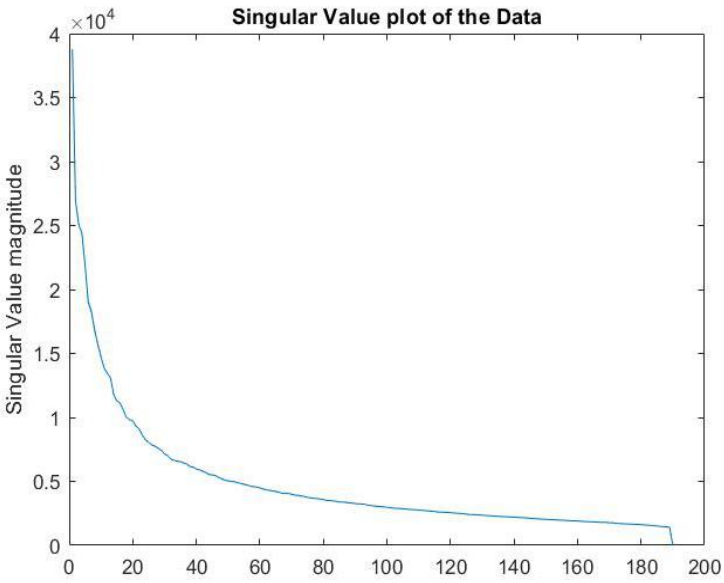
# ECE269 - Linear Algebra
## Project- Face Recognition using Principal Component Analysis.

Shiladitya Biswas
PID- A53303020

a) The Singular Value Plot of the Data Matrix is shown below:




The singular values of a matrix represents how important the corresponding eigenvector is to the Dataset. In other words higher the magnitude of the singular value higher is the information carried by the respective eigenvector/PCs. In my code I used the MATLAB SVD function to calculate the eigenvalues,singular values and their corresponding ortho-normal eigenvectors also known as Principal components. I chose a set those eigenvalues and corresponding PCs that contribute to 90% of the sum of all the eigenvalues. In this way I discarded the PCs that has minimum contribution to the reconstructed image and achieved dimension reduction.

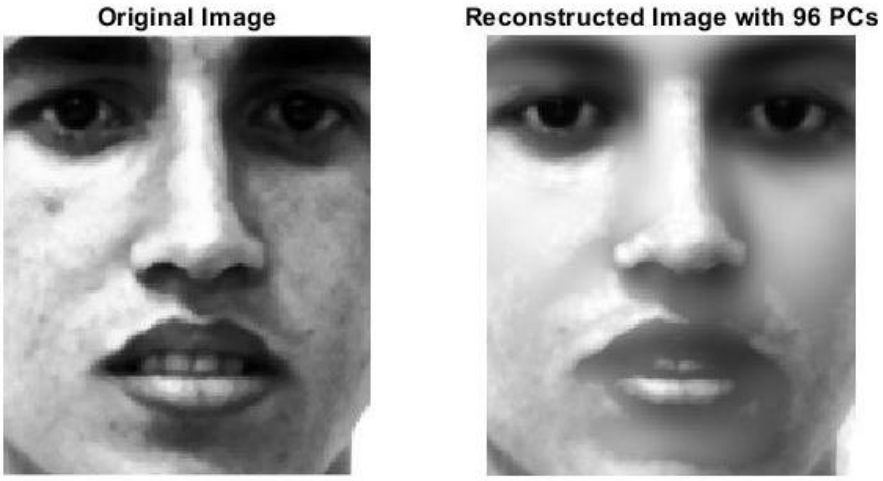
Mathematically, I chose the value of k such that

$$\frac{(\sigma_1)^2+(\sigma_2)^2+(\sigma_3)^2+.....+(\sigma_k)^2}{\sum_{i=1}^{190}(\sigma_i)^2} > 0.95$$

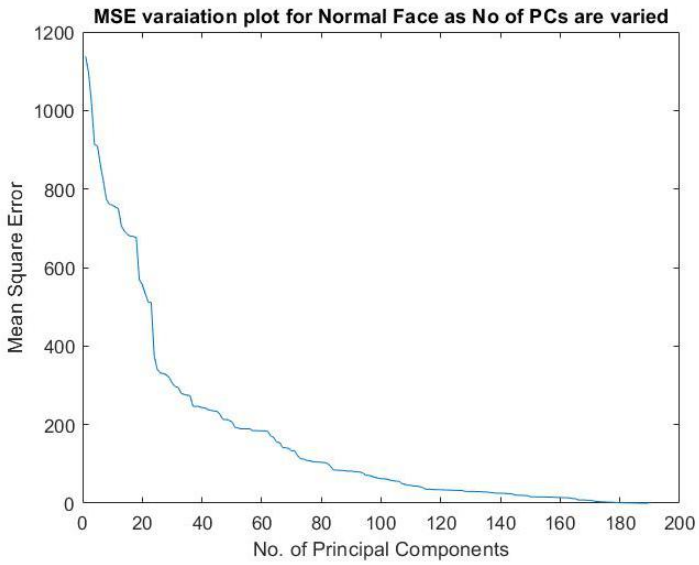
Where $\sigma$ represents singular values and $\sigma_1 > \sigma_2 > \sigma_3 ..... > \sigma_n$

