Final Project: Facial Identification System

John McAvoy
ECE 09.341, Section 3
14 December 2018
Email: mcavoyj5@students.rowan.edu

CONTENTS

I. INTRODUCTION

A. Objectives

This four parts of this project are designed to accomplish:

1) transforming an image into a two dimensional Discrete Cosine Transform (2-D DCT)
2) performing feature extraction on different subjects using "zigzag" scanning of their 2-D DCT and visualizing this method can be used to classify subjects,
3) training a face identification system to be used for classification of subjects,
4) and creating a software implementation of a face classification system and determine how k and DCT length effect the success rate of the classification system.

B. Background and Relevant Theory

Image classification system design has, and continues to be, an important field in computer science since it has so many real-world applications. One of the largest types of image recognition used today is facial recognition. The technology to classify human faces is mature and fast enough now that it is a common feature in many smart phones and cameras.
Image classification algorithms have been around since as early as the 1960s for the purpose of enabling computers to be able to classify different images similarly to the human mind, Since the boom in computational performance, more advanced image classification algorithms have been developed. A popular method that is seen in many image recognition algorithms is feeding a two-dimensional discrete cosine transform (2-D DCT) of images into either a mathematical discriminator function or more recently, a neural-network which performs a set of functions that are able to classify particular images with over 90% accuracy [1] [2] [3]. A common mathematical discrimination technique is using a K nearest neighbor algorithm to find a class that best fits a test image, similar to the way a line of best fit is determined from a scatter plot [4]. Combining these two image recognition methods, a 2-D DCT of an image of a face can be turned into a 1-dimensional vector and then fed into a k nearest neighbor classifier that can match the test image to the closest matching face in a dataset of 1-DCT images. In this project, this method will be used to design a facial recognition classifier that can be used to classify images of human faces as a certain subject.

II. PROTOCOL, RESULTS AND DISCUSSION

This project was executed in four parts, each to accomplish one of the main objectives. A MATLAB™ scripts were developed in order to test the various facial recognition concepts and AT&T Laboratories' face database, consisting of ten 112x92 grayscale (256 bit) images of forty different subjects, was used as the test sample throughout.

A. Part 1: Two dimensional (2-D) DCT and inverse 2-D DCT

A MATLAB™ script (Listing 1) was developed to plot the 2-D DCT of an image of a subject's face. The chosen subject's face was first plotted as an image (Figure 1).

Fig. 1. Part 1: att_faces/s1/1.pgm

Next, the 2-D DCT of the image was calculated using the `dct2` function and plotted (Figure 2).

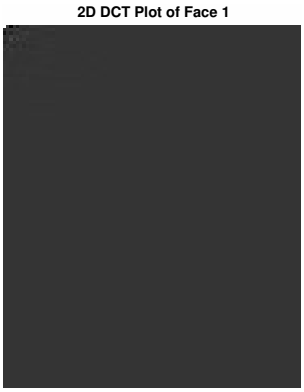

Fig. 2. Part 1: DCT

In this plot, the image is nearly all black except for the very top-left corner. In order to visualize the 2-D DCT plot, the same matrix was plotted using a logarithmic scaled axis (Figure 3).

Fig. 3. Part 1: DCT Logarithmic Scale

The logarithmic plot shows a concentration if white pixels in the top-left corner implying, that the DCT coefficients with the greatest magnitudes in the image are in the lowest-frequencies [1].
In order to demonstrate how the image can be recovered by calculating the inverse-DCT of the 2-D DCT matrix, the `idct2` function was used on the transform and the result was plotted (Figure 4).

Fig. 4. Part 1: Recovered Image

The resultant matrix matches the original image, demonstrating that the 2-D DCT matrix holds the same signal information as the original image matrix and can be used in feature extraction.

B. Part 2: Feature Extraction

Two MATLAB™ scripts (Listings 2 and 6) were used to explore feature extraction and how 2-D DCT of facial images can be used to discriminate between subjects. The `findfeatures.m` script generates a discrete signal from an image by first converting the 2-D matrix into a 1-D vector using zigzag 1-dimension reduction (Figure 5).

