Advanced Engineering Science
http://publish.mersin.edu.tr/index.php/ades
e-ISSN 2791-8580

Leveraging SVD for efficient image compression and robust digital watermarking

Ornela Gordani [*1], **Aurora Simoni** [1]

[1] *University of Tirana, Faculty of Natural Science, Department of Applied Mathematics, Tirana, ornela.gordani@fshn.edu.al, aurora.simoni@fshn.edu.al*

Cite this study: Gordani, O., & Simoni, A. (2024). Leveraging SVD for efficient image compression and robust digital watermarking. Advanced Engineering Science, 4, 103-112

Keywords
Image
Algebra
Watermarking
Compression
Image processing

Research Article
Received: 21.12.2023
Revised: 12.05.2024
Accepted: 23.08.2024
Published: 03.09.2024

Abstract

In terms of digital files, compression is the act of encoding information using fewer bits than what's found in the original file. When we say image compression, we have in mind an image that has fewer bytes than the original image but has the most important features that describe the original image. So, the aim of image compression is to reduce the image size without degrading image quality below an acceptable threshold. In MATLAB, an image is stored as a matrix. One approach is to apply the Singular Values Decomposition (SVD) to the image matrix. This method is implemented in MATLAB. In order to divide the matrix of the given image into three other matrices in MATLAB, we can use the function *svd()*. As performance metrics, we can use PSNR and Compression ratio. Digital Watermarking is defined as the process of hiding a piece of digital data in the cover data which is to be protected and extracted later for ownership verification. In an SVD-based watermarking scheme, the singular values of the cover image are modified to embed the watermark data. All tests and experiments are performed using MATLAB as the computing environment and programming language. Also, in the RStudio programming language we can see the implementation of the SVD method in image compression.

1. Introduction

The concept of Singular Value Decomposition, as we know it today, emerged in the mid-20th century. It was independently introduced by several researchers. In 1936, Eugenio Beltami introduced the concept, but it was not widely recognized at the time [1]. In 1950, Peter Henrici discussed a form of SVD in the context of numerical analysis [2]. In 1955, Roger Penrose independently rediscovered the SVD and highlighted its significance [3]. The method gained more attention in the 1960s and 1970s as computers became more powerful, allowing for efficient numerical implementation of the SVD [4-12]. Today, SVD method is a standard technique in various fields, including machine learning, image and signal processing, statistics, and data science.

The main objective of image compression is to reduce the redundancy of the image data to be able to store or transmit data in an efficient form. Image compression may be *lossy* or *lossless*. Lossy methods [13] are especially suitable for natural images such as photographs in applications in which minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. In other hand the Lossless compression is preferred for archival purposes.

Digital Watermarking is used for a wide range of applications, such as: copyright protection, source tracking, broadcast monitoring, medical application, improvement of legacy systems, authentication & tamper detection, usage control, ownership identification etc [14]. A digital watermarking can be visible or invisible. A visible watermark typically consists of a conspicuously visible message or a company logo indicating the ownership of the image. On the other hand, an invisible watermarked image appears very similar to the original. The existence of a watermark can only be determined using an appropriate watermark extraction or detection algorithm.

In MATLAB (or in RStudio), we first have to find the optimal number of singular values that we need in order to have a compressed image with the essential information. After that, we can see the error in values (numbers) or even in images. This means that we can take an image with the features that we removed from our original

image during the compression process. R programming language is not convenient to work with matrices or with method like SVD due to its statistical orientation. However, we will leverage it to assist users in understanding the intricacies of image compression.

On other hand, MATLAB can be used for the digital watermarking of a given image by implementing methods such as SVD and DWT (discrete Wavelet Transform).

2. Material and method

Singular Values Decomposition (SVD) is a numerical technique used to diagonalize matrices in numerical analysis [1]. SVD aims to approximate the dataset of large number of dimensions using fewer dimensions. SVD considers a highly variable, high dimensional data points and exposes the substructure of the original data by reducing the higher dimensional data into lower dimensional data. Exposure of the substructure orders the data from most variation to the least. This helps to find the region of most variation and then later SVD can be used for reduction. The Singular Value Decomposition of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformation.

