Applied Mathematics Homework report

Face Recognition using PCA

January 14, 2018

Authors:
Daria Zotova
Karim Botrus

Supervisor:
Désiré Sidibé

1 Introduction

One of the most interesting applications in Image Processing and Computer Vision is Face Recognition (FR). For many of us it is known through the Facebook that uses the system to help users tag people on photos. Also FR has promising perspectives in security, marketing and healthcare.

The main principal of FR is to find set of interesting features from face images in order to reduce the number of variables. In our case we have an input 2D image that should be compared with the database. For example, we have a face image with dimensions 300x300. It can be represented as a vector of size 300^2, and we would have to compare vectors in R^{9000} to find the best match. However, it is difficult to compute distance in high dimensional spaces. We need to find the subspace of Rd where the data structure is preserved. For this purpose PCA tool is used. The aim of this project is to use SVD and PCA for solving FR problem and propose a software to demonstrate how the system works.

2 Normalization

At the beginning we were provided with the set of 200 images each of the size 240x320 pixels (5 images of every student) and text files with coordinates of main face features. Before we can start recognition, images should be normalized to take into account scale, orientation and location variations. For the normalization step we have used an iterative scheme based on affine transformation. The transformation is needed to map specific facial features from a face image to predetermined locations in a fixed size window. We have used 5 following features with predetermined locations in the 64x64 window: left eye center(13, 20); right eye center(50, 20); tip of nose(34, 34); left mouth corner(16, 50) and right mouth corner(48, 50). To compute the parameters of an affine transformation the scheme below has been applied:

1) We stored the average locations of each facial feature over all images into a vector $\overline{F}$. For the first iteration $\overline{F}$ stores the feature locations of the first image.

2) The affine transformation is defined by 6 parameters

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \mathrm{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

.

Each feature of an image $f_i = \begin{bmatrix} x \\ y \end{bmatrix}$ can be mapped to the predetermined location f_i^P by the following equation:

$\mathrm{f}_i^P = Af_i + b$

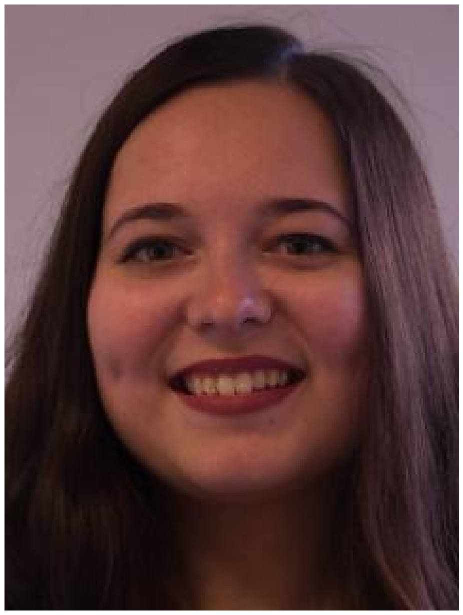
Totally we have 6 unknown parameters and 10 equations, so the system is over-determined and can be solved using SVD. When the transformation is obtained, we apply it to $\overline{F}$ and update it by setting $\overline{F} = \overline{F'}$.

3) Then we computed the best transformation that aligns the facial feature F_i with the average facial feature $\overline{F}$.

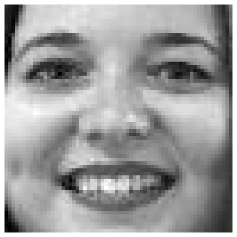
4) The aligned feature locations for each face image are then averaged and the vector $\overline{F}$is updated.

5) We compare vectors $\overline{F_t}$ and $\overline{F_{t-1}}$. If the error is more than a threshold we repeat steps 2-5.

For our computations the above described algorithm converged after 4 iterations. As a result, it yields an affine transformation that maps each face image to the 64x64 window.



(a) *Original image 240x320*



(b) *Mapped image 64x64*

Figure 2: *Result of normalization*

3 Recognition

In this work we have applied Principal Component Algorithm (PCA) for the Face Recognition after the normalization part for face detection and re-sizing to 64x64 for each image in our database. The steps of PCA are as follows:

- we have set the PCA parameters.
- we have read all the images and formed our training and testing data sets
- we assigned labels to each image if the images are taken from camera, the label is ! followed by the name of the person example: "!karim" to distinguish between the photos taken by the laptop camera and the photos in our data sets
- we have subtracted the mean and computed covariance matrix and then perform PCA
- we have compared the test image that was chosen by the user with all the training images in the training data set
- at the end we calculated the Accuracy of the Application.

