ISSN 2229-5518

Analysis of Face Recognition in MATLAB

Sanjay Kr Singh, Ashutosh Tripathi, Ankur Mahajan, Dr S Prabhakaran

Abstract— This paper mainly focus the recognize a person's identity is important mainly for security reason, but it could also be used to obtain quick access to medical, criminal, or any type of records. Solving this problem is important because it could allow personnel to take preventive action, provide better service - in the case of a doctor's appointment, or allow a person access to a secure area.

Index Terms— MATLAB, Image Processing, Eigen Vector, Eigen Values, Euclidean Distance. Signal System, Image Reconition, Image Segmentation

1. INTRODUCTION

The idea for this paper came up while studying soft computing. Mainly paper is based upon MATLAB comprising image processing. We have read different algorithms based on image processing like Adam's algorithm and fisher face algorithm etc.The images based on these algorithms as mentioned above had low quality and contrary to this context we tried to enhance the quality of the images.

2. REQUIREMENTS

SYSTEM:

PROCESSOR: Intel(R) Core(TM) i3-2310M CPU@2.27Ghz

INSTALLED MEMORY (RAM): 2.00 GBSYSTEM TYPE: 64 bit operating system

SYSTEM NAME: Windows 7 ultimate

SOFTWARE:

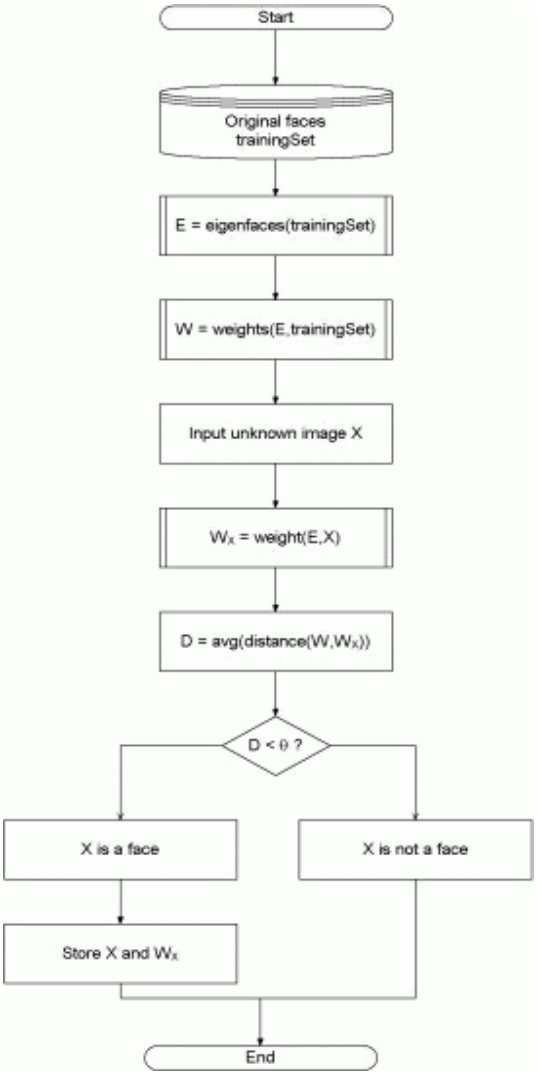
MATLAB R2009a

PICASSA 3.0, JRE 6.

- *Mr Sanjay Kr Singh is currently working in Amity university Rajathan-JAIPUR, INDIA,. E-mail: ASHU20034@Gmail.com*
- *Mr Ashutosh Tripathi is working in Dept of ECE, amity university rajasthan JAIPUR, INDIA, PH+91--9694403636. E-mail: ASHU20034@Gmail.com*

3. PROPOSED DESIGN OF MODEL

FLOWCHART



4. IMPLEMENTATION

1. First obtain a set S of M face images. Each image is transformed into a vector of size n and placed into the set.

$$S = \{ \Gamma_1, \Gamma_2, \Gamma_3, \ ..... \ , \Gamma_M \}$$

2. After we have obtained our set, we will obtain mean image Ψ

$$\Psi = \frac{1}{M} \sum_{n=1}^{M} \Gamma_n$$

3. Then we will find the difference Φ between the input image and mean image.

$$\Phi_i = \Gamma_i - \Psi$$

4. Now we seek a set of M orthonormal vector, un, (which best describe the distribution of data). Now the kth vector, uk, is chosen such that.

