Université de Bourgogne

Applied Mathematics

Project Report

Face Recognition using PCA

Authors:
Tewodros W. AREGA
Manju Kumar BASAVARAJ
Olivier SEHOU AGBOHOUI

Supervisor:
Prof. Désiré Sidibé

January 13, 2019

Contents

1 Introduction

One of the simplest and most effective PCA approaches used in face recognition systems is the so-called eigenface approach. This approach transforms faces into a small set of essential characteristics, eigenfaces, which are the main components of the initial set of learning images (training set). Recognition is done by projecting a new image in the eigenface subspace, after which the person is classified by comparing its position in eigenface space with the position of the training eigenfaces.[1, 2]

2 Methodology

2.1 Normalization

The images are normalized to account for scale, orientation and location variations. Each image's facial features is run through an iterative scheme. Using affine transformations, we compare the facial features of the image with predefined values of facial features on a 64 x 64 window.
The facial features under consideration are (1) right eye, (2) left eye, (3) nose, (4) left mouth corner and (5) right mouth corner.
The main facial features of each image are mapped to predetermined values of facial features using affine transformation matrix. The predetermined positions of these features in a 64 x 64 window are defined as follows:

- Right eye (13,20)
- Left eye (50,20)
- Nose (34,34)
- Left mouth corner (16,50)
- Right mouth corner (48,50)

2.1.1 Reading the Data

Using dir matlab command we find all the .jpg and .txt files in the directory. In a for loop, the images are read into a matrix using imread matlab function. In a second for loop, the text files are read using load matlab function.

2.1.2 Finding and applying the affine transformation

The affine transformation matrix can be defined as : $\begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix}$

To ensure the dimensions match for matrix multiplication, we have included a parameter in the Z direction which has value 1 for all facial features throughout all images. We have 10 equations (X and Y values of five features) and 6 unknowns(a11, a12, a21, a22, b1, b2).

Using the matlab command pinv(I), we find the values of the six unknowns in the affine matrix by the method of least squares. This transformation is applied to the facial features.

2.1.3 Procedure to find Ft

Initially, we set Fbar as the values of F1(features of first image).

We run the algorithm until the error between two consecutive average facial features is lesser than a threshold, say $10^{-}6$. We get the final average facial features.

Below is the flowchart of the algortithm to calculate the average facial features of all the images.

