Mathematics for Engineering Research

[ME5701]

Image Compression by Fast Fourier Transform, Gaussian Low-Pass Filter, and Singular Value Decomposition

By

Jin Hoontae

Department of Mechanical Engineering

National University of Singapore

Submitted on 6th, December 2021

Abstract

Three methods (Fast Fourier Transform (FFT) with and without Gaussian low-pass filter and Singular Value Decomposition (SVD)) were coded in MATLAB and used to evaluate applicability for image compression. The FFT algorithm ensures fast computation of the discrete Fourier Transform (DFT) with large matrices and draws a diagram of frequency coefficients that shows the regime of the important data in the frequency domain, while the SVD is a method capable of decomposing a matrix into three sub-matrices regardless of its size. While frequency coefficients are controlled to construct compressed images for FFT, different values of the rank *r* are used to produce compressed images for SVD. Both methods were proven to be practical and efficient for such a performance. Further, the Gaussian filter was used to suppress high spatial frequencies and it was found that the quality of the compressed images was as good when the appropriate cutoff frequency value was selected. A comparative analysis was done to determine the most effective method by conducting an image quality examination, analyzing distortion percent and Peak signal-to-noise ratio (PSNR). From the results and analysis, it was concluded that FFT without the Gaussian filter is a superior option for image compression.

1. Introduction

Uncompressed images oftentimes require high amounts of storage space and transmission cost. In modern society, the use of digital images is indispensable due to the extremely widespread use of the Internet, multi-media platforms, and other applications for medicine, surveillance, etc. Methods for efficient image compression are sought to reduce unnecessary, neglectable, and redundant data while not impairing or blurring original images to an acceptable extent. The amount of redundant data removed is not always the same, but it depends on the purpose of its use, and the viewer or user of the image [1, 2].

Discrete Fourier Transform (DFT) has been extensively used by means of efficient implementations for image compression, and the Fast Fourier Transform (FFT) has subsequently been introduced and actively utilized to improve the compression processing performance. Although the same results are achieved by both DFT and FFT, the latter significantly saves the processing time and provides more interpretable information when the frequency domain is computed. The FFT, which was discovered in 1965 by Cooley and Tukey,

immensely revolutionized the field of Digital Signal Processing (DSP) because its algorithms reduce the number of complex calculations considerably as compared to DFT [3]. The fundamental characteristic of the FFT is that it exploits the symmetry and periodicity properties of DFT thereby discretizing the DFT sequence of length N into progressively smaller DFTs [4]. The number of complex multiplications is reduced from N^2 to $\left(\frac{N}{2}\right)\log_2 N$ and so is the number of additions from $N(N-1)$ to $\mathrm{N}\cdot\log_2 N$ by the FFT.

High frequency components of its Fourier Transform of an image are comprised of features such as edges and noises in the image. The gaussian low-pass filter is commonly used to realize the smoothing in the frequency domain by attenuating such high spatial frequencies [5]. Hence, one feasible approach can be assumed; that is, images containing high frequency components can be compressed more efficiently after smoothing them with the Gaussian filter priorly.

Many studies have been widely done to investigate the use of Singular Value Decomposition (SVD) for image compression [6-9]. The SVD has the ability to approximate an image sufficiently by a matrix of low rank and this approximated image with a low rank can be represented as compactly as the original image thereby efficiently reducing the storage space.

In this paper, three different methods (FFT, Gaussian low-pass filter, and SVD) for image compression were studied, and then the results and performances were analyzed and compared analyzing the compressed images, the image distortions and Peak signal-to-noise ratio (PSNR) versus compression ratio.

2. 1-D Discrete Fourier Transform

One dimensional DFT function is generally used for signal processing such as audio, radar, and biomedical signals, and the function is governed by one independent variable with respect to time [4]. DFT obtains complex-valued frequency components by transforming and decomposing time-domain components.

The DFT for a 1-D signal is expressed as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \tag{1}$$

for n = 0, 1, ……, N-1.

In the equation, since n is the time domain index of the discrete function, $x(n)$ is the time domain, and $X(k)$ represents the complex-valued frequency components where k is the index of the normalized frequency spectrum. W_N^{nk} is called Twiddle factor, which is the simplified exponential form of $e^{j2\pi nk/N}$. The exponential term can be generally expressed using Euler's formula as below:

$$e^{j\theta} = cos\theta + jsin\theta \tag{2}$$

The inverse DFT is given by:

$$x(n) = {}^{1}/_{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk} \tag{3}$$

for k = 0, 1, ……, N-1.

3. 2-D Discrete Fourier Transform

Two dimensional DFT function can be utilized for digital image processing, and the function is governed by two variables. The DFT for a 2-D signal is expressed as:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) W_N^{(ux+vy)} \tag{4}$$

for x and y = 0, 1, 2, ……, N-1.

