Detección de caras por medio del Análisis por Componentes Principales (PCA)

Table of Contents


```
clc
clear all;
```

Imagenes de entrenamiento

```
Im=[];
for i=1:38
    str=strcat('YaleCropped/yale_',int2str(i),'.pgm');
    eval('img=im2double(imread(str));');
    Im(:,:,i)=img;
end
```

Se hace una ecualización del histograma a las imágenes de entrenamiento

```
for i=1:38
    Im(:,:,i) = histeq(Im(:,:,i),256);
end
```

Imagenes de prueba

```
Im_p = [];
numIm = 80;
for i=1:1:numIm
    str = strcat('C:\Users\Jesus\Pictures\Images_DataBases\BioID\BioID_',sprintf('
    eval('Im_p(:,:,i)=im2double(imread(str));')
end
```

De igual forma las imágenes de prueba son ecualizadas

```
for i=1:numIm
    Im_p(:,:,i) = histeq(Im_p(:,:,i),256);
end
```

Imagenes Escaladas

Se hace primero la reducción de las imágenes de prueba a un 20% de su tamaño original, de 286x384 a 58x77

```
escala = 0.2;

Im_p_m = [];
for i=1:numIm
    Im_p_m(:,:,i) = imresize(Im_p(:,:,i),escala);
end

[I,J,L] = size(Im_p_m);
```

Tres diferentes tamaños de las imágenes de entrenamiento dependiendo de la escala de las imágenes de prueba. Se hicieron mediciones de tres imágenes diferentes y se encontró un aproximado de la proporción de la cara con el resto de la imagen para escoger la escala.

```
Im_m = [];
[col, fil] = size(Im_p_m(:,:,1));


% % Imagen 37
% for i=1:38
%     Im_m(:,:,i) = imresize(Im(:,:,i),[fil/3.2, col/3.6]);
% end

% Imagen 16
for i=1:38
    Im_m(:,:,i) = imresize(Im(:,:,i),[fil/3.2, col/2.7]);
end

% % Imagen 70
% for i=1:38
%     Im_m(:,:,i) = imresize(Im(:,:,i),[fil/2.6, col/1.8]);
% end
```

Análisis por Componentes Principales

Para ahorrar espacio en el código se hicieron dos funciones para el análisis por componentes principales, la primera tiene como prototipo:

`[eigValues, eigVectors, avFace, eigFaces, media, weights] = PCA(Imagen,K)`

donde:

- **`eigValues`**: Eigen Valores
- **`eigVectors`**: Eigen Vectores
- **`avFace`**: Cara Media

- **`eigFaces`**: Eigen Caras
- **`media`**: Media
- **`weights`**: Pesos
- **`Imagen`**: Imagenes de entrenamiento
- **`K`**: Número de Autovectores a usar

De pruebas anteriores se obtuvo que al usar 25 eigen vectores, se utiliza el 90% de la información necesaria para la reconstrucción de las imágenes.

```
K = 25;     % Numero de Eigenvectors

[eigValues, eigVectors, avFace, eigFaces, media, weights] = PCA(Im_m,K);

[J,I] = size(avFace);
tamano = (size(Im_p_m(:,:,1))-size(Im_m(:,:,1)));
N = tamano(1);
M = tamano(2);
```

Cara Promedio

```
figure(1)
imshow(avFace,'InitialMagnification',900)
```

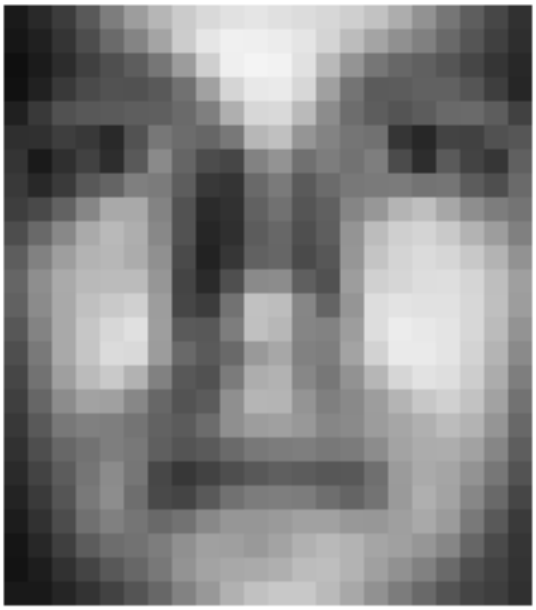


Imagen de prueba

Anteriormente se cargaron varias imágenes de prueba, para mostrar el funcionamiento de este programa se elige una imagen de prueba de la cual se obtuvo uno de los valores de escala, por ejemplo la imagen número 16

```
imagen = 16;
```

Se obtienen (N-J)*(M-I) segmentos de la imagen original de tamaño J x I, donde: [N x M] es el tamaño de la imagen de prueba y [J x I] es el tamaño de las imagenes de entrenamiento. Es decir, para esta escala en particular se obtuvieron 33*55 = 1815 imágenes cada una de tamaño 25x22

```
Im_p_R = [];
tic;    % Inicia el reloj
i=1;
for m=1:M
    for n=1:N
       Im_p_R(:,:,i) = Im_p_m(n:J+n-1,m:I+m-1,imagen);
       i=i+1;
    end
end
```