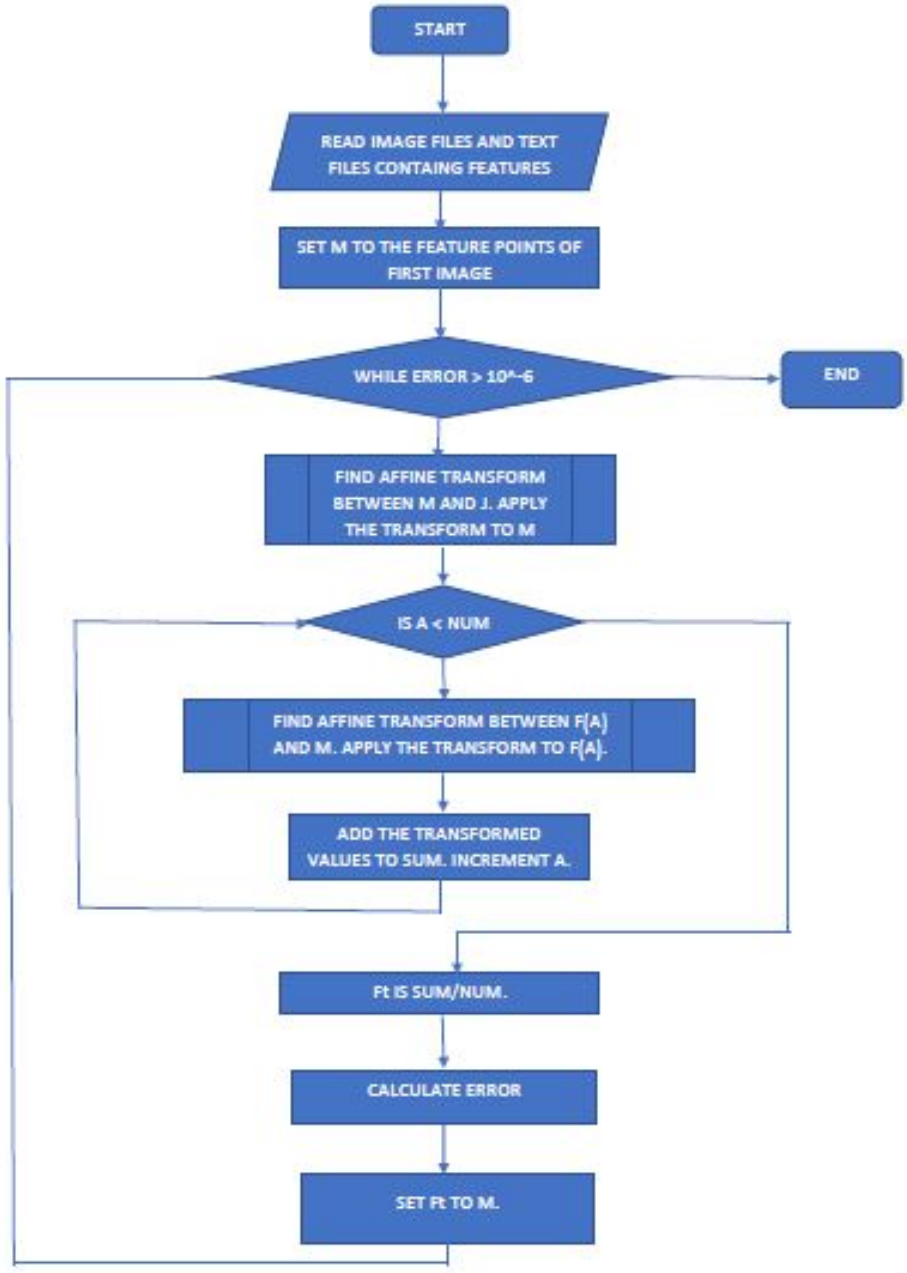

Figure 1: Flowchart of Normalization of Facial Features

2.1.4 Applying Normalization on the Images

Now, for each image we find the transform between the facial features of the image to the final average facial features. Using this transform, we apply it on the image to convert the image to a 64 x 64 window with robust orientation. This final set of normalized images are divided into two directories, ie "train_images" and "test_images".
The above obtained data will be used throughout the rest of the procedure.

2.2 Recognition using PCA

Recognition involves the following steps:

2.2.1 Prepare the dataset

First we convert each normalized image from $M * N$ image into $MN * 1$ vector. This conversion is done by concatenating the rows of the image:

$$V_i = [I_i(1,1), I_i(1,2), ..., I_i(1,N), ..., I_i(M,1), I_i(M,2), ..., I_i(M,N)]$$

where $I_i(x,y)$ is the pixel intensity at position (x, y) in I. Then the training set matrix D will be like this:

$$D = \begin{bmatrix} I_1(1,1) & I_1(1,2) & \dots & I_1(1,N) & \dots & I_1(M,1) & I_1(M,2) & \dots & I_1(M,N) \\ I_2(1,1) & I_2(1,2) & \dots & I_2(1,N) & \dots & I_2(M,1) & I_2(M,2) & \dots & I_2(M,N) \\ & & \vdots & & & & \vdots & & \\ I_p(1,1) & I_p(1,2) & \dots & I_p(1,N) & \dots & I_p(M,1) & I_p(M,2) & \dots & I_p(M,N) \end{bmatrix}_{p \times d}$$

Figure 2: Matrix D

In our case, we have 99 training images of size 64*64, therefore matrix D will be 99*4096 matrix.

2.2.2 Compute the Average Vector

The next step is to compute the mean of the vectors of matrix D.

$$mean = \frac{1}{p} \sum_{i=1}^{p} I_i$$

2.2.3 Subtract the mean

The images are mean centered by subtracting the mean image(mean of matrix D) from each image vector.

$$I_centered = I_i - mean$$

After that we created a new matrix M that consists of $I_centered$ vectors.

2.2.4 Calculate the Covariance matrix

To calculate the Covariance matrix C:

$$C = \frac{1}{p-1} M^T M$$

But the Covariance will be 4096*4096 matrix, as a result will take a lot of time to compute its eigenvalues and eigenvectors. So, we also calculated C':

$$C' = \frac{1}{p-1} M M^T$$

which is matrix of 99*99. We used both options but the latter one is fast.

2.2.5 Calculate the eigenvectors and eigenvalues of the Covariance matrix

Here we caluclate the eigenvectors and eigenvalues of C'. Then, we keep the k largest eigenvectors as a columns of projection matrix P'. To get the projection matrix of the covariance matrix C:

$$P = M^T P'$$

2.2.6 Eigenspace

Each of the image vectors V_i are projected into eigenspace using:

$$E_i = V_i P$$

Then we create a matrix F that stores the feature vectors E_i of each images.

2.2.7 Read Test Image

Read the test image I_t and convert it to vector T_t. Then we fellow the same procedure to the test image the same as the training image. Thus we compute the feature vector of T_t using:

$$F_t = T_t * P$$

and we get the feature vector F_t of the test image.