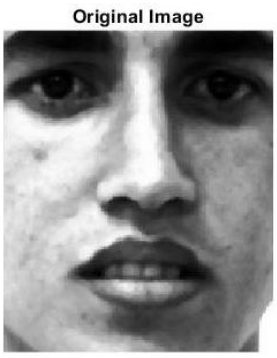
And finally chose the eigen-vectors/principal components corresponding to the k eigenvalues. I got k= 96 i.e. 96 PCs are enough to represent all the face images. Although, I found this k selection technique on my own, this technique is also available on Wikipedia here.

The reconstructed image with 96 PCs is show below. We see that the reconstructed image resembles the original image hence my selection of number of PCs is justified.



b) The Mean Squared Error (MSE) variation of neutral face expression as a function of Number of Principal Components is shown below. Furthermore the Original and reconstructed image (I chose image “107a.jpg”) is also shown.

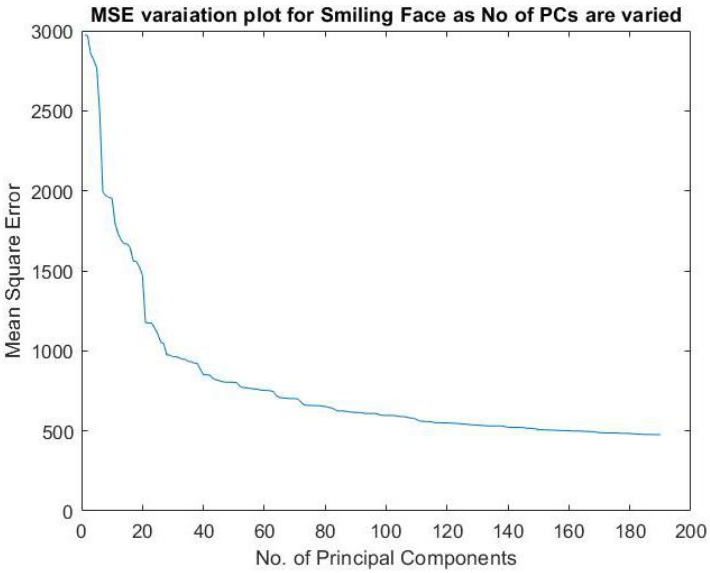




Comments: On observing the MSE vs No of PCs plot we see that the MSE of the reconstructed image goes down to a very small value as we increase the No of Principal components to reconstruct the image. This is because as the no of PCs increases we keep adding more information to the reconstructed image. Hence the reconstructed image resembles the input image even more, as a result the MSE decreases. Another reason why the MSE reduces to a very small value is because the image to be reconstructed already belongs to the training set, from which we evaluated our PCs.

c) The Mean Squared Error (MSE) variation of Smiling face expression as a function of Number of Principal Components is shown below. Furthermore the Original and reconstructed image (I chose image “107b.jpg”) is also shown.

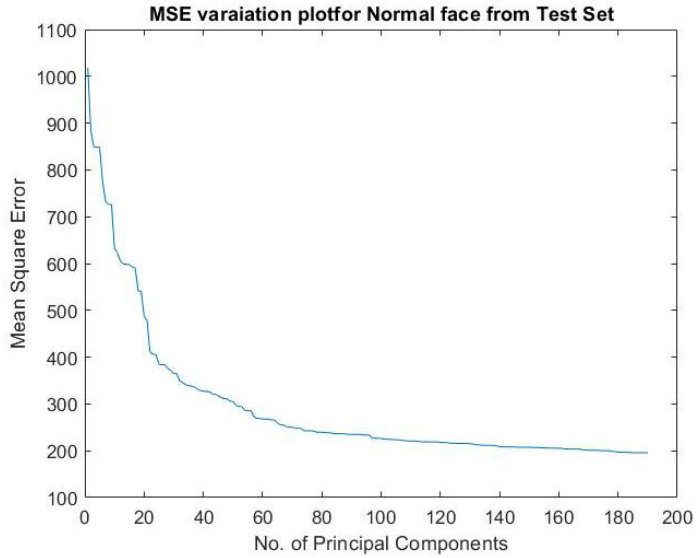





Comments: In general, when a person smiles the area’s near his mouth undergoes maximum change and other areas like eyes, nose, eyebrow etc undergo minimal change. As a result the smiling image of a person still has a lot of similarities (although not 100% similar) to the same person’s neutral expression image. Thus we are able to partially reconstruct (especially the image areas which has

undergone minimal/unrecognizable change for eg. Eyes, nose etc) the smiling face expression image of the person using the principal components of neutral expression images. This argument is further endorsed by the MSE plot, we observe that as the No of PC's increases the error keeps decreasing. But then we also see that the MSE converges to a relatively high value as compared to the MSE plot of Neutral Expression image reconstruction. This is because some feature like eyes,nose etc of the reconstructed face image matches the original image and other feature near the mouth area (wherever there is drastic change in the image) doesn't match.

