Technion - Israel Institute of Technology

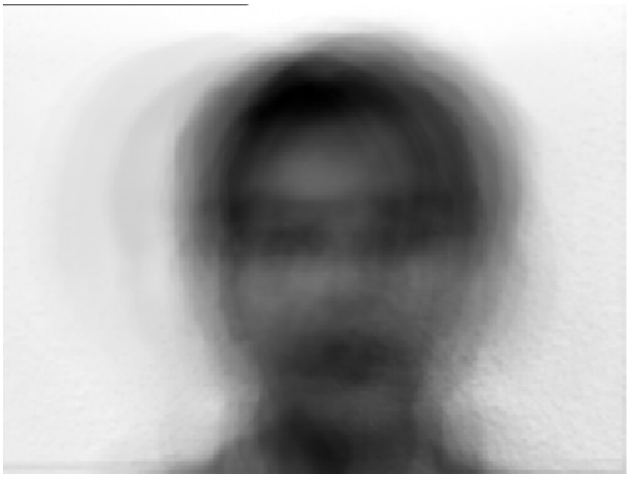
Hw_3 - PCA and Image Stitching

Chen Katz, ID 203511043

Daniel Engelsman, ID 300546173

---

1. Prepare the training set:
   a. The training set is loaded using the function `readYaleFaces.m` (line 3 on the given code). Read the documentation for more details.
   b. Compute the mean face of the trainset and subtract it from all the training images.
   c. Show an image representing the mean face.



c. Training set mean face

2. Compute the eigen-faces:
   a. Implement a function that finds the r largest eigen-vectors of the trainset covariance matrix: $\Sigma = E\left[(X-\mu)(X-\mu)^T\right]$

      To avoid memory issues, use SVD as describes in the tutorial, or read the relevant section in [1]. Implement your algorithm accordingly and explain in detail.
   b. Show the **first** five eigen-faces as images (corresponding to the **largest** eigenvalues).

a. Let a face image be a two-dimensional $m \times n$ array of 8-bit intensity values, stored as a vector column ($\equiv \Gamma_i \in \mathbb{R}^{m \times n}$) in a $M = 150$ training images matrix $\equiv A \in \mathbb{R}^{(m \cdot n) \times M}$. As a pre-processing stage, we shall subtract from each image the mean face ($\equiv \psi$) :

$$\Phi_i = \Gamma_i - \psi \qquad \text{where} \qquad \psi = \frac{1}{M}\sum_{k=1}^{M}\Gamma_k$$

$$\text{such that :} \qquad \lambda_k = \frac{1}{M}\sum_{j=1}^{M}(u_k^T \Phi_j)^2$$

We define the mean-subtracted face matrix as $\chi \equiv A - \psi$, and the eigenvalues and eigenvectors as [λ_k, u_k] of the corresponding covariance matrix :

$$C = E\left[(X-\mu)(X-\mu)^T\right] = \frac{1}{M}\sum_{k=1}^{M}\Phi_k \Phi_k^T \;\Leftrightarrow\; \chi\chi^T \in \mathbb{R}^{(mn)\times(mn)}$$

Unfortunately, $(mn)^2$ turns to be a huge number, such that calculating that covariance is intractable. We therefore utilize the presented method in the paper, by first solving the eigenvectors of an $M \times M$ matrix - (150^2 vs. $77,760^2$). Then taking the appropriate combinations of the face images Φ_i consider the eigenvectors v_i of $\chi^T\chi$:

$$\chi^T\chi\, v_i = \gamma_i v_i \quad \backslash \cdot(\chi) \quad \rightarrow \quad (\chi\chi^T\chi)\, v_i = \gamma_i(\chi\, v_i)$$

That system is equivalent to a new linear system :

$$(\chi\chi^T\chi)\, v_i = \gamma_i(\chi\, v_i) \quad \Leftrightarrow \quad (\chi\chi^T)\, \chi v_i = \gamma_i(\chi v_i) \quad \Leftrightarrow \quad C'\, \mathrm{v}_i = \gamma_i \mathrm{v}_i$$

Whose eigenvectors are - χv_i , and the system's matrix is a covariance that satisfies :

$$R = \mathrm{rank}(C') \;\leq\; M \ll (mn)^2$$

Its eigenvectors can be obtained by SVD decomposition (in a descending eigenvalues) :