2.2.8 Calculate Euclidean Distance

Compare the feature vector of test image with the feature vector of all training images by computing the Euclidean distance between them. Then, the feature vector which has minimum distance from the test image′s feature vector will be the best match.

2.2.9 Calculate Accuracy

Finally, we calculate the accuracy using the following formula:

$$accuracy = (1 - \frac{\epsilon}{totalnumberoftestimages}) * 100$$

Where ϵ is the error count which is incremented if no match is found for the test image.

3 Description of Experiments

Firstly, we read the facial features data of each image and calculate the average locations of each facial feature. The features of each image are compared with Fbar which is compared to the predetermined features. We find the affine transformation on each comparison and apply it onto the image. After getting the average location of all the facial features, we normalize the images and crop it into a 64 x 64 window. These images are stored under a directory containing all the normalized images. The above step takes care of variations such as orientation, scaling and translation.

After we get the normalized images of size 64*64, we transformed each training image into a row vector of 1*4096 then concatenate each row into a matrix D. Then we compute the mean of the vectors of matrix D and

subtracted the mean from matrix D and calculate the covariance matrix which will be 4096*4096. After that, we calculate the eigenvector of the covariance matrix and select the k largest eigenvectors to be the columns of the projection matrix (4096 by k). We tested the system with different values of k. We transformed each face vector into the PCA space by multiplying the face vector by projection matrix. Then we stored all the feature vectors of the training image in the feature matrix.

The final part is recognition. In recognition, first we read the test image, then convert it to row vector. After that we transformed it to PCA space by multiplying the row vector with the transformation matrix . Finally, we compare the distance between the feature vector of the test image and all the training images feature vectors and the one which has the minimum distance is the best match for the test image.

4 Graphical User Interface

After implementing the face recognition algorithm, we created a graphical user interface using GUIDE.

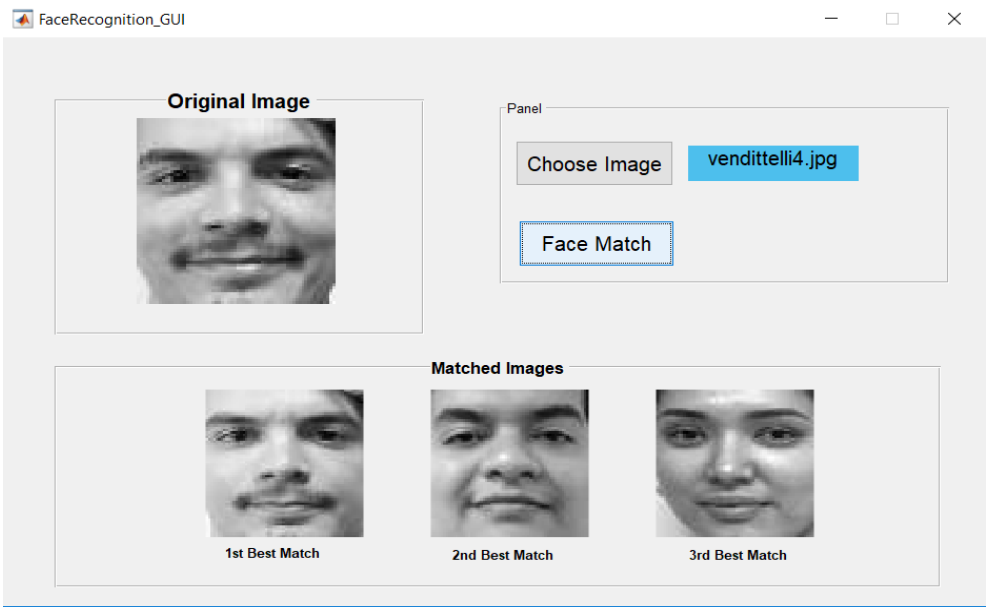

Figure 3: Face Recognition GUI

As you can see from **figure 2**, the GUI has two buttons. The first one **"Choose Image"** button, which help us to choose the test image. After the test image is selected, its file name will be displayed in static text positioned on the right of the button. The **"Face Match"** button help us to match the test data with the training data. Then, the first three best matches, which has 3 least minimum distances are displayed on the three boxes.

