Multiple Stage Residual Model for Image Classification and Vector Compression

Song Bai, Xiang Bai, *Senior Member, IEEE*, and Wenyu Liu, *Senior Member, IEEE*

***Abstract*—Feature coding is a fundamental issue with many vision tasks, such as image classification, image retrieval and image segmentation, etc. There is no doubt that the encoding procedure leads to information loss, due to the existence of quantization error. The residual vector, defined as the difference between the feature and its corresponding visual word, is the chief culprit to be responsible for the quantization error. Many previous algorithms consider it as a coding issue, and focus on reducing the quantization error by reconstructing the feature with more than one visual word, or by the so-called soft-assignment strategy. In this paper, we consider the problem from a different point of view, and propose an effective and efficient model called multiple stage residual model (MSRM). It makes full use of the residual vector to generate a multiple stage code. MSRM is a hierarchical structure, with the bottom stage producing the coarsest quantization, and the top stage producing the finest quantization. Moreover, our proposed model is a generic framework, which can be built upon many coding algorithms. The interplay of such a coarse-to-fine quantization procedure with a discriminative classifier (e.g., SVM) can improve the classification accuracy of the baseline algorithms significantly. As a special case of MSRM, multiple stage vector quantization (MSVQ) can be directly used for vector compression and approximate nearest neighbor search, and achieves competitive performances with high efficiency.**

***Index Terms*—Approximate nearest neighbor (ANN) search, image classification, residual vector, shape recognition, vector compression.**

I. Introduction

FEATURE coding is a fundamental and important procedure in computer vision with many applications, such as image classification [1], image retrieval [2]–[4], video retrieval [5]–[7], 3D model recognition [8], vector compression [9], web content analysis [10] and remote sensing [11], [12]. Given a codebook learned on the training set off-line, feature coding algorithms allow each feature vector to activate several codewords, and the amplitudes of responses with regard to those activated codewords are taken as its code. Those codes are later manipulated with some specific operations (e.g., sum-pooling or max-pooling), or organized using some advanced indexing strategies (e.g., inverted index) for specific tasks.

Manuscript received January 13, 2016; accepted April 03, 2016. Date of publication April 20, 2016; date of current version June 15, 2016. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 6122308 and Grant 61573160. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Wolfgang Hurst.

The authors are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: songbai@hust.edu.cn; xbai@hust.edu.cn; liuwy@hust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2557071

The primary difference of feature coding algorithms is the way of activating the codewords, i.e., which codewords are chosen to activate and how to determine the activation values. In recent years, many coding algorithms [13]–[26] have been proposed. One of the most curial criterions to design feature coding algorithms is reconstruction error. It is known that low reconstruction error leads to less information loss, thus giving better performances. For example, as a representative algorithm, Hard-assignment Coding [13] only counts the nearest visual word of an input feature, resulting in severe quantization error and inferior performances. Some improvements on HC are developed to reduce the reconstruction error by constructing the feature with more than one visual word (e.g., Sparse Coding [22]), or by the so-called soft-assignment strategy (e.g., kernel codebook learning [21]).

Almost all the aforementioned algorithms consider reducing the quantization error, caused by the residual vector (the difference between the feature and its corresponding visual word), as a feature coding issue. In this paper, we propose a generic model called Multiple Stage Residual Model (MSRM) to make full use of the residual vector instead, and claim that the discriminative power of residual vectors is ignored by those feature coding algorithms more or less. MSRM is a hierarchical framework with a coarse-to-fine quantization strategy, with the bottom stage producing the coarsest quantization and the top stage producing the finest quantization. It consists of several concatenated sub-codebooks. The output of each stage are the residual vectors, which serve as the input of the next stage. As a result, the final output of MSRM is a multiple stage code, with different stages in different quantization resolutions. We experimentally prove that our proposed model leads to less information loss compared with HC [13], and can achieve a better performance. Moreover, we manage to separate the quantization procedure from the feature coding procedure, making MSRM a generic framework by embedding various state-of-the-art coding algorithms [13], [14], [18], [20], [27]. In this sense, the aim of our paper is not to propose a new feature coding algorithm. Instead, it provides a generic tool that can be helpful to improve the performances of many feature coding algorithms further.

MSRM is evaluated with different visual tasks requiring feature coding algorithms. In this paper, two primary types of experiments are considered. In image classification task, we incorporate Hard-assignment Coding [13], Locality-constrained Linear Coding (LLC) [14], Salient Coding (SC) [20], Vector of Local Aggregated Descriptors (VLAD) [27] and Improved Fisher Kernel (IFK) [18] respectively into our proposed generic model. The codes of local descriptors (e.g., SIFT [28]) generated by MSRM are pooled together to get an equal-sized feature

vector for each image. We observe the complementarity of the codes from different stages, and discriminative classifiers, such as SVM, can be used to select several representative stages to achieve a higher classification accuracy.

For vector compression task, we restrict the coding algorithms used in MSRM as HC, since the codes of HC are either 0 or 1, which can be stored in the format of integer. By doing so, each feature is compressed and stored with a few bytes. We also find that with such a hierarchical coding model, each input vector can be represented by the sum of several codewords. As a result, approximate nearest neighbor (ANN) search with limited memory and time cost is feasible. We compare it with several state-of-the-art algorithms (e.g., product quantization (PQ) [9], additive quantization (AQ) [29]). The experimental results demonstrate the superiority of MSRM in terms of both speed and accuracy.

A preliminary version of this paper appeared in [30]. Compared with [30], extensions are made in several aspects listed as follows.

1) We prove, both theoretically and experimentally, that our proposed MSRM is well suitable for vector compression and approximate nearest neighbor search. The theoretical proof is introduced in Section III-D, and the performance comparison with other representative algorithms is given in Section IV-B.
2) We extend our proposed method to shape recognition task, and state-of-the-art performances are achieved on MPEG-7 dataset [31].
3) A more comprehensive description is presented in sections of "introduction" and "related work".

The rest of this paper is organized as follows. In Section II, we introduce some related methods briefly. The introduction of MSRM is given in Section III. In Section IV, we carry out comprehensive experiments on different tasks. Conclusions are drawn in Section V.

II. Related Work

In this section, we give a brief review about some representative algorithms in feature encoding and vector compression, which have close relationships with the proposed method.

A. Encoding Strategies

MSRM is a generic model, in which many encoding methods can be embedded. Here we give a detailed introduction to those coding strategies which are plunged successfully into MSRM.

Based on the classification standard for coding methods in [1], these methods are grouped into five categories: voting-based methods, reconstruction-based methods, saliency-based methods, local tangent-based methods and fisher coding-based methods. Let $\mathcal{C} = [c_1, c_2, \ldots, c_N] \in \mathbb{R}^{d\times N}$ denote the codebook learned previously in the training set, where N represents the codebook size and d represents the feature dimension. Assume $x \in \mathbb{R}^d$ denotes the feature to be encoded. Let $f = \varphi(x)$ be the code of x and f_i be the response value of x with respect to c_i.

Hard-assignment coding. Hard-assignment coding [13] (also known as vector quantization (VQ)) is a representative of voting-based methods. When encoding the feature x, HC assigns it to the nearest visual word in the codebook under a certain metric. It means there is only one non-zero element in f. f_i satisfies

$$f_i = \begin{cases} 1, & \text{if } i = \underset{i'=1,2,\ldots,N}{\arg\min} \|x - c_{i'}\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

when $\mathcal{L}_2$ metric is used.

Locality-constrained linear coding. Locality-constrained Linear Coding [14] is a typical example of reconstruction-based methods. Unlike traditional sparse coding methods, LLC emphasizes the importance of locality instead of sparsity, since locality must lead to sparsity but not necessary vice versa. Specifically, LLC solves the following optimization:

$$\begin{aligned} &\min \|x - \mathcal{C}f\|_2^2 + \lambda \|d \odot f\|_2^2 \\ &\text{s.t. } \mathbf{1}^{\mathrm{T}} f = 1. \end{aligned} \tag{2}$$

$d \in \mathbb{R}^N$ is the locality adaptor defined as

$$d = \exp\left(\frac{dist(x, \mathcal{C})}{\sigma}\right) \tag{3}$$

where $dist(x, \mathcal{C})$ is the Euclidean distance between x and each visual word $c_i \in \mathcal{C}$. σ is used to adjust the weight decay speed for the locality adaptor. An approximated solution of LLC is proposed in [14] to improve the computational efficiency.

