Assignment Report

Software Assignment using SVD decompositions

Mahesh Chollangi - EE25BTECH11017

I. Introduction

The goal of this software project is to implement **image compression using Singular Value Decomposition (SVD)** in the **C programming language**. Image compression is an important technique in digital image processing that reduces the amount of data required to represent an image while maintaining acceptable visual quality. SVD is a powerful linear algebra method that decomposes an image matrix into its fundamental components — singular values and vectors — which represent the major patterns and features of the image. By keeping only the top k singular values, we can reconstruct an approximate version of the original image with much less data, achieving efficient compression. This project implements the SVD process using the **power iteration algorithm** and **deflation method**, converts images to grayscale using the **STB image library**, and saves the compressed output as a JPEG file. The objective is to understand both the mathematical concept of SVD and its practical application in reducing image size without significant loss of quality.

II. Summary of Strang's Video

Gilbert Strangâs MIT lecture on Singular Value Decomposition (SVD) explains that every matrix can be decomposed as

$$A = U\Sigma V^{\top},$$

where U and V are orthogonal matrices and Σ contains the singular values (square roots of eigenvalues of $A^{\top}A$). Strang shows that SVD reveals the geometric meaning of a matrix: it rotates and stretches space. The largest singular values correspond to the most important directions of variation in the data or image. In image compression, keeping only a few of these values captures most of the visual information while discarding small singular values that represent fine details and noise.

III. Types of Algorithms for SVD (in Image Compression)

III.1 1. Power Iteration Method

Method:

It repeatedly multiplies a random vector by the matrix and normalizes it until the vector aligns with the direction of maximum energy - the dominant singular vector.

Pros:

- Very easy to implement.
- Works even for large images.
- Good if we only need top few singular values (like image compression).

Cons:

- Converges slowly for closely spaced singular values.
- Needs multiple restarts for multiple singular vectors.

III.2 2. Jacobi Method (Jacobi SVD)

Method:

Applies a series of orthogonal rotations to make the off-diagonal elements of

$$A^{T}A$$

zero, turning it into a diagonal matrix - that diagonal gives the squared singular values.

Pros:

- High numerical accuracy.
- Works well for small and medium matrices.

Cons:

- Computationally expensive for large images.
- Not used much in real-time compression.

III.3 3. QR Decomposition Method

Method:

Two main steps:

- Bidaigonalization of A using Householder reflections.
- Apply QR iterations on the bidaigonal matrix until it becomes diagonal - singular values appear on the diagonal.

.

Pros:

- Very stable and accurate.
- Works for dense, general matrices.

Cons:

- More complex and slower for large images.
- Needs more memory.

III.4 4. Lanczos Bidiagonalization Method

Method:

Projects A into a smaller bidiagonal matrix using the Lanczos process, then computes SVD of the small matrix.

Pros:

- Much faster for big images or sparse data.
- Good accuracy for top-k SVD.

Cons:

- Needs careful implementation to maintain numerical stability.

IV. Implemented Algorithm (Why i choose Power Iterated Algorithm)

Criterion	Explanation
Simplicity	Power iteration is mathematically simple and easy to implement using basic matrixâvector operations (Av and $A^T u$).
Memory Efficiency	It doesnât require storing large intermediate matrices; ideal for image data where the matrix size can be very large.
Speed for Top-k SVD	It focuses only on the largest few singular values (k values), which are enough for compression, instead of computing the full SVD.
Numerical Stability	By normalizing vectors in each iteration, it maintains numerical stability.

TABLE I: Reasons for choosing the Power Iteration Algorithm for Image Compression

V. Pseudocode for Implemented Algorithm

```
SVD-Based Image Compression using Power Iteration
  Input: Input image file infile, Output file outfile, Compression rank $k$
  Output: Compressed grayscale image and Frobenius error
  1. Read and Preprocess:
       a) Read image file from disk.
       b) Decode it into a grayscale matrix $A \in \mathbb{R}^{h\times w}$.
       c) Create a working copy $M \leftarrow A$.

  2. Initialize Variables:
       Allocate arrays $U[k][h]$, $V[k][w]$, and $S[k]$ for top-$k$ singular vectors and values.

  3. For each rank $t = 1$ to $k$:
       a) Compute the top singular triplet $(u, v, s)$ using Power Iteration (see below).
       b) Store $U[t] \leftarrow u$, $V[t] \leftarrow v$, $S[t] \leftarrow s$.
       c) Update matrix $M$ by removing the contribution of this singular triplet:
            $M[i, j] \leftarrow M[i, j] - s \cdot u[i] \cdot v[j]$, for all $i, j$.

  4. Reconstruction of Image:
       Initialize reconstructed matrix $R$ as zeros.
       For each $t = 1$ to $k$:
            $R[i, j] \leftarrow R[i, j] + S[t] \cdot U[t][i] \cdot V[t][j]$.

  5. Compute Frobenius Error:
       Compute the error and relative error as:
```

$$||A - R||_F = \sqrt{\sum_{i,j} (A[i, j] - R[i, j])^2}, \quad \text{Relative Error} = \frac{||A - R||_F}{||A||_F}.$$

```
  6. Post-processing and Saving:
       a) Clamp all values of $R$ to range [0, 255].
       b) Convert to unsigned 8-bit integers.
       c) Save as a compressed grayscale JPEG image with quality = 90.
```

