ECE515 Midterm Project

Implementation of the Karhunen-Loeve (KL) Transform for Face Detection and Recognition after the Approach of Moghaddam and Pentland

Instructor: Prof. Ben-Arie

Name: Arindam Bose
UIC ID: 665387232
Date: 04/09/2015

MATLAB Code:

```
%% ECE 515: Mid-Term Projects
clear all;
```

```
%% read images from training database
trainingFaces1 = load('KL_norm_train.dat');
trainingFaces2 = load('KL_norm_train_2.dat');
[MTF1, NTF1] = size(trainingFaces1);
[MTF2, NTF2] = size(trainingFaces2);
```

```
%% storing in a common bean
trainingFaces = zeros(MTF1,NTF1 + NTF2);
for i = 1:NTF1 + NTF2
    if i<=NTF1

trainingFaces1(:,i)=trainingFaces1(:,i)/norm(trainingFaces1(:,i));
        trainingFaces(:,i) = trainingFaces1(:,i);
    else
        trainingFaces2(:,i-NTF1)=trainingFaces2(:,i-
NTF1)/norm(trainingFaces2(:,i-NTF1));
        trainingFaces(:,i) = trainingFaces2(:,i-NTF1);
    end
end
```

```
%% Display each image
figure;
for i = 1:NTF1 + NTF2
    subplot(3,6,i)
    tempFace = reshape(trainingFaces(:,i),40,40);
    imagesc(tempFace);
    colormap(gray);
    title(num2str(i));
end
suptitle('Original Face Images');
```

```
%% Average Image
sumFace = zeros;
for i = 1:NTF1 + NTF2
    sumFace = sumFace + trainingFaces(:,i);
end
avgFace = sumFace / 18;
avgFace = reshape(avgFace,40,40);
figure;
imagesc(avgFace);
colormap(gray);
title('Average Face Image');
```

```
%% Eigenvalues and EigenFaces
covFace = cov(trainingFaces',1);
[eigenFace, eigenValue] = eig(covFace);
eigenFace = fliplr(eigenFace);
eigenValue = rot90(eigenValue);
eigenValue = fliplr(eigenValue);
eigenValue = diag(eigenValue);
```

```
figure;
for i = 1:9
    subplot(3,3,i);
    imagesc(reshape(eigenFace(:,i),40,40));
    colormap(gray);
end
suptitle('9 Eigen Face Images');
```

```
%% Reconstruction from 6 Eigen Faces
L = 6;
trainingFacesReCon = eigenFace(:,1:L) * eigenFace(:,1:L)' *
trainingFaces;
trainingFacesReCon = trainingFacesReCon +
reshape(avgFace,1600,1)*ones(1,18);
figure;
for i = 1:9
    subplot(3,3,i);
    imagesc(reshape(trainingFacesReCon(:,i),40,40));
    colormap(gray);
end
suptitle('9 Reconstructed Face Images');
```

```
%% Read image from testing database and Display
testFace = load('Test_image.dat');
figure;
imagesc(testFace);
colormap(gray);
title('Test Face Image');
```

```
%% Global Normalization
testFace = testFace - mean(mean(testFace));
testFace = testFace./sqrt(sum(testFace(:).^2));
```

```
%% Distance Image
[MA, NA] = size(avgFace);
[ME, NE] = size(eigenValue);
[MT, NT] = size(testFace);
distance = zeros(MT-MA+1,NT-NA+1);
for i = 1:MT-MA+1
    for j = 1:NT-NA+1
       tempTestFace = testFace(i:i+MA-1,j:j+NA-1);
       tempTestFace = tempTestFace - mean(mean(tempTestFace));
       tempTestFace = tempTestFace./sqrt(sum(tempTestFace(:).^2));