Salient coding. Salient Coding [20] is a representative method of saliency-based methods. SC deems that saliency is a fundamental property in coding, and defines a "saliency" degree based on the nearest visual word of x. Assume $\{c_{i'}\}_{i'=1,\ldots,k}$ is the set of k-nearest visual words to x, then

$$f_i = \begin{cases} \Phi(x, c_i), & \text{if } i = \underset{i'=1,2,\ldots,N}{\arg\min} \|x - c_{i'}\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\Phi(\cdot)$ is a monotonically decreasing function, usually defined as

$$\Phi(x, c_i) = 1 - \frac{\|x - c_i\|_2}{\frac{1}{k-1}\sum_{i'\neq i}^{k} \|x - c_{i'}\|_2}. \tag{5}$$

Vector of local aggregated descriptors. Vector of Local Aggregated Descriptors [27], [32] aggregates the features based on a locality criterion in the feature space, which is defined as

$$f_i = \begin{cases} x - c_i, & \text{if } i = \underset{i'=1,2,\ldots,N}{\arg\min} \|x - c_{i'}\|_2^2 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Note that f_i is a vector in d dimension.

A very similar idea can be found in [19], where Super Vector Coding (SVC), a representative of local tangent-based methods, is proposed. SVC considers feature coding as a manifold approximation problem using the visual words by assuming that all features constitute a smooth manifold. Slightly different from VLAD, SVC adds an extra scalar constant into f and cross-validation on the training set is needed to determine the value of the scalar. Considering the simplicity of VLAD, we adopt

VLAD to get a compact representation for images throughout our experiments.

Improved fisher kernel. Improved Fisher Kernel [18] is a representative of fisher coding-based methods. In IFK, the probability density distribution of the features is described by the Gaussian mixture models (GMM). Assume $\theta_i = \{w_i, \mu_i, \Sigma_i\}$ denotes the parameters, i.e., the weight, the mean vector and the covariance matrix, of the ith Gaussian distribution, then

$$\begin{aligned} f_i &= [\mathcal{G}_{u,i}, \mathcal{G}_{\Sigma,i}] \\ \mathcal{G}_{u,i} &= r_i \Sigma_i^{-\frac{1}{2}} (x - \mu_i)/\sqrt{w_i} \\ \mathcal{G}_{\Sigma,i} &= r_i \left((x - \mu_i)\Sigma_i^{-1}(x - \mu_i) \right)/\sqrt{2w_i} \\ r_i &= \frac{w_i p(x|\theta_i)}{\sum_{i'=1}^{N} w_{i'} p(x|\theta_{i'})} \end{aligned} \tag{7}$$

where $p(x|\theta_i)$ describes the probability estimation of x to the ith Gaussian.

B. Vector Compression Strategies

Based on the universal codebook, encoding strategies offer a way to represent features with codes. When the code f only contains 0 or 1, we can compress floating-point features into only a few bytes. In this case, among all the encoding strategies in Section II-A, only vector quantization defined in (1) can be applied to vector compression task.

However, as suggested in [9], directly using vector quantization to learn the compression model on large scale data is infeasible in practice. To this end, product quantization [9] decomposes the input vector into several sub-vectors of equal length, and all the sub-vectors are quantized separately using standard vector quantization. In [33], an optimized version of product quantization (OPQ), called Cartesian K-Means, is proposed.

Different from PQ and OPQ that work on sub-vectors, Additive Quantization [29] uses the sum of multiple codewords, each from one sub-codebook, to approximate the input vector. A similar idea can be found in [34]. AQ achieves better performances than PQ, but its optimization in the procedure of codebook construction and encoding is shown to be time-consuming. The proposed MSRM shares a similar principle with AQ, but the solution of MSRM is more accurate and efficient as demonstrated in the experiments.

III. Multiple Stage Residual Model

In this section, a brief introduction to the preliminary knowledge is given firstly in Section III-A, where MSRM are derived. In Section III-B, we present the basic idea of MSRM and show how to build image-level representations using MSRM by incorporating various coding strategies. Section III-C exhibits the corresponding codebook learning algorithm. At last, MSRM is proven suitable for vector compression theoretically in Section III-D.

A. Preliminary

Multiple Stage Vector Quantization (MSVQ) [35] is a classic channel coding algorithm commonly used in Digital Voice Processing.

The theory of MSVQ is as follows: 1) consider an input signal represented by a vector $x \in \mathbb{R}^d$ and the multiple stage codebook $\mathcal{C} = \{\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^M\}$, where M is the number of stages. Each component $\mathcal{C}^j (1 \leq j \leq M)$ is the sub-codebook in the jth stage of $\mathcal{C}$ with N entries $\{c_1^j, c_2^j, \ldots, c_N^j\}$; 2) denote x as x^1 for notation convenience. When x^1 is fed into the first stage $\mathcal{C}^1$, the codeword c_i^1 with the minimum distortion is determined, and the subscript i, as well as the stage number 1, is passed into channel; 3) the input of the next stage, i.e., the second stage, is the residual vector $x^2 = x^1 - c_i^1$. Following the same principle, c_i^2 is determined again; 4) the procedure of 2), 3) is iteratively conducted, until reaching the final stage; 5) in the receiving terminal, the decoder reconstructs the signal by using the subscripts and the multiple stage codebook.

MSVQ is developed to reduce computation complexity via replacing a single codebook with several cascaded and much smaller codebooks. The key characteristic of MSVQ is quantization on residual vectors. In this paper, we give a thorough study about such a property with applications to image classification, vector compression and approximate nearest neighbor search.

Indeed, many advanced encoding methods can be used despite vector quantization. Based on MSVQ, we propose a generic model called Multiple Stage Residual Model, in which many state-of-the-art encoding methods can be embedded. Hence, MSVQ becomes a special case of MSRM when the encoder is vector quantization.

B. Extension to Generic Model

Although vector quantization is an efficient coding method both in speed and memory usage, it leads to severe quantization error and achieves inferior performances in many tasks.

As presented in Section II, various advanced coding strategies are proposed. In order to embed these coding strategies with our proposed model, we separate the quantization procedure from the feature coding procedure. Specifically, for a given feature x, on the one hand we only consider the nearest visual word to compute the residual vector and pass the residual vector to the next stage, which is the new feature vector to be encoded in the next stage. On the other hand, we do not restrict the way and the number of the visual words used to generate the code for x, which is usually determined by the encoder function φ. The pseudocode of multiple stage coding procedure is presented in Algorithm 1.

For example, if LLC [14] is chosen as the encoder φ of MSRM, we use k (k is usually set to 5) visual words to encode x, and use only the nearest visual word to generate the residual vector. Our interpretation is that the nearest visual word to x captures its main pattern, and all the local image descriptors lying in the same cluster will eliminate the information redundancy if all of them are deprived with their common pattern. The operations of computing the residual vector and encoding the

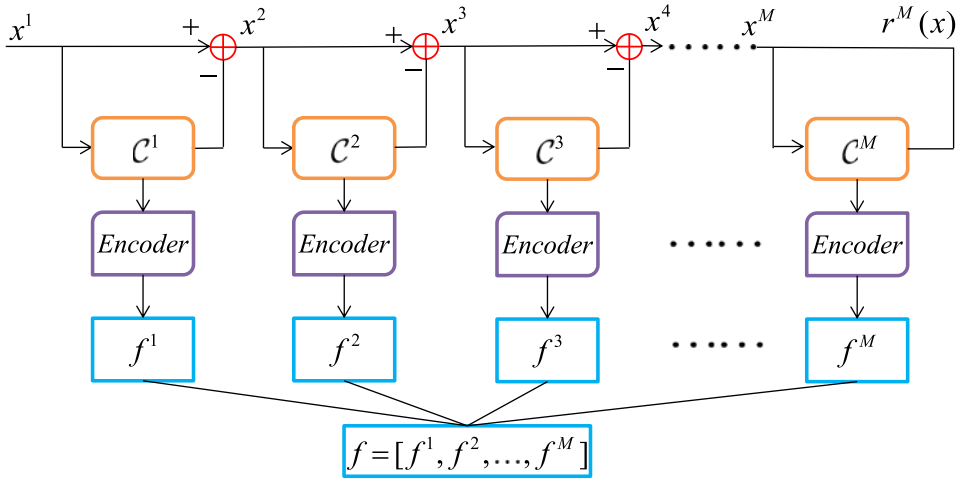

Fig. 1. Pipeline of MSRM. The input $x^1 = x$ is the feature to be encoded. $\mathcal{C}^j$ $(1 \le j \le M)$ is the jth sub-codebook. The encoder function φ can vary. f^j $(1 \le j \le M)$ is the encoded feature via the encoder. The final representation of x is the concatenation of the output from all stages.

