Khalid El Asnaoui*

Image Compression Based on Block SVD Power Method

https://doi.org/10.1515/jisys-2018-0034
Received January 16, 2018; previously published online April 2, 2019.

Abstract: In recent years, the important and fast growth in the development and demand of multimedia products is contributing to an insufficiency in the bandwidth of devices and network storage memory. Consequently, the theory of data compression becomes more significant for reducing data redundancy in order to allow more transfer and storage of data. In this context, this paper addresses the problem of lossy image compression. Indeed, this new proposed method is based on the block singular value decomposition (SVD) power method that overcomes the disadvantages of MATLAB's SVD function in order to make a lossy image compression. The experimental results show that the proposed algorithm has better compression performance compared with the existing compression algorithms that use MATLAB's SVD function. In addition, the proposed approach is simple in terms of implementation and can provide different degrees of error resilience, which gives, in a short execution time, a better image compression.

Keywords: Image compression, singular value decomposition, block SVD power method, lossy image compression, PSNR.

1 Introduction

Singular value decomposition (SVD) is a generalization of the eigen-decomposition used to analyze rectangular matrices. SVD plays an important role in many applications, and it is the most useful tool of linear algebra with several applications including image compression [23]; mathematical models in economics, physical processes, and biological processes; data mining applications; search engines to rank documents in very large databases, including the Web; image processing applications; etc. In addition, the use of SVD in image compression has been widely studied [2, 13, 25, 27]. In this paper, we will study SVD applied to image compression.

Image compression is a type of data compression that involves encoding information in images using fewer bits than the original image representation. The main idea of image compression is reducing the redundancy of the image and transferring data in an efficient form [15]. Image compression takes an important place in several domains, like web design. In fact, maximally reducing an image allows us to create websites faster and saves bandwidth for users; it also reduces the bandwidth of the servers, and thus saves time and money. When talking about compression, we generally take into account two aspects: image size in pixels and the degree of compression. There is also another aspect: pixel depth or bit cost to represent each pixel. The nature of the image also plays a significant role. The main goal of such a system is to reduce the storage quantity as much as possible while ensuring that the decoded image displayed in the monitor can be visually similar to the original image as much as it can be.

Our main goal in this work is to find an algorithm that can involve encoding information in images using fewer bits than the original image representation. Toward this end, we developed and tested our algorithm for image compression based on various and real images.

***Corresponding author: Khalid El Asnaoui,** Complex Systems Engineering and Human Systems, Mohammed VI Polytechnic University, Lot 660, Hay Moulay Rachid, Ben Guerir 43150, Morocco, e-mail: Khalid.ELASNAOUI@um6p.ma.
https://orcid.org/0000-0003-4260-6898

The rest of this paper is structured as follows. We briefly introduce a review of previous related works in Section 2. Section 3 deals with the algorithm of SVD function. Section 4 describes the proposed system for image compression. The performance evaluation of the proposed algorithm is reported in Section 5. Finally, conclusions are drawn in Section 6.

2 Related Work

In recent years, numerous image compression schemes and their applications in image processing have been proposed [18–20]. In this section, a brief review of some important contributions from the existing literature is presented.

In general, there are two approaches for image compression: lossy or lossless [16, 26]. A lossless compression is a kind of image compression method that allows no loss of data, and which retains the full information needed to reconstruct the original image. This type of compression is also known as entropy coding because of the fact that a compressed signal is generally more random than the original one and the patterns are removed when a signal is compressed. Lossless compression can be very useful for exact reconstruction of images. The compression ratio provided by this kind of method is not sufficiently high to be truly used in image compression. Lossless image compression is particularly useful in image archiving as in the storage of legal or medical records. The lossless image compression methods include run-length coding, bit-plane coding, Huffman coding [1], LZW (Lempel-Ziv-Welch) codes, and entropy coding.

Lossy compression is another type of image compression technique in which the original signal cannot be exactly reconstructed from the compressed data. The reason behind this is that much of the detail in an image can be discarded without greatly changing the appearance of the image. In lossy image compression, even very fine details of the images can be lost, but ultimately, the image size is drastically reduced. Lossy image compressions are useful in many applications such as broadcast television, video conferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. Among methods for lossy compression, we find fractal compression [14], transform coding, Fourier-related transform, discrete cosine transform [5, 17], and wavelet transform.

Generally, SVD is a lossy compression technique that achieves compression by using a smaller rank to approximate the original matrix representing an image. Furthermore, lossy compression yields good compression ratio compared with lossless compression while lossless compression gives good-quality compressed images [10].

When we give the definition of lossless or lossy methods, it is necessary to clarify that near-lossless algorithms are theoretically lossless; however, they may suffer from numerical floating-point accuracy reconstruction issues.

According to the state of the art, several works suggested using the SVD with other compression methods or with variations of SVD [e.g. Shuffled Singular Value Decomposition (SSVD) [22], Joint Singular Value Decomposition (JSVD), and K-Singular Value Decomposition (KSVD)]. Awwal et al. [3] presented a new compression technique using SVD and the wavelet difference reduction (WDR). The WDR is used for further reduction. This technique has been tested with other techniques such as WDR and JPEG2000, and gives a better result than the other techniques. Furthermore, using WDR with SVD enhances the peak signal-to-noise ratio (PSNR) and compression ratio.

A technique based on wavelet-SVD, which uses a graph coloring technique in the quantization process, is presented in Ref. [21]. This technique worked well and enhanced the PSNR and compression ratio. The generated compression ratio by this work ranged between 50% and 60%, while the average PSNR ranged between 40 and 80 db.

Ranade et al. [22] suggested a variation of SVD-based image compression. This approach is a slight modification to the original SVD algorithm, which gives much better compression than the standard compression using the SVD method. In addition, it performs substantially better than the SVD method. Typically, for any given compression quality, this approach needs about 30% fewer singular values and vectors to be retained.

Bryt and Elad [6] suggested a method for compressing facial images using the KSVD algorithm, which is an algorithm for training over complete dictionaries that lead to sparse signal representations. This method was applied to grayscale images and shown to be most efficient, surpassing the JPEG2000 performance significantly. This work is extended in Ref. [7].