       w = (tempTestFace - avgFace);
       y = eigenFace(:,1:L)' * w(:);
       wSquared =  w(:)'*w(:);
       ySquared = y'*y;
       dis = ySquared ./ eigenValue;
       errorSquared = wSquared - ySquared;
       rho = sum(eigenValue(L+1:ME))/(ME-L);
       distance(i,j) = sum(dis(1:L)) + errorSquared/rho;
    end
end

figure;
imagesc(distance);
```

```
colormap(gray);
title('Distance Image');
```

```
%% Find Minima using Non-Minimal Suppression method
p = MA - 1;
q = NA - 1;
[M, N] = size(distance);
flag = zeros(M-p+1, N-q+1);
for i = 1:M-p+1
    for j = 1:N-q+1
        temp = distance(i:i+p-1, j:j+q-1);
        flag(i,j) = ~(temp(ceil(p/2),ceil(q/2)) == min(min(temp)));
    end
end

[r,c] = size(flag);
tempMinPosition = [];
minDist = [];
for i = 1:r
    for j = 1:c
        if(flag(i,j)==0)
            tempMinPosition = [tempMinPosition; [i+MA/2-1,j+NA/2-1]];
            minDist = [minDist; sqrt((i+MA/2-1)^2 + (j+NA/2-1)^2)];
        end
    end
end
```

```
%% Sorting Coordinates in ascending order according to its distance
from origin
minPosition = [];
for i = 1:3
    tempDist = [minDist(3*(i-1)+1); minDist(3*(i-1)+2); minDist(3*(i-
1)+3)];
    sortedDist = sort(tempDist);
    for j = 1:3
        for k = 3*(i-1)+1:3*(i-1)+3
            if sortedDist(j) == minDist(k)
                minPosition = [minPosition; tempMinPosition(k,:)];
            end
        end
    end
end
```

```
%% Face Recognition
y = zeros(size(minPosition,1), L);
distanceScore = zeros(size(minPosition,1),9);
for i=1:size(minPosition,1)
    tempTestFace = testFace(minPosition(i,1) : minPosition(i,1) + MA -
1, ...
        minPosition(i,2) : minPosition(i,2) + NA - 1);
    tempTestFace = tempTestFace - mean(mean(tempTestFace));
    tempTestFace = tempTestFace./sqrt(sum(tempTestFace(:).^2));
    w = (tempTestFace - avgFace);
    y(i,:) = w(:)'*eigenFace(:,1:L);

    for j=1:9
```

```
        wbar = reshape(trainingFaces(:,j),40,40) - avgFace;
        ybar = wbar(:)'*eigenFace(:,1:L);
        distanceScore(i,j) = norm(y(i,:) - ybar);
    end
end

findNumber = zeros;
for i=1:size(distanceScore,1)
    findNumber(i,1) = find(distanceScore(i,:) ==
min(distanceScore(i,:)));
end
```

```
%% Print Detected Face
figure;
imagesc(testFace);
colormap(gray);
hold on;
plot(minPosition(:,2) + NA/2, minPosition(:,1) +
MA/2,'r+','LineWidth',2);
for i = 1:size(findNumber,1)
    text(minPosition(i,2), minPosition(i,1),...
        num2str(findNumber(i)), 'Color', 'r', 'FontSize',20);
end
hold off;
title('Face Detected Image');
```

```
%% Display Results
disp(minPosition); % The coordinates of detected faces
disp(y); % Test Face KL Coefficients (9x6)
disp(distanceScore); %Test and Training Face KL Coefficients Matching
Score (9x9)
```