Algorithm 1: Multi-stage Encoding.

Input: The feature x to be encoded; the encoder φ; the multi-stage codebook $\mathcal{C} = \{\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^M\}$, where $\mathcal{C}^j = \{c_1^j, c_2^j, \ldots, c_N^j\}$.

Output: The code $f = [f^1, f^2, \ldots, f^M]$ of x.

1: initialize $x^1 = x$;
2: **for** each $j \in [1, M]$ **do**
3: use $\mathcal{C}^j$ for coding: $f^j = \varphi(x^j)$;
4: quantization of x^j: $i = \arg\min_{i'} \|x^j - c_{i'}^j\|^2$;
5: compute the residual vector: $x^{j+1} = x^j - c_i^j$;
6: **end for**
7: concatenate $f = [f^1, f^2, \ldots, f^M]$;

feature are iteratively conducted until the final stage is reached. The pipeline of Multiple Stage Residual Model is illustrated in Fig. 1.

According to the taxonomy presented in [1], we introduce some typical coding strategies as the encoder function φ to our proposed model, including Hard-assignment Coding [13], Local-constrained Linear Coding [14], Salient Coding [20], Vector of Local Aggregated Descriptors [27] and Improve Fisher Kernel [18].

Two noteworthy comments on IFK should be made here. When IFK is selected as the encoder φ of MSRM. 1) The codebook used for encoding the features are GMM, learned using expectation maximization (EM), i.e., $\theta_i^j = \{w_i^j, \mu_i^j, \Sigma_i^j\}, 1 \le j \le M, 1 \le i \le N$. 2) To compute the residual vector for the input x^j in the jth stage, only the mean vector μ_i^j of the Gaussian distribution with the largest $p(x^j|\theta_i^j)$ is considered. Specifically, $x^{j+1} = x^j - \mu_i^j$ and $i = \arg\max_{i'} p(x^j|\theta_{i'}^j)$.

For image classification task, all the generated codes of local descriptors from a certain image are pooled together (max-pooling or sum-pooling) to get the final representation of the image. As shown in Section IV, the codes from different stages are greatly complementary to each other. If the classifier cannot give the image a right label prediction with the codes from a certain stage, the prediction can be revised with the usage of the codes from other stages in most cases.

C. Codebook Learning

Codebook learning is a necessary step before encoding the features. There are various codebook learning algorithms in unsupervised way [36], weakly-supervised way [37], or supervised way [38], [39].

Among all the codebook learning algorithms, K-means is probably the most widely used one for its simplicity and stableness. Given a randomly selected subset $\mathcal{X}$ of training features and the codebook size N, K-means seeks N vectors $\mathcal{C} = [c_1, c_2, \ldots, c_N]$ iteratively, and minimizes the approximation error $\mathcal{E}$ defined as

$$\mathcal{E} = \sum_{x \in \mathcal{X}} \|x - c_i\|^2 \tag{8}$$

$$x \to c_i = \arg\min_{c \in \mathcal{C}} \|x - c\|^2. \tag{9}$$

Our proposed model adopts k-means in a greedy manner to learn a multiple stage codebook $\mathcal{C} = \{\mathcal{C}^j, 1 \le j \le M\}$, i.e., we minimize the approximation error and do clustering process stage by stage. Only when the clustering in jth stage ends, the clustering in $(j+1)$th stage begins. It indicates that the multiple stage codebook is not trained independently, instead, the training of latter stage requires the output of the former stage as its input. The pseudocode of multiple stage codebook construction is presented in Algorithm 2.

For the sake of convergence efficiency of K-means, we use triangular inequalities to reduce the operation of sample-to-center comparisons if possible. In practice, for each stage we run K-means three times, and the generated cluster centers with the minimum approximation error are chosen as the codebook.

Note that the codebook learning is different when IFK is chosen as the encoder of MSRM, since the codebook is GMM.

Algorithm 2: Multi-stage Codebook Learning with K-means.

Input: The traning features $\mathcal{X}$ for codebook learning; The codebook size N; The number of stage M.

Output: The learned multi-stage codebook $\mathcal{C} = \{\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^M\}$.

```
initialize X^1 = X;
for each j ∈ [1, M] do
    do clustering on X^j through Eq. (8) and Eq. (9);
    and output the cluster centers C^j;
    for each x^j ∈ X^j do
        i = arg min_{i'} ||x^j − c^j_{i'}||^2;
        x^{j+1} = x^j − c^j_i;
    end for
end for
```

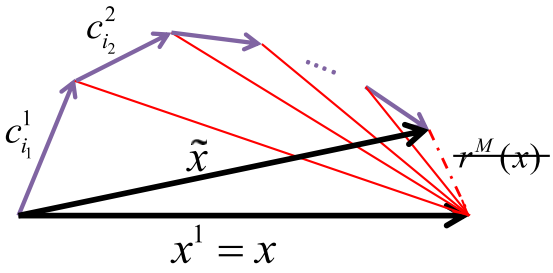

Fig. 2. Illustration of approximation. By removing $r^M(x)$, the original vector x can be approximated by $\tilde{x}$.

In the jth stage, we train the GMM codebook using EM on $\mathcal{X}^j$. Then for each $x^j \in \mathcal{X}^j$, only the mean vector μ_i^j of the Gaussian distribution with the largest $p(x^j|\theta_i^j)$ is considered to compute the residual vector.

D. Vector Compression Using MSRM

When vector quantization is chosen an the encoder, MSRM degenerates to MSVQ that can be directly used for vector compression.

Given a vector $x \in \mathbb{R}^d$, the code f^j generated by MSRM records the index of the nearest codeword to x in the jth stage. In other words, MSRM actually represents x using a sum of M codewords, with each codeword from one stage of the multiple stage codebook. Then x can be represented by

$$x = r^M(x) + \tilde{x} \tag{10}$$

where $r^M(x)$ represents the residual vector of x in the final stage, and the approximated x is defined as

$$\tilde{x} = \sum_{j=1}^{M} c^j_{i_j}, \quad i_j \in [1, N] \tag{11}$$

where $c^j_{i_j}$ denotes the i_jth codeword in the jth stage. An illustration of this approximation is given in Fig. 2. As a result, the original vector x can be compressed into N_{Byte} bytes that store the indices of entries in the multiple stage codebook, where N_{Byte} is determined by

$$N_{Byte} = \frac{M \log_2 N}{8}. \tag{12}$$

Using the same number and the same size of sub-codebook, MSRM shares the same compression rate with product quantization [9], optimized product quantization [33], and additive quantization [29].

We now discuss the asymmetric distance computation (ACD) between a given query vector q and a certain vector x in the database compressed by MSRM. The Euclidean distance is defined as

$$\|q - x\|_2^2 = \|q\|^2 + \|x\|^2 - 2\langle q, x\rangle \tag{13}$$

in which $\|q\|^2$ and $\|x\|^2$ denote the square norm of q and x respectively, and $\langle q, x\rangle$ calculates their inner product.

Note that $\|q\|^2$ is only determined by the query side, so we need to compute it on-the-fly and used repeatedly. However $\|q\|^2$ does not influence the ranking of the database vectors actually, and we can ignore this item if we only want to know the retrieved order, but do not need their distances to q. As a result, we only need to determine $\|x\|^2$ and $\langle q, x\rangle$ now.

