Received: 9 October 2018 | Revised: 15 June 2019 | Accepted: 29 July 2019

DOI: 10.4218/etrij.2018-0557

ORIGINAL ARTICLE

ETRI Journal WILEY

Supervised-learning-based algorithm for color image compression

Xue-Dong Liu | **Meng-Yue Wang** | **Ji-Ming Sa**

Key Laboratory of Broadband Wireless Communications and Sensor Networks, School of Information Engineering, Wuhan University of Technology, Wuhan, Hubei, China

Correspondence
Xue-Dong Liu, Key Lab. of Broadband Wireless Communications and Sensor Networks, School of Information Engineering, Wuhan University of Technology, Wuhan, Hubei, China.
Email: zxndp@126.com

A correlation exists between luminance samples and chrominance samples of a color image. It is beneficial to exploit such interchannel redundancy for color image compression. We propose an algorithm that predicts chrominance components Cb and Cr from the luminance component Y. The prediction model is trained by supervised learning with Laplacian-regularized least squares to minimize the total prediction error. Kernel principal component analysis mapping, which reduces computational complexity, is implemented on the same point set at both the encoder and decoder to ensure that predictions are identical at both the ends without signaling extra location information. In addition, chrominance subsampling and entropy coding for model parameters are adopted to further reduce the bit rate. Finally, luminance information and model parameters are stored for image reconstruction. Experimental results show the performance superiority of the proposed algorithm over its predecessor and JPEG, and even over JPEG-XR. The compensation version with the chrominance difference of the proposed algorithm performs close to and even better than JPEG2000 in some cases.

KEYWORDS
color image compression, interchannel redundancy, LapRLS, prediction model, supervised learning, total prediction error

1 | INTRODUCTION

Advances in information technology have led to an increased use of multimedia such as images and videos to share information over the Internet. Furthermore, users are increasingly demanding better quality multimedia as well as faster transmission rates. To meet these demands, researchers are continuously pursuing improvements in image compression techniques. It is well-known that redundancy exists in the adjacent pixels of an image. By exploiting this redundancy, an image can be compressed to reduce the necessary storage space and transmission bandwidth. The existing compression techniques such as JPEG [1], JPEG2000 [2], and JPEG-XR [3] transform images into the frequency domain, and then store the coefficients of the frequency domain, which are encoded elaborately. At the decoder, the coefficients are transformed inversely into the spatial domain to reconstruct the original image either precisely or approximately.

Traditionally, a color image coder processes three color channels independently. In fact, an interchannel correlation exists between the three color components of an image. This correlation has been exploited to improve the coding efficiency of the state-of-the-art video coding standard, that is, High Efficiency Video Coding (HEVC) [4–6]. Models that exploited interchannel correlation using the reconstructed luma to linearly predict the chroma with parameters derived from neighboring reconstructed luma and chroma pixels for HEVC were proposed in [5] and [6].

In colorization-based coding, the encoder selects a few representative pixels (RPs) whose chrominance values and positions are stored or transmitted together with the luminance component; in the decoding stage, the chrominance values for all the pixels are colorized by the RPs and the decompressed luminance image. The main issue in colorization-based coding is how to extract the RP so that the rate-distortion (RD) performance is good. Lee and others [7] formulated the colorization-based coding problem into an L_1 minimization problem. For a fixed reconstruction error value and a given colorization matrix, the selected set of RPs is guaranteed to be the smallest set. To improve the image coding performance, Uruma and others [8] proposed the RP compression algorithm using the graph Fourier transform, and accordingly introduced the colorization technique using a representative graph spectrum based on the graph Fourier transform.

Recently, remarkable progress has been made in the application of machine learning technologies to image compression. Reference [9] proposed the use of color information from a few RPs to learn a model that predicts the colors of the rest of the pixels and recovers the original image by storing the RPs and grayscale edition of the image. Based on the statistical theories of optimal experimental design [10], He and others [11] proposed an active learning algorithm, called graph-regularized experimental design (GRED), and showed that it was more effective than the method used in [9]. Noting that all the real colors are known at the encoding stage, [12] proposed an active learning algorithm that selects a subset of pixels as seeds, that is RPs, to achieve total prediction error minimization (TEM) of chrominance. The seeds were stored together with the luminance component for the decoder, which were used to train a prediction model with a semi-supervised-learning algorithm. The model was then applied to predict the colors of all the pixels. It has been reported that TEM outperforms the methods proposed in [9] and [11]. The three aforementioned algorithms share two key procedures: selecting representative color seed pixels by means of standard active learning and predicting color values with a model trained by a semi-supervised-learning algorithm.

Seed point selection procedures by active learning are time-consuming. In fact, it is not necessary to use complex active learning to ensure total error minimization. Because all the color values are available at the encoder, we can use supervised learning to construct a Laplacian-regularized least square (LapRLS) objective function to minimize the total prediction error naturally over all the training points. Consequently, time-consuming active learning is omitted. In view of the massive number of training points and size of matrices in the expression of the optimal solution of the prediction model as well as the resultant computational burden, we use kernel principal component analysis mapping (KPCAM) [13]. KPCAM generates a space with reduced dimension m instead of the original number of training points n. Aside from KPCAM, effective strategies such as chrominance subsampling and entropy coding are applied to further reduce the computational cost and bit rate. In experiments, we explored the laws governing the effects of various factors on the RD performance of the proposed algorithm.

The rest of this paper is organized as follows. A brief review of the previous similar works from which the proposed algorithm evolved is given in Section 2. Our supervised-learning-based algorithm (SLBA) for color image compression is described in Section 3. Experimental results are discussed in Section 4, and conclusions are drawn in Section 5.

2 | PREVIOUS SIMILAR WORKS

Exploiting the correlation between different channels of a color image to further reduce data size is a recent concept. In previous similar works [9,11,12], semi-supervised learning was utilized to minimize the loss of labeled examples for training and then generalized to all examples for prediction. Graph-based semi-supervised-learning methods construct a problem graph G to denote the neighborhood relation among examples (both labeled and unlabeled). A function of the G acts as a regularizer to implement the manifold assumption or local consistency assumption [13]. Then, another regularizer is added to ensure the numerical stability of the solution. Integration of the two regularizers with prediction error forms the LapRLS objective function, whose optimal solution leads to a prediction or regression model. The optimal solution is of a closed form that deals with the inverse of an $n \times n$ matrix, where n is the number of total examples, including both labeled and unlabeled ones. To select the most representative seed points, the previous methods used various active learning schemes. Reference [9] used a heuristic strategy to select the RPs from regions where a high prediction error occurs. Reference [11] selected seeds such that the determinant of the covariance matrix of the regression coefficients is minimized to ensure that the optimal solution is as stable as possible. Making full use of the fact that all true colors are available at the encoder, Reference [12] selected the most informative pixels to ensure that the total prediction error is minimized and the algorithm outperforms the two previous ones.

In the aforementioned machine learning-based color image compression methods, luminance component Y was stored whereas the two chrominance components, Cb and Cr, were predicted from a model f trained by semi-supervised learning. f predicts the color values of Cb and Cr corresponding to the input feature vector (R, C, I), where (R, C) is the location of a pixel, and I is the luminance value. The predicted chrominance values are called labels, denoted by y. In semi-supervised learning, a set of labeled pairs $\{(\mathbf{x}_1, y_1), \ldots,$

$(\mathbf{x}_l, y_l)\}$ is needed to train the prediction model while another set of unlabeled pixels $\{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_n\}$ helps improve the learning performance of model f, where $\mathbf{x}_i$ is the ith feature point and y_i is the corresponding label.

References [9,11,12] present three algorithms of the same type. The proposed algorithm SLBA evolves from TEM [12], which is considered the best one of its kind so far. As a variant of TEM, SLBA outperforms it. For a better understanding of SLBA and the reasons why it is superior to TEM, we introduce TEM briefly as follows.

In TEM, the regression model is trained by all the training points although the objective function attempts to minimize the prediction error only on the smaller number of labeled points. This is shown in

$$\arg\min\{J(f) = \left\|\mathbf{Y}_X - \mathbf{F}_X\right\|^2 + \lambda_1 \|f\|^2 + \lambda_2 \mathbf{F}_U^T \mathbf{L} \mathbf{F}_U\},$$

where X is the set of labeled points and U is the set of all training points; $\mathbf{Y}_X$ is a vector containing color labels corresponding to set X; $\mathbf{F}_X$ and $\mathbf{F}_U$ are the vectors containing color values predicted by f corresponding to X and U, respectively; $\mathbf{L}$ is the so-called graph Laplacian matrix; λ_1 and λ_2 are two constants; $\|\cdot\|$ is the norm operator; T denotes the transpose operation. The first term is the prediction error over the labeled points. The two latter terms are the regularizers that help improve the regression model. The points in set X are called color seeds. This is the semi-supervised-learning paradigm. Seed selection plays a very important role because different seeds may result in different prediction performances. Reference [12] adopted active learning to select points as seeds so that f can minimize the total prediction error.

There are two schemes to select the optimal seed pixel points: one starts with an empty set X and then adds the selected color seeds iteratively; the other starts with all the pixels, that is, X = U and then removes pixels iteratively. The latter is adopted in TEM because it performs better. The selection process also needs to compute the total prediction error each time a point is removed. This is implemented exhaustively for every candidate point. Those points are removed if the resultant X can produce the total prediction error minimum. The iteration continues until the predefined number of points in set X, that is l, is reached. The selection of seed points is a time-consuming process.