d) In this case we are asked to reconstruct the neutral expression image of an individual not belonging to the training set. The reconstructed image and the corresponding MSE plot is shown below.






Comments: We observe that the reconstructed image is quite similar to the original image and is recognizable. It is observed that the MSE plot goes down to a much lower value as compared to that in part (c) but is higher than that obtained in part (b). This is due to the fact that that unlike in part C where we tried to reconstruct smiling images, here the original image is expressionless, hence it belongs to the face space and can be reconstructed to get a recognizable image. However, here the original image doesn't belong to the training set (from where we calculated our PCs), hence we cannot obtain the original image perfectly (even if we use all the eigenvectors/PCs for reconstruction), but the individual is still recognizable. The MSE can be further reduced by increasing the size of the training set.

e) Reconstructed image of a non-human object i.e. a flower vase is shown below. As we can see, the reconstructed image not at all resembles the original image. This reinforces the most basic property of a vector space i.e. "**A vector space is closed both under vector addition and scalar multiplication**". Here my vector space is of dimension 190 (since we have 190 Principal components and from lectures we know that PCs are nothing but basis of a vector space, hence the Face vector space has dimension 190) and each point in this space represents a face. Each of these faces is linear combination of the 190 PCs. Now, the closure property of Vector space states that :



Given vectors $u, v \in$ Vector space $V$

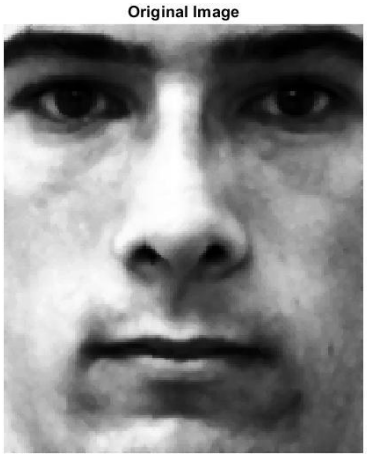
$\Rightarrow u + v \in V$

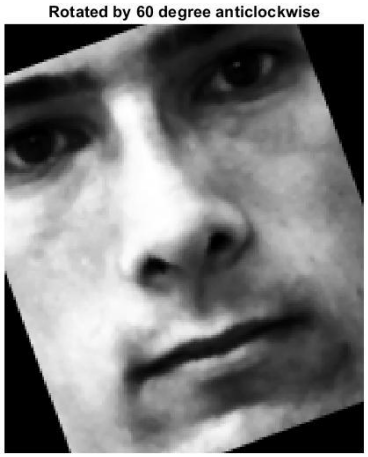
$\Rightarrow \alpha.u$ and $\alpha.v \in V$ , where $\alpha \in R$

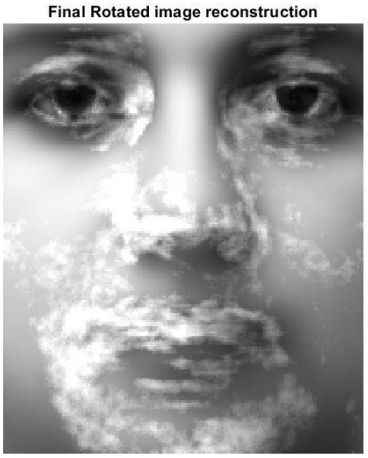
Which further implies $\alpha.u + \beta.v \in V$ Where $\alpha, \beta \in \mathrm{R}$

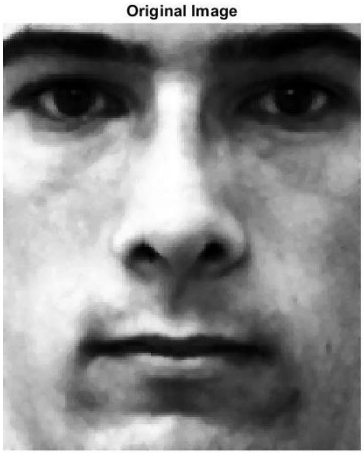
This implies any linear combination of all the 190 PCs of the face vector space will also belong to the Face vector space and not in any other vector space. As a result we are not able to construct the image of a flower vase by linearly combining the basis of the face vector space. The flower vase image doesn't belong to the face vector space at all. Hence we aren't able to reconstruct it.