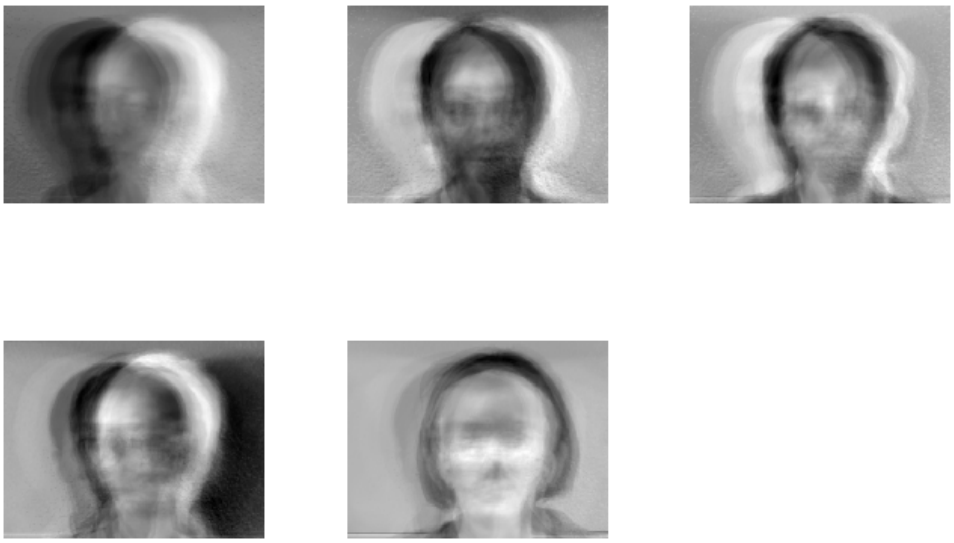
$$\chi = U\Sigma V^T \quad \rightarrow \quad C' = \chi^T\chi = V\Sigma^T\Sigma V^T = V\Lambda V^T$$
$$\text{Top r eigenvectors :} \qquad W = \chi \cdot V \quad \in \ \mathbb{R}^{(m\cdot n)\times \mathbf{R}}$$

W is an orthonormal matrix with eigenvectors of - C' covariance, and R is its valid rank. It's a much smaller image basis that can represent the majority of the Γ_i faces.

b. The first 5 eigen-faces ($=W_{1:5}$) can be presented by the eigenvectors of C' w.r.t to χ :

$$\chi(:,\, 1:5) \cdot W_{1:5} \quad \in \quad \mathbb{R}^{(m\cdot n)\times \mathbf{5}}$$

5 Eigenfaces of train images



3. Reconstruction:
   a. For each image in the train set $\{x_j\}$, subtract the average face (calculated in 1.b) and calculate its low dimension representation vector $y_j = W^T\left(x_j - \bar{x}\right)$ according to the $r = 25$ largest eigen-faces (W is a matrix build from the top r eigen-faces).

The low-dimension subspace corresponds to the first 25 eigen-faces ($=W_{1:25}$) as such :

$$Y_{\lambda(1:25)} = \sum_{j=1}^{25} y_j = \sum_{j=1}^{25} \chi(:,\, j) \cdot W_{1:25} \quad \in \quad \mathbb{R}^{(m\cdot n)\times \mathbf{25}}$$

b. Reconstruct each of the images in the train set $\{x_j\}$ according to its representation y_j. Compute the following representation error:

Reconstructed images (x_j) is obtained by multiplying the eigen-faces by the training set :

$$\hat{x}_j = \psi + y_j W_{(j,:)} \quad \Leftrightarrow \quad X_{rec.}^{train} = \psi + Y_{\lambda(1:25)} W_{1:25}$$