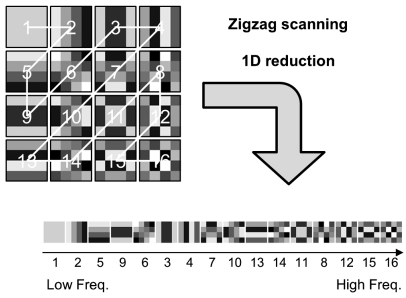

Fig. 5. ZigZag 1-Dimension Reduction [5]

Next, this 1-D vector is applied a discrete cosine transform and returned. The dimensions of the result vector is configured with the `dctlength` parameter, the greater this value, the more DCT coefficients are returned.
In order to visualize the signals generated from `findfeatures`, subject #3's images were loaded into the `findfeatures` function and the output signal of each image was plotted and overlayed to show the common shape of the signal that subject #3's images generate. This process was repeated three times while varying `dctlength` in order to show how increasing the amount of DCT affects the result outputted (Figures 6, 7, and 8).

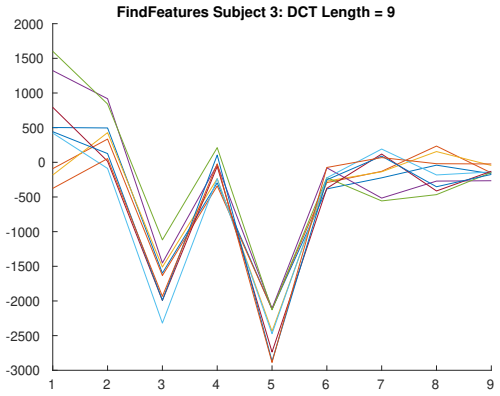

Fig. 6. Part 2: Subject 3, DCT Length = 9

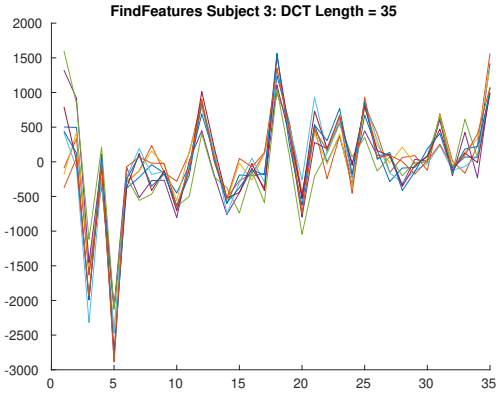

Fig. 7. Part 2: Subject 3, DCT Length = 35

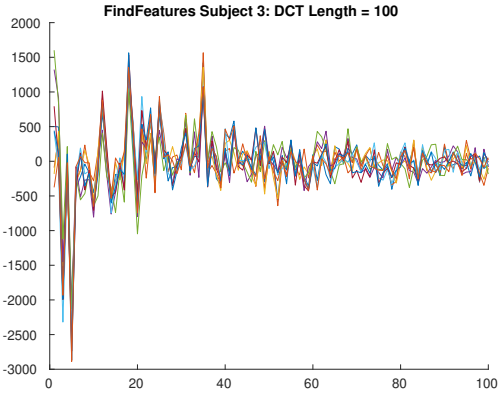

Fig. 8. Part 2: Subject 3, DCT Length = 100

Next, this same procedure was applied to another subject in order to determine how this method can be used to differentiate between different subjects (Figures 9, 10, and 11).