Assuming that x is compressed by MSRM, then

$$\|x\|^2 \approx \|\tilde{x}\|^2 = \|\sum_{j=1}^{M} c^j_{i_j}\|^2 = \sum_{j=1}^{M}\sum_{j'=1}^{M} \langle c^j_{i_j}, c^{j'}_{i_{j'}}\rangle \tag{14}$$

which is an independent operation of the query q. We can precompute the inner product of all the pairs of codewords that come from different stages, and store them as a lookup table in memory. The space complexity of this storage is $O(M^2N^2)$, and it is reduced to $O(MN(M+1)(N-1)/4)$ if the symmetry property of distance is considered. Then, it needs $O(M(M+1)/2)$ lookups and $O(M(M+1)/2)$ additions to calculate the approximated square norm of x.

The inner product $\langle q, x\rangle$ is approximated by

$$\langle q, \tilde{x}\rangle = \left\langle q, \sum_{j=1}^{M} c^j_{i_j} \right\rangle = \sum_{j=1}^{M} \langle q, c^j_{i_j}\rangle. \tag{15}$$

For a given query, all the above items should be computed online. Since it is an inner product of two vectors, its computational cost is tiny.

Compared with PQ and APQ that deal with sub-vectors, MSRM directly works on the full vector by subtracting the residual iteratively. Although such a paradigm leads to higher memory usage (e.g., the space complexity for the codebook storage in MSRM is $O(DNM)$, while that in PQ is $O(ND)$), such extra minor cost can be tolerable. On the other hand, MSRM can be a very flexible quantizer, i.e., the number of sub-codebook M is independent of the input feature dimension D, while M must be a submultiple divisor of D in the case of PQ and OPQ.

Both AQ and MSRM share the same principle that each vector is approximated by the sum of M codewords from different sub-codebooks. However, the primary difference is that AQ considers each sub-codebook equally. By contrast, in MSRM, the sub-codebooks at the bottom layer are used for the coarsest quantization while those at the top layer are used for the

finest quantization. Moreover, AQ uses Beam Search algorithm for codebook learning and vector quantization, which is time-consuming. By contrast, MSRM adopts a greedy method derived from K-means, which enables efficient quantization.

Moreover, the flexibility of MSRM also lies in that we can achieve the desired compression rate according to the requirements of specific cases, by simply cutting off the later stage of MSRM without re-training, while PQ, OPQ and AQ all need re-train the quantization models.

Note that we can use more advanced encoding methods (e.g., Sparse Coding). However, these methods usually leads to codes in the float format instead of integer format, thus occupying more memory usage. Hence those encoding methods are usually not acceptable in vector compression task.

IV. Experiments

In this section, we evaluate the effectiveness of MSRM with applications to image classification, vector compression, approximate nearest neighbor search and shape recognition.

A. Image Classification

Different from previous works [29] that focus on image classification with limited memory usage, we address the complementarity of the codes generated from multiple stages for accurate image classification.

1) Implementation Details: If not specified, we adopt the following setup for all our experiments.

In order to get a proper assessment of MSRM, we re-implement the baselines of HC [13], SC [20], LLC [14], VLAD [27] and IFK [18] according to the following experimental setup. The comparisons to the results reported in the original papers are also given below.

Feature extraction. The standard dense SIFT is extracted on a patch size 32×32, with step size fixed to 4 pixels, by using the vl_dsift command available in the public toolbox VLFeat [40].

Codebook generation. According to Algorithm 2, we learn a multiple stage codebook using the training set. The codebook size depends on the encoder plunged into MSRM. In particular, HC, LLC and SC adopt a relatively larger codebook with a size of 1024 on all datasets, except for Caltech-256 where 4096 codes are used following [16]. As for VLAD and IFK, 256 codes are used in Caltech-256 and 64 codes for the other datasets.

Coding and pooling. As for the implementation of VQ, LLC, SC and IFK, we use the codes released by Huang in [1] to encode the local features. In order to include the spatial information, SPM [13] with 3 levels: 1×1 , 2×2 and 4×4 is adopted with a same weight for each level. We implement the VLAD encoder using vl_vlad function in VLFeat, and no spatial information is included. The max-pooling operation is performed with LLC and SC. HC, VLAD and IFK use the sum-pooling operation.

Classifier. Linear SVM, implemented by Liblinear toolbox [41], is used. We set the penalty parameter in SVM to 10.

Databset and Setup. Four databsets are used:

1) Scene-15 dataset [13]: it contains 15 categories and 4485 images, with 200 to 400 images per category. The categories vary from indoor scenes like bedrooms, to outdoor scenes like mountains. Based on the common experimental setting, 100 images per category are taken as training data, and the rest are used for testing.
2) UIUC 8-Sport [42]: it is particularly collected for image-based event classification, and it consists of 1579 images grouped into 8 sport categories: badminton, bocce, croquet, polo, rock climbing, rowing, sailing and snow boarding. According to the standard setup for classification, we use 10 splits of the data, and random select 70 images from each category for training and 60 images for testing. The average accuracy, as well as the standard deviation, is reported.
3) Caltech-101 dataset [43]: it consists of 101 object categories including animals, faces, plants etc, with 31 to 800 images per category. Following the standard experimental setting, we use 10 random splits of the data, while taking 30 random images per class for training and the rest for testing.
4) Caltech-256 dataset [44]: it is a more challenging dataset than Caltech-101 dataset, which contains $29,780$ objects divided into 256 categories including a background category. The conventional experimental setup is that random-selected 30 images per category are used for training and the rest for testing.

Images are resized to no more than 300×300 on all datasets except for UIUC 8-Sport, since images on this dataset have higher resolutions. We keep the maximum image size of UIUC 8-Sport dataset 400×400.

2) Experimental Comparison: As we can draw from Fig. 3(a), MSRM can significantly improve the performance compared with the original state-of-the-art coding algorithms on Scene-15 dataset. Generally, the classification accuracy improves as the stage number increases. It gets saturated when the stage number comes to 5. The classification accuracy is increased by 6.94% for HC, 2.90% for LLC, 4.79% for SC, 9.67% for VLAD and 1.23% for IFK when MSRM sets the stage number to 5. We can also find that our proposed model is especially suitable for HC and VLAD. The reason might be that both HC and VLAD only consider the nearest visual word in the codebook to encode a local feature, and our proposed model can boost their performances largely by making use of the computed residual vector. In comparison, LLC, SC and IFK take into account the contributions from more than one visual word, so the improvements of MSRM with LLC, SC and IFK are not as obvious as that with HC and VLAD, but are still convincing.

In Fig. 3(b), (c) and (d), we plot the performances of MSRM on the UIUC 8-Sport dataset, the Caltech-101 dataset and the Caltech-256 dataset respectively. Similar to our experiments in the previous dataset, MSRM improves our selected encoders significantly, e.g., on the Caltech-101 dataset, MSRM improves the performance of HC by 5.07%, LLC by 3.47%, SC by 5.33%, VLAD by 7.02% and IFK by 4.44%.

The comparison with other algorithms is given in Table I. All the results presented in the table are reported by author themselves or quoted from the survey paper [1]. As can be seen, our proposed method also obtains competitive performances against many state-of-the-art algorithms.

