RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Program: Electrical Engineering

Course Number	ELE882
Course Title	Digital Image Processing
Semester/Year	Winter 2012
Instructor	Yun Tie

Course Project 1

Report Title	Facial Recognition with PCA - An Analysis on Accuracy with Varying Sample Numbers

Section No.	1
Submission Date	April 9, 2012
Due Date	April 9, 2012

Name	Student ID	Signature*
William Bradley		
Imtiaz Miah		

**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/policies/pol60.pdf.*

.

Facial Recognition with PCA

An Analysis on Accuracy with Varying Sample Numbers

Imtiaz Miah
Department of Electrical Engineering
Ryerson University
Toronto, ON

William Bradley
Department of Electrical Engineering
Ryerson University
Toronto, ON

***Abstract*— The purpose of the experiment was to examine the effect varying the number of face samples of a subject, would have on the overall accuracy of a PCA based eigenface detection algorithms.**

Keywords-component; Face Recognition, Principal component analysis, eigenfaces

I. INTRODUCTION

The eigenface face detection algorithm has proven to be a robust solution to the problem of facial detection. Developed in the year 1991 at MIT the eigenface method for facial detection has become a standard in quick efficient detection. However eigenface detection does have its limitation, it is very sensitive to varying lighting as well as random image noise[5].

Furthermore the algorithm requires a large database of face faces in order to be accurate. In this report we will examine the latter flaw in depth, and attempt to determine a threshold value for the number of samples required in a facial detection test-set, for accurate facial detection.

Some applications of facial recognition are: Verification for security purposes, biometric applications, criminal justice systems, video indexing, and witness face reconstruction [5].

II. THEORY

A. PCA and Eigenfaces

The eigenface detection algorithm is a method of principal component extraction using statistical methods. Ergo, the eigenface represents statistically relevant components of the human face (e.g. noses, eyebrow, ridges, glasses, etc). Each real face can be described by a unique series of weights of an eigenface. The detection algorithm works by finding the weightings for the imputed image and compares it to the weightings of images in its face database[4]. Constructing the eigenface is a three step process, The first step is to subtract a the mean face from each face in the sample set. The equation below demonstrates this[1]:

$$\Phi = \text{Img} - \mu \quad (1)$$

$$A = [\Phi_1, \Phi_2, \Phi_3, \ldots \Phi_n] \quad (2)$$

The second step is to find the eigenvectors of the covariance of A. Using the standard approach however would be computationally impossible, as the size of the covariance for an image with N pixels would be N^2xN^2. However the problem can be reduced significantly through the use of linear algebra identities. Particularly with the following formula[1]:

$$\text{Cov} = AA^T \quad (4)$$

However as previously stated AA' is N2xN2, therefore we find the eigenvectors vi of A'A, which is only of size MxM, and use the identity:

$$u_i = Av_i \quad (3)$$

where ui are the largest eigenvalues of A, and therefore represent the most significant eigenfaces of the sample set.

We can think of eigenfaces as sort of a puzzle template to build certain faces. Certain combinations of eigenfaces with varying degree of weights can add up to a face[2].

B. Face Recognition

The same steps are taken except that it's considered for one image. First the image to be tested is taken out of the set and the mean difference is taken. Then it is projected onto the eigenvector. The difference between each normalized face in the training set and the normalized test mean difference set is compared and minimum euclidean distance is taken as the recognized face. Varying degrees of accuracy is obtained based on the number of sets and training faces for each set[1].

III. METHODOLOGY

Determining the accuracy of the algorithm based on the number of faces in the sample set. Was done by using a standard database of faces from the publicly available AT&T face database. The experiment was comprised of 20 different individuals, with up to 10 samples of each individual face, making the total number of images analyzed 200. There are actually 40 sets but the students decided to stick with a maximum of 20 sets for simplicity

The facial recognition algorithm was run repeatedly incrementing the numbers of face samples corresponding to each individual by one, starting at 2 samples per individual. Then in order to ensure that samples of faces where not more robust than others, we chose different samples for each individual and averaged the resulting accuracy rates. This method was repeated for sets of 5, 10 and 20 with the number of samples varying from 2 to 9.