5 Result and Discussion

5.1 Result

After we implement the system, the accuracy that we get using the testing images is 62.1212% by setting the *k value* to be 80. As you can see from the images below, "Original image" which is on the top left corner is the test image, "1st Matched Image" which is on the top right corner is the best matched image to the test image. "2nd Matched Image" is second best matched image according to the Euclidean distance. "3rd Matched Image" is third best matched image according to the Euclidean distance.
Here are two test images which are correctly matched(k-value is 80):

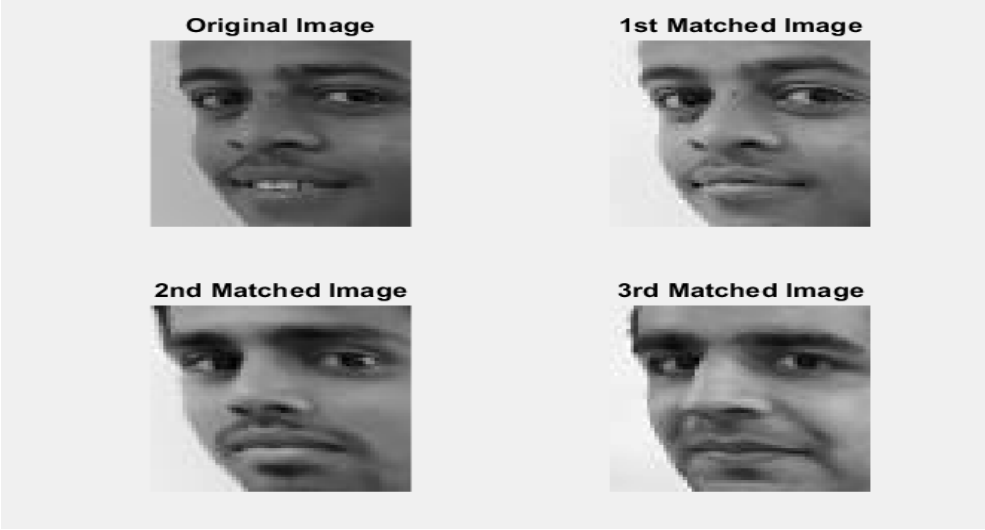

Figure 4: Face Recognition Result: Correctly Matched

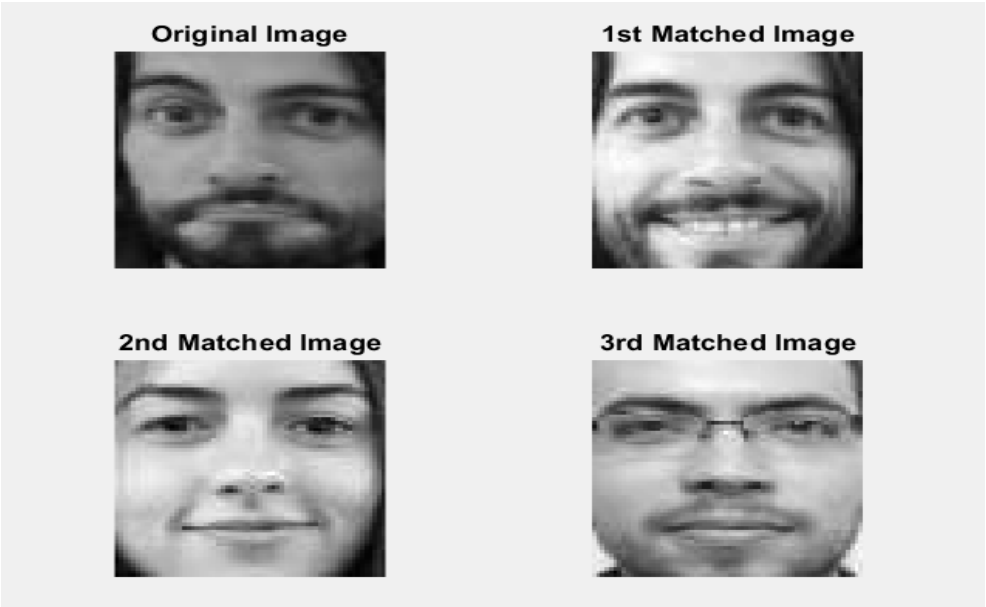

Figure 5: Face Recognition Result: Correctly Matched

Here are two test images which are wrongly matched(k-value is 80):