In summary, TEM uses an active learning algorithm to select a subset of pixels as seeds. The stored luminance and color seeds are used to train a regression model by a semi-supervised-learning algorithm. The model is then applied to all the pixels to predict their chrominance values. The selection of seeds is time-consuming, and the storage requirement of the seeds is higher. In view of these disadvantages of TEM, we developed SLBA.

3 | SUPERVISED-LEARNING-BASED ALGORITHM FOR COLOR IMAGE COMPRESSION

3.1 | Supervised-learning by LapRLS

Our work focuses on color image compression by exploiting interchannel correlation. The proposed SLBA evolves from the semi-supervised-learning approach, LapRLS, mentioned earlier [12]. The previous objective function of LapRLS shared a term denoting the prediction error over only l labeled pixels. In addition, there are other regularizers constructed by all (both labeled and unlabeled) n training pixel points in the objective function. The resultant optimal solution that minimizes the objective function involves the inverse of a large-sized matrix. In TEM, there are n training points, but the learnt f minimizes the prediction error over only l ($l \ll n$) labeled points. To achieve total error minimization, TEM launches an iterative active learning procedure, which accounts for the majority of time consumed. In fact, we can achieve total error minimization by modifying the objective function of LapRLS, thereby avoiding the time-consuming active learning process. Compared with previous methods based on semi-supervised and active learning, it is necessary to highlight the improvements and innovations proposed to the method in this paper.

1. SLBA can minimize the total prediction error without the time-consuming active learning procedure.
2. KPCAM is adopted to reduce computational cost. The required m representative points are sampled from the same fixed grid of the reconstructed luminance component, which makes it possible for both the encoder and decoder to use the same parameters to predict chrominance values of pixels without signaling location information.
3. Unlike previous methods [9,11,12], which need a set of color seeds, including location, luminance, and chrominance values, SLBA only uses $2m$ parameters to convey Cb and Cr information. SLBA further refines our previous method [14]. The fractional parameters are rounded off and then entropy coded, which improves the RD performance of SLBA because the needed bit overhead is lower.
4. Finally, utilizing the insensitivity of the human eye to chrominance, SLBA improves our previous method [14]. Chrominance subsampling is adopted, which results in a lower bit rate and remarkable RD performance.

In TEM, the term in the LapRLS objective function denoting prediction error over l labeled points is $\sum_{i=1}^{l} \left(y_i - f(\mathbf{x}_i)\right)^2$, where $\mathbf{x}_i$ is the ith seed point in X and y_i is the corresponding

label in $\mathbf{Y}_X$. Because all pixel colors are available at the encoder, we modified the term to $\sum_{i=1}^{n}\left(y_i - f\left(\mathbf{u}_i\right)\right)^2$, where $\mathbf{u}_i$ is the ith training point and n is the total number of training points. This is the supervised-learning paradigm that attempts to minimize the prediction error over all the n training pixels:

$$\arg\min_f \sum_{i=1}^{n}\left(y_i - f\left(\mathbf{u}_i\right)\right)^2 = \arg\min_f \left\|\mathbf{Y}_U - \mathbf{F}_U\right\|^2, \quad (1)$$

where $\mathbf{Y}_U = [y_1, \ldots, y_n]^T$ and $\mathbf{F}_U = [f(\mathbf{u}_1), \ldots, f(\mathbf{u}_n)]^T$.

To enhance the nonlinear prediction ability, we will learn a function f in the reproducing kernel Hilbert space (RKHS) [15]. According to the representer theorem, there exists a set of coefficients α_i so that f can be expressed as a linear combination $f(\mathbf{u}) = \sum_{i=1}^{n}\alpha_i k\left(\mathbf{u}, \mathbf{u}_i\right)$, where $k(\cdot,\cdot)$ is a kernel function that satisfies the reproducing kernel property $<f, k(\mathbf{u},\cdot)> = f(\mathbf{u})$ in the RKHS, and $<\cdot,\cdot>$ denotes the inner product operation. To ensure the numerical stability of the solution of the objective function, we attempt to minimize $\|f\|^2$, which forms a regularizer in the objective function, where $\|\cdot\|$ denotes the norm operation. Using the reproducing kernel property, we have

$$\|f\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k\left(\mathbf{u}_i, \mathbf{u}_j\right) = \boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}, \quad (2)$$

where $\boldsymbol{\alpha} = [\alpha_1 \cdots \alpha_n]^T$, with the superscript T denoting transpose operation and $\mathbf{K}$ is the symmetrical kernel Gram matrix with elements $K_{ij} = k(\mathbf{u}_i, \mathbf{u}_j)$.

In addition, we make the local consistency assumption or manifold assumption [13] that if two pixels are close and have similar luminance values, then their chrominances should be similar. This means that if $\|\mathbf{u}_i - \mathbf{u}_j\|$ is small, then $|f(\mathbf{u}_i) - f(\mathbf{u}_j)|$ should also be small. This assumption forms another regularizer that helps improve the learning performance. The regularizer can be formulated by spectrum graph theory [16]. According to the theory, the similarity information between points is represented by an adjacency matrix $\mathbf{W}$, in which elements $W_{ij} = 1$ if $\mathbf{u}_i$ and $\mathbf{u}_j$ are close enough; otherwise, $W_{ij} = 0$. We adopted the k-nearest neighbors (KNN) graph to construct matrix $\mathbf{W}$, that is, $W_{ij} = 1$ if $\mathbf{u}_i$ is within the k-nearest neighbors of $\mathbf{u}_j$, or $\mathbf{u}_j$ is within the k-nearest neighbors of $\mathbf{u}_i$. Then, we obtain a diagonal degree matrix $\mathbf{D}$ with elements $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is referred to as the graph Laplacian matrix. According to the manifold assumption,

$$G(f) = 0.5^* \sum_{i,j}^{n} W_{ij}\left[f\left(\mathbf{u}_i\right) - f\left(\mathbf{u}_j\right)\right]^2$$

should be small and can be formulated concisely as $G(f) = \mathbf{F}_U{}^T\mathbf{L}\mathbf{F}_U$, where $\mathbf{F}_U$ is defined as before. Integrating (1) with the two aforementioned regularizers, we obtain the LapRLS objective function:

$$\arg\min_f \left[J = \left\|\mathbf{Y}_U - \mathbf{F}_U\right\|^2 + \lambda_1\|f\|^2 + \lambda_2\mathbf{F}_U^T\mathbf{L}\mathbf{F}_U\right], \quad (3)$$

where λ_1 and λ_2 are constants similar to the Lagrange multiplier. Considering the representer theorem, we have

$$\mathbf{F}_U = \mathbf{K}\boldsymbol{\alpha}. \quad (4)$$

Substituting (2) and (4) into (3), we obtain

$$\arg\min_{\boldsymbol{\alpha}} \left[J = \left\|\mathbf{Y}_U - \mathbf{K}\boldsymbol{\alpha}\right\|^2 + \lambda_1\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} + \lambda_2\boldsymbol{\alpha}^T\mathbf{K}\mathbf{L}\mathbf{K}\boldsymbol{\alpha}\right], \quad (5)$$

which becomes a minimization problem with respect to the coefficient vector $\boldsymbol{\alpha}$. Let $\partial J/\partial\boldsymbol{\alpha} = 0$. We obtain the optimal solution

$$\alpha^* = \left(\mathbf{K} + \lambda_1\mathbf{I} + \lambda_2\mathbf{L}\mathbf{K}\right)^{-1}\mathbf{Y}_U, \quad (6)$$

where $\mathbf{I}$ is an $n \times n$ identity matrix.

3.2 | Improve computational efficiency

It can be seen from (6) that an $n \times n$ matrix must be inverted to calculate optimal $\boldsymbol{\alpha}^*$. The calculation complexity is $O(n^3)$. When the image is large, the calculation imposes a huge computational burden. Moreover, the prediction using (4) needs all the n data points even if only one output $f(\mathbf{u})$ is to be computed. In this section, we introduce a nonlinear mapping method to transform the data points into an m-dimensional space so that linear LapRLS can be conducted. As a result, the complexity depends on the space dimension m instead of the number of data points n. KPCAM was adopted to achieve the nonlinear map $\varphi_z: X \to \mathbb{R}^m$, where $\mathbf{x} \mapsto \varphi_Z(\mathbf{x}) = \mathbf{K}_{ZZ}^{-1/2}\left[k\left(\mathbf{x}, \mathbf{z}_1\right), \ldots, k\left(\mathbf{x}, \mathbf{z}_m\right)\right]^T$, in which $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ is a selected subset of the training point set U and $\mathbf{K}_{ZZ}$ is the kernel Gram matrix whose (i, j)th element is a certain nonlinear kernel function value $k(\mathbf{z}_i, \mathbf{z}_j)$. We use notation $\boldsymbol{\varphi}_i$ to denote $\varphi_z(\mathbf{u}_i)$ for simplicity. The linear LapRLS can be implemented using linear kernel function $k(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j) = \boldsymbol{\varphi}_i^T\boldsymbol{\varphi}_j$ in the RKHS so that the regression model is.

