Face Recognition with KNN in MATLAB

By Yangbingxue

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 can protect everyone's personal information from being leaked.

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

1.INTRODUCTION

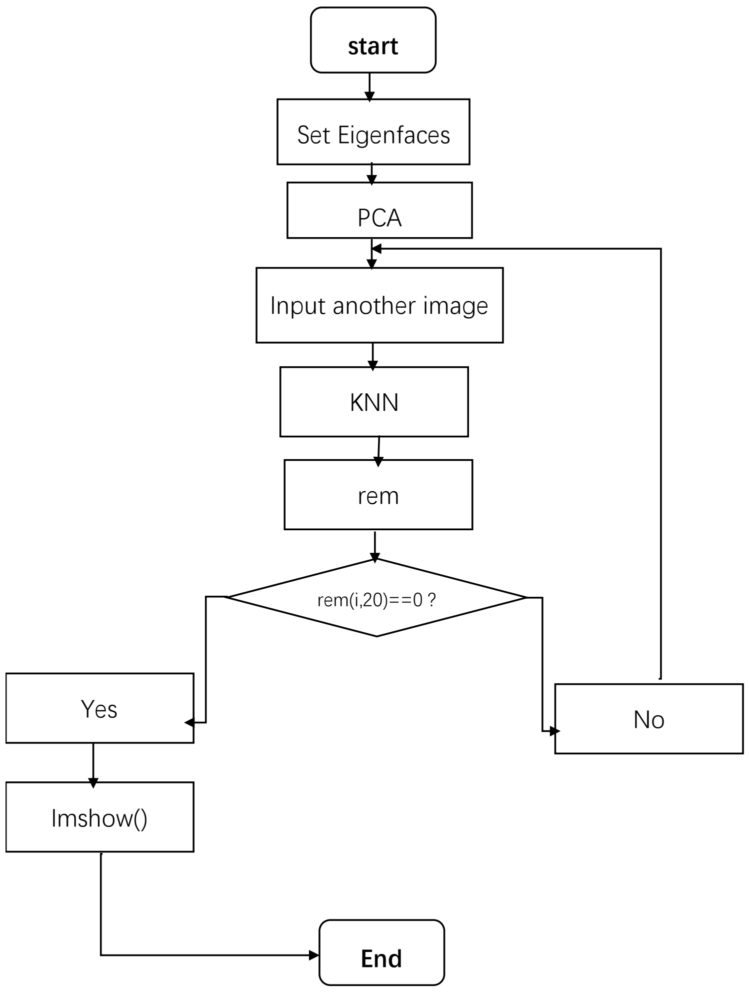
The idea for this paper came up while through gate of the campus and FaceID in Macbook or smartphone, mainly paper is based upon MATLAB comprising image processing. I had read different algorithms based on image processing like Adam's algorithm, fisher face algorithm, Back Propagation Neural Network etc. But I will show the simplest way of implementing a face recognition system using MATLAB , Here no machine learning or Convolutional neural network (CNN) is required to recognize the faces. To keep the face recognition system as simple as possible, I used eigenvector based recognition system.

2.REQUIREMENTS

SYSTEM: CentOS 7.5.1804 Core (RHEL 7.x)
Kernel: x86_64 Linux 3.10.0-862.el7.x86_64
Shell: bash 4.2.46
CPU: Intel Core2 Duo P8600 @ 2x 2.401GHz
RAM: 2358MiB / 3847MiB
GPU: Mesa DRI Mobile Intel® GM45 Express Chipset

SOFTWARE: MATLAB R2018a

3.PROPOSED DESIGN OF MODEL



4.IMPLEMENTATION

1.First obtain a set of people image. Each image is into a vector of size N and placed into the set.

$$S = \{\Psi_1, \Psi_2, \Psi_3, \dots, \Psi_N\} \quad (1)$$

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

$$\Gamma = \frac{1}{N}\sum_{i=1}^{N} \Psi_i \quad (2)$$

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

$$\Delta_i = \Psi_i - \Gamma \quad (3)$$

4.We find N orthogonal unit vectors u_N, these unit vectors are descriptive the distribution Δ (the difference in step 3). The k in u_N $(k = 1,2,3 \dots, N)$

$$\lambda_k = \frac{1}{N}\sum_{i=1}^{N} (u_k^T \Delta_i)^2 \quad (4)$$

λ_k: Eigen value

When λ_k is at least, this u_k is pretty much guaranteed

And u_k also satisfies the flowing equation.