Doaa and Chadi [9] proposed block truncation coding (BTC), it is an image compression method proposed by Delp and Mitchell [8–24]. It is one of the simplest and easiest image compression algorithms, and also an efficient image coding method that has been adopted to obtain the statistical properties of a block in the compressed image.

The technique given by El Abbadi et al. [10] proposes to use SVD and moment preserving quantizer-BTC. Using this method, the input image is compressed by reducing the image matrix rank, by using the SVD process and then the result matrix compressed by using BTC.

The technique presented by El Asnaoui et al. [11] introduces two new approaches. The first one is an improvement of the BTC method that overcomes the disadvantages of the classical BTC, while the second one describes how to obtain a new rank of SVD method, which gives a better image compression. This method is extended in Ref. [12].

Following the same objective of image compression using SVD, the most important problem is which K rank to use for giving a better image compression. For this reason, El Asnaoui et al. [11] proposed a new technique that gives a maximum value for K, which provides better compression.

To the best of our knowledge, this is the first work to suggest an image compression method based on the block SVD power method.

3 Singular Value Decomposition (SVD)

As we have earlier mentioned, the main goal of studying the SVD of an image (matrix of $m \times n$) is to create approximations of an image using the least amount of the terms of the diagonal matrix in the decomposition. This approximation of the matrix is the basis of image compression using SVD, as images can be viewed as matrices with each pixel being an element of a matrix.

In mathematics, the linear algebra of the SVD method of a matrix is an important tool for factorization of real or complex rectangular matrices.

Let us say we have a matrix A with m rows and n columns, with rank r and $r \leq n \leq m$. Then, A can be factorized into three matrices: U, $\sum$, and V^T such that [17] $A = U \sum V^T$, where U and V are orthogonal, $\sum$ is a diagonal matrix, and $U \sum V^T$ is the closest rank K approximation to A (see Figure 1).

The purpose of transforming matrix A into $U \sum V^T$ is to approximate matrix A by using far fewer entries than in the original matrix. By using the rank of a matrix, we remove the redundant information (the dependent entries). SVD can be used to provide the best lower-rank approximation to matrix A and thus be used for image compression.

- Theorem

Let $A \in M_{m*n}(\mathbb{R})$ be two-unit matrices $U \in M_m(\mathbb{R})$, $A \in M_{m*n}(\mathbb{R})$ and a diagonal matrix $\sum \in M_{m*n}(\mathbb{R})$.

Then, there is a factorization of the form

$$A = U \sum^{V^T},$$

where

$$\sum = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \text{ and } D = diag(\sigma_1, \sigma_2, \ldots, \sigma_r).$$

$* \sigma_i$ are the singular values of A such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$.

$* r$ is the maximum rank of the matrix A such that $r \leq min(n, m)$.

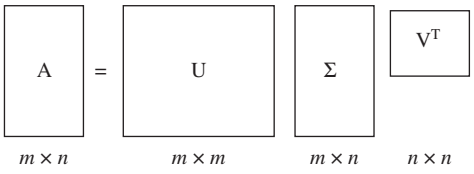

Figure 1: Decomposition of Matrix A.

3.1 SVD vs. Memory

First, we need to know how much memory an original image I requires.

For the image of $m \times n$ pixels in grayscale, the memory has to store $m \times n$ values, one value for each pixel: $I_M = m \times n$.

SVD is composed of three matrices: U, $\sum$, and V. Originally, U is a matrix of $m \times m$; however, we only want the first columns of K. Then, $U_M = m \times K$. Similarly, we only want the first K columns of V (which become the V^T lines); thus, we only need to store V as an $n \times K$ matrix with values $n \times K$, then: $V_M = n \times K$.

Finally, because we exploit the first K columns of U and V, we need only the first K singular values: $\sum_M = K$. Now, it is possible to calculate the total number of values needed to store this K rank. This rank was determined by using this formula [11]:

$$K = \frac{m \times n}{m + n + 1}, \tag{1}$$

where m and n are the size of original image.

3.2 Algorithm of SVD

The main idea of this section is to present two algorithms: the first one is MATLAB's SVD function, while the second one describes how to obtain a new SVD using the block SVD power method.

3.2.1 Algorithm of MATLAB's SVD Function

Input: $A \in M_{m \times n}(\mathbb{R})$

Output: $A = U_{m \times m} \times \sum_{m \times n} \times {V_{n \times n}}^T$, where

$$\begin{cases} U \text{ and } V \text{ are unitary matrices} \\ \sum = diag(\sigma_1, \sigma_2, \ldots, \sigma_p) \\ \text{where} \quad p = min(m, n) \\ r = rank(A) \\ \text{and } \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \\ \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_p = 0 \end{cases}$$

$[U, \sum, V] = svd(A)$ (MATLAB notation is used).

3.2.2 Block SVD Power Method [4]

The main goal in this section is to give a block iterative algorithm that computes the SVD. The idea is based on the technique used in the block power method. From a block-vector $V^{(0)} \in \mathbb{R}^{m \times s}$, we construct two block-vector sequences, $V^{(k)} \in \mathbb{R}^{m \times s}$ and $U^{(k)} \in \mathbb{R}^{n \times s}$, which converge, respectively to the s first right and left singular vectors corresponding to singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s$.

Input: A matrix $A \in \mathbb{R}^{n\times m}$, a block-vector $V = V^{(0)} \in \mathbb{R}^{m\times s}$
and a tolerance *tol*
Output: An orthogonal matrix
$U = [u_1, u_2, \ldots, u_s] \in \mathbb{R}^{n\times s}$,
$V = [v_1, v_2, \ldots, v_s] \in \mathbb{R}^{m\times s}$
and a positive diagonal matrix
$\sum = diag(\sigma_1, \sigma_2, \ldots, \sigma_s)$
such that $AV = U\sum$
While $(err \succ tol)$ **do**
$AV = QR$ (factorization QR),
$U \leftarrow Q(:, 1 : s)$
(the s first vector columns of Q)
$A^T U = QR$,
$V \leftarrow Q(:, 1 : s)$ and $\sum \leftarrow R(1 : s, 1 : s)$
$err = \|AV - U\sum\|$
End.