The inverse DFT is given by:

$$f(x,y) = (1/MN) \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) W_N^{-(ux+vy)} \quad (5)$$

for u and v = 0, 1, 2, ……, N-1.

In the computation of DFT for image processing, the crucial component is Twiddle factor. The component is considered as a rotating vector that performs the incrementing process according to the number of samples and can produce a symmetrical butterfly diagram when it is utilized with the FFT algorithm [11].

4. Fast Fourier Transform

FFT can be regarded as an upgraded algorithm for faster computation of DFT [10], which efficiently computes the DFT and its inverse. Especially, FFT shows its superior effectiveness when the matrix is extremely large. This is because, as the size of a matrix increases, the computation speed is exponentially increased. Table 1 illustrates how efficient the FFT can perform compared to the DFT for the high numbers of N.

Table 1. Efficiency comparison between DFT and FFT in terms of the number of computations

N	Number of Required Computations		Computation Ratio
	DFT (N^2)	FFT ($N/2 \log_2 N$)	
16	256	32	8:1
256	65536	1024	64:1
512	262144	2304	114:1
1024	1048576	5120	205:1
2048	4194304	11264	372:1

The FFT-transformed domain is comprised of low and high frequency quantized coefficients [11]. In the regime of the quantized high frequency coefficients, some of their values are nearly close to zero. These values are considered insignificant and can be removed. The frequency removal process leads to the stage for image compression and the reconstructed image is created using the compression ratio and analyzed with visual checking, distortion and PSNR measurements. The compression ratio is a percentage expressed by dividing the number of

elements in the reconstructed image by the number of elements in the original image, and any PSNR value less than 35dB is considered indistinguishable from original images.

5. Gaussian Low-Pass Filter

The equation for the image filtering process can be expressed as:

$$G(u,v) = F(u,v) \cdot H(u,v) \tag{6}$$

Where $G(u,v)$: Filtered frequency response, $F(u,v)$: Frequency response of the original image, and $H(u,v)$: Symmetric filter function. By convoluting the original image with the transfer function, a filtered image can be created.

Gaussian filter in the frequency domain is governed by the standard deviation, which determines the size of the kernel. The high values are located around the center of the kernel and lower values are located further from the origin, and this behavior is characterized by the normal distribution. The Gaussian low-pass filter can be expressed as:

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2} = e^{-D^2(u,v)/2{D_0}^2} \tag{7}$$

Where $\sigma = D_0$: Standard deviation or Cutoff frequency, and $D(u,v)$: Distance between a point (u, v) and the center of the frequency domain.

6. Singular Value Decomposition

The SVD function divides a single matrix (A) into three sub-matrices ($U \sum V^T$) to represent the data. Given the matrix A with the size of mxn, U is represented as a matrix of mxm, Σ is a diagonal matrix with the same dimension as A, and V^T is a matrix of nxn. SVD is a method of diagonalizing a matrix like eigen-decomposition. The uniqueness of this method is that it is applicable to all mxn matrices regardless of whether the matrix is a square matrix or not. The SVD for any mxn matrix in real space is defined as:

$$A = U \sum V^T \tag{8}$$

where U: mxm orthogonal matrix ($AA^T = U(\Sigma\Sigma^T)U^T$), V: nxn orthogonal matrix $A^TA = (V(\Sigma^T\Sigma)V^T)$, and Σ: mxn rectangular diagonal matrix.

As described in equation 8, U is an orthogonal matrix obtained by eigen-decomposition of AA^T, and the column vectors of U are generally called left singular vectors of A. In addition, V is an orthogonal matrix obtained by decomposing A^TA into eigenvalues, and the column vectors of V are called right singular vectors. Since U and V are orthogonal matrices, the following relationships are true: $UU^T = VV^T = E,\ U^{-1} = U^T,\ and\ V^{-1} = V^T$. Σ is a mxn rectangular diagonalized matrix with eigenvalues that result from the decomposition of AA^T or A^TA, and its diagonal elements are in descending order and called singular values of A. Hence, the matrix, A, can be represented as:

$$A = [u_1 \quad u_2 \quad \cdots \quad u_m] \begin{bmatrix} \Sigma_1 & 0 & \cdots & 0 \\ 0 & \Sigma_2 & \cdots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \Sigma_n \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}$$

Since the matrix of Σ is arranged in descending order ($\Sigma_1 > \Sigma_2 > \cdots > \Sigma_n > 0$), the bottom row values close to 0 can be removed [12] as they are assumed to contain unnecessary information.

