Image Compression using SVD

Introduction

Data compression is an important application of linear algebra. The need to minimize the amount of digital information stored and transmitted is an ever growing concern in the modern world. Singular Value Decomposition is an effective tool for minimizing data storage and data transfer. This report explores image compression and noise reduction through the use of singular value decomposition on image matrices.

The Singular Value Decomposition SVD

The singular value decomposition of a matrix $A \in C^{m\times n}$ is

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

Where $U \in C^{m\times m}$ and $V \in C^{n\times n}$ are orthogonal. $\Sigma \in C^{m\times n}$ is diagonal, and has non-negative diagonal elements ordered in non-increasing order.

A matrix $A \in C^{m\times n}$ maps from R^n to R^m. The SVD decomposition reduces A to the clearest representation of that map. The columns of U and V form orthonormal bases of R^m and R^n respectively. The diagonal elements of Σ are the amplifications of the map.

SVD Calculation Example

$$A = \begin{pmatrix} -2 & 11 \\ -10 & 5 \end{pmatrix}$$

For $A = U\Sigma V^*$, $A^*A = V\Sigma^2 V^*$ which is an eigenvalue decomposition, i.e. $\sigma_i = \sqrt{\det(A^*A)}$ for $r = rank(A)$ singular values. Similarly $AA^* = U\Sigma^2 U^*$.

$$A^*A = \begin{pmatrix} 104 & -72 \\ -72 & 146 \end{pmatrix}$$

Find the eigenvalues

$$A^*A -= \begin{pmatrix} 104-\lambda & -72 \\ -72 & 146-\lambda \end{pmatrix}$$

$$\det(AA^* - \lambda I) = (104-\lambda)(146-\lambda) - 72^2 = 10000 - 250\lambda + \lambda^2 = 0$$

$$\lambda_1 = 200$$

$$\lambda_2 = 50$$

For $\lambda_1 = 200$, $(A^*A - \lambda_1 I)v_1 = 0$

$$\left(\begin{array}{cc|c} -96 & -72 & 0 \\ -72 & -96 & 0 \end{array}\right)$$

Thus the elements of v_1 must satisfy

$$72v_{1_2} = -96v_{1_1} \Rightarrow v_{1_2} = -{}^4\!/_3\, v_{1_2}$$

or

$$v_1 = \begin{bmatrix} 1 \\ -{}^4\!/_3 \end{bmatrix} v_{1_1}$$

To normalize the eigenvector, let $v_{1_2} = \frac{1}{\sqrt{{}^4\!/_3{}^2+1}}$

$$v_1 = \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix}$$

In the same way $v_2 = \begin{bmatrix} -0.8 \\ -0.6 \end{bmatrix}$ is obtained.

As $Av_i = \sigma_i u_i$

$$u_1 = \frac{1}{\sqrt{200}} \begin{pmatrix} -2 & 11 \\ -10 & 5 \end{pmatrix} \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix} = \begin{bmatrix} -\sqrt{2}/_2 \\ -\sqrt{2}/_2 \end{bmatrix}$$

$$u_2 = \frac{1}{\sqrt{50}} \begin{pmatrix} -2 & 11 \\ -10 & 5 \end{pmatrix} \begin{bmatrix} -0.8 \\ -0.6 \end{bmatrix} = \begin{bmatrix} -\sqrt{2}/_2 \\ \sqrt{2}/_2 \end{bmatrix}$$

The full SVD decomposition of A is

$$A = \begin{pmatrix} -2 & 11 \\ -10 & 5 \end{pmatrix} = \begin{pmatrix} 0.6 & -0.8 \\ -0.8 & -0.6 \end{pmatrix} \begin{pmatrix} \sqrt{200} & 0 \\ 0 & \sqrt{50} \end{pmatrix} \begin{pmatrix} -\sqrt{2}/_2 & -\sqrt{2}/_2 \\ -\sqrt{2}/_2 & \sqrt{2}/_2 \end{pmatrix}^*$$

Image Compression Using SVD

Consider a colored image storage usage. A pixel color is stored as specific intensities of three base colors: red, green and blue. The intensity of each base may vary from 0 (none) to 255 (max), which requires 8 bits or a byte of storage. Thus a $m * n$ colored image is stored as a 3D matrix, each level describing a base color, which requires $3mn$ bytes of storage. A lower rank SVD approximation of a photo matrix level $A_k = U_{m\times k}\Sigma_{\mathrm{k\times k}}\mathrm{V}^*_{\mathrm{n\times k}}$ would require $mk + k^2 + nk$ bytes of storage. As Σ is diagonal, only its k diagonal entries need be stored. This storage scheme compresses the input matrix when $k(m+n+1) < mn$ or $k < \frac{mn}{m+n+1}$.

