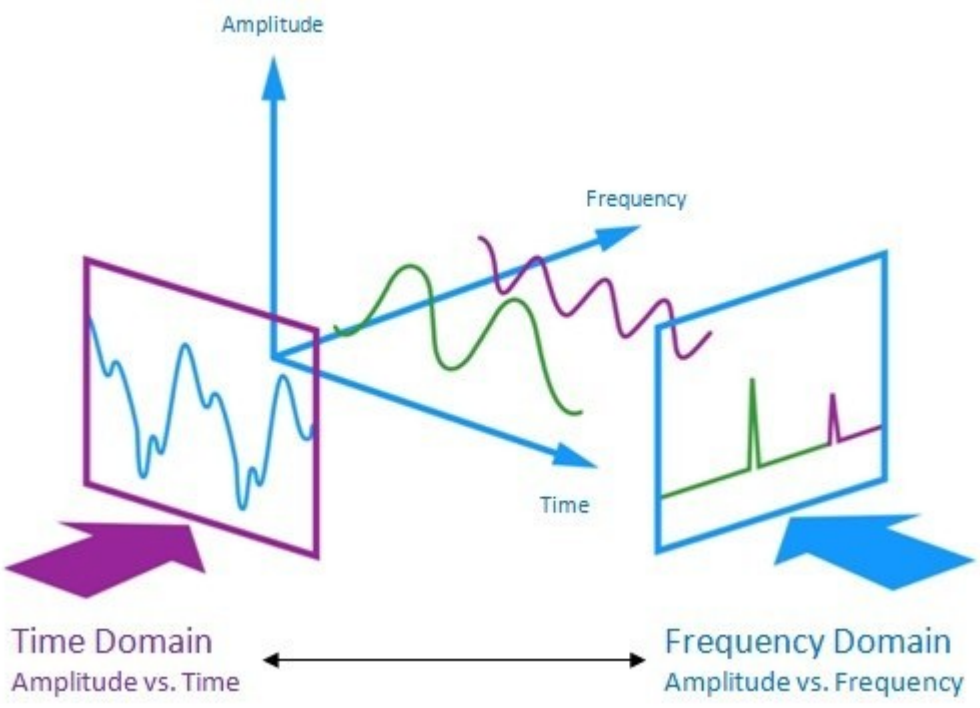
Digital Signal Analysis and Application Report

Assignment - 4



AAYUSH GOEL

20171188

QUESTION 1

1.

a. To calculate the DCT matrix for 8x8 window:-

$$T_{i,j} = \begin{cases} \frac{1}{\sqrt{N}} & if\ i = 0 \\ \sqrt{\frac{2}{N}} cos\left[\frac{(2j+1)i\pi}{2N}\right] & if\ i > 0 \end{cases}$$

- We apply the above formula using meshgrid.

b. To get the DCT of image of size 8x8:-

- We apply the DCT matrix obtained from part a from the left side of image by multiplying and the transpose of the same from the right side.
- Dct_image = F * im * F'

c. To get the inverse DCT of image of size 8x8:-

- We apply the DCT matrix obtained from part a from the right side of dct_image by multiplying and the transpose of the same from the left side.
- Dct_image = F' * im * F (As F * F' = I, I is the Identity Matrix)

d. To get the quantised image :-

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

- We apply the quantisation matrix to the image by point-wise division with the image after multiplying a scalar c to it, to increase the quantisation effect.

e. To get the DCT of image of size 8x8:-

- We apply the quantisation matrix to the image by point-wise multiplication with the image after multiplying the scalar c to it (same as used in the quantisation step).

f. To calculate the RMSE of two images-

- The images are subtracted to get the point-wise difference which is then squared.
- Mean of all the elements of the newly obtained matrix is taken, say M.
- RMSE = sqrt(M).

g. To get the entropy of image:-

- imhist() function is used to get a count of the different pixels that appear in the image.
- The count obtained is normalised by dividing each element by the size of image (height * width)
- Entropy = -sum(p * log2(p)) is used.

2.