$$f(\boldsymbol{\varphi}) = \sum_{i=1}^{n}\alpha_i k\left(\boldsymbol{\varphi}, \boldsymbol{\varphi}_i\right) = \boldsymbol{\varphi}^T\left(\sum_{i=1}^{n}\alpha_i\boldsymbol{\varphi}_i\right) = \boldsymbol{\varphi}^T\mathbf{w}, \quad (7)$$

where $\mathbf{w} = \sum_{i=1}^{n}\alpha_i\boldsymbol{\varphi}_i$. It is clear that f becomes a linear function with respect to coefficient vector $\mathbf{w} \in \mathbb{R}^m$. Let $\boldsymbol{\Phi} = \left[\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_n\right]$ and $\mathbf{F}_\Phi = [f\left(\boldsymbol{\varphi}_1\right), \ldots, f\left(\boldsymbol{\varphi}_n\right)]^T = \boldsymbol{\Phi}^T\mathbf{w}$. In addition, regularizer $\|f\|^2$ exhibits a new form:

$$\|f\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k\left(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j\right) = \mathbf{w}^T\mathbf{w}. \quad (8)$$

Substituting $\mathbf{F}_{\mathrm{U}}$ in (3) with $\mathbf{F}_{\mathbf{\Phi}}$ and (8) into (3), we get the new form of the objective function

$$\arg\min_{w}\left[J(\mathbf{w})=\left\|\mathbf{Y}_{\mathrm{U}}-\mathbf{\Phi}^{\mathrm{T}}\mathbf{w}\right\|^{2}+\lambda_{1}\mathbf{w}^{\mathrm{T}}\mathbf{w}+\lambda_{2}\mathbf{w}^{\mathrm{T}}\mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^{\mathrm{T}}\mathbf{w}\right].$$

Let $\partial J(\mathbf{w})/\partial\mathbf{w}=0$. The optimal solution is determined as

$$\mathbf{w}^{*}=(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}+\lambda_{1}\mathbf{I}_{m}+\lambda_{2}\mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^{\mathrm{T}})^{-1}\mathbf{\Phi}\mathbf{Y}_{\mathrm{U}}, \tag{9}$$

where $\mathbf{I}_m$ is an $m \times m$ identity matrix.

Compared with (6), (9) needs to calculate the inverse of a matrix of $m \times m$ instead of $n \times n$. The complexity of computing $\mathbf{w}^*$ is $O(m^3)$. When m is significantly smaller than n, the computational efficiency can be improved significantly.

3.3 | Complete codec framework

The proposed complete codec algorithm is summarized as follows. After an image is input to the encoder, color space conversion is first performed to get luminance component Y and chrominance components Cb and Cr. Because the human eye is insensitive to chrominance, we implement chrominance subsampling; the subsampling pattern is 4:2:0, as shown in Figure 1. Y is coded by the standard technique. The code stream of Y is transmitted through the channel or stored in the memory. The code stream is decompressed at both the encoder and decoder into a reconstructed luminance $\hat{\mathrm{Y}}$.

In theory, n in (1) is the number of pixels in an image. In the supervised-learning paradigm, n is also the number of training points. It is computationally very demanding that all the pixels act as training points because LapRLS needs to construct a large adjacency matrix $\mathbf{W}$. Therefore, it is necessary to reduce the computational cost and memory requirements. Selecting only the representative points for training is a natural solution. Reference [9] segmented the image using a normalized cut [17] into regions. From each region, a pixel is randomly selected for subsequent learning. However, the segmentation itself imposes a high computational cost. To achieve a trade-off between the computational cost and representativeness of selected training points, we uniformly subsample n pixel points in $\hat{\mathrm{Y}}$ as training points for supervised learning, which is similar to but simpler than the schemes in [9,11,12]. The training points are sampled from the same grids of $\hat{\mathrm{Y}}$ at the coder and decoder, which ensures that the predictions generated at the two ends are equal. The experimental results show that this sampling scheme is effective despite its simplicity.

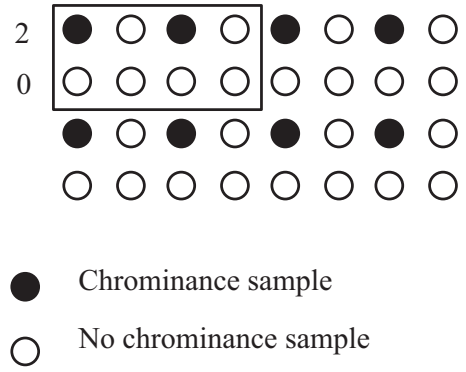

FIGURE 1 Chrominance subsampling of 4:2:0

To implement KPCAM, a set that includes m points, that is, $\mathrm{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\} \subset \mathrm{U}$, must be determined. In general, points representative of U are chosen as $\mathbf{z}_i$'s. In [12], k-means clustering was used to ensure the representativeness of $\mathbf{z}_i$'s. Here, we sought a more feasible and effective scheme. The experimental results show that simple uniform sampling from the training points can determine the $\mathbf{z}_i$'s by which satisfactory predictions can be made. Hence, we resampled the n training points to obtain $\mathbf{z}_i$ ($i = 1, \ldots, m$) for KPCAM, as shown in Figure 2. The two coefficient vectors $\mathbf{w}^*_{\mathrm{Cb}}$ and $\mathbf{w}^*_{\mathrm{Cr}}$ determined by (9) are used to predict Cb and Cr, respectively. They are entropy coded and denoted by $\mathbf{w}^*$s hereinafter for simplicity.

At the decoder, $\hat{\mathrm{Y}}$ is sampled in the same way as that at the encoder to obtain the same training points and representative points $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ for KPCAM, without the need for location information. By means of KPCAM, the mapping $\boldsymbol{\varphi}_i$ of every $\mathbf{u}_i$ can be computed, and then the two chrominance components are predicted by (7). In the end, a color image is recovered by color space conversion into the RGB space. The codec framework of SLBA is shown in Figure 3.

Similar to TEM, the proposed SLBA also has an improved version, called SLBA-Compensation (SLBA-C), a counterpart of TEM-Compensation (TEM-C) in [12], which uses prediction difference images to compensate the regression