SVD method can be used for:

i. Image compression
ii. Data compression and dimensionality reduction
iii. Signal processing
iv. Principal Component Analysis
v. Solving linear systems of equations
vi. Economics and Finance
vii. Facial Recognition and Image Recognition

2.1. Mathematics behind SVD:

The SVD of mxn matrix A is given by the Equation (1):

$$A = U\Sigma V^T \qquad (1)$$

where the matrix

- U is a mxm matrix of the orthonormal eigenvectors of AA^T ;
- Σ is a diagonal matrix with r elements equal to the root of the positive eigenvalues of AA^T or A^TA (both matrices have the same positive eigenvalues anyway)
- V^T is transpose of a nxn matrix containing the orthonormal eigenvectors of A^TA.

So, if we want to find the SVD for a given matrix we have to follow those steps:

a) Finding the eigenvalues of AA^T
b) Finding the right singular vectors i.e orthonormal set of eigenvectors of A^TA.
c) Calculating the U matrix as $U_i = \frac{AV_i}{\Sigma_{ii}}$

2.2. Image compression

The primary aim of image compression is to reduce the size of the image files for efficient storage and transmission while maintaining an acceptable level of image quality [15]. Types of Image Compression are: Lossless and Lossy.

Lossless compression is a technique that reduces the size of a file without any loss of information.

Lossy compression is a technique that reduces the size of a file by losing information.

So, the difference between lossy and lossless is that in lossless compression after compression and decompression the data is identical to the original (resulting in perfect reconstruction). In contrast, in lossy compression, the reconstructed data after decompression is an approximation of the original.

We can use image compression for:

i. Web Design and Development: image compression is utilized to optimize image for web pages to improve loading time and enhance user experience.
ii. Mobile Applications: image compression is utilized to reduce storage space and accelerate loading time.
iii. E-commerce Platforms: it is used to increase page load speed.
iv. Social Media Platforms: we use it to facilitate faster image sharing
v. Document Scanning: we use it for efficient storage and transmission in document management systems
vi. Art and Design: to reduce the file size without compromising visual quality

vii. Video Compression
viii. Gaming Industry: it is applied to optimize storage and enhance real-time rendering performance

Our main goal is to determine the minimum number of singular values needed to ensure that our compressed image has a smaller size than the original. To do this we will use the SVD method. The steps we need to follow to compress an image using SVD method are shown in the Figure 1.