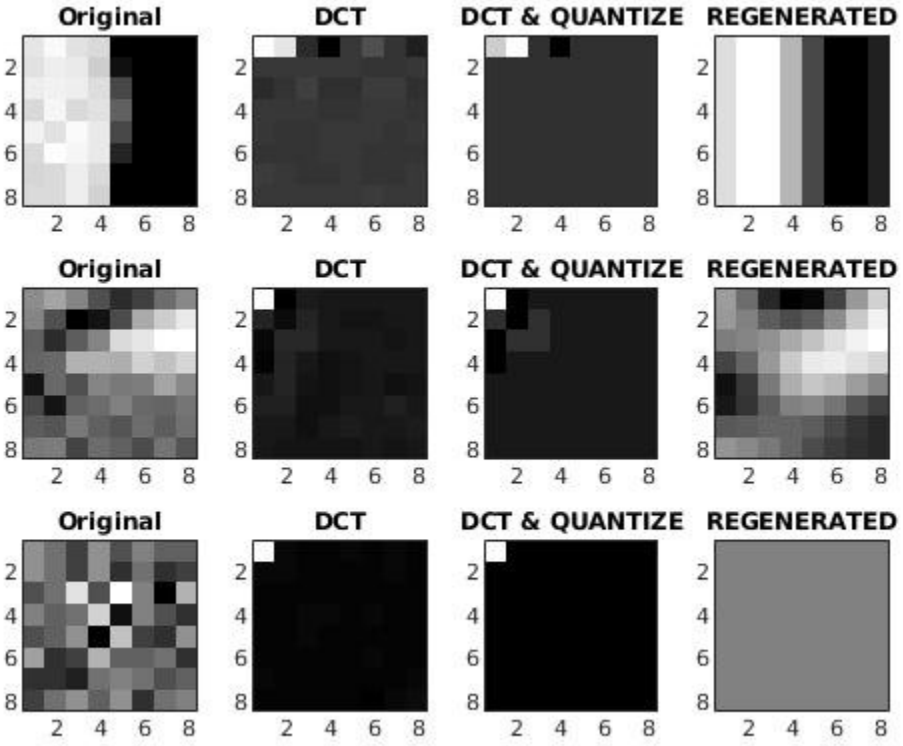
The given windows with their top left corners at points (420, 45), (427, 298) and (30, 230) are selected and then the following pipeline is applied to the windows:-

- DCT of image using myDCT function
- Quantisation using myDCT_quantisation
- Rounding of the image
- Dequantisation using myDCT_dequantisation
- Inverse DCT using myIDCT
- All the images are reconstructed and plotted at each stage

Observations:-

- The irregularities of different color shades is not seen in the reconstructed image.
- The sudden and irregular changes as seen in the 3rd window, which cannot be perceived by human eye are removed in the reconstructed image.
- The final images are smooth and there is no sudden fluctuation in the colors.

3.



