ-O2	Record the error of PRVQ	Record *E* (obtained from database vectors) generated in the experiment MAP-PRVQ-O2
Error-PRVQ-O3-2	Record the error of PRVQ	Record *E* (obtained from database vectors) generated in the experiment MAP-PRVQ-O3-2
Error-PRVQ-O3-4	Record the error of PRVQ	Record *E* (obtained from database vectors) generated in the experiment MAP-PRVQ-O3-4
Error-PRVQ-O3-8	Record the error of PRVQ	Record *E* (obtained from database vectors) generated in the experiment MAP-PRVQ-O3-8

* For encoding purpose, 8 bytes were used in different methods.

method (which directly uses the PCA-compressed vectors), our PRVQ method (which considers the projection errors) provides more accurate results, especially when we use extremely low-dimensional PCA-compressed vectors. Second, by considering projection errors, our PRVQ algorithm is more stable than the original RVQ method when we use different values of *T*. These results prove the effectiveness of the proposed PRVQ algorithm.

Performance of the Proposed Optimization Methods

We also performed several experiments to test the effectiveness of the proposed optimization methods, implementing the PQ, TC, and RVQ methods for comparison purposes. For PQ, we divided the original vector space into an average of eight subspaces and used the same number of bytes to encode different subvectors. For TC, we encoded the PCA-compressed vectors and assigned the number of bits for each principal component according to its variance. For RVQ, we used original vectors and assigned 1 byte to each stage for quantization and encoding purpose.

When using our optimization methods, we tested the following three optimization strategies:

- Optimize the parameter *T* with fixed *K* and *N*.
- Optimize the stage quantizer and PCA matrix with fixed *T*, *K*, and *N*. We performed several rounds of the optimization process until the change in the value of *E* between two rounds was less than 0.1 percent.
- Optimize the encoding process by using the optimized stage quantizer and PCA matrix.

For the sake of clarity, Table 2 gives a detailed description of the performed experiments. The exhaustive search strategy was used in different methods, and the performance was measured by mean average precision (MAP). Based on the results in Figure 4, we can draw the following conclusions:

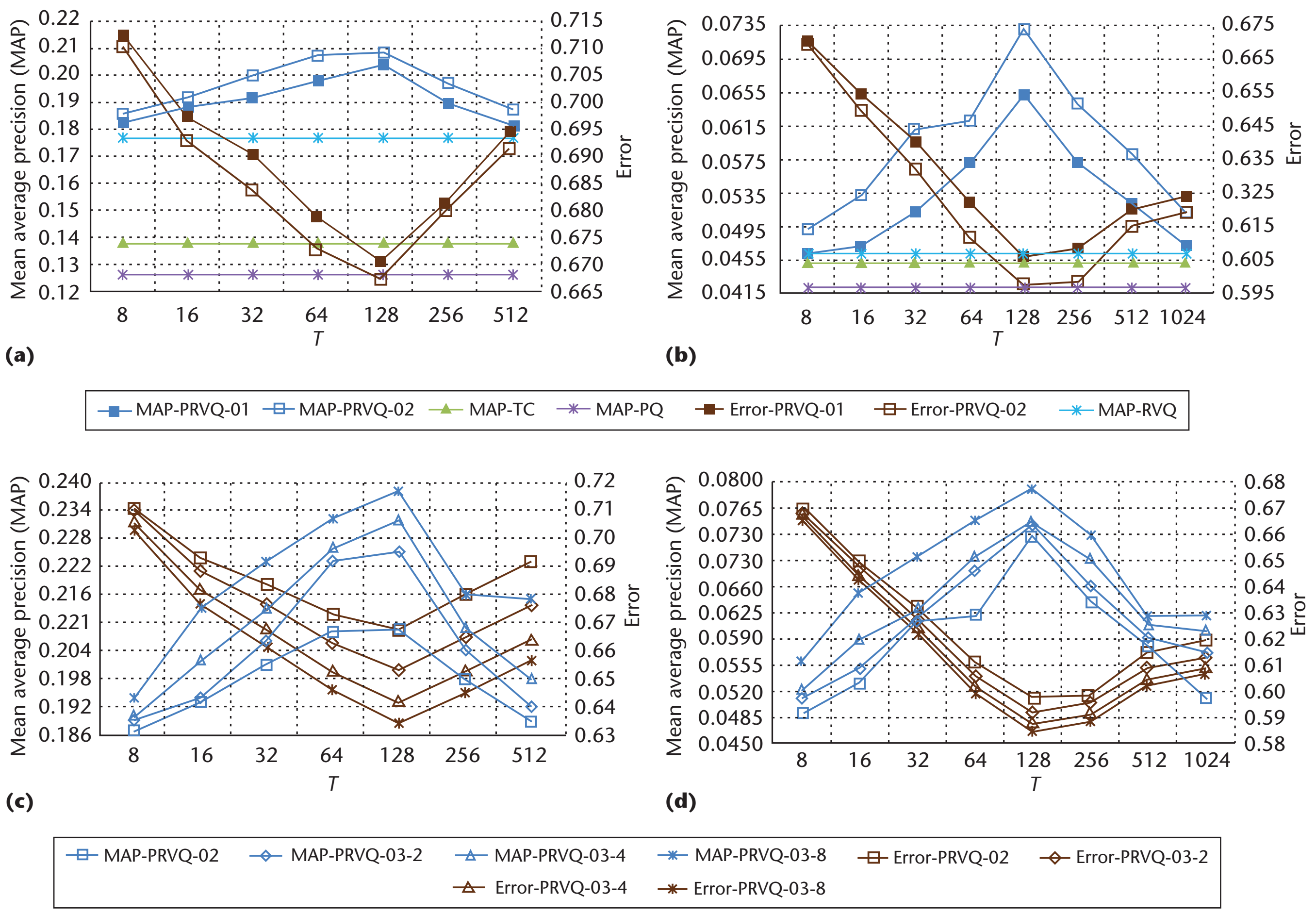


Figure 4. Performance of the proposed optimization methods. (a) Results for the GIST dataset. (b) Results for the VLAD dataset. For encoding purposes, we used 8 bytes for the different methods.

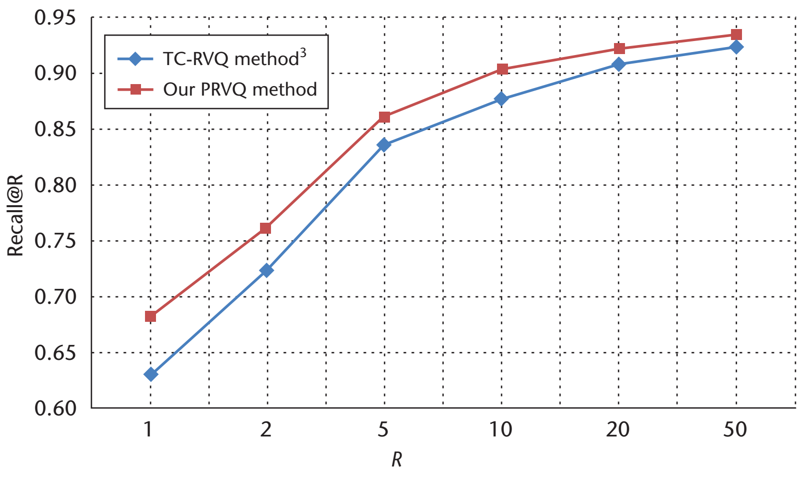


Figure 5. Landmark recognition accuracy of the TC-RVQ method and our PRVQ method.

- From experiments MAP-PRVQ-O1 and Error-PRVQ-O1 (Figures 4a and 4b), we can see that by minimizing *E* we can always obtain the best *T*, which can provide the best search precision.
- From experiments MAP-PRVQ-O2 and Error-PRVQ-O2 (Figures 4a and 4b), we can see that optimizing the stage quantizer and PCA matrix can reduce *E* and improve search accuracy.
- From experiments MAP-PQ, MAP-RVQ, and MAP-TC (Figures 4a and 4b), we can see that even the nonoptimized PRVQ algorithm can provide more accurate results than the PQ, TC, and RVQ methods.
- From Figures 4c and 4d, we can see that the search precision can be further improved by optimizing the encoding process.

