SVD Based Image Processing Applications

Siju K S, *Member, IEEE* and John Doe

***Abstract*—This study investigates the application of Singular Value Decomposition (SVD) as an effective mathematical framework for various image processing tasks. SVD offers a unique decomposition approach, making it suitable for applications like image compression, denoising, and watermarking by enabling optimal rank approximations and noise separation. The robustness of SVD in handling large matrices allows it to capture key image characteristics, preserving essential features while reducing data requirements. By leveraging SVD's ability to separate data into dominant and subdominant subspaces, this research demonstrates enhanced image compression, effective noise reduction, and secure watermark embedding. Experimental results validate SVD's utility in optimizing image storage, clarity, and fidelity, with potential implications for advancing adaptive image processing techniques.**

***Index Terms*—Singular Value Decomposition (SVD), Image Processing, Image Compression, Image Denoising, Digital Watermarking, Noise Filtering, Matrix Factorization, Rank Approximation, Frobenius Norm, Energy Compaction, Digital Forensics, Signal Processing, Adaptive Image Processing, Orthogonal Subspaces**

I. Introduction

Image processing has become integral to numerous fields, from medical imaging to digital forensics, where large volumes of visual data demand efficient storage, transmission, and quality retention techniques. Among the many mathematical transformations applied to images, Singular Value Decomposition (SVD) has emerged as a particularly valuable tool. SVD is a matrix factorization technique that represents a given matrix as a product of three matrices: U, Σ, and V^T. This decomposition is significant in image processing because it maximizes the energy contained in the largest singular values, enabling the creation of compact, high-quality approximations of the original data. Unlike other transformations, SVD does not require a specific image size or type, making it highly adaptable and robust for various image processing tasks.

The primary strength of SVD lies in its capacity to separate image data into meaningful components. For instance, in an image represented by SVD, the larger singular values and their corresponding vectors encode most of the structural content, while smaller singular values can often represent noise. This property is beneficial for applications requiring data reduction, such as image compression and denoising, where maintaining the primary structure while reducing extraneous information is essential. Additionally, SVD's stable mathematical foundation and adaptability have made it increasingly popular in other specialized applications, including watermarking for digital forensics and security.

In image compression, SVD enables reduced data storage by approximating the image using fewer singular values, providing a balance between quality and compression ratio. This application is critical in fields where storage and bandwidth are constrained. Similarly, in denoising, SVD can isolate noise by exploiting the decomposition's ability to differentiate between dominant and subdominant subspaces, allowing effective noise suppression without significantly affecting the image's core structure. Furthermore, SVD is also used in watermarking, where slight modifications to specific singular values embed unique patterns within images, enhancing security and ensuring authenticity.

Despite these advantages, SVD in image processing remains an area with unexplored potential. This paper explores these established applications while addressing underutilized SVD properties to uncover new applications. By investigating SVD's adaptive properties in compressing and filtering images, as well as its potential for encoding data securely, this work contributes to a growing body of research on SVD-based image processing and presents promising directions for further study.

II. SVD Application in Image Processing

Singular Value Decomposition (SVD) has several important applications in image processing. The SVD can be used to reduce the noise or compress matrix data by eliminating small singular values or higher ranks [1]. This allows for the size of stored images to be reduced [2]. Additionally, the SVD has properties that make it useful for various image processing tasks, such as enhancing image quality and filtering out noise. The main theorem of SVD is reviewed in the search results, and numerical experiments have been conducted to illustrate its applications in image processing.

A. Image Compression

Image compression represents a vital technique to reduce the data needed to represent an image. This is crucial for achieving efficient storage and transmission across various applications, including digital photography, video streaming, and web graphics. Compression methods are primarily categorized into two distinct types: lossy and lossless.

Lossy compression diminishes file size by irreversibly eliminating certain image data, which can result in a degradation of image quality, as observed in JPEG formats. This method is frequently employed when the reduction of file size is of paramount importance, and any resultant loss in quality is considered acceptable.

The `quarto-ieee` template is freely available under the MIT license on github: https://github.com/dfolio/quarto-ieee.

Siju K S is with School of Artificial Intelligence, Amrita Vishwa Vidyapeetham, Coimbatore, 18800 India Corresponding author: siju.swamy@saintgits.org

Unknown affiliation

John Doe is with Anonymous University

Template created June 23, 2023; revised `r format(Sys.Date(),format='%B %d, %Y')`.

Conversely, lossless compression techniques allow for the compression of images without any loss of data, facilitating the exact reconstruction of the original image, as exemplified by PNG formats. This approach is beneficial when preserving image quality is essential and minimizing file size is of lesser importance.

The decision to use either lossy or lossless compression hinges on the specific needs of the application, balancing the trade-offs between file size and image quality.

SVD-based image compression functions by decomposing the image matrix into three components and subsequently approximating the original matrix with only the most significant singular values and vectors. This process results in a compact image representation while preserving the essential information.