4 Proposed Lossy Image Compression Technique

As it is mentioned in the related work, the SVD-based compression is lossy due to the nature of the process.

The contribution of this paper is the introduction of the concept of application of the block SVD power method to image compression, as the main idea of image compression is reducing the redundancy of the image and transferring data in an efficient form. As we have earlier mentioned, this work is the first one that proposes an application of the block SVD power method to image compression. Most of the methods focus on other methods and other variations of SVD. Moreover, our method is novel and efficient for solving our problem. It is general, and many other computer visions can benefit from using it.

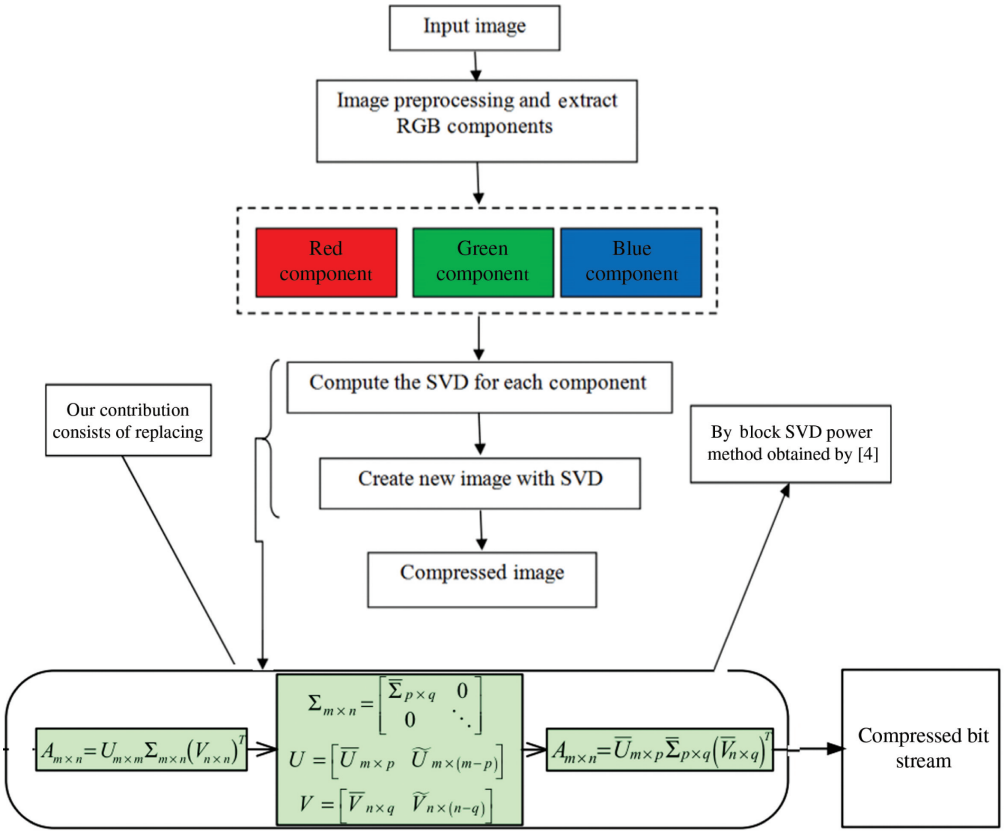

Figure 2: Color Image Pre-processing Using SVD.

In this section, we propose our contribution in which we integrate the block SVD power method and adopt it to create an algorithm that compresses an image. Figure 2 shows the main pipeline of the proposed method.

When the SVD is applied to an image, it is not compressed, but the data take a form in which the first singular value has a great amount of image information. With this, we can use only a few singular values to represent the image with little differences from the original. The input image can be a color image with RGB color components or may be a grayscale image. Furthermore, for creating a new image with MATLAB's SVD function, as indicated in Figure 2, we use the following:

$$I_{comp} = U(:, 1:K) \times \sum(1:K, 1:K) \times \left(V(:, 1:K)^T\right). \tag{2}$$

Our contribution in this paper is to set up a new algorithm for image compression that overcomes some inconveniences encountered in existing methods that use MATLAB's SVD function. Our modification consists of computing the SVD for each component, in which the entries in image I are computed using the block SVD power method obtained by Bentbib and Kanber [4] instead of MATLAB's SVD function, and keeps the K rank determined by Eq. (1).

Thus, for the same compression, we have a better quality. We also provide a heuristic argument to justify our experimental finding.

In the next section, the experimental results are reported. The results clearly show the superiority of the proposed lossy image compression technique over MATLAB's SVD function and some different compression techniques in the state of the art.

5 Experimental Results

Our work is aimed at image compression. For this purpose, our experiments were performed on several images available on Windows 7 Professional and numerical examples. Simulations were done in MATLAB 2009a using a personal computer with Processor: Intel(R) Core (TM) 2 CPU T5200 @ 1.60 GHz, 1.60 GHz, 2Go RAM running on a Microsoft Windows 7 Professional (32-bit). In addition, the results and discussion of the proposed method are given in this section.

5.1 Parameters for Comparison

To evaluate the performance of the proposed method, the quality of the image is estimated using several quality measurement variables, like mean square error (MSE), root MSE (RMSE), and PSNR. These variables are signal fidelity metrics and do not measure how viewers perceive the visual quality of an image [3, 21, 22].

5.1.1 Compression Ratio

The degree of data reduction obtained by a compression method can be evaluated using the compression ratio (Q_{comp}), which is defined by the following formula:

$$Q_{comp} = \frac{size\ of\ original\ image}{size\ of\ compressed\ image}. \tag{3}$$

5.1.2 MSE

The MSE for two $m \times n$ monochrome images I and J, where one of the images is considered noisy approximation of the other, is defined as follows:

$$e_{mse} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - J(i,j)]^2. \tag{4}$$

5.1.3 RMSE

RMSE is one of many ways to quantify the difference between values implied by an estimator and the true value of the quantity being estimated. The use of RMSE is very common, and it makes an excellent general-purpose error metric for numerical predictions. This parameter is given as follows:

$$RMSE = \sqrt{e_{MSE}}. \tag{5}$$

A lower RMSE indicates that the model is better fitting.

