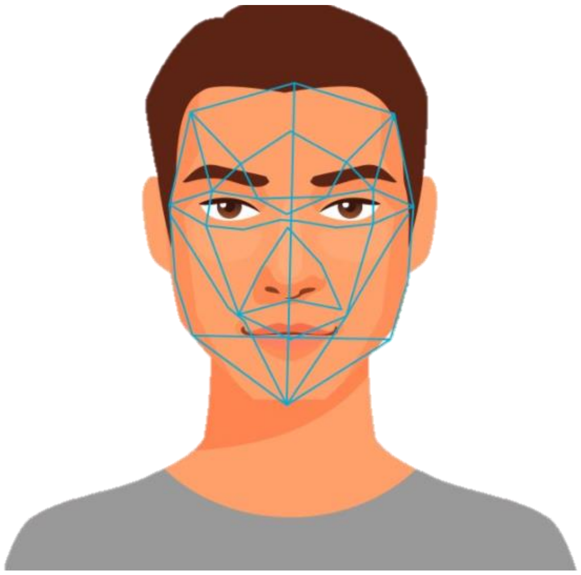


Laboratory Project on Face Identification System

Dr. Bhavani, Signals and System, Section 1

Thai Nghiem
Rowan University Class of 2019
Electrical & Computer Engineering Major
Computer Science Minor
Rowan ID: 916186794

Table of Contents

Introduction

Every face is different and each person holds certain features that can make us all unique. Luckily, the ability to recognize faces is innate for most people and other primates. It is an inherent skill that our brains just know how to do immediately from birth. Distinguishing human faces has been a skill that each human possess and trains over many years. Despite the fact that the simple act of recognizing a face appears to be insignificant to humans, only recently has technology possessed the ability to do this as well.

Without a doubt, facial recognition technology is a substantially developed field. In recent years, numerous advancements in algorithm and technology have risen as to instruct computers to recognize a human's face using two dimensional (2-D) pictures. The applications for this technology are very important and have a wide range of uses, such as the Facebook auto-tagging feature and police uses, such as identifying wanted criminals through security footage in public areas.

Since the utilization of this technology is very broad and continuously growing, it is crucial to comprehend the concepts of this area and fundamental ways computers have been using to classify differences and similarities. One of the most important ways a computer can perceive a face is through geometrically, utilizing features, such as the exact space between eyes, the precise curve of the cheek or the fullness of the lips. Another fundamental method that a computer uses is 'linear discriminant analysis'. This method utilizes a more mathematical approach, such as the kNN arrangement, as to recognize a face using the pixels of a picture. This report will concentrate on the later strategy for facial identification.

In spite of the fact that the facial recognition technology has become quite advanced, there is still significantly more room to develop. According to Alessandro Acquist, a professor at Carnegie Mellon, there are three main problems facing facial recognition's effectiveness at the moment. First, there are not enough databases of people's faces to recognize a random person walking down a street. Second, databases that are large enough tends to 'confuse' computers and may result in many 'false positives' where objects are identified incorrectly. Lastly, computer algorithms aren't fast enough to be used in some situations [1]. These current weaknesses are shaping the current work done in improving facial recognition and explains the necessity of this project.

The objective of this project was to create and run a facial identification system in MATLAB using a provided AT&T database. In order to achieve this, the use of the Discrete Cosine Transform (DCT) of an image and a 'k' nearest neighbor (kNN) classifier is necessary to be understood. Next, the created system will be assessed in order to find the success rate of the system. The result will then be plot in 3-D plot to identify the best values of ‘k’ and dimension feature vector. The observation of this system can likewise help extend the comprehension of facial identification innovation in general.