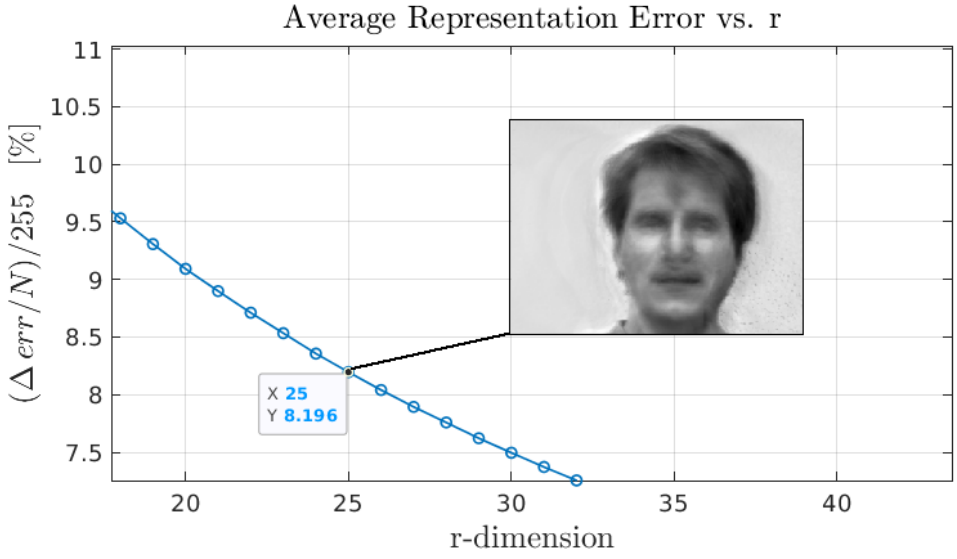
The following definition was utilized in context of RMSE, normalized by maximal 8-bit :

$$\mathrm{RMSE}(j) = \sqrt{\frac{1}{N}(\Gamma_j - \hat{x}_j)^T(\Gamma_j - \hat{x}_j)} \cdot \left(\frac{100}{255}\right)$$

This metric is a second moment (@ origin) of the error, and thus incorporates both the variance of the estimator and its bias. The reconstruction's quality highly depends on r :

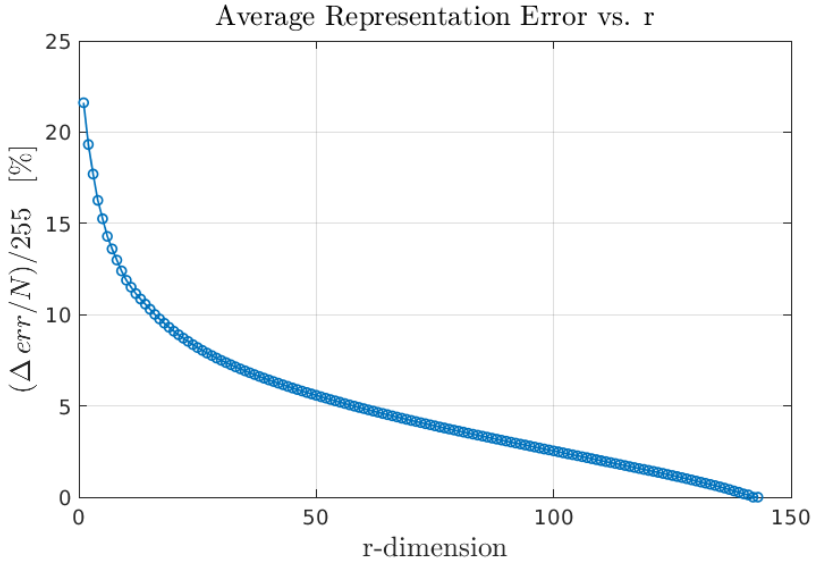
r = 10 r = 75 r = 140

Unsurprisingly, the less the dimension reduction the better the reconstruction is, and vice versa. Here, using r=25, we get a representation error of 8.196 [%] :



c. Describe and explain your results. What is the average representation error? Could we have foreseen it (hint: eigenvalues)?

Let us examine the training set average error w.r.t to the dimensionality reduction :

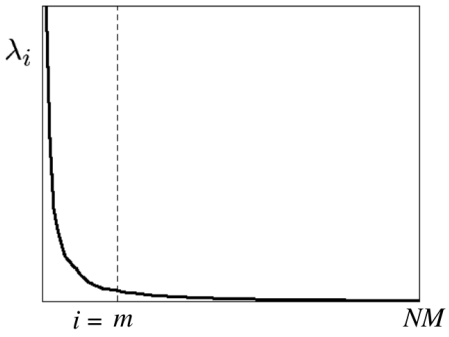


Overall, we can see that the more eigenvectors are taken, the better the reconstruction accuracy is. However, note that I limited the eigenvalues under the following constraint :

$$R = \text{rank}(C') \;\leq\; M \quad \Rightarrow_{usually} \quad R < M$$

M is the small covariance (C') dimension, and its smallest eigenvalues satisfy ($\lambda_M \approx 0$) :

$$\Sigma \equiv \begin{bmatrix} \sigma_{\tilde{1}} & & & 0 \\ & \ddots & & \\ & & \sigma_{\tilde{r}} & \\ 0 & & & \ddots \\ & & & 0 \end{bmatrix} \Rightarrow$$