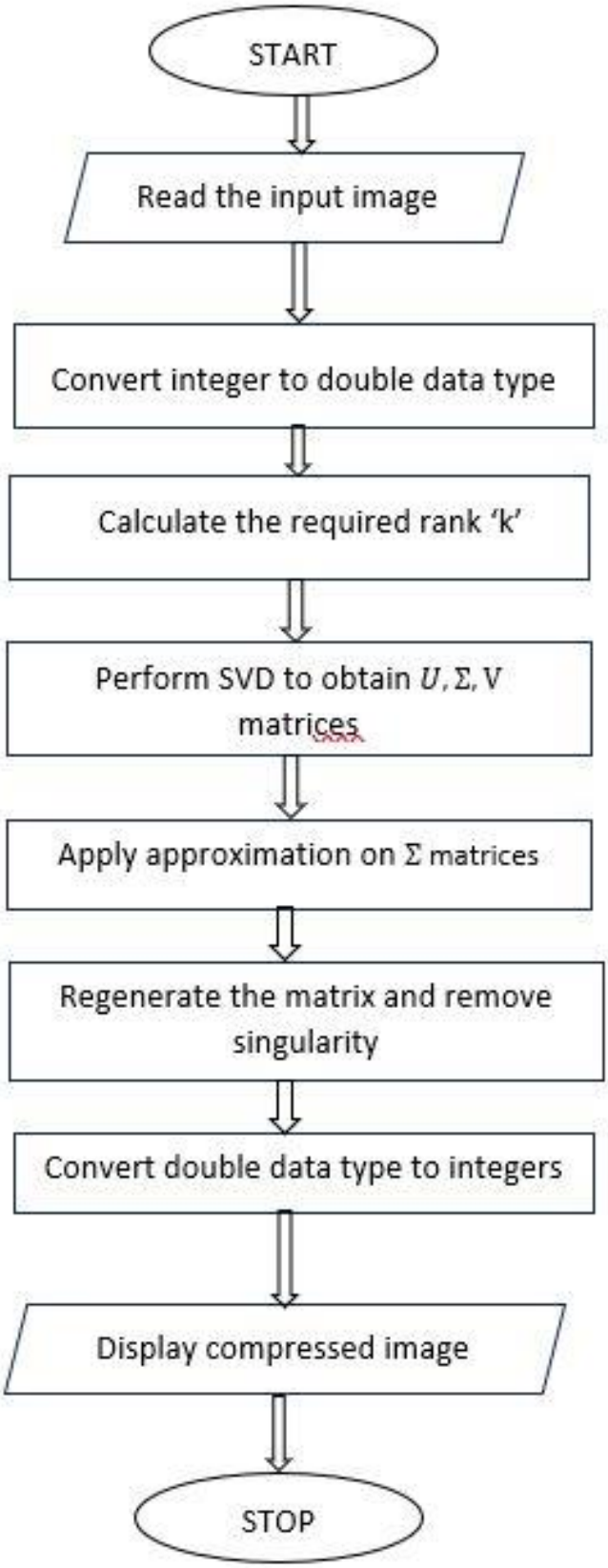

Figure 1. A Scheme on how to compress an image using the SVD method.

We have implemented those steps on MATLAB and R programming languages. In MATLAB singular value decomposition method is implemented. The function is svd(). We know that MATLAB stands for MATrix LABoratory. It is recognized as a powerful programming language and environment that is particularly well-suited for numerical and matrix computations. One of MATLAB's strengths is its native support for matrices, making it an excellent choice for various scientific and engineering applications, including image processing. In MATLAB, an image is typically represented as a 2D matrix (for grayscale images) or a 3D matrix (for color images). For grayscale images, each element of the matrix corresponds to the intensity of a pixel, and for color images each element of the 3D matrix represents the intensity of a color channel (Red, Green, Blue).

We can also use the R programming language for image compression by creating a Shinny app where the user can import the image they want and choose the number of singular values that they desire. After that, the program can display the compressed image. However, image compression in R may not be as efficient as in other programming languages like MATLAB. This is because of its primary focus on statistical computing.

The metrics of Singular Value Decomposition are "Peak Signal-to-Noise Ratio (PSNR)" and "Compression Ratio".

The PSNR is used to measure the quality of an image after compression and it expressed in decibels and provides an indication of how well the compressed image preserves the quality of the original image. It is calculated by comparing the peak signal strength (maximum possible pixel value) to the noise introduced during compression. Higher PSNR value indicates better image quality.

Compression ratio is a significant metric in image compression, indicating the degree to which an image has been reduced in size through compression. It is defined as the ratio of the size of the original image to the size of the compressed image.

We have used Figure 2 as the original image, so this is the image that we want to compress. We have tried using different numbers of singular values. In Figure 3, we can see the compressed image where 11 singular values are used. The compression ratio and PSNR value when 11 singular values are used are 540.5542 and 10.4012, respectively. Figure 4 shows us a part of the first image that has been removed in order to compress the image using 11 singular values. Figure 5 helps us determine the suitable number of singular values needed to ensure that our compressed image has a smaller size than the original without losing important information. We can see that if we use more than 11 singular values, there is insignificant change in the value of the compression ratio. So, the ideal number of singular values that we can use is 11.