The 2-norm of the error matrix or the discarded portion of A relative to the 2-norm of A is

$$\frac{\|A - A_k\|_2}{\|A\|_2} = \frac{\sigma_{k+1}}{\sigma_1}$$

The root mean square error of an approximation is

$$\sqrt{\frac{\sum_{m,n}\left(A_{ij} - A_{k_{ij}}\right)^2}{n}}$$

The compression ratio is the original to approximate matrix storage usage

$$CR = \frac{mn}{k(m+n+1)}$$

Example

Compress this $600 * 480$ colored image:

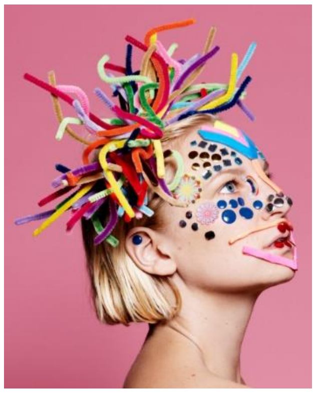

Figure 1 Original Image

Code

```
clear
clc
a=imread('a.jpg');
[m,n,d]=size(a);
kmax=floor((m*n)/(m+n+1));
da=double(a);
U=zeros(m,m);S=zeros(m,n);V=zeros(n,n);e=zeros(kmax,d);cr=zeros(kmax,1);rmse=zeros(kmax,d
);
for i=1:d
    [U(:,:,i),S(:,:,i),V(:,:,i)]=svd(da(:,:,i));
end
for k=1:kmax
    ca=zeros(m,n,d);
    cr(k)=m*n/(k*(m+n+1));
    for i=1:d
        cai=zeros(m,n,d);
        [ca(:,:,i),cai(:,:,i)]=deal(U(:,1:k,i)*S(1:k,1:k,i)*V(:,1:k,i)');
        e(k,i)=S(k+1,k+1,i)/S(1,1,i);
        rmse(k,i)=sqrt(sum(sum(((da(:,:,i)-ca(:,:,i)).^2)))/(m*n));
        imwrite(uint8(cai), sprintf('%dk%d.jpg',k,i));
    end
    imwrite(uint8(ca), sprintf('%dk.jpg', k));
end
figure
p=plot(1:kmax,e,'MarkerEdgeColor','r','MarkerEdgeColor','g');
set(p,{'color'},{'red';'green';'blue'})
xlabel('Approximation Rank k');
ylabel('Relative 2-Norm');
xlim([1 kmax])
legend('Red','Green','Blue')
grid on

figure
p=plot(1:kmax,rmse,'MarkerEdgeColor','r','MarkerEdgeColor','g');
set(p,{'color'},{'red';'green';'blue'})
xlabel('Approximation Rank k');
ylabel('RMS Erorr');
```

```
xlim([1 kmax])
legend('Red','Green','Blue')
grid on

figure
plot(1:kmax,cr);
xlabel('Approximation Rank k');
ylabel('Compression Ratio');
xlim([1 kmax])
grid on
```

Results

The reconstructed images of lower rank approximations $k = 5,10,50$ and 100 are showed in Figure 2. The exact values for said ranks properties are shown in Table 1. On visual inspection, approximations become acceptable around A_{50}, and indistinguishable from the original around A_{100}. The relative 2-norm, RMS error and compression ratio versus the approximation rank are shown in Figure 3, Figure 4 and Figure 5 respectively. The relative 2-norm decays exponentially as approximation rank increases. The error is less than 1.5% for $k > 50$ at all base levels. The compression ratio is generally higher for larger matrices (see formula).

