

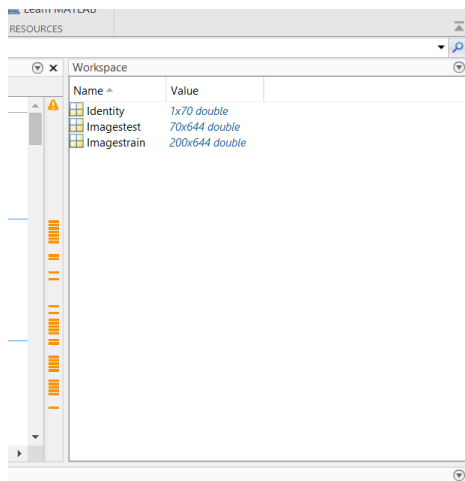
# Machine Learning for Visual Data Analysis – Face Recognition using Eigenfaces

## Description

1. **Read in the training and test images. The two functions for loading the training and test image files have been provided in separate “.m” files**

“loadImagesInDirectory.m”: The method follows the path for training images and loads, flattens, and returns all 200 images in a 200 by 644 matrix (flattening 23x28 image).

“loadTestImagesInDirectory.m”: This method loads, flattens and returns all 70 images in a matrix of size 70 by 644 (called "Imagetest" in the assignment). It also produces a one by 70 vector called "Identity", which holds the face label for each image.



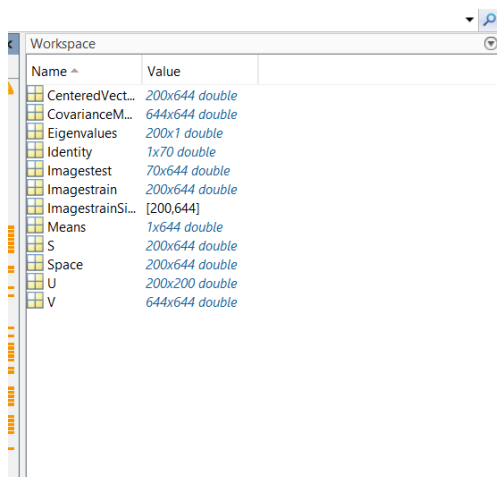
The screenshot shows the MATLAB Workspace window with the following variables:

Name	Value
Identity	1x70 double
Imagetest	70x644 double
Imagetrain	200x644 double

Workspace output

2. **Construct the mean image and the covariance matrix of the training image set. Code is provided in lab2.m file.**

Using the code provided in lab2.m, a covariance matrix (644x644) was created. It computes the means of the training images (using the 200x644 variable "Imagetrain"). It computes the covariance matrix for the training images by subtracting the mean from the images (CenteredVectors) (has a size of 644 x644).



The screenshot shows the MATLAB Workspace window with the following variables:

Name	Value
CenteredVect...	200x644 double
CovarianceM...	644x644 double
Eigenvalues	200x1 double
Identity	1x70 double
Imagetest	70x644 double
Imagetrain	200x644 double
ImagetrainSi...	[200,644]
Means	1x644 double
S	200x644 double
Space	200x644 double
U	200x200 double
V	644x644 double

Workspace output

### 3. Compute the Eigenface of the training set. Code has been provided in lab2.m

Eigenvalues are listed in the diagonal entries of the 200x644 matrix, "S," and corresponding eigenvectors for the eigenvalues are contained in the matrix, "V." To represent the dataset in terms of its eigenpairs, Singular Value Decomposition is applied to a normalised training dataset.

*Note: The workspace output is same as the one obtained in Section 2.*

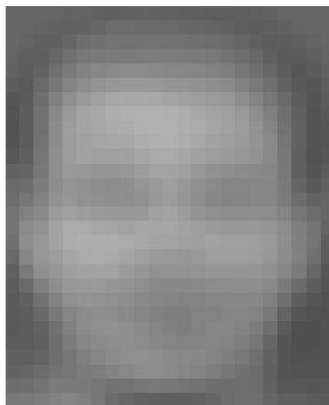
### 4. Display the mean image. Code has been provided in lab2.m

With the template code provided in the lab2.m , a Mean Image is generated with the output as shown below.