The students created a function for the main part of this lab with two inputs. Groups takes in how many sets the student wants to take in and faces takes in the number of samples to be taken into account for training. The students decided to take the maximum number of eigenfaces for the code as the main concern was to see how different samples affect the accuracy.

A random image is taken out of the whole set of images and then a for loop is made to iterate through every iamge within the boundaries specified by the user (group and faces). This why a maximum of 9 faces can be taken out of each set, to ensure that each set has equal representation. The set is resized and then it follows the algorithm outlined in the theory.

For keeping track of the accuracy of the face recognition function, a script was made that iterates the function a number of times that keeps a counter of how many times the test image matches the set in the determined matched index. Since each folder contains the same individual, the students kept index numbers in sets of 10 (eg: s2 folder would have index images of 21, 22... 29).

IV. RESULTS

A. *Eigenfaces*

The results varied based on how much sets were used and how many faces were taken from each set. The eigenfaces obtained for using 10 sets with 2 faces in each is shown as below:

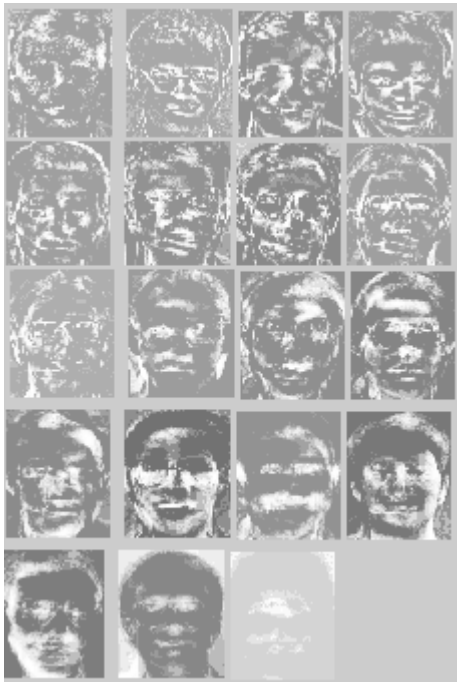

Figure 1. Eigenfaces obtained

These eigenfaces represent the order of how faces can be represented. Although ordered backwards, the eigenfaces actually get more noisy as you go along more and more faces that will eventually resemble a gaussian if enough samples are taken into consideration. Sometimes it is not always necessary to take the maximum number of eigenfaces and this is mentioned in the discussion and conclusion in more detail.

B. *Face Recogntion*

To detect whether the right face was checked, the program verified whether the sample index(randIndex) matched the recognized index(matchedIndex) within the same base 10. If it did, then the accuracy check was done correctly. A correct and wrong detection are shown below:

Figure 2. Correct Recognition

Figure 3. False Positive Recognition

A script was used to iterate the face recognition function a number of times to determine the accuracy. The results were obtained varied depending on the number of sets used and the number of faces used from each set. The graphs below depict for sets of 5, 10, 20 and 40.