Rank	**Relative $\|\|\,.\,\|\|_2$**			**RMS Error**			**Compression Ratio**
	Red	**Green**	**Blue**	**Red**	**Green**	**Blue**	
1	0.0915	0.1489	0.1484	46.007	45.708	43.902	266.42
10	0.04047	0.05664	0.04832	28.021	28.231	26.863	26.64
50	0.009286	0.01435	0.01385	10.046	10.93	11.197	5.33
100	0.003811	0.006282	0.006197	5.112	5.735	5.978	2.66

Table 1 Relative 2-norm, RMS Error and Compression Ratio for $k = 1,10,50,100$

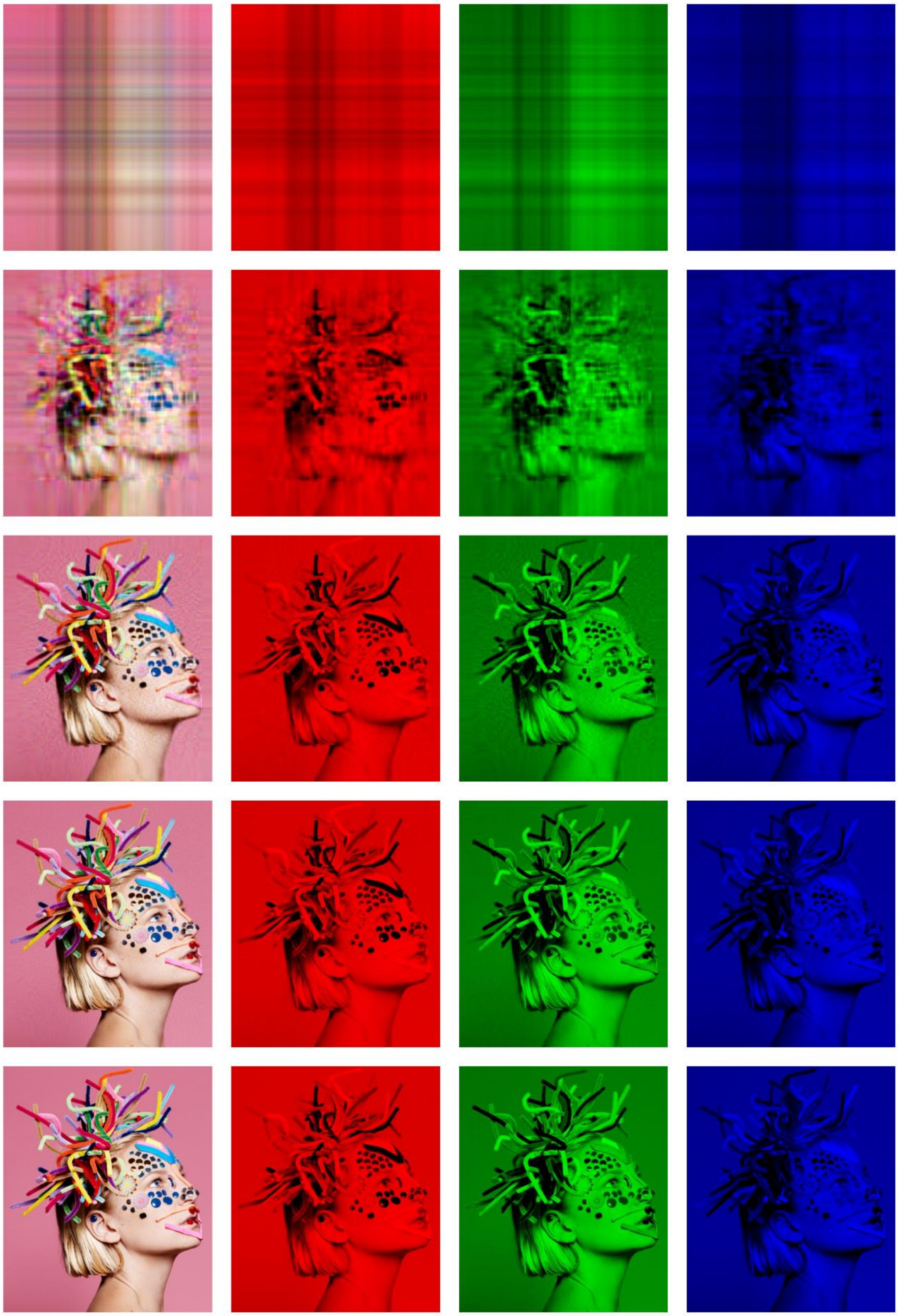

Figure 2 Approximated images and respective RGB components for $k = 1, 10, 50, 100$ *and original image*