$$\lambda_k = \frac{1}{M} \sum_{n=1}^{M} \left(u_k^T \Phi_n \right)^2$$

5. We obtain the covariance matrix C in the following manner.

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T$$
$$= AA^T$$
$$A = \{ \Phi_1, \Phi_2, \Phi_3, \ ..... \ , \Phi_n \}$$

6. AT

7. $$L_{mn} = \Phi_m^T \Phi_n$$

8. $$u_l = \sum_{k=1}^{M} v_{lk} \Phi_k \qquad l = 1, ... \ , M$$

9. Once we have found the Eigen vectors we obtain the Eigen images of the set of images.

5. RECOGNITION PROCEDURE

A new face is transformed into its Eigen image components. Now we compare our input image with our mean image and multiply their difference with each Eigen vector of matrix. Each value would represent a weight and would be saved on a vector (say omega).Now we determine which face class provides the best description for the input image. It is done by minimizing the Euclidean distance. The input face is considered to belong a class, if Euclidean distance is below than established threshold, and then face is considered to be a known face if difference is above than given threshold, the image can be considered as unknown face. If input image is below than two thresholds, the image is determined not to be a face.

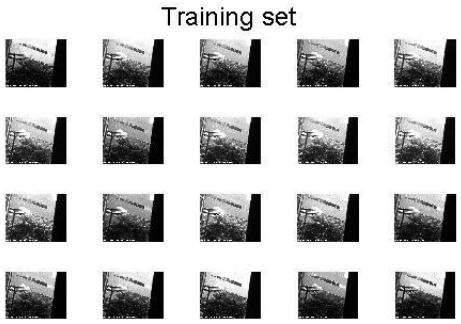
6. MATLAB CODING

```
clear all
close all
clc
% number of images on your training set.
M=20;

% Chosen std and mean.
% It can be any number that it is close to the std and mean
of most of the images.
um=100;
ustd=80;

% read and show image
S=[];   % img matrix
figure(1);
for i=1:M
str=strcat(int2str(i),'.bmp');      % concatenates two strings
that form the name of the image
eval('img=imread(str);');

subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
imshow(img)
if i==3
title('Training set','fontsize',18)
end
drawnow;
[irow icol]=size(img);    % get the number of rows (N1) and
columns (N2)
temp=reshape(img',irow*icol,1);      % creates a (N1*N2)x1
vector
```

```
S=[S temp];    % S is a N1*N2xM matrix after finishing the sequence
end

% Here we change the mean and std of all images. We normalize all images.
% This is done to reduce the error due to lighting conditions and background.
for i=1:size(S,2)
temp=double(S(:,i));
m=mean(temp);
st=std(temp);
S(:,i)=(temp-m)*ustd/st+um;
end

% show normalized images
figure(2);
for i=1:M
str=strcat(int2str(i),'.jpg');
img=reshape(S(:,i),icol,irow);
img=img';
eval('imwrite(img,str)');
subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
imshow(img)
drawnow;
if i==3
title('Normalized Training Set','fontsize',18)
end
end


% mean image
m=mean(S,2);  % obtains the mean of each row instead of each column
tmimg=uint8(m); % converts to unsigned 8-bit integer. Values range from 0 to 255
img=reshape(tmimg,icol,irow); % takes the N1*N2x1 vector and creates a N1xN2 matrix
img=img';
figure(3);
imshow(img);
title('Mean Image','fontsize',18)

% Change image for manipulation
dbx=[];    % A matrix
for i=1:M
temp=double(S(:,i));
dbx=[dbx temp];
end
%Covariance matrix C=A'A, L=AA'
A=dbx';
L=A*A';
% vv are the eigenvector for L
% dd are the eigenvalue for both L=dbx'*dbx and C=dbx*dbx';
[vv dd]=eig(L);
% Sort and eliminate those whose eigenvalue is zero
v=[];
d=[];
for i=1:size(vv,2)
if(dd(i,i)>1e-4)
v=[v vv(:,i)];
d=[d dd(i,i)];
end
end

%sort, will return an ascending sequence
[B index]=sort(d);
ind=zeros(size(index));
dtemp=zeros(size(index));
vtemp=zeros(size(v));
len=length(index);
for i=1:len
dtemp(i)=B(len+1-i);
ind(i)=len+1-index(i);
vtemp(:,ind(i))=v(:,i);
end
d=dtemp;
v=vtemp;
%Normalization of eigenvectors
for i=1:size(v,2) %access each column
kk=v(:,i);
temp=sqrt(sum(kk.^2));
v(:,i)=v(:,i)./temp;
end
%Eigenvectors of C matrix
u=[];
for i=1:size(v,2)
temp=sqrt(d(i));
u=[u (dbx*v(:,i))./temp];
end
%Normalization of eigenvectors
for i=1:size(u,2)
kk=u(:,i);
temp=sqrt(sum(kk.^2));
u(:,i)=u(:,i)./temp;
end
% show eigenfaces
figure(4);
for i=1:size(u,2)
img=reshape(u(:,i),icol,irow);
img=img';
img=histeq(img,255);
subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
imshow(img)
drawnow;
if i==3
title('Eigenfaces','fontsize',18)
end
end
% Find the weight of each face in the training set
omega = [];
for h=1:size(dbx,2)
WW=[];
for i=1:size(u,2)
t = u(:,i)';
```

```
WeightOfImage = dot(t,dbx(:,h)');
WW = [WW; WeightOfImage];
end
omega = [omega WW];
end
% Acquire new image
% Note: the input image must have a bmp or jpg extension.
% It should have the same size as the ones in your training
set.
% It should be placed on your desktop
InputImage = input('Please enter the name of the image
and its extension \n','s');
InputImage = imread(strcat('D:\Documents and Set-
tings\sis26\Desktop\',InputImage));
figure(5)
subplot(1,2,1)
imshow(InputImage); colormap('gray');title('Input im-
age','fontsize',18)
InImage=reshape(double(InputImage)',irow*icol,1);
temp=InImage;
me=mean(temp);
st=std(temp);
temp=(temp-me)*ustd/st+um;
NormImage = temp;
Difference = temp-m;
p = [];
aa=size(u,2);
for i = 1:aa
pare = dot(NormImage,u(:,i));
p = [p; pare];
end
ReshapedImage=m+u(:,1:aa)*p; %m is the mean image, u is
the eigenvector
ReshapedImage = reshape(ReshapedImage,icol,irow);
ReshapedImage = ReshapedImage';
%show the reconstructed image.
subplot(1,2,2)
imagesc(ReshapedImage); colormap('gray');
title('Reconstructed image','fontsize',18)

InImWeight = [];
for i=1:size(u,2)
t = u(:,i)';
WeightOfInputImage = dot(t,Difference');
InImWeight = [InImWeight; WeightOfInputImage];
end

ll = 1:M;
figure(68)
subplot(1,2,1)
stem(ll,InImWeight)
title('Weight of Input Face','fontsize',14)
% Find Euclidean distance
e=[];
for i=1:size(omega,2)
q = omega(:,i);
DiffWeight = InImWeight-q;
mag = norm(DiffWeight);
e = [e mag];
end
kk = 1:size(e,2);
subplot(1,2,2)
stem(kk,e)
title('Eucledian distance of input image','fontsize',14)
MaximumValue=max(e)  % maximum eucledian distance
MinimumValue=min(e)    % minimum eucledian distance
```

