MSSC 6040 - Homework 2

Henri Medeiros Dos Reis

October 13, 2022

Problem 1 (5 pts)**.** T-B 4.1 (Hint: You do *not* need to compute any eigenvalues/vectors so find these SVDs. Instead, try and find the U, Σ, V matrices directly by expressing A as a product of elementary matrices, or as an outer product.)

Solution 1. .

a-)

$$A = \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix} = U\Sigma V^* \Rightarrow$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}, \ V^* = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

b-)

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} = U\Sigma V^* \Rightarrow$$

$$U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \ \Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}, \ V^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

c-)

$$A = \begin{bmatrix} 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = U\Sigma V^* \Rightarrow$$

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \ V^* = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

d-)

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$U\Sigma V^* = \begin{bmatrix} 1 & a \\ 0 & b \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ c & d \end{bmatrix}$$

We need to normalize the matrices.

$$\Rightarrow \sqrt{2}\begin{bmatrix}1 & a\\ 0 & b\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ c & d\end{bmatrix}$$

Now we can swap the order of the first column vector of the left singular values, and make one value negative, the result will be our second column vector. Then, do the same with the row vector of the right singular values.

$$U = \begin{bmatrix}1 & -0\\ 0 & 1\end{bmatrix},\ \Sigma = \begin{bmatrix}\sqrt{2} & 0\\ 0 & 0\end{bmatrix},\ V^* = \begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}$$

e-) Following the same steps as we did in letter d:

$$A = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} = \begin{bmatrix}1\\ 1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix}$$

$$U\Sigma V^* = \begin{bmatrix}1 & a\\ 1 & b\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ c & d\end{bmatrix} \Rightarrow$$

$$\sqrt{2}\sqrt{2}\begin{bmatrix}\frac{1}{\sqrt{2}} & a\\ \frac{1}{\sqrt{2}} & b\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ c & d\end{bmatrix} \Rightarrow$$

$$\begin{bmatrix}\frac{1}{\sqrt{2}} & a\\ \frac{1}{\sqrt{2}} & b\end{bmatrix}\begin{bmatrix}2 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ c & d\end{bmatrix} \Rightarrow$$

$$U = \begin{bmatrix}\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix},\ \Sigma = \begin{bmatrix}2 & 0\\ 0 & 0\end{bmatrix},\ V^* = \begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}$$

Problem 2 (5 pts, MATLAB)**.** T-B 4.3. You may use MATLAB's `svd` command to find the left and right singular vectors. Submit a printout/screenshot of your code as well as the plots showing the singular vectors for the indicated examples.

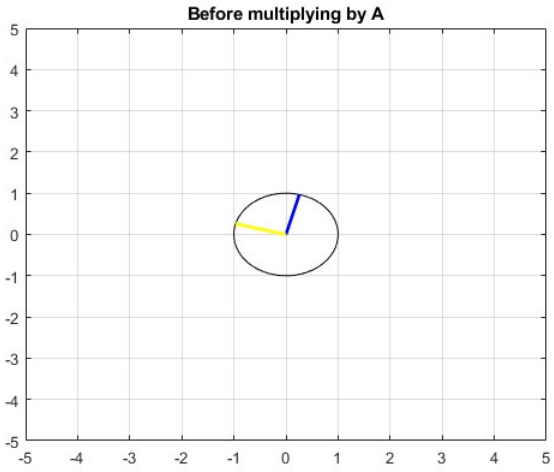
Solution 2.

