EE 417 Term Project

Hakan Buğra Erentuğ

Nidanur Günay

Face recognition has become one of the most attracted considerable attention research are computer vision. Beside computer scientists, neuroscientist and psychologist also interested in due to the nature of the problem. Their aim is gaining some useful insights to neuroscientists about how the human brain works etc. Other main are that has been face recognition is used is security. Security has become important after the rapid rise of number of criminalities and rising public concern of security. It is currently used in security cameras in airports, atm, offices, university, bank and even at home. Nowadays, it has started to be used in personal cell phones to detect the owner. Due to the significance of application areas, face recognition's goals should be accuracy and speed. It should detect a face in an image that involves extracting features and recognize disregarding illumination, translation, rotation, scaling of image, lighting and ageing.

First and essential step of face recognition is face detection. It requires a wide range of knowledge from image processing and pattern recognition. Techniques can be categorized in two approaches[1].

- Feature-based approach
- Image-based approach

They distinguished to each other based on their different ways to utilize face knowledge.

Feature-based Approaches: This approach uses apparent features of faces such as face geometry and skin color. These tasks are achieved by manipulating angles, distance and area measurements of visual features. One of the basic approaches uses the technique of skin-like features. This technique detects skin-like regions in image and group them together by the connected component-analysis. As a last step, verification uses local features and predefined templates helps to model facial features[1].

Image-based Approaches: Despite the feature-based approach, image-based approach learns to recognize facial patterns from the examples. Thus, image-based approach reduces the modelling errors due to incomplete or inaccurate face knowledge. This approach basically relies on statistical analysis and machine learning techniques. Linear-subspace approach is common image-based approach. Initially it treats NxM images as vector that is in NM dimensional space and this technique projects image patterns into a lower dimensional space which is achieved by

dimensional reduction in order to decrease computational complexity. After that step, eigenvectors and covariance matrix are used[1].

Face detection is used to detect faces from images and face recognition is the next step to find out whose faces is that faces. In order to achieve face recognition several face recognition techniques has been researched and found in computer vision area. LBA (local binary patterns), ICA (independent component analysis), LDA (linear discriminant analysis) and PCA (principal component analysis) are examples of these techniques. PCA is most commonly used techniques and in our research we've focused on that technique[2].

Principal Component Analysis:

This technique invented by Karl Pearson in 1901 and it involves calculation of eigenvalue decomposition of a data covariance matrix or singular value decomposition of a data matrix.

· Suppose each data point is N- dimensional.

$$var(v) = \sum_x ||(x - \bar{x})^T . v||$$

$$= v^T A v \text{ where } A = \sum_x (x - \bar{x})(x - \bar{x})^T$$

This A defines new coordinate system and data can be compressed by only using top few eigenvectors. We can find the best subspace using that technique. Points can be represented by line plane or hyper-plane. In order to place the faces, PCA uses set of training faces. Main idea is set of all faces are subset of space of all images. PCA helps to find best subspace[3].

· Extracting eigenfaces is the next step of PCA. We can think faces like weighted combination of some components and by extracting eigenvectors of matrix A, we get eigenfaces. Thus, eigenfaces span the space of faces. Therefore, linear combination of eigenfaces gives us the main face.

After projection of training faces in principal components space (eigenface space), target image also projects on eigenspace.Distance is a numerical measurement of how far apart objects or points are.In mathematics, a distance function on a given set M is a function d: $M \times M \rightarrow \mathbf{R}$, where **R** denotes the set of real numbers, that satisfies the following conditions:

- $d(x,y) \geq 0$, and $d(x,y) = 0$ if and only if $x = y$. (Distance is positive between two different points, and is zero precisely from a point to itself.)
- It is symmetric: $d(x,y) = d(y,x)$. (The distance between x and y is the same in either direction.)
- It satisfies the triangle inequality: $d(x,z) \leq d(x,y) + d(y,z)$. (The distance between two points is the shortest distance along any path).

Thus, distance is used to measure how close the target image's eigenfaces to other eigenfaces on principal component subspace[5].Therefore, following step is measuring distance between target image and training images projections in eigenface space.If the difference is smaller than the threshold, target face's corresponding face is founded[4].

In order to make face recognition task accurately and fast, distance measurement is the most significant step of PCA analysis. Our research project focuses on the distinguishing which distance measurement technique is supposed to be used in PCA. We've examined 11 distance measurement techniques in our research.