Discussions:

I. Why is it necessary to do normalization on training and test images?

Each image window is first normalized by making it zero mean and unit variance. Normalization is necessary by the need to match the statistics of the test image and that of the training images. Since training images and test images might be obtained under very different environments and probably though different devices, they are characterized by different statistics. Though normalization the differences are removed and the steps taken in detection and recognition become meaningful. For example, the subtraction of average face from the test image is meaningless if we don't normalize the test image first.

II. Why is it necessary to subtract the average face from *w*?

The reason to subtract the average face from W because of the definition of Mahalanobis distance which is in fact the exponent of a Gaussian density function:

$$d(x)=(x-\vec{x})^T C^{-1}(x-\vec{x})$$

And it is consistent with our computation this definition is that in computation of KL coefficients: $y = U^T(x-\bar{x})$. A way to understand this definition is that in computing $d(w)$, we are in fact measure the similarity of w with the average face. The lower the distance is, the more similar that part of image is to the average face.

III. The minima of the distance image and coordinates of detected faces

22	21
21	60
21	101
61	20
61	60
61	101
102	21
101	61
101	101

IV. Test Face KL Coefficients (9×6)

-0.4429	0.0425	0.1847	-0.1303	-0.0524	0.2018
-0.2943	-0.0836	-0.1840	0.1046	0.1043	0.0706
-0.3074	0.0362	-0.0223	0.0100	0.0737	0.0857
-0.4636	0.0441	0.2425	-0.3780	0.0791	-0.0439
-0.1844	0.3868	0.2172	0.1391	-0.0002	-0.2375
-0.3820	-0.0629	-0.0179	-0.1256	-0.0186	0.0841
-0.1430	0.1693	-0.0825	-0.0074	0.3261	0.0012
0.0009	-0.1665	-0.3286	0.1393	-0.2514	-0.0452
0.0153	-0.0663	-0.2417	0.0965	-0.0395	0.1322

Table 1: Test Face KL coefficients

V. Test and Training Face KL Coefficients Matching Score (9×9)

0.3903	0.4421	0.5076	0.5017	0.7609	0.4707	0.4718	0.5622	0.6102
0.7390	0.1124	0.1782	0.6840	0.8524	0.4537	0.2770	0.4010	0.3668
0.5880	0.1720	0.2012	0.5679	0.7094	0.3929	0.2028	0.4498	0.4160
0.7401	0.5893	0.6448	0.2398	0.7819	0.4493	0.5323	0.7164	0.7569
0.8049	0.6702	0.5862	0.8741	0.1992	0.6597	0.5232	0.8234	0.6836
0.5938	0.2333	0.3666	0.4322	0.8064	0.3008	0.3319	0.3593	0.4161
0.8488	0.4130	0.3105	0.6924	0.7435	0.6007	0.2823	0.7210	0.6250
0.9605	0.5498	0.6225	0.9538	1.0049	0.6148	0.6767	0.4200	0.3286
0.8144	0.4223	0.4571	0.8542	0.9453	0.6003	0.5323	0.4903	0.4063

Table 2: Test and Training Face KL Coefficients Matching Score

Expected minimum distances are in underlined positions and actual minimum distances are in shaded region. 6 out of 9 faces are matched correctly with the original ones.

Images:

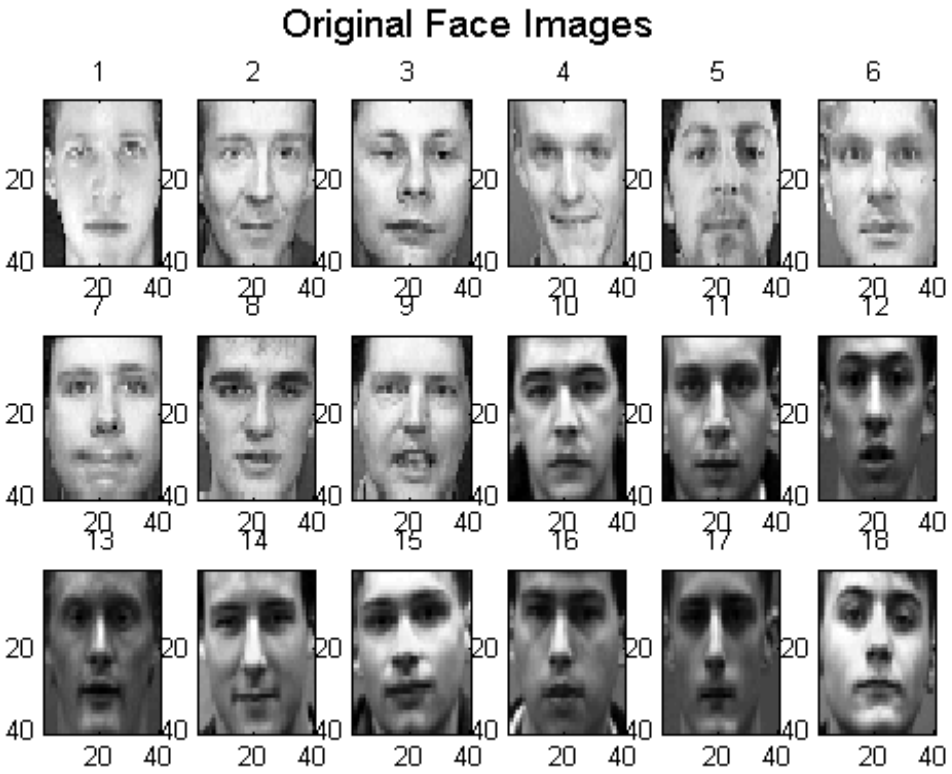

Fig. 2: Original Face Images

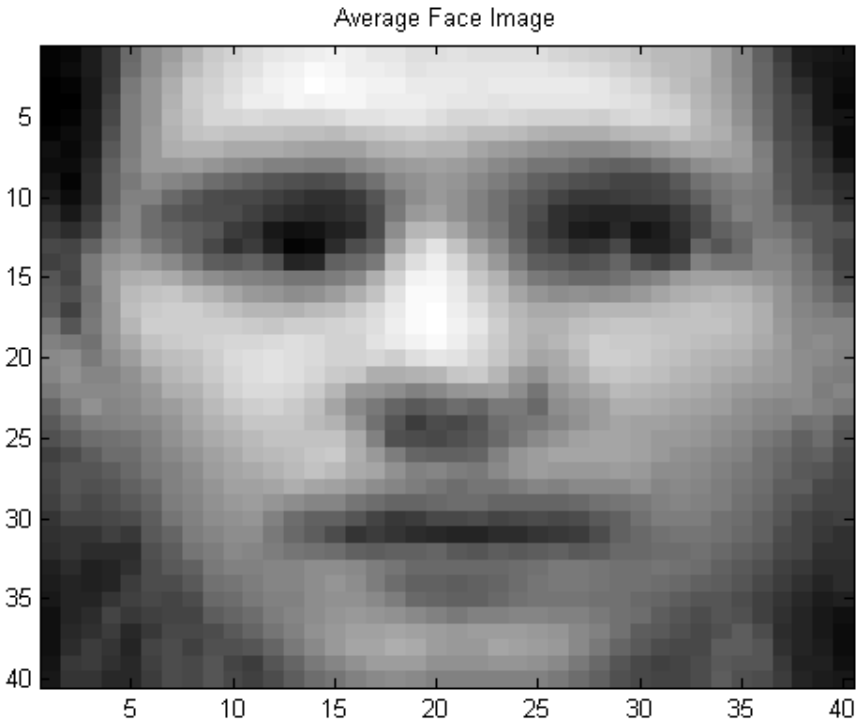

Fig. 2: Average Face Image

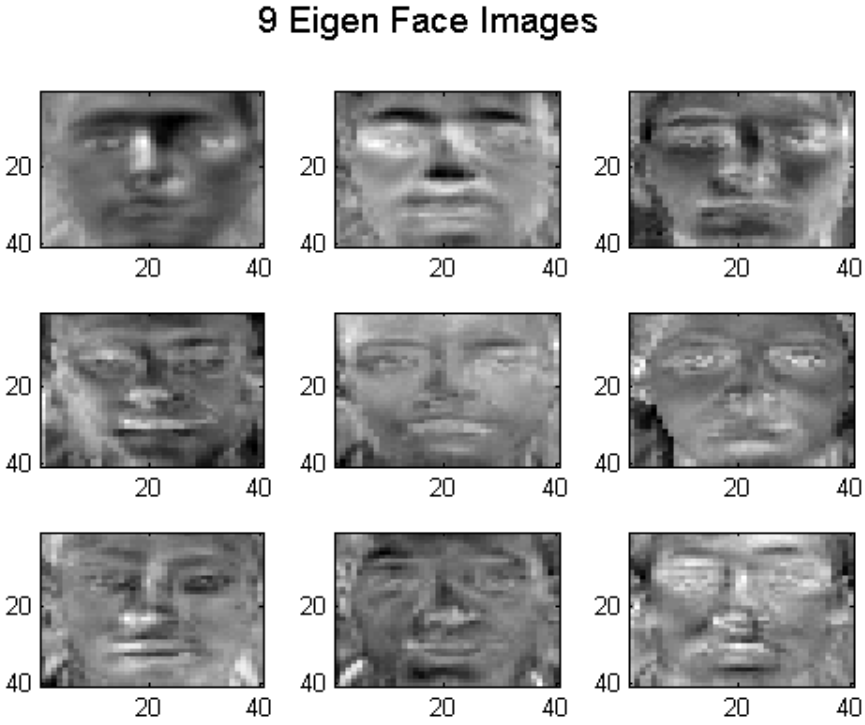