[1] Meyer, R. (2015, July 02). Who Owns Your Face? Retrieved November 25, 2017, from https://www.theatlantic.com/technology/archive/2015/07/how-good-facial-recognition-technology-government-regulation/397289/

Protocol, Results, and Discussion

Part 1 – Two Dimensional (2-D) DCT and inverse DCT

In order to be familiarized with the AT&T database, a picture is picked out to be examined. In the database, there is a total of 40 subjects, with 10 images per subject, which comes out to 200 images for the entire database. Each of these images is similar in the dark homogenous background with upright, frontal position, but varies in lighting, facial characteristics, and facial expressions. All pictures are saved as PMG format, so that it can be conveniently read into MATLAB. Each image has a size of 112 by 92 pixels, with 256 levels per pixel.

When one of these pictures is read, a 112 x 92 matrix, which has values range from 0 to 256, is created. The size of the matrix represents the size of the image in pixels, and the values contained in the matrix associate with the grayscale value of each pixel. In here, grayscale is a range of shades of gray without apparent color, and gray scale values determine the intensity of a pixel from black to white [2]. The darkest shade of gray, which is black, is represented as 0, while the lightest shade of gray, which is white, is represented as 256. Of course, the grayscale values in between signify various shades of gray amongst black and white. The pictures seen in MATLAB use these grayscale values by taking the values stored in the 112 x 92 matrix and color each corresponding pixel in the 112 x 92 image.

After gaining an understanding of grayscale, the first image of subject 7 is picked out from the database to be read and plotted. This is done using MATLAB, which produces the result as shown in Figure 1 below.

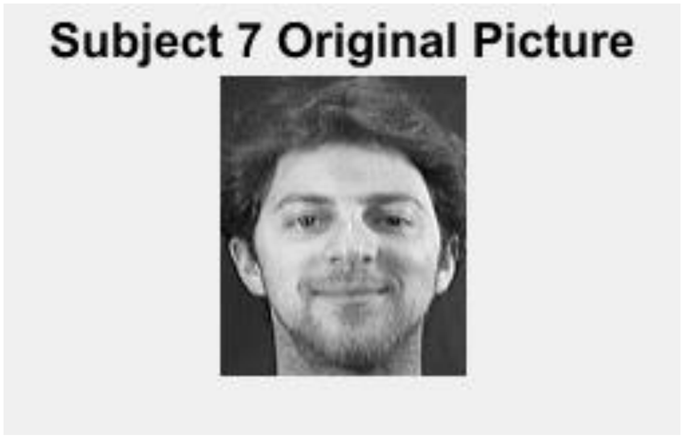


Figure 1. Subject 7- Image 1

Next, by using the dct2() MATLAB function, the Two-Dimensional Discrete Cosine Transform (2-D DCT) of the picture is found and generated. The discrete cosine transforms (DCT) is a technique that represents an image as a sum of sinusoids of varying magnitudes and frequencies. This technique is similar to the Fast Fourier transform, but more efficient in the way that it can approximate lines with fewer coefficients. Subject 7's 2D DCT is presented in Figure 2.

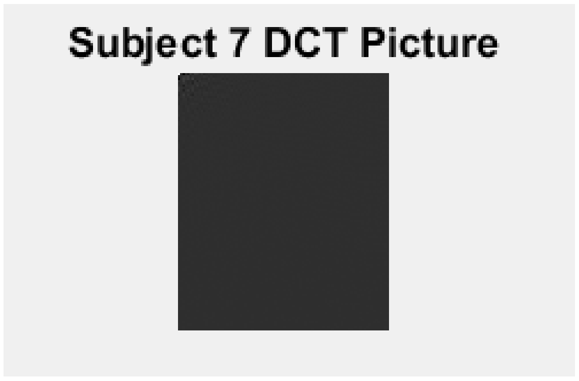


Figure 2. Subject 7 - Image 1 2-D DCT

From the above figure, the result seems to be a plain gray image, with no apparent different, except for a slight different in the upper left-hand corner. This difference, however, is hard to detect.

In order to recover the original picture, the inverse cosine transform (IDCT) is needed. This can be done using MATLAB idct2() function, and the result is shown in Figure 3 below.