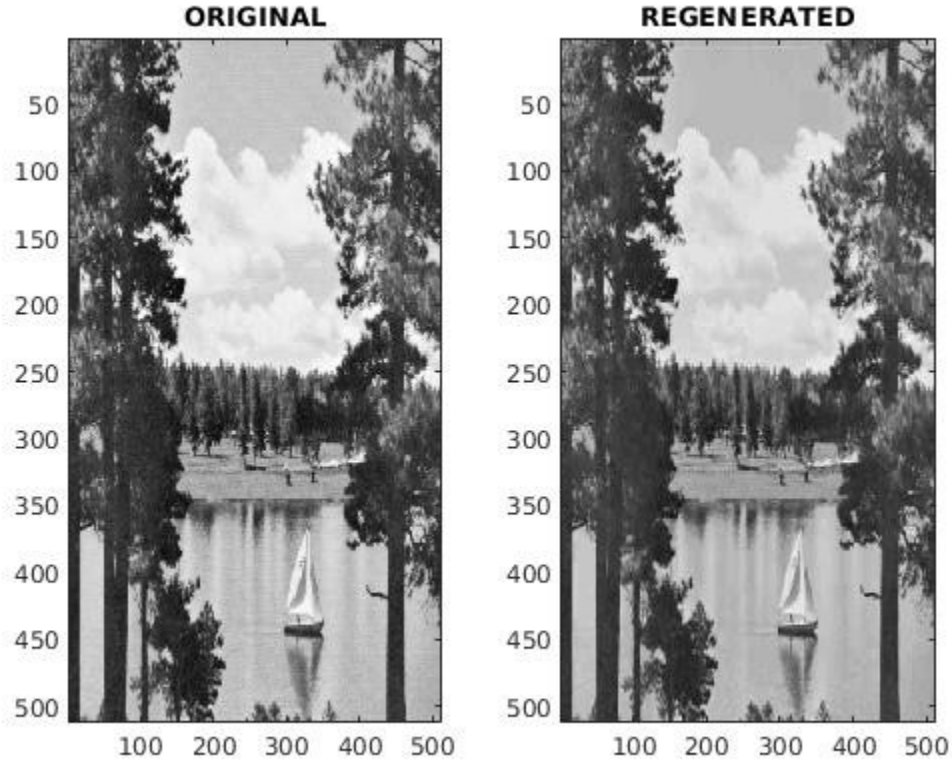
Method:-

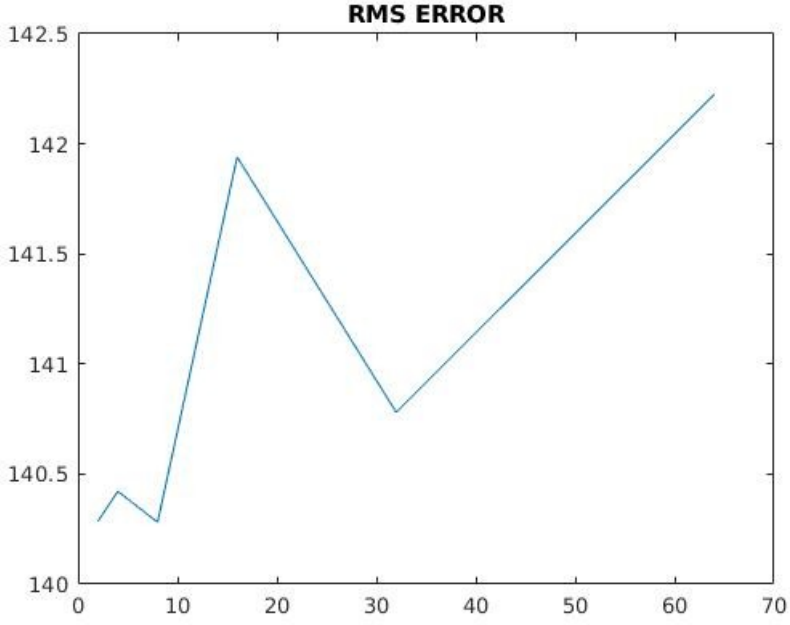
- The input image is divided into windows of size 8x8
- The pipeline mentioned above is applied to each window
- The old image is replaced by the new windows in the same order respectively to create the final image.