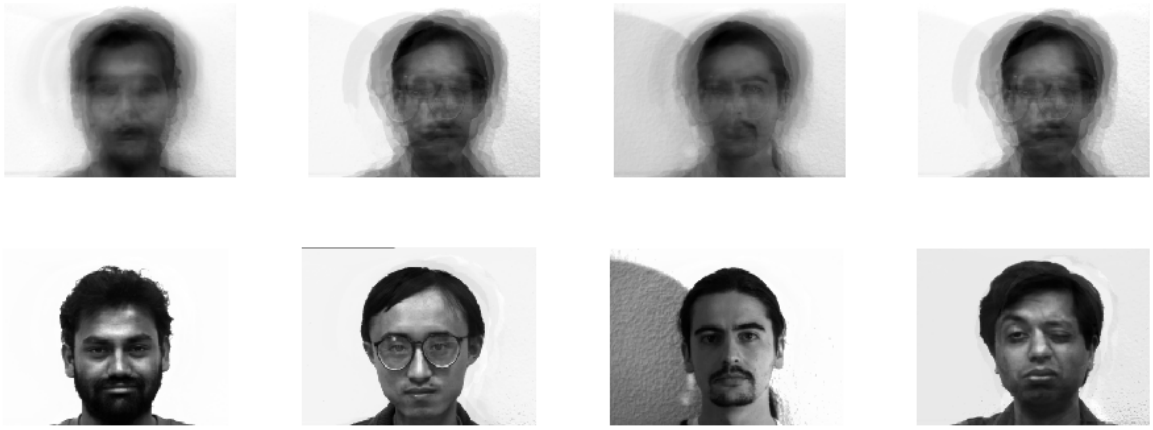


Therefore, we could foreseen the error rate by focusing on eigenvalues with high variance in the direction of the eigenface, and ignore the low ones.

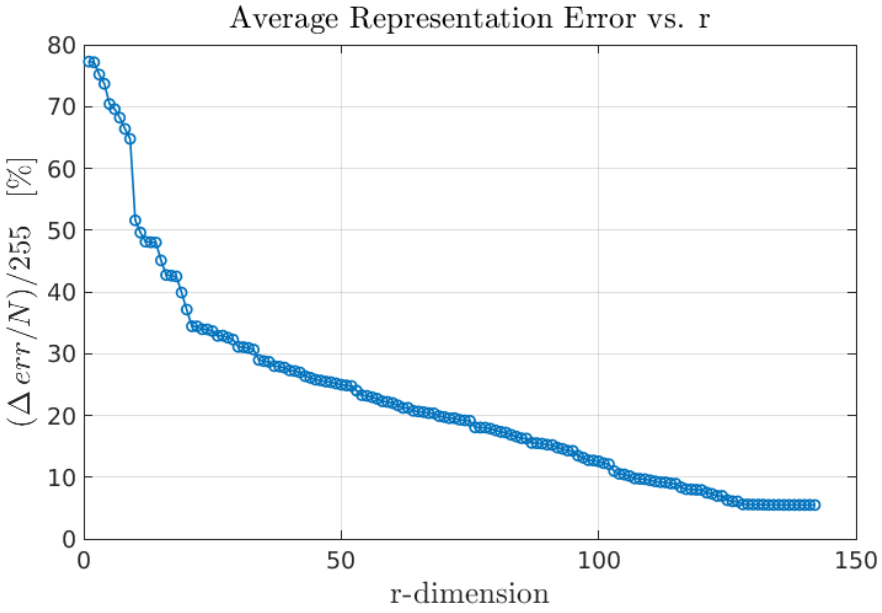
4. Recognition:
   a. For each image in the **Test set** $\{x_i\}$, find the representation vector y_i according to the $r = 25$ top eigen-faces (don't forget to subtract the train set average face). What is the mean representation error? Was there any change in comparison to the train set? Explain. Demonstrate a good reconstruction and a bad one.

