Eigenfaces, an application of SVD

Angelo Barbera

February 1, 2023

Outline

Eigenfaces

The *eigenfaces* are a set of eigenvectors used in the problem of human face recognition. The method that makes use of eigenfaces for recognition was developed by Sirovich and Kirby in 1987.

In order to compute eigenfaces it is necessary to flatten each image into a column vector and create a matrix using these vectors.

Next we compute the mean of the matrix over the rows to calculate the "average face", and subtract the mean from each column to zero-mean center the data.

Finally we can compute the Singular Value Decomposition of the data matrix obtaining the eigenvectors of the covariance matrix, which are also called eigenfaces.

The eigenfaces express the directions of greatest variance and can be used to project the input data onto the space generated from the eigenvectors related to the directions of greatest variance allowing to reduce the memory used to store the images.

Principal Component Analysis

The method previously described is based on Principal Component Analysis (PCA) that is a technique for reducing the dimensionality of a dataset.

Given a set of data $\{x_1, x_2, \ldots, x_n\}$ where $x_i \in \mathbb{R}^d$, we need to find a base of vectors $W = \{w_1, \ldots, w_k\}$ where $w_i \in \mathbb{R}^d$, such that the variance of

$$y_1 = \begin{bmatrix} w_1^T x_1 \\ \vdots \\ w_k^T x_1 \end{bmatrix}, \ldots, y_n = \begin{bmatrix} w_1^T x_n \\ \vdots \\ w_k^T x_n \end{bmatrix} \tag{1}$$

is maximized. Considering the one dimensional case (k = 1) for semplicity, the variance is

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (w_1^T x_i - \mu)^2 \tag{2}$$

$$\mu = \frac{1}{n} \sum_{i=1}^{n} w_1^T x_i = w_1^T \frac{1}{n} \sum_{i=1}^{n} x_i = w_i^T \overline{x} \tag{3}$$

where μ is the mean of the new coordinates and $\overline{x}$ is the mean vector.

Principal Component Analysis

Substituiting (3) into (2) we get

$$\frac{1}{n}\sum_{i=1}^{n}(w_1^T x_i - w_1^T \overline{x})^2 = \frac{1}{n}\sum_{i=1}^{n}(w_1^T (x_i - \overline{x}))^2 =$$
$$= w_1^T \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T \right) w_1 = w_1^T C\, w_1 \qquad (4)$$

where C is the covariance matrix. To maximize (4) we need to find the largest eigenvector of C in the one-dimensional case and the k largest eigenvectors $W = \{w_1, \ldots, w_k\}$ of C, in the k-dimensional case.

Singular Value Decomposition

Given a matrix $A \in \mathbb{R}^{m \times n}$ it can be decomposed as

$$A = U \Sigma V^T \tag{5}$$

where $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times n}$

U and V are orthogonal matrices containing respectively the eigenvectors of AA^T (left singular vectors) and $A^T A$ (right singular vectors).

Σ is such that

$$\Sigma_{ij} = \begin{cases} 0 & i \neq j \\ \sigma_i & i = j \end{cases} \tag{6}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, $p = min(m, n)$ and $\sigma_i = \sqrt{\lambda_i}$ where λ_i are the eigenvalues of $A^T A$.

Computing Eigenfaces

If the data are zero mean-centered, considering the k-dimensional case, we can rewrite (4) as

$$W^T(XX^T)W \tag{7}$$

therefore it is possible to calculate the k largest eigenvectors computing SVD of X

$$X = U\Sigma V^T \tag{8}$$

where U contains the eigenvectors of XX^T, and selecting only the first k left singular vectors.

We can project new faces to the first k eigenfaces to store data using less memory in the following way

$$R = U_k^T X \tag{9}$$

Where U_k is a dxk matrix containing the first k eigenfaces and X is a dxn matrix containing the flattened images, where n is the number of images and d is the height and width product of each image. The result is a kxn matrix with $k \ll d$.

