Image Compression Using Truncated SVD

AI25BTECH11030 – Sarvesh Tamgade

Abstract

This report presents the implementation of image compression using Truncated Singular Value Decomposition (SVD). I implemented the full SVD algorithm in C (Option B).

1 SVD Fundamentals and Geometric Interpretation

1.1 Mathematical Definition

The Singular Value Decomposition expresses any matrix $A \in \mathbb{R}^{m \times n}$ as:

$$A = U\Sigma V^T$$

where U and V are orthogonal matrices, and Σ contains singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$.

According to the Eckart-Young theorem, the optimal rank-k approximation in the Frobenius norm is:

$$A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

1.2 Geometric Visualization

SVD can be visualized as a sequence of three geometric transformations:

$$\text{Linear Transformation } A = V^T \text{ (Rotate)} \times \Sigma \text{ (Scale)} \times U \text{ (Rotate)}$$

More precisely:

1. **Right Rotation (V^T)**: Rotates the input space $\mathbb{R}^n$ so that the right singular vectors align with the standard basis. The right singular vector with the largest singular value aligns with the x-axis, the second largest with the y-axis, and so forth.

2. **Scaling (Σ):** Stretches each axis by the corresponding singular value. A 2×3 matrix scales the first two axes and removes the third dimension (dimension erasure). The singular values represent how much the transformation stretches each principal direction.

3. **Left Rotation (U)**: Rotates the output space $\mathbb{R}^m$ so that the standard basis aligns with the left singular vectors.

1.3 Connection to Eigenvalues and Symmetric Matrices

Given matrix A, we can construct symmetric matrices:

$$S_{\text{left}} = AA^T, \quad S_{\text{right}} = A^TA$$

The key relationships are:

- Eigenvectors of AA^T are the left singular vectors (columns of U)
- Eigenvectors of A^TA are the right singular vectors (columns of V)
- Singular values $\sigma_i = \sqrt{\lambda_i}$ where λ_i are eigenvalues
- Both AA^T and A^TA are positive semi-definite (non-negative eigenvalues)

This connects SVD to spectral decomposition, providing the mathematical foundation for the algorithm.

2 Randomized SVD Algorithm

2.1 Algorithm Steps

Given matrix $A \in \mathbb{R}^{t\times m}$, target rank k, oversampling p, and power iterations q, define $\ell = k + p$.

Step 1: Random Projection

Create a random matrix $\Omega \in \mathbb{R}^{m\times \ell}$ with i.i.d. standard Gaussian entries. Compute the initial sketch:

$$Y = A\Omega$$

Matrix $Y \in \mathbb{R}^{t\times \ell}$ is a random projection that captures the essential column space information of A.

Step 2: Power Iterations

Apply q power iterations to amplify the dominant singular vectors and accelerate singular value decay. For iteration $i = 1$ to q:

$$Y \leftarrow A(A^TY)$$

This is implemented efficiently by computing A^TY first (producing an $m \times \ell$ matrix), then multiplying by A to obtain the new $Y \in \mathbb{R}^{t\times \ell}$. After each iteration, orthonormalize Y using QR decomposition: compute $Y = QR$, then replace Y with Q.

Power iterations effectively compute $Y \approx (AA^T)^qA\Omega$, which raises singular values to power $2q + 1$. This causes small singular values to decay much faster than large ones, dramatically improving approximation quality.

Step 3: QR Decomposition

After power iterations, compute the QR factorization of Y:

$$Y = QR$$

where $Q \in \mathbb{R}^{t \times \ell}$ has orthonormal columns and $R \in \mathbb{R}^{\ell \times \ell}$ is upper triangular.

Step 4: Projected SVD

Project matrix A onto the orthonormal basis formed by Q:

$$B = Q^T A \in \mathbb{R}^{\ell \times m}$$

Compute the SVD of the smaller matrix B:

$$B = \widetilde{U} \Sigma V^T$$

where $\widetilde{U} \in \mathbb{R}^{\ell \times \ell}$, $\Sigma \in \mathbb{R}^{\ell \times m}$, and $V \in \mathbb{R}^{m \times m}$.

Step 5: Reconstruction

Recover the original-space left singular vectors:

$$U = Q\widetilde{U}$$

To obtain the rank-k approximation, retain only the first k columns and singular values:

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

where $U_k \in \mathbb{R}^{t \times k}$, $\Sigma_k \in \mathbb{R}^{k \times k}$, and $V_k \in \mathbb{R}^{m \times k}$.