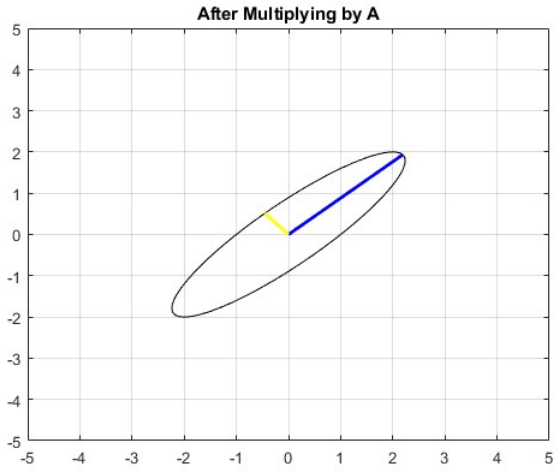
```
%Load the matrices
matrix3_7A = [1 2; 0 2];
matrix4_1aA = [3 0; 0 2];
matrix4_1bA = [2 0; 0 3];
matrix4_1dA = [1 1; 0 0];
matrix4_1eA = [1 1; 1 1];

%Put them in an array
arrayOfMatrices ={ matrix3_7A};
arrayOfMatrices{2} = matrix4_1aA;
```

```
arrayOfMatrices{3} = matrix4_1bA;
arrayOfMatrices{4} = matrix4_1dA;
arrayOfMatrices{5} = matrix4_1eA;

%For all matrices in the array, run the created function
for i= 1:length(arrayOfMatrices)
    PlotCircleEllipse(arrayOfMatrices{i},i)
end


function PlotCircleEllipse(A, i)
%Optional parameter, in case this was used in a for loop
if ~exist('i', 'var')
    i=1;
end
%get the svd of A
[U, Sigma, V] = svd(A);

%------ Start with the unit circle
% Grab all values between 0 and 2pi increasing by 0.01
t=0:0.01:(2*pi);
% Grab all values between 0 and 1 increasing by 0.1
ZeroToOne=0:0.1:1;
% x and y for all values of t
xCircle = cos(t);
yCircle = sin(t);
% Separate the vectors that will construct the lines in the
   circle, by using
% the right singular values
CircleVector1 = V(:,1);
CircleVector2 = V(:,2);

% Plot the circle
figure(i);
plot(xCircle, yCircle, 'k');
hold on;
plot(CircleVector1(1)*ZeroToOne, CircleVector1(2)*ZeroToOne, 'b
   ');
hold on;
plot(CircleVector2(1)*ZeroToOne, CircleVector2(2)*ZeroToOne, 'y
   ');
grid on;
```

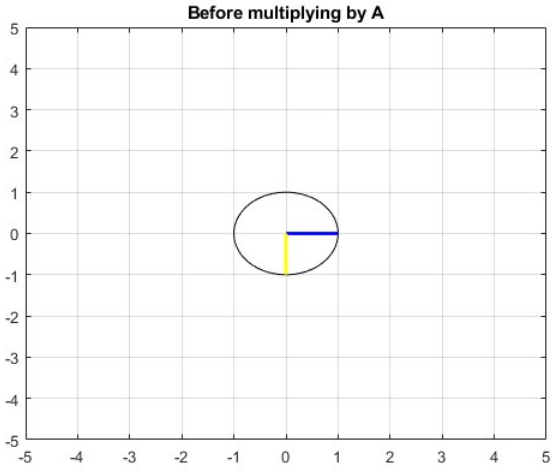
```
title('Before multiplying by A');
xlim([-5 5])
ylim([-5 5])

%------ Now create the ellipse
% The ellipse is A multiplied with the unit circle
ATimesCircle = A*[xCircle;yCircle];
% x and y for the Ellipse
xEllipse = ATimesCircle(1,:);
yEllipse = ATimesCircle(2,:);
% Separate the vectors that will construct the lines in the
   ellipse, by using
% the left singular values multiplied with the actual singular
   values, so
% that it scales properly
ellipseVector1 = Sigma(1,1)*U(:,1);
ellipseVector2 = Sigma(2,2)*U(:,2);

% Plot the ellipse
figure(i+10);
plot(xEllipse, yEllipse, 'k');
hold on;
plot(ellipseVector1(1)*ZeroToOne, ellipseVector1(2)*ZeroToOne,
   'b');
hold on;
plot(ellipseVector2(1)*ZeroToOne, ellipseVector2(2)*ZeroToOne,
   'y');
grid on;
title('After Multiplying by A');
xlim([-5 5])
ylim([-5 5])
end
```