1. **Sum of Absolute Difference (SAD):** That technique uses summation of two vectors in principal component space.

$$||x - y|| = \sum_{i=1}^{n} |x_i - y_i|$$

2. **Sum of Squared Difference (SSD):** It also known as squared Euclidian distance.

$$||x - y|| = \sum_{i=1}^{n} (x_i - y_i)^2$$

3. **Mean Absolute Error (MAE):** It is the normalized version of the sum of absolute difference.

$$||x - y|| = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i|$$

4. **Mean Squared Error (MSE):** It is the normalized version of the sum of squared difference.

$$||x - y|| = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2$$

5. **Euclidian Distance:** It is the special case of Minkowski distance with p=2.

$$||x - y|| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

6. **Manhattan Distance:** It is also special case of Minkowski distance with p=1. It equivalent to sum of absolute difference. In our research we implement this distance technique in nomdistance1 function.

$$||x - y|| = \sum_{i=1}^{n} |x_i - y_i|$$

7. **Chebyshev Distance:** It is a special case of Minkowski distance where p goes to infinity. It is also known as Chessboard distance.

$$||x - y|| = \lim_{p \to \infty} (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} = max\, |x_i - y_i|$$

8. **Minkowski Distance:** It generalized -norm difference. We've chosen p value as 3 and 4 in our nomdistance3 and nomdistance4 functions.

$$||x - y|| = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

9. **Canberra Distance:** It is weighted version of Manhattan distance. It is often used for data scattered around an origin, as it is biased for measures around the origin and very sensitive for values close to zero.

$$||x - y|| = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

10. **Cosine Distance:** The cosine distance contains the dot product scaled by the product of the Euclidean distances from the origin. It represents the angular distance of two vectors while ignoring their scale.

$$||x - y|| = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} + \sqrt{\sum_{i=1}^{n} y_i^2}}$$

11. **Mahalanobis Distance:** This technique treats variation of all axes as equally significant.

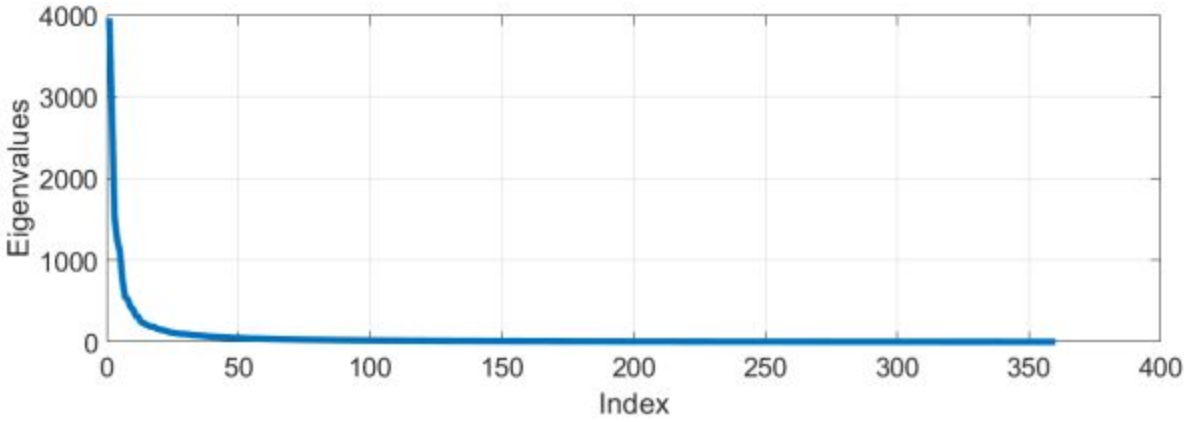
$$||x - y|| = \frac{1}{\lambda_i} \sum_{i=1}^{n} (x_i - y_i)^2$$

Test Data

In the experiment design the faces94 database which collected by Essex University, have been used. There are 20 image of each 153 individuals which contains male and female subjects, various racial origins, and wide age range. The format of images is 24bit colour JPEG with 180 by 200 pixel portrait format image resolution . In the experiment the all database, resized by 48 by 48 pixels and reduced to grayscale. The database is not labeled and licenced. The file name of each image starts with exact same seven character per individual which help us to find the correct recognition.