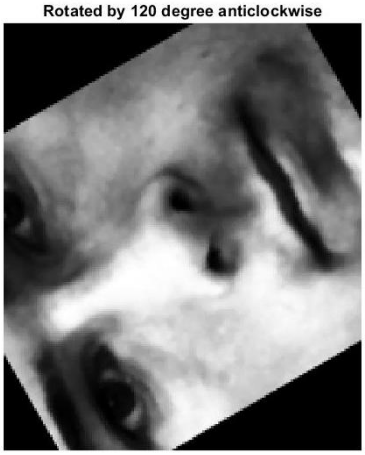
f) The reconstructed image of rotated original image is shown below for angles 60,120,180,240,300 and 360 degree.



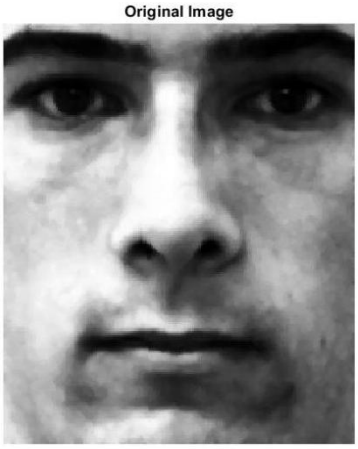



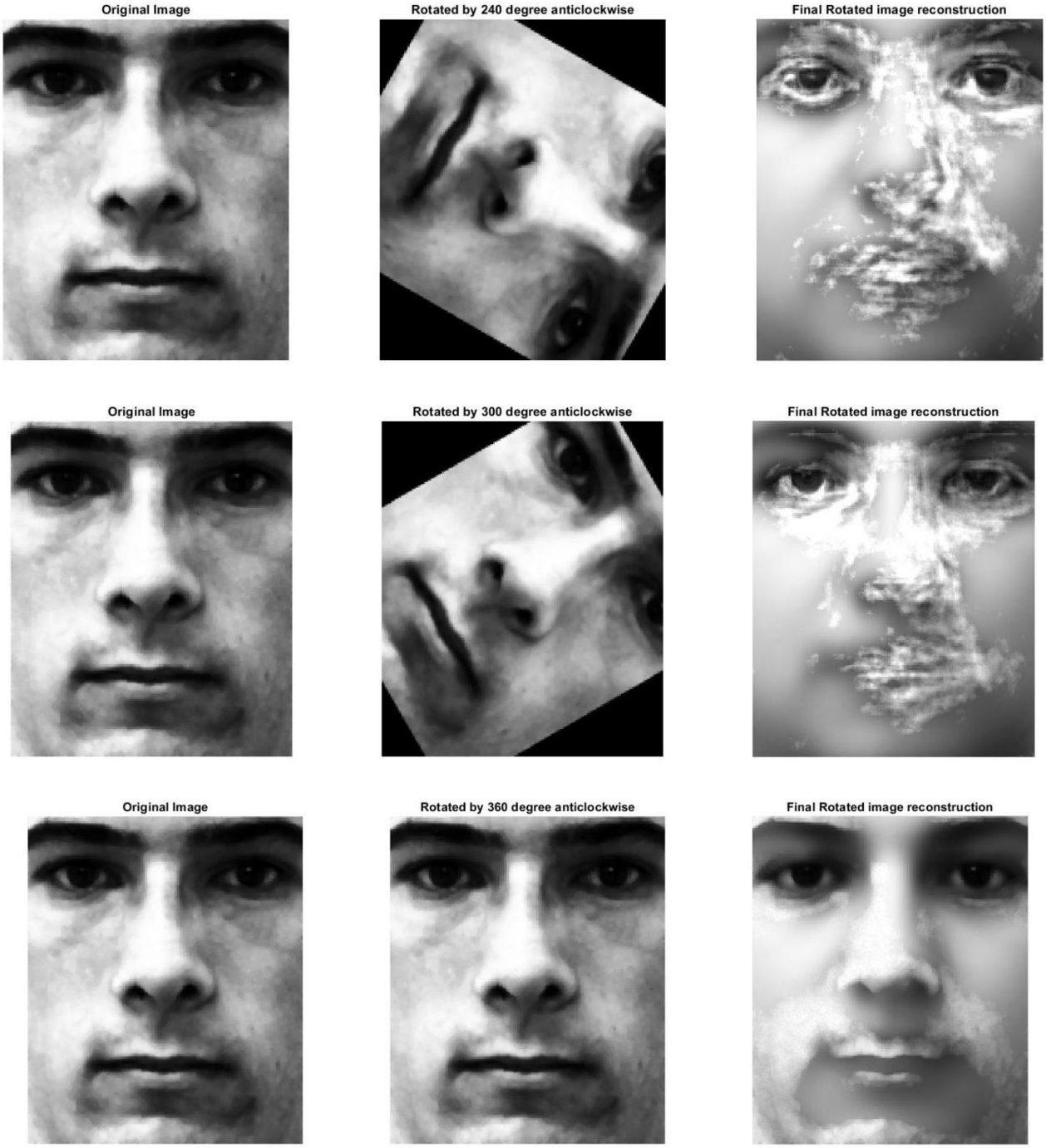





g)

