```
images_50 = cell(1,50); % Creates a 1-by-50 cell array of empty
 matrices
images_30 = cell(1,30); % Creates a 1-by-30 cell array of empty
 matrices
files = cell(1,50);
image_vec = zeros(900,1);
count_30=1;
sum_30 = zeros(30,30);

mean_ = zeros(30,30);
mean_face = zeros(900,1);

X_mat = zeros(900,30);

C = zeros(900,900);



dir_files = dir('proj03_face_images/*.bmp'); % dir: lists files and
 folders in the current folder.
num_files = length(dir_files); % Number of files found


for i = 1:num_files %1 to 50

    currentfilename =  [dir_files(i).name]; %Index into the dir_files
 to access a particular item
    currentimage = double(imread(currentfilename)); % read current
 image file and save it as an image


    image_vec = reshape(currentimage,900,1);
    images_50{i} = image_vec; % append current image to list of 50
 image matrices (array of vectors)

    % getting 30 images
    if (mod(i,5)>0) && (mod(i,5)<4)

        sum_30 = sum_30 + currentimage; % does NOT work. gives matrix
 of
        %all elements 505

%         image_vec = reshape(currentimage,900,1);
        images_30{count_30} = image_vec; % append current image to
 list of 30 image vectors 900 x 1



        count_30 = count_30 + 1;

    end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mean face calculation
mean_ = sum_30 / 30;
disp("Mean face:");
imshow(mean_,[]);
snapnow
mean_face = reshape(mean_,900,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate igen_faces/basis faces
mat_images_30 = cell2mat(images_30); % converts array of matices to a
 single matrix 900 x 30


for col = 1:30
    X_mat(:,col) = mat_images_30(:,col) - mean_face; % 900 x 30 =
 900x30 - 900x1
end

% Covariance matrix C
C = X_mat * transpose(X_mat); % C is 900 x 900


% eigen vector and eigen value calculation
[vec_mat,val_d_mat] = eigs(C,50); %returns a diagonal matrix of
 eigenvalues and a matrix V whose columns are the corresponding
 eigenvectors.

eigVal_col_vec = diag(val_d_mat); %returns a column vector of the main
 diagonal elements of val_d_mat.
[d,ind] = sort(eigVal_col_vec, 'descend');
all_eig_vals = val_d_mat(ind,ind); %reorder the diagonal elements
all_eig_vec = vec_mat(:,ind);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% top 50 eigen values
E_50_mat = zeros(900,50);
for t=1:50
    E_50_mat(:,t)= all_eig_vec(:,t);
end

disp("eigen faces");
% print top 50 eigen faces
for k = 1:50
    eig_face = E_50_mat(:,k);

    eig_face_img = reshape(eig_face,30,30);
    %%%%%%
    eig_face2 = vec_mat(:,k);
    eig_face_img2 = reshape(eig_face2,30,30);
    imshow(eig_face_img2,[]);
    snapnow
```

```
    %%%%%%

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Coefficient
% subjectsCoeff = zeros(10,NUMBERTRAININGS(loops)*6);
mat_images_50 = cell2mat(images_50);
for i = 1:50

    coefs_mat(:,i) = transpose(vec_mat) * (double(mat_images_50(:,i)-
 mean_face(:,1))); % 50x1 = 900x50 *(900x1 - 900x1)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate genuine scores and impostor scores by computing the
 Euclidean distance
%between the feature vectors of every pair of face images.
imp_score = zeros(1150,1);
gen_score = zeros(50,1);

count = 1;
g_count = 1;
im_count = 1;


for i = 1:50
    for j = i+1:50

        d_ = sum((coefs_mat(:,i) - coefs_mat(:,j)).^2);
        dist = sqrt(d_);

        diff_ (count,1)= dist;

        count = count + 1;

        if idivide(j-1,int32(5)) ~= idivide(i-1,int32(5))

            imposter(im_count) = dist;
            im_count = im_count + 1;

        else
            genuine(g_count) = dist;
            g_count = g_count + 1;


        end

    end

end
```

```
figure();
histogram(genuine);
hold on;
histogram(imposter);
xlabel("Distance")
ylabel("Count")
legend("Genuine", "Imposter")
hold off;
t_genuine = transpose(genuine);
t_imposter = transpose(imposter);
drawROC(t_genuine,t_imposter,'d');
```