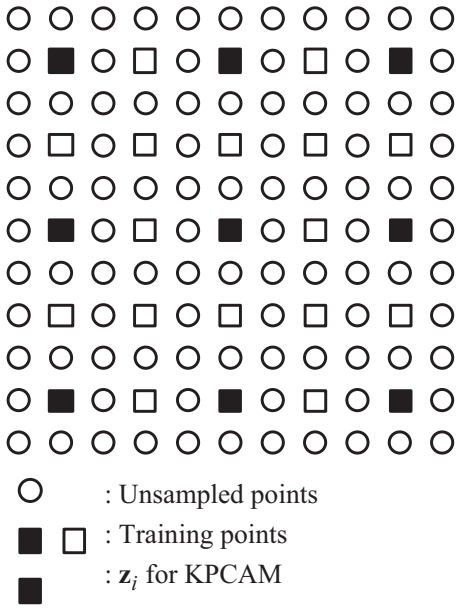

FIGURE 2 Training points and representative points $\mathbf{z}_i$ for KPCAM

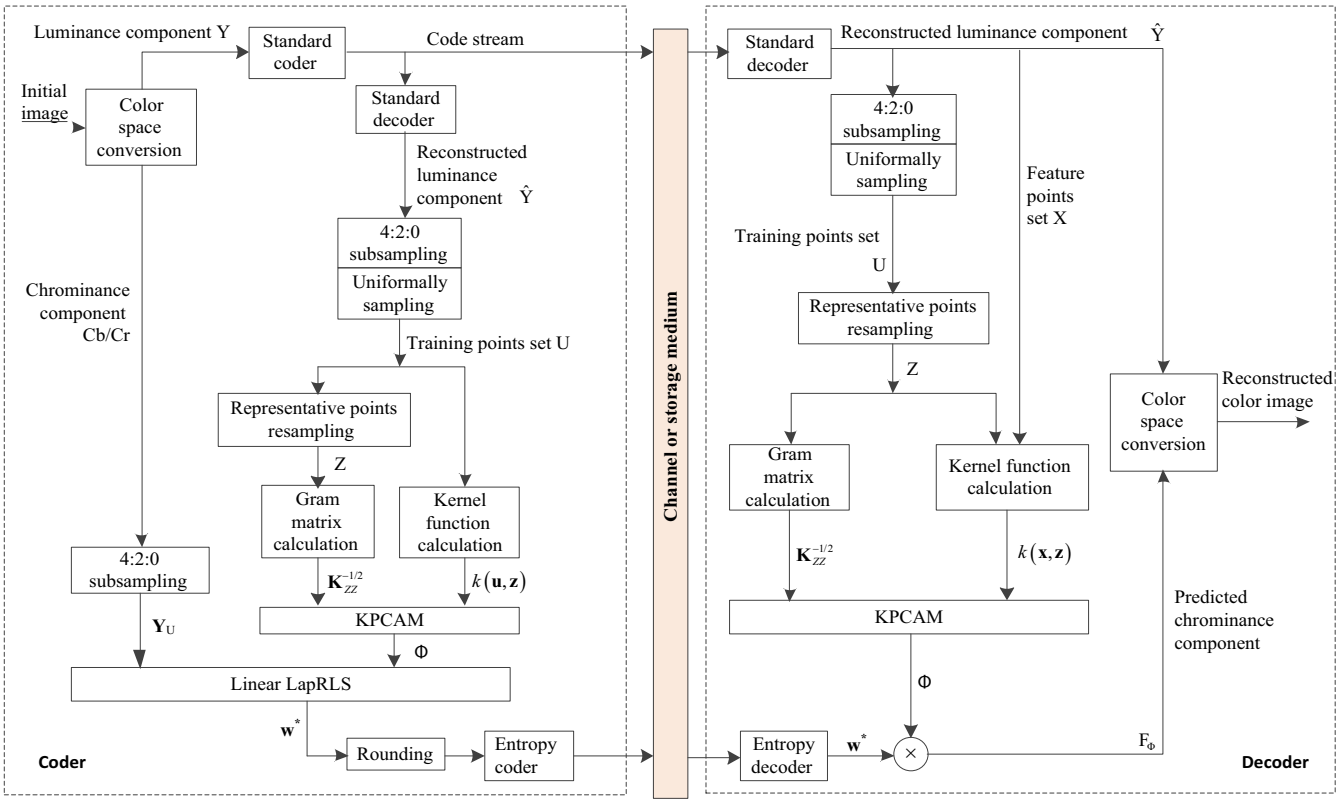

FIGURE 3 Codec framework of SLBA

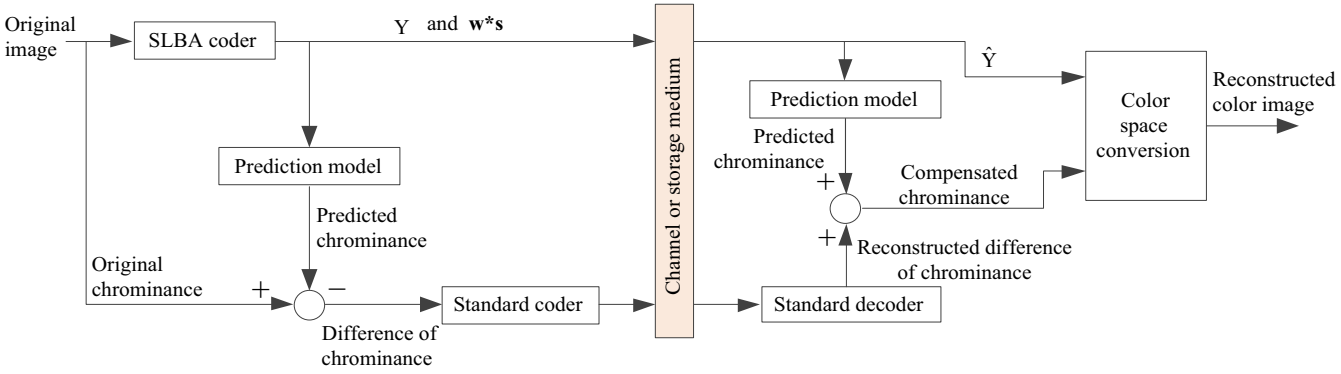

FIGURE 4 Codec framework of SLBA-C

results. In SLBA-C, Cb and Cr are predicted by (7) at the encoder. Chrominance difference images are generated by subtracting the predicted Cb and Cr from the original ones, and then coded by JPEG2000, as in TEM-C. At the decoder, the decoded difference images are added to the predicted Cb and Cr to refine the regression results. The codec framework of SLBA-C is shown in Figure 4.

In summary, SLBA is superior to TEM in three aspects. The first and the most important is that SLBA needs no time-consuming seed pixel point selection. Total error minimization is achieved by supervised learning instead of selecting optimal seed points. Second, chrominance information is conveyed by **w***s that are entropy coded and occupy significantly less bits than are allocated to seeds in TEM. Third, utilizing the insensitivity of the human eye to chrominance, chrominance subsampling is adopted to further reduce computational cost and speed up implementation.

4 | EXPERIMENTS

In this section, we first present the implementation details. Then, we present the experimental results for various parameter configurations and the comparisons between SLBA/SLBA-C and other algorithms. Test images used in our experiments are shown in Figure 5, all of which are 512 × 512

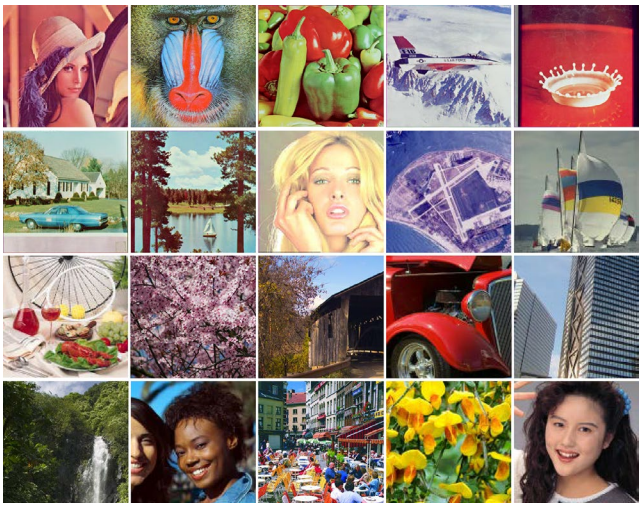

FIGURE 5 Test images used in our experiments. Each image is 512 × 512 in size

in size. The first three images in the top row of Figure 5 are called *Lena*, *baboon*, and *peppers*, respectively.

We measured the prediction accuracy with structural similarity (SSIM) [18] and peak signal-to-noise ratio (PSNR). The PSNR is formulated as PSNR = $10\lg(255^2/\text{MSE})$, where $\text{MSE} = ||I - I'||_F^2/(MN)$ is the mean squared error between the original image I and reconstructed image I'; M and N are the row and column sizes of the images, respectively; and $\|\cdot\|_F$ is the Frobenius norm [19].

4.1 | Implementation details

Here, $n = 8192$ points were sampled uniformly as training points from all the pixel points of the test images used in our experiments, as shown in Figure 2. We also subsampled luminance component Y, which is identical to the chrominance subsampling, to obtain training points for the prediction model. To compare the performance with TEM/TEM-C, which is reportedly the best among the homogeneous learning-based color image compression methods, we set the primary operation parameters to be the same as those in TEM. Specifically, λ_1 and λ_2 in (9) are 0.001 and 0.002, respectively; besides the linear kernel function in (7), we also adopted the Gaussian kernel function as the normal nonlinear kernel function, that is, $k(\mathbf{u}_i, \mathbf{u}_j) = \exp(-||\mathbf{u}_i - \mathbf{u}_j||^2/\sigma^2)$; parameter σ in the expression is also formulated as $\sigma = (0.34 \cdot \sum_{i=1}^{n} \|\mathbf{u}_i - \bar{\mathbf{u}}\|)/n$, where $\bar{\mathbf{u}}$ is the mean of $\mathbf{u}_i$, with $i = 1, \ldots, n$.

We used the KNN graph to construct the adjacency matrix $\mathbf{W}$ mentioned in Section 3.1. Generally, the training pixel point (R, C, I) is the closest to the eight pixel points around it. A KNN graph with $K = 8$ (8-neighbor) should be superior to the case of $K = 4$ (4-neighbor) used in TEM. Tests were performed to verify this conjecture, which were conducted under the conditions

TABLE 1 Average PSNRs (dB) of predicted chrominance in SLBA with $K = 4$ and $K = 8$ in KNN graph for three images

Image name	$K = 4$	$K = 8$
Lena	39.9103	39.9122
Baboon	29.5927	29.5976
Peppers	33.0487	33.0536

where $n = 8192$, $m = 2048$, with the luminance component Y coded losslessly by JPEG2000, and the chrominance components Cb and Cr predicted with $\mathbf{w}$*s rounded off. The average PSNRs of the reconstructed Cb and Cr by SLBA with the KNN graph of different K values are compared in Table 1. We can see that $K = 8$ results in a slightly improved performance compared with that of $K = 4$. Therefore, we set $K = 8$ in all our experiments.

The $\mathbf{w}$*s determined by (9) are fractional. To reduce the bit overhead, all the entries of $\mathbf{w}$*s are rounded off and then Huffman coded to further exploit the statistical redundancy.

4.2 | Impact of m on performance of SLBA/SLBA-C

The number of representative points in set Z used by KPCAM, denoted by m, impacts both the prediction accuracy and time consumed by SLBA/SLBA-C. PSNRs were calculated for the predicted Cr and Cb, and then the average PSNR was presented. To explore the impact of different m values, we compared the PSNRs of the reconstructed chrominance and the SSIMs of the reconstructed images *Lena*, *baboon*, and *peppers*. The metrics were evaluated under the condition that luminance and chrominance differences are coded losslessly by JPEG2000.

The average PSNR and SSIM values are shown in Figure 6A and 6B, respectively. From the two figures we can see that the chrominance prediction quality measured by PSNR and SSIM increases with m in SLBA. This is because the regression model in (7) is of more generality when $\mathbf{w}$*s are obtained by the larger set Z. However, when m exceeds 1024, the PSNR and SSIM curves become flat. This is because an excessive m leads to information redundancy. Figure 6 also shows that although the chrominance predicted by SLBA is inaccurate when the value of m is small, the chrominance quality by SLBA-C remains nearly constant. This is because prediction errors can be compensated even when they are large owing to the small value of m. The quality of the reconstructed chrominance by SLBA-C depends mainly on the compression ratio specified for the luminance component and the chrominance prediction error in the JPEG2000 coder.

We also tested the time consumed by SLBA-C at various values of m, for example, 128, 256, 512, 1024, and 2048. Figure 7 depicts the time consumed by SLBA-C vs m. We can see that the time consumed increases almost linearly with m. For the trade-off between regression performance and running speed, we set $m = 1024$ in later experiments.

