Scientific Research Publishing

Journal of Signal and Information Processing, 2019, 10, 59-72
http://www.scirp.org/journal/jsip
ISSN Online: 2159-4481
ISSN Print: 2159-4465

Adaptive Lossy Image Compression Based on Singular Value Decomposition

Marcos Roberto e Souza, Helio Pedrini

Institute of Computing, University of Campinas, Campinas, SP, Brazil
Email: marcoosrs@gmail.com, helio@ic.unicamp.br

How to cite this paper: e Souza, M.R. and Pedrini, H. (2019) Adaptive Lossy Image Compression Based on Singular Value Decomposition. *Journal of Signal and Information Processing*, **10**, 59-72.
https://doi.org/10.4236/jsip.2019.103005

Received: June 16, 2019
Accepted: August 5, 2019
Published: August 8, 2019

Abstract

Image compression techniques aim to reduce redundant information in order to allow data storage and transmission in an efficient way. In this work, we propose and analyze a lossy image compression method based on the singular value decomposition using an optimal choice of eigenvalues and an adaptive mechanism for block partitioning. Experiments are conducted on several images to demonstrate the effectiveness of the proposed compression method in comparison with the direct application of the singular value decomposition.

Keywords

Image Compression, Adaptive Decomposition, Lossy Compression

1. Introduction

Advances in digital imaging and video acquisition equipments, such as cameras, cell phones, and tablets, have enabled the generation of large volumes of data. Many applications use images and videos, such as medicine [1], remote sensing [2], microscopy [3], surveillance and security [4].

Adjacent pixels in images and videos usually have a high correlation, which leads to redundancy and demands high storage consumption. In order to reduce the space required and the transmission time of images and videos, several compression techniques have been proposed in the literature. These techniques explore data redundancies and are generally classified into two categories of compression methods: lossless and lossy.

In lossless compression methods, the original data can be completely recovered after the decompression process [4], whereas, in lossy compression methods, certain less relevant information is discarded, such that the resulting image is different from the original image. Ideally, this loss should be tolerated by the receiver [4].

As the main contribution of this work, we present and evaluate a lossy-based image compression method based on the singular value decomposition (SVD) technique [5] [6]. The originality of the method resides mainly in the choice of eigenvalues and the partitioning of the images in an adaptive way, as well as a detailed investigation in the use of the SVD for the image compression. Experiments show that the proposed method is able to achieve better results than those obtained with the direct application of SVD. In addition, new insights to the singular value decomposition-based compression method are provided.

This paper is organized as follows. Some relevant concepts and approaches related to the topic under investigation are briefly described in Section 2. The proposed method for image compression using singular value decomposition is presented in Section 3. The results are reported and discussed in Section 4. Final considerations are presented in Section 5.

2. Related Work

Compression techniques [7]-[16] play an important role in the storage and transmission of image data. Their main purpose is to represent an image in a very compact way, that is, through a small number of bits without losing the essential content of the information present in the image.

Image compression methods are generally categorized into lossless and lossy approaches [17] [18] [19]. In lossless compression, all information originally contained in the image is preserved after it is uncompressed. In lossy compression, only part of the original information is preserved when the images is uncompressed.

Image compression methods are based on reducing redundancies in the data representation. The redundancies present in images can be divided into three categories [4] [20]. The coding redundancy refers to the use of a number of bits greater than absolutely necessary to represent the intensities of the pixels of the image. The interpixel redundancy is associated with the relation or similarity of neighboring pixels. Finally, the psychovisual redundancy concerns the imperceptible details of the human visual system.

Some lossless compression methods that explore coding redundancy are Huffman coding [21], Shannon-Fano coding [22], arithmetic coding [23] and dictionary-based encoding such as LZ78 and LZW [24]. Run-length coding [25], bit plane coding [26] and predictive coding [27] explore interpixel redundancy. Lossy compression methods, such as predictive coding by delta modulation (DM) [28] [29] or by differential pulse-code modulation (DPCM) [30] and transform coding [31] [32], typically explore psychovisual redundancy.

A common transformation employed in image compression is the singular value decomposition (SVD). Waldemar *et al.* [33] proposed a hybrid system with Karhunen-Loéve (KL) vectors and SVD for image compression. Ranade *et al.* [34] performed permutations on the input image as a preprocessing, before applying the SVD. Rufai *et al.* [35] combined the SVD with wavelet difference reduction

(WDR), in which the input image is first compressed using the SVD and then compressed again with the WDR.

In this work, we apply the use of singular value decomposition in image compression and analyze its use locally and with an optimal choice of eigenvalues.

3. Adaptive Image Compression

In this section, we present our adaptive image compression approach based on singular value decomposition, the image partitioning process and the compression evaluation metrics.

