Asian Journal of Engineering and Applied Technology
ISSN: 2249-068X (P) Vol.10 No.1, 2021, pp.22-28

DOI: https://doi.org/10.51983/ajeat-2021.10.1.2724

Image Compression Techniques Using Linear Algebra with SVD Algorithm

S. Karthigai Selvam[1] and S. Selvam[2]
[1]Assistant Professor, Department of Mathematics,
[2]Head and Assistant Professor, Department of Computer Applications
[1&2]N.M.S.S.V.N. College, Madurai, Tamil Nadu, India
E-mail: s.karthic4@gmail.com, s.selvammphil@gmail.com

(Received 24 February 2021; Revised 14 March 2021; Accepted 31 March 2021; Available online 8 April 2021)

Abstract **- In recent days, the data are transformed in the form of multimedia data such as images, graphics, audio and video. Multimedia data require a huge amount of storage capacity and transmission bandwidth. Consequently, data compression is used for reducing the data redundancy and serves more storage of data. In this paper, addresses the problem (demerits) of the lossy compression of images. This proposed method is deals on SVD Power Method that overcomes the demerits of Python SVD function. In our experimental result shows superiority of proposed compression method over those of Python SVD function and some various compression techniques. In addition, the proposed method also provides different degrees of error flexibility, which give minimum of execution of time and a better image compression.**
Keywords: **Image Compression, Singular Value Decomposition, MSE, Lossy Image Compression, PSNR**

I. INTRODUCTION

The Singular Value Decomposition (SVD) is a generalization of the Eigen-decomposition used to analyze rectangular matrices. It plays an important role for many exciting real-world applications such as Mathematical models, physical and biological processes, data mining, search engines to rank in huge databases, including the Web, image processing etc. The purpose of this paper is to present the SVD applied to the image compression.

The main idea of image compression is reducing the redundancy of the image and the transferring data in an efficient form. The image compression takes an important place in several domains like web designing, in fact, maximally reduce an image allows us to create websites faster and saves bandwidth users, it also reduces the bandwidth of the servers and thus save time and money. Here, we used two aspects: image size in pixels and its degree of compression. The main goal of such 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.

II. EXISTING METHODS

In past few years, various image compression schemes and their applications in image processing have been proposed. In this section, a empirical review of few important contributions from the existing method is presented. In general, there are two approaches for image compression: lossy or lossless [1,2]. A lossless compression is an image compression technique 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.

The lossless compression can be very useful for exact reconstruction of images. The compression ratio provided by this kind of methods 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: Bit-plane coding, Huffman coding [3], Run-Length coding 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 a very fine detail 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 same amount of error is an acceptable trade-off for increased compression performance. Among methods for lossy compression, we find: Transform coding Fourier-related transform, Fractal compression [4], Discrete Cosine Transform [5,6] and Wavelet transform.

Generally, SVD is a lossy compression technique which achieves compression by using a smaller rank to approximate the original matrix representing an image. Furthermore, lossy compression yields good compression ratio comparing with lossless compression while the lossless compression gives good quality of compressed images.

According to the state-of-the-art, there are several works suggested to use the SVD with other compression methods or with variation of SVD. Awwal *et al*., [7] presented new compression technique using SVD and the Wavelet Difference Reduction. The WDR used for further reduction. This technique has been tested with other techniques such as WDR and JPEG 2000 and gives a better result than these techniques. Furthermore, using WDR with SVD enhance the PSNR and compression ratio.

A technique based on Wavelet-SVD, which used a graph coloring technique in the quantization process, is presented in [8]. This technique worked well and enhanced the PSNR and compression ratio. The generated compression ratio by this work ranged between 50-60%, while the average PSNR ranged between 40-80db.

Ranade *et al.,* [9] suggested a variation on 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 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.

The technique given by El Abbadi *et al.,* [13], proposes to use SVD and MPQ-BTC, 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. Following the sole objective of image compression using SVD, the most problem is which K rank to use for giving a better image compression. For this reason, the method presented in El Asnaoui *et al.,* [14], introduces two new approaches: The first one is an improvement of the Block Truncation Coding method that overcomes the disadvantages of the classical Block Truncation Coding, while the second one describes how to obtain a new rank of SVD method, which gives a better image compression.

III. IMAGE COMPRESSION TECHNIQUE USING SVD

The main intention of studying the SVD of an image (matrix of m · 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, since images can be viewed as matrices with each pixel being an element of a matrix. The main concept of this section is to present two algorithms: The first one is the Python SVD function, while the second one describes how to obtain a new SVD using Block SVD Power Method.