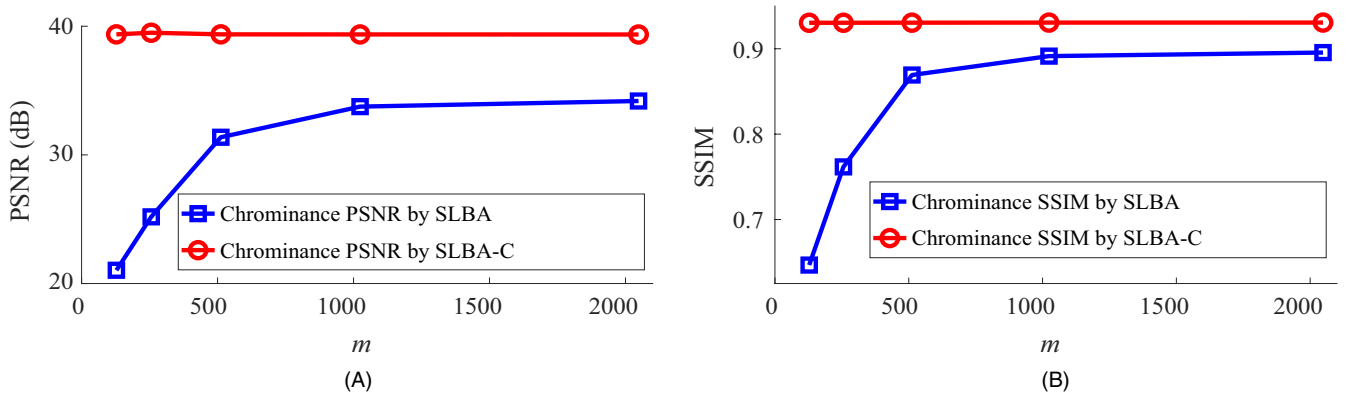

FIGURE 6 (A) Average chrominance PSNR and (B) average SSIM at various values of m

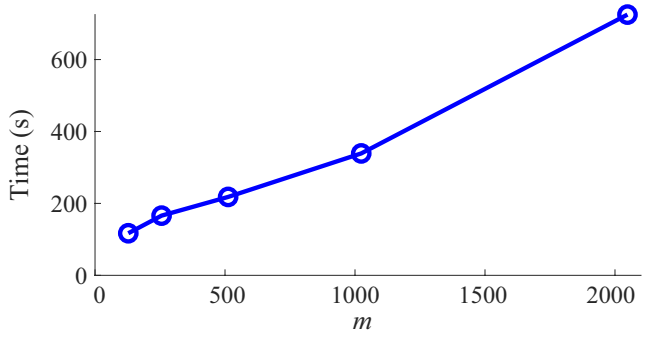

FIGURE 7 Time consumed by SLBA-C codec vs m

4.3 | Impact of compression ratio of Y component

In SLBA/SLBA-C, chrominance components Cb and Cr are predicted by the regression model and then compensated in SLBA-C, while the luminance component Y is still compressed by the standard method. The compression ratio of Y (CR_Y) impacts the predicted Cb and Cr. Figure 8 depicts the PSNR curves of the reconstructed chrominance vs m when CR_Y is 1, 25, and 50. We can see that m is dominant, though CR_Y also impacts the reconstructed chrominance quality. Generally, the smaller the CR_Y is, the larger the PSNR is, which is particularly obvious for *baboon*. This is so because Y with a better quality can keep the manifold assumption; thus, the regression model yields more accurate predictions.

4.4 | Impact of compression ratio of chrominance difference image on colorization quality

Chrominance components Cb and Cr have equal significance; therefore, the two chrominance difference images are compressed with the same ratio CR_{diff}. Experiments were also designed to explore the impact of CR_{diff} on colorization quality in SLBA-C, where m was fixed to be 1024. The average PSNR of the reconstructed Cb and Cr was evaluated at various compression ratios of Y (CR_Y) and chrominance difference (CR_{diff}) on images *Lena*, *baboon*, and *peppers*.

The PSNR vs CR_Y and CR_{diff} curves are shown in Figure 9. It is clear that the reconstructed chrominance is heavily dependent on CR_{diff}, while being slightly dependent on CR_Y. This result is straightforward because the regression model depends mainly on the value of m rather than the fidelity of Y, as shown in Figure 8; however, the chrominance compensation depends mainly on the accuracy of the difference images. Therefore, the smaller the CR_{diff} is, the more accurate the difference images are, and the better the reconstructed chrominance is. These results imply that a higher bit overhead should be allocated to chrominance difference than to luminance for the purpose of colorization.

4.5 | Comparison of colorization quality

Because TEM/TEM-C is currently considered the best learning-based compression method of its kind, we compared SLBA/SLBA-C with only TEM/TEM-C in terms of the prediction quality for chrominance measured by PSNR. The bits consumed by chrominance were used for coding coefficient vectors $\mathbf{w}$*s, each of which includes m entries. D_{Cr} and D_{Cb} represent the overhead of chrominance difference images coded by JPEG2000 in SLBA-C. Apparently, D_{Cr} and D_{Cb} are 0 for SLBA. Consequently, the bits consumed by chrominance are computed as $2m \times \text{bpw} + D_{Cb} + D_{Cr}$ for SLBA-C, where bpw is the bits per entry of $\mathbf{w}$*s. In TEM-C, the bits consumed by chrominance were $48l + D_{Cb} + D_{Cr}$, where l is the number of color seeds. Note that bpw in SLBA/SLBA-C ranges from 6 to 9, which is significantly below 48. Hence, SLBA, particularly SLBA-C, has significantly better RD performance than that of TEM/TEM-C. For our experiments, m was fixed to be 1024; for SLBA, the bit rate varied with CR_Y. For SLBA-C, CR_Y was fixed to be 100, and the bit rate varied with CR_{diff}.

Figure 10 depicts the plot of average chrominance PSNR vs the bit rate, which gives the trends in the colorization

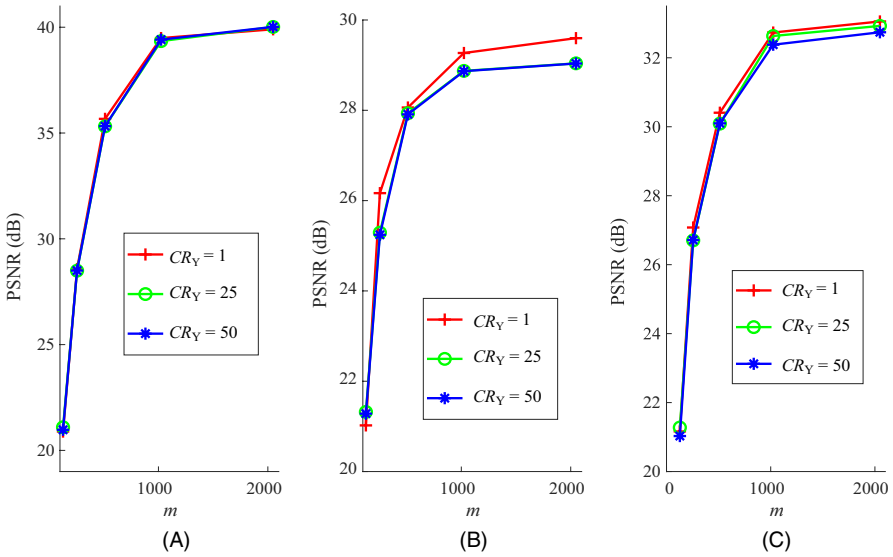

FIGURE 8 PSNR of reconstructed chrominance vs m and CR_Y for images (A) *Lena,* (B) *baboon,* and (C) *peppers*

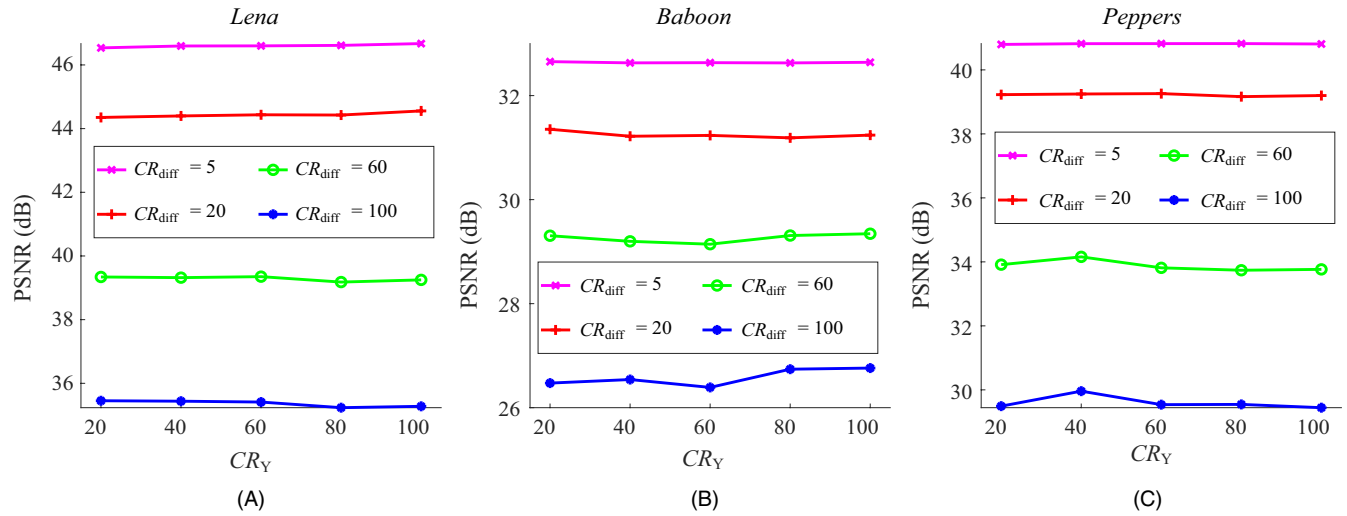

FIGURE 9 Colorization quality (PSNR) by SLBA-C vs CR_Y and CR_{diff} on (A) *Lena*, (B) *baboon*, and (C) *peppers*