5.1.4 PSNR

PSNR is measured in decibels (dB), and is only meaningful for data encoded in terms of bits per sample or bits per pixel.

For example, an image with 8 bits per pixel contains integers from 0 to 255. PSNR is given by the following equation:

$$PSNR = 10\log_{10}\log_{10}\frac{(2^B - 1)^2}{e_{MSE}}. \tag{6}$$

A high PSNR value indicates that there is less visual degradation in the compressed image.

5.2 Numerical Examples

The results of the image compression depend strongly on the goodness of the algorithm to compress an image. To illustrate it, first we checked if the chosen block SVD power method correctly detects the relative errors occurring when computing the singular values and CPU time. For this purpose, we did several tests where we chose some numerical examples.

We have compared and tested in this section the numerical results obtained by the algorithm [4] with MATLAB's SVD function. Toward this end, let $A \in \mathbb{R}^{n*m}$ be a rectangular matrix defined as follows: $A \in Q \sum U^T$, where Q and U are random orthogonal matrices. We give below the relative errors occurring when computing the singular values and the CPU time. The started block-vector in the algorithm [4] is given by $V = V(0) = eye(m, s)$ (Matlab notation). The results are given from the algorithm [4] after only at most $K = 1$ iteration. We have stopped the algorithm [4] whenever the error of the reduction, $err = \|AV - U \sum\|$, is smaller than that achieved by MATLAB's SVD function (see Figures 3 and 4).

Example 1: Let

$$\sum = diag\left(\left[10^4, 10^4, 10^{-11}, 10^{-11}, 10^{-12}, 10^{-12}, 10^{-13}, 10^{-13}, 10^{-14}, 10^{-14}\right]\right)$$

$m = 1000$; n= 1000; s = rank (A) = 10; after only $K = 1$ iteration, we have obtained:

	MATLAB's SVD function	Algorithm [4]
Error $\|AV - U \sum\| =$	2.3936e−011	9.1089e−012
CPU time (s)	19.3789	0.2830

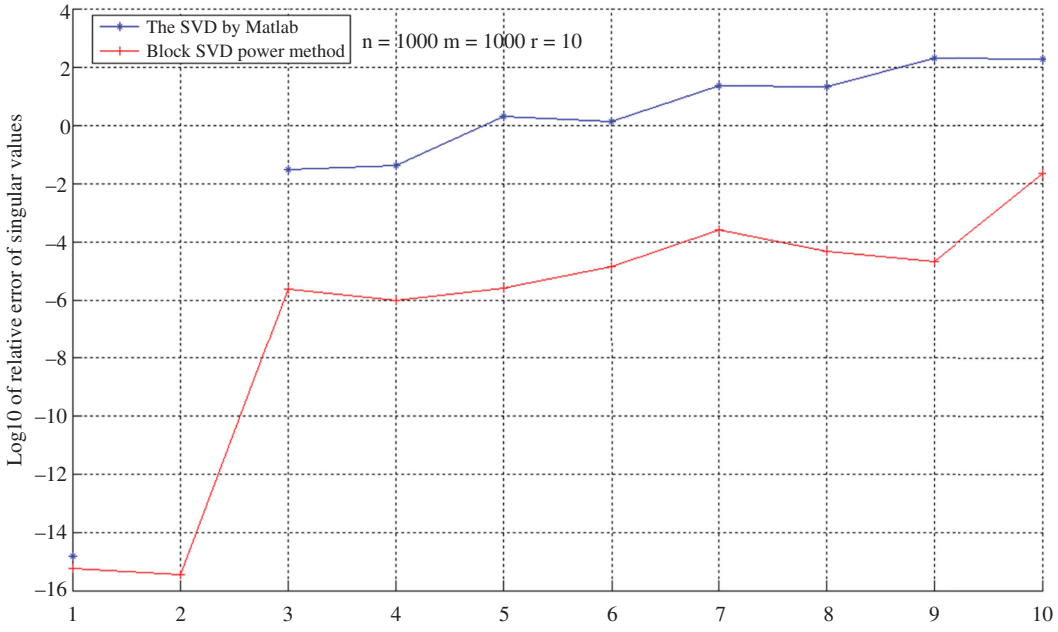

Figure 3: Relative Errors Occurring When Computing the Singular Values.

Example 2: Let

$$\sum = diag\left(\left[10^3, 10^3, 10^3, 10^{-12}, 10^{-12}, 10^{-13}, 10^{-13}, 10^{-13}, 10^{-13}, 10^{-13}, 10^{-13}, 10^{-13}\right]\right)$$

$m = 1000$; $n = 1000$; $s = \text{rank (A)} = 12$; after only $K = 1$ iteration, we have obtained:

	MATLAB's SVD function	Algorithm [4]
Error$\|AV - U\sum\| =$	2.6714e−012	1.2523e−012
CPU time (s)	21.0487	0.5318

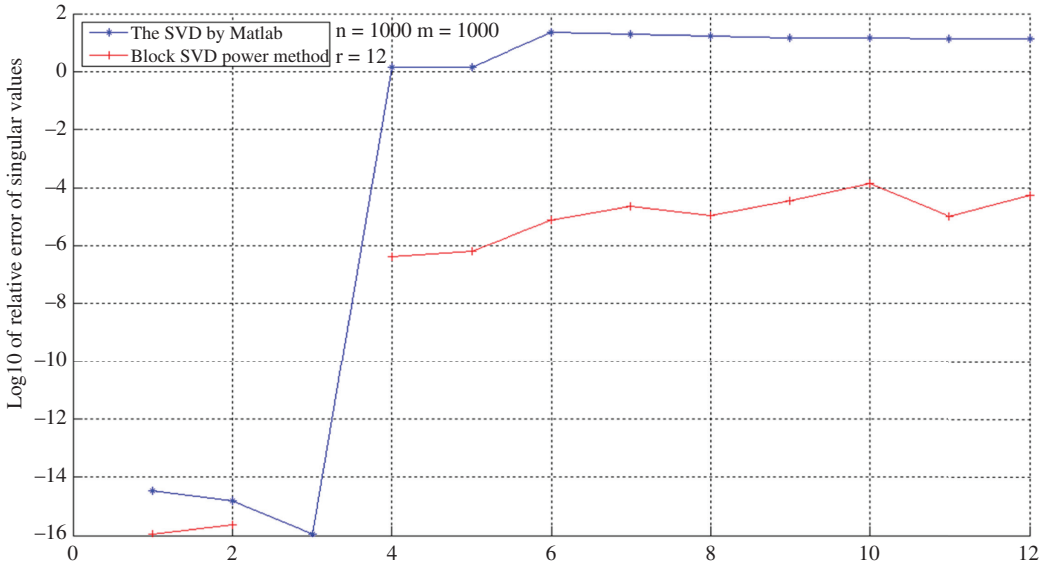

Figure 4: Relative Errors Occurring When Computing the Singular Values.