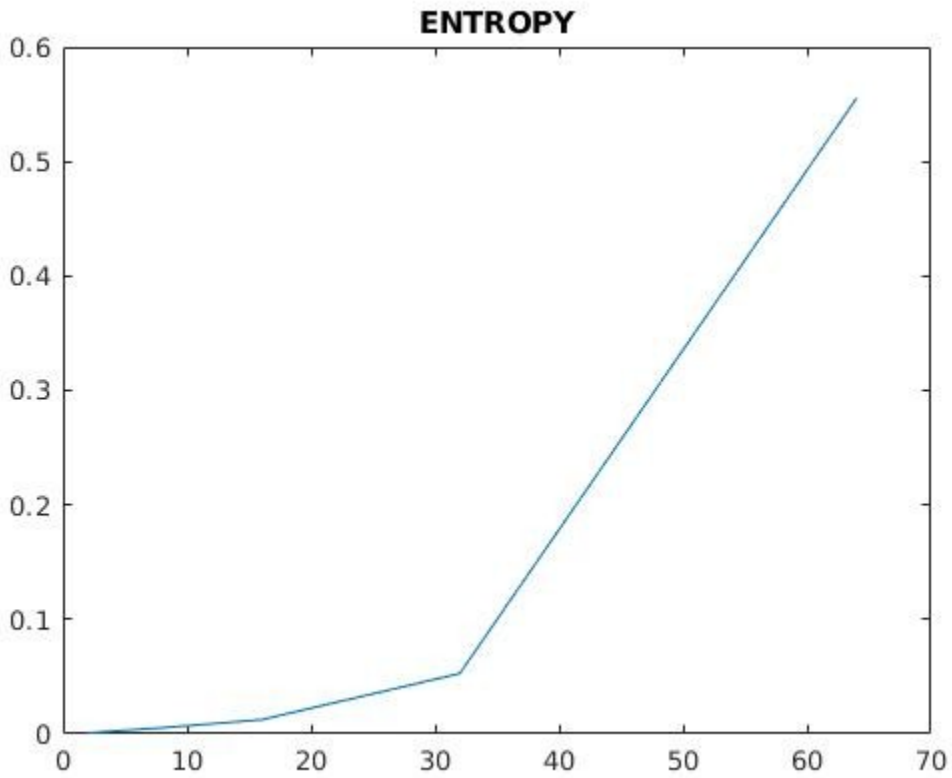
Observation:-

- **Quality of Image is reduced after JPEG compression** owing to the loss of
- actual content of the image.
- **Sharp edges and lines** becomes blocky/pixelated in nature.
- **Brightness of image is reduced** due to dividing by quantisation matrix and rounding off.