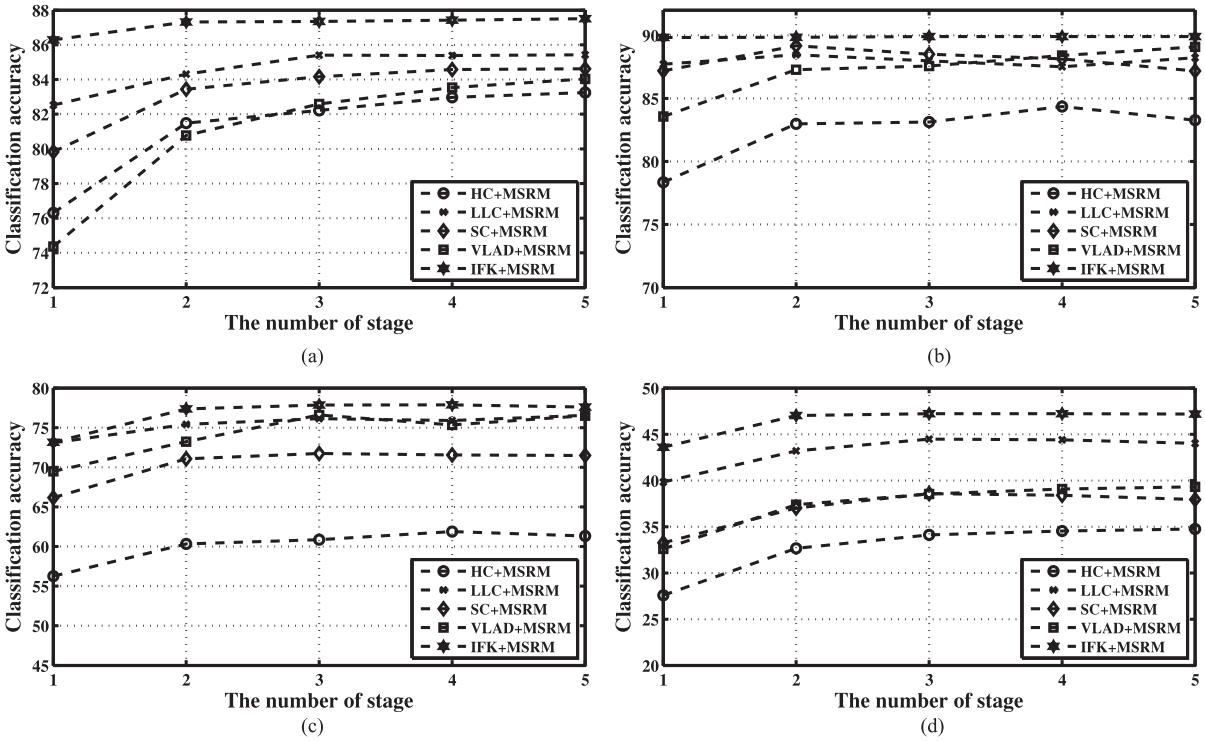

Fig. 3. Experimental results of MSRM with different encoders on (a) Scene-15 dataset, (b) UIUC-8 Sport dataset, (c) Caltech-101 dataset, and (d) Caltech-256 dataset. The x-axis denotes the stage number M in MSRM, and the y-axis denotes the classification accuracy.

TABLE I
COMPARISON OF CLASSIFICATION ACCURACIES

Algorithms	Scene-15	UIUC 8-Sport	Caltech-101	Caltech-256
HC [13]	78.87 ± 0.52	79.98 ± 1.67	69.43 ± 0.52	21.82 ± 0.22
LLC [14]	80.50 ± 0.63	81.77 ± 1.51	71.67 ± 0.86	37.41 ± 0.21
SC [20]	82.55 ± 0.41	85.44 ± 1.54	69.55 ± 0.83	34.60 ± 0.27
LCSRC [17]	82.67 ± 0.57	87.23 ± 1.14	73.23 ± 0.81	–
LSC [15]	82.70 ± 0.39	82.29 ± 1.84	72.58 ± 1.08	38.15 ± 0.26
LRSC [16]	–	88.17 ± 0.85	75.02 ± 0.74	41.04 ± 0.23
MOC [47]	83.38 ± 0.20	–	72.97 ± 0.80	–
SVC [19]	84.90 ± 0.52	–	–	–
IFK [18]	87.00 ± 0.36	–	–	$\mathbf{47.40 \pm 0.10}$
HC [13] + MSRM	83.25 ± 0.50	84.35 ± 1.16	61.88 ± 1.15	34.76 ± 0.26
VLAD [27] + MSRM	84.03 ± 0.64	89.09 ± 0.96	76.59 ± 0.51	39.31 ± 0.25
SC [20]+MSRM	84.62 ± 0.61	89.07 ± 1.49	71.73 ± 1.47	38.57 ± 0.25
LLC [14] + MSRM	85.42 ± 0.72	88.46 ± 1.13	76.56 ± 0.90	44.47 ± 0.32
IFK [18] + MSRM	$\mathbf{87.51 \pm 0.43}$	$\mathbf{89.90 \pm 1.00}$	$\mathbf{77.60 \pm 0.67}$	47.18 ± 0.19

On the Scene-15 dataset, the accuracy of our implementation for HC is much lower than that in [13], MSRM also enhances its discriminative power, and even outperforms the performances of LLC and SC. As for SC, Huang *et al*. [20] report a higher classification accuracy due to the use of multi-scale dense SIFT and a larger codebook. The baseline performance of IFK is 86.28%, slightly lower than that reported in [1]. However, MSRM with IFK as the encoder can easily achieve the best performance among all the compared algorithms. One should note that the performance of MSRM, as well as many other coding methods, can be improved easily by merely sampling image patches more finely as stated in [45]. Compared with the setup (extracting SIFT using 32×32 patches) of MSRM, all the experiments in [1] are done by extracting SIFT more densely (16×16, 24×24, 32×32), which generates much more descriptors.

On the UIUC 8-Sport dataset and the Caltech-101 dataset, the performances of MSRM are better than many coding methods [13]–[15], [17], [20]. MSRM even outperforms Low Rank Sparse Coding [16], which achieves the state-of-the-art performance recently.

On the challenging Caltech-256 dataset, the performance of IFK reported in [18] is 40.80%, which is lower than the best performance of MSRM. By extracting SIFT at 5 scales, Sanchez *et al* . [46] report 47.40% accuracy for IFK, still comparable with the performance 47.18% of MSRM.

Multi-layer orthogonal codebook (MOC) [47] is also a multiple layer encoding method which uses the orthogonal residual vectors. Its best performance is merely 83.38% on the Scene-15 dataset and 72.97% on the Caltech-101 dataset, achieved by integrating some additional techniques (e.g., kernel fusion and soft weighting).

3) Discussion: The complementary nature of codes from different stages, as well as the codebook size and the influence of power normalization, is discussed in this section.

The complementarity of codes. We observe that the output of each stage in MSRM is complementary to each other, and the powerful classifier SVM is able to select several distinctive stages to distinguish images from different categories.

In order to prove our conjecture, we conduct an interesting experiment shown in Fig. 4. Here we set the encoder of MSRM as VLAD, and the features from each stage in MSRM form the training set and the testing set. We select one misclassified image per category, and find that although the classifier cannot

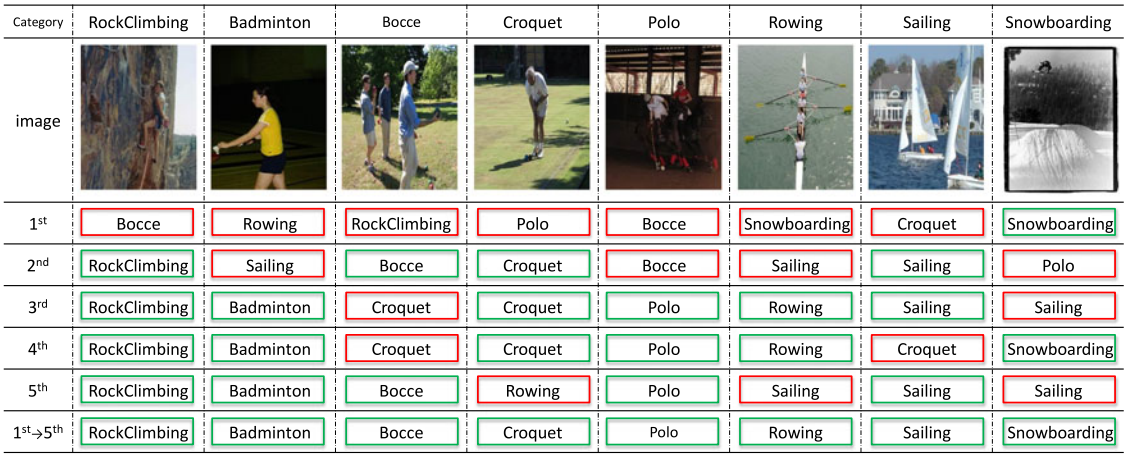

Fig. 4. Some misclassified images on the UIUC 8-sport dataset. The "Category" in the first row indicates the ground truth of the image, and the third to the seventh row present the predicted label that SVM outputs based on the features of each stage in MSRM. The last row shows the result when we concatenate the features from all stages in MSRM. The red box means a false prediction, and the green box means a correct prediction.