5.3 Image Compression

To test our method, we have developed a user interface. The method was applied to various and real images to demonstrate the performances of the proposed image compression algorithm.

This proposed algorithm is tested with many images, some of them are shown in Figures 5–10. Indeed, we used in this paper only two color images, *Chrysanthème* and *Desert* available on Windows Professional (32-bit), and one in grayscale (see Figure 5). Furthermore, we have used *Pepper* and *lena* (see Figure 10). Figures 6, 7, 10, and 11 show the test images and the result of compressed images using MATLAB's SVD function [11] and the proposed compression method. From Figures 6, 7, 10, and 11, it is clear that the compressed images are perceptually similar to the original images.

We recall that our goal is to approximate an image (matrix of $m \times n$) using the least amount of information. Thereby, to obtain a better quality of the compressed image using SVD, we use the K rank determined by El Asnaoui et al. [11] [Eq. (2)].

The compression ratio and the variation of PSNR with image rank when we apply MATLAB's SVD function [11] and the proposed compression method are showed in Figures 8, 9, and 12.

5.3.1 Application to Color Image

To evaluate the performance of the proposed method, we apply MATLAB's SVD function [11] and the new method to color image. After rank $K = 438$, we obtain (see Figures 5–7):

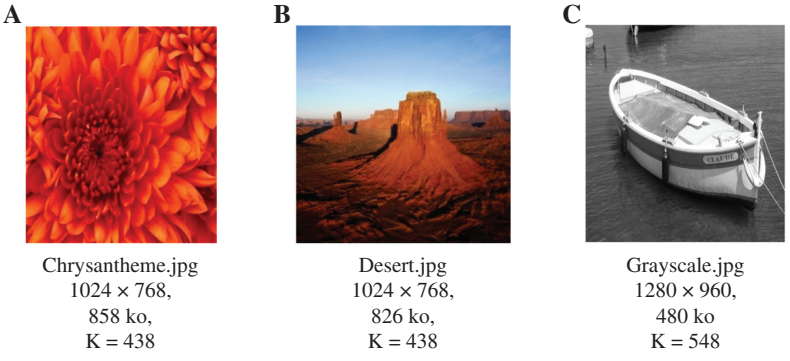

Figure 5: Original Images.
(A) Chrysantheme, (B) desert, and (C) grayscale.

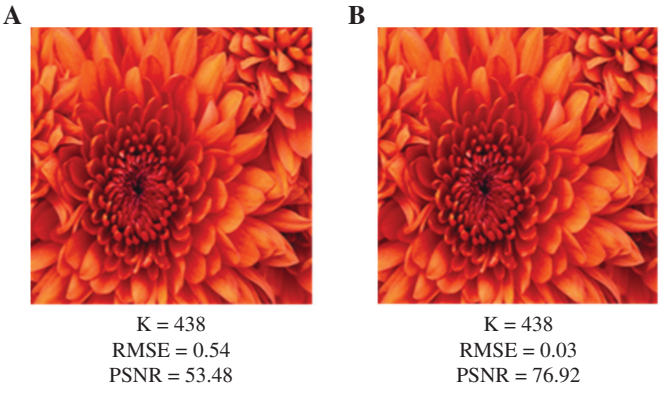

Figure 6: Compressed Results.
Compressed results obtained by (A) MATLAB's SVD function [11] and (B) the proposed method.

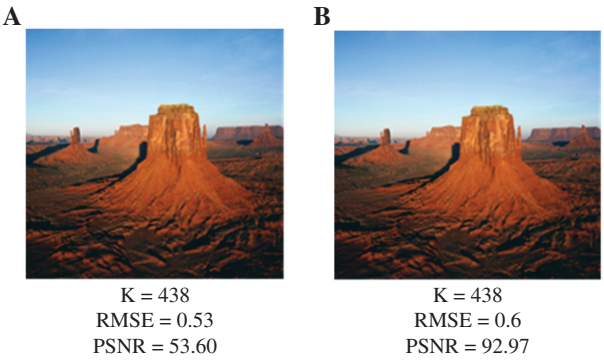

Figure 7: Compressed Results.
Compressed results obtained by (A) MATLAB's SVD function [11] and (B) the proposed method.

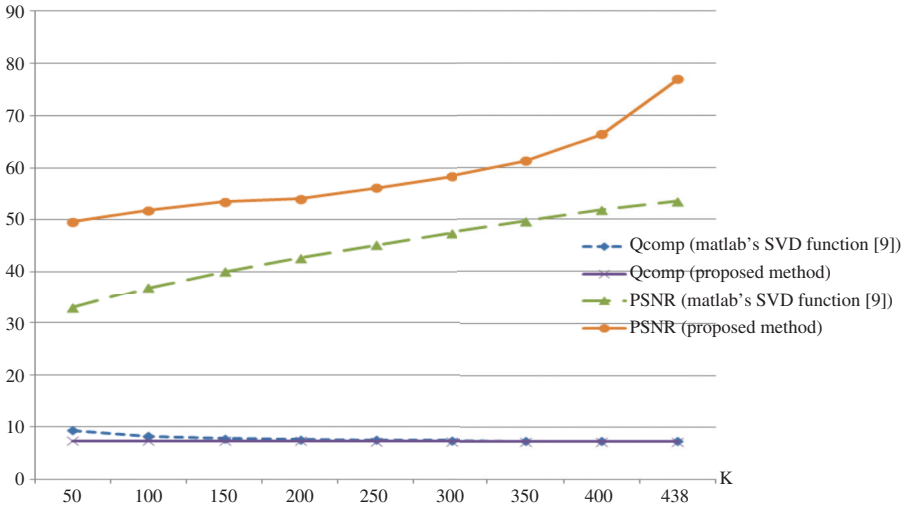

Figure 8: Relation between Image Rank, Compression Ratio, and PSNR for Chrysantheme.jpg, 1024 × 768, 858ko.