Fig. 3: 9 Eigen Face Images

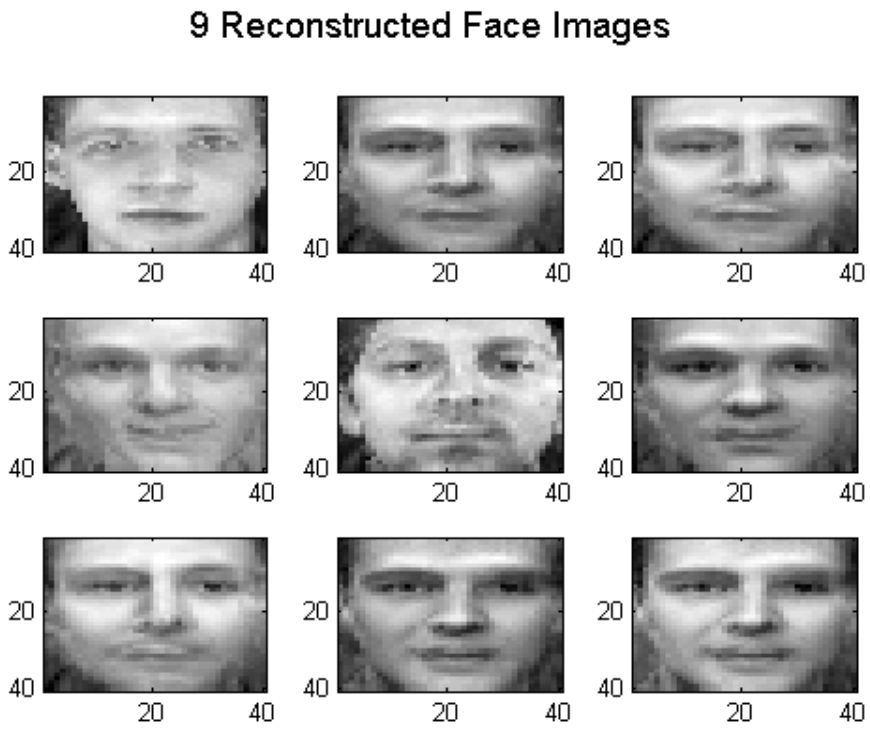

Fig. 4: 9 Reconstructed Face Images

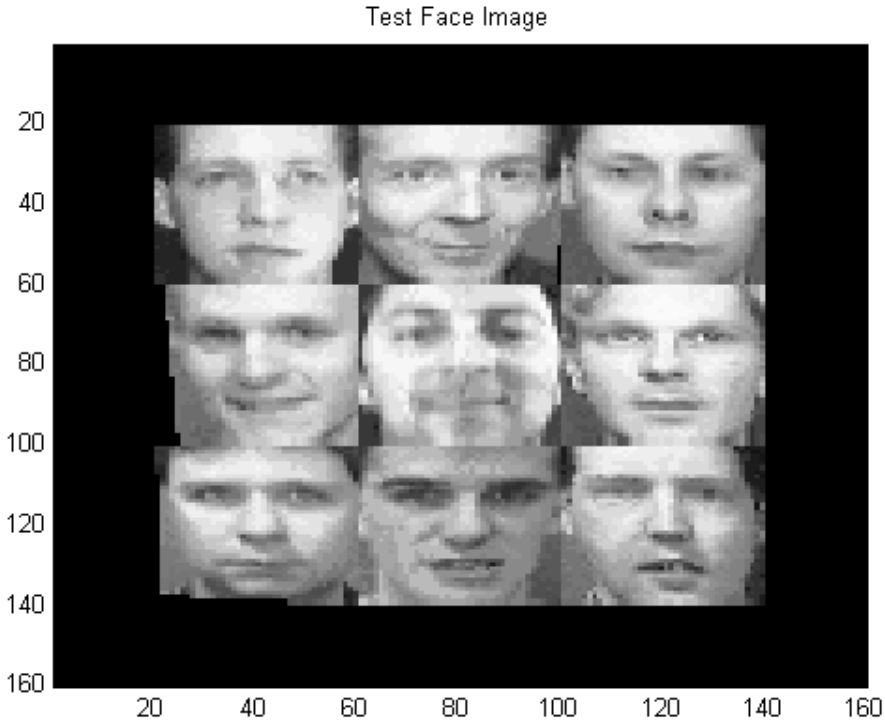