3.1. Image Compression

The diagram presented in **Figure 1** presents the main steps of the image compression and decompression method proposed in this work. The compression algorithm has as input an image $I_{n\times p}$ and, as output, the compressed image represented by the singular value decomposition matrices in a binary file. The decompression algorithm has as input the binary file containing the decomposition matrices and, as output, an image $J_{n\times p} \approx I_{n\times p}$. The steps of the method are presented in detail as follows.

Initially, we calculate the singular value decomposition (SVD) for the entire image. The SVD of a real matrix $A_{n\times p}$ corresponds to the factorization

$$A = U\Sigma V^{\mathrm{T}}$$

where U is a real unit matrix $n\times n$, Σ is a diagonal rectangular matrix $n\times p$ with diagonal nonnegative real numbers and V is a real unit matrix $p\times p$.

Since U and V^{T} U are real unitary matrices, then $UU^{\mathrm{T}} = I$ and $VV^{\mathrm{T}} = I$, where I is the identity matrix. Thus, U and V^{T} are orthogonal matrices. The columns of U are the eigenvectors of the matrix AA^{T}, the columns of V are the eigenvectors of the matrix $A^{\mathrm{T}}A$ and the diagonal elements of the matrix Σ are the square roots of eigenvalues of AA^{T} or $A^{\mathrm{T}}A$. The eigenvalues are arranged in the matrix Σ in descending order, that is, if σ_i are the diagonal elements of Σ, then $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$. Then,

$$\begin{aligned} A = U\Sigma V^{\mathrm{T}} &= \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_n \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_n^{\mathrm{T}} \end{bmatrix} \\ &= \boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^{\mathrm{T}} + \boldsymbol{u}_2\sigma_2\boldsymbol{v}_2^{\mathrm{T}} + \cdots + \boldsymbol{u}_n\sigma_n\boldsymbol{v}_n^{\mathrm{T}} \end{aligned} \tag{1}$$

Using the entire SVD arrays, with $\sigma_1, \cdots, \sigma_n$, would typically make SVD matrices larger than the original matrix. Since we are interested in lossy compression, some of the eigenvalues above a certain σ_i can be discarded, decreasing the final size of the three matrices.

Considering that the σ_i elements are sorted in descending order, the first terms $\boldsymbol{u}_i\sigma_i\boldsymbol{v}_i^{\mathrm{T}}$ contribute more significantly to the summation. Thus, we determine the i value that optimizes

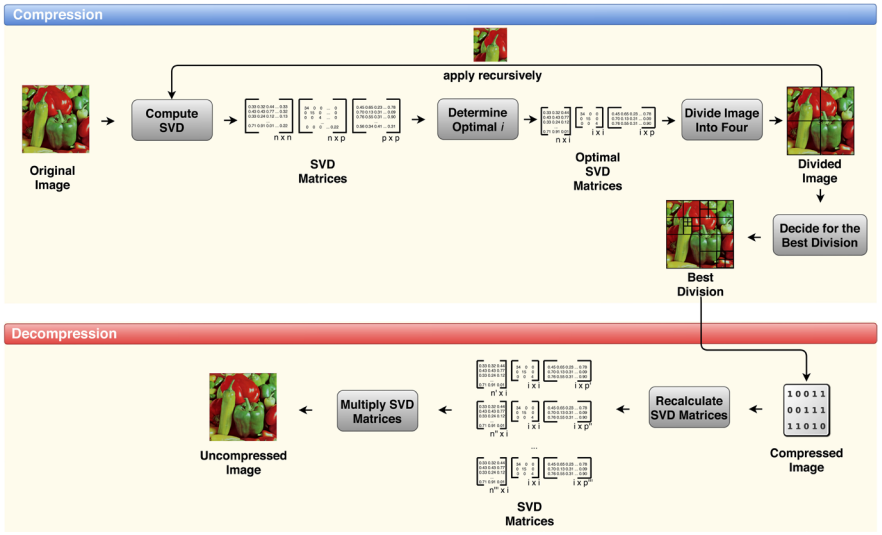


Figure 1. Main stages of the image compression and decompression process.

$$\text{opt\_value} = \arg\max_i (\alpha - 1)(\lambda_i) + (\alpha)(\delta_i) \tag{2}$$

where λ_i is the variance maintained by choosing the first i eigenvalues, calculated as

$$\lambda_i = \lambda_{i-1} + \sigma_i^* \tag{3}$$

where $\sigma_i^* = \dfrac{\sigma_i}{\sum \sigma_i}$ and δ_i is the relative redundancy of data, which can be expressed as

$$\delta_i = 1 - \frac{i + ni + pi}{np} \tag{4}$$

The factor α is used to weight the two terms, calculated as

$$\alpha' = \begin{cases} 1 - \dfrac{r}{n}, & \text{if } n < p \\ 1 - \dfrac{r}{p}, & \text{otherwise} \end{cases} \tag{5}$$

$$\alpha = \begin{cases} 0.3, & \text{if } \alpha' < 0.3 \\ 0.6, & \text{otherwise} \end{cases} \tag{6}$$

The adaptive choice of the i value is made by means of the optimization described previously and not by assigning a fixed minimum variance, as in many approaches of the literature [33] [34]. The motivation for this decision lies in the fact that we will consider a given eigenvalue if the cost to add it will compensate for the significance it carries, that is, adding the eigenvalue σ_{i+1} would increase the number of bytes in the compressed image as opposed to improving the image quality.