A rank, *r*, approximation to the matrix A can be written as:

$$A_r = U_r \Sigma_r V_r^T \tag{9}$$

Σ_r is the $r \times r$ submatrix of Σ, U_r contains the first r columns of U, and V_r^T consists of the first r rows of V^T [13]. The SVD decomposition can be used to approximate the best rank r to pack the maximum energy from matrix A. For compression, given a matrix A of $N \times N$, unlike A that contains N^2 entries, it only takes 2Nr+r entries in U_r, Σ_r, V_r^T for the SVD decomposition. The important characteristic of the $U \sum V^T$ matrix is that the most significant data that represent the image are stored on the top column and rows of each sub-matrix [14]. Hence, from image analysis, it can be found that even with a small rank *r*, A_r is capable of producing an adequately great image quality that contains as compact energy as that of the original matrix A if an appropriate value is used.

7. Method

The three methods, FFT with and without Gaussian low-pass filter and SVD, were used to compress an image. The algorithms were written based on their actual equations (Eq 4 to 9) and the built-in functions were not used to comprehend the mechanisms of the methods. The codes were designed to compress any $m \times n$ image where the entry (*i,j*) is interpreted as the brightness of a pixel. The image "Jin_Hoontae.jpg", self-photographed by the author, was used to demonstrate the image compression performance for the methods.

The original RGB image contained 24 bits for each pixel (8 for red, 8 for green, and 8 for blue). In order to reduce the storage space, the image was then converted into a grayscale image where only 8 bits were needed per pixel.

Two equations were further used to validate the reconstructed images and investigate their trends with respect to the increased compression ratio: Distortion percent and PSNR.

The distortion percent of the compressed image can be approximated as [15]:

$$D = \frac{||A-O||^2}{||A||^2} \tag{9}$$

where A: the original image in grayscale, and O: an approximated image compressed by the functions.

The peak signal-to-noise ratio (PSNR) represents the power of noise to the maximum power that a signal can have. It is mainly used to evaluate the loss information of the image, which can be written as:

$$PSNR = 10 \cdot \log_{10}(\frac{MAX_I^2}{MSE}) \tag{10}$$

where MAX_I: the maximum possible pixel value of the image, MSE: the mean squared error.

The proposed method procedures are as follows:

1. Read the input RGB image file and convert it to a grayscale image.

2. Compute the FFT of the image using Eq.4 and analyze the frequency spectrum by shifting the low frequency coefficients located at the edges to the center for better visualization.

3. Eliminate low-value coefficients and compute the IFFT using Eq. 5 to produce four reconstructed images containing 30%, 5%, 1%, and 0.2% remaining frequency coefficients.

4. Perform the same steps (2 to 3) after applying a Gaussian low-pass filter to the original image.

5. Investigate the frequency spectrums of each compressed image and analyze the changes compared to the original one.

6. Compute the SVD of the original image using Eq. 8 and compress the image by a low-rank r (=10, 15, … , 50) using Eq. 9.

7. Calculate the distortion level and PSNR for FFT and SVD and compare for validation.

8. Experimental Results and Analysis

8.1. Compressed Images by FFT

In Figure 1, the first bottom left frequency image displays the frequency spectrum transformed from the original image. The highest magnitude values regarded as the lowest frequency components are located near the origin (0, 0), and the magnitude level decays as it goes further from the origin, showing that the high frequency components are formed around the edges. This implies that lower frequency modes generally have more energy in images and play a significant role in controlling the compression ratio as they determine important image features and quality. As shown in the figure, by diminishing the percentage of the overall frequency coefficients from 30% to 0.20005% gradually, the regime of the bright and low frequency coefficients becomes smaller. When the reconstructed frequency domains are transferred back to the gray images, it can be seen that low frequency coefficients enormously determine the quality of the images as the blurry level becomes exacerbated with respect to the decreasing regime of low frequency coefficients. For the reconstructed image with only 5 percent of the low frequency components, it still holds an excellent visual quality without impairing important features and is clearly distinguishable from the original image. This demonstrates that the FFT is capable of removing a large amount of redundant data in the image, leading to an efficient compression performance.