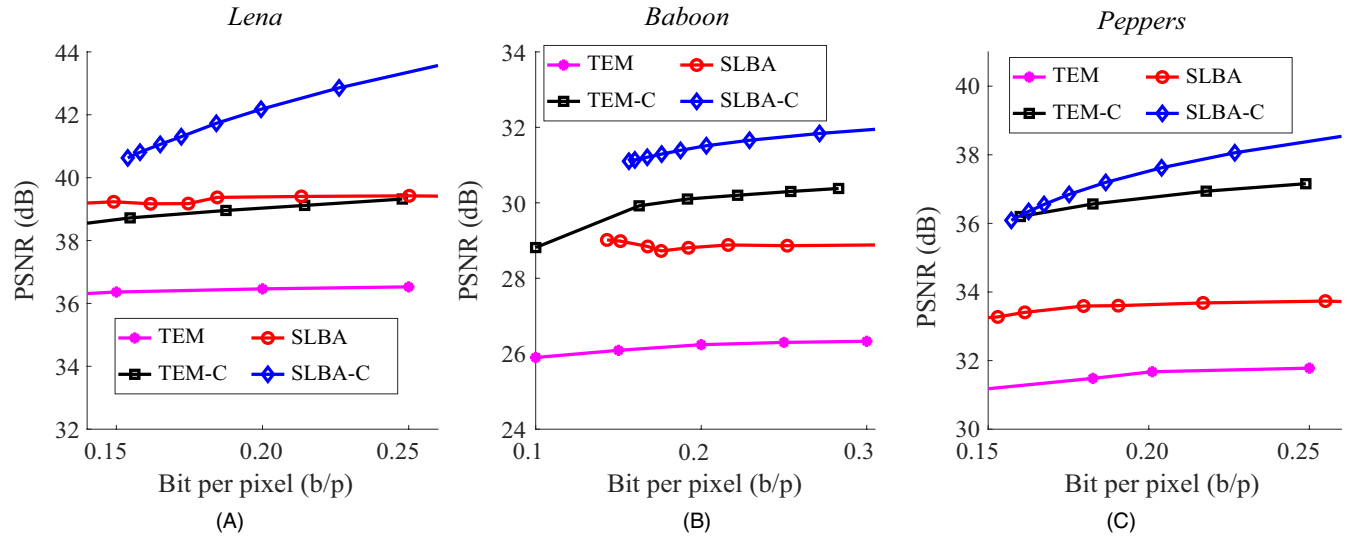

FIGURE 10 Colorization quality comparison of different methods on (A) *Lena*, (B) *baboon*, and (C) *peppers*

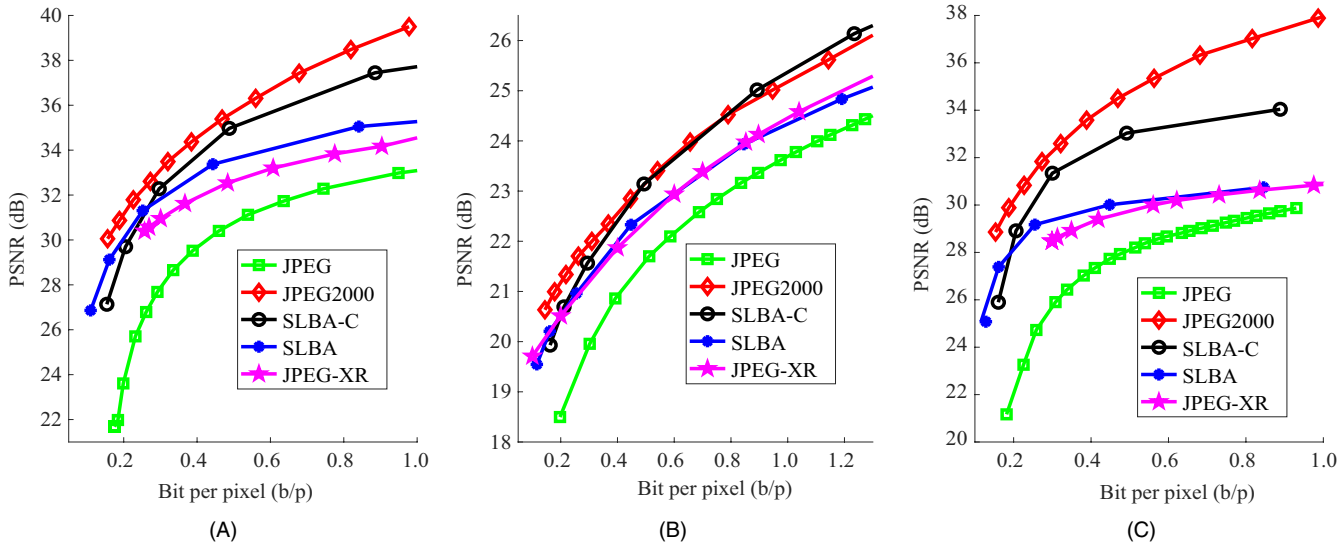

FIGURE 11 Comparison of the average PSNRs over R, G, and B components of the reconstructed images by SLBA, SLBA-C, JPEG, JPEG2000, and JPEG-XR on test images (A) *Lena*, (B) *baboon*, and (C) *peppers*

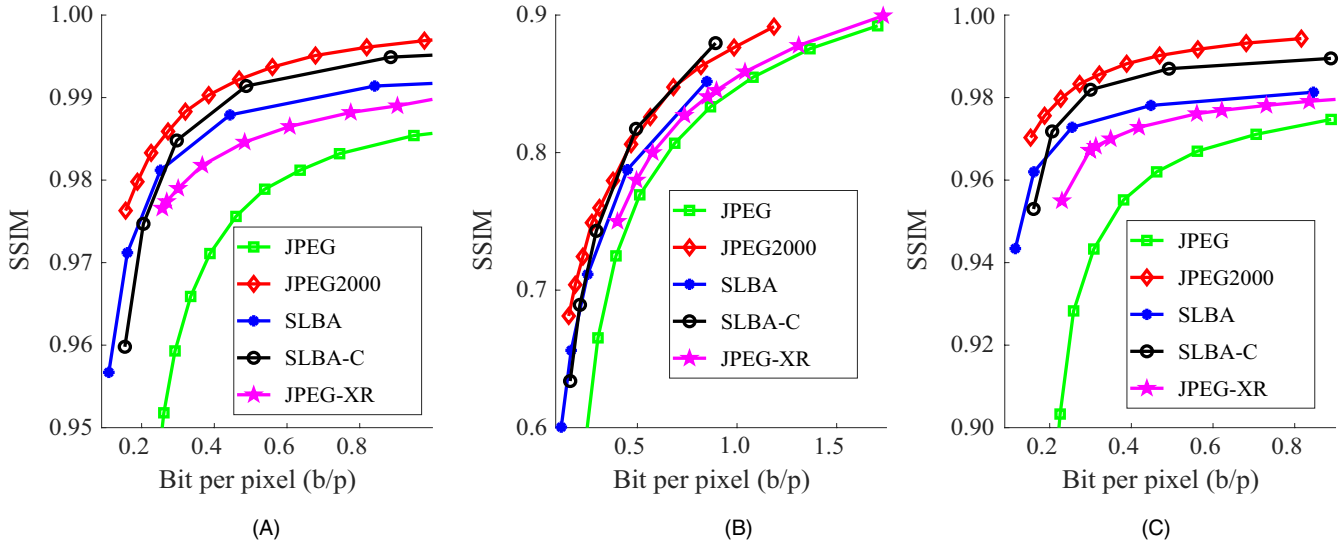

FIGURE 12 Comparison of SSIM of reconstructed images by SLBA, SLBA-C, JPEG, JPEG2000, and JPEG-XR on test images: (A) *Lena*, (B) *baboon*, and (C) *peppers*

quality of different methods. It can be seen that SLBA outperforms TEM, and SLBA-C is significantly better than the other methods. This is attributed to significantly less bit consumption of **w***s achieved by entropy coding and chrominance subsampling.

4.6 | Comparison with image compression standards

In a typical color image compression framework, RGB images are first converted into YCbCr images for coding. JPEG, JPEG2000, and JPEG-XR process each channel of the YCbCr image independently. However, SLBA/SLBA-C encodes Cb and Cr based on luminance Y, and decodes them by the regression model. To make a fair comparison, it is necessary to consider both luminance and chrominance. At the decoder, Cb and Cr were reconstructed by decoded luminance $\hat{Y}$ together with **w***s in SLBA/SLBA-C, and compensated in SLBA-C. Then, the decoded YCbCr image was converted back to the RGB image. PSNRs were computed over each channel of the RGB image before calculating the average PSNR.

The m is still set to be 1024. For SLBA, the bit rate varies with CR_Y. For SLBA-C, the total bits are consumed by the $2m$ entries of **w***s, Y, and D_{Cr}, and D_{Cb}; therefore, it is necessary to present the detailed bit allocation scheme. Although

CR_Y has a slight impact on the prediction accuracy of chrominance (as shown in Figure 8), the experiments showed that CR_Y significantly impacted the average PSNR over R, G, and B components. This result is natural because the R, G, and B components are highly correlated with Y, as shown by the following color space conversion formulas:

$$R = 1.164(Y-16)+1.596(Cr-128),$$

$$G = 1.164(Y-16)-0.813(Cr-128)-0.391(Cb-128),$$

$$B = 1.164(Y-16)+2.018(Cb-128).$$

Therefore, CR_Y is the dominant factor in the reconstructed RGB image by SLBA-C. We set CR_{diff} to be 100, and the bit rate varies with CR_Y in SLBA-C.