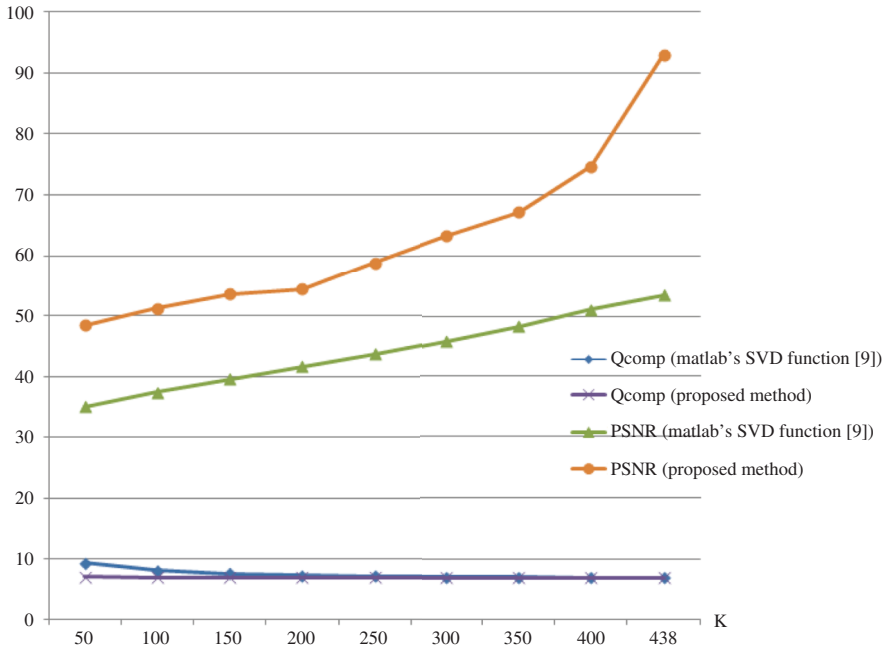

Figure 9: Relation between Image Rank, Compression Ratio, and PSNR for Desert.jpg, 1024 × 768, 826ko.

5.3.2 Application to Other Images

The proposed algorithm has also been tested with Pepper and lena images (see Figure 10). After K rank, we obtain the results shown in Table 1.

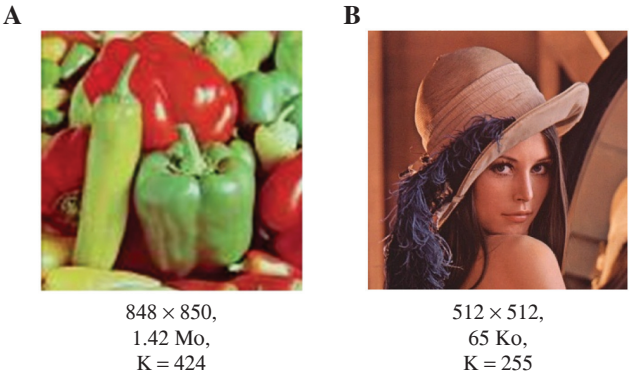

Figure 10: Images used for Proposed Algorithm.
Images used: (A) pepper.jpg and (B) lena.jpg.

Table 1: Application to Other Images.

	Pepper.jpg		Lena.jpg	
	RMSE	**PSNR**	**RMSE**	**PSNR**
MATLAB's SVD function [11]	0.03	76.54	0.83	49.71
Proposed method	0.009	88.23	0.11	67.12

5.3.3 Application to Grayscale Image

In order to compare this performance, we also applied MATLAB's SVD function [11] and the new method to the grayscale image. After rank $K = 548$, we obtain (see Figure 11):

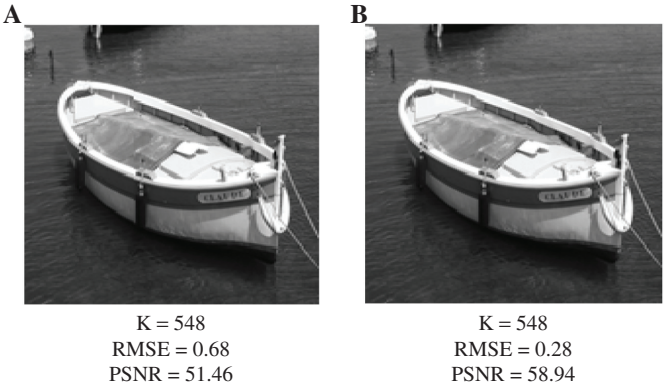

Figure 11: Compressed Results.
Compressed results obtained by (A) MATLAB's SVD function [11] and (B) the proposed method.