Output:

Workspace output

**Mean Image**



Workspace	
Name ▲	Value
CenteredVect...	200x644 double
CovarianceM...	644x644 double
Eigenvalues	200x1 double
Identity	1x70 double
Imagetest	70x644 double
Imagetrain	200x644 double
ImagetrainSi...	[200,644]
k	643
MeanImage	28x23 uint8
Means	1x644 double
S	200x644 double
Space	200x644 double
U	200x200 double
V	644x644 double

### 5. Display the first 20 Eigenfaces. You need to write this part of the code

Hint: take a look at the code for displaying the mean face in Step 4.

In this section there is displaying of the first 20 eigenfaces. The eigenvectors of the largest 20 eigenvalues describe the first 20 eigenfaces. The Singular Value Decomposition (SVD) method in MATLAB automatically arranges eigenpairs in descending eigenvalue order. The first 20 entries of the matrix "Space" will hold the corresponding eigenvectors of the largest 20 eigenvalues via matrix operations. The code provided below has been written to reshape the flattened image back to its original shape.

Code:

```
%% Display of the 20 first eigenfaces : Write your code here
%Jahnvi's__code
EigenFace = zeros(1, 644); % create a flat col matrix with 0s
EFMatrix = S*V'; % using SVD V compute EF matrix
```

```
%normalization eigenfaces for better visualisation
EFMatrix = 255 *(EFMatrix - min(EFMatrix(:))) ./ (max(EFMatrix(:)) - min(EFMatrix(:)));
```

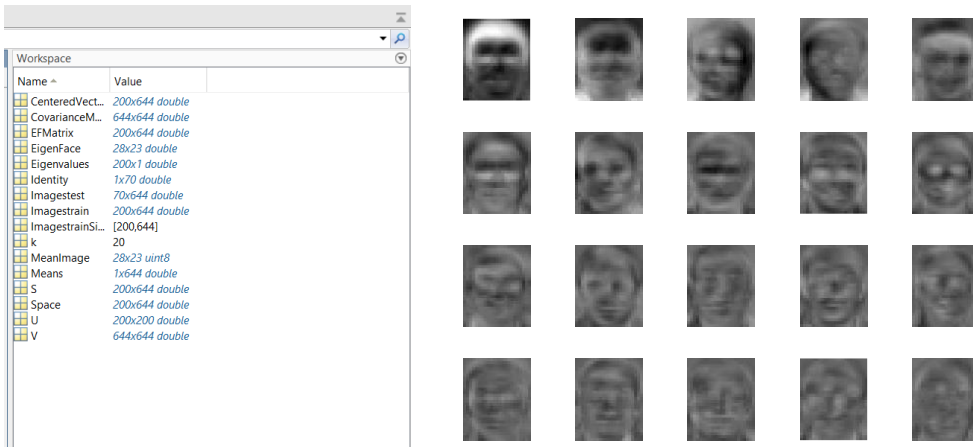
```
for k = 1:20 % iterate over and plot plot
```

```

EigenFace = EFMatrix(k,:);
EigenFace = reshape(EigenFace, [28,23]);
subplot (4,5,k);
imshow(uint8(EigenFace));
end

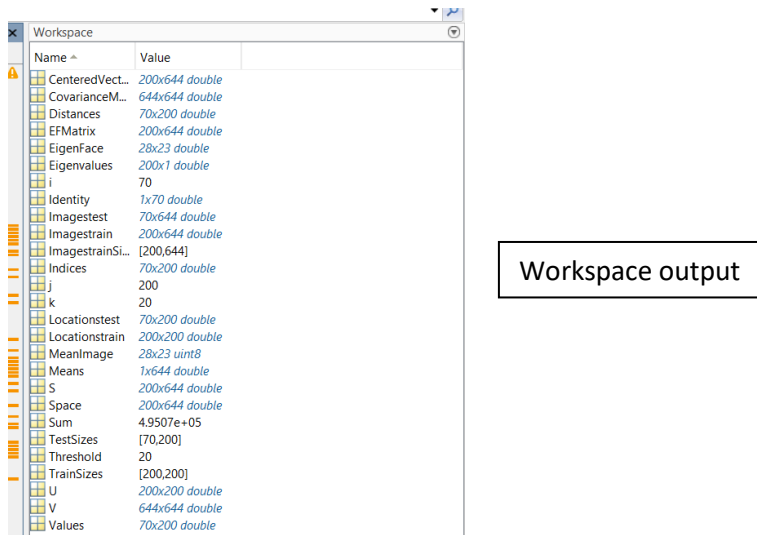
```