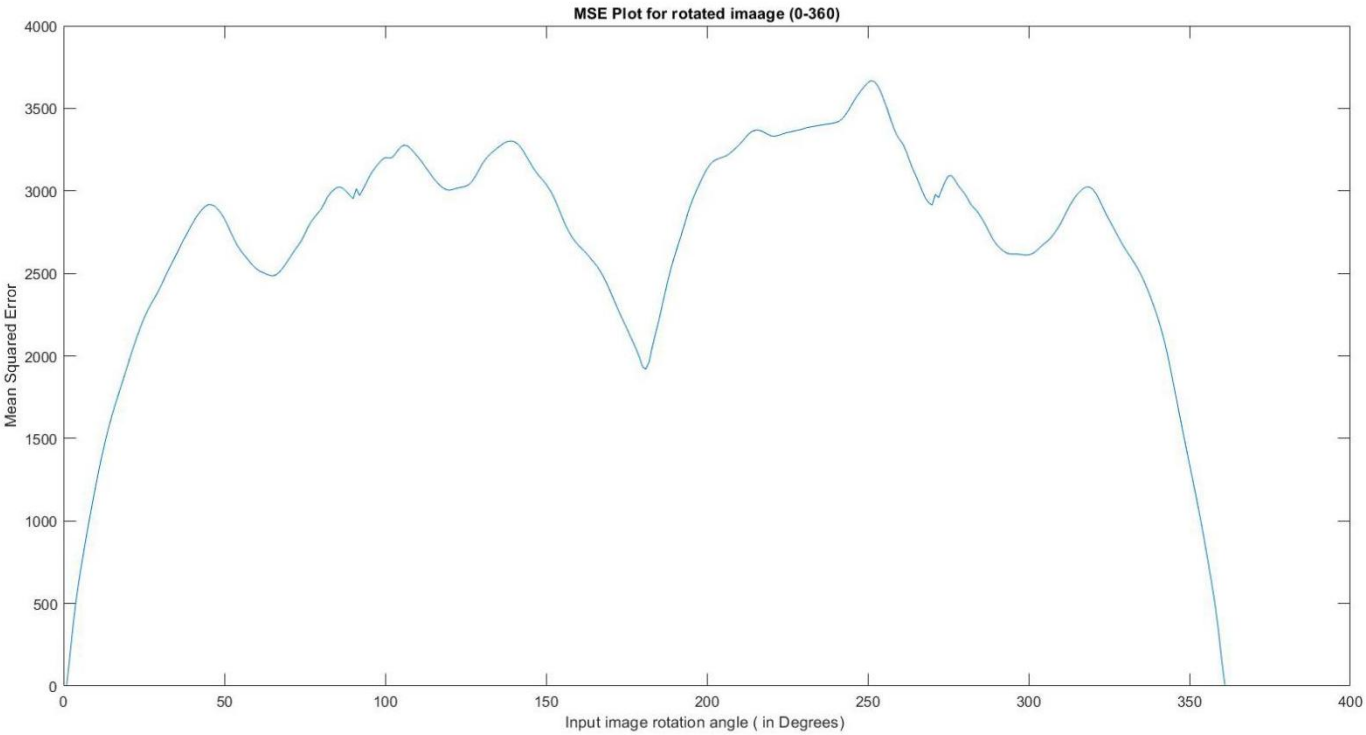




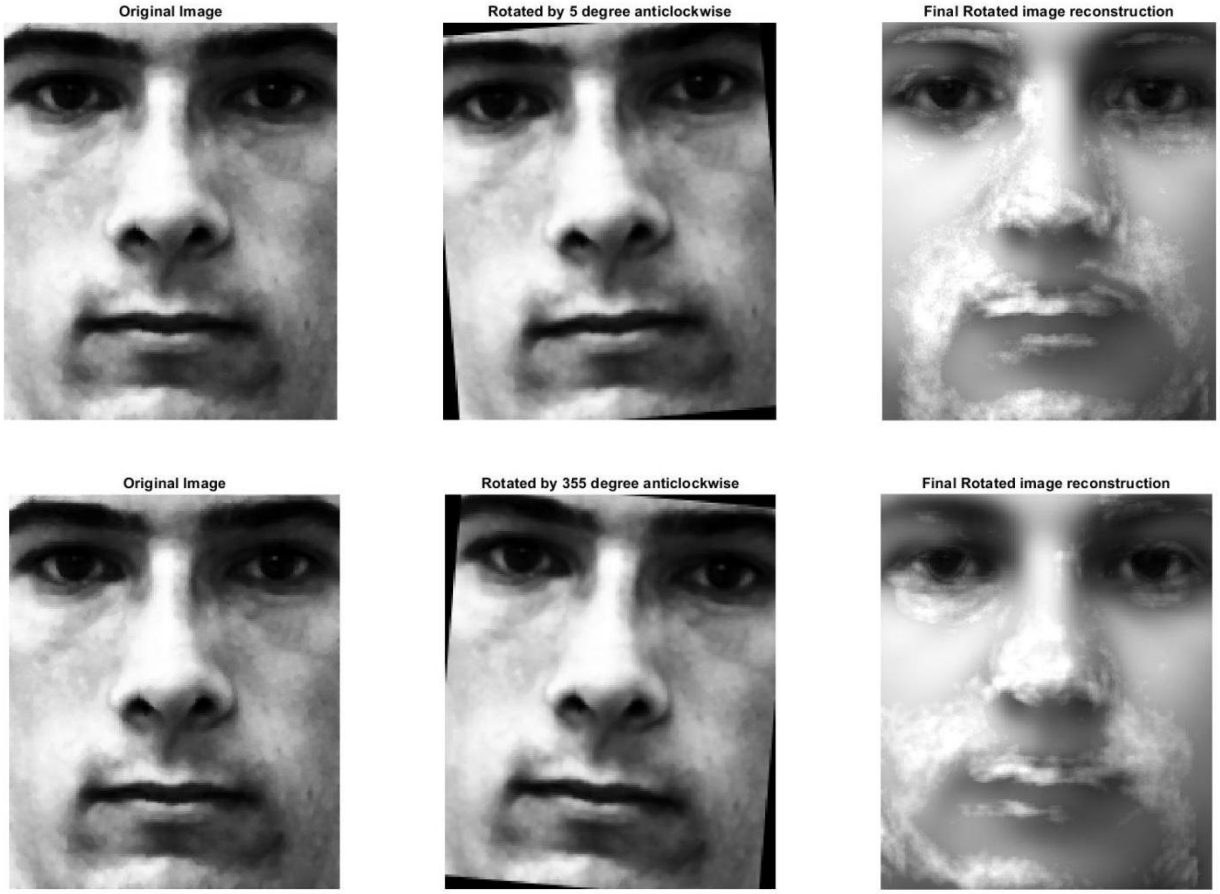



We observe that the reconstructed image is nowhere close to the input image (except for 360 degree or upright face) and is always straight/upright regardless of the input image rotation angle. This is because there were no rotated image in the training set, as a result the PCs obtained can only define the face space containing upright faces. Therefore a linear combination of these PCs will always give up upright faces.

The MSE plots for the reconstructed image by rotating the input image from 1 to 360 degree is shown below. Plus I have recorded a small video (because it looks cool !!) please follow the link.
The plot endorses my previous claim that the model fails miserable while trying to reconstruct images rotated at angle between 5 to 350 (or -5).




For angle between 0-5 and 355 to 360 degrees the MSE is low and the constructed image is recognizable. This is because for small angle of rotation nearly 90-95% of the image is still the face and we have less noise (in terms of zero paddings) in the input image. For angles in the range between 5 and 355, the image has a lot of noise due to zero padding and hence the MSE error is very high. At 180 degrees the input image has zero noise, thus the MSE is low (as compared to other angles between 5-355). But at this angle the image is upside down, and as mentioned before, our training set never had rotated images, hence it again fails to reconstruct the image.
The plots for 5 and 355 degrees is shown below.