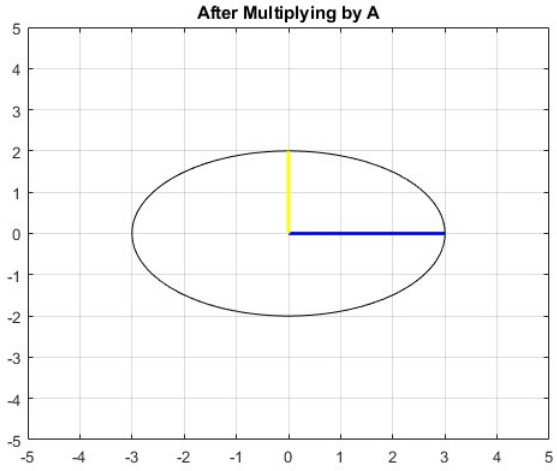
$$A = \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix}$$



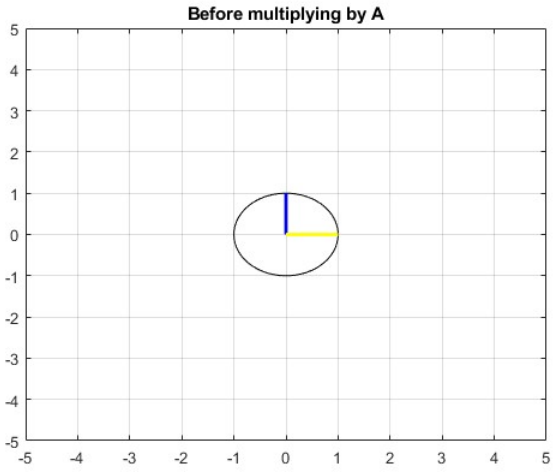


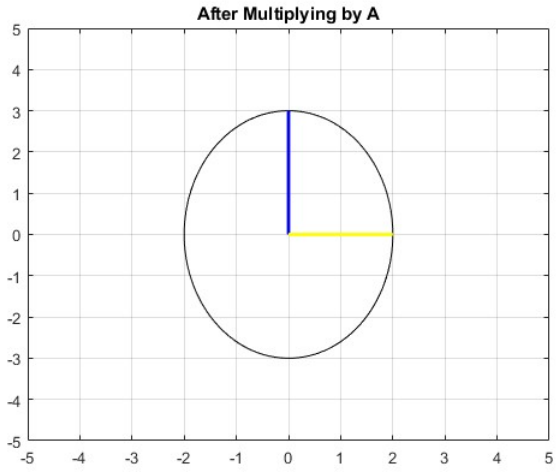
$$A = \begin{bmatrix} 3 & 0 \\ 0 & -2 \end{bmatrix}$$