Error mínimo

Una vez obtenidas todos los segmentos de la imagen de prueba, se pocede a hacer la reconstrucción de cada una para obtener el segmento con el error mínimo el cual, idealmente, correspondrá a la cara de-

tectada. Este proceso se realiza con la segunda función hecha:

[error] = PCA_error(Segmentos, eigVectors, K, media, wights)

donde:

- **error**: Error mínimo con respecto al conjunto de eigen caras
- **Segmentos**: Segmentos de la imagen de prueba
- **K**: Número de Autovectores a usar
- **media**: Media
- **weights**: Pesos

```
for i=1:M*N
    e(i) = PCA_error(Im_p_R(:,:,i),eigVectors, K, media, weights);
end
```

El vector **e** contiene el error mínimo de cada uno de los segmentos probados, es decir este vector es de tamañno (N-J)*(M-I), para este ejemplo el vector de errores es de tamaño 1815

Por último, se encuentra el valor mínimo del vector de errores el cual será el correspondiente al segmento de la imagen original que contiene la cara. La función `[x,y]=min(e)` devuelve el valor del error mínimo (**x**) y la posicón (**y**) de ese valor en el vector `e`, de esta forma es fácil conocer el segmento de la imagen que nos intereza.

```
[x,y] = min(e);
```

```
disp('El minimo error fue de');disp(x);
segundos = toc;     % Paro del reloj
Tiempo = segundos/60
disp('Minutos')
```

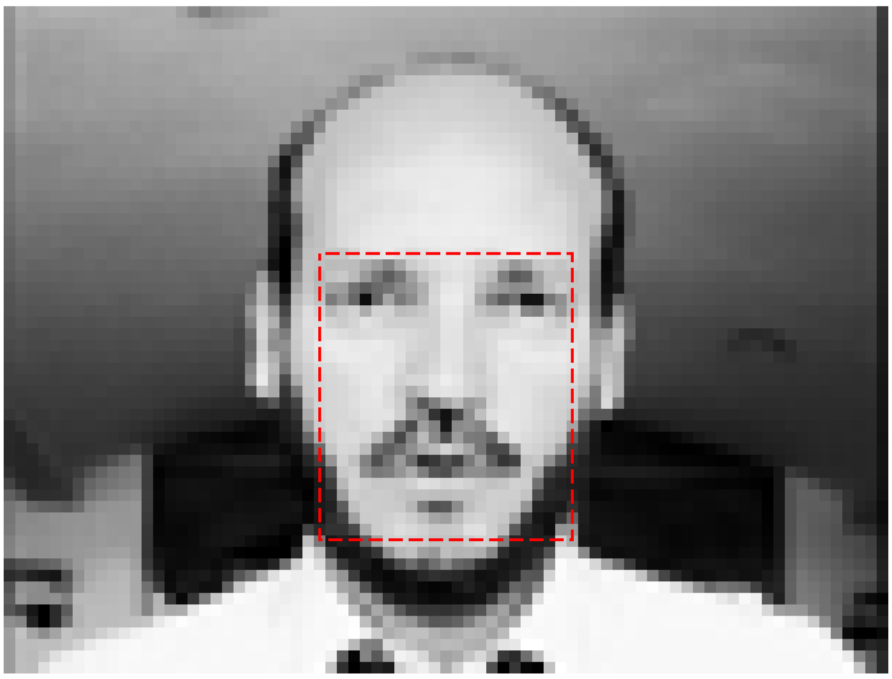
```
El minimo error fue de
    4.2101


Tiempo =

    0.2261

Minutos
```

Se midió el tiempo de procesamiento a partir de la obtención de los segmentos de la imagen original hasta la obtención del error mínimo.

Recuadro

Teniendo la información de la posición de la cara encontrada, se dibuja un recuadro al rededor de ésta con la función `patch`. El recadro fue dibujado sobre la imagen de prueba escalada, pero puede dibujarse también sobre la imagen de prueba original con las cuentas apropiadas que en éste código aparecen como comentarios.

```
figure(2)
imshow(Im_p_m(:,:,imagen),'InitialMagnification',800)

% P = I/escala;
% Q = J/escala;

a = floor(y/N);
b = (y-a*N);

% a = a/escala;
% b = b/escala;

% Vertices
verts = [a b; a+I b;  a+I b+J; a b+J];

% Cuadro
faces = [ 1 2 3 4];

% Propiedades del recuadro
caraEncontrada.Vertices = verts;
caraEncontrada.Faces = faces;
caraEncontrada.FaceColor = 'none';
caraEncontrada.LineStyle = '--';
caraEncontrada.Edgecolor = 'red';
caraEncontrada.LineWidth = 2;

patch(caraEncontrada);
```



Published with MATLAB® 7.11