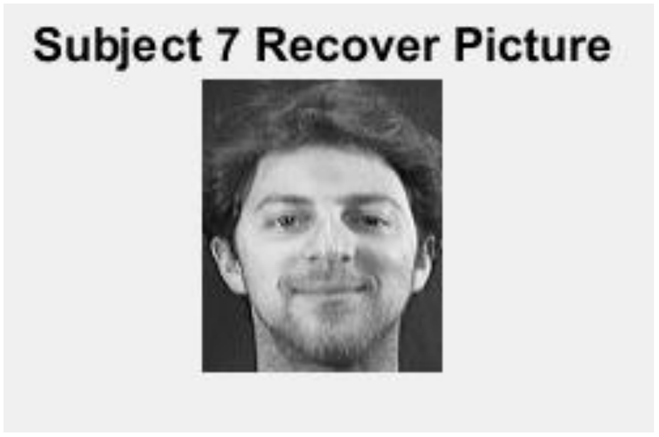


Figure 3. Subject 7 - image 1 Recover Picture

It can clearly be seen that the photo of subject 7 is fully recovered using IDCT, so it can be concluded that for any given DCT image, the original image can be recreated without any additional information.

Since it was not clear that the DCT coefficients magnitude of the upper left-hand corner is larger than the rest of the image, a script that finds the log magnitude of these coefficient is given. From taking the log of the DCT coefficients, the differences between them are more obvious. The resulting plot is shown in Figure 4.

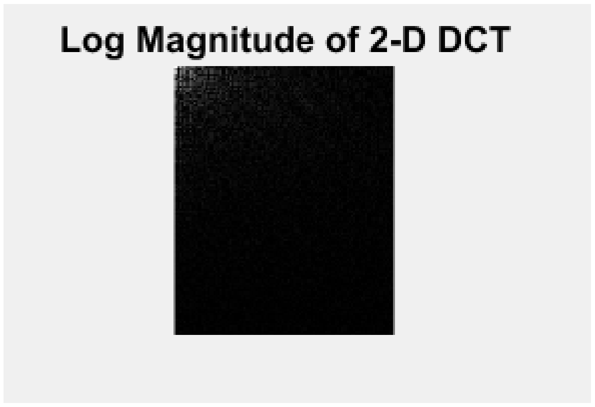


Figure 4. Subject 7 - Image 1 Log magnitude

Figure 4 clearly shows that the pixels located in the upper left-hand corner of the picture are a lot lighter than the rest of the picture's pixels. This means that the values in that area of the matrix is bigger than the other area of the matrix.

From the first part of the project, knowledge of 2-D DCT is fully learned and is put into practice. It is understood that the DCT coefficients of an image can be utilized to reconstruct the original image, and vice versa using MATLAB functions. Notably, the largest DCT values are generally located in upper left-hand corner of the matrix. The information yield from this part is critical to the rest of the project.

Part 2 – Feature Extraction

A program called "findfeatures.m" is supplied in order to develop an in-depth understanding of how to convert a picture into a DCT feature vector of a specific dimension. This program takes 2 inputs, a 2-D picture and a length, and outputs a one-dimensional DCT feature vector. By utilizing a 'zigzag' pattern shown in Figure 5 when scanning over the image, the function can generate a 1-D vector from the 2-D DCT matrix of the picture. As a result, a 1-D vector with a length size equals to the area of the input picture is created.

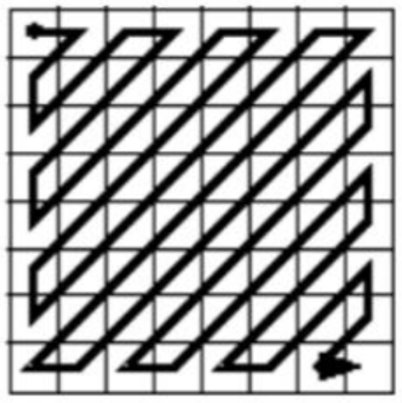


Figure 5. The 'zigzag' scanning pattern.