In addition, other methods available in the literature [33] [34] do not consider or discuss the fact that the matrices generated by the SVD are real numbers (usually 64 bits), whereas an image is typically integer (8 bits), which makes the

direct application of SVD unfeasible. Thus, we apply a rounding in the matrices obtained by the decomposition, so that we consider only the first three decimal places. This rounding can be expressed as

$$f = \lfloor h10^d + 0.5 \rfloor \tag{7}$$

where h is the original value of the (real) matrix and f is the value obtained after rounding (integer). In this work, we consider $d = 3$ for two main reasons. First, this will cause us to have 3-digit numbers since the original values are in the range $[0,1]$, which is interesting because the linear transformation that will be applied considers the interval from 0 to 255. Thus, a lower degree of loss will be obtained by changing the range of numbers. In addition, using smaller values for d showed, empirically, a great loss of information, whereas using larger values did not show significant improvement. This rounding is not applied to the Σ matrix, since its value is not between 0 and 1.

Then, we apply a linear transformation that maps the original interval to the range 0 to 255 expressed as

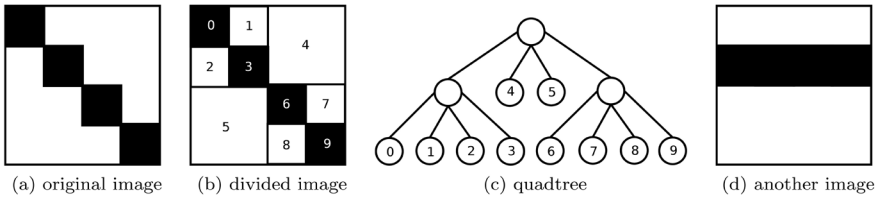
$$g = \frac{g_{\max} - g_{\min}}{f_{\max} - f_{\min}}\left(f - f_{\min}\right) + g_{\min} \tag{8}$$

where $f_{\min}$ and $f_{\max}$ are the minimum and maximum values of the matrix, respectively, whereas $g_{\min}$ and $g_{\max}$ are 0 and 255, respectively. The variable f represents the original value and g the value obtained after the transformation.

Then, the image is divided into four blocks of the same size, and the compression is recursively applied to each of the blocks. This division is performed until the image has a certain minimum size. A quadtree decomposition is formed from this process, where the inner nodes represent the divisions of the image and the leaf nodes represent the SVD matrices.

Figure 2(a) presents an example of a binary image of 4×4 pixels. White pixels have a value of 0, whereas black pixels have a value of 1. The rank of the matrix representing this image is $r = 4$ since no row or column is linearly dependent on the other. Figure 2(b) presents a partitioning, in which only pixels of the same intensity were kept together. The quadtree corresponding to the division performed is illustrated in Figure 2(c).

Since all regions have pixels of the same intensity, the rank of each region in Figure 2(b) is equal to $r = 1$. Figure 2(d) presents an image where 4 pixels are black, but with another configuration. This image has rank $r = 1$, which shows a dependence of the rank and, consequently, of the good performance of the SVD in relation to the image content.



(a) original image (b) divided image (c) quadtree (d) another image

Figure 2. Example of image partitioning using the quadtree decomposition.

The number of units (number of values) required to store the SVD matrices can be expressed as

$$\text{Units} = r + pr + nr \tag{9}$$

where r is the rank, whereas p and n are the dimensions of the original image. The first term of the sum (r) corresponds to the diagonal matrix size Σ, whereas the other values correspond to the two other matrices U and V. Thus, the number of units needed to store the image shown in **Figure 2(a)** is given by

$$4 + (4)(4) + (4)(4) = 36 \tag{10}$$

On the other hand, for the image after partitioning, illustrated in **Figure 2(b)**, the number of units can be calculated as

$$\begin{aligned} &1 + (2)(1) + (2)(1) = 5 \\ &1 + (1)(1) + (1)(1) = 3 \\ &(5)(2) + (3)(8) = 34 \end{aligned} \tag{11}$$

We note that, for this division in the image, the number of units needed to store the SVD matrices has decreased. The amount of units needed to store SVD matrices with the division (considering all divisions with the same rank) is smaller than in the entire image, since

$$\begin{aligned} &4\left(r' + \frac{p}{2}r' + \frac{n}{2}r'\right) < r + pr + nr \\ &4r' + 2pr' + 2nr' < r + pr + nr \\ &2r'(1 + p + n) + 2r' < r(1 + p + n) \\ &r'\left(2 + \frac{2}{1 + p + n}\right) < r \end{aligned} \tag{12}$$

where r' is the rank of the matrices after division and r' is the rank of the original matrix.