TABLE II
COMPARISON BETWEEN MSRM AND THE SINGLE-STAGE CODING STRATEGY UNDER THE SAME FEATURE DIMENSION ON THE SCENE-15 DATASET

M	1	2	3	4	5
VLAD	74.36%	78.15%	79.35%	80.14%	80.18%
MSRM	74.36%	80.77%	82.59%	83.53%	84.03%

give a correct recognition according to the features from 1-st stage in MSRM, i.e., the original coding strategy, it can revise its prediction result with more complementary information from the latter stages in MSRM. For example, the image from Rowing category in the seventh column of Fig. 4, is misclassified as snowboarding in the first stage and sailing in the second stage. However, it obtains a correct labeling in the third stage and the fourth stage. When combining the features from all stages, the false prediction is corrected as shown in the last row. The last column presents an image from snowboarding is misclassified in the second, third and fifth stage, but is assigned a right label if the features from all the stages are combined together. This phenomena reveals the robustness of our proposed model.

To further confirm the complementarity between the features from each stage of MSRM, we compare our proposed MSRM with a single stage coding strategy in the same feature dimension. Specifically, Scene-15 dataset is used and VLAD is selected as the encoder. For MSRM, the codebook size in each layer of MSRM is 64, and the stage number M ranges from 1 to 5. Hence the feature dimension of MSRM for an image is $128 * 64 * M$. For the single stage coding, the codebook size is set directly to $64 * M$ ($1 \leq M \leq 5$). Hence the final feature dimension for an image is also $128 * 64 * M$. The results are presented in Table II, which suggest that MSRM works significantly better than only single stage coding.

Codebook size. The size of codebook is crucial for the classification accuracy. It has been proven that a better performance can be achieved by using a relative larger codebook. However,

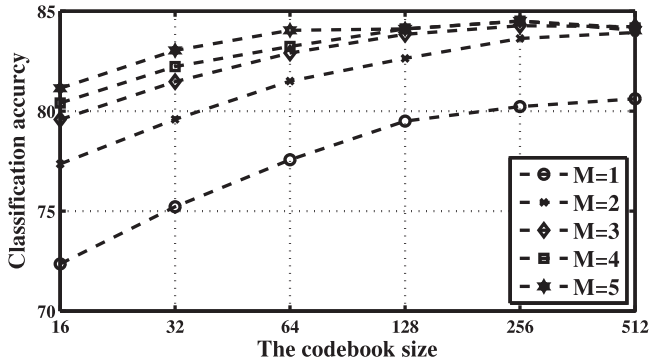

Fig. 5. Influence of codebook size on classification accuracy.

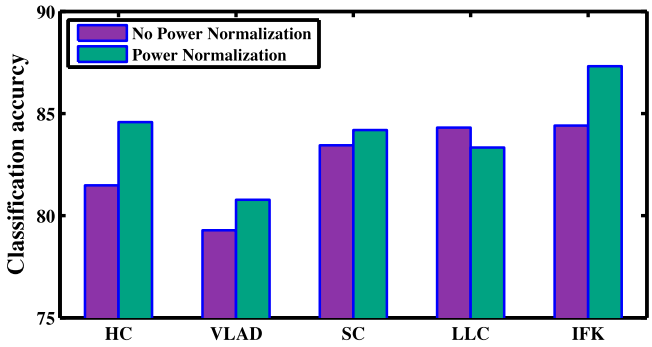

Fig. 6. Influence of power normalization.

overfitting effect, leading to the plateau of performance curves, occurs when the codebook size keeps increasing. In Fig. 5, we plot the influence of codebook size when setting the encoder of MSRM as VLAD, and the stage number M varying from 1 to 5. It can be found that the classification accuracy gets saturated when the codebook size is 256.

Power normalization. After pooling step, the generated codes are usually $\mathcal{L}_2$-normalized. However, in some cases, some extremely large elements in the code dominate the similarity calculation with other images, so that the contribution of other

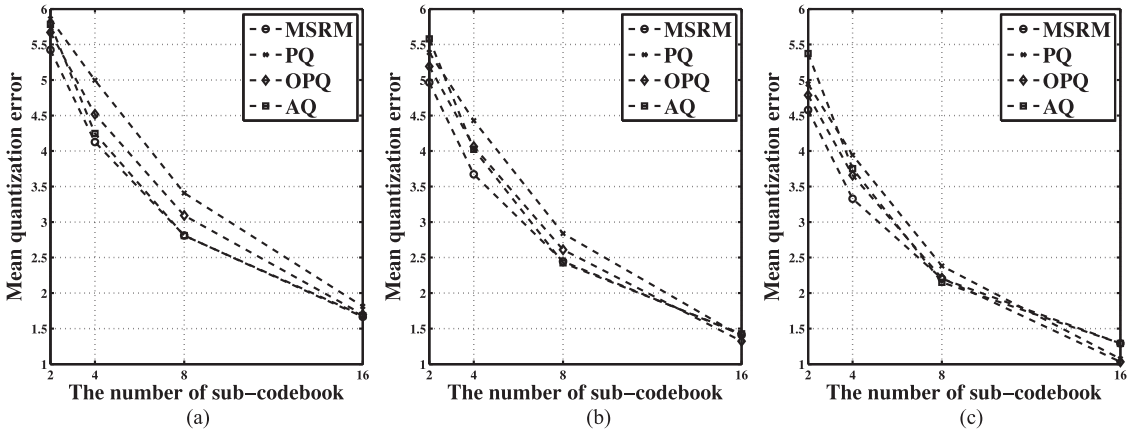

Fig. 7. Comparison of quantization error ($\times 10^4$) on the SIFT1M dataset. The size of sub-codebook is set to (a) 64, (b) 128, and (c) 256.

important dimensions is restrained. In order to remedy this, power normalization, i.e., square root, is used before the final $\mathcal{L}_2$ normalization, as suggested in [18].

In Fig. 6, we give the influence of power normalization on the Scene-15 dataset. We follow the default setting for codebook size, and the stage number of MSRM is 2. As the figure shows, power normalization is more effective to improve the performance when the encoder of MSRM (e.g., HC, VLAD and IFK) uses sum-pooling. It is easy to understand, since sum-pooling is more likely to produce extreme large values than max-pooling.

B. Vector Compression

Quantization error. We give a thorough comparison on quantization error with some state-of-the-art algorithms, including product quantization [9], the optimized product quantization [33] and additive quantization [29].

Two datasets, SIFTIM [9] and GIST1M [9], are used for evaluation.

a) *SIFT1M:* It consists of $1M$ SIFT [28] descriptors in single format of 128 dimension, and 10K descriptors are considered as the query. $100K$ independent descriptors are used for codebook learning task.

b) *GIST1M:* It consists of $1M$ GIST descriptors in single format of 960 dimension, and $1K$ descriptors are considered as the query. $500K$ independent descriptors are used for codebook learning task.

In Fig. 7, we compare the mean approximation error when the stage number M varies from 2 to 16. Several observations are made.

1) In general, MSRM achieves the lowest quantization error compared with other algorithms.
2) When a coarse quantization is applied (i.e., N is 128 and M is 2), the advantage of MSRM over other methods is more obvious.
3) When M is 16 and $N = 256$, the mean quantization error of MSRM is slightly higher to that of AQ and OPQ.
4) Although AQ has a same optimization objective as MSRM, but the solution of MSRM is more accurate, which demonstrates the benefit of using the residual vectors stage by stage.

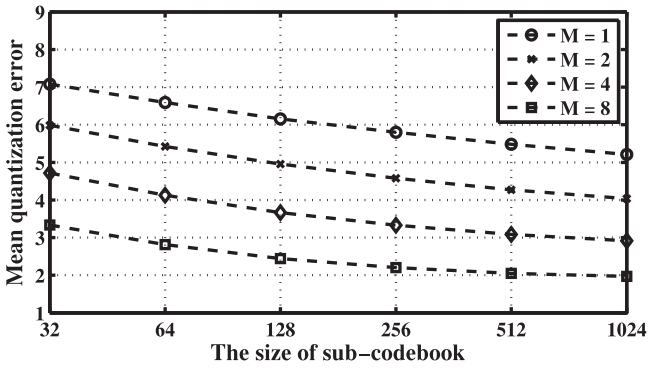

Fig. 8. Influence of codebook size on mean quantization error on the SIFT1M dataset.