After extracting 11 valid samples (=people), we'll subtract the mean training set, and use the same relations as before regarding the representation vector y_i and the reconstruction :

$$\chi_{test} \equiv A_{test} - \psi_{train} \quad \Rightarrow \quad X_{rec.}^{test} = \psi_{train} + W_{1:\mathbf{r}} Y_{\lambda(1:\mathbf{r})}$$



The upper row (r=25) and the bottom row (r=125) demonstrate a good vs. bad reconstruction, depending on r. Examining the test set error over the r-spectrum :



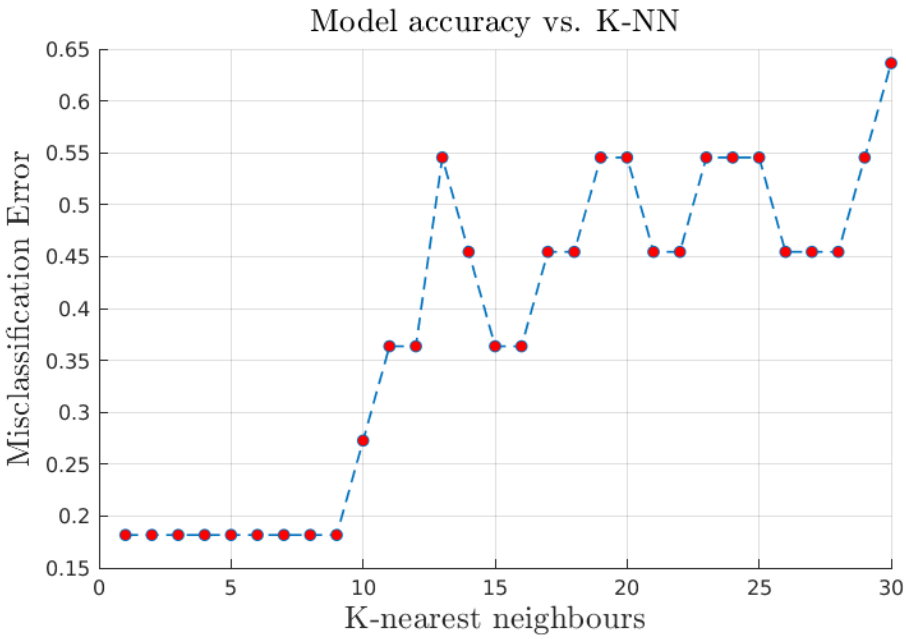
Unlike the training set (N=150), here the test set is almost 15 times smaller (N=11). Therefore it is not surprise to see higher variance rate when the sample size is smaller.

b. Classify each image from the test set (use only the images of people who appear also on the train set – use the face_id parameter). This can be done by using nearest neighbor classifier between each y_i to the train set represent vectors $\{y_j\}$. For nearest neighbor classification, use the matlab function `fitcknn.m`. What are the success ratios (for face_id)?

Using *fitcknn.m* function, we'll calculate the succes ratio vs. the k-nn - ($\frac{f_{knn}(X_{test})}{Y_{test}}$) :

```
KNN = 1  - success ratio : 0.81818
KNN = 2  - success ratio : 0.81818
KNN = 3  - success ratio : 0.81818
KNN = 4  - success ratio : 0.81818
     ⋮           ⋮           ⋮
KNN = 26 - success ratio : 0.54545
KNN = 27 - success ratio : 0.54545
KNN = 28 - success ratio : 0.54545
KNN = 29 - success ratio : 0.45455
KNN = 30 - success ratio : 0.36364
```

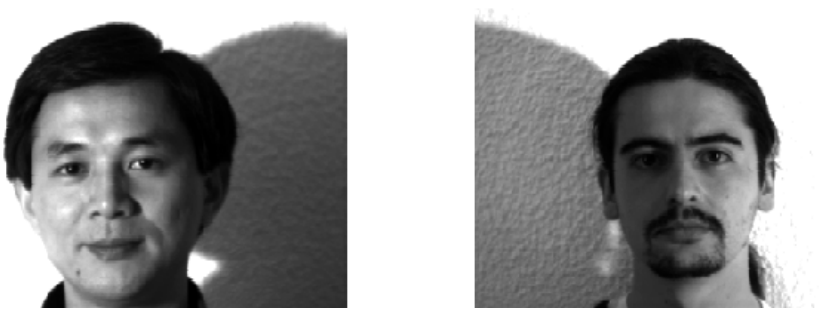
Equivalently, we can span the wrongful prediction ratio, and see the K-nn upper limit :



c. Explain why each of the unsuccessful classifications failed

Taking a representative prediction example so we can find the wrongful labelling :

