```
%Going to directory
location_all = 'All/*.TIF';
All_image_files = dir(location_all);

%Finding length of folder called image_files
ALL_len = length(All_image_files);

%Initializing all_images
ALL_init = [];

%Loading each image in the folder under ALL
for i = 1 : ALL_len
    All_file_name = append('ALL/', All_image_files(i).name);
    All_image = imread(All_file_name);

    %Vectorizing the images
    All_image = reshape(All_image, [], 1);
    ALL_init = [ALL_init, All_image];
end;

%Type casting to double
ALL_init = cast(ALL_init, 'double');

%Computing the mean of the image
ALL_u = mean(ALL_init, 2);

%Constructing covariance matrix
ALL_C = (ALL_init - repmat(ALL_u, 1, ALL_len)) * (ALL_init - repmat(ALL_u, 1,
ALL_len)).';

%Solving the eigenvalue problem
[V, D] = eig(ALL_C);

%Vectorize the diagonal component of a matrix
d = diag(D);

%Sorting eigen vectors
[sorted_eigen, sort_indices] = sort(d, 'descend');
sortedVectors = V(:, sort_indices);

%Selecting top k Eigen Vectors
k = 10
W = sortedVectors(:, 1:k);


%Going to FA directory and reading

%Going to directory
location_fa = 'FA/*.TIF';
FA_image_files = dir(location_fa);

%Finding length of folder called image_files
FA_len = length(FA_image_files);

%Initializing all_images
```

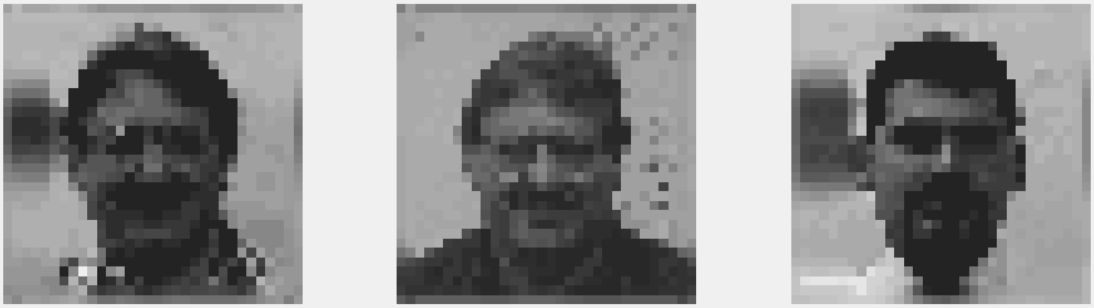
```
FA_init = [];

%Loading each image in the folder under ALL
for i = 1 : FA_len
    FA_file_name = append('FA/', FA_image_files(i).name);
    FA_image = imread(FA_file_name);

    %Vectorizing the images
    FA_image = reshape(FA_image, [], 1);
    FA_init = [FA_init, FA_image];
end;

%Type casting to double
FA_init = cast(FA_init, 'double');

% Constructing DB of known faces
C_FA = W'*(FA_init - repmat(ALL_u, 1, FA_len));

%Testing
%Going to FA directory and reading

%Going to directory
location_fb =  'FB/*.TIF';
FB_image_files = dir(location_fb);

%Finding length of folder called image_files
FB_len = length(FB_image_files);

%Initializing all_images
FB_init = [];

correct = 0;
incorrectly_predicted_images = {};
expected_images = {};
predicted_images = {};

for i = 1 : FB_len
    FB_filename = append('FB/', FB_image_files(i).name);
    FB_img = imread(FB_filename);

    FB_img = reshape(FB_img, [], 1);
    calc_Z = double(FB_img);

    % NN classification
    projected_Z = W' * (calc_Z - ALL_u);

    minDistance = 10000000;
    minIndex = 0;

    for j = 1 : FA_len
        currDistance = norm(C_FA(:,j) - projected_Z);
        if currDistance <= minDistance
            minDistance = currDistance;
            minIndex = j;
        end
```

```
    end

    input = FB_image_files(i).name(1:11);
    predicted = FA_image_files(minIndex).name(1:11);

    if strcmp(input, predicted) == 1
        correct = correct + 1;
    end

    if strcmp(input, predicted) == 0
        incorrectly_predicted_images = [incorrectly_predicted_images,
FB_filename];
        expected_images = [expected_images, strcat('FA/', input, 'FA0.TIF')];
        predicted_images = [predicted_images, strcat('FA/',
FA_image_files(min_index).name)];
    end
end

%Testing and Displaying the images
% Guided by Ishika Shah
disp('Accuracy')
disp((correct/len_FB) * 100);

for i = 1 : length(expected_images)
    figure;
    subplot(1, 3, 1);
    img = imread(incorrectly_predicted_images{i});
    imshow(img);
    xlabel("Input Image");
    subplot(1, 3, 2);
    img = imread(expected_images{i});
    imshow(img);
    xlabel("Actual Image");
    subplot(1, 3, 3);
    img = imread(predicted_images{i});
    imshow(img);
    xlabel("Predicted Image");
end
```