A. Algorithm of Python SVD Function

We will use *numpy.linalg* library's svd function to compute svd of a matrix in python. The svd function returns U,s,V.

1. U has left singular vectors in the columns
2. s is rank 1 numpy array with singular values
3. V has right singular vectors in the rows -equivalent to V transpose in traditional linear algebra literature

The reconstructed approximation of the original matrix is done using a subset of singular vectors as below in the compress_svd function. We use numpy array slicing to select k singular vectors and values. Instead of storing m×n values for the original image, we can now store k(m+n)+k values.

```
reconst_matrix = np.dot(U[:,:k],np.dot(np.diag(s[:k]),
V[:k,:]))
def compress_svd(image,k):
"""
Perform svd decomposition and truncated (using k singular
values/vectors) reconstruction
returns --------
reconstructed matrix reconst_matrix, array of singular
values s
"""
U,s,V = svd(image,full_matrices=False)
reconst_matrix = np.dot(U[:,:k],np.diag(s[:k]),V[:k,:]))
return reconst_matrix,s
```

B. Algorithm of SVD Power Method

Input: A matrix $A \in (R)^{n.m}$, a block-vector
$V = V(0) \in R^{m.s}$ and a tolerance tol
Output: An orthogonal matrices
$U = [u_1,u_2,...,u_s] \in R^{n.s}$
$V = [v_1,v_2,...,v_s] \in R^{m.s}$
and a positive diagonal matrix
$\Sigma = diag(\alpha_1,\alpha_2,...,\alpha_s)$
such that : $AV = U\Sigma$
While (err > tol) do
AV = QR(factorization QR),
$U \leftarrow Q(:, 1:s)$
(the s first vector colonne of Q)
ATU = QR,
$V \leftarrow Q(:, 1:s)$ and $\Sigma \leftarrow R(1:s, 1:s)$
$err = ||AV - U\Sigma||$
End

C. Proposed Image Compression Technique

The contribution of this paper is the introduction of the concept of application of Block SVD Power Method to image compression, the main idea of image compression is reducing the redundancy of the image and the transferring data in an efficient form.

We propose our method to integrate the Block SVD Power Method and adopt it to create an algorithm that compress an image. Fig. 1 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 large amount of the image information. With this, we can use only a few singular values to represent the image with slight differences from the original. The input image can be a color image with RGB color components or may be a grayscale image. Additionally, for creating new image with Python SVD function as indicated in the Fig. 1, we use:

$I_{comp} = U(:, 1:K) * \Sigma(1:K, 1:K) * (V(:, 1:K)^T)$ (1)

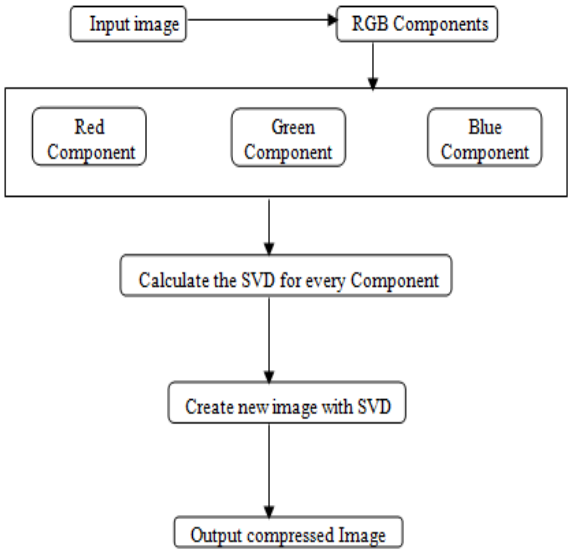


Fig. 1 New Architecture of Image pre-processing using SVD

In this paper is to set up a new algorithm for image compression that overcomes some inconveniences encountered in existing methods that use Python SVD function. Our modification consists of a computing the SVD for each component step, in which the entries in the image I are calculated using Block SVD Power Method obtained by [15] instead of Python SVD function [14] and keeps the K rank determined by (see Eq.5).

We suggesting an image compression based on Block SVD Power Method. Most of methods focus on other methods and other variation of SVD. Moreover, our method is novel, efficient for solving our problem. It is general and many other computer visions can benefit from using it. The results are clearly showing the superiority of the proposed lossy image compression technique over those of Python SVD function and some different compression techniques.

IV. EXPERIMENTAL RESULTS

Main aim of our work is Image compression. Our experiments were performed on several images available on WANG Databases. Simulations were done in Python.