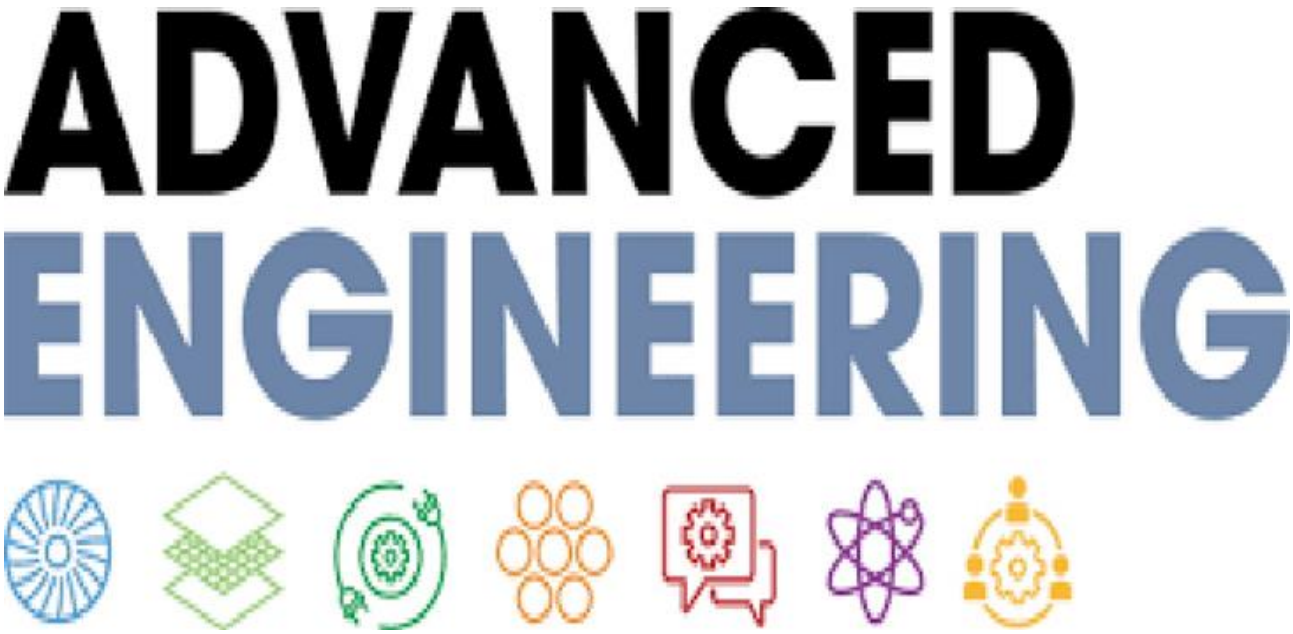

Figure 2. The original image.

After importing Figure 2 into MATLAB, we apply the SVD method, resulting in the Figure 3. The new figure size is smaller than the original, enabling us to save the compressed image and occupy less memory space. This not only reduces memory space but also facilitates faster transmission.

When compressing a given image, we need to remove a portion of it. The Figure 4 illustrates the segment of the original image that has been omitted for compression purposes.

We can perform compression using different numbers of singular values. Each choice of singular values results in a different level of error. One of the metrics we can use to measure this error is compression ratio (Figure 5).

For individuals unfamiliar with image compression and seeking to understand its functionality, they can use this shiny app (Figure 6).

After using 24 singular values, we can now use 12 singular values to observe the difference (Figure 7).

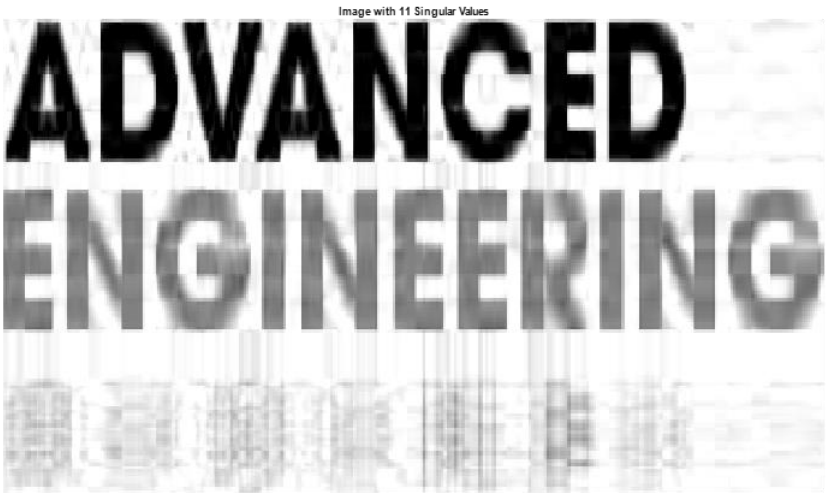

Figure 3. Image compression using 11 singular values.

Figure 4. The error or the part of part of the image that we "removed".

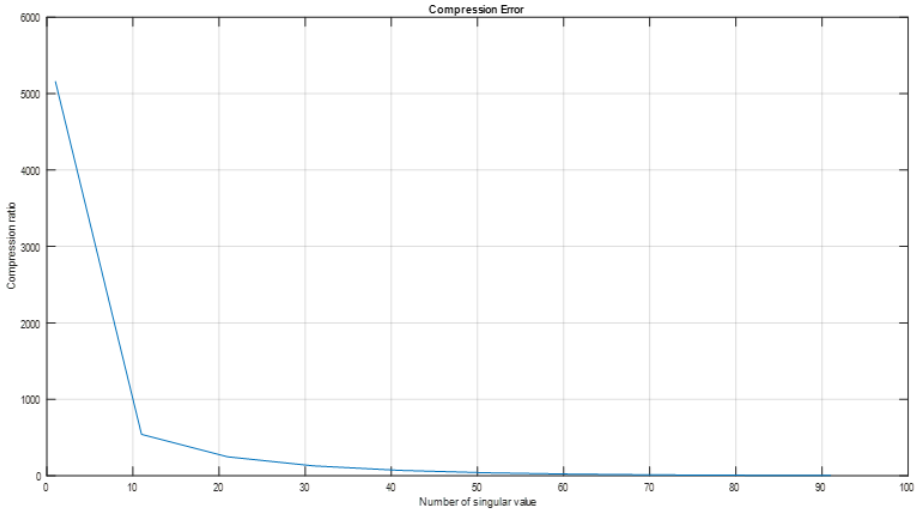

Figure 5. The relationship between the number of singular value and compression ratio.