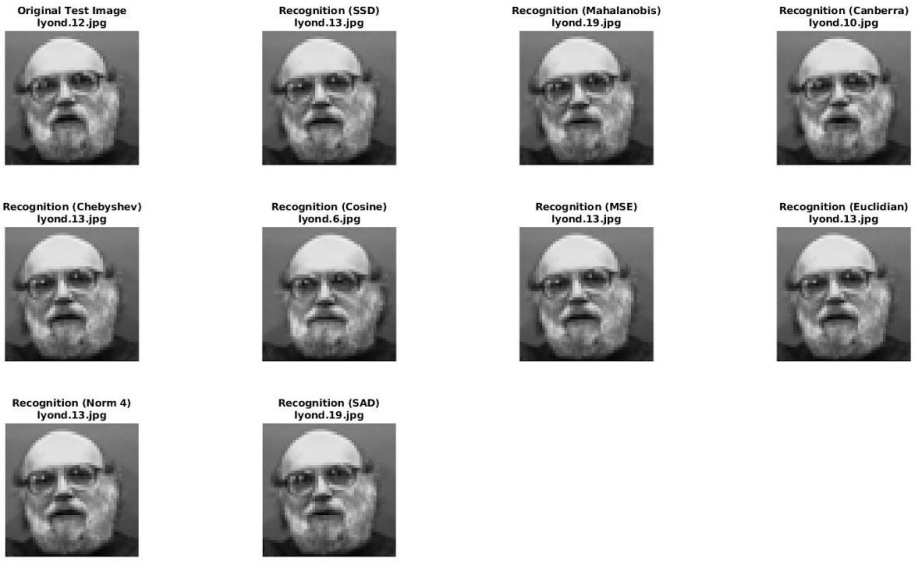
The Experiment Design

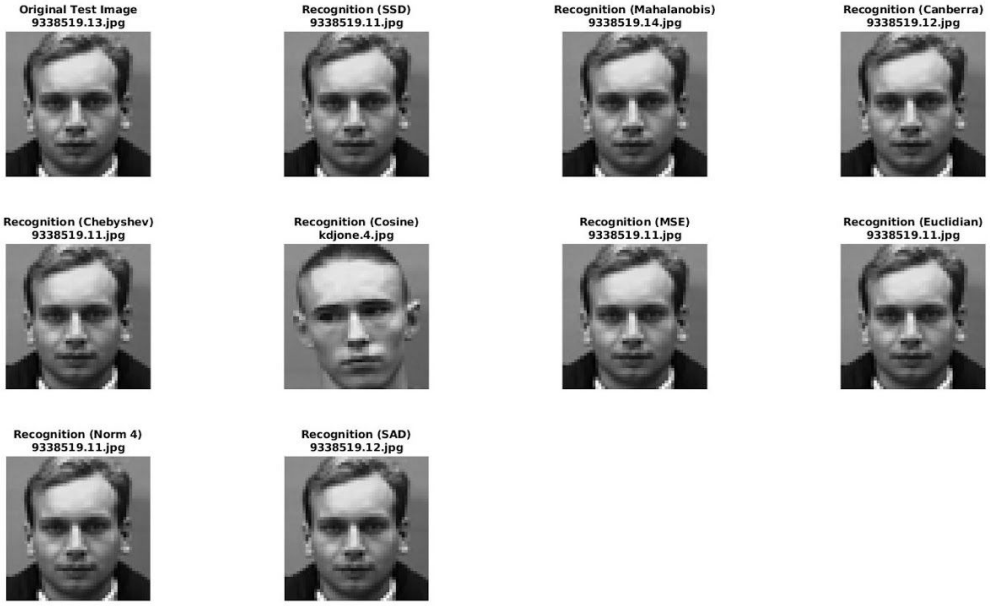
This experiment inspired from Yambor (2002) And Perlibakas (2004) studies which investigates the best geometric distance model for facial recognition method. The candidate distance models are respectively; Euclidian, Mahalanobis, Canberra, Chebyshev, MSE, SSD, SAD and Norm4. After the collecting 3060 different faces of 153 individual 600 of them randomly selected for test data. The rest of them labeled as train data. Each of the images resized to 48x48 and converted to grayscale. In addition, for clarity the file names also stored in the experiment. The number of eigenvalues are assumed 60 because of the performance accuracy tradeoff which is subjectively selected from graph below [3][5].



Then, the PCA is used for facial recognition. Each 48x 48 image is transformed to 1D array 1x2304 pixel. The first thing that achieved is decomposition of eigenfaces. Due to we assumed K = 60 which is the K largest eigenvalue and for each eigenvalue, the corresponding eigenvector exists, 60 eigen faces could be obtained. The first 12 eigen face of random test image is placed below.