Recovering Images

We can recover original images using the k eigenfaces as a base for linear combinations of the reduced size images.
Given $U_K \in \mathbb{R}^{dxk}$ containing the first k eigenfaces, $R \in \mathbb{R}^{kxn}$ containing the images projected into the space generated from the k eigenfaces, and $\bar{X} \in \mathbb{R}^{dx1}$ containing the flattened mean face, let's recover an image as follows

$$I = U_K \mathbf{r}_i + \bar{X} \tag{10}$$

Where $\mathbf{r}_i$ is the i-th column vector of R corrisponding to the image to reconstruct.
Finally we have to reshape I as a matrix having the same dimensions of the original image in order to display the recovered image.

Implementation

The following functions were used to implement the previously described algorithm and to analyze results:

- *flatten* : returns the matrix X containing the flattened images.
- *reduce* : returns the matrix U_K containing the first k eigenfaces, the matrix R containing the reduced images, the matrix M containing the mean face.
- *recover_image* : returns the recovered image from reduced data size.
- *space_used* : shows the space used to store the images as the captured information varies.
- *eigenfaces_used* : shows the eigenfaces used as the captured information varies.

The dataset used is "The ORL Database of Faces" created by AT&T Laboratories Cambridge. This datased contains 400 images, the size of each image is 112 x 92 with 256 grey levels per pixel.

flatten

```
function [X] = flatten(image_path, n, height, width)
% Returns X given in input image_path, n, height, width.
% The name of the image must be "img (i)" where i is
% the number of the image.
% The file extension of the image must be ".pgm".
% image_path is the path of the images.
% n is the number of the images.
% height is the heigth of each image.
% width is the width of each image.
% X is the matrix containing the flattened images.

% dimension of each image
d = height*width;

% computing X
X = zeros(d,n);

for i=1:n
    filename = strcat(image_path, 'img (', num2str(i), ').pgm');
    image = imread(filename);
    image_double = im2double(image);
    X(:, i) = reshape(image_double, [d, 1]);
end
```

reduce

```
function [U_K, R, M] = reduce(X, captured_info_tol)
% Returns the matrix U_K, R, M given in input X, height, width and
% captured_info_tol.
% X is the matrix containing the flattened images.
% captured_info_tol (between 0 and 1) is the desidered information
% to be captured.
% U_K is the matrix containing the first k eigenfaces.
% R is the matrix containing the flattened reduced images.
% M is the matrix containing the mean face.

% computing matrix M
M = mean(X, 2);

% zero mean centering of X
X_zero_mean = X - M;

% Singular Value Decomposition
[U, S, ~] = svd(X_zero_mean, 'econ');
```

reduce

```
% Selecting the number of eigenvectors to capture the
% desidered information
S_diag = diag(S).^2;
captured_info = 0;
k = 1;
while(captured_info < captured_info_tol)
    captured_info = sum(S_diag(1:k)) / sum(S_diag);
    k = k + 1;
end

% selecting only the first k eigenvectors
U_K = U(:, 1:k);

% computing the reduced images
R = U_K' * X_zero_mean;
```

recover_image

```
function [I] = recover_image(U_K, R, M, i, height, width)
% Returns the recovered image from reduced size data
% given U_K, R, M, i, height, width.
% U_K is the matrix containing the first k eigenfaces.
% R is the matrix containing the flattened reduced images.
% M is the matrix containing the mean face.
% i is the index of the image to be recovered.
% height is the heigth of each image.
% width is the width of each image.
% I is the recovered image.

% recovering the flattened zero mean centered image
image_zero_mean = U_K * R(:, i);

% adding mean faces to zero mean centered image
image_flattened = image_zero_mean + M;

% reshaping the image to original size
I = reshape(image_flattened, [height, width]);
```

space_used

```
function [x, y] = space_used(X)
% Returns the vectors relative to the memory used to
% store the images as the captured information varies.
% X is the matrix containing the flattened images.
% x is the array containing the captured information.
% y is the array containing the saved space.

% initializing arrays
x = zeros(1, 9);
y = zeros(1, 9);

for i=1:1:9
    captured_info = i / 10;

    % reducing the images
    [U_K, R, M] = reduce(X, captured_info);

    % computing size of matrices
    size_X = size(X);
    size_R = size(R);
    size_U_K = size(U_K);
    size_M = length(M);
```