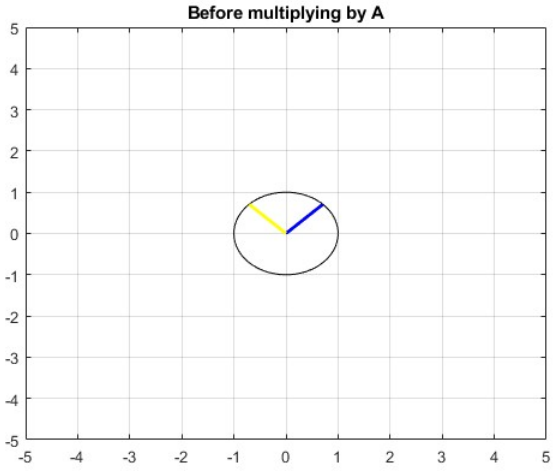


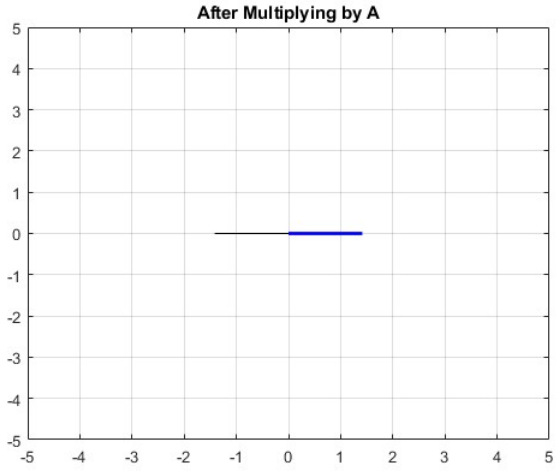
$$A = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$



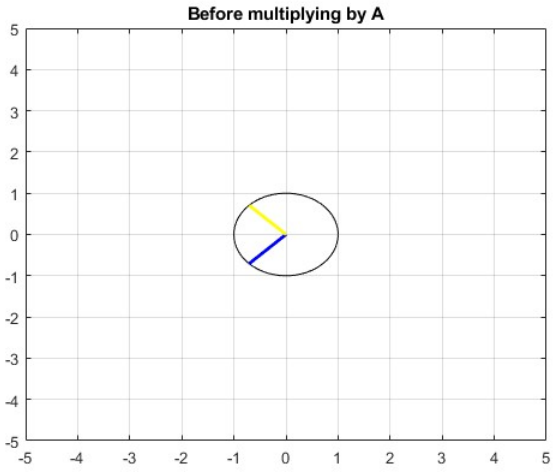


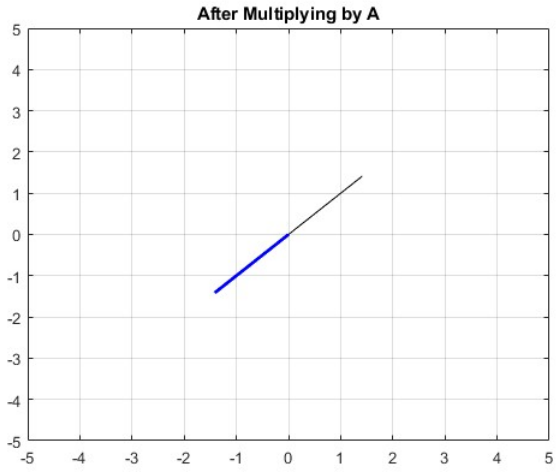
$$A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$





$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$





Problem 3 (5 pts)**.** T-B 5.3 parts (c)-(g) only. You may use the following SVD of A:

$$A = U\Sigma V^* = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{200} & 0 \\ 0 & \frac{\sqrt{200}}{2} \end{bmatrix} \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & -\frac{3}{5} \end{bmatrix}$$

For part (e), find the eigenvalues using the characteristic equation $\det(A - \lambda I) = 0$.

Solution 3.

$$A = \begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix}$$

$$U\Sigma V^* = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{200} & 0 \\ 0 & \frac{\sqrt{200}}{2} \end{bmatrix} \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & -\frac{3}{5} \end{bmatrix}$$

c-)

$$||A||_1 = max\{12, 16\} = 16$$

$$||A||_2 = \sqrt{200}, \text{ since it is the largest } \sigma$$

$$||A||_\infty = max\{13, 15\} = 15$$

$$||A||_F = \sqrt{(\sqrt{200})^2 + \left(\frac{\sqrt{200}}{2}\right)^2} = \sqrt{250}$$

d-)

$$A^{-1} = (U\Sigma V^*)^{-1} = (V^*)^{-1}\Sigma^{-1}U^{-1} = V\Sigma^{-1}U^*$$

Then since U and V are unitary, we just need to worry about Σ.