In our method, we determine the best alternative, that is, whether the decomposition will be calculated in the entire image or in its four blocks, from the optimal value calculated with Equation (2). Thus, the block division will be chosen if

$$\text{opt\_value} > \frac{\sum_{j=0}^{4} \text{opt\_value}_j}{4} \tag{13}$$

where opt_value is the optimal value for the entire image and opt_value_j is the optimal value for the j-th block.

For color images, a decomposition is calculated separately for each color channel. Finally, having the necessary information, the compressed image is stored in a binary file. **Figure 3** shows the protocol used to store the image, with header and data.

The binary file has at its beginning a global header, shown in **Figure 3(a)**. In this header, three values are stored: the width (n) and height (p) of the image,

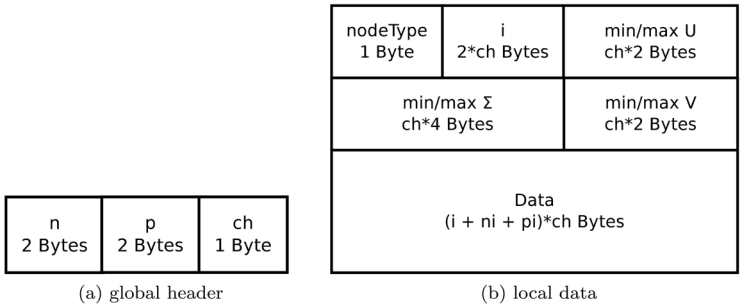


Figure 3. Image representation in a binary file.

and the number of channels (*ch*) of the image. They determine the size of each node in the tree, which is necessary to correctly retrieve the data in the decompression step.

The tree nodes are stored, as shown in **Figure 3**, having a local header and data. The first information in this header is the type of node: 1) internal or 2) leaf. For internal nodes, we do not have data, but four children. Thus, after an internal node, four other nodes are expected. For the leaf nodes, besides the node type, we have the rank value *r*, the minimum and maximum values of each of the matrices. Finally, we have the data for that particular node.

Given the compressed image, the file is read and the tree is reconstructed. For each SVD matrix, a linear transformation is applied in order to return it to the original interval. This is done by using Equation (8), where, in this case, $f_{\min}$ and $f_{\max}$ are 0 and 255 and $g_{\min}$ and $g_{\max}$ are the various minimums and maximums of the original matrix. Next, we apply a transformation inverse to that made in Equation (7) to retrieve the decimal values. Finally, the matrices are multiplied to obtain an uncompressed image.

3.2. Evaluation Metrics

We evaluated the compression under two different aspects: the size of the compressed image and the quality of the uncompressed image. For the first, we adopted the compression rate that can be expressed as

$$S = \frac{S(I)}{S(I')} \tag{14}$$

where $S(I)$ refers to the size of the uncompressed image in bytes, whereas $S(I')$ refers to the size of the compressed image in bytes. The result of this measurement indicates how many times the compressed image is smaller than the original image.

To evaluate the quality of the decompressed image, we used the mean of the structural similarity index (MSSIM), expressed as

$$\mathrm{MSSIM}(I,J) = \frac{\sum_{i=0}^{k} \mathrm{SSIM}(I_i, J_i)}{k} \tag{15}$$

where I and J are the original and uncompressed images, respectively. I_i and

J_i are the i-th windows of the images and k is the total number of windows. The structural similarity index (SSIM) [36] is expressed as

$$\mathrm{SSIM}(x,y)=\frac{\left(2\mu_x\mu_y+\left(k_1+L_{\max}\right)^2\right)\left(2\sigma_{xy}+\left(k_2+L_{\max}\right)^2\right)}{\left(\mu_x^2+\mu_y^2+\left(k_1+L_{\max}\right)^2\right)\left(\sigma_x^2+\sigma_y^2+\left(k_2+L_{\max}\right)^2\right)} \tag{16}$$

where μ_x and μ_y are the means of x and y, σ_x^2 and σ_y^2 are the variances of x and y. The variable σ_{xy} is the covariance between x and y, whereas k_1 and k_2 are constants. The SSIM values are in a range of $[-1,1]$, where the higher the value, the greater the similarity.

4. Experimental Results

This section presents the results of experiments conducted on a set of twenty input images. Table 1 summarizes some relevant characteristics of the images. Seventeen images were extracted from a public domain repository [37], whereas the other three were collected separately.