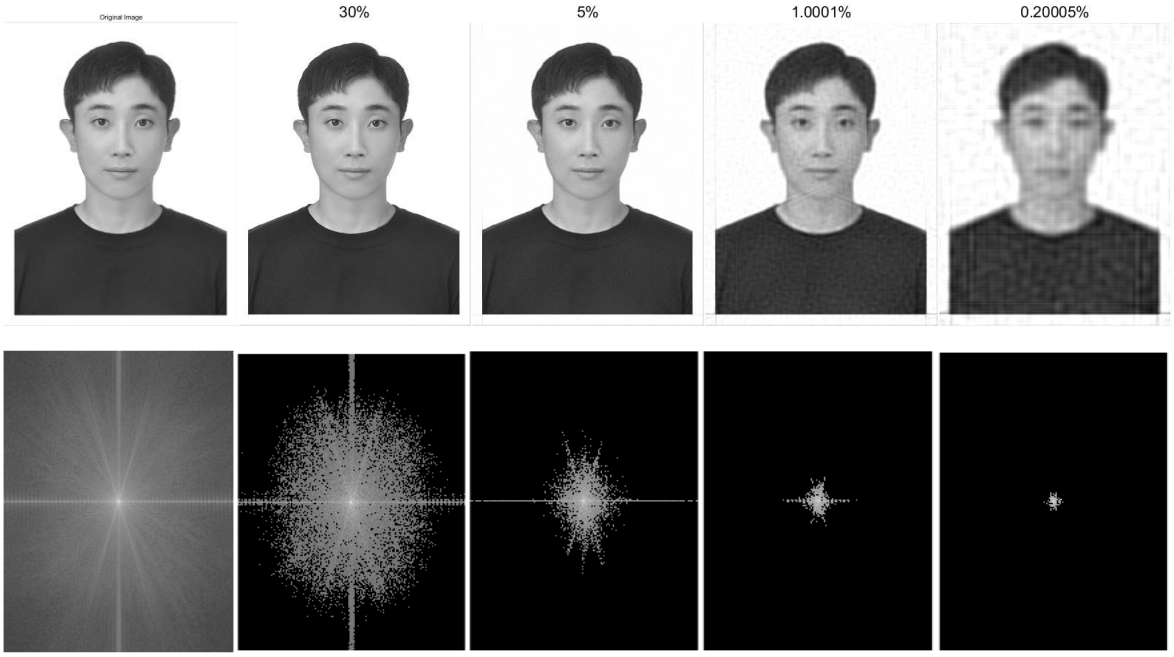

Figure 1. The comparison between original and reconstructed images with respect to the different percentages of frequency coefficients

8.2. Compressed Images by FFT with Gaussian Low-Pass Filter

Five different Gaussian low-pass filters were applied to the original image, and each reconstructed image was compared to determine the most suitable image to proceed with the compression performance for eliminating low frequency components. **Figure 2** illustrates the behavior of the gaussian filter. As the value of the cutoff frequency increases, the regime of the low frequency components expands as shown in the figure, which produces an image with the better quality once convoluted with the frequency spectrum of the original image. At the cutoff frequency values of 240 and 300, the reconstructed frequency spectrums contain a significant amount of low frequency components, which could perhaps lead to the same performance outcome of **Section 8.1**. Hence, the cutoff value of 180 was selected to further compress the image.