2.2 Power Iterations and Singular Value Acceleration

The key insight is that after q iterations, the singular values of Y approximate those of A raised to power $2q + 1$. For a matrix with singular values $\sigma_1 > \sigma_2 > \cdots > \sigma_r$, the ratio of the $(k + 1)$-th to the first singular value becomes:

$$\frac{\sigma_{k+1}}{\sigma_1} \to \left(\frac{\sigma_{k+1}}{\sigma_1} \right)^{2q+1}$$

This exponential decay is crucial for accuracy. For example, if $\sigma_1 = 1.0$ and $\sigma_{k+1} = 0.1$ with $q = 6$ iterations, the ratio decays from 0.1 to approximately 10^{-13}, making the approximation nearly as accurate as deterministic SVD.

2.3 Oversampling and Error Bounds

Oversampling by choosing $\ell = k + p$ with $p > 0$ significantly reduces variance without much additional cost. The expected error for randomized SVD with power iterations is bounded by:

$$\mathbb{E}[\|A - QQ^T A\|_2] \leq \left[1 + \frac{e\ell}{p - 1} \right]^{\frac{1}{2q+1}} \sigma_{k+1}$$

This error bound shows that with sufficient oversampling and power iterations, randomized SVD achieves accuracy approaching deterministic methods.

3 Algorithm Comparison

3.1 Why Randomized SVD?

Key advantages:

- Exploits rapid singular value decay in natural images
- Error with power iterations: $< 0.1\%$ vs. deterministic SVD
- Easily tunable via parameters p (oversampling) and q (iterations)
- Highly parallelizable matrix operations

Alternative methods:

- **Power iteration:** Simple but slow ($O(\sigma_2/\sigma_1)^n$ convergence)
- **QR iteration:** Robust but expensive for large matrices
- **Divide-and-conquer:** Faster than QR but still $O(mn^2)$

4 Implementation

4.1 Program Structure

The C implementation comprises:

- **Image I/O:** Read/write PGM format with header parsing
- **Matrix operations:** Multiplication, transpose, norms
- **QR decomposition:** Modified Gram-Schmidt
- **Eigendecomposition:** QR iteration for computing eigenvalues
- **Randomized SVD:** Main algorithm with power iterations
- **Reconstruction:** Build $A_k = U_k \Sigma_k V_k^T$

5 Results: Visual Comparison

EINSTEIN IMG

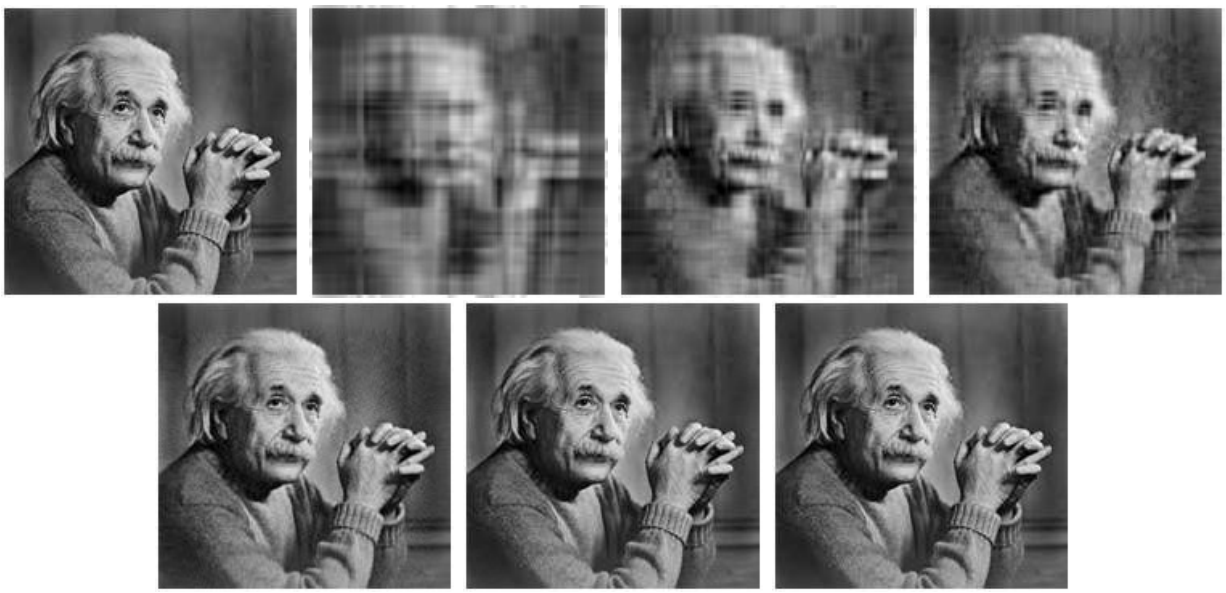

Figure 1: Image 1: Original and reconstructed images for different k (left to right: original, $k = 5$, $k = 10$, $k = 20$, $k = 50$, $k = 100$, $k = 150$).