space_used

```
    % computing used space, reduced space and saved space
    original_used_space = size_X(1) * size_X(2);
    reduced_used_space = size_R(1) * size_R(2) + size_U_K(1) *
    size_U_K(2) + size_M;
    saved_space = round((1 - reduced_used_space /
    original_used_space) * 100);

    x(i) = round(captured_info * 100);
    y(i) = saved_space;

end

% plotting results
bar(x, y);
xlabel('Captured information (%)');
ylabel('Saved space (%)');
grid on
```

eigenfaces_used

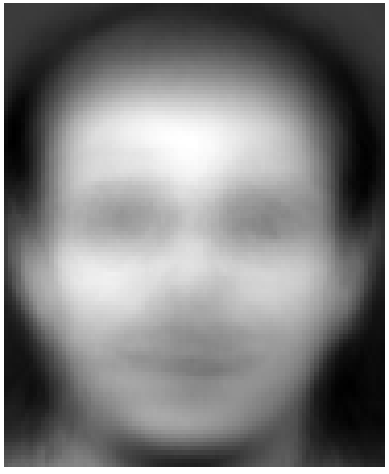
```
function [x, y] = eigenfaces_used(X)
% Returns the vectors relative to the eigenfaces used
% as the captured information varies.
% X is the matrix containing the flattened images.
% x is the array containing the captured information.
% y is the array containing the saved space.

% initializing arrays
x = zeros(1, 9);
y = zeros(1, 9);

for i=1:1:9
    captured_info = i / 10;

    % reducing the images
    [U_K, ~, ~] = reduce(X, captured_info);
    size_U_K = size(U_K);

    eigenfaces_used = size_U_K(2);

    x(i) = round(captured_info * 100);
    y(i) = eigenfaces_used;

end
```

eigenfaces_used

```
% plotting result
plot(x, y);
xlabel('Captured information (%)');
ylabel('Eigenfaces used');
grid on
```