Table 1. Images used in our experiments.

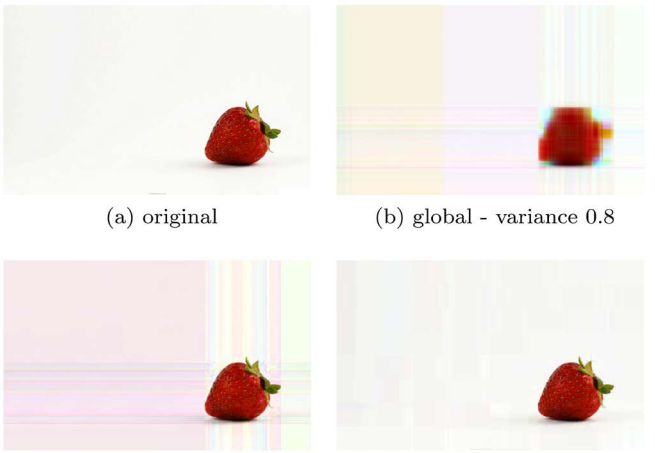
#	Image	Dimensions (pixels)	Source
1	airplane	512×512	Public Domain
2	baboon	512×512	Public Domain
3	barbara	512×512	Public Domain
4	boat	512×512	Public Domain
5	boy	768×512	Public Domain
6	cameraman	256×256	Public Domain
7	chessboard	1024×1024	Collected Separately
8	fruits	512×512	Public Domain
9	girl	768×512	Public Domain
10	goldhill	512×512	Public Domain
11	lena	512×512	Public Domain
12	line	1024×640	Generated by the Authors
13	monarch	768×512	Public Domain
14	mountain	640×480	Public Domain
15	peppers	512×512	Public Domain
16	sails	768×512	Public Domain
17	strawberry	1024×640	Collected Separately
18	tulips	768×512	Public Domain
19	watch	1024×768	Public Domain
20	zelda	512×512	Public Domain

In the experiments performed, we compared the relationship between the compression ratio and the MSSIM value. Initially, Table 2 presents different strategies for choosing the σ_i eigenvalue. The first strategy, used in other approaches available in the literature [34], considers a minimum variance to be preserved in the image. Thus, the eigenvalue σ_i is chosen so that the variance maintained by the eigenvalues $\sigma_1, \cdots, \sigma_i$ is greater than or equal to the minimum variance, whereas the second strategy chooses an optimal eigenvalue σ_i, as defined in Equation (2).

We can notice that the results obtained by the optimal choice have a compression ratio always greater than 1, in order to obtain a smaller image in all cases, and with high MSSIM values. Since different images have different variance, requiring different amounts of eigenvalues to obtain the same variance, as discussed in Section 0, the minimum variance cannot be used as a fixed value. For example, for images #1 and #5, the compression ratio is less than 1 with variance 0.95, whereas, for image #14, maintaining a variance of 0.9 has already caused the compression ratio to be less than 1. This demonstrates the need to make an adaptive choice. Figure 4 compares the different versions obtained for image #17.

Table 2. Comparison between minimum variance and optimal choice.

#	Minimum Variance: 0.8		Minimum Variance: 0.85		Minimum Variance: 0.9		Minimum Variance: 0.95		Optimal Choice	
	Compression	MSSIM	Compression	MSSIM	Compression	MSSIM	Compression	MSSIM	Compression	MSSIM
1	4.677	0.917	3.250	0.944	2.119	0.964	1.498	0.973	3.032	0.952
2	1.585	0.932	1.268	0.956	1.016	0.974	0.848	0.983	1.501	0.940
3	2.609	0.828	2.079	0.869	1.559	0.915	1.195	0.941	1.937	0.881
4	2.779	0.840	2.062	0.880	1.470	0.916	1.142	0.932	2.325	0.866
5	1.693	0.930	1.327	0.952	1.091	0.967	0.926	0.976	2.716	0.886
6	2.774	0.799	2.059	0.844	1.520	0.884	1.073	0.924	2.553	0.813
7	290.867	0.982	203.934	0.884	102.158	0.847	47.563	0.948	41.744	0.949
8	3.292	0.921	2.412	0.943	1.639	0.962	1.241	0.970	2.600	0.939
9	3.474	0.933	2.274	0.954	1.659	0.967	1.306	0.973	3.360	0.936
10	2.691	0.885	1.908	0.920	1.390	0.941	1.093	0.951	2.435	0.896
11	3.170	0.945	2.237	0.964	1.585	0.978	1.210	0.984	2.600	0.959
12	2.385	0.959	1.831	0.925	1.485	0.936	-	-	4.799	0.947
13	3.448	0.948	2.684	0.963	1.891	0.978	1.446	0.985	2.661	0.963
14	1.481	0.806	1.176	0.851	0.968	0.889	0.823	0.917	1.465	0.808
15	3.008	0.935	1.972	0.957	1.423	0.970	1.113	0.977	2.950	0.941
16	2.074	0.927	1.558	0.957	1.243	0.974	1.033	0.983	2.065	0.928
17	130.700	0.970	60.448	0.975	24.205	0.984	8.696	0.991	16.393	0.986
18	3.698	0.926	2.816	0.949	1.934	0.973	1.471	0.983	2.877	0.949
19	2.616	0.957	2.015	0.972	1.639	0.979	-	-	1.424	0.983
20	5.326	0.890	3.455	0.919	2.062	0.943	1.470	0.953	3.551	0.917