Figure 6: Face Recognition Result: Wrongly Matched

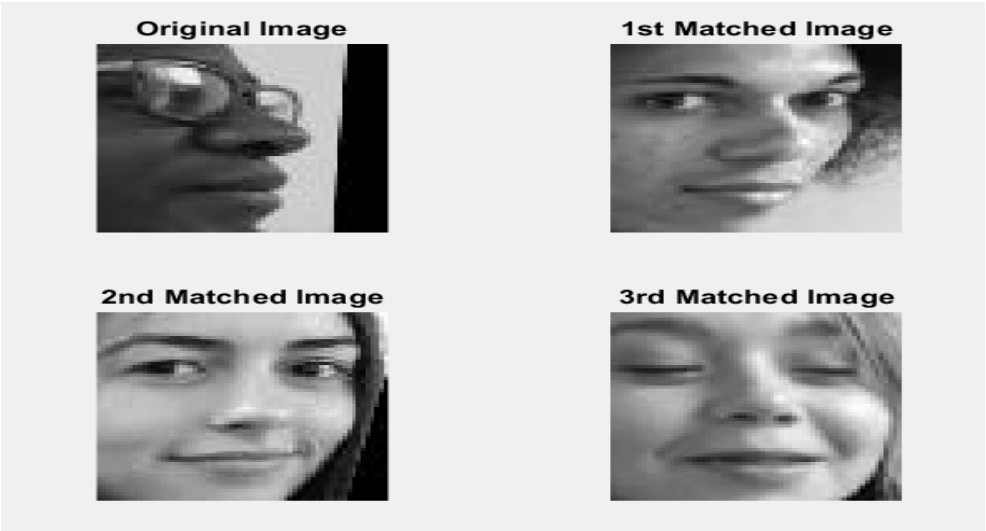

Figure 7: Face Recognition Result: Wrongly Matched

We have also tested the system by **varying the k values**. Here are the results:

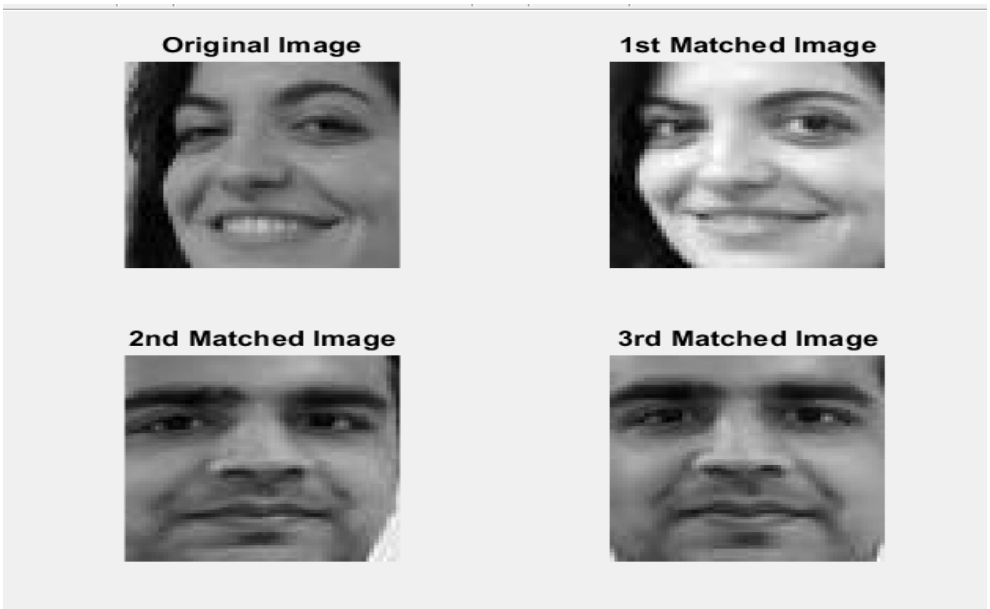

Figure 8: Face Recognition Result: K-value 80

The first is tested with k-value of 80.

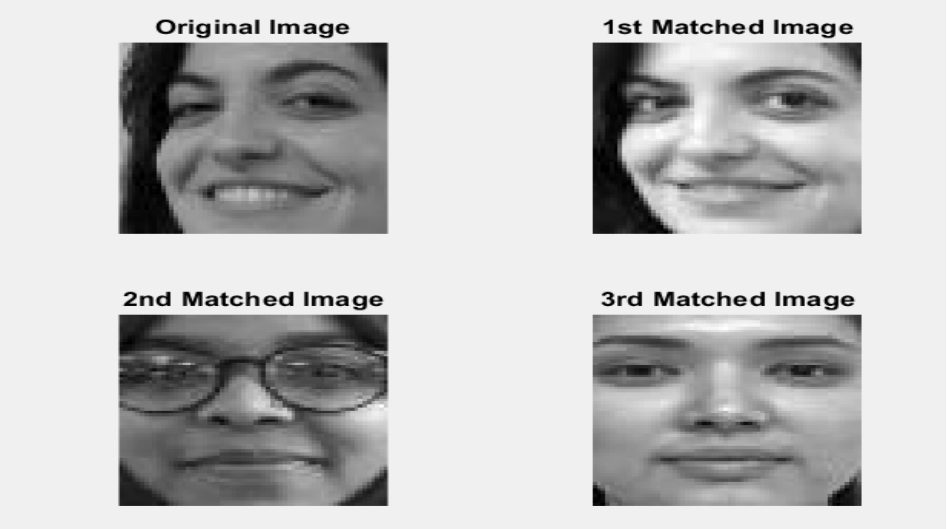

Figure 9: Face Recognition Result: K-value 40

The second one is tested with k-value of 40.