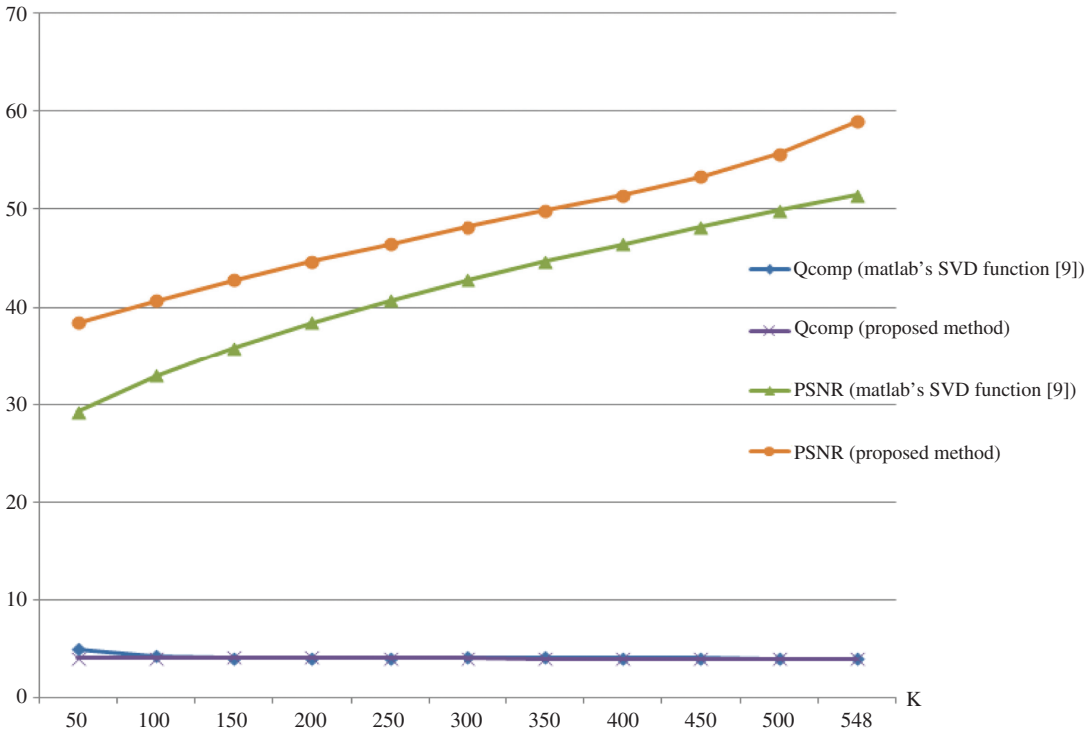

Figure 12: Relation between Image Rank, Compression Ratio, and PSNR for grayscale.jpg 1280 × 960, 480ko.

5.3.4 Comparison against Various Algorithms

To evaluate the robustness of our method, we have tested it with different compression techniques [7, 9–11]. The additional experimental results for three images are listed in Table 2.

5.4 Discussion

This study sets up a new algorithm for image compression that must be considered as an application of the block SVD power method to image compression. With our preliminary experimental analysis, observation, and evaluation, the proposed method achieves promising performance and provides a good PSNR. In addition, the proposed algorithm is compared with MATLAB's SVD function and other state-of-the-art algorithms.

The numerical examples given above show the efficiency of the new SVD [4] approach in computing the decomposition, error, and CPU time.

When applying the new method to image compression (see Figures 6, 7, 10, and 11), it is clear that the images compressed by two approaches are perceptually similar to the original images. However, the human visual response to image quality is insufficient.

To compare the performances of the proposed method, several values were used in this study to measure the quality of the compressed image. We will only discuss PSNR and MSE values, because they are used to compare the squared error between the original image and the reconstructed image. There is an inverse relationship between PSNR and MSE. Therefore, a higher PSNR value indicates higher quality of the image.

The above analysis shows the comparison when MATLAB's SVD function [11] and the proposed method are applied on real images. In these experiments, we used the K rank for different images. We see in this case that the compression ratio, PSNR, and other values of images varied when changing the rank of the

Table 2: Comparison against Various Algorithms.

	Color image (Figure 4A) 1024×768, 858ko, $K = 438$					Color image (Figure 4B) 1024×768, 826ko, $K = 438$					Grayscale image (Figure 4C) 1280×960, 480ko $K = 548$				
Method	Q_{comp}	**MSE**	**RMSE**	**PSNR**	**Size (ko)**	Q_{comp}	**MSE**	**RMSE**	**PSNR**	**Size (ko)**	Q_{comp}	**MSE**	**RMSE**	**PSNR**	**Size (ko)**
El Abbadi et al. [10]	9.27	62.09	7.88	30.20	92	9.27	65.02	8.06	30	89	5.39	159.98	12.64	26.09	89
Doaa and Chadi [9]	7.34	7.96	2.82	39.12	116	8.21	9.75	3.12	38.24	100	3.98	18.88	4.34	35.37	120
BTC method [11]	6.72	2.74	1.65	43.75	127	7.34	7.46	2.73	39.40	119	2.84	3.47	1.85	42.76	169
SVD method [11]	7.35	0.29	0.54	53.48	116	6.92	0.28	0.53	53.60	119	4.02	0.47	0.68	51.46	119
KSVD [7]	7.33	0.10	0.32	57.97	117	6.92	0.33	0.57	52.89	119	4.02	0.19	0.43	55.32	119
Proposed method	7.31	0.0013	0.03	76.92	117	6.92	0.36	0.6	92.97	122	4.01	0.08	0.28	58.94	119

image during the SVD process, as shown in Figures 8, 9, and 12. Moreover, it is evident that the proposed method gives better performance compared to MATLAB's SVD function [11]. In addition, for MATLAB's SVD function [11], the value of K, which provides better PSNR value, is the maximum value of $K = 438$, while for the proposed method, a better compression ratio, PSNR, is provided from $K = 150$ for color images (see Figure 8). We can say that in this case, MATLAB's SVD function approximately presents one-third of the proposed method in terms of K rank. For Figure 9, the value of K that provides better PSNR value is the maximum value of $K = 438$, while for the proposed method, a better compression ratio, PSNR, is provided from $K = 200$.

Concerning the grayscale image tested (see Figure 11), it seems that the value of K that gives better PSNR value is the maximum value of $K = 548$, while for the proposed method, a better compression ratio, PSNR, is provided from $K = 400$ (see Figure 12).

We mainly compared the proposed algorithm with the other algorithms as illustrated in Table 2, because these algorithms are well known and are mainly using MATLAB's SVD function. Hence, we see that our proposed algorithm performs comparably to current state-of-the-art techniques and is able to produce a compressed image with better visual quality, as indicated by its PSNR.

6 Conclusion

We suggest in this work a novel lossy image compression technique by using the block SVD power method. This approach is simple in terms of implementation, and can be used to overcome the limitations of existing algorithms that use MATLAB's SVD function. The results obtained by the proposed technique are compared with those of several state-of-the-art image compression techniques, and indicated that the proposed approach might be considered as a solution for the development of image compression. Satisfactory compression of expected images is provided faster due to the lower number of iterations in the compression algorithm. Of course, MATLAB's SVD function is accurate. However, in numerical analysis, we always focus on improving results.