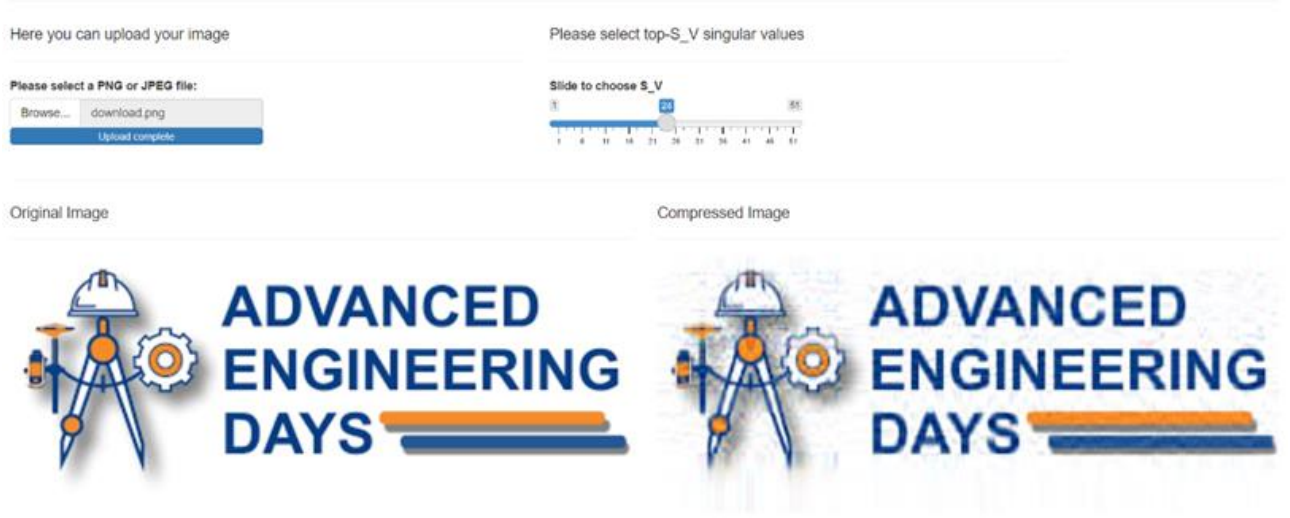

Figure 6. Compressed image using 24 singular values.