(a) original (b) global - variance 0.8

(c) global - optimal choice (d) local - adaptive partitioning

Figure 4. Result comparison for image #17.

It can be observed that, despite a considerably high result in the MSSIM of the global version with variance 0.8 (0.970), the visual result, shown in **Figure 4(b)**, is very poor, whereas the results are superior with the optimal choice, however, still having some problems.

The artifacts that can be seen in **Figure 4(c)** occur mainly due to the rounding and linear transformation defined in Equations (7) and (8). This problem can be overcome by a more local strategy, such as the one adopted in this work and with the result shown in **Figure 4(d)**. This improvement probably occurs since the values obtained from the matrices after the decomposition are in a smaller range and closer to 0 to 255, because they have a smaller variation in the image.

As discussed in Section 0 and observed from the previous results, the SVD technique applied globally may not be adequate. Thus, **Table 3** presents the results obtained by considering the image divided into square blocks of different sizes. In addition, it presents the results obtained with the method presented in this work, that is, with the adaptive choice of blocks and eigenvalues.

Comparing the results obtained with those shown in **Table 2**, overall, the compression rates of the images increased, maintaining high MSSIM values. This demonstrates the validity of the local division strategy which, in addition to the reduction of the artifacts shown in **Figure 4(c)**, obtains better quantitative results.

The compression and MSSIM values obtained with the adaptive division are generally similar to those obtained with the fixed block size. However, for image #7, which consists of a chessboard, the result obtained is considerably superior in terms of compression ratio, which demonstrates superiority of the adaptive division. This result can vary considerably in different images with the change of the constant values given in Equation (6). The values used were chosen empirically in order to obtain a good result in all images.

Figure 5 illustrates the result obtained for image #4 considering the different local versions. We can observe, through the resized region, that some fine details

of the image are lost in the compression process. In this case, this occurs most notably with a smaller block size. This is an apparent limitation of our adaptive division, which could select a larger block size in those regions, in order to preserve more detail. In order to circumvent this problem, we could choose a more suitable value for α, used in Equation (2), or limit the minimum size of the block adaptively.

original fixed partitioning - 64 fixed partitioning - 16 adaptive partitioning

Figure 5. Result comparison for image #4.

Table 3. Comparison between fixed and adaptive block partitioning.

#	Fixed Division: 64		Fixed Division: 32		Fixed Division: 16		Fixed Division: 8		Adaptive Division	
	Compression	MSSIM	Compression	MSSIM	Compression	MSSIM	Compression	MSSIM	Compression	MSSIM
1	6.037	0.951	5.909	0.949	4.017	0.944	1.735	0.948	3.759	0.941
2	2.304	0.906	2.610	0.883	2.500	0.850	1.570	0.823	1.755	0.823
3	3.253	0.898	3.575	0.894	2.994	0.880	1.588	0.845	2.591	0.877
4	3.881	0.848	4.407	0.829	3.573	0.823	1.701	0.856	2.506	0.833
5	2.799	0.901	2.996	0.894	3.034	0.878	2.087	0.857	2.323	0.856
6	3.268	0.871	3.665	0.876	3.127	0.877	1.603	0.901	2.407	0.893
7	18.811	0.999	10.745	0.999	4.780	1.000	1.755	1.000	75.827	0.996
8	4.647	0.926	5.102	0.919	3.835	0.920	1.696	0.940	2.948	0.926
9	3.950	0.958	4.116	0.958	3.684	0.953	2.137	0.949	2.991	0.950
10	3.844	0.872	4.258	0.852	3.649	0.828	1.742	0.851	2.338	0.834
11	4.496	0.955	4.907	0.949	3.661	0.945	1.702	0.948	2.838	0.947
12	14.258	0.974	13.081	0.976	6.860	0.989	3.309	0.989	14.846	0.988
13	4.206	0.969	4.398	0.969	3.703	0.969	2.122	0.965	3.102	0.967
14	1.921	0.814	2.012	0.822	2.190	0.807	2.146	0.782	2.233	0.779
15	4.349	0.942	4.589	0.940	3.478	0.940	1.658	0.947	2.509	0.942
16	2.583	0.908	2.848	0.895	3.008	0.868	2.135	0.839	2.150	0.840
17	25.406	0.994	15.686	0.995	7.985	0.996	3.354	0.996	26.559	0.995
18	3.099	0.950	3.120	0.949	2.966	0.940	2.019	0.931	2.339	0.934
19	2.733	0.976	3.264	0.975	3.525	0.974	2.754	0.970	3.218	0.971
20	5.453	0.917	5.340	0.911	3.854	0.907	1.737	0.924	3.076	0.906

It is noteworthy that the requirements for storing the compressed images could still be reduced. For example, a flag used to determine the node type could be only 1 bit long, instead of 1 byte as employed in our implementation for simplification purposes.