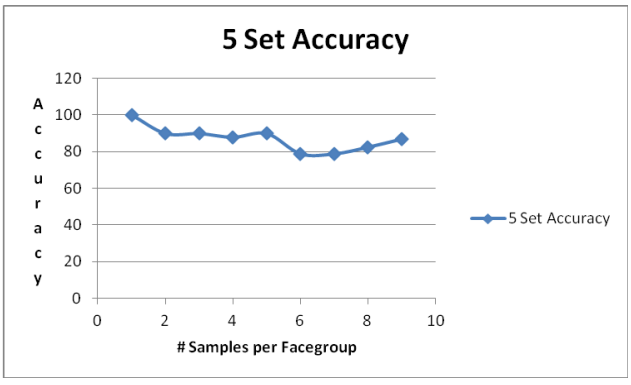

Figure 4. 5 set accuracy for varying face samples

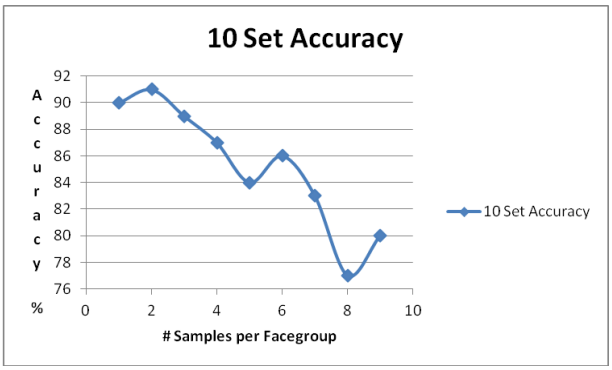

Figure 5. 10 set accuracy for varying face samples

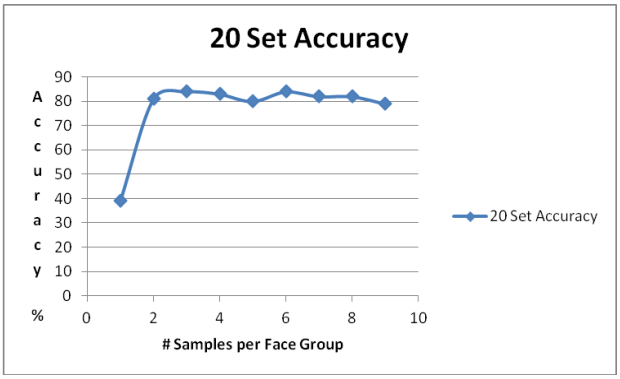

Figure 6. 20 set accuracy for varying face samples

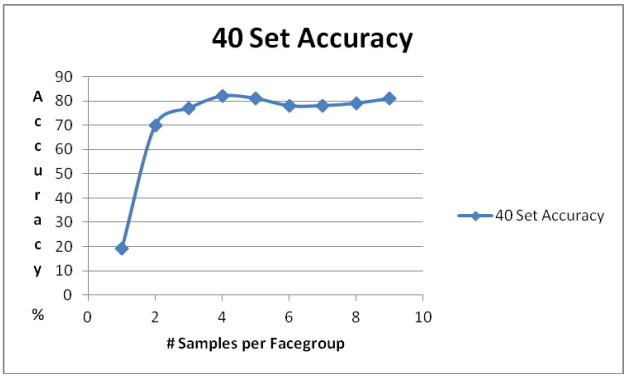

Figure 7. 40 set accuracy for varying face samples

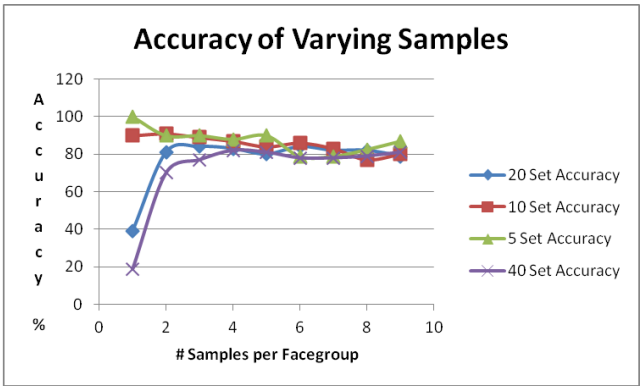

Figure 8. All the sets in one graph

It is clear that the number of samples affects the degree of accuracy. The explanations behind these graph patterns are explained more through in the discussion section of the lab.

In general, the 5 and 10 set accuracy seemed to decrease as you added more training samples per individual, but for the 20 to 40 set, it reached an asymptotic state. It was particularly interesting to note how inaccurate using 1 sample was at the 20 and 40 set. It should also be noted how much more computationally extensive it was to use the full 40 set as opposed to something as small as the 5 set.

V. Discussion

Increasing the number samples will only have a modest effect on the accuracy of the sample set past a certain asymptotic point. Also the accuracy of the detection algorithm is highly dependent on the efficacy of the sample set. We saw large disparities in accuracy based on which images from the sample set we where using, for the lower numbers of samples.