The main research question that interested is which of the distance model would give better results. That means which distance formula will give more consistent and accurate recognition. By saying distance it means the geometric difference between the corresponding matrix in face space and rest of the all 3060 image. Technically, the lowest value of distance matrix would be the most similar face to our test image. The few of the test cases have been displayed for clarity. Below, the most upper right is our original test image and rest of them is most similar face (recognition) by different metrics.





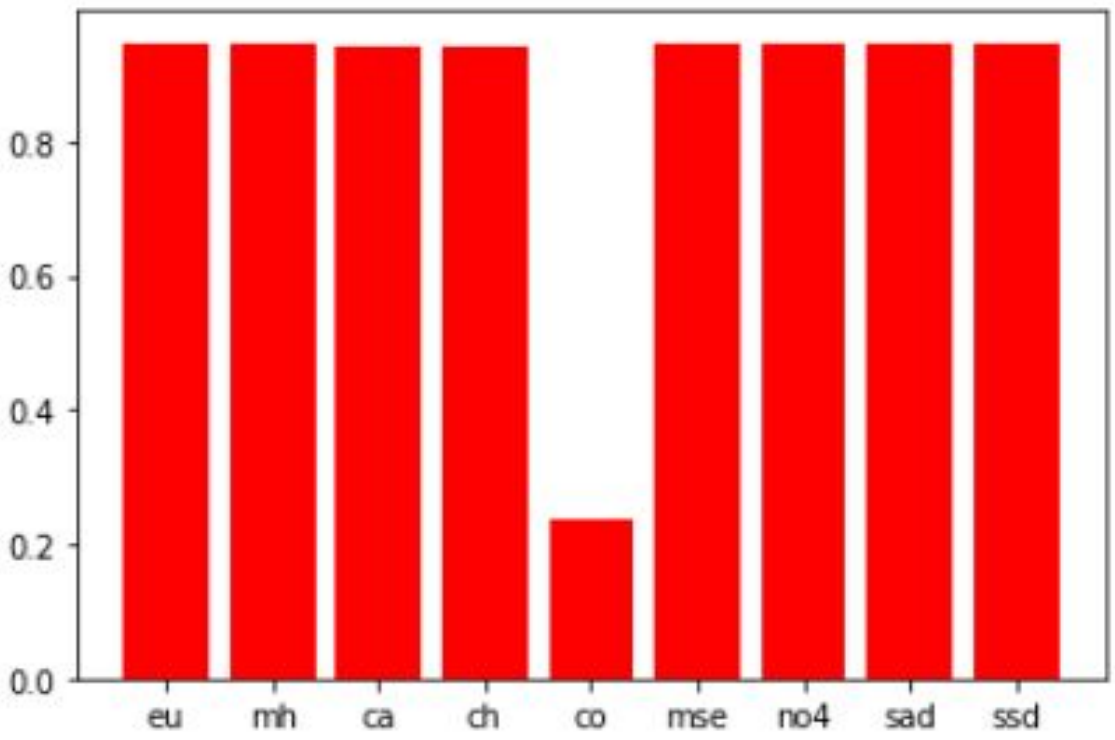
Respectively; the test image, SSD, Mahalanobis, Canberra, Chebyshev, Cosine, MSE, Euclidian, Norm-4 and SAD. In figure-1, all distance metrics successfully recognizes the individual however, not by the same picture. It is possible to check the different photos by filename extension like lyond.19. However, In figure-2 The cosine distance metric recognize the face inaccurately.

The inaccurate recognitions can be checked by the filenames' first seven characters thanks to named data. In output matrix the false recognition is labeled as -1.
Nevertheless, only accuracy rates is not a comprehensive and valid comparison. Because, the train size is too much. For each test image, at least nine train images are expectedly close for most of the distance metrics. As authors we predict more than 90% percent accuracy for each metric except for cosine distance. Therefore, in the results section, also consistency graphs have been placed. In this report the consistency is a measure of how many other distance metrics voted to the same image. This especially could enlighten which distance method could be more comprehensive when case is unknown because more consistent means more alternative to other metrics. The results have been stored in an excel file.