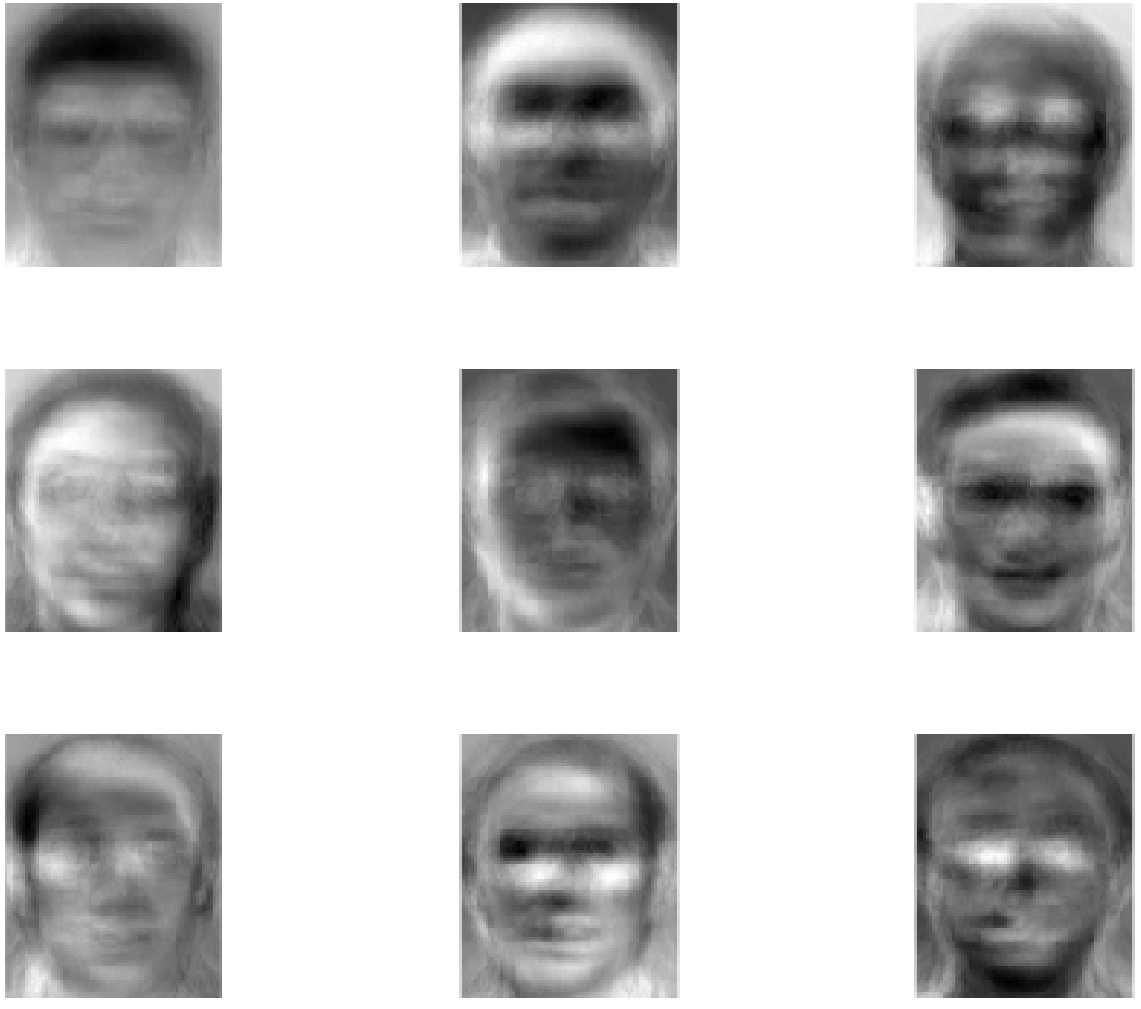
Mean Face

The following image is the mean face, it is composed of the average features of the images in the dataset.



Eigenfaces

The following images are the first 9 eigenfaces, they represent the main features of the images in the dataset.