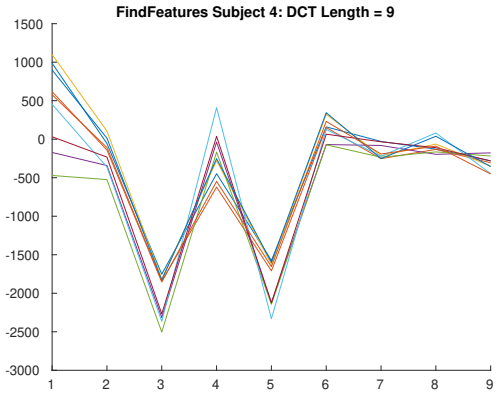

Fig. 9. Part 2: Subject 4, DCT Length = 9

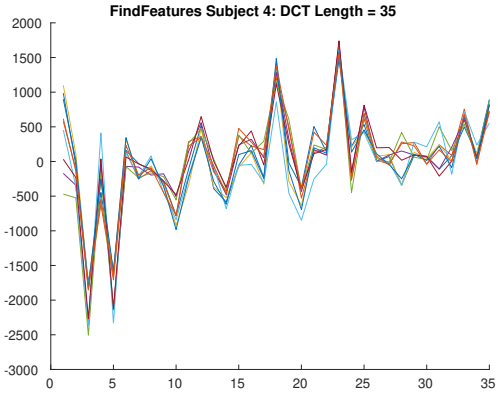

Fig. 10. Part 2: Subject 4, DCT Length = 35

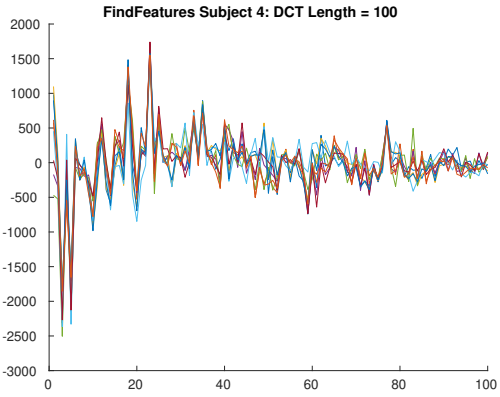

Fig. 11. Part 2: Subject 4, DCT Length = 100

The results of this test showed that facial images taken of the same subject will produce similar 1-D DCT signals that differ from the signals generated from another subject's 1-D DCT. Thus, property can be used to in image classification to discriminate between images of different subjects. These plots also reveal that descrimination becomes more apparent as the number of DCT coefficients increases because the signals become more unique as `dctlength` increases.

C. Part 3: Training the Face Identification System

In this portion of the project, a MATLAB™ script (Listing: 3) was developed to run `face_recog_knn_train.m`, a script that combines one-half of the 1-D DCT of the test images together, along with their corresponding subject label (`trclass`) in order to create a `trdata_raw.mat` file that can be used in the classification of the other half of the test images. The `face_recog_knn_train.m` function generates the trained data based on the range of subjects inputted as well as the `dctlength` desired. Ultimatley, this `trdata_raw.mat` data will be used to find the closest subject DCT signal to a test image in the face identification system in Part 4.

D. Part 4: Performance Evaluation of the Face Identification System

The final stage of this project was incorporating all of the knowledge gained in the previous steps to develop a custom face identification system. The method developed used the resultant `trdata_raw.mat` from `face_recog_knn_train.m` in order to generate a 200xN matrix, where N is the desired `dctlength`. The identification system MATLAB™ scripts developed (Listings 4 and 5 used k nearest-neighbor as the basis of classifying test images as belonging to a certain subject. The `classifyFaces` algorithm worked as follows:

1) A desired `dct_coef` and `k` are set.
2) `face_recog_knn_train` is called to generate the trained dataset.
3) Each test image file is iterated and transformed into a 1-D DCT using `findfeatures`
4) Each row in `trdata_raw` which corresponds to 1-D DCT of training data is iterated through and compared to each test image.
5) The `knnsearch` function is used to find the index of the nearest neighbor to the test 1-D DCT vector from `trdata_raw`. This index is used to find the corresponding subject number. This step is repeated k-times and the classes of the k nearest neighbors are added to a vector.
6) The mode of the k nearest neighbor vector is taken to find which class occurs the most in the k set and this class is used to classify the test image.
7) A counter keeps track each iteration and a correct variable is incremented every time the classification matches the subject number. The final `success_rate` is returned as the fraction of times that the classification system successfully classifies a test image.

A second script (Listing 4) is used to generate a 3-D data plot that compares the values for `k` and `dctlength` and the resultant `success_rate` to find what the optimal values of these parameters yield the greatest success (Figure 12).

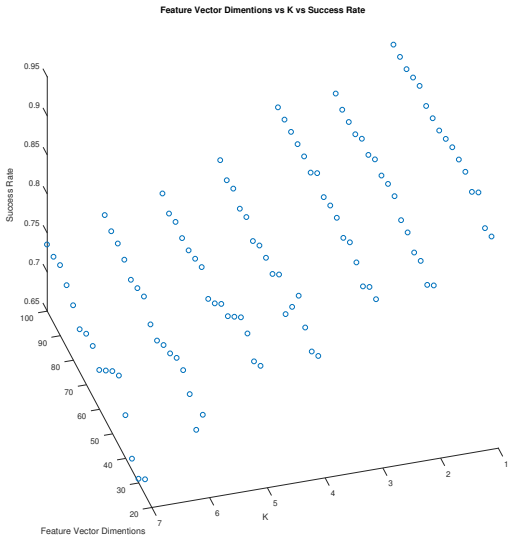

Fig. 12. Feature Dimensions vs K vs Success Rate

The 3-D plot determined the relationship between `k`, `dctlength`, and `success_rate` to be success was greatest when k is 1 and the number of DCT dimensions was as high as possible.

III. Summary and Conclusions

This facial image classification project was successful in completing each of the objectives. The final image classification system was able to achieve 96% classification of the 200 test images of the 40 subjects. The classification method was able to incorporate the theory and methods used in the other three parts to create the final system.