Test Results

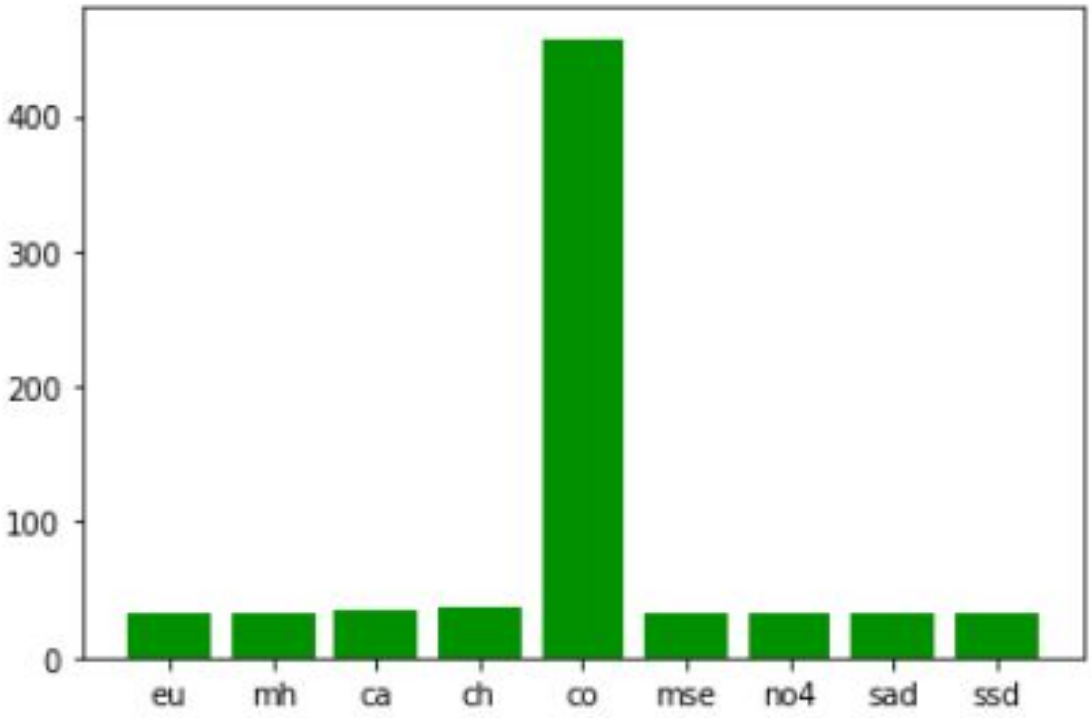
The results are satisfiably high as expected. Below the accuracy rates have been given in a table, accuracy histogram and error histogram.

The distance metric	The accuracy rate
Euclidian	94,5%
Mahalanobis	94,5%
Canberra	94,1%
SSD	94,6%
MSE	94,5%
Chebyshev	94%
Cosine	23,8%
SAD	94,5%
Norm-4	94,5%

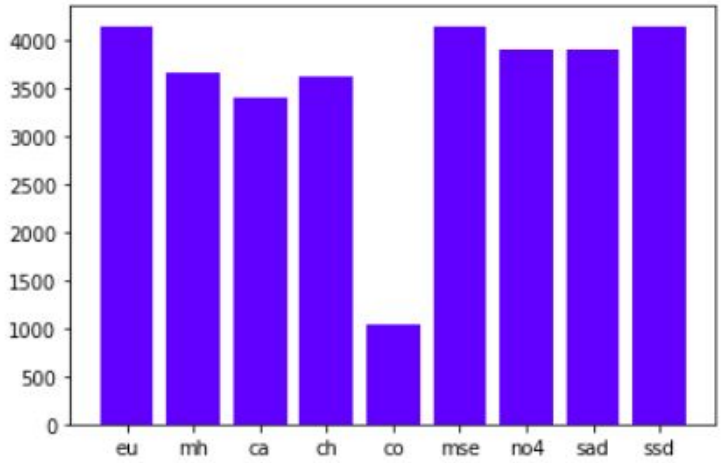


The number of error table is given below.

Metric Name	Number of Error out of 600 test
Euclidian	33
Mahalanobis	33
Canberra	35
Chebyshev	36
Cosine	457
MSE	33
Norm-4	33
SAD	33
SSD	32



The consistency metric is also a case in our test experiment. Due to the fact that there is no way to know which prediction is the best in the same individual, the authors decided to the best estimation is the highest rated candidate across the metrics. Therefore, the most voted candidate will take the reward. The readers should know that, consistency rates are irrelevant to the correct option. It is the measure of the more capable to find alternative and comprehend other metrics. Therefore, the mathematical similarity of formulas are significantly important still the inadequate difference between metrics enables to apply this criteria. Below, the consistency graph of metric have been given.