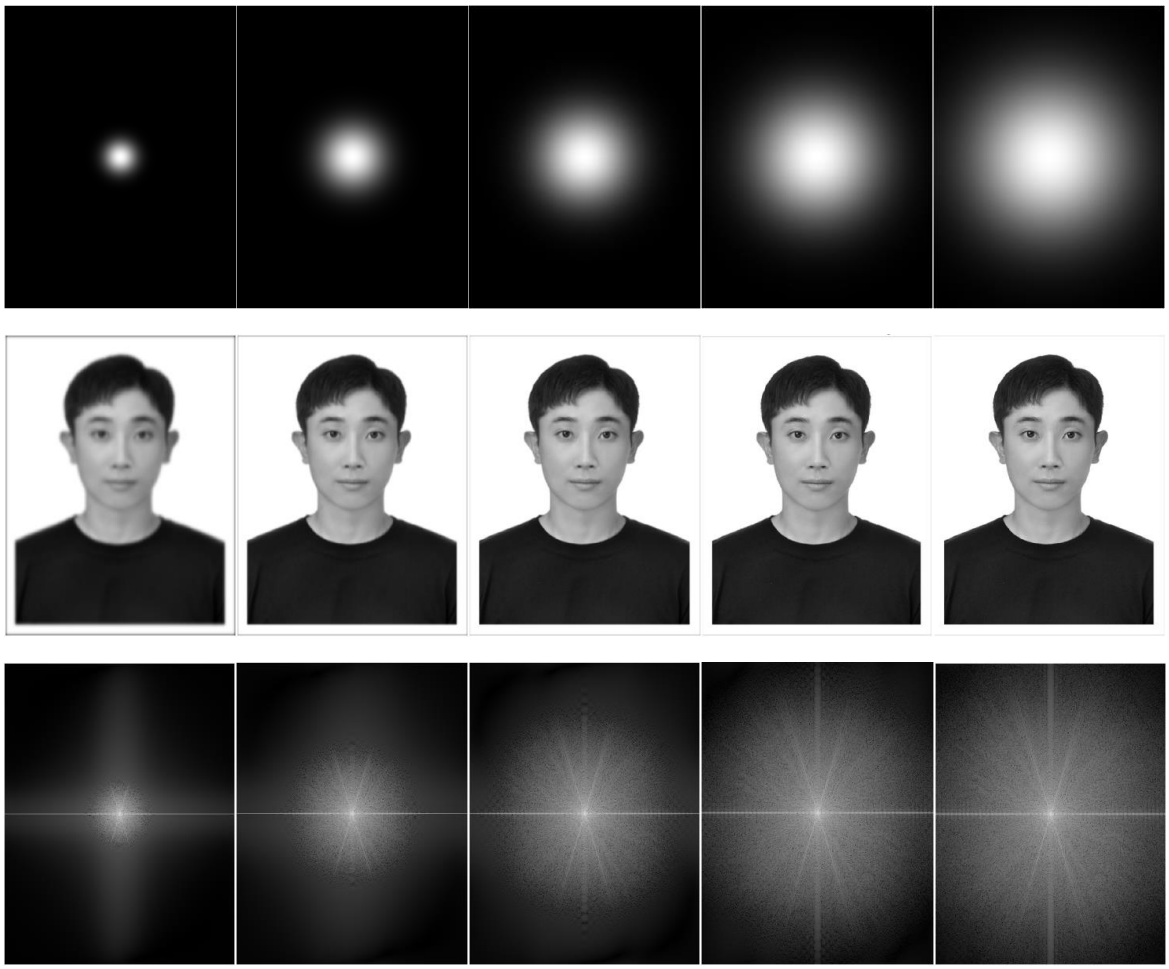

Figure 2. Gaussian Filters at the cutoff frequency values of 60, 120, 180, 240, and 300 (from left to right), the reconstructed images convoluted with the filters, and the reconstructed frequency spectrums after the convolution process

In **Figure 3** below, the top images show the compressed images after eliminating certain percentages of frequency components and the bottom images illustrate the amount of the remaining components in the frequency domain. Although the same percentages were diminished as done in **Figure 1**, the frequency regimes hold seemingly a round shape as compared to the ones that were not filtered by the Gaussian filter in **Section 8.1**. In terms of the image quality by eye, the Gaussian-filtered/compressed images did not produce blurriness although high spatial frequencies were priorly eliminated. In theory, the filter is defined as a frequency transfer function, which suppresses high spatial frequencies while preserving low spatial frequencies, which results in smoothing the image. Hence, under the same conditions, for the images with and without the filter, the filtered image is deemed to contain fewer frequency components when compressed due to the elimination of high spatial frequencies beforehand. In order to further analyze the difference, the distortion and PSNR measurements were performed, which will be covered in **Sections 8.4 and 8.5**.