So basically each image is a vector of its own rows concatenated.

$$X_i = [I_i(1,1), I_i(1,2), ..., I_i(1, MaxiColum), I_i(M,1) ..., I_i(M,N)] \tag{1}$$

After we have our data sets ready, we calculate the covariance matrix and then get the Eigenvectors of the covariance matrix. Then these Eigenvectors are multiplied by the data set to get the projection, and we do the same for the test image, then we compare this test image with the training data set to get the least error difference between them two, resulting in getting our best first match for this algorithm. Then the user is asked if he wants to get the 2nd and the nth best match by using the same algorithm.

3.1 HOW to improve The Results

First of all, in normalization we ignored the face contour which is a very important feature. In our second method of using Matlab computer vision too box "vision.CascadeObjectDetector" you will see how it uses all the features in the face to determine the exact match. Secondly, the Accuracy increases proportionally linear by increasing the number of the training images, so this could be the second best solution. Please see our Results section for further clarifications.

4 Results

To check the accuracy of the system, it can be calculated by the following formula

$$accuracy = (1 - \frac{\epsilon}{\text{number of test images}}) * 100$$

.

In our case we have totally 80 images in testing set and there were 13 errors. We have reached the accuracy of 83.75%.

The recognition system can be run by the application.

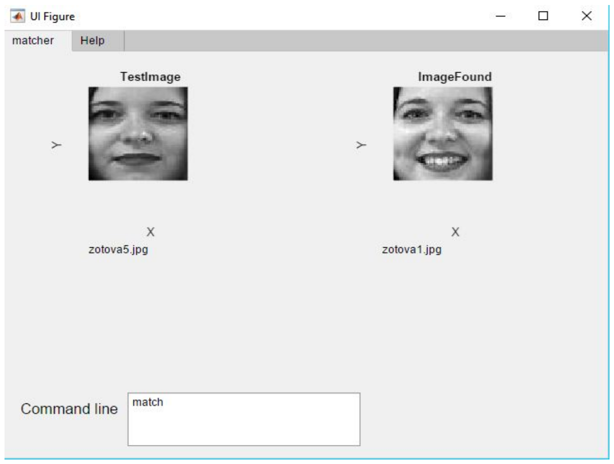


Figure 3: *Matched image has been found*

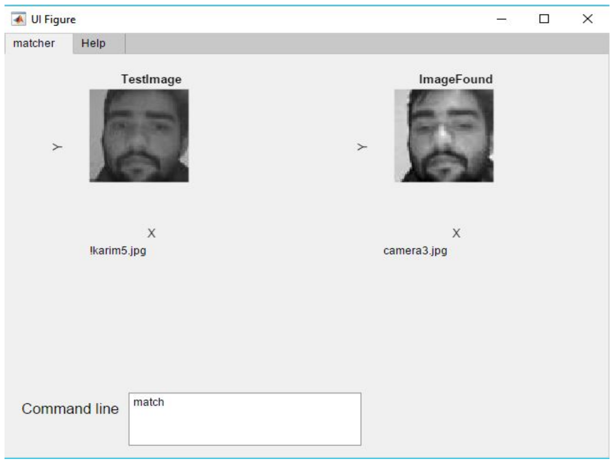
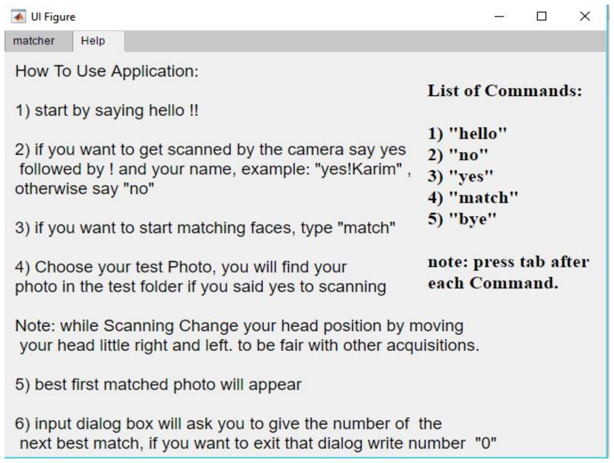


Figure 4: *Matched image for the one taken with webcamera has been found*

Please, go to the Help section to learn how to use the application.



How To Use Application:

1) start by saying hello !!