A. Measurement 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) and Peak Signal-to-Noise Ratio (PSNR). These variables are signal fidelity metrics and do not measure how viewers perceive visual quality of an image.

B. Measurement of Compression Ratio

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

$$Q_{comp} = \frac{Size\ of\ Original\ image}{Size\ of\ Compressed\ image} \quad (2)$$

C. Mean Square Error (MSE)

MSE, which for two M * N monochrome images X and Y where one of the images is considered noisy approximation of the other and is defined as follows:

$$e_{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [X(i,j) - Y(i,j)]^2 \quad (3)$$

D. Peak Signal-to-Noise Ratio (PSNR)

PSNR is measured in decibels (dB), and is only meaningful for data encoded in terms of bits per sample bits per pixel. For example, an image with 8 bits per pixel contains integers from 0–255. PSNR is given by the following equation:

$$PSNR = 10\ log_{10} \frac{(2^B - 1)^2}{e_{MSE}} \quad (4)$$

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

A. Image Compression

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

Here, we used 2 color images such as Giraffe and India Gate available in WANG Database and one in grayscale. Figures 4, 5, 6 and 7 show the test images and the resulting compressed images using Python SVD function [14] and the proposed compression method.

We recall that our goal is to approximate an image (matrix of $m \cdot n$) 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.*, [14]:

$$K = \frac{m \times n}{m + n + 1} \quad (5)$$

Where *m* and *n* are the size of original image.

(a) Giraffe

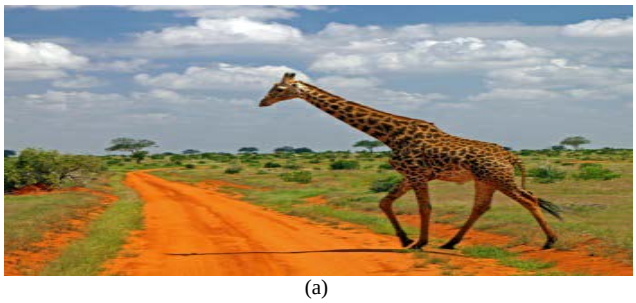
(b) India Gate

(c) Grayscale

Fig. 4 Original Images

B. Analysis with Color Image

After rank K = 438, we obtain:



(a)

(b)

Fig. 5 Image compressed results obtained by:
a. Python SVD function, b. Proposed method

TABLE I IMAGE COMPRESSION RESULTS FOR GIRAFFE.JPG, 1024 × 768, 858KB, BY USING:

	Python SVD function			Proposed method		
K	Q_{comp}	**MSE**	**PSNR**	Q_{comp}	**MSE**	**PSNR**
25	9.8635	30.8213	30.7839	7.5023	46.8792	48.0017
50	9.4127	31.2231	32.8613	7.4123	47.2051	49.6834
75	8.8454	33.4624	34.9174	7.3923	48.2928	50.7678
100	8.2839	35.5016	36.9711	7.3722	49.3804	51.8522
125	8.0601	36.9601	38.5246	7.3628	50.2263	52.6677
150	7.8416	38.4162	40.0752	7.3532	51.0722	53.4831
175	7.7425	39.6034	41.4147	7.3538	51.3384	53.7384
200	7.6453	40.7905	42.7541	7.3543	51.6045	53.9934
225	7.5935	41.8770	43.9656	7.3481	52.7398	55.0594
250	7.5402	42.9643	45.1763	7.3419	53.8751	56.1254
275	7.5041	44.0238	46.3197	7.3371	55.0086	57.2463
300	7.4661	45.0832	47.4630	7.3323	56.1422	58.3672
325	7.4393	46.1445	48.5717	7.3312	57.4115	59.8254
350	7.4123	47.2051	49.6804	7.3301	58.6809	61.2833
375	7.3923	48.2943	50.7653	7.3283	60.5681	63.8087
400	7.3722	49.3834	51.8542	7.3264	62.4553	66.3341
425	7.3628	50.2573	52.6687	7.3234	66.6775	71.6270
438	7.3532	51.0722	53.4831	7.3203	70.8998	76.9199

(a)

(b)

Fig. 6 Image compressed results obtained by:
a. Python SVD function, b. proposed method