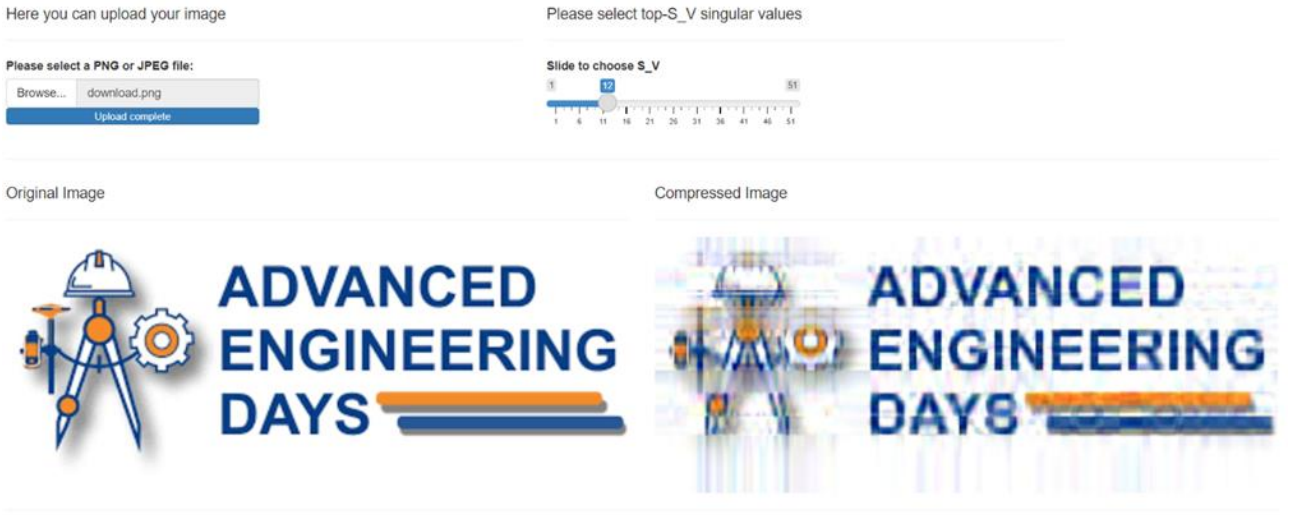

Figure 7. Compressed image using 12 singular values

In Figures 6 and 7, you can observe a shiny app, an interactive platform for users. Within this app, you have the option to select an image from your browser. Afterward, you can choose the number of singular values you desire, and the compressed image will be displayed.

It's worth noting that while R programming may not be the most commonly associated tool for methods like Singular Value Decomposition (SVD) and image compression., this shinny app exemplifies how R can serve as a versatile and user-friendly environment for such tasks. This app allows the users to explore and comprehend the effects of different compression parameters with ease.

2.3. Digital watermarking

In the context of digital watermarking, SVD can be utilized as a method to embed and extract watermark information in digital media. The general steps involved in digital watermarking are (Figure 8):

i. **Choose a watermark**: decide what type of watermark you want to embed. It may be a text message, a logo, or a pattern.

ii. **Preprocess the original image**: Before embedding the watermark, preprocess the original image is necessary (you can resize, normalize or color space conversion).

iii. **Embed the watermark:** choose the embedding algorithm. This algorithm typically modifies pixel values or transforms the image in a way that is not easily perceptible. We have used the Discrete Wavelet Transform (DWT), implemented in MATLAB with the function "dwt2()". DWT is part of Frequency Domain Techniques, embedding the watermark in the coefficients of wavelet-transformed image components. Another technique, which is part of Frequency Domain Techniques, is Discrete Fourier Transform (DFT). Other techniques, such as Spread Spectrum Techniques and Least Significant Bit (LSB) Embedding, are part of the Spatial Domain Techniques [14].

iv. **Postprocess the watermarked image:** to enhance its quality or make it suitable for specific applications, we way need to postprocess the watermarked image.
v. **Add security measures:** it may be used or not. If added, it can make encryption or scrambling techniques more difficult.
vi. **Save or transmit the watermarked image:** it saves the watermarked image in a suitable format.
vii. **Extract the watermark:** it is optional.
viii. **Verify and assess quality:** Verify the effectiveness of the watermarking by extracting the watermark and comparing it with the original. Assess the quality of the watermarked image and evaluate the robustness of the watermark against common attacks.
ix. **Documentation and reporting**
x. **Deployment and integration**

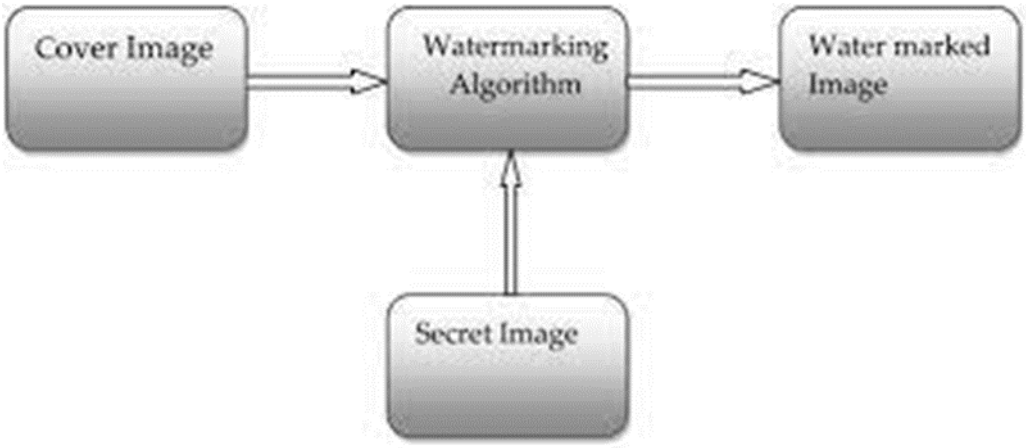

Figure 8. The flow chart represents the digital watermarking process.