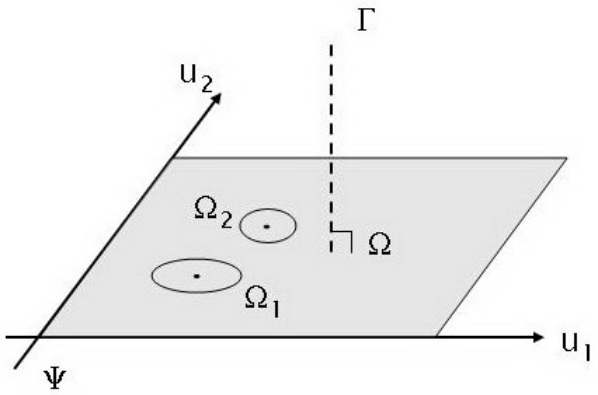
```
X_test_label =
     1     2     4     6     7     8     9    10    12    13    15

K_nn_prediction =
     1     2     4     3     7     8     7    10    12    13    15
```



Contrarily to other images in the test, these images have a significant shade background that's interfering with the image data, and perhaps leads to misclassification.

d. Propose an algorithm that will improve the results obtained here. (tip: use the eigenfaces as an input to the new algorithm)

Based on the following article, a new test face (x_i) is projected into the face space by the set of significant eigenvectors ($W_{1:r}$), and represents a new face in the face subspace :



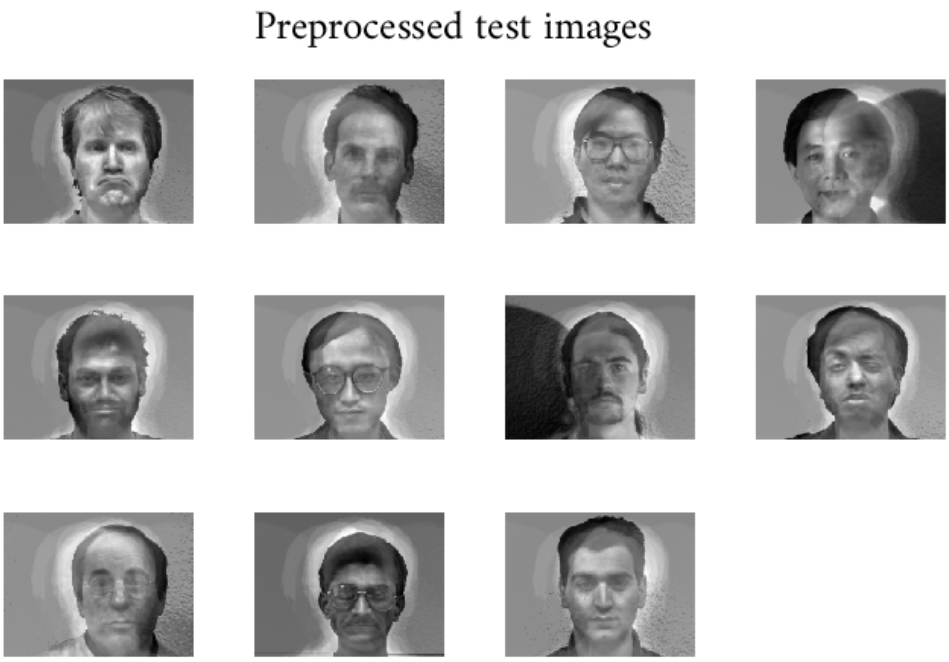
That is to say that each projection is a point in the r-dimensional face subspace. Determining the face class x_i can be obtained by minimizing the Euclidean distance :