TABLE II IMAGE COMPRESSION RESULTS FOR INDIAGATE .JPG, 1024 × 768, 858KB, BY USING

	Python SVD function			Proposed method		
K	Q_{comp}	**MSE**	**PSNR**	Q_{comp}	**MSE**	**PSNR**
25	9.5409	27.6201	33.9037	7.0768	43.7210	46.9405
50	9.2389	28.4011	35.1045	6.9845	44.5122	48.4405
75	8.6088	29.8450	36.2988	6.9623	46.1665	49.8235
100	7.9788	31.2889	37.4932	6.9401	47.8209	51.2066
125	7.7346	32.5600	38.5471	6.9317	49.2150	52.4031
150	7.4904	33.8312	39.6011	6.9234	50.6091	53.5996
175	7.3713	35.0760	40.6256	6.9227	51.0844	54.0164
200	7.2523	36.3209	41.6502	6.9221	51.5598	54.4332
225	7.1760	37.5911	42.7026	6.9072	53.7449	56.6567
250	7.0998	38.8613	43.7551	6.8923	55.9301	58.8803
275	7.0621	40.2133	44.8726	6.8863	58.021	61.0707
300	7.0245	41.5653	45.9901	6.8804	60.1123	63.2612
325	7.0028	43.0392	47.2161	6.8779	62.271	65.2258
350	6.9811	44.5132	48.4421	6.8754	64.4297	67.1905
375	6.9595	46.1671	49.8244	6.8720	67.6065	70.9153
400	6.9379	47.8211	51.2067	6.8687	70.7834	74.6401
425	6.9306	49.2151	52.4031	6.8687	77.7668	83.8061
438	6.9234	50.6091	53.5996	6.8688	84.7503	92.9721

C. Analysis with Grayscale Image

In order to compare this performance, we also applied the new method to the gray scale image After rank K = 548, we obtain:

(a)

(b)

Fig. 7 Image compressed results obtained on the: a. Python SVD function, b. proposed method

TABLE III IMAGE COMPRESSION RESULTS FOR GRAYSCALE.JPG, 1024 × 960, 480KB, BY USING

	Python SVD function			Proposed method		
K	Q_{comp}	**MSE**	**PSNR**	Q_{comp}	**MSE**	**PSNR**
25	4.9878	80.3421	27.6723	4.0621	9.5381	39.5372
50	4.9789	78.5091	29.2222	4.0589	9.4523	38.4098
75	4.6460	55.8911	31.08	4.0693	7.5163	39.5551
100	4.3132	33.2732	32.9378	4.0798	5.5803	40.7005
125	4.2067	25.1121	34.4039	4.1016	4.5246	41.7318
150	4.1002	16.9510	35.8701	4.1234	3.4689	42.7631
175	4.0779	13.2016	37.1421	4.1219	2.8566	43.7176
200	4.0556	9.4523	38.4142	4.1205	2.2443	44.6722
225	4.0699	7.5168	39.5574	4.1053	1.8652	45.5686
250	4.0843	5.5813	40.7006	4.0901	1.4861	46.4651
275	4.1037	4.5258	41.732	4.0756	1.2432	47.3227
300	4.1231	3.4704	42.7634	4.0611	1.0004	48.1803
325	4.1237	2.8574	43.7169	4.0531	0.8417	49.0217
350	4.1243	2.2444	44.6705	4.0452	0.6831	49.8631
375	4.1082	1.8626	45.5675	4.0346	0.5765	50.6617
400	4.0921	1.4808	46.4645	4.0241	0.4699	51.4603
425	4.0796	1.2405	47.3226	4.0198	0.3855	52.4072
450	4.0671	1.0003	48.1808	4.0156	0.3011	53.3541
475	4.0551	0.8422	49.0215	4.0139	0.2422	54.4873
500	4.0432	0.6842	49.8622	4.0123	0.1834	55.6206
525	4.0336	0.5770	50.6612	4.0112	0.1353	57.2296
548	4.0241	0.4699	51.4603	4.0101	0.0872	58.8387

D. Analysis with Other Methods

To evaluate the robustness of our scheme, we test it with other methods like: [10, 13, 14]. Added experiment results for two images are listed in Table IV.

TABLE IV IMAGE COMPARISON WITH VARIOUS ALGORITHMS

	Color image (Fig. 4a)			Grayscale image (Fig.4c)		
	Q_{comp}	**MSE**	**PSNR**	Q_{comp}	**MSE**	**PSNR**
BTC method [13]	9.2713	62.0951	30.2004	5.3912	16.1183	26.0905
BTC method [10]	7.3406	7.9635	39.1261	3.9808	19.0298	35.3689
BTC method [14]	6.7203	2.7451	43.7507	2.8441	3.4708	42.7644
SVD method [14]	7.3508	0.2900	53.4831	4.0261	0.4722	51.4612
Proposed method	7.3107	0.0013	76.9199	4.0110	0.0804	58.8387

In this paper, the proposed algorithm is compared with the Python SVD function [14] and the other state-of-the-art algorithms. When applying the proposed method to image compression, Figs. 5, 6 and 7, it is shown that the compressed images by two approaches are similar to original images. But the human visual response to image quality is insufficient.