4. At c=2


ORIGINAL
REGENERATED
50
100
150
200
250
300
350
400
450
500
100
200
300
400
500


RMS ERROR
142.5
142
141.5
141
140.5
140
0
10
20
30
40
50
60
70



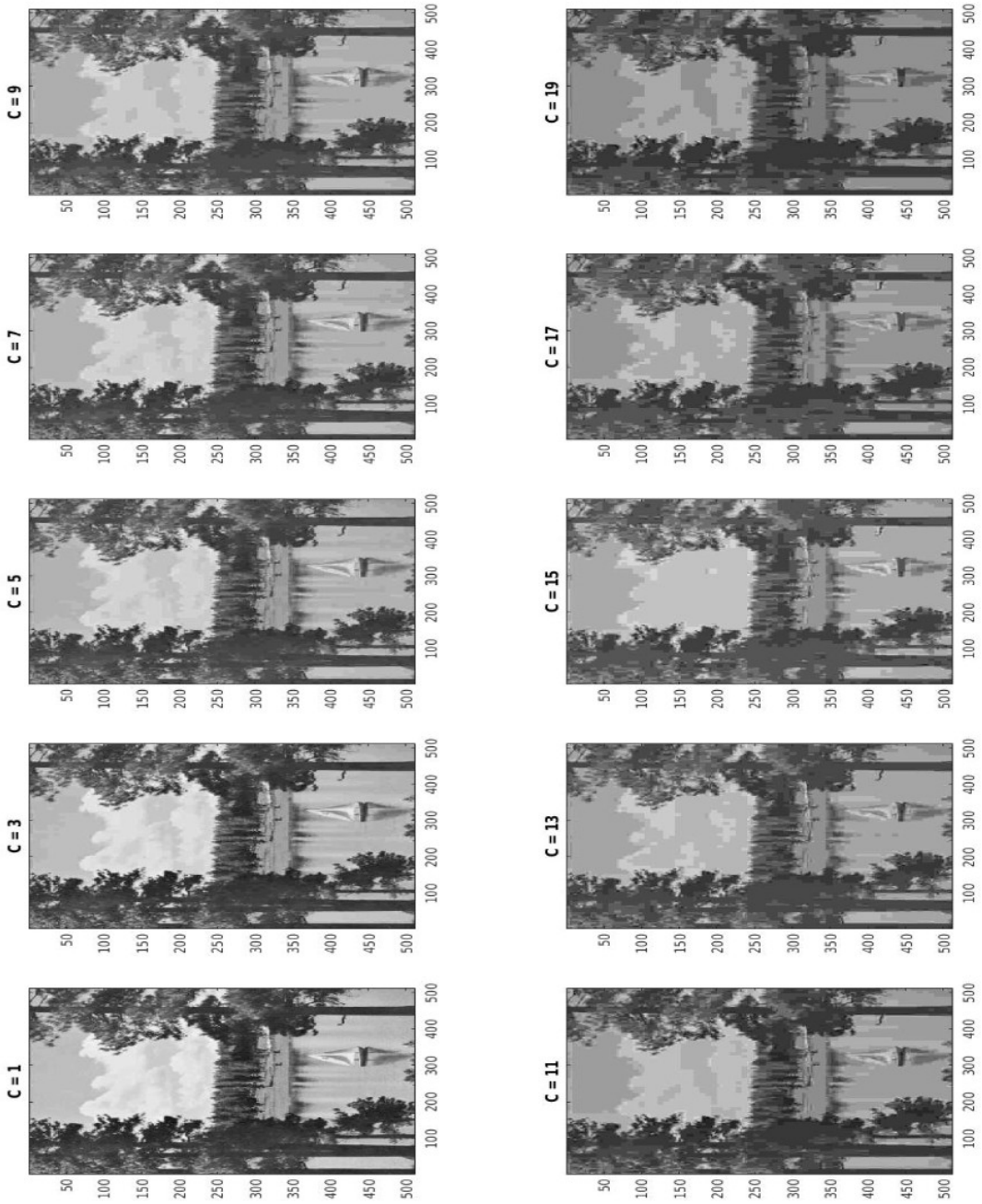
Observation

- As seen in image above, both RMSE and Entropy increases with increase in c due to loss of huge amount of data as a lot values in image tends to zero.
- This makes the final images different from the original, thus increasing RMSE and Entropy.
- At around c=9, the image starts to get pixelated and we can sense the difference between the smooth original and reconstructed image.
- At c=10, since the values of quantisation matrix increase it leads to more loss of data leading to less brightness and pixelated image.

C	RMSE	Entropy
2	140.2857	0.0008
4	140.4214	0.0010
8	140.2810	0.0025
10	140.0419	0.0071

16	141.9398	0.0035
32	140.7808	0.0045
64	142.2246	0.0059

IMAGE PLOT FOR DIFFERENT VALUES OF C



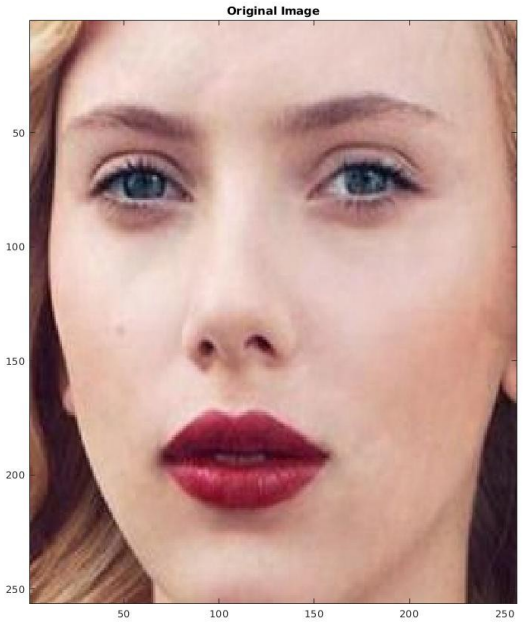
QUESTION 2

1.

Steps:-

- First, the data matrix A is created by reading the images and converting all the channel(RGB) matrices to rows and appending one below other.
- $L = A * A^T$ is calculated and its eigenvector matrix V is found out which forms the basis for our PCA space.
- But we need eigenvectors for matrix $K = A^T * A$, which can be found out by left multiplying eigenvector matrix of L with A^T ($U = A^T * V$) because K is the real data matrix of dimensions (256*256*3) x (256*256*3).
- To select N (Number of Principal Components), we sort the eigenvectors based on the eigenvalues, that is, the eigenvector with the largest eigenvalue contribute most to the reconstruction of the images, to get matrix U with dimension 520 x N.
- To find PCA, we need to maintain the condition of $U^TU = I$, so U should be normalised before use.
- To go to the PCA space, we perform PCAspace = A * U.
- Each row in PCAspace matrix represents an image in the order as the original data matrix.
- We can take any row from the PCAspace say X, apply the transform $U*X^T$ to it and reconstruct the image but with reduced components N as specified above.