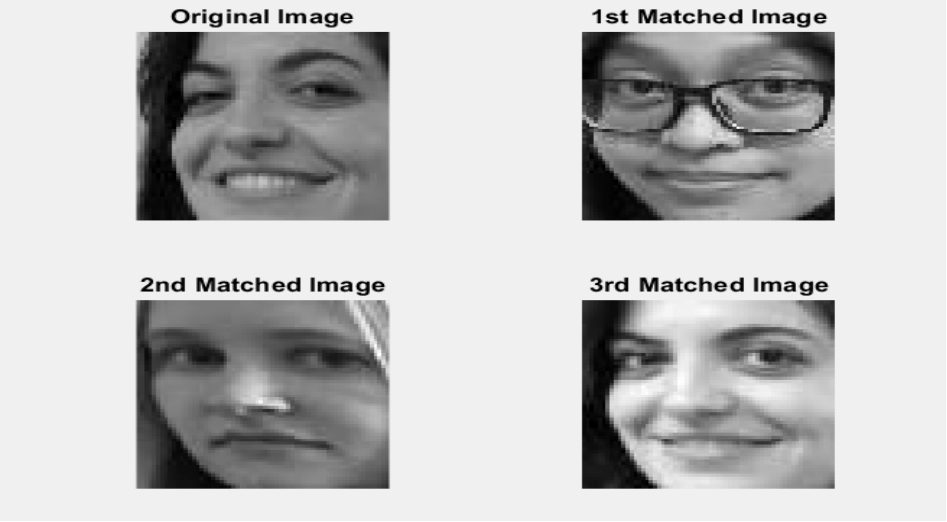

Figure 10: Face Recognition Result: K-value 10

The third one is tested with k-value of 10.

5.2 Discussion

From the results that we have got, the accuracy of the system for **k-value 80** is **62.1212%**. As we decreased the value of k, the accuracy of the system also decreased. When **k-value is 40**, the accuracy decreased into **59.0909%** and when **k-value is 10** the accuracy decreased into **50%**.

Here is a graph of the accuracy of the system for different K-values:

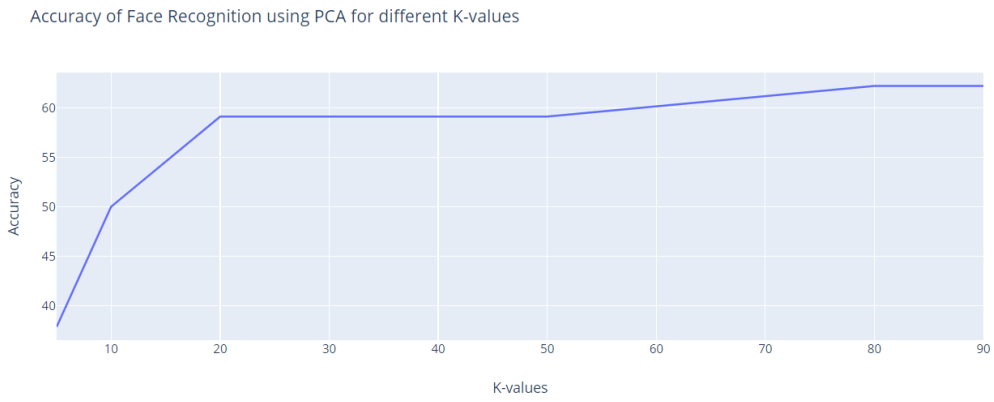

Figure 11: Accuracy of Face Recognition System

6 What We Learn

By doing this project, we have learned a lot. To mention some: • We learned how to normalize data
• We learned how to find and use affine transformation to modify data.
• We learned one application of PCA which is applying PCA on faces to reduce the dimension and use it to recognize the faces by computing the distance
• We learn how to use various Matlab functions to read, store and manipulate data into files.
• We also learned how to create a GUI in matlab using GUIDE