We use MATLAB's built-in function *imwrite* for JPEG and JPEG2000 testing, and reference software *jxrlib* [20] for JPEG-XR testing. The bit rates of compression using JPEG and JPEG2000 were changed by modifying the quality factor "*Quality*" and compression ratio "*CompressionRatio*" of the *imwrite* function, respectively; the bit rates using JPEG-XR was changed by "*-q*" of the encoder program JXREncApp.exe.

PSNR and SSIM were adopted to assess the reconstructed images by SLBA, SLBA-C, JPEG, JPEG2000, and JPEG-XR, as presented in Figures 11 and 12, respectively. Unlike TEM that cannot achieve a PSNR comparable to that of JPEG, we can see, from Figures 11 and 12, that SLBA significantly outperforms JPEG and even JPEG-XR at various bit rates in terms of both the objective metric PSNR and subjective metric SSIM. Although SLBA is inferior to JPEG2000, SLBA-C is very close to JPEG2000, particularly in terms of SSIM, and is even superior to JPEG2000 for image *baboon* at a high bit rate, as shown in Figures 11B and 12B. Overall, compared with JPEG2000, SLBA-C is still not good enough. This is because not only Y, Cb, and Cr difference images are coded by JPEG2000 but also the $2m$ coefficients in $\mathbf{w}$*s need to be entropy coded in SLBA-C. Therefore, the bit overhead of SLBA-C is greater than that of JPEG2000 to code the same image and achieve the same reconstructed quality.

From Figures 11 and 12, it is worth noting that SLBA outperforms SLBA-C at bit rates approximately below 0.2 b/p, as is also shown in Figures 13A2, A3, B2, B3, C2, and C3. The reason is the bit allocation. When the bit rate is small, besides the overhead for $\mathbf{w}$*s, the remaining limited bits have to be distributed between the luminance and chrominance difference in SLBA-C. The bits allocated to chrominance difference are tiny, which is the key factor affecting the performance of SLBA-C. Therefore, the effect of chrominance compensation is negligible. However, the bits are entirely concentrated on luminance, which is crucial for the average PSNR over R, G, and B components in SLBA. Consequently, SLBA outperforms SLBA-C. With the bit rate increasing, chrominance compensation works effectively, and Cb and Cr are significantly improved. The

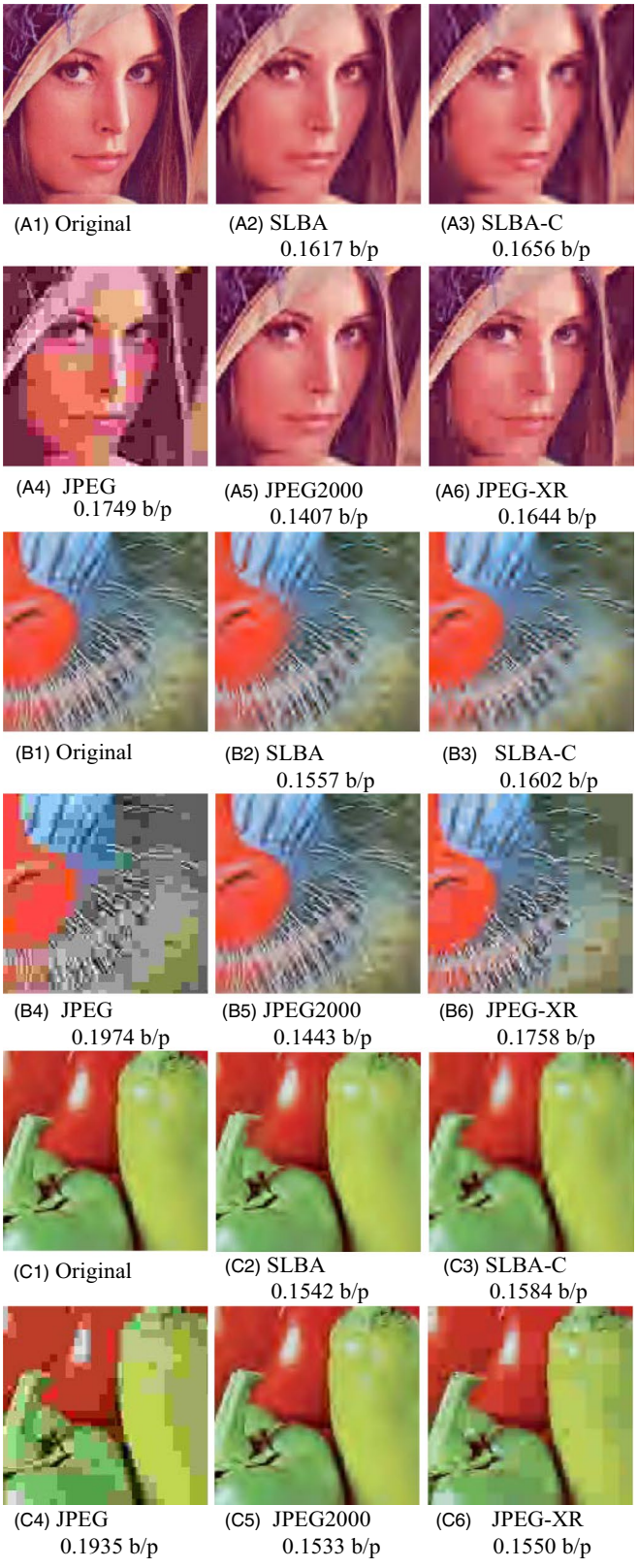

FIGURE 13 Reconstructed images for (A) *Lena*, (B) *baboon*, and (C) *peppers* by SLBA, SLBA-C, JPEG, JPEG2000, and JPEG-XR

R, G, and B components are better reconstructed accordingly. As a result, SLBA-C outperforms SLBA. Figure 13A4, B4, and C4 also shows that, at a low bit rate, JPEG has the worst quality with the most bits per pixel (bpp), with extremely obvious blocking artifacts in the reconstructed images. Figure 13A5, B5, and C5 shows that JPEG2000 always achieves the best quality with the least bpp. JPEG-XR reconstructed the images at low bit rate also show some blocking artifacts although not so obvious as JPEG, as shown in Figure 13A6, B6, and C6. In the proposed SLBA/SLBA-C, such blocking artifacts do not exist.

The SSIM and PSNR plots shown in Figure 14 were generated by interpolating the SSIM and PSNR curves of each image, evaluating the values at the specified bpp points, and then averaging them because the bpp coordinates of different images were different. The average SSIM curves in Figure 14A show that SLBA is superior to JPEG but inferior to JPEG2000. Compared with JPEG-XR, SLBA outperforms it when $bpp < 0.4$ b/p and is very close to it at the higher bpp. SLBA-C outperforms both JPEG and JPEG-XR significantly. Although it is still inferior to JPEG2000, it is close to JPEG2000 when $bpp > 0.4$ b/p. We again find that SLBA performs better than SLBA-C when $bpp < 0.2$ b/p. The average PSNR curves in Figure 14B give approximately the same comparison trend but with more apparent differences between the curves when $bpp > 0.3$ b/p. Based on the data in Figure 14, we demonstrated the average gain in SSIM and PSNR, that is, BD-SSIM and BD-PSNR [21] values, with respect to JPEG in Table 2. BD-SSIM is similar to BD-PSNR and the logarithmic bit rate is not used here. The data in Table 2 reveal that, on average, SLBA-C outperforms SLBA; and they both perform better than JPEG/JPEG-XR but are still inferior to JPEG2000.

TABLE 2 Average gain in SSIM and PSNR with respect to JPEG measured with Bjontegaard metric

SLBA	SLBA-C	JPEG-XR	JPEG-2000
+0.0607	+0.0665	+0.0580	+0.0759
+2.3346	+3.6655	+1.8571	+5.0068

4.7 | Comparison of computation complexities

It is noteworthy that the superiority of SLBA/SLBA-C over its predecessor TEM/TEM-C is not achieved at the cost of higher computational complexity. For TEM, the total number of training points is assumed to be n; if l seed points are to be selected, then TEM has to iterate the selection process for $n - l$ times, as described in Section 2. Although the KPCAM of size m is adopted in TEM, the computational complexity for selecting one point is still as large as $O(nm^2)$. The total computational complexity of the seed selection process is $O((n-l)nm^2)$, which accounts for the main computational burden of TEM. On the contrary, SLBA has no such computational burden. It significantly reduces the computation time compared with that of TEM. We also modified the computation of $\mathbf{w}$*s of (9) as $\mathbf{w}^* = [\mathbf{\Phi J \Phi}^{\mathrm{T}} + \lambda_1 \mathbf{I}_m]^{-1} \mathbf{\Phi Y}_{\mathrm{U}}$, where $\mathbf{J} = \mathbf{I}_n + \lambda_2 \mathbf{L}$, which further reduces the computational cost of the matrix multiplication of $\mathbf{\Phi \Phi}^{\mathrm{T}}$.