Fig. 5: Test Face Image

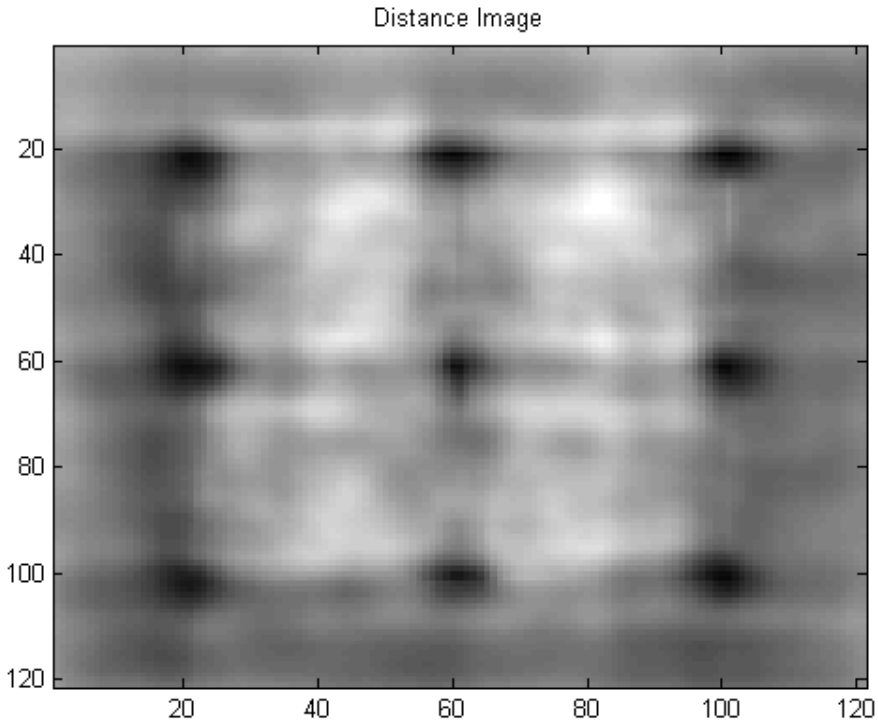

Fig. 6: Mahalanobis Distance Image

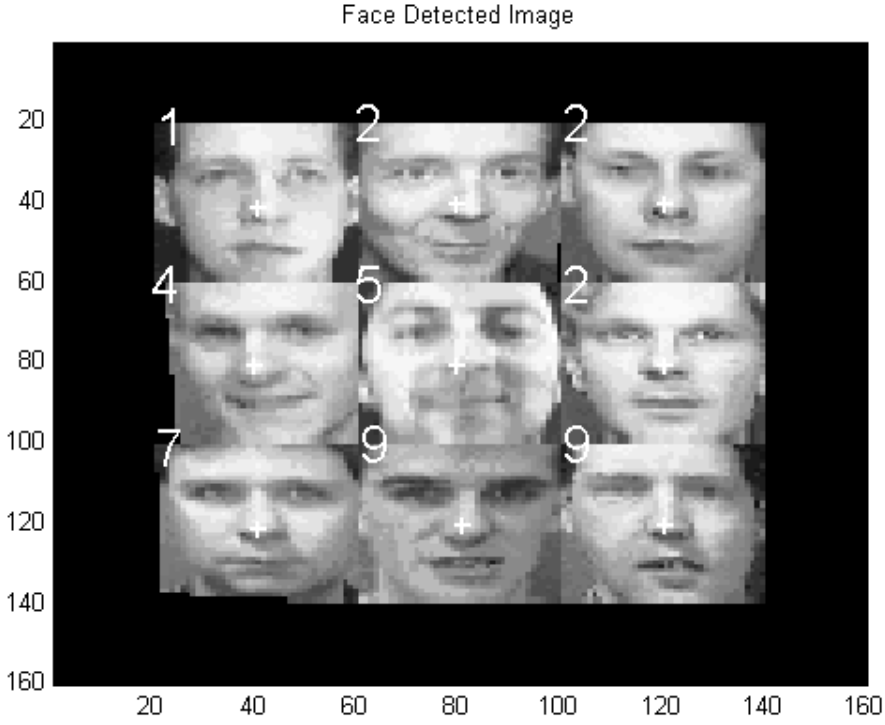

Fig. 7: Face Detected Image