Conclusion

In this project, we investigated various distance metrics to understand the applicability of these metrics in face recognition efficiently. By looking at the results, we could say there is no significant difference between metrics except for cosine metrics. It is clearly obvious that cosine distance metric is not valid and applicable method to recognize faces. For the rest of them, we could say that Euclidean and also MSE, Mahalanobis and SSD distance is the most comprehensive metrics among all of them. In this project we have tried different K values to optimize the results. However, as authors we expect the high accuracy rate by looking at the train size. Almost all of the metrics and the implementation success is more than 90%. As authors, we satisfied with face recognition method by using PCA and its implementation. However, there are few design flaws which we have to admit. Firstly, the train size is more than adequate to test metrics in accuracy criteria. For almost each test image, at least 9 train image exists. Therefore, it is highly possible that giving the high results. Secondly, consistency criteria, is not related with preferability of the metric. It is based on mathematically background of distance formulas which only shows that the alternatives of metrics. Lastly, the random selection of a test data from the 3060 instance is not preferably method in data science. Instead it was better to split to data by a user-defined rate and picking testing images from the specially segregated set of images was better method.

Appendix:

SADdist.m

```
function [saddist] = SADdist(K, trainsize, database, descriptor)

saddist=[];

for i=1:1:trainsize

     dist = 0;

     for j=1:1:K

     dist = dist + abs((database(i,j)-descriptor(j)));

     end

    saddist=[saddist; dist];

end

end
```

SSDdist.m

```
function [ssddistance] = SSDdist(K, trainsize, database, descriptor)

ssddistance=[];

for i=1:1:trainsize

     dist = 0;

     for j=1:1:K

     dist = dist + (database(i,j)-descriptor(j))^2;

     end

    ssddistance = [ssddistance; dist];

end
```

```
end
```

MSEdist.m

```
function [msedist] = MSEdist(K, trainsize, database, descriptor)
msedist=[];
for i=1:1:trainsize
      dist = 0;
      for j=1:1:K
      dist = dist + (database(i,j)-descriptor(j))^2;
      end
      msedist = [msedist; dist/K];
end
end
```

MAEdist.m

```
function [maedist] = MAEdist(K, trainsize, database, descriptor)
maedist=[];
for i=1:1:trainsize
      dist = 0;
      for j=1:1:K
      dist = dist + abs((database(i,j)-descriptor(j)));
      end
      maedist = [maedist; dist/K];
```

```
end


end
```

euclidiandistance.m

```
function [eucdistances] = euclidiandistance(K, trainsize, database,
descriptor)

eucdistances = [];

for i=1:1:trainsize

      eucdist = 0;

      for j=1:1:K

         eucdist = eucdist + (database(i,j)-descriptor(j))^2;

      end

    eucdistances = [eucdistances; sqrt(eucdist)];

end

end
```

Nomdistance1.m

```
function [nomdist] = nomdistance1(K, trainsize, database, descriptor)

nomdist=[];

for i=1:1:trainsize

      dist = 0;

      for j=1:1:K

      dist = dist + (database(i,j)-descriptor(j));
```

```
      end

    nomdist=[nomdist; dist];

end

end
```

Chebyshewdist.m

```
function [chebdist] = Chebyshevdist(K, trainsize, database,
descriptor)

chebdist=[];

for i=1:1:trainsize

      dist = [];

      for j=1:1:K

      dist = [dist ;abs(database(i,j)-descriptor(j))];

      end

      chebdist = [chebdist; max(dist)];

end

end
```

nomdistance3.m

```
function [nomdist] = nomdistance3(K, trainsize, database, descriptor)

nomdist=[];


for i=1:1:trainsize

      dist = 0;
```

```
      for j=1:1:K
          dist = dist + (database(i,j)-descriptor(j))^3;
      end
    nomdist=[nomdist; nthroot(dist,3)];
end
end
```

nomdistance4.m

```
function [nomdist] = nomdistance4(K, trainsize, database, descriptor)
nomdist= [];
for i=1:1:trainsize
      dist = 0;
      for j=1:1:K
      dist = dist + (database(i,j)-descriptor(j))^4;
      end
    nomdist=[nomdist; nthroot(dist,4)];
end
end
```