Mean face:

eigen faces

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

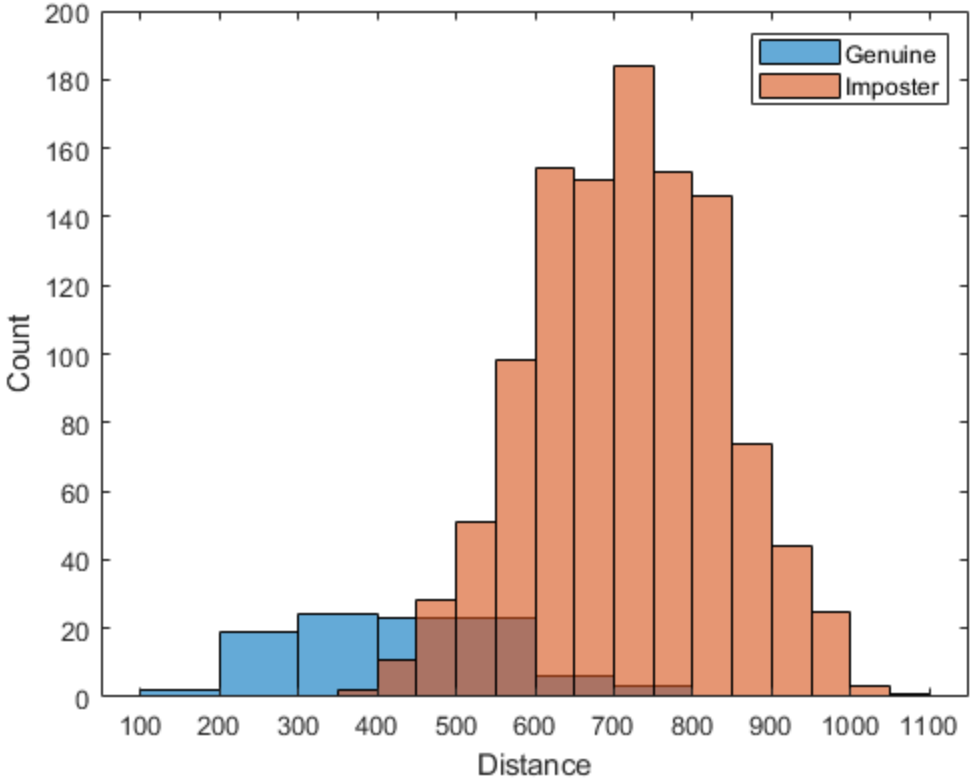
Warning: Displaying real part of complex input.

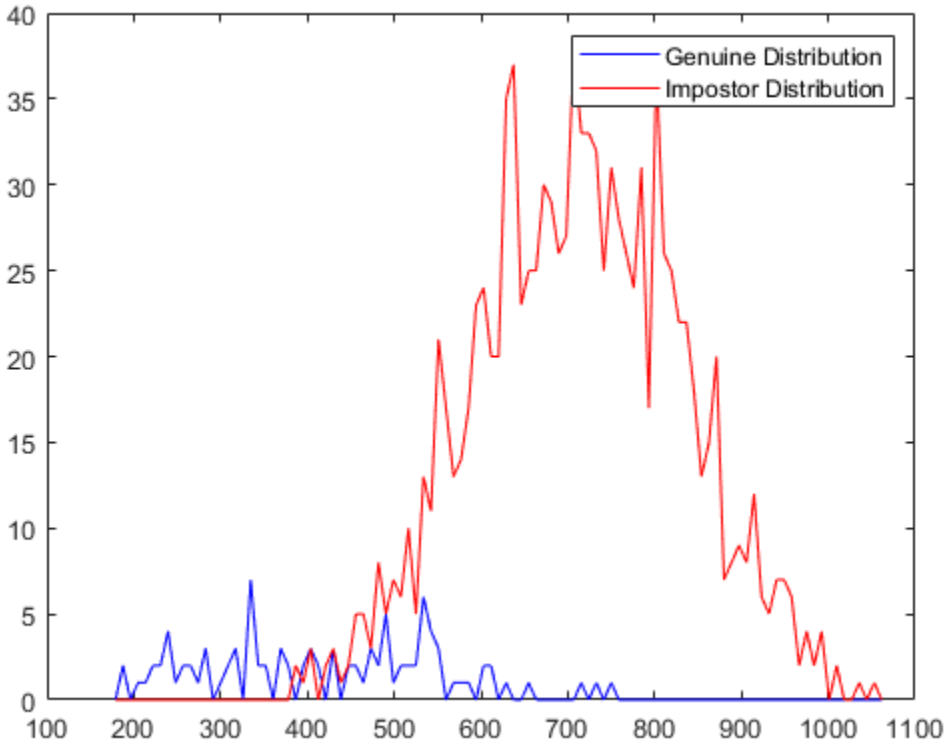
Warning: Displaying real part of complex input.