2) if you want to get scanned by the camera say yes followed by ! and your name, example: "yes!Karim" , otherwise say "no"

3) if you want to start matching faces, type "match"

4) Choose your test Photo, you will find your photo in the test folder if you said yes to scanning

Note: while Scanning Change your head position by moving your head little right and left. to be fair with other acquisitions.

5) best first matched photo will appear

6) input dialog box will ask you to give the number of the next best match, if you want to exit that dialog write number "0"

List of Commands:

1) "hello"
2) "no"
3) "yes"
4) "match"
5) "bye"

note: press tab after each Command.

Figure 5: *Help section*

During this assignment we have solved a Face Recognition problem, found the way to improve the system, learnt how SVD and PCA are used in practice, improved our coding skills in Matlab.

Source-code

Source-code in Matlab of the Normalization part.

```
clear all;
clc;
dname = uigetdir('C:\MATH');
files = dir(cat(2,dname,'\*.txt'));
E = [1.0000;1.0000;1.0000;1.0000;1.0000];
% Fp is a matrix with predetermined positions
Fp = [13.0000 20.0000 1.0000;
    50.0000 20.0000 1.0000;
    34.0000 34.0000 1.0000;
    16.0000 50.0000 1.0000;
    48.0000 50.0000 1.0000];
%prepare variables to store coordinates of features(f),
%transformation(Anew) and updated coordinates after applying Anew (new_F)
f = cell(length(files),1);
Anew = cell(length(files),1);
new_F = cell(length(files),1);
iterations = 0;

for i=1:length(files) %open all files with coordinates
    f{i,1} = dlmread(cat(2,dname,'\',files(i).name));
    f{i,1} = cat(2,f{i,1},E);
end
%For the 1st iteration we put a first image into a matrix with averaged
%features F
Fnew = f{1,1}
F = zeros(5,3)
%Find best transformation and compare current F with a previous F. If the
%difference is big, repeat again.
while (abs(F(:,1:2)-Fnew(:,1:2)) > 0.0001)
    F = Fnew
    A=F\Fp
    Fnew = F*A
    accum = zeros(5,3);
    for i=1:length(files)
        Anew{i,1} = f{i,1}\Fnew;
        new_F{i,1} = f{i,1}*Anew{i,1};
        accum = accum + new_F{i,1};
    end;
    Fnew = accum/length(files);
    iterations = iterations+1;
end
%Once we have found best transformation, we have a matrix Fnew. We take it
%for each picture, find transformation A and then
for n=1:size(files)
    F = f{n};
    A = F\Fnew;
    Ainv=A^(-1);
    point = cell(4096,1);%in order to store coordinates of pixels which
%should be copied into a new image with the size 64x64
    l=1;
    for x=1:64
        for y=1:64
            point{l} = ceil(abs([x y 1]*Ainv));
            l=l+1;
        end
    end

    img = imread(char(strcat(files(n).name(1:end-4), '.jpg')));
    img = rgb2gray(img);
    imshow(img);
    impixelinfo;
    %output image 64x64. For each pixel out(i,j) it takes a pixel from original
    %image, coordinates of which are stored at point{k}.
    out = zeros(64,64);
    k=1;
    for i=1:64
        for j=1:64
            if (point{k}(1)>240)
                point{k}(1)=240;
            end
            out(j,i) = img(point{k}(2),point{k}(1));
            k=k+1;
        end
    end
    imshow(out,[]);

    %Save the new cropped picture
    out=out-min(out(:)); % shift data such that the smallest element of out image is 0
    out=out/max(out(:)); % normalize the shifted data to 1
    path = 'C:\MATH\Cropped\';
    imwrite(out,[char(strcat(path, files(n).name(1:end-4), '.jpg'))]);
end
```