The final classification test was able to conclude that the best discrimination parameters is to have $k = 1$ and the greatest number of 1-D DCT dimensions. Additionally, this project showed that the kNN classifier algorithm is both effective and fast as it was able to run through the 200 images, each with 100 1-D DCT dimentions in under 2 seconds and able to classify correctly within only a 4% margin of error.

This classification system can be improved upon by exploring more classification methods such as applying weights to certain DCT coefficients in order to use the mode of the k nearest neighbors in a more effective way, for instance, taking into account that the first and second nearest neighbors can differ only slightly and in which case, they would have a similar weight in the classification process. Also, more advanced deep learning training algorithms could also be investigated and compared to the current method to see which method is fastest and/or most successful.

References

[1] Y. Xiao, N. Chandrasiri, Y. Tadokoro and M. Oda, "Recognition of facial expressions using 2D DCT and neural network", *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 82, no. 7, pp. 1-11, 1999.

[2] Y. Xiao, L. Ma and K. Khorasani, "A New Facial Expression Recognition Technique using 2-D DCT and Neural Networks Based Decision Tree",*The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006.
[3] M. Peterson and M. Eckstein, "Perceptual learning of discriminating features for facial recognition", *Journal of Vision*, vol. 6, no. 6, pp. 154-154, 2010.
[4] Y. Xu, Q. Zhu, Z. Fan, M. Qiu, Y. Chen and H. Liu, "Coarse to fine K nearest neighbor classifier", *Pattern Recognition Letters*, vol. 34, no. 9, pp. 980-986, 2013.
[5] G. Seo, J. Lee and C. Lee, "Frequency sensitivity for video compression", *Optical Engineering*, vol. 53, no. 3, p. 033107, 2014.
[6] N. Bouaynaya, "Continuous-Time Signals", Rowan University, 2016.

APPENDIX

```
% part_1.m
[img,map] = imread('../att_faces/s1/1.pgm');
img2dct=dct2(img);
imgrecover=idct2(img2dct);
% Compute and plot log magnitude of 2-D DCT
t1=0.01.*abs(img2dct);
t2=0.01*max(max(abs(img2dct)));
c_hat=255*(log10(1+t1)/log10(1+t2));

figure
hold on
imshow(img,map);
title('Face 1');
print ('../report/img/part_1_face.eps', '-deps',...
'-r100');
hold off
close all

figure
hold on
imshow(img2dct,map);
title('2D DCT Plot of Face 1');

print ('../report/img/part_1_dct.eps', '-deps',...
'-r100');
hold off
close all

figure
hold on
imshow(imgrecover,map);
title('Face 1 Recovered');
print ('../report/img/part_1_recovery.eps', '-deps',...
'-r100');
hold off
close all

figure
hold on
% Compute and plot log magnitude of 2-D DCT
t1=0.01.*abs(img2dct);
t2=0.01*max(max(abs(img2dct)));
c_hat=255*(log10(1+t1)/log10(1+t2));
imshow(c_hat,map);
title('Log Magnitude of 2-D DCT');
print ('../report/img/part_1_log_dct.eps', '-deps',...
'-r100');
hold off
close all
```

Listing 1. part_1.m

```
% part_1.m
features_3_9 = [];
features_3_35 = [];
features_3_100 = [];

features_4_9 = [];
features_4_35 = [];
features_4_100 = [];

for i = 1:9
 subject = ['../att_faces/s3/', num2str(i), '.pgm'];
 features_3_9(i,:) = findfeatures(subject, 9);
 features_3_35(i,:) = findfeatures(subject, 35);
 features_3_100(i,:) = findfeatures(subject, 100);
end

for i = 1:9
 subject = ['../att_faces/s4/', num2str(i), '.pgm'];
 features_4_9(i,:) = findfeatures(subject, 9);
 features_4_35(i,:) = findfeatures(subject, 35);
 features_4_100(i,:) = findfeatures(subject, 100);
end


figure
hold on
title('FindFeatures Subject 3: DCT Length = 9');
for i = 1:9
 plot([1:9],features_3_9(i,:));
end
print ('../report/img/part_2_s_3_dct_9.eps', ...
'-depsc', '-r100');
hold off
close all

figure
hold on
title('FindFeatures Subject 4: DCT Length = 9');
for i = 1:9
 plot([1:9],features_4_9(i,:));
end
print ('../report/img/part_2_s_4_dct_9.eps', ...
'-depsc', '-r100');
hold off
close all

figure
hold on
title('FindFeatures Subject 3: DCT Length = 35');
for i = 1:9
 plot([1:35],features_3_35(i,:));
end
print ('../report/img/part_2_s_3_dct_35.eps', ...
'-depsc', '-r100');
hold off
close all

figure
hold on
title('FindFeatures Subject 4: DCT Length = 35');
for i = 1:9
 plot([1:35],features_4_35(i,:));
end
print ('../report/img/part_2_s_4_dct_35.eps', ...
'-depsc', '-r100');
hold off
close all

figure
hold on
title('FindFeatures Subject 3: DCT Length = 100');
for i = 1:9
 plot([1:100],features_3_100(i,:));
end
print ('../report/img/part_2_s_3_dct_100.eps', ...
'-depsc', '-r100');
hold off
close all

figure
hold on
title('FindFeatures Subject 4: DCT Length = 100');
for i = 1:9
 plot([1:100],features_4_100(i,:));
end
print ('../report/img/part_2_s_4_dct_100.eps', ...
'-depsc', '-r100');
hold off
close all
```