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1 & if\ \ l = k \\ 0 & otherwise \end{cases}$$

5.Actually, to compute the u_k above is to compute the eigen-vectors of the following covariance matrix

$$C = \frac{1}{N}\sum_{i=1}^{N} \Delta_i \Delta_i^T = AA^T \quad (5)$$

($A = \{\Delta_1, \Delta_2, \Delta_3, \dots, \Delta_N\}$)

But for an $M \times M$ dimensional image, equation (5) is difficult to calculate, so we can use the other solution. If the number of image is less than the dimensions of image ($N < M^2$), the number of vectors that matter is $N - 1$,not M, because the eigen-values of the other eigen vectors are 0, solve for the eigen vectors , it just to solve for an $M \times M$ dimensional matrix L,The Mth row of L, the Nth column elements can be represented as

$$L_{mn} = \Delta_m \Delta_n^T \quad (6)$$

Once we know the N eigen vectors of the L, then the eigen vectors of covariance matrix are expressed as

$$u_l = \sum_{i=1}^{N} v_{ml}\, \Delta_m \ \ (l = 1,2, \dots, N) \quad (7)$$

6.Finally, we have to recognition, lth Eigenface that weights can form a vector

$$\Omega^T = [\omega_1, \omega_2, \dots, \omega_N] \quad (8)$$

According to the Euclidean distance

$$\varepsilon_k = \|\Omega - \Omega_k\|^2 \quad (9)$$

5.RECOGNITION PROCEDURE

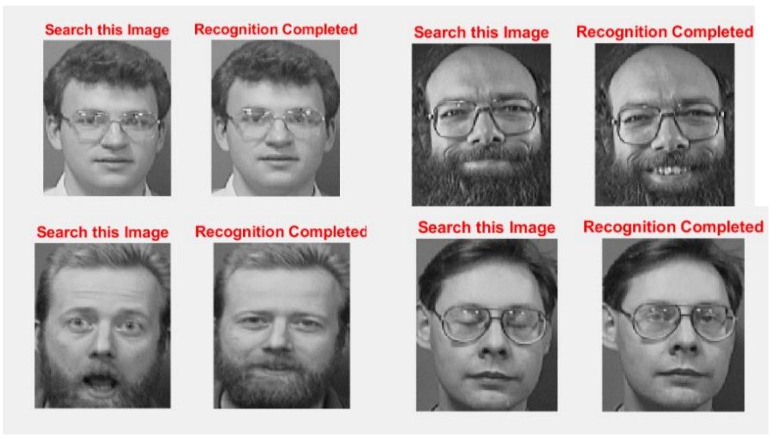
A face image is transformed into its eigen image components. Now according KNN ,we can realize face recognition.

6.MATLAB CODING

```
function output_value = load_database();
persistent loaded;
persistent numeric_Image;
if(isempty(loaded))
    all_Images = zeros(10304,400);
    for i=1:40
        cd(strcat('s',num2str(i)));
        for j=1:10
            image_Container = 
imread(strcat(num2str(j),'.pgm'));
            all_Images(:,(i-
1)*10+j)=reshape(image_Container,size(ima
ge_Container,1)*size(image_Container,2),1);
        end
        display('Doading Database');
        cd ..
    end
    numeric_Image = uint8(all_Images);
end
loaded = 1;
```

```
output_value = numeric_Image;

loaded_Image=load_database();
random_Index=round(400*rand(1,1));
random_Image=loaded_Image(:,random_In
dex);
rest_of_the_images=loaded_Image(:,[1:rand
om_Index-1 random_Index+1:end]);
image_Signature=20;
white_Image=uint8(ones(1,size(rest_of_the_
images,2)));
mean_value=uint8(mean(rest_of_the_image
s,2));
mean_Removed=rest_of_the_images-
uint8(single(mean_value)*single(white_Imag
e));
L=single(mean_Removed)'*single(mean_Re
moved);%求协方差矩阵
[V,D]=eig(L);%求协方差矩阵的特征值
V=single(mean_Removed)*V;
V=V(:,end:-1:end-(image_Signature-1));
all_image_Signatire=zeros(size(rest_of_the_i
mages,2),image_Signature);

for i=1:size(rest_of_the_images,2);

all_image_Signatire(i,:)=single(mean_Remov
ed(:,i))'*V;
end
subplot(121);
imshow(reshape(random_Image,112,92));
title('Looking for this
Face','FontWeight','bold','Fontsize',16,'color'
,'red');
subplot(122);
p=random_Image-mean_value;
s=single(p)'*V;
z=[];

for i=1:size(rest_of_the_images,2)
    z=[z,norm(all_image_Signatire(i,:)-s,2)];

if(rem(i,20)==0),imshow(reshape(rest_of_th
e_images(:,i),112,92)),end;
    drawnow;
end
[a,i]=min(z);
subplot(122);
imshow(reshape(rest_of_the_images(:,i),112,
92));
title('Recognition
Completed','FontWeight','bold','Fontsize',16
,'color','red');
```

7.TEXT RESULT



8.LIMITATION

1.Large amount of calculation
2.Sample imbalance problem
3.Consume a lot memory

9.APPLICATION

1.FaceID in PC and phone
2.Door lock for person

10.CONCLUSION

First we input a image and observed the Euclidean distance that can tells us how close the input image from the image on set. Then according that we know the face whether know.

11.REFERENCES

[1] International Journal of Scientific & Engineering Research, Volume 3, Issue 2, February 2012 5
ISSN 2229-5518
[2] MIT University media laboratory, USA
[3] Olivetti Research Laboratory, UK