Mobile Landmark Recognition Application

We also tested the performance of the proposed PRVQ algorithm by integrating it into the

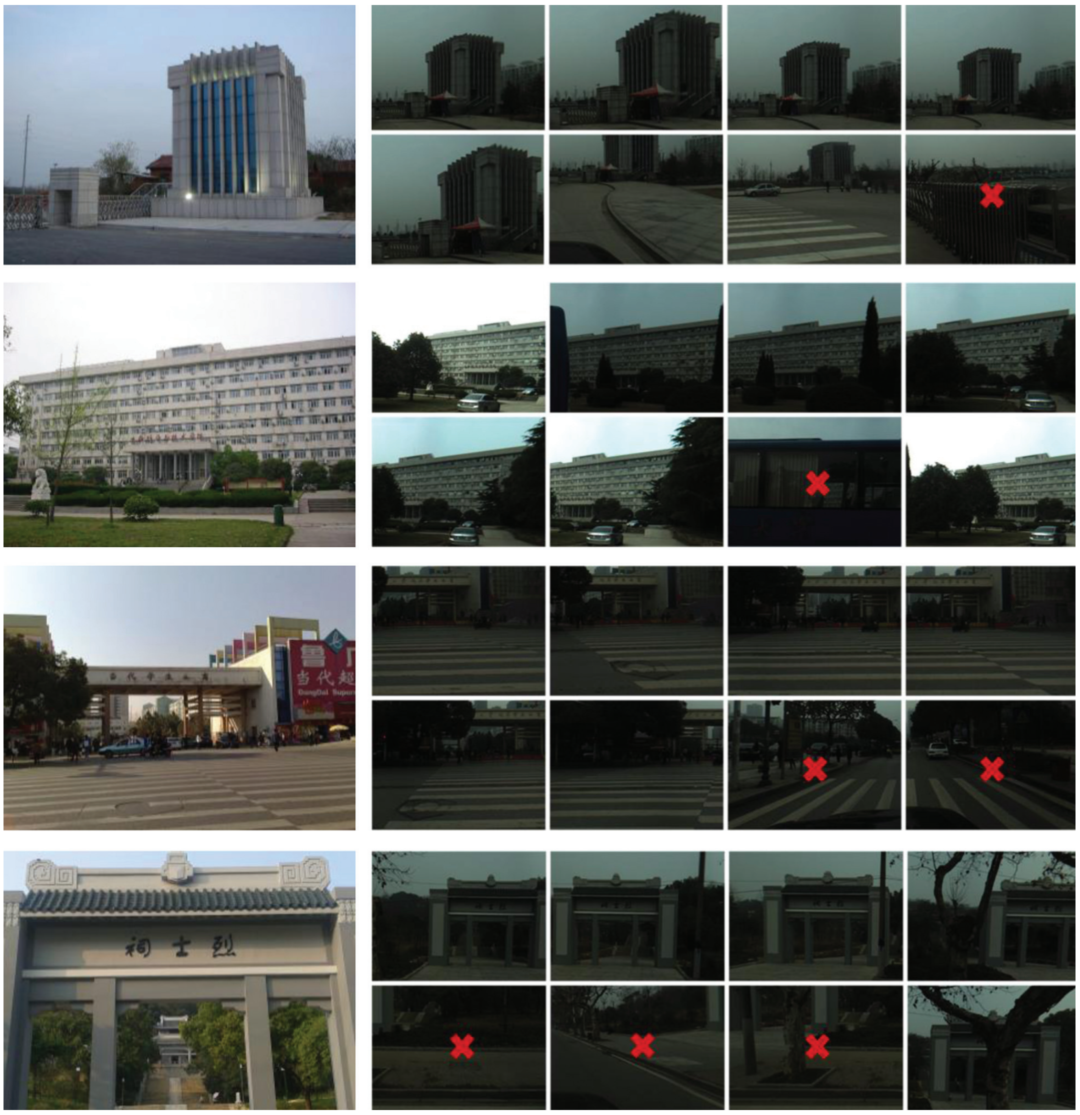


Figure 6. Landmark recognition results. Each row gives one group of recognition results. In each group, the query image is shown on the left, and each line on the right corresponds to an approach. The top line in each row is our method; the bottom line in each row is the TC-RVQ method.[5] *Incorrect results are marked with red semitransparent tags.*

mobile landmark recognition framework introduced in earlier work.[5] We used the Wuhan street view database[5] and divided the workspace into 64 regions according to the GPS information attached to each database image. The VLAD descriptor of each database image was encoded into 10 bytes using our PRVQ method.