In Figure 9, we can see the image in which we want to insert the watermark, Figure 10 is our watermark. In Figure 11 is the watermarked image. There we have decided that 90% of our watermarked image should be involved in the original image.

Figure 9. Original image.

After we choose the image that we want to add the watermark we have to choose our watermarked. In the image where we want to add the watermark firstly, we have implemented the SVD method.

Figure 10. Original watermarked.

After we choose the watermarked, we have decided to implement the Discrete Wavelet Transform (DWT).

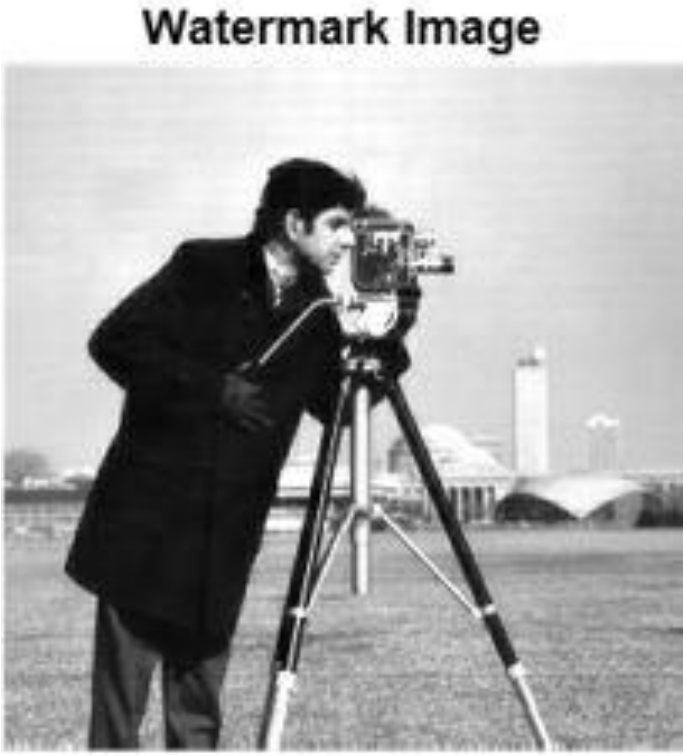

Figure 11. Watermark an image to another image.

For image compression, we previously used PSNR. It also serves as a measure for digital watermarking. What we have observed is that when we decide to involve a high percentage of the watermarked image in the original image, the PSNR is smaller. And this is what we expect because we know that if we add more elements of the watermarked image to the original one, it will change more of the original image.

3. Results

In Table 1, we present the image compression metrics obtained using SVD in the MATLAB programming language. For varying numbers of singular values, we observe different values of PSNR and Compression Ratio.

Table 1. PSNR and compression ratio for different number of singular values.

Number of singular Values	PSNR	Compression ratio
11	10.4012	37.6809
31	13.5235	13.3706
51	16.1671	8.1272
71	18.8835	5.8379
91	21.9250	4.5548

We see that if we increase the number of singular values the value of PSNR will increase and the value of compression ratio will decrease. Our duty is to see where the increase of the PSNR value is insignificant and similarly where the decrease of the Compression Ratio is insignificant.

In Table 2, we present the digital watermarking metrics obtained using SVD method and DWT algorithm in the MATLAB programming language. For varying numbers of singular values, we observe different values of PSNR and MSE.

Table 2. PSNR and MSE value in watermarked image.

Percentage of watermarked image	PSNR	MSE
10%	33.9053	26.4574
30%	24.3629	238.1165
50%	19.9259	661.4347
70%	17.0034	129.64
90%	14.8205	214.30

We can observe that when we add a low percentage of the watermarked image to the original one, the watermarked image closely resembles the original. Therefore, there is an insignificant change between the original image and the watermarked image. In other words, the watermark is less perceptible from the viewer if we use a low percentage of the watermarked image.

4. Conclusion

The SVD method has many different applications. We have decided to show you two of them: image compression and digital image watermarking. This is because we can use image compression to perform digital watermarking. First, we can compress the image, using the SVD technique to reduce its dimensionality while preserving essential information, and then we can use this compression as a "watermark". This dual application highlights the versatility of the SVD method in both enhancing data storage efficiency through compression and augmenting data security and authenticity through digital watermarking.

When we use image compression and digital watermarking our responsibility is to identify where our original image loses its main information or where the point of minimal error occurs. To find this point we can use different measures. The mostly used measure is the PSNR (Pick Signal to Noise Ratio). The higher the PSNR value, the better the quality of the image reconstruction.