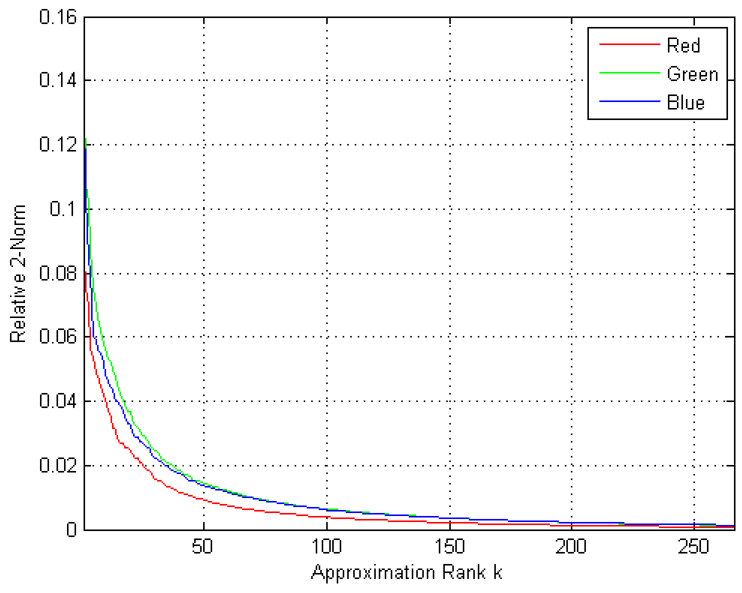

Figure 3 Relative 2-norm vs Approximation Rank k

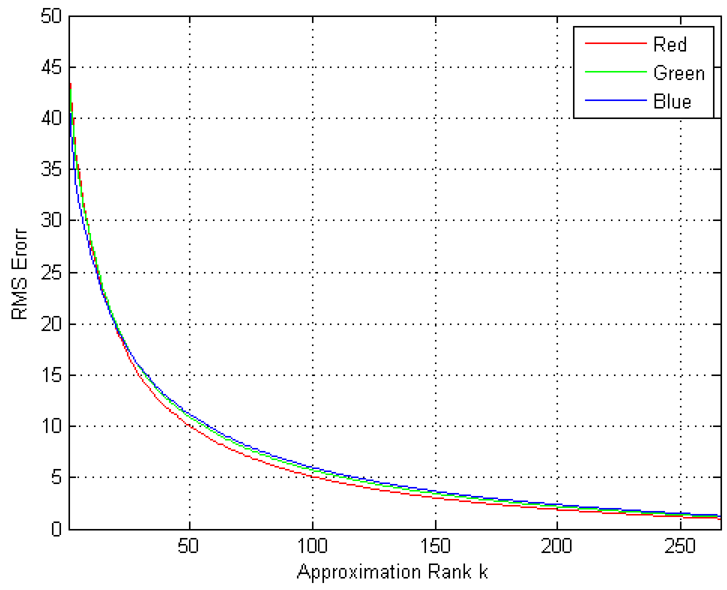

Figure 4 RMS Error vs Approximation Rank k

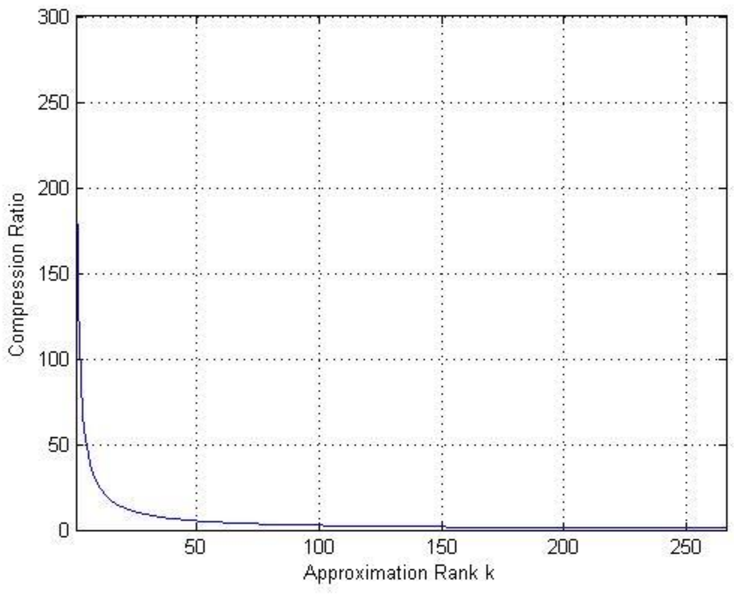

Figure 5 Compression Ratio vs Approximation Rank k

Noise Reduction

Consider an SVD decomposition of a noisy matrix. The smallest singular values that mainly represent noise are discarded to de-noise that matrix. Segmenting the matrix into blocks allows for better de-noising results compared to applying SVD to the entire matrix as each block is affected by noise differently.