COMPRESSED Images get stored in folder compressed_dataset on running the program



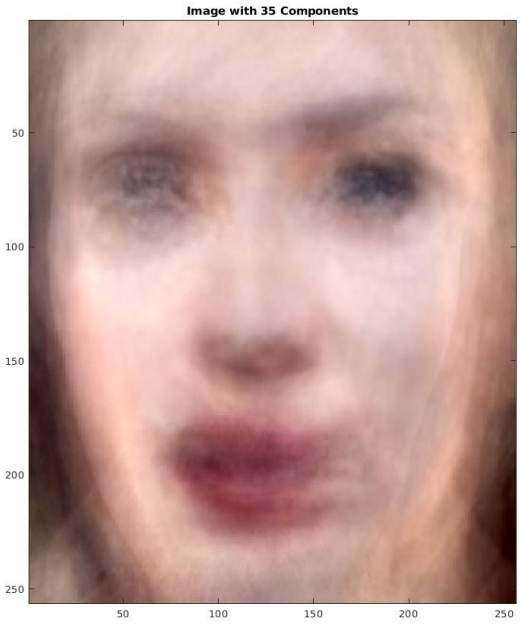
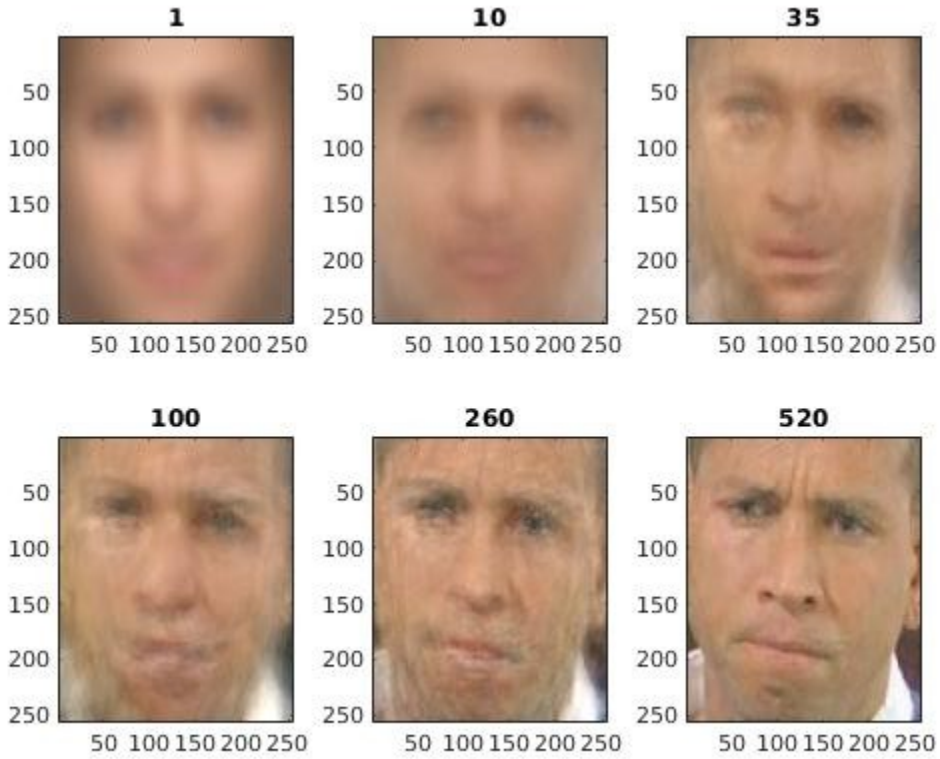