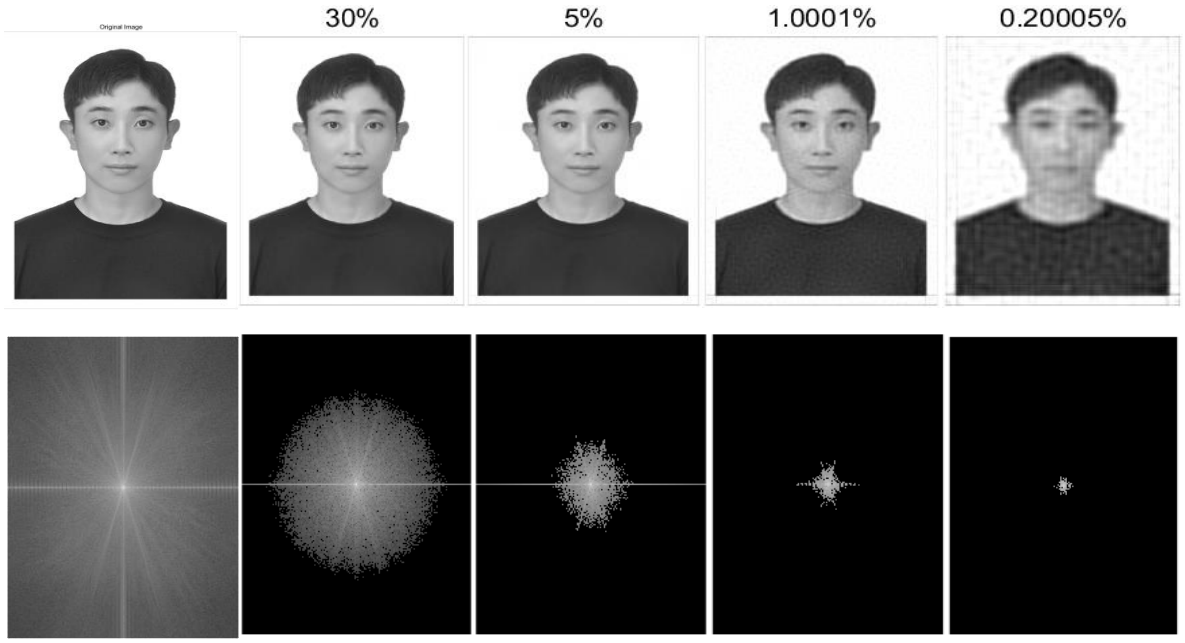

Figure 3. The comparison between original and reconstructed images with respect to the different percentages of frequency coefficients

8.3. Compressed Images by SVD

The SVD truncation algorithm was run to analyze the image quality depending on the size of low-rank *r* using equation 9 as shown in Figure 3. The low-rank SVD approximations only keep dominant singular vectors. For example, when r is equal to 50, the algorithm approximates the entire image by 50 linearly independent rows and columns. During the compression process, although some amount of image loss occurs, important features can remain unimpaired as long as the rank *r* does not decrease extremely low. From the images in the figure, it is obvious that the image begins to become blurry when the singular value *r* decreases below 25, which means that, for the sample image given in this research, at least up to the 25th rank of the decomposed matrices contains important data for determining the image quality. With the rank equal to 30, the image quality is still adequately visible although it only stores 8.17% data of the original image. This proves that the SVD algorithm also works efficiently for image compressions like the conventional image processing methods such as Fourier Transforms and Wavelet Transforms.