$$\Sigma^{-1} = \frac{1}{det}\begin{bmatrix} \frac{\sqrt{200}}{2} & 0 \\ 0 & \sqrt{200} \end{bmatrix}, det(\Sigma) = \sqrt{200}\frac{\sqrt{200}}{2} = 100$$

$$\Sigma^{-1} = \begin{bmatrix} \frac{\sqrt{200}}{200} & 0 \\ 0 & \frac{\sqrt{200}}{100} \end{bmatrix}$$

$$A^{-}1 = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & -\frac{3}{5} \end{bmatrix} \begin{bmatrix} \frac{\sqrt{200}}{200} & 0 \\ 0 & \frac{\sqrt{200}}{100} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{20} & \frac{-11}{100} \\ \frac{1}{10} & \frac{-1}{50} \end{bmatrix}$$

e-)

$$det(A - \lambda I) = 0$$

$$\begin{bmatrix} -2-\lambda & 11 \\ -10 & 5-\lambda \end{bmatrix} = 0 \Rightarrow$$

$$(-2-\lambda)(5-\lambda) - (11 * (-10)) = 0 \Rightarrow$$

$$\lambda^2 - 3\lambda + 100 = 0\lambda$$

$$\lambda = \frac{3 \pm \sqrt{9 - 4 * 1 * 100}}{2} \Rightarrow$$

$$\lambda_1 = \frac{3 + i\sqrt{391}}{2}, \lambda_2 = \frac{3 - i\sqrt{391}}{2}$$

f-)

$$det(A) = (-2)(5) - (-10)(11) = 100$$

$$\lambda_1\lambda_2 = \left(\frac{3 + i\sqrt{391}}{2}\right)\left(\frac{3 - i\sqrt{391}}{2}\right) = 100$$

And

$$|det(A)| = 100$$

$$\sigma_1\sigma_2 = (\sqrt{200})\left(\frac{\sqrt{200}}{2}\right) = 100$$

f-) Area of ellipsoid is $\pi\sigma_1\sigma_2$ where σ_1 and σ_2 are the semi axes radius of the ellipsoid. Then:

$$Area = \pi(\sqrt{200})\left(\frac{\sqrt{200}}{2}\right) = 100\pi$$

Problem 4 (10 pts, MATLAB)**.** For this problem, you will modify the provided MATLAB script `problem4.m` to create a face detection system using Eigenfaces.

(a) Your first task is to define a MATLAB function in the file `facedist.m` that takes in a 32× 32 pixel image and measures its distance from "face space"; the steps needed to do this are detailed below. The function has three inputs and one output: `dist = facedist(x,mu,U)`. Here `x` is an image patch given as a 32×32 array, `mu` is the mean face image given as a 1024×1 column vector, and `U` is the 1024×400 matrix whose columns are the ordered Eigenfaces. The output `dist` is the Euclidean distance of the image to "face space" (i.e., the function returns a small value if the image is a face, and a large value if the image is not a face). Your function should consist of the following steps:

1. Vectorize the input image.
2. De-mean the input by subtracting the mean face image from the vectorized input image.
3. Project the de-meaned input onto the "face space" described by the span of the top 16 Eigenfaces.
4. Finally, compute the 2-norm of the difference between the de-meaned input image and its projection onto face space as computed in step 3.

(b) Test out your `facedist` function by applying it to the face images saved in the files `greg.mat` and `nic_cage.mat`, and a random image `x = rand(32,32)`. Is the output consistent with your intuition?

(c) Now run the code block in the `problem4.m` script labelled `%%Part (c), step 1`. This will compute a "distance image" for the Friends image, which computes the distance to face space of every 32×32 pixel patch in the image. (Warning: This code block may take a long time to run if you coded "facedist" in an inefficient way, but an efficient implementation should only take a few seconds). In the next code block labelled `%%Part (c), step 2`, find a threshold `thresh` to apply to the distance image such that at least five of the six faces are recognized as faces, with no false positives. Can you do better than five out of six?