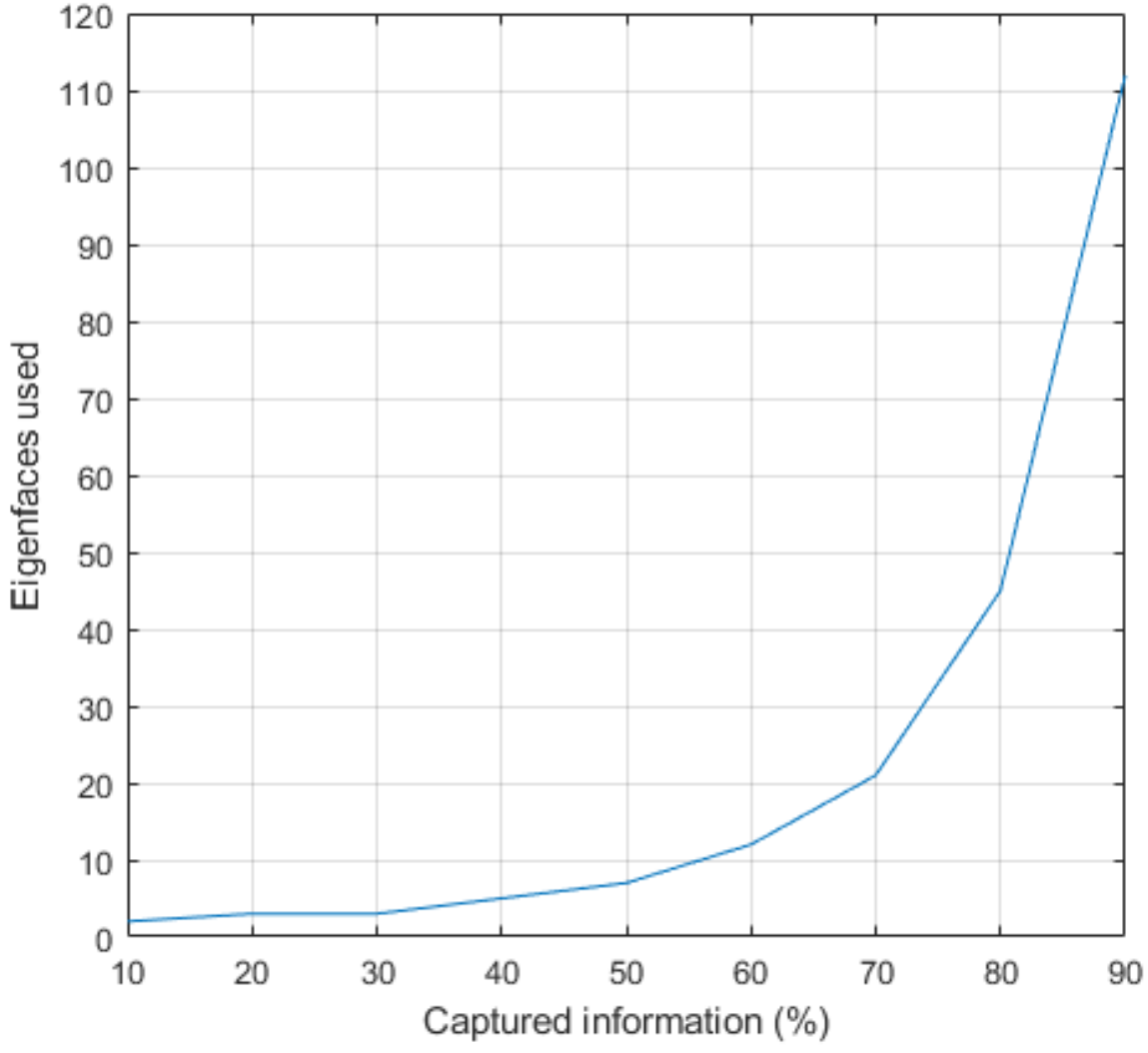


Recovered image

The picture on the left is the original image, the picture on the right is the recovered image from reduced data with 90% of captured information, the picture below is the recovered image from reduced data with 95% of captured information.