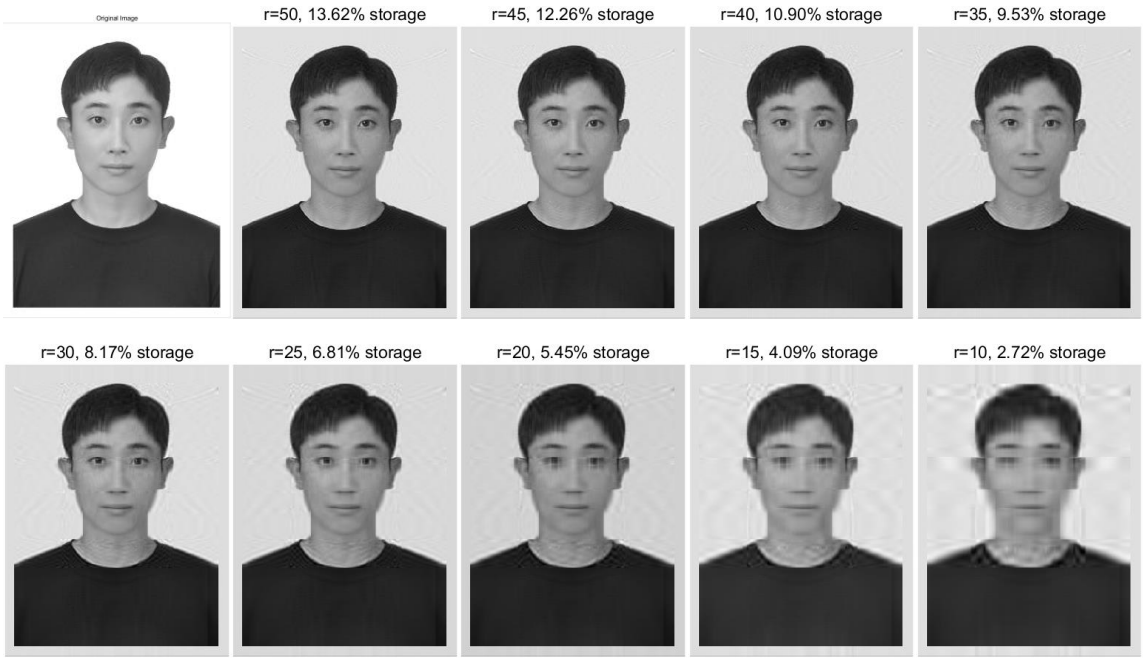

Figure 3. The reconstructed images by the SVD method

8.4. Distortion percent analysis between FFT and SVD

For SVD compression, it can be seen from **Figure 4** that the distortion percent is significantly high when the compression ratio is low. The lowest SVD compression ratio calculated here is 0.01 with the distortion percent close to 95%, which means that the compressed image is completely indistinguishable from the original image due to the tremendous loss of the important data eliminated by the low-rank value. As the compression ratio increases, the distortion percent begins to decline exponentially and form a flattening line starting at the compression ratio greater than 0.2. For FFT compression with and without the Gaussian filter, even when the lowest compression ratio is nearly zero and lower than that of SVD, its distortion percent values remain relatively lower. However, as the compression ratio increases, the FFT compression algorithm with the Gaussian filter shows a poor trend in which the distortion percent does not decrease below 0.2 when the cutoff frequency value of 180 was used, forming a flattening line in an early stage. The poor results are significantly affected by the cutoff frequency value, as it determines the amount of suppression of spatial frequencies beforehand. As shown in the Figure, when the high cutoff value of 300 is selected, the distortion percent decreases further down. On the other hand, the behavior of distortion by FFT without the filter performs better than the other two approaches at all points. Hence, the graph demonstrates that the FFT algorithm without the filtering process is superior to the SVD algorithm as it produces images with less distortion percent and finer quality.

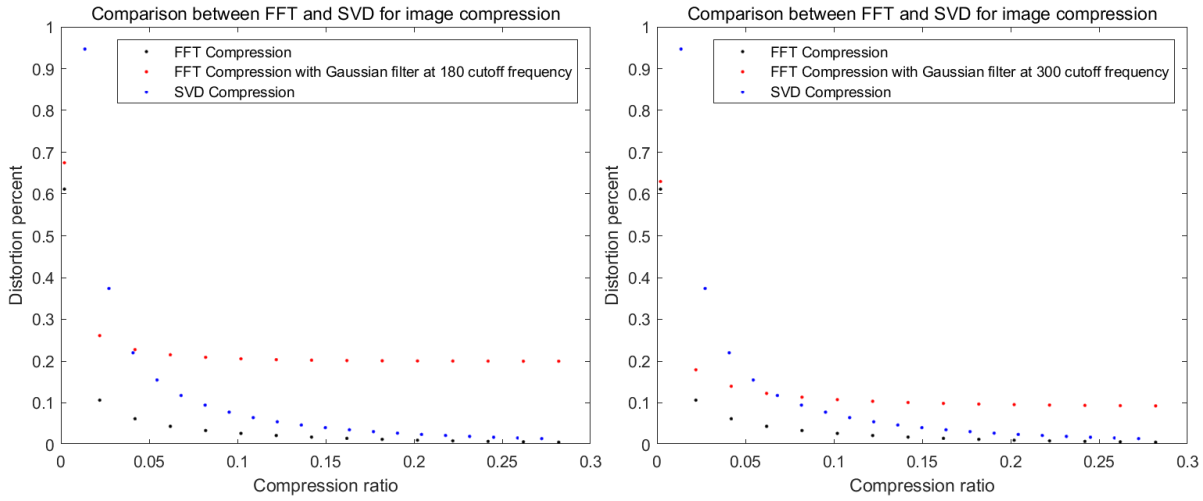

Figure 4. A graph for Distortion Comparison between FFT and SVD

8.5. PNSR analysis between FFT and SVD

Any PSNR value equal to or above 35dB indicates that the compressed image is adequately distinguishable from the original image [4]. As shown in **Figure 5**, PSNR values continuously increase along with the values of compression ratio for FFT without the filter and SVD methods. However, the FFT with the filter algorithm is unable to produce a fine image whose PSNR is higher than 35dB, which shows a significant disadvantage of the Gaussian filter as its primary purpose of function is to smooth an image and remove Gaussian noise. While FFT-compressed images without the filter obtain PSNR values greater than 35dB starting at the compression ratio of 0.06, SVD-compressed images are not able to obtain fine PSNR values until the compression ratio reaches 0.12. Furthermore, it can be seen that the FFT algorithm predominantly produces PSNR values higher than the SVD algorithm at all times, which explains that images reconstructed by the FFT without the filter always generate finer image quality at the same compression ratio in this case.