Mathematically, given an image represented as a matrix A with dimensions $m \times n$, the Singular Value Decomposition (SVD) decomposes A into three matrices: U, Σ, and V^T. Here, U is an $m \times m$ orthogonal matrix containing the left singular vectors, Σ is an $m \times n$ diagonal matrix containing singular values, and V^T is the transpose of an $n \times n$ orthogonal matrix containing the right singular vectors. To compress the image, we keep only the top k singular values (where k is significantly smaller than both m and n). The compressed image can be reconstructed as

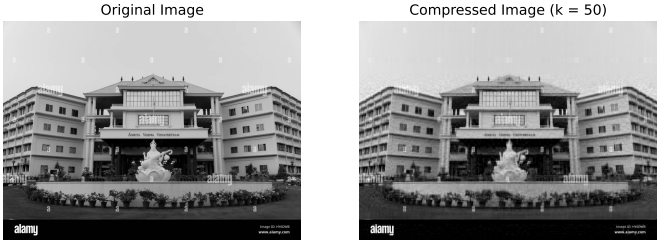
$$A_k = U_k \Sigma_k V_k^T,$$

where U_k contains the first k columns of U, Σ_k is a $k \times k$ diagonal matrix of the top k singular values, and V_k^T consists of the first k rows of V^T.

B. Code

```
% Read and convert the image to grayscale
img = imread('amrita_campus.jpg'); % Specify your image file
gray_img = rgb2gray(img); % Convert to grayscale
A = double(gray_img); % Convert to double for SVD computation

% Apply Singular Value Decomposition (SVD)
[U, S, V] = svd(A)

% Choose the number of singular values to ke
k = 50; % You can adjust this value to see d

% Create a compressed version of the image u
S_k = zeros(size(A)); % Initialize a zero ma
S_k(1:k, 1:k) = S(1:k, 1:k); % Keep only the

% Reconstruct the compressed image
A_k = U*S_k*V'; % Reconstruct the image fron

% Display the original and compressed images
figure;
subplot(1, 2, 1);
imshow(uint8(A)); % Display original image
title('Original Image');
subplot(1, 2, 2);
imshow(uint8(A_k)); % Display compressed image
title(['Compressed Image (k = ', num2str(k), ')'])
```

C. Output



To assess the quality of the original and compressed images, various metrics can be employed. Two common measures are Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE).

The Mean Squared Error quantifies the average squared difference between pixel values of the original and compressed images. It is defined as:

$$\text{MSE} = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (A(i,j) - A_k(i,j))^2$$

where m and n are the dimensions of the image, $A(i,j)$ is the pixel value of the original image, and $A_k(i,j)$ is the pixel value of the compressed image.

The Peak Signal-to-Noise Ratio (PSNR) is a measure that compares the maximum possible power of a signal to the power of corrupting noise that affects the fidelity of its representation. It is given by:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{M_{\text{AX}}^2}{\text{MSE}} \right)$$

where M_{AX} is the maximum pixel value, typically 255 for an 8-bit image.

Below is the additional MATLAB code to calculate MSE and PSNR:

```
% Calculate Mean Squared Error (MSE)
mse = mean((A(:) - A_k(:)).^2);

% Calculate Peak Signal-to-Noise Ratio (PSNR)
max_pixel_value = 255; % Maximum pixel value for 8
psnr = 10 * log10((max_pixel_value^2) / mse);

% Calculate sizes
original_size = numel(A) * 8; % Size of the origina
compressed_size = (k * (size(A, 1) + size(A, 2)))

% Display results
fprintf('Mean Squared Error (MSE): %.4f\n', mse);
fprintf('Peak Signal-to-Noise Ratio (PSNR): %.4f d
fprintf('Original Image Size: %.2f KB\n', original_
fprintf('Compressed Image Size: %.2f KB\n', compre
fprintf('Size Reduction: %.2f KB\n', (original_siz
```

1) Code:

2) Output:

```
Mean Squared Error (MSE): 110.2853
Peak Signal-to-Noise Ratio (PSNR): 27.7056 dB
Original Image Size: 9709.38 KB
Compressed Image Size: 900.78 KB
Size Reduction: 8808.59 KB
```

[1] Z. Chen, "Singular value decomposition and its applications in image processing," *Proceedings of the 2018 1st International Conference on Mathematics and Statistics*, 2018 [Online]. Available: https://api.semanticscholar.org/CorpusID:53245257

[2] L. Cao, "Singular value decomposition applied to digital image processing," *Division of Computing Studies, Arizona State University Polytechnic Campus, Mesa, Arizona State University polytechnic Campus*, pp. 1–15, 2006.

Siju K S Use `IEEEbiography` with figure as option and the author name as the argument followed by the biography text.

John Doe Use `IEEEbiographynophoto` and the author name as the argument followed by the biography text.