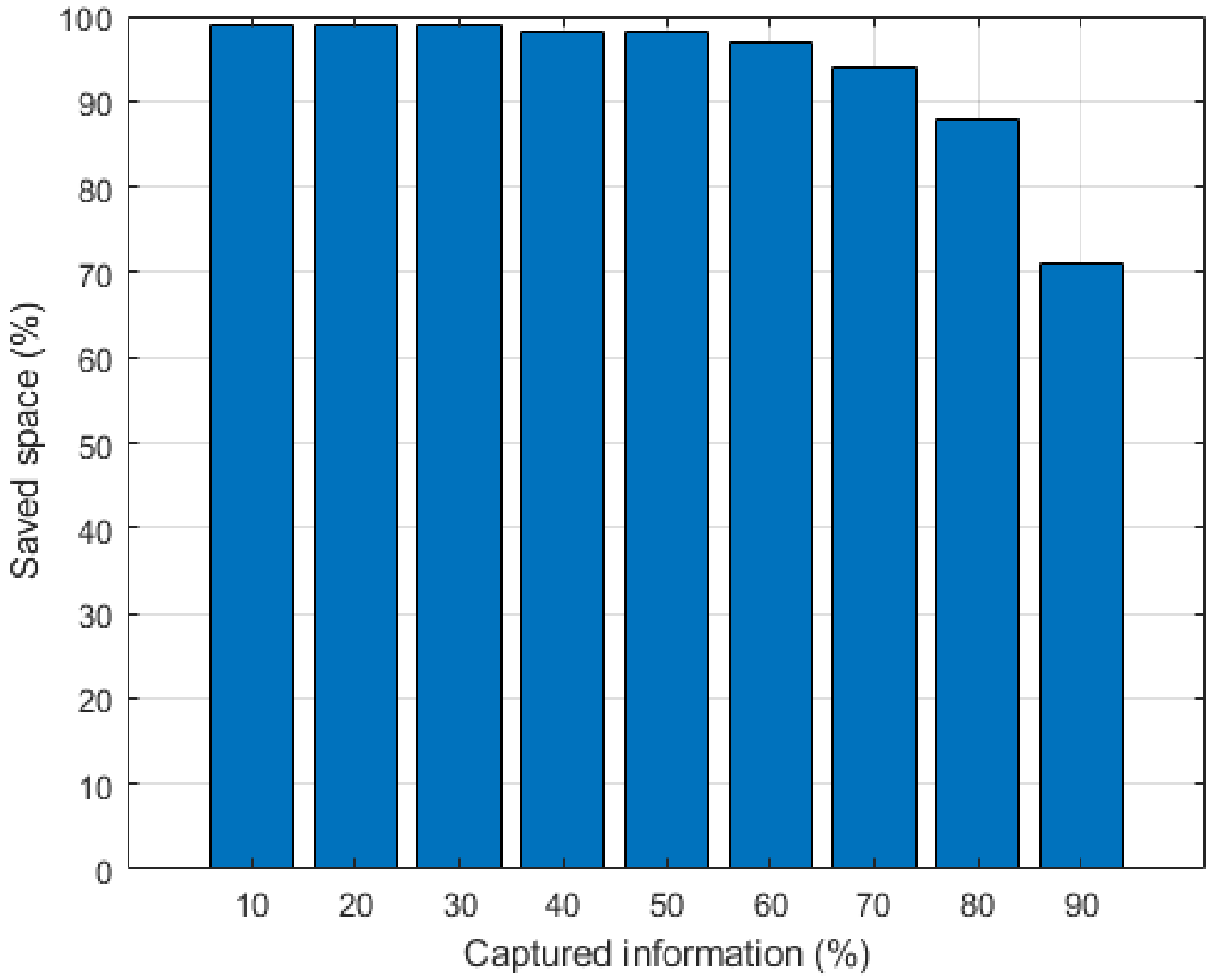
Eigenfaces used

The following image shows the eigenfaces used as the captured information varies.



Saved space

The following image shows the saved space as the captured information varies.



Face Recognition

The face recognition can be implemented computing the matrices U_K, R, M using a training set and using an image belonging to a different set to recognize the image as a face belonging to the training set.
Given the d x 1 vector i (corresponding to the flattened image having size h x w) and the matrices U_K, R, m (the flattened mean face), it is possible to zero mean center i and project it into the space generated by the first k eigenfaces as follows

$$r = U_K^T(i - m) \tag{11}$$

To recognize the face we need to find the image $l \in R$ that minimizes $e = \|r - l\|_2$
Dataset of 99 images of 11 people, 9 image per person for training Dataset of 11 images of 11 people, 1 image per person for testing

Implementation

The following functions were used to implement the algorithm :

- flatten_img : returns the vector *img* containing the flattened image
- reduce_img : returns the vector *r* containing the flattened reduced image
- recognize_img : returns the image found

The training set is composed of 99 images of 11 people (9 images per person), while the testing set consists of 11 image (1 image per person). The dataset used is extracted from "The ORL Database of Faces" created by AT&T Laboratories Cambridge.

flatten_img

```
function [img] = flatten_img(image_path, i, height, width)
% Returns img given in input image_path, i, height, width.
% The name of the image must be "img (i)" where i is the
% number of the image.
% The file extension of the image must be ".pgm".
% image_path is the path of the image
% i is the number of the image
% height is the heigth of the image
% width is the width of the image
% img is the vector containing the flattened image

% image size
d = height*width;

% computing img
img = zeros(d, 1);

    filename = strcat(image_path, 'img (', num2str(i), ').pgm');
    image = imread(filename);
    image_double = im2double(image);
    img(:, 1) = reshape(image_double, [d, 1]);
```

reduce_img

```
function [r] = reduce_img(img, M, U_K)
% Returns the vector r given in input img, M, U_K.
% img is the vector containing the flattened image
% M is the matrix containing the mean face
% U_K is the matrix containing the first k eigenfaces
% r is the vector containing the flattened reduced image

% zero mean centering
img_zero_mean = img - M;

% computing the reduced image
r = U_K' * img_zero_mean;
```

recognize_img

```
function [img] = recognize_img(U_K, R, M, r, height, width)
% Returns img given in input U_K, R, M, r, height, width.
% U_K is the matrix containing the first k eigenfaces
% R is the matrix containing the flattened reduced images
% M is the matrix containing the mean face
% r is the image to be recognized
% height is the heigth of the image
% width is the width of the image.
% img is the image found

% searching the image
n = size(R, 2);
error = inf;
for i=1:n
    new_error = norm(r - R(:, i), 2);
    if new_error < error
        error = new_error;
        k = i;
    end

    % recovering the image
    img = recover_image(U_K, R, M, k, height, width);
end
```

Results

The picture on the left is the image found, the picture on the right is the original image.

Image found

Original image

References

- cs.nju.edu.cn/zlj/Course/DM_16_Lecture/Lecture_2.pdf
- nextjournal.com/yuewangpl/svd-and-eigenfaces
- cam-orl.co.uk/facedatabase.html
- M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, vol.3, no. 1, pp. 71-86, 1991