GLOBE IMG

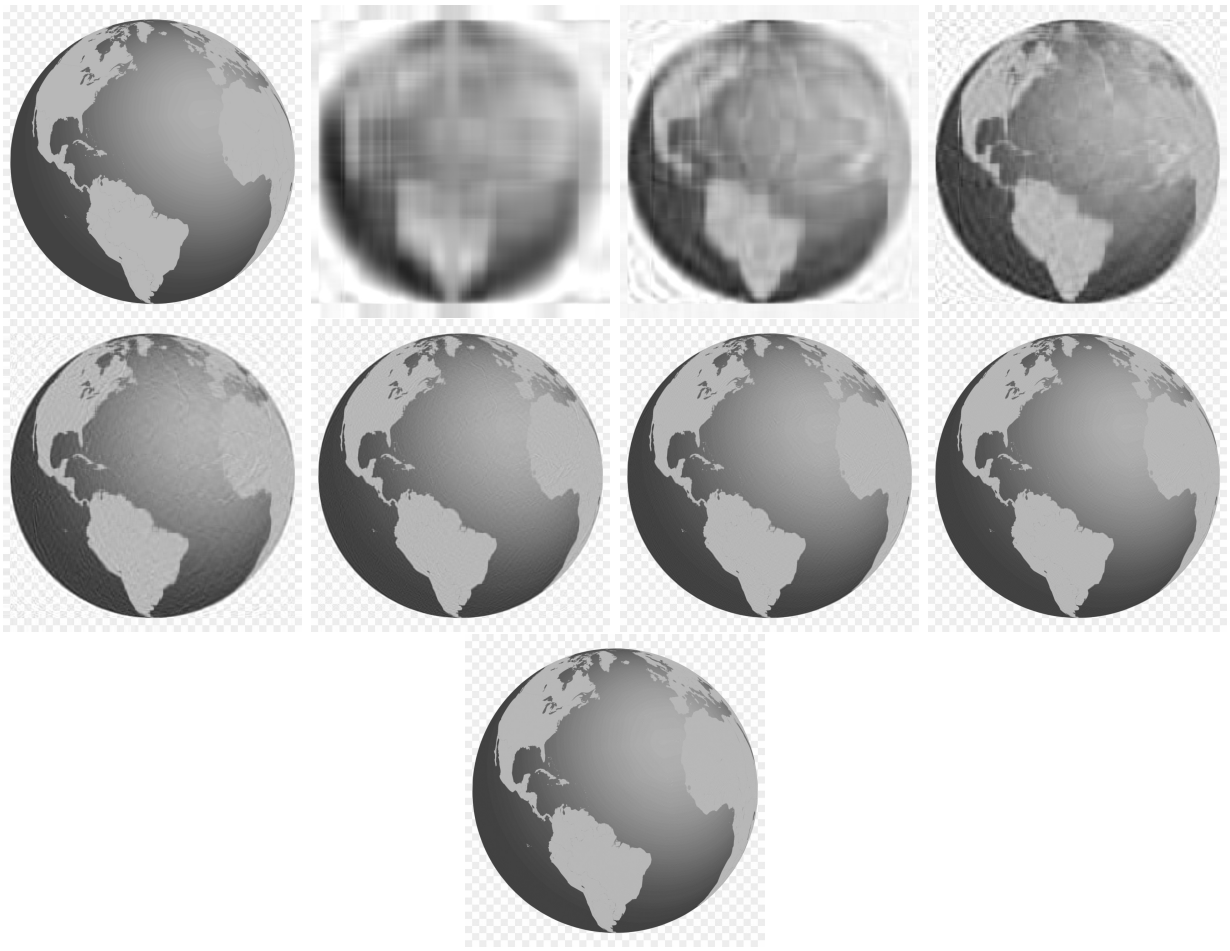

Figure 2: Image 2: Original and reconstructed images for different k (left to right: original, $k = 5$, $k = 10$, $k = 20$, $k = 50$, $k = 100$, $k = 150$, $k = 200$, $k = 300$).