Below is the part for the PCA with the GUI implementation using Matlab 2017.

```
    value = app.CommandlineTextArea.Value;

    %list of commands
        Command3='yes';
        Command2='bye';
            Command1='hello';
            Command0='match me';
            Command01='match';
            Command10='no';
            Command5 = 'matchnext';
            Current_Folder = pwd;  %project directory

            trainingImagePath      = strcat(Current_Folder,'\trainingset');
            testingImagePath       = strcat(Current_Folder,'\testingset');
```

```
         for k = 1:length(value)
            if(strcmp(value{k},Command1))
                app.Label.Text = 'hello, can I take your picture and add it to my dataset ?';
                break;
            elseif(strcmp(value{k},Command2))
                app.Label.Text = 'byebye';
                pause(2);
               close all force;
                break;
```

```
elseif (strncmp(value{k},Command3,3))
        cam = webcam(1);
        preview(cam);
[name remain] = strtok(value{k},'!');
for i =1:3
     Im = snapshot(cam);
 faceDetector = vision.CascadeObjectDetector;

   bboxes = step(faceDetector, Im);
   face = Im(bboxes(1,2):bboxes(1,2)+
  bboxes(1,4),bboxes(1,1):bboxes(1,1)+bboxes(1,3));
    %imshow(face);
  face2 = imresize( face, [64 64]);
 imwrite(face2,strcat(trainingImagePath,'\',
          remain,num2str(i),'.jpg'));
                     pause(1);
end
for i =4:5
Im = snapshot(cam);
 faceDetector = vision.CascadeObjectDetector;
     bboxes = step(faceDetector, Im);
 face = Im(bboxes(1,2):
 bboxes(1,2)+bboxes(1,4),
 bboxes(1,1):bboxes(1,1)+bboxes(1,3));
 face2 = imresize( face, [64 64]);%resize by 64*64
imwrite(face2,strcat(testingImagePath,'\',
remain,num2str(i),'.jpg'));
                      pause(1);
                 end
                     break;
    elseif (strcmp(value{k}, Command10))
                app.Label.Text = ' ';
                break;
    elseif (strcmp(value{k},Command01))
       app.Label.Text = ' ';
  trainingImages =dir(cat(2,trainingImagePath,
  '\*.jpg'));
noOfTrainimages = length(trainingImages);
          %Creating the training dataset
  trainingdataset = zeros(noOfTrainimages,64*64);
  %creating the dataset with the same size of number of images

 for i = 1:noOfTrainimages
       %read image and convert to double and add to trainingdataset as rows
    imageReader =
    imread(char(strcat(trainingImagePath
    ,'\',trainingImages(i).name(1:end-4), '.jpg')));
    %reading each image along with its name

trainingdataset(i,:) =reshape(imageReader,1,64*64);
%reshaping each image into a row in the data matrix
 end
trainingdatasetaveraged = trainingdataset -
mean(trainingdataset')';
%then get the covariance of the dataSet
trainingdatasetaveragedCovar =
                    trainingdatasetaveraged' *
                    trainingdatasetaveraged /
                    noOfTrainimages;

                        k = 100;
                        [V,D] =
                eigs(trainingdatasetaveraged
                Covar,k,'lm');
                 projection = trainingdataset * V;

                     %Open test faces
                         imagetestingfiles = dir(cat(2,testingImagePath,'\*.jpg'));
 noOftestingimages = length(imagetestingfiles);

[FileName,PathName] = uigetfile('*.jpg','Select
Image',testingImagePath);
app.UIFigure.Visible ='off';  %fixing matlab bug by
%hiding and showing the application because of the
%previous bug
app.UIFigure.Visible = 'on';
%showing the app again
if ~ischar(FileName)
     return;
end
image=imread(strcat(PathName,FileName));
          imTesting = double(image);
chosentestimage = reshape(imTesting,1,64*64);
   projTestimage = chosentestimage * V;
    imshow(image,'parent',app.UIAxes);        %axis(app.UIAxes,'on');
    %if you want to show the axes
 error_vec = zeros(1,noOfTrainimages);
for t = 1:noOfTrainimages
error_vec(t) = immse(projTestimage,projection(t,:)); end
 iter= 1;
[out,error_vect_orderd] = sort(error_vec);
imshow(reshape(trainingdataset
(error_vect_orderd(iter),:),
64,64),[], 'parent' ,app.UIAxes_2);

app.Label_3.Text = trainingImages
(error_vect_orderd(iter)).name;
    app.Label_2.Text =FileName;

x = inputdlg('Number of Next best Match',...
               'Match Number', [1 15]);
            app.UIFigure.Visible ='off';
            app.UIFigure.Visible = 'on';
             data = str2num(x{:});
     % app.Label_4.Text = strcat('Error =
     %',num2str(error_vec(error_vect_orderd(data)));
  while ~data==0
    imshow(reshape(trainingdataset(error_vect_orderd
    (data),:),64,64),[], 'parent' ,app.UIAxes_2);
       app.Label_3.Text =
trainingImages(error_vect_orderd(data)).name;
x = inputdlg('Number of Next best Match :',...
               'Match Number', [1 15]);
              data = str2num(x{:});

                         end
                break;
         end
    end
```