$$\epsilon_k = \min\Big(\big\|\Omega(:, k) - \Omega_k\big\| \Big)$$

The minimal distance is equivalent to the highest similarity between two eigenfaces, hence the likeliest labelling. Here is the implemented algorithm using the an input eigenfaces :

```
function X_test_prediction = improved_Recognition(X_train_subt, X_test_subt, W, train_face_id)

n_tests_imgs  = size(X_test_subt,  2);
X_test_Proj   = X_test_subt * W(1:n_tests_imgs, :) * W(1:n_tests_imgs, :)';
X_train_Proj  = X_train_subt * W * W';
n_train_imgs  = size(X_train_Proj, 2);

for i = 1 : n_tests_imgs
    for j = 1:n_train_imgs
        eigenface_dist(i, j) =  norm( X_test_Proj(:, i) - X_train_Proj(:, j)  );
    end
    [~, min_distance] = min( eigenface_dist(i, :) );
    X_test_prediction(1, i) = train_face_id(min_distance);
end
```

Unfortunately it only manages an accuracy as good as the K-nn model : 0.81818 .

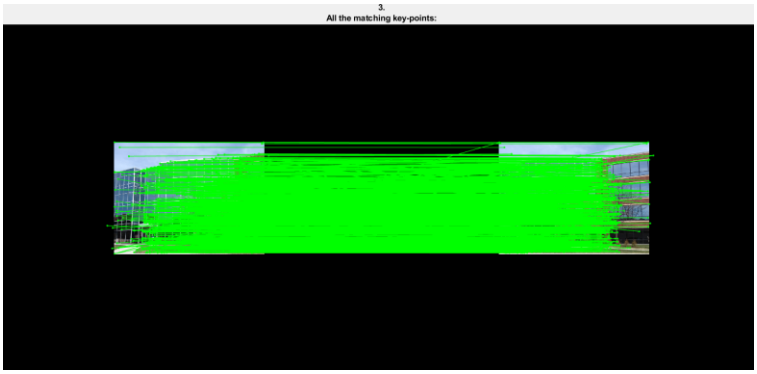


Again, apparently because of the shading background of images[4, 7].

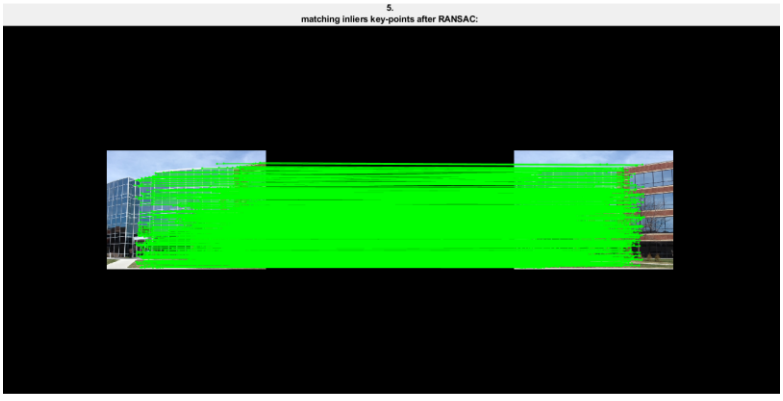
Question 2:

1)
The image pair we choose:

2-3)
I padded the images with zeros and then I searched for key-points:



4-5)



The projective transformation that I found:

```
1.2861   -0.0151 -214.6231
0.1499    1.1585 -135.2101
0.0002    0.0000    1.0000
```

The minimum number of points required for calculating the projective transform is 4 pairs of matching points.

Explanation:
the relation between every pair of matching point and the projection matrix is:

p'= H*p

$$\begin{bmatrix} u_i' \\ v_i' \\ w_i' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad x_i' \to \frac{u_i'}{w_i'} \qquad y_i' \to \frac{v_i'}{w_i'}$$

<=>

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i`x_i & -x_i`y_i & -x_i` \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i`x_i & -y_i`y_i & -y_i` \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ 1 \end{bmatrix} = \begin{bmatrix} \vdots \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

From every pair of matching point 2 equations are obtained, for finding the 8 parameters we need 8 equations, so in order to be able to solve we need at least 4 pairs of matching points.

6-7)

image warping steps:

- find from where each pixel in the projected image comes in the original image:
    - finding the inverse transformation.
    - create a vector of all the image 1 homogeneous coordinates
    - apply the transformation we found on image 1 coordinates
- the values of where each pixel in the projected image comes in the original image aren't integers so we interpolated the value of inversed pixels and assign it to the pixels in the projected image.

8)

We added the non-zero pixels in the warped image (created by image 1) to image2 and catted the zeros padding.