7 Codes

7.1 Normalization

```
function [A]=normal

% Reading image files from the directory
file1=dir('*.jpg');
num=length(file1);
for n=1:num
  img{n} = imread(file1(n).name);
end

% Reading text files that contain the facial features
file=dir('*.txt');
num=length(file);
for n=1:num
    f{n} = load(file(n).name);
end

% J is predetermined facial features
J=[13 20; 50 20; 34 34; 16 50; 48 50];
% M is Fbar
M=f{1};
flag=true;
ft=zeros(5,2);
X=0;

while flag
    fto=ft;
%   findt function finds and applies the affine transformation
    A=findt(M,J);
    M=A;
    sum=zeros(5,2);
%   For each image we compare it with Fbar and apply the transformation
    for a=1:num
        A=findt(f{a},M);
```

```
        sum=sum+A;
    end
%   ft is the final average facial features
    ft=sum/num;
%   Checking for error between two consecutive ft, if lesser than 10^-6
%   stop running the while loop
    if max(max(abs(ft-fto)))<0.000001
        flag=false;
    end
    M=ft;
    X=X+1;
end

%  Image normalization
for a=1:num
    q=rgb2gray(img{a});
    t=transform(f{a},ft);
%   transform finds the affine transform only
    tran=maketform('affine',t);
%   Image is normalized and cropped to 64 x 64 window
    image=imtransform(q,tran,'XData',[1 64],'YData',[1 64]);
%   Saving the normalized image
    path=strcat('./normalized/',file1(a).name);
    imwrite(image,path,'jpg');
end


end
```

7.2 Finding the Transform function

```
function [P]=findt(F,J)
    A=ones(5,1);
    F=([F A]);
    J=([J A]);
    X=pinv(F)*J;
    P=F*X;
```

```
    P=P(:,1:2);
end



function [P]=transform(F,J)
    A=ones(5,1);
    F=([F A]);
    J=([J A]);
    P=pinv(F)*J;
    P=P(:,1:2);
    A=[0;0;1];
    P=[DP A];
end
```

7.3 Implementation of PCA Algorithm

```
% Implementation of PCA Algorithm
function [D_old, Phi, Ft] = pca_algorithm(kval)


%Create training data matrix D
faceImgs = dir('train_images\*.jpg');
numfiles = length (faceImgs);

% kval = 80; %
d = 64*64; %size of the image
p = numfiles; %no of images
D = zeros(p,d);

trainpath = 'train_images\';
for k = 1: numfiles
  str = faceImgs(k).name;

  trainimgname = strcat(trainpath,str);
  Iface = imread (trainimgname);
```

```
%    Iface = rgb2gray(Iface);
   X1 = reshape(Iface',1,[]); %convert it to row vector
   D(k,:) = X1;
end

%calculate the mean of D
Xmean = mean((D'));
Xmean = Xmean';

D_old = D;
D = D - Xmean;

%covariance matrix
Cov_1 = (D' * D)./(p-1);

%Eigen values and eigenvectors
[EgVec,DiagV] = eig(Cov_1);

%sort them
[Dvalues, ind] = sort(diag(DiagV),'descend');
EgnVector = EgVec(:,ind);

% k principal components
PComp = EgnVector(:,1:kval);

% projection matrix(d by k), phi
Phi = PComp;

% change each image to pca space
% feature vectors
Ft = zeros(p,kval);

for m=1:p
    x1 = D_old(m,:);
    f1 = x1 * Phi;
    Ft(m,:) = f1;
end
```

```
end
```

7.4 Testing all test images and calculating the accuracy

```
% Testing all test images and calculating the accuracy
%-------------------------------------------------------------------
pcaerror = 0;
faceImgs_ = dir('test_images\*.jpg');
numfiles_ = length (faceImgs_);
lbl = 1;
oldFN='';
kvalue=30
[Dmatrix, projMatrix, featureM] = pca_algorithm(kvalue);
trainpath_ = 'test_images\';
for k = 1: numfiles_
    str_ = faceImgs_(k).name;
    testingimgname = strcat(trainpath_,str_);

    % labeling the images
    lenstr = length(str_);
    familyIndex = str2num(str_(lenstr-5:lenstr-4));

    familyName = str_(1:lenstr-5);
    if(k == 1)
        oldFN = familyName;
    elseif(length(oldFN) == length(familyName))
      if(oldFN ==  familyName)

      else
          lbl = lbl + 1;
      end
    else
      lbl = lbl + 1;
    end
```

```
oldFN = familyName;
I_test = imread (testingimgname);

%-------------------------------------------
x_test = double(reshape(I_test',1,[]));

%calculate the feature vector of test
f_test = x_test * projMatrix;
p=99; % total number of train images
%compute the Euclidean distance
Eucl_distance = ones(p,1);

for n=1:p
    f_training = featureM(n,:);
    Eucl_distance(n) = norm(f_training - f_test);
end

%find the minimum distance
[sortedDist, indexSort] = sort(Eucl_distance);
indexMin = indexSort(1); %index of minimum distance

V_matched = Dmatrix(indexMin,:);
I_matched = vec2mat(V_matched,64);

V_matched2 = Dmatrix(indexSort(2),:);
I_matched2 = vec2mat(V_matched2,64);

V_matched3 = Dmatrix(indexSort(3),:);
I_matched3 = vec2mat(V_matched3,64);
%-------------------------------------------
% check if it matches
modv = mod(indexMin,3);
if(modv == 0)
   matchInd = indexMin/3;
elseif(modv == 1)
    matchInd = (indexMin+2)/3;
elseif(modv == 2)
    matchInd = (indexMin+1)/3;
```

```
    end


    if(lbl ~= matchInd)
        pcaerror= pcaerror + 1;
    end
    Display the figure
    figure,
    subplot(2,2,1), imshow(I_test)
    title('Original Image')
    subplot(2,2,2), imshow(I_matched,[])
    title('1st Matched Image')
    subplot(2,2,3), imshow(I_matched2,[])
    title('2nd Matched Image')
    subplot(2,2,4), imshow(I_matched3,[])
    title('3rd Matched Image')
end
pca_accuracy = (1 - double(pcaerror/numfiles_))*100;
pca_accuracy
```