We set $T = 8$ and $Q = 2$ and performed several rounds of the optimization method we described earlier until the change in the value of E between two rounds was less than 0.1 percent.

In the search stage, we first used the GPS information to find the four nearest regions and then used the GPS and visual fusion method[5] to fulfill the landmark recognition task. We also implemented the TC-RVQ algorithm[5] for comparison purposes. We did this by encoding each VLAD descriptor into 10 bytes and used the same method to perform landmark recognition.

Figures 5 and 6 show that our PRVQ method can provide better recognition performance than the TC-RVQ method,[5] which proves the effectiveness of our PRVQ algorithm in dealing with the mobile landmark recognition problem.

Search Time

Lastly, we measured the search time for the different methods using the GIST dataset. The search time indicates the time used to process a single query. We used the fast PCA algorithm to reduce dimensionality and measured the time on a PC with a Core2Duo 2.2-GHz CPU and 3-GByte memory. All the methods used 8 bytes to encode a single vector.

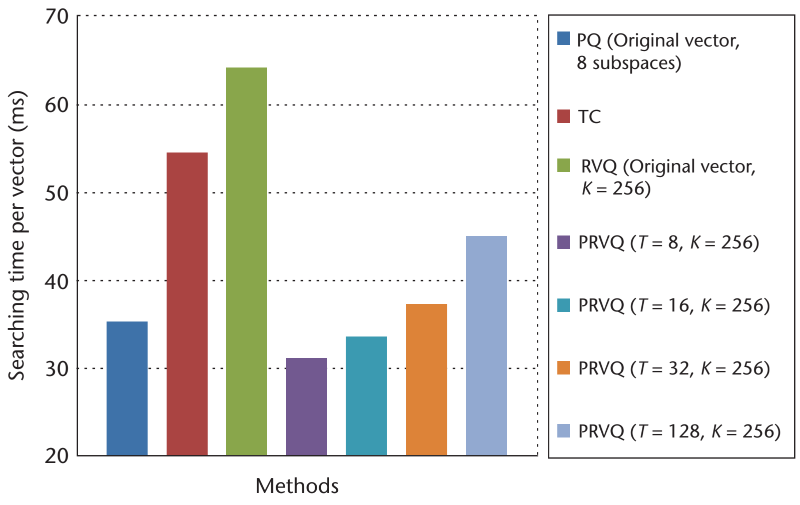


Figure 7. Search time for different methods on the GIST dataset. The search time indicates the time used to process a single query.

Figure 7 shows the search time for the different methods. We can see that the search time of PRVQ is comparable to that of PQ when the value of T is smaller than 32. From Figure 4a, we can see that even when the value of T is smaller than 32, our PRVQ method can still provide more accurate search results than the other methods.

Discussion and Future Work

To improve the proposed PRVQ algorithm, we can further address several issues in future work. Specifically, in this research, we propose a PRVQ method to improve the performance of RVQ by incorporating the projection errors into the vector quantization process. However, there is no doubt that we can integrate our approach into other kinds of quantization algorithms such as product quantization and transform coding to further improve their performance. For example, to incorporate the projection errors into the transform coding algorithm, we can simply encode the PCA-compressed vector by using the method of transform coding and then reproject the residuals to the original space for the next-stage quantization use. In our future work, we will try to build a unified framework that can be used to enhance different quantization algorithms by considering projection errors.

We optimized the proposed PRVQ algorithm by optimizing the stage quantizer and PCA matrix, and have demonstrated the effectiveness of this method. However, we have not provided the theoretical proof of the convergence of the optimization algorithm. In our future work, we will solve this problem to theoretically complete our optimization method. **MM**

Acknowledgment

This research is supported by the National Natural Science Foundation of China (NSFC) under grant 61272202.