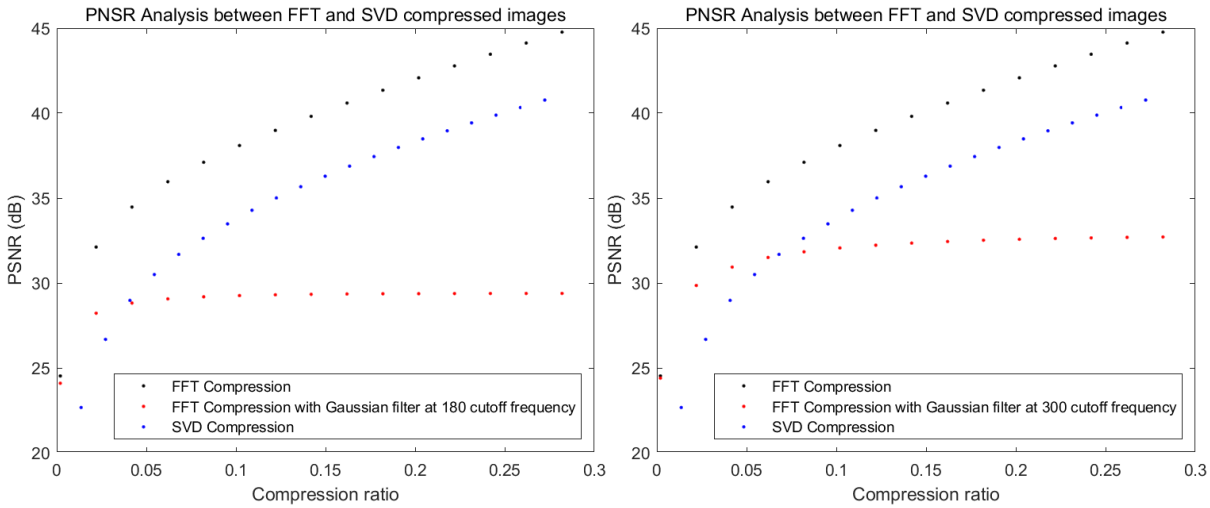

Figure 5. A graph for PNSR analysis between FFT and SVD compressed images

9. Conclusion

In conclusion, although using SVD for image compression was proven to be practical and applicable, according to the results obtained in this research, it did not show a better performance than FFT in terms of image distortion and PSNR, which are paramount factors in image processing. Even though the Gaussian filter was able to produce visually un-impaired and fine images within the certain range of the compression ratio, the distortion percent and PSNR values did not show any promising results as compared to the other two methods, meaning that applying the filter would consume more time without improvements if only used for the image compression purpose. However, if the filter is used for an image with high noises, different results could be shown as its fundamental role is to remove noises created by high frequencies. It was found that frequency coefficients govern the image quality more effectively than the rank of decomposed matrices at every point of the compression ratio. Thus, FFT remains a better option for this particular purpose, and further research on similar types of Fourier Transform such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) with or without the filter can be carried out to discover a better method for compressing images containing noises.

10. References

[1] Sayood K. Introduction to data compression. 2nd ed. Academic Press, Morgan Kaufman Publishers; 2001.

[2] Rao KR, Yip P. Discrete cosine transform: algorithms, advantages, applications. San Diego, CA: Academic Press; 1990.

[3] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, Discrete -Time Signal Processing, Prentice Hall, Second Edition, pp. 646-652, 1999.

[4] Anitha T. G. and Ramachandran S., "Novel algorithms for 2-D FFT and its inverse for image compression," 2013 International Conference on Signal Processing, Image Processing & Pattern Recognition, pp. 62-65, 2013

[5] Xu J., Ling Y. and Zheng X., "Forensic detection of Gaussian low-pass filtering in digital images," 2015 8th International Congress on Image and Signal Processing (CISP), pp. 819-823, 2014

[6] Andrews H.C., Patterson C.L., "Singular value decomposition (SVD) image coding," IEEE Transactions on Communications 24, pp. 425– 432, 1976

[7] Goldrick C.S.M., Dowling W.J., Bury A., "Image coding using the singular value decomposition and vector quantization, in: Image Processing and its Applications," IEE, pp. 296–300, 1995

[8] Waldemar P., Ramstad T.A., "Image compression using singular value decomposition with bit allocation and scalar quantization," in: Proceedings of NORSIG Conference, pp. 83–86, 1996

[9] Yang J.-F., Lu C.-L., "Combined techniques of singular value decomposition and vector quantization for image coding," IEEE Transactions on Image Processing 4(8), pp. 1141–1146, 1995

[10] Narkhede N. D., Salunke J. N., Narkhede N., "Fundamentals and Literature Review of Discrete Fourier Transform in Digital Signal Processing," *International Journal on Recent and Innovation Trends in Computing and Communication*, 4(11), pp. 295, 2016

[11] AnithaT G., Vijayalakshmi K.. "FFT Based Compression approach for Medical Images.", *International Journal of Applied Engineering Research*, 13(6), pp. 3550-3567, 2018

[12] Sandhu K., Maninder S. Er., "Image Compression Using Singular Value Decomposition (svd)", *International Journal of Latest Research in Science and Technology,* 7(5), pp 5-8, 2018

[13] Ranade A., Mahabalarao S. S., Kale S., "A variation on SVD based image compression",

Image and Vision Computing, 25(6), pp. 771-777, 2007
[14] Mathews B., “Image Compression using Singular Value Decomposition (SVD)”, 12 December 2014, The University of Utah
[15] Cheepurupalli V., Tubbs S., Boykin K. and Naheed N., "Comparison of SVD and FFT in Image Compression," 2015 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 526-530, 2015