A proper asymptotic limit was established in the the results. The limit was shown to be approximately 87% proper detection rate. It can be seen form the results that attaining the asymptotic efficiency is not worth the increased computation time as increasing the number of samples is a linear increase in detection time, yet yields a logarithmic increase in performance[4].

It is clear however that for low sample numbers such as the one with 5 and 10 sets do not follow this asymptotic pattern as mentioned above. The reason for this maybe simply that at low number of samples, it is much easier to distinguish as the variances between each image is more noticable. As you add more samples, they get harder to distinguise since there are more similar features to consider. It should be noted though that at a certain point, the accuracy starts increasing again for the 5 and 10 sets. Though not as accurate as using 1 sample, the students believe that it will eventually reach an asymptotic state just like the 20 and 40 set patterns.

As mentioned in the preceeding section, the accuracy was considerably lower for using 1 sample in the 20 and 40 set. This was because there are more faces to consider and some faces can appear completely similar to each other based on the algorithm used. More training samples are needed to distinguish the faces more accurately.

For the experiment we used all of the eigenfaces generated for the image samples used. However this is unnecessary as there are only a limited number of significant eigenfaces. The number of eigenfaces which are significant could be determined by using a threshold euclidean distance value from the eigenface to images in our sample set. If the eigenface is too distant from the sample images then it can be discarded as insignificant.

VI. Conclusion

In general, the number of samples is proportional to the accuracy of the eigenface detection algorithm. However the increases in accuracy where far more modest then originally

anticipated. After the 3rd sample face for each individual, however, in a set with increased noise and variance in lighting conditions may benefit more from the increased samples of individuals. However, as the set increased to 40, it reached an asymptotic value around the 4th sample number per group. This implies that as you increase the database, more training samples are needed for each face for each individual.

It is clear though at a certain point, it becomes redundant and costly to add more samples per individual. The students believe this asymptotic value can be achieved at less than 10% of the number of sets available but more tests need to be done to verify this.

Although the students emphasize on using more than one training face for recognition, there are more advanced techniques that can be used. One article that interested the students was an appearance-based expression for face detection article. This is based on using a more synthesized images and a more advanced feature extractor [3].

One thing the students did not get a chance to check is what is stopping the face recognition algorithm from detecting arbitrary objects as faces. One way to counter this problem is to set a threshold based on the euclidean distance. If the euclidean distance passes a certain threshold that the user deems not a face, then it will result in less false positives. The problem with a threshold though is that it may reject perfectly fine faces so a smart threshold value has to be determined [2].

There are many different methods to improve face recogntion and almost endless permutations you can combine. Some methods call for SVM's, ICA's, LDAs, RBF and LVQ netowrks, etc [5]. The list goes on and on and the students see that the methods will grow more and more over time.

Investigating prepossessing noise reduction and light equalization techniques would be preferable to increasing the number of faces in the sample set in order to achieving gains in facial detection accuracy. As attaining extra samples of each individual is often difficult, as well as generating the eigenfaces is the most computationally expensive part of the algorithm.

REFERENCES

[1] M. Turk, and A. Pentland, "Eigenfaces for Face Detection/Recognition" Journal of Cofnitive Neuroscience, vol. 3, pp. 71-86, 1991.

[2] K. Kim, *Face Recognition using Principal Component Analysis*- Maryland, USA: University of Maryland, November 3, 2011

[3] H. Mommahzade "An Expression Transformation for Improving the Recognition of Expression-Variant Faces from One Sample Image per Person" University of Toronto, September 2010.

[4] L. Introna, "Facial Recognition Technology - A Survey of Policy and Implementation Issues" Ney York University, April 2009.

[5] R. Jafri, H. Arabnia "A Survey of Face Recognition Techniques" Journal of Information Processing Systems, vol. 5, 2009.