Output:



**6. Project both training images and testing images onto the first 20 Eigenfaces. The function for the project has been provided.**

The given template code was used to project both training and test images onto 20 Eigenfaces. It projects training and testing datasets using means and eigenvectors extracted from the training dataset using a function called "projectImages.m." To project the images onto the given eigenspace, in the code there is subtraction of the mean from the given dataset (train or test) and performance of matrix multiplication with the training dataset's eigenvectors.



**7. Compute the distance from the project test images to the projected training images**


The template code provided computes the distance between projected test images and projected train images was calculated in the following section. The task is handled by two separate for-statements. The first two for-loops compute the Euclidean distance between each test and training sample one by one. The third, innermost loop computes the distance

between training and testing projections using the first "k" eigenfaces, which for this question are set to 20. The second loop sorts the best matching training images for later use with the test images. It is accomplished by sorting based on the distances between training and testing projections.

**8. Display the top 6 best matched training images for each test image. Code has been provided in lab2.m**

On the right side, the best matching training image with 20 eigenfaces is displayed, along with the projected image index. The test and training images are visually very similar. The test image is labelled by the training image label that is closest to the test image projection, using 1-nearest neighbours. Displaying the top 6 best-matched images:

Output:

Image tested 	Image recognised with 20 eigenfaces:85 
Image tested 	Image recognised with 20 eigenfaces:13 
Image tested 	Image recognised with 20 eigenfaces:18 
Image tested 	Image recognised with 20 eigenfaces:24 
Image tested 	Image recognised with 20 eigenfaces:200 
Image tested 	Image recognised with 20 eigenfaces:29 

Name	Value
CenteredVect...	200x644 double
CovarianceM...	644x644 double
Distances	70x200 double
EFMatrix	200x644 double
EigenFace	28x23 double
Eigenvalues	200x1 double
6	
Identity	1x70 double
Image	28x23 uint8
Imagerec	28x23 uint8
Imagetest	70x644 double
Imagetrain	200x644 double
ImagetrainSi...	[200,644]
Indices	70x200 double
j	200
k	643
Locationtest	70x200 double
Locationtrain	200x200 double
MeanImage	28x23 uint8
Means	1x644 double
S	200x644 double
Space	200x644 double
Sum	4.9507e+05
TestSizes	[70,200]
Threshold	20
TrainSizes	[200,200]
U	200x200 double
V	644x644 double
Values	70x200 double
x	6
y	2

**9. Compute the recognition rate using 20 Eigenfaces. Write your own code here.**

The obtained recognition rate is 82.8571%.