In image compression, when we increase the number of singular values, the PSNR value will also increase. Additionally, where there the PSNR increases, the compression ratio, which is another important measure, will decrease. In this case, the compressed image is closely resembling the original one.

In digital watermarking, we have to decide how perceptible the watermarked we want to embed should be to the viewer. When we add a low percentage, our original image and the watermarked image are similar to each other. In this case, the mean square error will be smaller. This is what we expected because we have added a small piece of our watermarked to the original image.

MATLAB programming language that is convenient to work with matrices, as every object in MATLAB is treated as a matrix. Singular Value Decomposition is implemented in MATLAB using “svd()” function. Additionally, MATLAB also implements the Discrete Wavelet Transform (DWT) through the “dwt2()” function.

Acknowledgement

This study was partly presented at the 8th Advanced Engineering Days [16].

Funding

This research received no external funding.

Author contributions

Ornela Gordani Visualization, Software, Writing-Reviewing and Editing. **Aurora Simoni:** Conceptualization, Methodology, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. Martin, C. D., & Porter, M. A. (2012). The extraordinary SVD. The American Mathematical Monthly, 119(10), 838-851. https://doi.org/10.4169/amer.math.monthly.119.10.838
2. Trefethen, L. N. (1992). The definition of numerical analysis. Cornell University.
3. Baksalary, O. M., & Trenkler, G. (2021). The Moore–Penrose inverse: a hundred years on a frontline of physics research. The European Physical Journal H, 46, 1-10.
4. Comon, P. (2002). Tensor Decompositions, State of the Art and Applications. Mathematics in Signal Processing. Oxford University Press.
5. De Lathauwer, L., & De Moor, B. (1998). From matrix to tensor: Multilinear algebra and signal processing. Institute of Mathematics and its Applications Conference Series, 67, 1-16. Oxford University Press.
6. Sidiropoulos, N. D., Bro, R., & Giannakis, G. B. (2000). Parallel factor analysis in sensor array processing. IEEE Transactions on Signal Processing, 48(8), 2377-2388. https://doi.org/10.1109/78.852018
7. Vasilescu, M. A. O., & Terzopoulos, D. (2002). Multilinear analysis of image ensembles: Tensorfaces. In Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7, 447-460. https://doi.org/10.1007/3-540-47969-4_30
8. Vasilescu, M. A. O., & Terzopoulos, D. (2002). Multilinear image analysis for facial recognition. 2002 International Conference on Pattern Recognition, 2, 511-514. https://doi.org/10.1109/ICPR.2002.1048350
9. Vasilescu, M. A. O., & Terzopoulos, D. (2003). Multilinear subspace analysis of image ensembles. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings, 2, 93-99. https://doi.org/10.1109/CVPR.2003.1211457
10. Savas, B., & Eldén, L. (2007). Handwritten digit classification using higher order singular value decomposition. Pattern Recognition, 40(3), 993-1003. https://doi.org/10.1016/j.patcog.2006.08.004
11. Kolda, T. G., Bader, B. W., & Kenny, J. P. (2005). Higher-order web link analysis using multilinear algebra. Fifth IEEE International Conference on Data Mining (ICDM'05). https://doi.org/10.1109/ICDM.2005.77
12. Mucha, P. J., Richardson, T., Macon, K., Porter, M. A., & Onnela, J. P. (2010). Community structure in time-dependent, multiscale, and multiplex networks. Science, 328(5980), 876-878. https://doi.org/10.1126/science.1184819
13. Elakkiya, S., & Thivya, K. S. (2022). Comprehensive review on lossy and lossless compression techniques. Journal of The Institution of Engineers (India): Series B, 103(3), 1003-1012. https://doi.org/10.1007/s40031-021-00686-3
14. Chahal, P. K., Singh, A., & Singh, P. (2013). Digital Watermarking Techniques. International Journal of Computers &Amp; Technology, 11(8), 2903–2909. https://doi.org/10.24297/ijct.v11i8.3009
15. Tan, L., Zeng, Y., & Zhang, W. (2019). Research on image compression coding technology. Journal of Physics: Conference Series, 1284(1), 012069. https://doi.org/10.1088/1742-6596/1284/1/012069
16. Gordani, O., & Simoni, A. (2023). The application of SVD method in image compression and digital watermarking. Advanced Engineering Days (AED), 8, 100-102.