Listing 2. part_2.m

```
% part_3.m

subject_range=[1 40];
dct_coef = 70;
[trdata_raw,trclass]=face_recog_knn_train(subject_range,dct_coef);

trdata_raw
```

Listing 3. part_3.m

```
% part_4.m

data_points = []
count = 0;
for k_val = 1:1:7
 for dim_val = 25:5:100
  [kNN_identity, success_rate] = ...
  classifyFaces(dim_val, k_val)
  data_points(count + 1,:) = ...
  [dim_val, k_val, success_rate]
  count = count + 1;
 end
end

hold on
title("Feature Vector Dimentions vs K vs Success Rate");
xlabel("Feature Vector Dimentions");
ylabel("K");
zlabel("Success Rate");
scatter3(data_points(:,1), data_points(:,2), data_points(:,3))
print ('../report/img/part_4_3d_plot.eps', ...
'-depsc', '-r100');
hold off
close all
```

Listing 4. part_4.m

```
% classifyFaces.m

function [kNN_identity, success_rate]...
= classifyFaces(dct_coef, k)

subject_range=[1 40];
f_range=subject_range(1):subject_range(2);
%dct_coef = k;
[trdata_raw,trclass]=face_recog_knn_train(subject_range,dct_coef);

correct = 0; % counts number of correct classifications
count = 0;

nsubjects = 40;

for i=1:nsubjects

 for j=6:10

  % Assign the filename for processing
  name = ['../att_faces/s'...
  num2str(f_range(i)) '/' num2str(j) '.pgm'];

  true_label = i;

  % Run "findfeatures" which returns a DCT vector (face_feat) with the
  % length defined in dct_coef.
  face_feat(j,:)=findfeatures(name,dct_coef);

  rawCopy = trdata_raw;
  nearest_classes = zeros(k,1);
  for x = 1:k
   IDx = knnsearch(rawCopy,face_feat(j,:));
   rawCopy(IDx, :) = []; % remove matched row
   nearest_classes(x) = trclass(IDx);
  end

  nearest_classes
  kNN_identity(count + 1) = mode(nearest_classes);

  if(kNN_identity(count+1) == true_label)
   correct = correct + 1;
  end

  count = count+1;
end
end
success_rate = correct / count;
```