5. Conclusions

This work described and implemented a new lossy image compression method based on the singular value decomposition. The proposed approach used an optimal choice of eigenvalues computed in the decomposition, as well as an adaptive block partitioning. We also presented a protocol for storing the SVD matrices.

Experiments were conducted on a dataset composed of twenty images—seventeen extracted from a public domain repository and commonly used in the evaluation of image processing tasks, the other three images collected separately. The results obtained show that the optimal choice of eigenvalues is relevant due to differences in different image contents.

Due to rounding performed in the compression process, the overall approach achieved lower compression rate and added artifacts to the images. In addition, the adaptive partitioning strategy obtained, in some cases, considerably superior results in terms of compression ratio. However, some fine image details may be lost in the compression process based on local strategy.

Acknowledgements

The authors are thankful to São Paulo Research Foundation (FAPESP grants #2017/12646-3 and #2014/12236-1) and National Council for Scientific and Technological Development (CNPq grant #309330/2018-1) for their financial support.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cho, Z.-H., Jones, J.P. and Singh, M. (1993) Foundations of Medical Imaging. Wiley, New York.

[2] Jensen, J.R. and Lulla, K. (1987) Introductory Digital Image Processing: A Remote Sensing Perspective. *Geocarto International*, **2**, 65. https://doi.org/10.1080/10106048709354084

[3] Crocker, J.C. and Grier, D.G. (1996) Methods of Digital Video Microscopy for Colloidal Studies. *Journal of Colloid and Interface Science*, **179**, 298-310. https://doi.org/10.1006/jcis.1996.0217

[4] Wall, M.E., Rechtsteiner, A. and Rocha, L.M. (2003) Singular Value Decomposition and Principal Component Analysis. In: Berrar, D.P., Dubitzky, W. and Granzow, M., Eds., *A Practical Approach to Microarray Data Analysis*, Springer, Boston, MA,

91-109. https://doi.org/10.1007/0-306-47815-3_5

[5] Andrews, H. and Patterson, C. (1976) Singular Value Decompositions and Digital Image Processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **24**, 26-53. https://doi.org/10.1109/TASSP.1976.1162766

[6] Bhavani, S. and Thanushkodi, K.G. (2013) Comparison of Fractal Coding Methods for Medical Image Compression. *IET Image Processing*, **7**, 686-693. https://doi.org/10.1049/iet-ipr.2012.0041

[7] Messaoudi, A. and Srairi, K. (2016) Colour Image Compression Algorithm Based on the DCT Transform Using Difference Lookup Table. *Electronics Letters*, **52**, 1685-1686. https://doi.org/10.1049/el.2016.2115

[8] Qureshi, M.A. and Deriche, M. (2016) A New Wavelet Based Efficient Image Compression Algorithm Using Compressive Sensing. *Multimedia Tools and Applications*, **75**, 6737-6754. https://doi.org/10.1007/s11042-015-2590-9

[9] Yao, J. and Liu, G. (2017) A Novel Color Image Compression Algorithm Using the Human Visual Contrast Sensitivity Characteristics. *Photonic Sensors*, **7**, 72-81. https://doi.org/10.1007/s13320-016-0355-3

[10] Prakash, A., Moran, N., Garber, S., Dilillo, A. and Storer, J. (2017) Semantic Perceptual Image Compression Using Deep Convolution Networks. 2017 *Data Compression Conference*, Snowbird, UT, 4-7 April 2017, 250-259. https://doi.org/10.1109/DCC.2017.56

[11] David, M., George, T., Michele, C., Troy, C., Nick, J., Joel, S., Hwang, S.J., Vincent, D. and Singh, S. (2017) Spatially Adaptive Image Compression Using a Tiled Deep Network. 2017 *IEEE International Conference on Image Processing*, Beijing, 17-20 September 2017, 2796-2800. https://doi.org/10.1109/ICIP.2017.8296792

[12] Zhang, Y., Xu, B. and Zhou, N. (2017) A Novel Image Compression-Encryption Hybrid Algorithm Based on the Analysis Sparse Representation. *Optics Communications*, **392**, 223-233. https://doi.org/10.1016/j.optcom.2017.01.061