TABLE III
COMPARISON OF QUANTIZATION ERROR UNDER THE SAME EXTRA COST ($\times 10^4$)

N	64	128	256	512
VQ	6.173	5.802	5.475	5.214
MSRM	5.424	4.964	4.577	4.268

In order to directly illustrate the influence of codebook size, we give a comparison in Fig. 8. As can be seen, larger codebook naturally leads to lower mean quantization error. We also find that with the same compression rate, it is more beneficial to use smaller codebooks with deeper stage. For example, The codebook storage and the compression rate is the same when $M = 2, N = 256$ and $M = 4, N = 128$. However, the quantization error of the former parameter setup is much higher than that of the latter one. The extreme case of MSRM, i.e., $M = 1$, is vector quantization. Its performance is also presented in Fig 8.

In order to present the difference of quantization error between VQ and MSRM more clearly, a comparison is given in Table III. We use two stages for MSRM, and set the sub-codebook size to N. For VQ, the codebook size is set to $2N$. Consequently, the extra cost of codebook storage is kept the same. As we can see from the table, MSRM leads to less information loss than VQ under the same extra cost, which

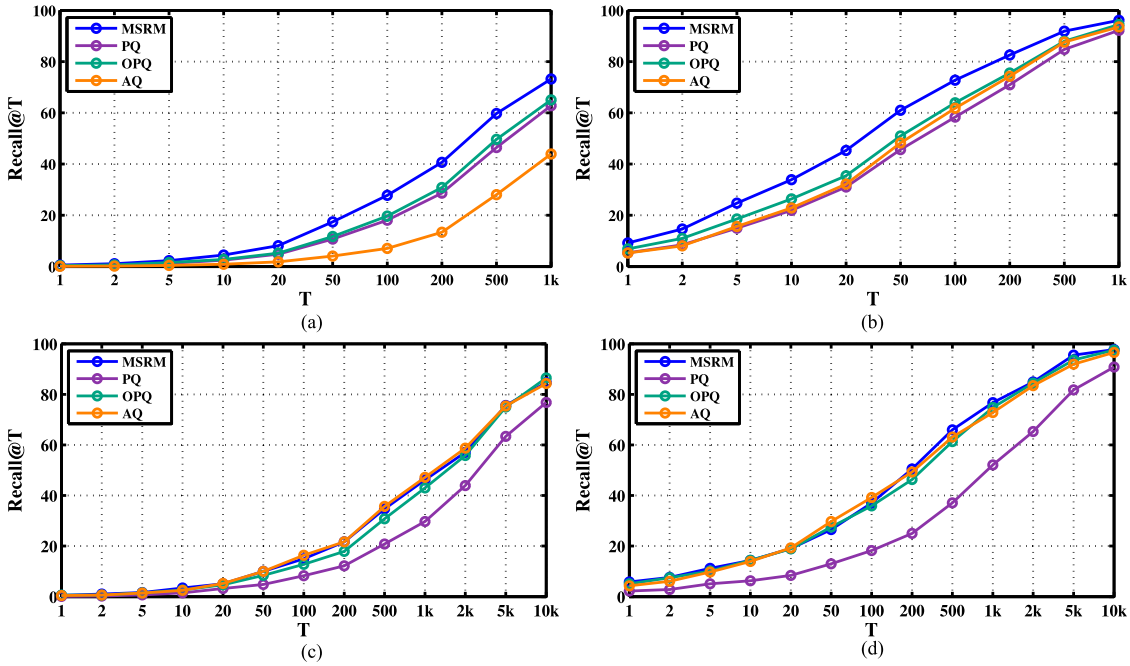

Fig. 9. Comparison of recall on the SIFT1M dataset and GIST1M dataset. The size of sub-codebook N is set to 256, and the number of sub-codebook M is 2 for (a) and (c), 4 for (b) and (d).

demonstrates again the effectiveness of our proposed hierarchical structure.

ANN Search. We also use SIFT1M dataset and GIST1M dataset to evaluate the performance in approximate nearest-neighbor search. Here we only consider Asymmetric Distance Computation, i.e., only the database is quantized and the query remains uncompressed. The performance measure is $recall@T$ (for varying values of T, the average rate that the true 1-nearest neighbor is ranked in the top T positions). In Fig. 9, we give a comparison when N is set to 256, and M is set to 2 or 4. The former setup only requires 2 byte per vector for storage, and the latter one uses 4 byte according to (12).

As can be seen in Fig. 9(a) and (b), MSRM achieves the best overall performance compared with PQ, OPQ and AQ on SIFT1M dataset. Consistent with the previous analysis in quantization error, the performance difference between MSRM and other methods is more distinctive when a coarse quantization is used. This is easy to understand, since the quantization error is the upper bound of the approximation of asymmetric distance computation as $|\|q-x\|_2 - \|q-\tilde{x}\|_2| \leq \|x-\tilde{x}\|_2$. The superior performance of MSRM in coarse vector compression makes it very suitable for the applications in mobile device, where the storage space is limited. On GIST1M dataset, OPQ, AQ and MSRM achieve comparable performances, probably due to the complex structure of vectors brought by the high dimension of GIST feature.

In Table IV, we compare the time cost of difference methods in codebook learning procedure and vector encoding procedure on SIFT1M dataset. The number of sub-codebook M is 4 and the size of sub-codebook is 256. The algorithms are implemented in Matlab with some utilities written in MEX files. All the experiments, except for AQ, are carried out on a desktop machine with an Intel(R) Core(TM) i5 CPU (3.40 GHz). Considering the low efficiency of AQ, the experiment of AQ is done on a computing server with Intel(R) CPU E5-2609 with 8 paralleled computing core. As can be seen from the table, PQ is the most efficient method, since it works on sub-vectors and does not include any complicated optimization. The time cost of MSRM is slightly larger than PQ, but achieves much better performances as presented in the previous section.

TABLE IV
COMPARISON OF RUNNING TIME

	Codebook Learning	Vector Encoding
PQ	85.46 s	24.22 s
OPQ	93.13 s	24.35 s
AQ	$\approx$ 1 h	$\approx$ 4 h
MSRM	146.71 s	26.18 s

C. Shape Recognition

Shape recognition [48], [49] is an appealing task, which attracts lots of attention for a long time. Some representative algorithms [50], [51] proposed recently also adopt coding-based framework for robust and efficient classification. Consequently, MSRM can be also applied to this task easily.

MSRM is evaluated on MPEG-7 dataset [31]. MPEG-7 dataset consists of 1400 silhouette images divided into 70 classes, with 20 shapes per category. Some representative shapes are given in Fig. 10. Following the conventional setup on this dataset, half shapes per category are used for training, and the remaining shapes are used for testing. We repeat the testing and training procedure for 5 times.

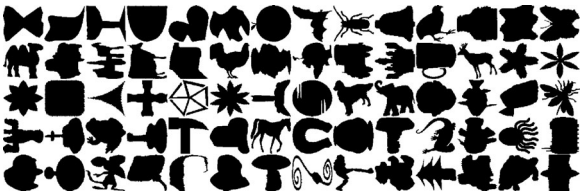

Fig. 10. Typical shapes from the MPEG-7 dataset. One shape per category is represented.

TABLE V
PERFORMANCE COMPARISON ON THE MPEG-7 DATASET

Methods	Original	MSRM
HC	95.80 ± 0.75	97.08 ± 0.38
SC	96.57 ± 0.54	97.20 ± 0.29
LLC	96.94 ± 0.63	97.51 ± 0.77
IFK	97.15 ± 0.40	97.80 ± 0.43
VLAD	96.45 ± 0.30	$\mathbf{98.00 \pm 0.70}$
BCF [50]	97.16 ± 0.79	
Class Segment Set [54]	90.9	
Contour Segments [55]	91.1	
Skeleton Paths [55]	86.7	
ICS [55]	96.6	

The raw descriptor used here is contour fragment developed in [50], which serves as a baseline method. First, we obtain lots of contour fragments by tracing a pair of vertices, and these vertices are generated by applying Discrete Contour Evolution (DCE) [52] to the outer contour of shape. Then for each contour fragment, 5 reference points are set evenly along the fragment, and described by Shape Context (SC) [53]. At last, each contour fragment is represented by the concatenation of the five shape context features.