Canberradistance.m

```
function [candist] = Canberradistance(K, trainsize, database,
descriptor)
candist=[];
for i=1:1:trainsize
```

```
      dist = 0;

      for j=1:1:K

      dist = dist + (abs(database(i,j)-descriptor(j)))/
(abs(database(i,j))+ abs(descriptor(j)));

      end

      candist = [candist; dist];

end

end
```

cosinedistance.m

```
function [cosdist] = cosinedistance(K, trainsize, database,
descriptor)

cosdist=[];

for i=1:1:trainsize

      totalxy=0;

      totalx=0;

      totaly=0;

      for j=1:1:K

       totalxy=totalxy+ (database(i,j)*descriptor(j));

       totalx=totalx +(database(i,j))^2;

       totaly=totaly +(descriptor(j))^2;

      end

    ratio=sqrt(totalx)+sqrt(totaly);

      cosdist = [cosdist; 1-(totalxy/ratio)];
```

```
end

end
```

mahdistance.m

```
function [mahdistances] = mahdistance(K, trainsize, database,
descriptor,D)

mahdistances = [];

for i=1:1:trainsize

      mahdist = 0;

      for j=1:1:K

         mahdist = mahdist + ((database(i,j)-descriptor(j))^2)/D(j,j);

      end

    mahdistances = [mahdistances; mahdist];

end

end
```

facerecognition.m

```
 clear all;clc; close all;

%% Modifying and loading the all faces  for the test


% Read all faces from 'faces' folder

Faces=[];
```

```
Names = string(missing);
Directory = 'faces';
Imgs = dir(Directory);
row = 48;
col = 48;
for j=1:length(Imgs)
    thisname = Imgs(j).name;
    thisfile = fullfile(Directory, thisname);
    try
      Img = imread(thisfile);  % try to read image
      Img = imresize(rgb2gray(Img),[row col]);
      Img = reshape(Img, [], 1);
      Faces = [Faces Img];
      Names(j)=thisname;
   catch
   end
end
Names=Names(3:3061);
%% Taking average faces
Faces = double(Faces)/255;
train = Faces(:,1:3059);
results=[]
Testsize=300;
for tr=1:Testsize
```

```
tr

rng=randi([1,3059]);

Test = Faces(:,rng);

testname=Names(rng);


xbar = mean(train')';

A = train - xbar;



K = 60;

[V D] = eigs(A'*A,K); % Returns K largest eigenvalues

newV = (A*V)/(norm(A*V));

diffx = Test - xbar;

faceX = 0;

database = [];


normav = norm(A*V);

eigfaces = [];

for i=1:1:K

    smalla = dot(diffx, newV(:,i));

    av = (smalla * newV(:,i));

    faceX = faceX + av;

    if i<=12

        eigfaces = [eigfaces av];
```

```
    end
end


faceX = faceX + xbar;



eigfacesimgs = [];
for i=1:1:12
eigfacesimgs = [eigfacesimgs double(reshape(eigfaces(:,i)*255,
[row,col]))];
end
%figure; colormap(gray); montage(eigfacesimgs, 'Size', [1 1]);
hold on;
%figure
% Bulding the database for K eigfaces
database = [];
for i=1:1:3059
    x = train(:,i);
    diffx = x - xbar;
    dbrow = [];
    for j=1:1:K
        smalla = dot(diffx, newV(:,j));
        dbrow = [dbrow smalla];
    end
```

```
    database = [database; dbrow];
end

descriptor = [];
for i=1:1:K
    testdiffx = Test - xbar;
    descriptor = [descriptor dot(testdiffx, newV(:,i))];
end


%% Face Recognition

trainsize=3059;
mh= mahdistance(K, trainsize, database, descriptor,D)  ;
eu= euclidiandistance(K, trainsize, database, descriptor);
ca=Canberradistance(K, trainsize, database, descriptor);
ch=Chebyshevdist(K, trainsize, database, descriptor);
co=cosinedistance(K, trainsize, database, descriptor);
mse=MSEdist(K, trainsize, database, descriptor);
no3=nomdistance3(K, trainsize, database, descriptor);
no4=nomdistance4(K, trainsize, database, descriptor);
sad=SADdist(K, trainsize, database, descriptor);
ssd=SSDdist(K, trainsize, database, descriptor);
```

```
[out,idx]=sort(eu);
euindex=idx(2);

[out,idx]=sort(mh);
mhindex=idx(2);

[out,idx]=sort(ca);
caindex=idx(2);

[out,idx]=sort(ch);
chindex=idx(2);

[out,idx]=sort(co);
coindex=idx(2);

[out,idx]=sort(mse);
mseindex=idx(2);

[out,idx]=sort(no3);
no3index=idx(2);

[out,idx]=sort(no4);
no4index=idx(2);
```

```
[out,idx]=sort(sad);

sadindex=idx(2);


[out,idx]=sort(ssd);

ssdindex=idx(2);


r=[rng euindex mhindex caindex chindex coindex mseindex no4index
sadindex ssdindex];

%% Displaying


% subplot(3,4,1);

% imshow(uint8(reshape(Test*255, [row,col])));

% title(['Original Test Image',testname]);

% subplot(3,4,2);

% imshow(uint8(reshape(train(:,ssdindex)*255, [row,col])));

% title(['Recognition (SSD)',Names(ssdindex)]);

%

tf = strncmpi(testname,Names(ssdindex),7);

if(~tf)

    r(2)=-1;

end

%

% subplot(3,4,3);
```

```
% imshow(uint8(reshape(train(:,mhindex)*255, [row,col])));
% title(['Recognition (Mahalanobis)',Names(mhindex)]);
%
tf = strncmpi(testname,Names(mhindex),7);
if(~tf)
    r(3)=-1;
end
%
% subplot(3,4,4);
% imshow(uint8(reshape(train(:,caindex)*255, [row,col])));
% title(['Recognition (Canberra)',Names(caindex)]);
%
tf = strncmpi(testname,Names(caindex),7);
if(~tf)
    r(4)=-1;
end
%
% subplot(3,4,5);
% imshow(uint8(reshape(train(:,chindex)*255, [row,col])));
% title(['Recognition (Chebyshev)',Names(chindex)]);
%
tf = strncmpi(testname,Names(chindex),7);
if(~tf)
    r(5)=-1;
```

```
end
%
% subplot(3,4,6);
% imshow(uint8(reshape(train(:,coindex)*255, [row,col])));
% title(['Recognition (Cosine)',Names(coindex)]);
%
tf = strncmpi(testname,Names(coindex),7);
if(~tf)
    r(6)=-1;
end
%
% subplot(3,4,7);
% imshow(uint8(reshape(train(:,mseindex)*255, [row,col])));
% title(['Recognition (MSE)',Names(mseindex)]);
%
tf = strncmpi(testname,Names(mseindex),7);
if(~tf)
    r(7)=-1;
end
%
% subplot(3,4,8);
% imshow(uint8(reshape(train(:,euindex)*255, [row,col])));
% title(['Recognition (Euclidian)',Names(euindex)]);
%
```

```
tf = strncmpi(testname,Names(euindex),7);
if(~tf)
    r(8)=-1;
end

% subplot(3,4,9);
% imshow(uint8(reshape(train(:,no4index)*255, [row,col])));
% title(['Recognition (Norm 4)',Names(no4index)]);
%
tf = strncmpi(testname,Names(no4index),7);
if(~tf)
    r(9)=-1;
end
%
% subplot(3,4,10);
% imshow(uint8(reshape(train(:,sadindex)*255, [row,col])));
% title(['Recognition (SAD)',Names(sadindex)]);

tf = strncmpi(testname,Names(sadindex),7);
if(~tf)
    r(10)=-1;
end

results = [results ; r]
```

```
end


writematrix(results,'output2.xls')
```