# CODE

```
clc
clear
%%
% M=190
Add=double(zeros(31266,1));
% For loop to go through 190 images and vetorize them. In the same loop I
% have used the Add variable to add all the images. The 'Add' variable will
% be used later to find the average of all the images and stored in
% variable 'psi'.
for i=1:190
    im=double(imread(strcat(int2str(i),'a.jpg')));
    im=reshape(im',[31266,1]);
    gamma(:,i)=im;
    Add=Add+im;
end
%% Average Image
psi=(Add/190);
psi_img=uint8(reshape(psi,[162,193]))'; % Reshaping 'psi' to 193x162 to
imshow(psi_img);                        % display  Average image
title('Average Image')
%% Getting the A matrix by subtracting the Average face vector 'psi' from
%% each 190 vectorized image and storing them in the columns of A.
for i=1:190
   phi(:,i)=(gamma(:,i))-psi;
end
A=phi;
A=double(A);
%% Plot singular values of Matrix Data matrix i.e. A
L=((A')*A); %  Calculating the L matrix i.e. L = (A^T)*A
[U,S,V]=svd(A);
% V is the eigenvector matrix of A'A or L
% Singular Value Plot for the Eigenvalues of data matrix
for i=1:190
    sing_Val(i)=S(i,i); % Putting Diagonal Values of S into sing_val array.
end
figure,plot((sing_Val)) % Plotting the singular values of the data in descending
order
title('Singular Value plot of the Data');
ylabel('Singular Value magnitude');
%% Section to find the most important PCs.
eigvalue=sing_Val.^2; % Getting the eigenvalues from singular values.
eigsum=sum(eigvalue);
csum=0;
k90=0;
% d=10;
% for d=1:190
    for i=1:190
       csum=csum+(eigvalue(i));
       tv=csum/eigsum;
       if tv>0.95
         k95=i;
         break;
       end
    end
% end
```

```
fprintf('No of Principal components: %d \n',k95)

%% Getting the Eigen vector matrix for (A*(A^T)) matrix.
%% In other words finding the eigen Face vectors
eignVect_A=A*V; %

%% Reconstructing 107a.jpg image using different nos of PCs. PC=1:190
%% finally plotting the MSE vs no of PCs plot.
mse_normal=[];
imge=double(imread(strcat('107a.jpg')));
imge=reshape(imge',[31266,1]);
w1=[];
for n=1:190 % For loop to plot the MSE vs No of PCs variation. Here n= No of PCs
and if we replace 190 by "k95" we get the reconstructed image by 96 PCs.
    Vmax=V(:,1:n); %getting the corresponding "orthonormal Eigenvectors" or PCs
    eignVect_phi_Max=A*Vmax;
    % Now I normalized the calculated Eigenvectors, else some pixel values
    % exceed 255 and as a result proper reconstruction of the image doesn't
    % occur.
    for i=1:n

eignVect_phi_Max(:,i)=eignVect_phi_Max(:,i)/norm(eignVect_phi_Max(:,i));
    end
    % Calculating the weight vector w where each element of w represents
    % the contribution of each eigenface vector to the reconstructed image.
    w1= (eignVect_phi_Max')*(imge-double(psi));
    rec_img=(eignVect_phi_Max*w1); % Reconstructing the image
    err=imge-(rec_img+psi); % Calculating the error w.r.t original image
    mse_normal(n)=(((err')*err)/31266); % Calculating Mean Square Error.

    % Displaying the reconstruction process as the number of PCs increases.
    % Uncomment the line below to see the reconstruction process.
    %imshow((reshape(uint8(rec_img),[162,193]))'+psi_img);   //
end

% %Plotting the final reconstructed image and MSE w.r.t no of PCs.
figure, plot(mse_normal);
title('MSE varaiation plot for Normal Face as No of PCs are varied');
xlabel('No. of Principal Components');
ylabel('Mean Square Error');
figure,subplot(1,2,1);
imshow(imread(strcat('107a.jpg')));
title('Original Image');
subplot(1,2,2);
% figure
imshow((reshape(uint8(rec_img),[162,193]))'+psi_img); %Adding the average
vector back and displaying the reconstructed image
title('Reconstructed Image with 190 PCs');

%% MSE for Smiling images

imge=double(imread(strcat('107b.jpg')));
imge=reshape(imge',[31266,1]);
w1=[];
% figure;
for n=1:190
% n=70;
Vmax=V(:,1:n); %getting the correspoding "orthonormal Eigenvectors" or PCs
```

```
    eignVect_phi_Max=A*Vmax;
    % Now I normalized the calculated Eigenvectors, else some pixel values
    % exceed 255 and as a result proper reconstruction of the image doesn't
    % occur and we get white patches here and there on the image.
    for i=1:n

eignVect_phi_Max(:,i)=eignVect_phi_Max(:,i)/norm(eignVect_phi_Max(:,i));
    end
    % Calculating the weight vector w where each element of w represents
    % the contribution of each eigenface vector to the reconstructed image.
    w1= (eignVect_phi_Max')*(imge-psi);
    rec_img=(eignVect_phi_Max*w1);% Reconstructing the image
    err=imge-(rec_img+psi);% Calculating the error w.r.t original image
    mse_smile(n)=(((err')*err)/31266);% Calculating Mean Square Error.
    % Displaying the reconstruction process as the number of PCs increases.
    % Uncomment the line below to see the reconstruction process.