Code:

```
% recognition rate compared to the number of test images: Write your code
here to compute the recognition rate using top 20 eigenfaces.
%Jahnvi's__code
% RR is for 20 indices of eignefaces.
% if traon indices are not equal to test Identity then RR = 0

RR = zeros(1,length(Imagetest(:,1))); % RR is recognition rate is of
length of imagetest
for i = 1: length(Imagetest(:,1))
    if ceil(Indices(i,1)/5) == Identity(i)
        RR(i) = 1;
    else
        RR(i) = 0;
    end
end
% The total recognition rate for the whole test set that has 70 images
NETRR = sum(RR)/70 *100;
```

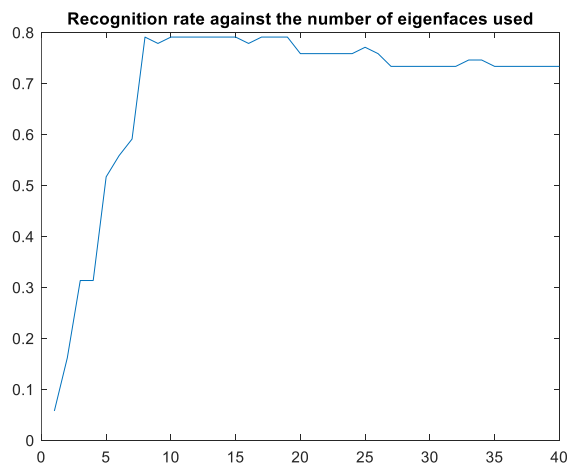
## Output:

Workspace	
Name	Value
CenteredVect...	200x644 double
CovarianceM...	644x644 double
Distances	70x200 double
EFMatrix	200x644 double
EigenFace	28x23 double
Eigenvalues	200x1 double
i	70
Identity	1x70 double
Image	28x23 uint8
Imagerec	28x23 uint8
Imagetest	70x644 double
Imagetrain	200x644 double
ImagetrainSi...	[200,644]
Indices	70x200 double
j	200
k	643
Locationtest	70x200 double
Locationtrain	200x200 double
MeanImage	28x23 uint8
Means	1x644 double
NETRR	82.8571
RR	1x70 double
S	200x644 double
Space	200x644 double
Sum	4.9507e+05
TestSizes	[70,200]
Threshold	20
TrainSizes	[200,200]
U	200x200 double
V	644x644 double
Values	70x200 double
x	6
y	2

## **10. Investigate the effect of using different number of Eigenfaces for recognition (e.g. plot the recognition rate against number of Eigenfaces). Code has been provided in lab2.m**

We can deduce from the graph below that the number of  $k$  required is between 10 and 20. The recognition rate rises until  $k = 10$  and then plateaus around  $k$ : 10-to-20 before dropping. This means that we only need 20 eigenfaces to fully represent the entire training set. The number of Eigenfaces is plotted against the average recognition rate as shown below:

## Output:



## **11. Investigate the effect of K in K-Nearest Neighbour (KNN) classifier. Plot the average recognition rate against K. You need to write your own code here.**

This code segment computes and displays the effect of using various KNN values. For each of the testing samples, a for-loop can be executed.

## Code:

```

%% effect of K: You need to evaluate the effect of K in KNN and plot the
recognition rate against K. Use 20 eigenfaces here.
%KNN evaluation
TrainingLabels = []; % create training labels
for i = 1:40
    TrainingLabels = horzcat(TrainingLabels, repmat(i,1,5));
end
%% Recognition rate against different k values

NETRR = zeros(1,200);
K=1:200; % value of k in K nearest neighbour

for k = 1:200
    KNNModel = fitcknn(Imagestrain, TrainingLabels, 'NumNeighbors',
k, 'BreakTies', 'nearest'); %fit knn model to training data
    KNNPredict = predict(KNNModel, Imagestest); % make prediction on test
data
    KNN_RR = zeros(1, length(Imagestest(:,1))); % initialise recognition
rate

    for i = 1:length(Imagestest(:,1))
        %compare the predictions with identity of test image and predicted
        if ceil(Indices(i,1)/5) == KNNPredict(i)
            KNN_RR(i) = 1;
        else
            KNN_RR(i) = 0;
        end
    end

    NETRR(k) = ((sum(KNN_RR)/70)*100);
end
figure
plot(K, NETRR);
xlabel('K'); ylabel('Recognition rate')
title('Recognition rate against different k values used');

```