References

1. D.G. Lowe, "Distinctive Image Features from Scale-Invariant Key-Points," *Int'l J. Computer Vision*, vol. 60, no. 2, 2004, pp. 91–110.
2. H. Liu et al., "Gradient Ordinal Signature and Fixed-Point Embedding for Efficient Near-Duplicate Video Detection," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 22, no. 4, 2012, pp. 555–566.
3. R. Ji et al., "Task-Dependent Visual-Codebook Compression," *IEEE Trans. Image Processing*, vol. 21, no. 4, 2012, pp. 2282–2293.
4. Y. Gao et al., "Visual-Textual Joint Relevance Learning for Tag-Based Social Image Search," *IEEE Trans. Image Processing*, vol. 22, no.1, 2013, pp. 363–376.
5. T. Guan et al., "On-Device Mobile Visual Location Recognition by Integrating Vision and Inertial Sensors," *IEEE Trans. Multimedia*, vol. 15, no. 7, 2013, pp. 1688–1699.
6. R. Ji et al., "Location Discriminative Vocabulary Coding for Mobile Landmark Search," *Int'l J. Computer Vision*, vol. 96, no. 3, 2012, pp. 290–314.
7. R. Ji et al., "Learning to Distribute Vocabulary Indexing for Scalable Visual Search," *IEEE Trans. Multimedia*, vol. 15, no. 1, 2013, pp. 153–166.
8. K. Beyer et al., "When Is 'Nearest Neighbor' Meaningful?" *Proc. Int'l Conf. Database Theory*, 1999, pp. 217–235.
9. C. Bohm, S. Berchtold, and D. Keim, "Searching in High-Dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases," *ACM Computing Surveys*, vol. 33, Oct. 2001, pp. 322–373.
10. H. Jegou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, 2011, pp. 117–128.
11. M. Muja and D.G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," *Proc. Int'l Conf. Computer Vision Theory and Applications*, 2009, pp. 331–340.

12. D. Nister and H. Stewenius, “Scalable Recognition with a Vocabulary Tree,” *Proc. 2006 IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition* (CVPR), 2006, pp. 2161–2168.
13. M. Muja and D.G. Lowe, “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration,” *Proc. Int’l Conf. Computer Vision Theory and Applications*, 2009, pp. 331–340.
14. M. Datar et al., “Locality-Sensitive Hashing Scheme Based on *p*-Stable Distributions,” *Proc. 20th Ann. Symp. Computational Geometry*, 2004, pp. 253–262.
15. J. Brandt, “Transform Coding for Fast Approximate Nearest Neighbor Search in High Dimensions,” *Proc. Int’l Conf. Computer Vision and Pattern Recognition* (CVPR), 2010, pp. 1815–1822.
16. Y. Chen, T. Guan, and C. Wang, “Approximate Nearest Neighbor Search by Residual Vector Quantization,” *Sensors*, vol. 10, no. 12, 2010, pp. 11259–11273.
17. H. Jegou et al., “Aggregating Local Image Descriptors into Compact Codes,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, 2012, pp. 1704–1716.
18. H. Bay et al., “SURF: Speeded Up Robust Features,” *Computer Vision and Image Understanding* (CVIU), vol. 110, no. 3, 2008, pp. 346–359.

Benchang Wei is a doctoral student in the School of Computer Science and Technology at the Huazhong University of Science and Technology, China. His research interests include mobile visual search. Wei has an MS in application technology from Shanghai Maritime University. Contact him at yyszcs04@sina.com.cn.

Tao Guan (corresponding author) is an associate professor in the School of Computer Science and Technology at the Huazhong University of Science and Technology, China. His research interests include mobile visual search and mobile augmented reality. Guan has a PhD from the Huazhong University of Science and Technology. Contact him at qd_gt@126.com.

Junqing Yu is a professor in the School of Computer Science and Technology at the Huazhong University of Science and Technology. His research interests include digital media processing and retrieval as well as multicore programming environments. Yu has a PhD in computer application technology from Wuhan University. He is a member of IEEE and ACM. Contact him at yjqing@mail.hust.edu.cn.