APPENDIX

Face Recognition

```
function [randIndex, matchedIndex] =test2(groups, faces)
N=groups*faces; %total number of faces
offset=0;
%the offset, this was of particular interest for the accuracy
%script as it was clear that taking the first 3 samples of each set
%gave different accuracy results than if we took another 3 samples in set

randIndex=ceil(N*rand); %
N_eig = groups*faces -1;%number of eigenfaces to be used
S = []; %image set.
B = [];
Phi = [];
%Reads all the images
for i=1:groups %how many face groups we're taking (upto 40)
    string =
('C:\Users\Neo\Documents\School\ELE882\facial\face_recognition\orl_faces\');
%opens directory of faces
    total_string = strcat(string, 's', int2str(i),'\');
    for j=1+offset:faces+offset %opens individual sets, offset is there for
accuracy check
        if (j+(i-1)*10==randIndex) %checks if the random image picked before
is picked here
            j=j+1;
            faces=faces+1;
        end
        final_string=strcat(total_string, num2str(j),'.pgm');
        Img=imread(final_string);

        B = reshape(Img, 1 ,112*92);
        S = vertcat(S,B);
    end
end

%-----------------------------------------------------

newSet=S(randIndex,:); %holds the random index
S=S([1:randIndex-1 randIndex+1:end],:);%ensures the random index is taken out

%find the average of the sets!
u = mean(S);% this should have width ~10340!
%size(u)
u = double(u);
S = double(S);
%now we need to find Phi = S - U to excentuate non-average features
for(i= 1:(N-1))
    C =S(i,:) - u;
    Phi = vertcat(Phi , C);

end
figure(1);imshow(reshape(uint8(u),112,92)); % this displays the average
```

```
%face
%okedoke . . .  now it gets tricky
%lots of crazy linear algebra to understand!
A = Phi';
A= double(A);

Cov = A'*A; % should be an MxM (8 by 8)?!
[eigenVect eigenValue] = eig(Cov); %ui=eigenValue
%so now we have M eigan values, or at least we should.
EigenFaces = A*eigenVect;

thetai = []; %projecting the normalized training faces toa  vector
for i = 1 : N_eig
    thetai = horzcat(thetai, EigenFaces'*A(:,i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%normalise the Faces! and determine Euclidean distances.
%is there a way to do this without loops?! I should check.
Euclidean_Dist = [];
Norm_Faces = [];
for i=1:N_eig
    Dist = sqrt(sum(EigenFaces(:,i).^2));
    Euclidean_Dist= vertcat(Euclidean_Dist, Dist);
    Norm_Face = EigenFaces(:,i)./Dist;
    Norm_Faces = horzcat(Norm_Faces, Norm_Face);
end
size(Norm_Faces);
size(Euclidean_Dist);
Dist = sqrt(sum(Norm_Faces(:,8).^2));


%display the faces!

figure(2); %displaying eigenfaces
for i=1:size(Norm_Faces,2)
    img=reshape(Norm_Faces(:,i),112,92);
    img=histeq(img,255);
    subplot(groups,faces,i)
    imshow(img)
    drawnow;
end

%figure(5);
%a=imshow(reshape(newSet,112,92));
a=reshape(newSet,112,92); %test image
newSet=double(newSet);


s=((newSet-u)*EigenFaces)'; %mean difference of test image

theta = []; %projecting onto eigenspace
for i = 1 :N_eig
    normdiff = ((norm(s - thetai(:,i))*(norm(s - thetai(:,i))))); %normalized
difference
```

```
    theta = horzcat(theta, normdiff);
end

[dist matchedIndex] = min(theta); %take the minimum value

RecogImg=S(matchedIndex,:);
figure(3); %displays the test image beside the matched image
b=reshape(uint8(RecogImg),112,92);
subplot(1,2,1), imshow(a);
subplot(1,2,2), imshow(b);
return
```

Accuracy Script

```
good=0;bad=0; %positive and false postive variables
for i=1:1000 %number of iterations
[randIndex, matchedIndex]=test2(40,3); %calls face recognition algorithm

   if (floor(randIndex/10)==floor(matchedIndex/10)) %floors and groups 10base
       good=good+1;
   else
       bad=bad+1;
   end
end

tot=(good)/(good+bad); %displays total accuracy percent
```