Listing 5. classifyFaces.m

```
function [result] = findfeatures(filename,dctlength)
%-------------------------------------------------
%
% function [result] = findfeatures(filename,dctlength)
% filename: example 's1.pgm'
% dctlength: length of desired dct
% example: [result]=findfeatures('s1.pgm',35);
%
%-------------------------------------------------
% Read image
[ingresso]=imread(filename);
Lout = dctlength;

% Do zig-zag scanning
[dimx,dimy] = size(ingresso);
dimprod     = dimx*dimy;
zigzag = zeros(dimx,dimy);
ii = 1;
jj = 1;
zigzag(ii,jj) = 1;
slittox = 0;
slittoy = 1;
last    = 0;
cont    = 2;

while cont<dimprod
    if slittox == 1 && (ii+1)<=dimx
        ii = ii+1;
        jj = jj;
        zigzag(ii,jj)=cont;
        cont = cont+1;
        slittox = 0;
        last    = 1;
        continue;
    end
    if slittox == 1 && (ii+1)>dimx && (jj+1<=dimy)
        ii = ii;
        jj = jj+1;
        zigzag(ii,jj)=cont;
        cont = cont+1;
        slittox = 0;
        last    = 1;
        continue;
    end

    if slittoy == 1 && (jj+1)<=dimy
        ii = ii;
        jj = jj+1;
        zigzag(ii,jj)=cont;
        cont = cont+1;
        slittoy = 0;
        last    = 0;
        continue;
    end
    if slittoy == 1 && (jj+1)>dimy && (ii+1<=dimx)
        ii = ii+1;
        jj = jj;
        zigzag(ii,jj)=cont;
        cont = cont+1;
        slittoy = 0;
        last    = 0;
        continue;
    end

    if (slittox == 0 && slittoy == 0) && last == 1
        if ii-1>=1 && jj+1<=dimy
            ii=ii-1;
            jj=jj+1;
            zigzag(ii,jj)=cont;
            cont = cont+1;
            continue;
        else
            slittox = 0;
            slittoy = 1;
```

```
            continue;
        end
    end

    if (slittox == 0 && slittoy == 0) && last == 0
        if ii+1<=dimx && jj-1>=1
            ii=ii+1;
            jj=jj-1;
            zigzag(ii,jj)=cont;
            cont = cont+1;
            continue;
        else
            slittox = 1;
            slittoy = 0;
            continue;
        end
    end
end
zigzag(dimx,dimy)=dimprod;
%-----------------------------------------------------
%-----------------------------------------------------
t                = dct2(ingresso);
vettore_t        = t(:);
vettore_t_zigzag = zeros(size(vettore_t));
vettore_zigzag   = zigzag(:);
for ii=1:length(vettore_t)
    vettore_t_zigzag(vettore_zigzag(ii)) = ...
    vettore_t(ii);
end
result = vettore_t_zigzag(2:Lout+1);
%-----------------------------------------------------
%-----------------------------------------------------
%-----------------------------------------------------
```

Listing 6. findfeatures.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function
% [trdata_raw,trclass]=face_recog_knn_train
%                   (subject_range,dct_coef)
% Name: face_recog_knn_train
% Function: Create a matrix of training vectors from
%       training photos to be used for KNN recognition
%
% Input: subject_range - range of faces to be used,
%                        maximum is 40.
%                        Usually input as a vector.
%                        For example [1 29] means
%                        subjects 1 to 29 inclusive.
%                        The first entry must be a 1.
%
%       dct_coef - length of the feature DCT vector
%                        used for comparison
% Output:
%       trdata_raw - trainng data of DCT vectors
%       trclass - class labels for each training data
%       vector Run: Loop through the user defined
%       number of subjects wanted (defined in the
%       subject range) and create a matrix of length #
%       of subjects x length of DCT (defined by the
%       dct_coef).
% Output file: Mat file that includes the training
% vectors with corresponding labels. This file will be
% used in performance evaluation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  [trdata_raw,trclass]=...
face_recog_knn_train(subject_range,dct_coef)

% Assign the vector f_range to the range of subject
% specified by subject_range
f_range=subject_range(1):subject_range(2);

% Check if subject_range(1) = f_range(1) = 1
if (f_range(1) ~= 1)
  error('The first subject must have a label of 1');
end

% Assign the number of subjects to the length of
% f_range
nsubjects = length(f_range);


% Initialize trdata_raw and tr_class trdata is the
% training data matrix trclass is the vector of class
% labels associated with the training data
trdata_raw=[];
trclass=[];

% Loop through the number of subjects
for i=1:nsubjects

% Loop through the first five faces in the subject
% folders. In this experiment, the first five faces are
% used for training.
    for j=1:5

% Assign the filename for processing
        name = ['../att_faces/s'...
            num2str(f_range(i)) '/' num2str(j) '.pgm'];

% Run "findfeatures" which returns a DCT vector
% (face_feat) with the length defined in dct_coef.
        face_feat(j,:)=findfeatures(name,dct_coef);
    end

% Add the five face_feat vectors to the end of trdata_raw.
trdata_raw=[trdata_raw face_feat(1:5,:)'];

% Add the corresponding label for the five face_feat vectors.
trclass=[trclass i*ones(1,5)];

% End of for i=1:nsubjects loop
end

% Switch the columns and rows of trdata_raw and trclass
trdata_raw=trdata_raw';
trclass=trclass';

% Save the variables (dct_coef, f_range, nsubjects,
% trclass, trdata_raw
save ...
('raw_data.mat','dct_coef','f_range','nsubjects','trclass','trdata_raw')
```

Listing 7. face_recog_knn_train.m