IMAGE PLOTS FOR DIFFERENT VALUES OF NUMBER OF COMPONENTS

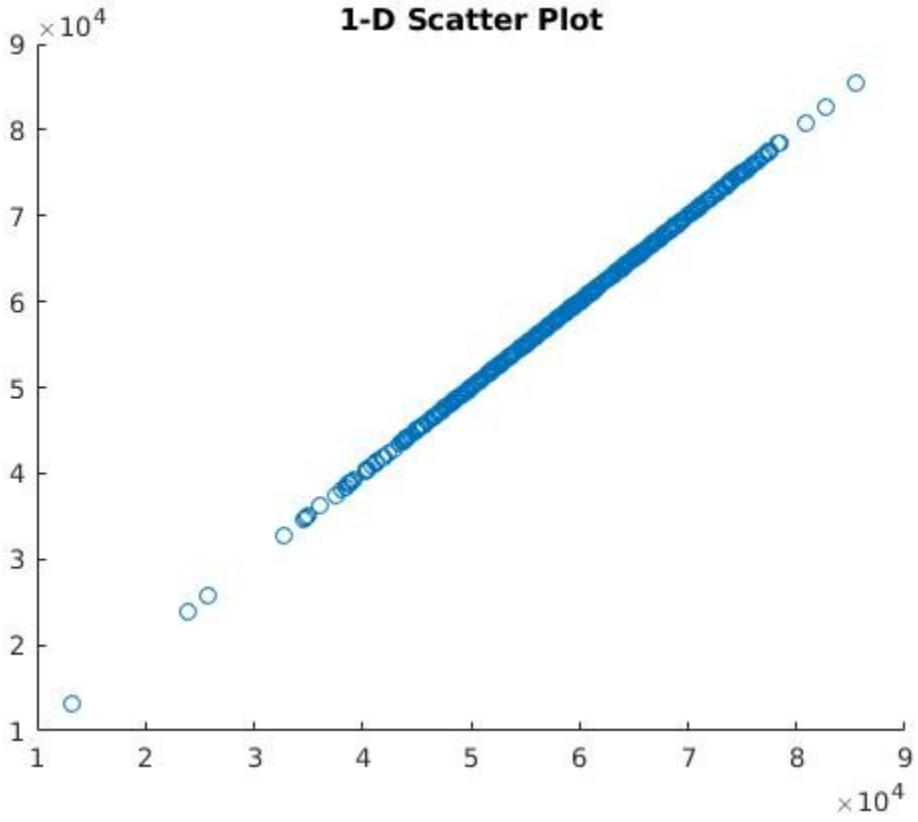


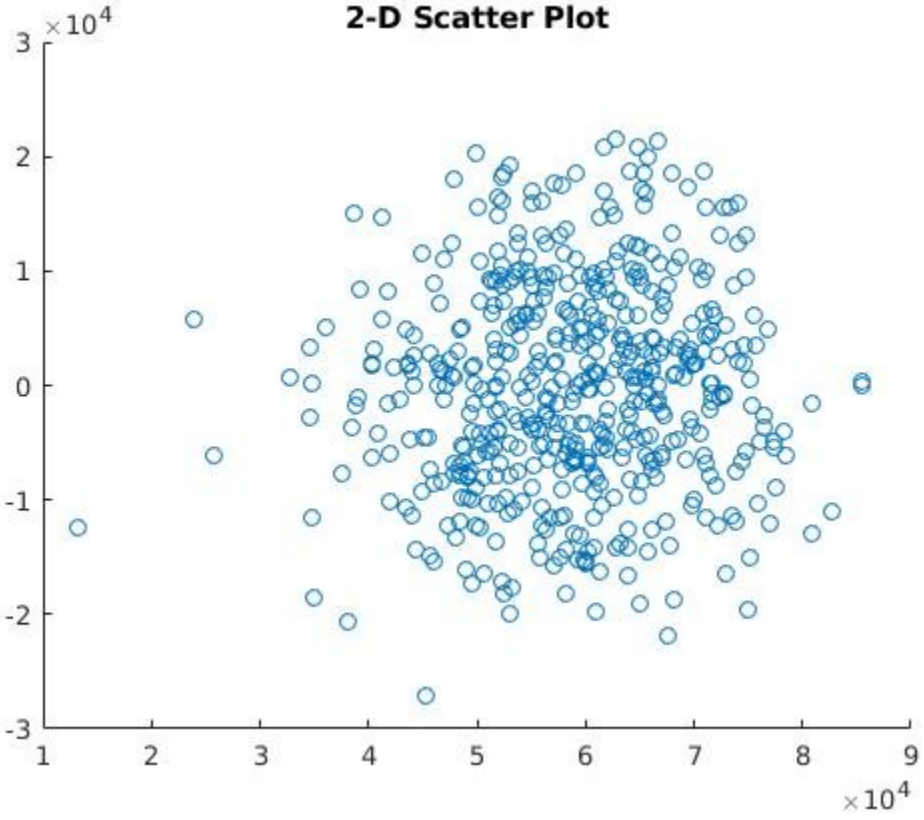
NOTE:

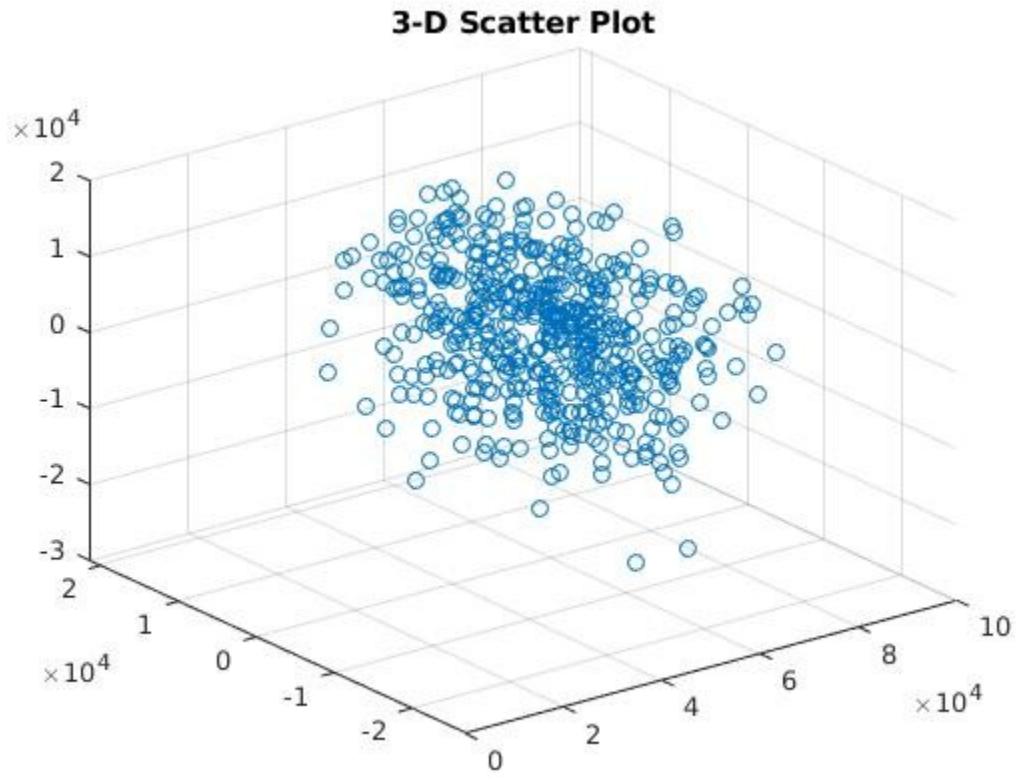
- We cannot use $A^T * A$ due to memory limitations.
- As the number of components increase, we tend towards our original image.

2.

Image Plots in Different Dimensions represents the spread of images along the different principal component axis.


1-D Scatter Plot
×10^4
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
×10^4


2-D Scatter Plot
×10^4
3
2
1
0
-1
-2
-3
1
2
3
4
5
6
7
8
9
×10^4


3-D Scatter Plot
×10^4
2
1
0
-1
-2
-3
2
1
0
-1
-2
×10^4
0
2
4
6
8
10
×10^4