[13] Zhao, C., Zhang, J., Ma, S., Fan, X., Zhang, Y. and Gao, W. (2017) Reducing Image Compression Artifacts by Structural Sparse Representation and Quantization Constraint Prior. *IEEE Transactions on Circuits and Systems for Video Technology*, **27**, 2057-2071. https://doi.org/10.1109/TCSVT.2016.2580399

[14] Ahanonu, E., Marcellin, M. and Bilgin, A. (2018) Lossless Image Compression Using Reversible Integer Wavelet Transforms and Convolutional Neural Networks. 2018 *Data Compression Conference*, Snowbird, UT, 27-30 March 2018, 395-395. https://doi.org/10.1109/DCC.2018.00048

[15] Toderici, G., Vincent, D., Johnston, N., Hwang, S.J., Minnen, D., Shor, J. and Covell, M. (2017) Full Resolution Image Compression with Recurrent Neural Networks. *Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 21-26 July 2017, 5435-5443. https://doi.org/10.1109/CVPR.2017.577

[16] Baxes, G.A. (1994) Digital Image Processing: Principles and Applications. Wiley, New York.

[17] Sayood, K. (2017) Introduction to Data Compression. Morgan Kaufmann, Burlington, MA.

[18] Nelson, M. and Gailly, J.-L. (1996) The Data Compression Book. M & T Books, New York.

[19] Gonzalez, R.C. and Woods, R.E. (2002) Digital Image Processing.

[20] Rabbani, M. and Jones, P.W. (1991) Digital Image Compression Techniques. SPIE Press, Bellingham, WA.

[21] Sharma, M. (2010) Compression Using Huffman Coding. *International Journal of Computer Science and Network Security*, **10**, 133-141.

[22] Connell, J.B. (1973) A Huffman-Shannon-Fano Code. *Proceedings of the IEEE*, **61**, 1046-1047. https://doi.org/10.1109/PROC.1973.9200

[23] Gonzales, C.A., Anderson, K.L. and Pennebaker, W.B. (1990) DCT-Based Video Compression Using Arithmetic Coding. *Electronic Imaging: Advanced Devices and Systems*, Santa Clara, CA, 1990, 305-312. https://doi.org/10.1117/12.19522

[24] Morita, H. and Kobayashi, K. (1989) An Extension of LZW Coding Algorithm to Source Coding Subject to a Fidelity Criterion. 4*th Joint Swedish-Soviet Int. Workshop on Information Theory*, Gotland, Sweden, 105-109.

[25] Pountain, D. (1987) Run-Length Encoding. *Byte*, **12**, 317-319.

[26] Rabbani, M. and Jones, P.W. (1991) Bit Plane Encoding. In: *Digital Image Compression Techniques*, SPIE Press, Bellingham, WA, 49-62.

[27] Jain, A.K. (1981) Image Data Compression: A Review. *Proceedings of the IEEE*, **69**, 349-389. https://doi.org/10.1109/PROC.1981.11971

[28] De Jager, F. (1952) Delta Modulation, a Method of PCM Transmission Using the 1-Unit Code. *Philips Research Reports*, **7**, 23.

[29] Schindler, H. (1974) Linear, Nonlinear, and Adaptive Delta Modulation. *IEEE Transactions on Communications*, **22**, 1807-1823. https://doi.org/10.1109/TCOM.1974.1092119

[30] Rao, K.R. and Yip, P. (2014) Discrete Cosine Transform: Algorithms, Advantages, Applications. Academic Press, Cambridge, MA.

[31] Lewis, A.S. and Knowles, G. (1992) Image Compression Using the 2-D Wavelet Transform. *IEEE Transactions on Image Processing*, **1**, 244-250. https://doi.org/10.1109/83.136601

[32] Watson, A.B. (1994) Image Compression Using the Discrete Cosine Transform. *Mathematica Journal*, **4**, 81.

[33] Waldemar, P. and Ramstad, T. (1997) Hybrid KLT-SVD Image Compression. 1997 *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, 21-24 April 1997, 2713-2716.

[34] Rufai, A.M., Anbarjafari, G. and Demirel, H. (2014) Lossy Image Compression Using Singular Value Decomposition and Wavelet Difference Reduction. *Digital Signal Processing*, **24**, 117-123. https://doi.org/10.1016/j.dsp.2013.09.008

[35] Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P. (2004) Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, **13**, 600-612. https://doi.org/10.1109/TIP.2003.819861

[36] Hu, Y.H. (2018) Public-Domain Test Images. https://homepages.cae.wisc.edu/~ece533/images/index.html

[37] Ranade, A., Mahabalarao, S.S. and Kale, S. (2007) A Variation on SVD Based Image Compression. *Image and Vision Computing*, **25**, 771-777. https://doi.org/10.1016/j.imavis.2006.07.004