6.1 Future Works

We believe that this kind of block SVD power method opens the door to many other applications of this method in the future. For example, using this method for statistical applications to find relations between data, in the area of medical image denoising with different thresholding techniques associated with these multi-wavelets, implements a compression technique using neural network. It is also useful with other techniques in image restoration.

In addition, our future approach, which is defined in parallel with the Golub-Kahan method, will be to present a JSVD decomposition of a $2n \times 2m$ rectangular matrix A based on the J-bidiagonal matrix computed by using symplectic reflectors. The obtained J-bidiagonal matrix is transformed to a diagonal matrix using symplectic Givens rotations. It allows us to compute the eigen-values of the skew-Hamiltonian matrix $A^J A$.

Bibliography

[1] M. A. Alkhalayleh and A. M. Otair, A new lossless method of image compression by decomposing the tree of Huffman technique, *Int. J. Imaging Robot.* **15** (2015), 79–96.

[2] H. C. Andrews and C. L. Patterson, Singular value decomposition (SVD) image coding, *IEEE Trans. Commun.* **24** (1976), 425–432.

[3] M. R. Awwal, G. Anbarjafari and H. Demirel, Lossy image compression using singular value decomposition and wavelet difference reduction, *Digital Signal Process.* **24** (2014), 117–123.

[4] A. H. Bentbib and A. Kanber, Block power method for SVD decomposition, *Anal. Stiintifice Univ. Ovidius Constanta Ser. Mat.* **23** (2015), 45–58.

[5] A. Bilgin, W. Michael, M. Marcellin and I. Altbach, Compression of electrocardiogram signal using JPEG2000, *IEEE Trans. Consum. Electron.* **49** (2003), 833–840.

[6] O. Bryt and M. Elad, Compression of facial images using the K-SVD algorithm, *J. Vis. Commun. Image Represent.* **19** (2008), 270–283.
[7] O. Bryt and M. Elad, Improving the K-SVD facial image compression using a linear deblocking method, in: *IEEE 25th Convention of Electrical and Electronics Engineers in Israel*, IEEEI, pp. 533–537, IEEE, 2008.
[8] E. J. Delp and O. R. Mitchell, Image compression using block compression, *IEEE Trans. Commun.* **27** (1979), 1335–1342.
[9] M. Doaa and A. F. Chadi, Image compression using block truncation coding, *Cyber J. Multidiscipl. J. Sci. Technol. J. Select. Areas Telecommun. (JSAT)* (2011).
[10] N. K. El Abbadi, A. Al Rammahi, D. S. Redha and M. Abdul-Hameed, Image compression based on SVD and MPQ-BTC, *J. Comput. Sci.* **10** (2014), 2095–2104.
[11] K. El Asnaoui and Y. Chawki, Two new methods for image compression, *Int. J. Imaging Robot.* **15** (2015), 1–11.
[12] K. El Asnaoui, M. Ouhda, B. Aksasse and M. Ouanan, An application of linear algebra to image compression, in: A. Badawi, M. Vedadi, S. Yassemi, A. Yousefian Darani (eds.), *Homological and Combinatorial Methods in Algebra*, SAA 2016, Springer Proceedings in Mathematics & Statistics, vol. 228, Springer, Cham, 2018.
[13] C. S. M. Goldrick, W. J. Dowling and A. Bury, Image coding using the singular value decomposition and vector quantization, in: *Image Processing and its Applications*, pp. 296–300, IEEE, 1995.
[14] W. Jianji, Z. Nanning, L. Yuehu and Z. Gang, Parameter analysis of fractal image compression and its applications in image sharpening and smoothing, *Signal Process. Image Commun.* **28** (2013), 681–687.
[15] N. S. Jindal, Performance analysis of SVD and SPIHT algorithm for image compression application, *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2** (2012).
[16] M. Joshi, *Digital Image Processing, An Algorithmic Approach*, pp. 175–217, PHI, New Delhi, 2006.
[17] A. Kapoor and R. Dhir, Image compression using fast 2-D DCT technique, *Int. J. Comput. Sci. Eng.* **3** (2011), 2415–2419.
[18] J. Li, An image feature point matching algorithm based on fixed scale feature transformation, *Optik-Int. J. Light Electron Optics* **124** (2013), 1620–1623.
[19] J. Li, An improved wavelet image lossless compression algorithm, *Optik-Int. J. Light Electron Optics* **124** (2013), 1041–1044.
[20] J. Li and L. Chang, A SAR image compression algorithm based on Mallat tower-type wavelet decomposition, *Optik-Int. J. Light Electron Optics* **126** (2015), 3982–3986.
[21] M. Maharani, B. K. Dewi, F. A. Yulianto and B. Purnama, Digital image compression using graph coloring quantization based on wavelet-SVD, *J. Phys. Conf. Ser.* **423** (2013), 012019.
[22] A. Ranade, S. S. Mahabalarao and S. Kale, A variation on SVD based image compression, *Image Vis. Comput.* **25** (2007), 771–777.
[23] A. Shnayderman, A. Gusev and A. M. Eskicioglu, An SVD-based gray scale image quality measure for local and global assessment, *IEEE Trans. Image Process. J.* **15** (2006), 422–429.
[24] C. C. Tsou, S. H. Wu and Y. C. Hu, Fast pixel grouping technique for block truncation coding, in: *Workshop on Consumer Electronics and Signal Processing (WCEsp'05)*, Yunlin, November 17–18, 2005.
[25] P. Waldemar and T. A. Ramstad, Image compression using singular value decomposition with bit allocation and scalar quantization, in: *Proceedings of NORSIG Conference*, 1996, pp. 83–86.
[26] M. J. Weinberger, G. Seroussi and G. Sapiro, The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS, *IEEE Trans. Image Process.* **9** (2000), 1309–1324.
[27] J. F. Yang and C. L. Lu, Combined techniques of singular value decomposition and vector quantization for image coding, *IEEE Trans. Image Process.* **4** (1995), 1141–1146.