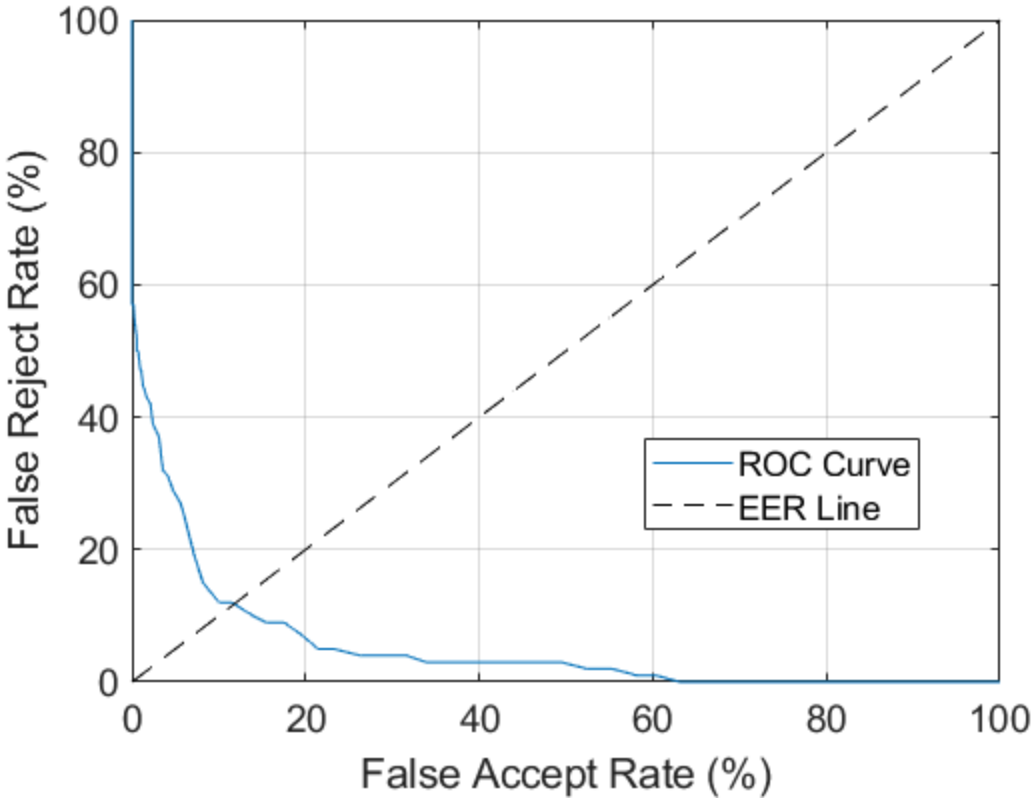
Warning: Displaying real part of complex input.

Warning: Displaying real part of complex input.

Begin ROC..
End ROC..


Genuine
Imposter
Count
Distance
0
20
40
60
80
100
120
140
160
180
200
100
200
300
400
500
600
700
800
900
1000
1100


Genuine Distribution
Impostor Distribution
0
5
10
15
20
25
30
35
40
100
200
300
400
500
600
700
800
900
1000
1100



Published with MATLAB® R2018b