It has been reported that previous methods take more than 1 hour to encode an image of size similar to that used in our experiments [12]. Using our experimental platform, TEM took more than 4 hour to encode an image 512×512 in size. Most of the time was spent on seed selection. However, SLBA/SLBA-C does not require seed selection. The typical ($m = 1024$) coding and decoding times required by SLBA-C for the same image are approximately 2.5 and 1.5 minutes, respectively. The coding time of SLBA-C is approximately 1% of that required by TEM [12]. The decoding time of SLBA-C is mainly spent on predicting Cb and Cr. However, compared with JPEG/JPEG2000/JPEG-XR, SLBA and SLBA-C still run slower because they have to compute and code $\mathbf{w}$*s in addition to coding Y and chrominance difference images. Additionally, chrominance prediction is more time-consuming than decoding in standard methods in that the prediction needs to be carried out pixel by pixel.

All the experiments described in this paper were implemented in a MATLAB 2018a environment running on Intel(R) Core i5-3210M CPU with a speed of 2.5 GHz.

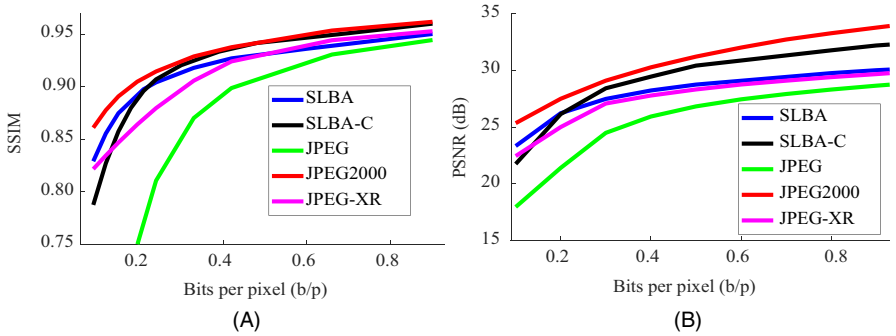

FIGURE 14 (A) Average SSIM (B) and average PSNR on 20 test images by SLBA, SLBA-C, JPEG, JPEG2000, and JPEG-XR

5 | CONCLUSIONS

In this paper, we proposed the SLBA/SLBA-C algorithm for color image compression. Although SLBA evolved from TEM, it outperforms TEM in that it does not require time-consuming active learning for pixel point selection. Furthermore, with faster running speed, the performance of SLBA is superior to TEM considered to be the best method of its kind. This superiority is attributed to achieving total prediction error minimization in the LapRLS objective function and implementing KPCAM on the fixed grids of training points. In addition, chrominance subsampling and entropy coding of $\mathbf{w}^*$s decrease the bit rate and play an important role in the excellent RD performance of SLBA/SLBA-C.

We also explored the factors that impact the performance of SLBA/SLBA-C. m is the leading parameter for chrominance prediction. However, the prediction performance tends to be saturated when m exceeds a certain value, for example, 1024 in our experiments. If m is fixed, CR_Y impacts the prediction accuracy slightly. For SLBA-C, the quality of the reconstructed chrominance depends heavily on CR_{diff} but slightly on CR_Y. However, the average PSNR on R, G, and B components still depends heavily on CR_Y in SLBA-C.

In terms of the implementation speed, there still exists a gap between SLBA/SLBA-C and JPEG/JPEG2000/JPEG-XR. However, it has been shown that SLBA and SLBA-C outperform JPEG and even JPEG-XR at low bit rates in terms of RD performance. In particular, SLBA-C performs close to JPEG2000 in SSIM metric at high bit rates, whereas SLBA outperforms SLBA-C at low bit rates. SLBA and SLBA-C outperform their predecessors [9,11,12] that unified semi-supervised learning and active learning in terms of both running speed and RD performance. However, compared with Lee's [7] and Uruma's methods [8], SLBA and SLBA-C are inferior to them because their methods can perform even better than JPEG2000. This is because SLBA and SLBA-C minimize the prediction error over the training points only, while Lee's and Uruma's methods minimize the colorization error over the whole original image. Therefore, more studies are required to improve and extend this algorithm using a prediction model trained by learning. To further reduce the computational complexity for optimization over all the pixels instead of training points, other alternatives to KPCAM, such as random feature [22], could be explored to improve computational efficiency. The possibility of parallel calculation could also be explored to speed up the proposed algorithm. To extend this research, in the future, we can investigate the possibility of using SLBA/SLBA-C to exploit the interchannel correlation between frames in a video and make comparisons with the methods in [5,6].

ORCID

Xue-Dong Liu https://orcid.org/0000-0002-6598-953X

REFERENCES

1. JPEG Std. ISO/IEC 10918 - 1 and ITU - T.81, *Information Technology: Digital Compression and Coding of Continuous - Tone Still Images: Requirements and Guidelines*, 1993.
2. M. Charrier, D. S. Cruz, and M. Larsson, *JPEG2000, the next millennium compression standard for still images*, in Proc. Int. Conf. Multimedia Comput. Syst., Florence, Italy, June 1999, pp. 131–132.
3. F. Dufaux, G. J. Sullivan, and T. Ebrahimi, *The JPEG XR image coding standard [Standards in a Nutshell]*, IEEE Signal Process. Mag., **26** (2009), no. 6, 195–199 and 204.
4. K. R. Rao, J. J. Hwang, and D. N. Kim, *High Efficiency Video Coding and Other Emerging Standards*, River Publishers, Aalborg, Denmark, 2017.
5. X. Zhang, F. Zou, and O. C. Au, *Chrominance intra-prediction based on inter-channel correlation for HEVC*, IEEE Trans. Image Process. **23** (2014), no. 1, 274–286.
6. K. Zhang et al., *Enhanced cross-component linear model for chroma intra-prediction in video coding*, IEEE Trans. Image Process. **27** (2018), no. 8, 3983–3997.
7. S. Lee et al., *Colorization-based compression using optimization*, IEEE Trans. Image Process. **22** (2013), no. 7, 2627–2636.
8. K. Uruma et al., *Colorization-based image coding using graph Fourier transform*, Signal Process.: Image Com. **74** (2019), 266–279.
9. L. Cheng and S. V. N. Vishwanathan, *Learning to compress images and videos*, in Proc. Mach. Learn., Corvallis, OR, USA, June 2007, pp. 161–168.
10. A. Atkinson, A. Donev, and R. Tobias, *Optimum Experimental Designs With SAS (Series Oxford Statistical Science)*, Oxford Univ. Press, Oxford, U.K., 2007, pp. 151–153.
11. X. He, M. Ji, and H. Bao, *A unified active and semi-supervised learning framework for image compression*, in Proc. IEEE Conf. CVPR, Miami, FL, USA, June 2009, pp. 65–72.
12. C. Zhang and X. He, *Image compression by learning to minimize the total error*, IEEE Trans. Circuits Syst. Video Technol. **23** (2013), no. 4, 565–576.
13. M. Belkin, P. Niyogi, and V. Sindhwani, *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*, J. Mach. Learn. **7** (2006), 2399–2434.
14. X. Liu and J. Yang, *Fast and high efficient color image compression using machine learning*, in Proc. IEEE Adv. Inf. Manag. Commun. Electron. Autom. Contr. Conf., Xi'an, China, May 2018, pp. 470–473.
15. B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2001.
16. F. R. K. Chung, *Spectral Graph Theory*, Am. Math. Soc., Providence, RI, 1997.
17. J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Patt. Anal. Mach. Intell. **22** (2000), no. 8, 888–905.
18. Z. Wang et al., *Image quality assessment: From error visibility to structural similarity*, IEEE Trans. Image Process. **13** (2004), no. 4, 600–612.
19. G. A. F. Seber, *A Matrix Handbook for Statisticians (Wiley Series in Probability and Mathematical Statistics)*, Wiley, Hoboken, NJ, USA, 2008.

20. CodePlex Archive, *Open source implementation of jpegxr*, Available from https://archive.codeplex.com/?p=jxrlib
21. G. Bjontegaard, *Calculation of Average PSNR Differences Between RD Curves*, Document VCEG-M33, ITU-T Q6/16, Austin, TX, USA, 2001.
22. A. Rahimi and B. Recht. *Random features for large-scale kernel machines*, in Proc. Int. Conf. Neural Inf. Process. Syst., Vancouver, Canada, Dec. 2008, pp. 1177–1184.

AUTHOR BIOGRAPHIES

Xue-Dong Liu received his bachelor's degree in engineering from the School of Engineering, North China University of Technology, Beijing, China, in 1998; master's degree in communications and information system from the School of Information Engineering, Wuhan University of Technology, Wuhan, Hubei, China, in 2003; and doctorate degree in control science and engineering from the Institute of Image Recognition & Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2009. Since 2003, he has been working at the School of Information Engineering, Wuhan University of Technology, Wuhan, Hubei, China, where he is now an associate professor. His main research interests are digital image processing, video compression, and machine learning.

Meng-Yue Wang received her bachelor's degree in engineering from North China University of Water Resources and Electric Power, Zhengzhou, Henan, China, in 2018, and is currently pursuing her master's degree at the School of Information Engineering, Wuhan University of Technology, Wuhan, Hubei, China. Her research interests include machine learning and digital image compression.

Ji-Ming Sa received his bachelor's degree in control engineering from the School of Automotive Engineering, Wuhan University of Technology, Wuhan, China, in 1995; master's degree in control engineering from the School of Automotive Engineering, Wuhan University of Technology, Wuhan, Hubei, China, in 2001; and doctorate degree in mechatronics engineering from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2007. Since 2008, he has been working at the School of Information Engineering, Wuhan University of Technology, Wuhan, Hubei, China, where he is now an associate professor. His main research interests are digital image processing and machine learning.