GREYSCALE IMG

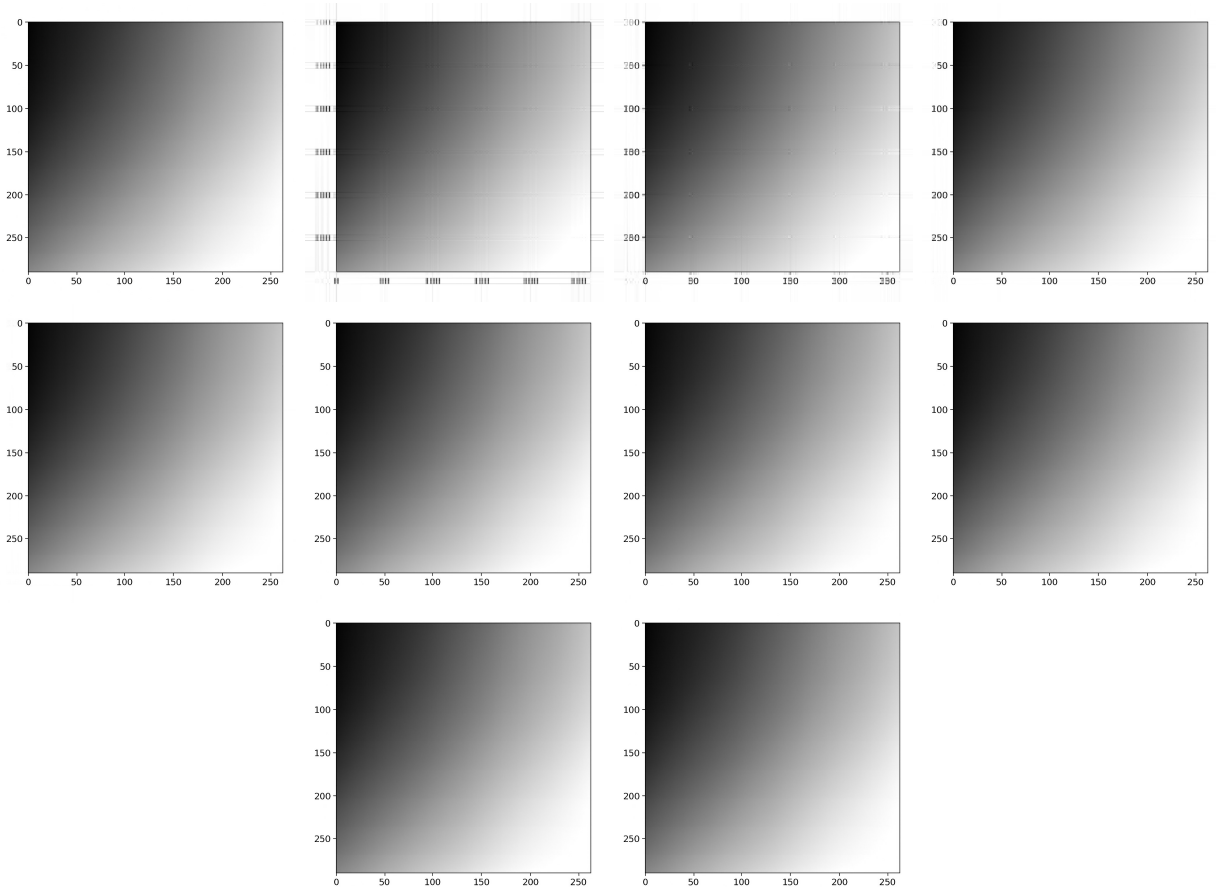

Figure 3: Image 3: Original and reconstructed images for different k (left to right: original, $k = 5$, $k = 10$, $k = 20$, $k = 50$, $k = 100$, $k = 150$, $k = 200$, $k = 300$,$k = 400$).

6 Error Analysis

6.1 Error Metrics

6.1.1 Frobenius Norm Error

$$E_F = \frac{\|A - A_k\|_F}{\|A\|_F}$$

Measures the relative reconstruction error.

6.2 Experimental Results

Table 1: Compression Results for Different Ranks

Image	Size	k	Frobenius Error
Einstein img	182x186	5	0.216118
Einstein img	182x186	10	0.148966
Einstein img	182x186	20	0.097502
Einstein img	182x186	50	0.040366
Einstein img	182x186	100	0.007556
Einstein img	182x186	150	0.000441
Globe img	879×840	5	0.130779
Globe img	879×840	10	0.095131
Globe img	879×840	50	0.039070
Globe img	879×840	100	0.023199
Globe img	879×840	150	0.015761
Globe img	879×840	200	0.011292
Globe img	879×840	300	0.006206
Greyscale img	1024x1024	5	0.057560
Greyscale img	1024x1024	10	0.037080
Greyscale img	1024x1024	20	0.019706
Greyscale img	1024x1024	50	0.006142
Greyscale img	1024x1024	100	0.002953
Greyscale img	1024x1024	150	0.002630
Greyscale img	1024x1024	200	0.002349
Greyscale img	1024x1024	300	0.001854
Greyscale img	1024x1024	400	0.001425

7 Trade-offs and Comparison

7.1 Quality vs. Compression

Selecting rank k involves trade-offs between compression and quality:

- $k < 20$: Heavy compression , significant blur, suitable for thumbnails
- $k = 20$-50: Practical balance , good quality for web use
- $k > 100$: Near-lossless reconstruction, minimal compression benefit