    %imshow((reshape(uint8(rec_img),[162,193]))'+psi_img);
end
figure, plot(mse_smile);
% title('MSE Smiling Face')
title('MSE varaiation plot for Smiling Face as No of PCs are varied');
xlabel('No. of Principal Components');
ylabel('Mean Square Error');
figure,subplot(1,2,1);
imshow(imread(strcat('107b.jpg')));
title('Original Image');
subplot(1,2,2);
imshow((reshape(uint8(rec_img),[162,193]))'+psi_img); %Adding the average
vector back and displaying the reconstructed image
title('Reconstructed Image with 190 PCs');

%% MSE for normal image in the test set

imge=double(imread(strcat('192a.jpg')));
imge=reshape(imge',[31266,1]);
w1=[];
% figure;

for n=1:190
% n=70;
Vmax=V(:,1:n); %getting the correspoding "orthonormal Eigenvectors" or PCs
    eignVect_phi_Max=A*Vmax;
    % Now I normalized the calculated Eigenvectors, else some pixel values
    % exceed 255 and as a result proper reconstruction of the image doesn't
    % occur.
    for i=1:n

eignVect_phi_Max(:,i)=eignVect_phi_Max(:,i)/norm(eignVect_phi_Max(:,i));
    end
    % Calculating the weight vector w where each element of w represents
    % the contribution of each eigenface vector to the reconstructed image.
    w1= (eignVect_phi_Max')*(imge-psi);
    rec_img=(eignVect_phi_Max*w1);% Reconstructing the image
    err=imge-(rec_img+psi);% Calculating the error w.r.t original image
    mse_norm_testset(n)=(((err')*err)/31266);% Calculating Mean Square Error.
   % Displaying the reconstruction process as the number of PCs increases.
    % Uncomment the line below to see the reconstruction process.
%     imshow((reshape(uint8(rec_img),[162,193]))'+psi_img);
```

```
%    title('Reconstructed image')
end
% subplot(1,2,1);
figure, plot(mse_norm_testset);
title('MSE varaiation plotfor Normal face from Test Set');
xlabel('No. of Principal Components');
ylabel('Mean Square Error');
figure,subplot(1,2,1);
imshow(imread(strcat('192a.jpg')));
title('Original Image');
subplot(1,2,2);
imshow((reshape(uint8(rec_img),[162,193]))'+psi_img); %Adding the average
vector back and displaying the reconstructed image
title('Reconstructed Image with 190 PCs');

%% Reconstructing Non-Human image (image of flower Vase) with all the
eigenvectors/PCs

imge=double(imread('vase1.jpg'));
imge=double(reshape(imge',[31266,1]));
% The next loop is to make sure the eigenvectors are normalized.
% Although not necessary, just a precaution
for i=1:190
      eignVect_A(:,i)=eignVect_A(:,i)/norm(eignVect_A(:,i));
end
   % Calculating the weight vector w where each element of w represents
   % the contribution of each eigenface vector to the reconstructed image.
w_test=(eignVect_A')*(imge-psi); % Weights

% Reconstruction
rec_img=uint8(eignVect_A*w_test); % Reconstructing the image
q=(reshape(uint8(rec_img),[162,193]))'+psi_img; % Resizing the image, adding the
average vector.

% Displaying the images
figure,subplot(1,2,1);
imshow(imread(strcat('vase1.jpg')));
title('Original Image');
subplot(1,2,2);
imshow(q);
title('Reconstructed Image with 190 PCs');
%% Reconstructing Rotated Image by rotating it to different angles and using all
the PCs.
%% Here I rotated the image by 1 degree anticlockwise every time and plotted the
MSE curve.
% I have also used this code snippet to also manually put values of angle of
% rotation and get the reconstructed image.

% The next loop is to make sure the eigenvectors are normalized.
% Although not necessary, just a precaution
for i=1:190
      eignVect_A(:,i)=eignVect_A(:,i)/norm(eignVect_A(:,i));
end
%%
%  for k=1:360 %% uncomment this line, line 222,230,231,233,235and 238 to 241.
       % to get MSE plot for every 1 degree rotation of the input image.
%     J=imrotate(imread('120a.jpg'),k,'bilinear','crop');
```

```
    J=imrotate(imread('120a.jpg'),355,'bilinear','crop'); % rotating and
cropping the image.
    imge=double(reshape(J',[31266,1])); %Resizing the image

    w_test=(eignVect_A')*(imge-psi); % Calculating Weights

% Reconstruction
    rec_img=((eignVect_A*w_test)); % Reconstructing the image
%     err=imge-(rec_img+psi); % Calculating the error w.r.t original image
%     mse_rotate(k+1)=(((err')*err)/31266); % Calculating Mean Square Error.
    q=(reshape(uint8(rec_img),[162,193]))'+psi_img; % Resize the image and
adding the average vctor
%     imshow(q); % show the live plot.

%  end
 %%
% figure,plot(mse_rotate)
% title('MSE Plot for rotated imaage (0-360)');
% xlabel('Input image rotation angle ( in Degrees) ')
% ylabel('Mean Squared Error');
 %%
figure,subplot(1,3,1);
imshow(imread(strcat('120a.jpg')));
title('Original Image');
subplot(1,3,2);
imshow(J);
title('Rotated by 355 degree anticlockwise');
subplot(1,3,3);
imshow(q);
title('Final Rotated image reconstruction');
```