In your write-up, include a print-out/screenshot of your `facedist.m` function from part (a), the results of the tests in part (b), and a picture showing where the faces were detected on the Friends image in part (c). Finally, based on your observations of how well this system performed on the Friends image, discuss the potential limitations of this system for face detection in general, and how one might improve it.

Solution 4. a-) .

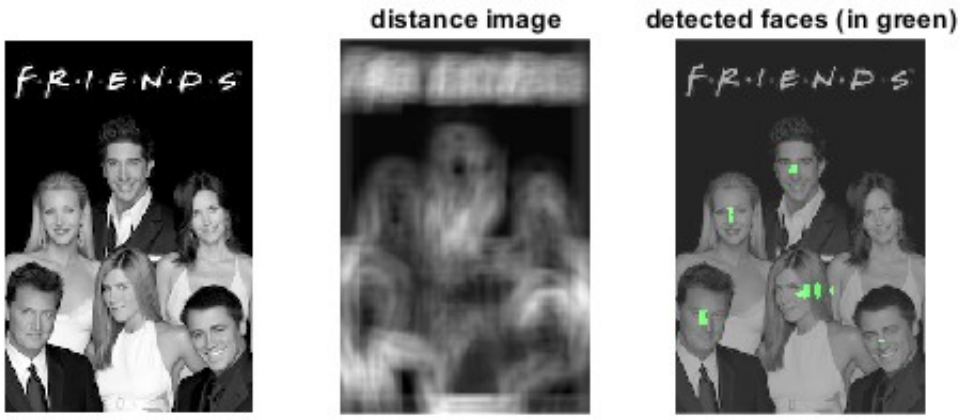
```
function dist = facedist(x,mu,U)
    %inputs:
    %x = input image as 32x32 matrix
    %mu = mean image as 1024x1 col vector
    %U = 1024x400 matrix of top eigenfaces
    vectorImage = x(:);
    demeanedImage = vectorImage-mu; %u
    UUTInv = (U(:,1:16).'*U(:,1:16))\eye(16);
    projectImg = (U(:,1:16)*UUTInv*((U(:,1:16).')*
       demeanedImage));
    %output:
    %dist = distance of de-meaned input image from face
       space

    dist=norm(projectImg-demeanedImage);
```

```
end
```

b-) .

```
load greg %loads image into variable x
figure(1);
imagesc(x); %show the image
axis image; axis off; colormap gray

dist1 = facedist(x,mu,U);
disp(dist1)
%do other tests here

load nic_cage %loads image into variable x
figure(2);
imagesc(x); %show the image
axis image; axis off; colormap gray

dist1 = facedist(x,mu,U);
disp(dist1)

x=rand(32,32); %loads image into variable x
figure(3);
imagesc(x); %show the image
axis image; axis off; colormap gray

dist1 = facedist(x,mu,U);
disp(dist1)
%--------------OUTPUT-------------
%    3.1553

%    2.5841

%    9.1413
```

c-) .



As we can see in the picture, it was only possible to detect 5 out of 6 faces. The threshold for this picture is 2.8, it is possible to see that the green is already showing in parts that are not faces, while the 5^{th} face (bottom right) has only a very small green area. So increasing the threshold is not only going to increase the areas that are faces, but also the areas that are not faces.

One of the possible ways to improve it would be if we could increase the number of faces in and have more diversity in `faces.mat` used to "train" the system. Since most of the faces in our data are males, it is not surprising that it has troubles trying to detect woman more than man.

Another possible way to improve this is the `facedist.m`, where we are using only the first 16 Eigenfaces. If this number is increased, the system is likely to perform better, however, this can get time and computational expensive. When testing with 150 Eigenfaces, I was able to see better results, but the run time went from couple of a seconds to almost 10 minutes. And I still got some false positives at the same spots, from previous tests.