7.5 Connection to GUI

```
%Connection to GUI
function [I_matched,I_matched2, I_matched3,indexMin]=connectPCA(I_test)
    kvalue = 80;
    [Dmatrix, projMatrix, featureM] = pca_algorithm(kvalue);
    x_test = double(reshape(I_test',1,[]));

    %calculate the feature vector
    f_test = x_test * projMatrix;
    p=99; % total number of train images
    %compute the Euclidean distance
    Eucl_distance = ones(p,1);

    for n=1:p
        f_training = featureM(n,:);
        Eucl_distance(n) = norm(f_training - f_test);
```

```
    end

    %find the minimum distance
    [sortedDist, indexSort] = sort(Eucl_distance);
    indexMin = indexSort(1); %index of minimum distance

    % indM = find(Lt == indexMin);

    V_matched = Dmatrix(indexMin,:);
    I_matched = vec2mat(V_matched,64);

    V_matched2 = Dmatrix(indexSort(2),:);
    I_matched2 = vec2mat(V_matched2,64);
    % I_matched3 = I_matched;
    V_matched3 = Dmatrix(indexSort(3),:);
    I_matched3 = vec2mat(V_matched3,64);
end
```

7.6 Face Recogninition GUI

```
function varargout = FaceRecognition_GUI(varargin)
%   FACERECOGNITION_GUI MATLAB code for FaceRecognition_GUI.fig

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
%FaceRecognition_GUI

% Last Modified by GUIDE v2.5 27-Dec-2018 17:55:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FaceRecognition_GUI_OpeningFcn, .
                   'gui_OutputFcn',  @FaceRecognition_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

```
                   'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before FaceRecognition_GUI is made visible.
function FaceRecognition_GUI_OpeningFcn(hObject, eventdata, handles, va
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FaceRecognition_GUI (see VARARGIN

% Choose default command line output for FaceRecognition_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes FaceRecognition_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = FaceRecognition_GUI_OutputFcn(hObject, eventdata, h
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in chooseImageBtn.
function chooseImageBtn_Callback(hObject, eventdata, handles)
% hObject    handle to chooseImageBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename pathname] = uigetfile('*.jpg','Image Selector');
faceImg_path = strcat(pathname,filename);
faceImage= imread(faceImg_path);
axes(handles.originalImgAxes);
imshow(faceImage,[]);
set(handles.filenameTxt,'string',filename);
%save the image path
handles.img_path = faceImage;
guidata(hObject, handles);
% --- Executes on button press in faceMatchBtn.
function faceMatchBtn_Callback(hObject, eventdata, handles)
% hObject    handle to faceMatchBtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
faceImg_Input = handles.img_path;

[fcimg1, fcimg2, fcimg3] = connectPCA(faceImg_Input);

axes(handles.matchedImgAxes2);
imshow(fcimg1,[]);

axes(handles.matchedImgAxes3);
imshow(fcimg2,[]);

axes(handles.matchedImgAxes4);
imshow(fcimg3,[]);
```

References

[1] Liton Chandra Paul, Abdulla Al Sumam
http://ijarcet.org/wp-content/uploads/IJARCET-VOL-1-ISSUE-9-135-139.pdf
(Accessed: 14 Dec 2018)

[2] Kyungnam Kim, Face Recognition using Principle Component Analysis,
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.383.6655&rep=rep1&type
(Accessed: 22 Nov 2018)