The codebook is learned only on the training set. The size of sub-codebook for HC [13], LLC [14] and SC [20] is 1000, and that for VLAD [27] and IFK [18] is 20. The stage number M is set to 2. Different from BCF [50], we do not apply SPM [13] to preserve the spatial arrangement of these contour fragments, since we experimentally find that MSRM can already achieve the state-of-the-art performances with only such a coarse-to-fine coding framework.

The performance comparison of MSRM with other representative algorithms is given in Table V. The special case of $M = 1$ of MSRM is referred to "Original" in the table. We can find only 2 stages can increase the performance of the original coding methods significantly. When VLAD is chosen as the encoder of MSRM, we also report the highest performance **98.00 $\pm$ 0.70**, which outperforms the new state-of-the-art algorithm BCF [50] by a large margin. When power normalization is used, no obvious improvements are observed.

V. CONCLUSION

In this paper, we propose a generic model called Multiple Stage Residual Model to make full use of the residual vector, while many coding algorithms focus on reducing it. MSRM has been proven effective in improving the performance of many state-of-the-art coding algorithms further in image classification and shape recognition task. As a special case, MSRM with VQ as the encoder can be adjusted to vector compression, and exhibits competitive performances against other classic algorithms, such as product quantization. In the future, we will study how to incorporate the spatial consistency with MSRM in a proper way.

In recent years, deep learning approaches (e.g., Convolutional Neural Network [56]) have demonstrated its superiority in image classification task with the help of large amounts of training images. It can be expected that the combination of those deep learning approaches and conventional feature coding approaches can yield better performances in many computer vision tasks.

REFERENCES

[1] Y. Huang, Z. Wu, L. Wang, and T. Tan, "Feature coding in image classification: A comprehensive study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 493–506, Mar. 2014.

[2] H. Jégou, "Triangulation embedding and democratic aggregation for image search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 3310–3317.

[3] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Packing and padding: Coupled multi-index for accurate image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 1947–1954.

[4] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Fast image retrieval: Query pruning and early termination," *IEEE Trans. Multimedia*, vol. 17, no. 5, pp. 648–659, May 2015.

[5] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997–2008, Dec. 2013.

[6] J. Yi, Y. Peng, and J. Xiao, "Exploiting semantic and visual context for effective video annotation," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1400–1414, Oct. 2013.

[7] O. de Rooij and M. Worring, "Active bucket categorization for high recall video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 898–907, Jun. 2013.

[8] S. Bai and X. Bai, "Sparse contextual activation for efficient visual reranking," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1056–1069, Mar. 2016.

[9] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.

[10] R. Kosala and H. Blockeel, "Web mining research: A survey," *ACM SIGKDD Explorations Newslett.*, vol. 2, no. 1, pp. 1–15, 2000.

[11] F. Hu *et al.*, "Unsupervised feature learning via spectral clustering of multidimensional patches for remotely sensed scene classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 5, pp. 2015–2030, May 2015.

[12] W. Yang, X. Yin, and G.-S. Xia, "Learning high-level features for satellite image classification with limited labeled samples," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 8, pp. 4472–4482, Aug. 2015.

[13] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2006, vol. 2, pp. 2169–2178.

[14] J. Wang *et al.*, "Locality-constrained linear coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3360–3367.

[15] L. Liu, L. Wang, and X. Liu, "In defense of soft-assignment coding," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2486–2493.

[16] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja, "Low-rank sparse coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 281–288.

[17] A. Shabou and H. LeBorgne, "Locality-constrained and spatially regularized coding for scene categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 3618–3625.

[18] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 143–156.

[19] X. Zhou, K. Yu, T. Zhang, and T. S. Huang, "Image classification using super-vector coding of local image descriptors," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 141–154.
[20] Y. Huang, K. Huang, Y. Yu, and T. Tan, "Salient coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 1753–1760.
[21] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 696–709.
[22] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2009, pp. 1794–1801.
[23] A. Shaban, H. R. Rabiee, M. Farajtabar, and M. Ghazvininejad, "From local similarity to global coding: An application to image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2794–2801.
[24] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 2223–2231.
[25] W. Shen *et al.*, "Exemplar-based human action pose correction," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1053–1066, Jul. 2014.
[26] X. Bai, S. Bai, Z. Zhu, and L. Latecki, "3D shape matching via two layer coding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 12, pp. 2361–2372, Dec. 2015.
[27] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3304–3311.
[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[29] A. Babenko and V. S. Lempitsky, "Additive quantization for extreme vector compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 931–938.
[30] S. Bai, X. Wang, C. Yao, and X. Bai, "Multiple stage residual model for accurate image classification," in *Proc. 12th Asian Conf. Comput. Vis.*, 2014, pp. 430–445.
[31] L. J. Latecki, R. Lakamper, and T. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2000, vol. 1, pp. 424–429.
[32] R. Arandjelovic and A. Zisserman, "All about VLAD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 1578–1585.
[33] M. Norouzi and D. J. Fleet, "Cartesian k-means," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 3017–3024.
[34] T. Zhang, C. Du, and J. Wang, "Composite quantization for approximate nearest neighbor search," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 838–846.
[35] B. H. Juang and A. Gray, "Multiple stage vector quantization for speech coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 1982, vol. 7, pp. 597–600.
[36] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[37] X. Wang, B. Wang, X. Bai, W. Liu, and Z. Tu, "Max-margin multiple-instance dictionary learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 846–854.
[38] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, "Supervised dictionary learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1033–1040.
[39] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3517–3524.
[40] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008. [Online]. Available: http://www.vlfeat.org/
[41] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/libsvm
[42] L.-J. Li and L. Fei-Fei, "What, where and who? classifying events by scene and object recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.
[43] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," *Comput. Vis. Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.
[44] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2007-001, 2007.
[45] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 2559–2566.
[46] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. J. Comput. Vis.*, vol. 105, no. 3, pp. 222–245, 2013.
[47] X. Li, Y. Song, Y. Lu, and Q. Tian, "Multi-layer orthogonal visual codebook for image classification," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 2312–2315.
[48] H. Ling and D. W. Jacobs, "Shape classification using the inner-distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 286–299, Feb. 2007.
[49] L. Luo, C. Shen, C. Zhang, and A. van den Hengel, "Shape similarity analysis by self-tuning locally constrained mixed-diffusion," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1174–1183, Aug. 2013.
[50] X. Wang, B. Feng, X. Bai, W. Liu, and L. J. Latecki, "Bag of contour fragments for robust shape classification," *Pattern Recog.*, vol. 47, no. 6, pp. 2116–2125, 2014.
[51] R. Bharath, X. Cheng, and H. L. Tong, "Shape classification using invariant features and contextual information in the bag-of-words model," *Pattern Recog.*, vol. 48, no. 3, pp. 894–906, 2015.
[52] L. J. Latecki and R. Lakämper, "Convexity rule for shape decomposition based on discrete contour evolution," *Comput. Vis. Image Understanding*, vol. 73, no. 3, pp. 441–454, 1999.
[53] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
[54] K. B. Sun and B. J. Super, "Classification of contour shapes using class segment sets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2005, vol. 2, pp. 727–733.
[55] X. Bai, W. Liu, and Z. Tu, "Integrating contour and skeleton for shape classification," in *Proc. Comput. Vis. Workshops*, 2009, pp. 360–367.
[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

Song Bai received the B.S. degree in electronics and information engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2013, and is currently working toward the Ph.D. degree in electronic information and communications at HUST.

His research interests include shape analysis, image classification, and retrieval.

Xiang Bai (S'07-M'09-SM'14) received the B.S., M.S., and Ph.D. degrees in electronics and information engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003, 2005, and 2009, respectively.

He is currently a Professor with the School of Electronic Information and Communications, HUST. He is also the Vice-Director of the National Center of Anti-Counterfeiting Technology, HUST. His research interests include object recognition, shape analysis, scene text recognition, and intelligent systems.

Wenyu Liu (M'08-SM'15) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1986, and the M.S. and Ph.D. degrees in electronics and information engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1991 and 2001, respectively.

He is currently a Professor and the Associate Dean of the School of Electronic Information and Communications, HUST. His current research interests include computer vision, multimedia, and sensor network.