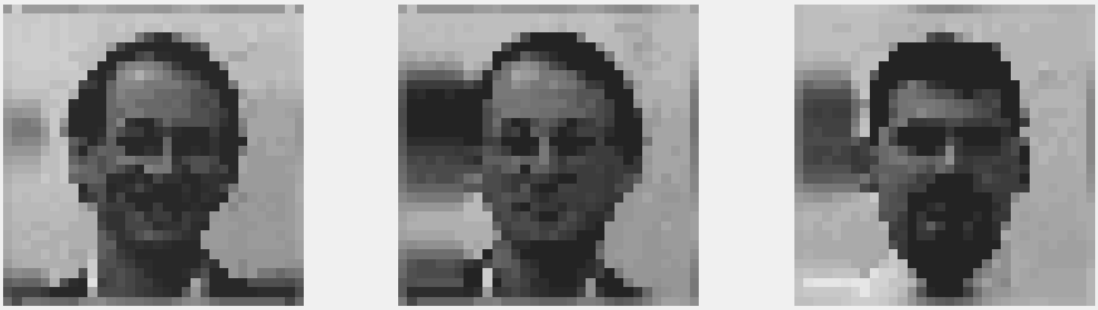
Result –

```
k =

    10

Accuracy
   73.9130
```

Images –



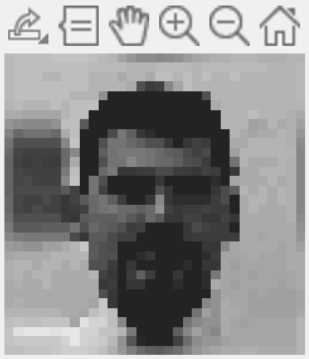
Input Image | Actual Image | Predicted Image



Input Image | Actual Image | Predicted Image

Input Image

Actual Image



Predicted Image

Input Image

Actual Image

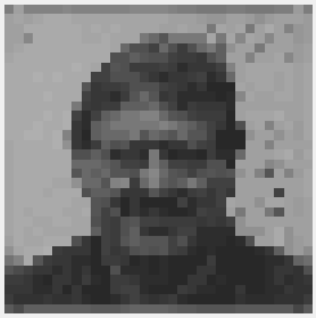
Predicted Image

Input Image

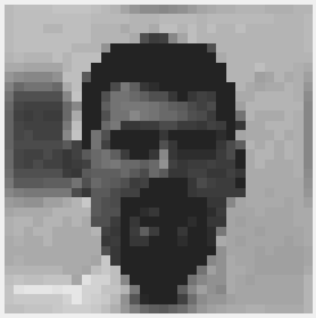
Actual Image

Predicted Image

Input Image



Actual Image



Predicted Image