We compare the performances of the proposed method, several values were used in this study to measure the quality of the compressed image. We will 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 the quality of the image.

This analysis shows the comparison when SVD and proposed method are applied on the original images. In these experiments, we used the *K* rank for different images. We see in this case that the compression ratio and PSNR, and other values of images varied when changing the rank of image during the SVD process as showed in Tables 1, 2 and 3, and it is evident that the proposed technique gives better performance compared to the SVD. In addition, for the Python SVD function, the value of *K* which provides better PSNR value is the maximum value of $K = 438$, while for the proposed technique, a better, compression ratio, PSNR is provided from $K = 150$ for color images. We concluded that our proposed method result is 1/3 of K rank compare to other methods.

Concerning the grayscale image analysis, it seems that the value of *K* which gives better PSNR value is the maximum value of $K = 548$, while for the proposed method, a better, compression ratio, PSNR is provided from $K = 400$.

We compared the proposed algorithm with the other algorithms as shown in Table IV. Hence, show our proposed algorithm performs com-parable to other existing techniques. It is able to produce a compressed image with better visual quality, as indicated by its PSNR.

V. CONCLUSION

In this work a novel method for image compression, this technique is very simple, and it can be used to overcome limitations of existing algorithms, that used in the Python SVD function. The results shown that the proposed approach might be considered as a solution for the development of image compression. Our proposed method of image compression is provided faster due to the minimum number of iterations in the compression algorithm.

VI. FUTURE SCOPE

The future scope of this work is using the SVD 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.

REFERENCES

[1] A. J. Madhuri, "Digital Image Processing. An Algorithmic Approach," pp. 175–217. PHI, New Delhi, 2006.

[2] M. J. Weinberger, G. Seroussi and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Transactions Image Processing*, Vol. 2, pp. 1309–1324, 2000.

[3] M. A. Alkhalayleh and A. M. Otair, "A new lossless method of image compression by decomposing the tree of Huffman technique," *International journal of imaging & rob*otics Vol. 15, No. 2, pp. 79–96, 2015.

[4] W. Jianji, Z. Nanning, L. Yuehu and Z. Gang,"Parameter analysis of fractal image compression and its applications in image sharpening and smoothing," *Signal Processing: Image Communication journa*l, Vol. 28, pp. 681– 687, 2013.

[5] A. Bilgin, W. Michael, M. Marcellin, I. Altbach, "Compression of electrocardiogram signal using JPEG2000," *IEEE Transactions on Communications Electronics* (ICIP), Vol. 49, No. 4, pp. 833–840, 2003.

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

[7] M. Adiwijaya, B. K. Dewi, F. A. Yulianto and B. Purnama, "Digital image compression using graph coloring quantization based on wavelet SVD," *Journal of Physics Conference Series*, Vol. 423, No. 1, pp. 012-019, 2013.

[8] A. Ranade, S. S. Mahabalarao and S. Kale, "A variation on SVD based image compression," *Image and Vision Computing journal*, Vol. 25, No. 6, pp. 771–777, 2007.

[9] M. Doaa and A. F. Chadi, "Image compression using block truncation coding," *Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. (JSAT*), February, 2011.

[10] E. J. Delp and O.R. Mitchell, "Image compression using block compression", *IEEE Transactions onCommunications* Vol. 27, No. 9, pp. 1335–1342, 1979.

[11] 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* (WCEsp05), Yunlin, pp. 17–18, Nov. 2005.

[12] N. K. El Abbadi, A. Al Rammahi, D. S. Redha and M. Abdul-Hameed, "Image compression based on SVD and MPQ-BTC", *Journal 0f Computer Sc*ience, Vol. 10, No. 10, pp. 2095–2104, 2014.

[13] K. El Asnaoui and Y. Chawki, "Two new methods for image compression," *International journal of imaging & robotics*, Vol. 15, No. 4, pp. 1–11, 2015.

[14] A. H. Bentbib and A. Kanber, "Block power method for SVD decomposition," *Analele Stiintifice ale Universitatii Ovidius Constanta Seria Matematic*, Vol. 23, No. 2, pp. 45–58, 2015.