Power Iteration Subroutine:

```
1. Initialize $v$ as a vector of ones.
2. Repeat 15 iterations:
    a) $u \leftarrow \frac{Av}{||Av||_2}$
    b) $v \leftarrow \frac{A^\top u}{||A^\top u||_2}$
3. Compute $s \leftarrow ||Av||_2$.
4. Return $(u, v, s)$ as the top singular triplet.
```

VI. Implementation

Let us take an example image for implementation
If we take an image "einstein.jpg" and set k=20 , we get the following results through the code

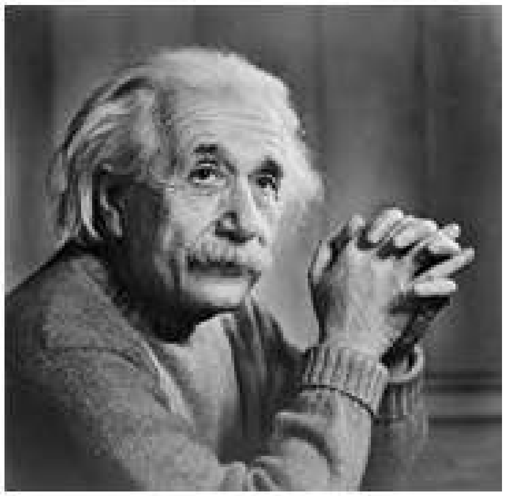
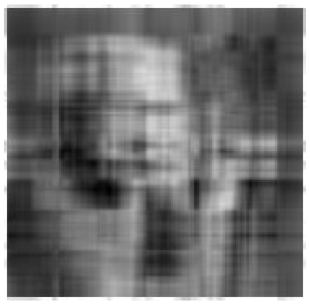
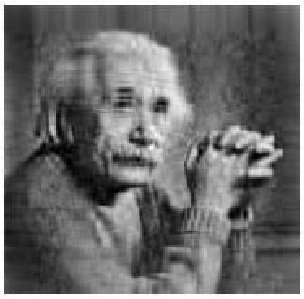
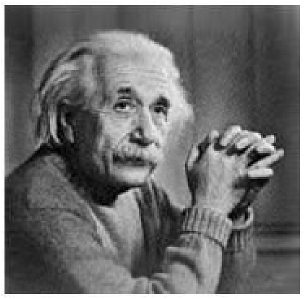
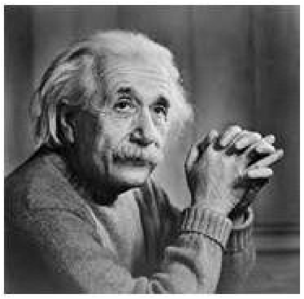


Fig. 1: Original


(a) k=5


(b) k=20


(c) k=50


(d) k=100

Similarly , we will get the results for remaining images with assigned k value of our choice .

VII. Error Analysis

To measure reconstruction quality, we compute the **Frobenius Error (FE)** and **Relative Error (RE)**:

TABLE II: Frobineous Error and Relative Error values for different images at various k values

k	Einstein		Globe		Greyscale	
	FE	**RE**	**FE**	**RE**	**FE**	**RE**
5	4177.7986	21.63	20704.2858	13.08	11146.3702	05.76
20	2127.7021	9.76	10635.1206	6.72	8888.7417	1.97
50	889.2415	4.05	6193.8654	3.91	1486.2853	0.60
100	165.2658	0.76	3680.3594	2.32	512.7180	0.25

VIII. Conclusion

Conclusion:

The project successfully implements image compression using Singular Value Decomposition (SVD) based on the Power Iteration algorithm. Through this approach, the image matrix is approximated using only its most significant singular values, reducing file size while maintaining acceptable visual quality. Power Iteration proved to be an efficient and memory-friendly method for estimating dominant singular values without computing the full SVD.

The results confirm that as the rank k increases, the reconstructed image becomes closer to the original, and the Frobenius error decreases. Hence, the method achieves a good balance between compression efficiency and image clarity, demonstrating the effectiveness of SVD-based image compression.