7. TEST RESULT



Dependind on the training set images and noramlised images and considering these eigen values and eigen faces, mean is matched with the input at the run time and if the mean is matched ,the result shows -known image and vice versa.

8. LIMITATIONS

The limitation of this model is that this will response to those inputs which will match the mean value of the training set image and corresponds to the eigen faces.

9. APPLICATIONS

This paper can be used in setting the passwords in the personal computers where your face is taken as the password of your personal computer.This can be used in the various offices to maintain the records of the employees with their faces as their account identity.This help in the medical assistance of the patients in the medical field, setting their face identity as the proof of all reports belonging to him.

10. CONCLUSION

First we input a known image and observed the Euclidean distance. This distance tells us how close the input image is from the image on our training set. Based on maximum and minimum distances we can make a decision of whether the face is a known face, an unknown face or not a face at all.
This model can be designed using other various programming techniques and the languages such as C++, JAVA, etc. but matlab can do this with ease using its image processing applications and tool box.

11. REFERENCES

[1] MIT University media laboratory, USA.

[2] Western Carolina university online library,Cullowhee,North Carolina(USA).

[3] History behind Eigen faces.

[4]D. Pissaarenko "Eigenface-based facial recognition.

[5]Delac, k.Grgic,M.Liatsis "Appearance based statistical methods for facial recognition"(Croatia).