References

1.Jin, Zhong, Zhen Lou, Jingyu Yang, and Quansen Sun. “Face Detection Using Template Matching and Skin-Color Information.” *Neurocomputing* 70, no. 4-6 (2007): 794–800. https://doi.org/10.1016/j.neucom.2006.10.043.

2.Wójcik, Waldemar, Konrad Gromaszek, and Muhtar Junisbekov. “Face Recognition: Issues, Methods and Alternative Applications.” *Face Recognition - Semisupervised Classification, Subspace Projection and Evaluation Methods*, June 2016. https://doi.org/10.5772/62950.

3.Yambor, Wendy S., Bruce A. Draper, and J. Ross Beveridge. “Analyzing PCA-Based Face Recognition Algorithms: Eigenvector Selection and Distance Measures.” *Series in Machine Perception and Artificial Intelligence Empirical Evaluation Methods in Computer Vision*, 2002, 39–60. https://doi.org/10.1142/9789812777423_0003.

4.Zuo, Wangmeng, Kuanquan Wang, and D. Zhang. “Bi-Directional PCA with Assembled Matrix Distance Metric.” *IEEE International Conference on Image Processing 2005*, 2005. https://doi.org/10.1109/icip.2005.1530216.

5.Perlibakas, Vytautas. “Distance Measures for PCA-Based Face Recognition.” *Pattern Recognition Letters* 25, no. 6 (2004): 711–24. https://doi.org/10.1016/j.patrec.2004.01.011.