In order to achieve the 1-D vector, the given function makes a new matrix named 'ingresso' that holds the grey-scale values matrix of the target picture. Also, a matrix called 'zigzag', which has the same size as 'ingresso', and a variable called 'dimprod' are made. 'dimprod' contains the area of the grayscale matrix, as it will be the last index of the 1-D feature vector output. Next, using a while loop and if-else statements, each value in 'zigzag' is labeled with its index for the 1-D output vector. For instance, value at (row 1, column 1) is labeled 1, value at (row 1, column 2) is labeled 2, value at (row 2, column 1) is labeled 3, etc... The way in which each value in 'zigzag' matrix is labeled can be understood graphically using Figure 5. At the end of the while loop, a total three vectors are made. The first vector – vector 'vettore_t' – holds the values of the input 2-D DCT matrix by reading this matrix from top to bottom and left to right. The second vector – vector 'vettore-zigzag' - contains the values in the 'zigzag' matrix by also reading the matrix from top to bottom and left to right. The last vector – vector 'vettore_t_zigzag' – is a zero matrix which has the same length as vector 'vettore_t'. These 3 vectors are used in the next for loop, which takes each value in 'vettore_t' and places it in 'vettore_t_zigzag'. The index of vector 'vettore_t_zigzag' is determined by the value of 'vettore-zigzag'. After this for loop, the vector 'vettore-zigzag' is filled with the values of the input 2-D DCT in the order of the 'zigzag' scanning pattern, and with 'dimprod' being the length. Finally, vector 'vettore-zigzag' is used as the output of the function, with only the first "Lout" samples being output. With these three vectors being made, another loop is implemented. In this loop, each value in vettore_t from index 1 to the end, is placed in in vettore_t_zigzag, the index of vettore_t_zigzag that the value of vettore_t is placed in is determined by the value of vettore_zigzag, from index 1 to the end of vettore_zigzag. The vector vettore_t_zigzag is now filled with the values of the 2-D DCT of the image in order of the zigzag scanning pattern, with length dimprod. The function then outputs the first 'Lout' samples of vettore_t_zigzag, with 'Lout' being the specified length of the feature vector, inputted by the user. This then results in an output of the image's feature vector of a specified length.

Function "findfeatures.m" is then utilized to find the 1-D feature vector at different lengths for subject 7's second image. The results are shown in Figure 6, 7 and 8, with the length being 9, 35 and 100, respectively.

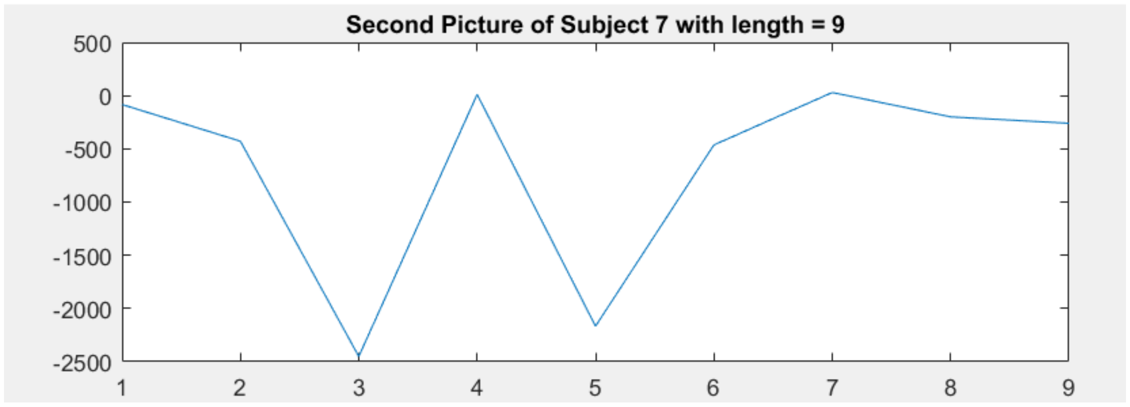


Figure 6. 1-D Feature Vectors of Dimension 9 of Subject 1's Second Image

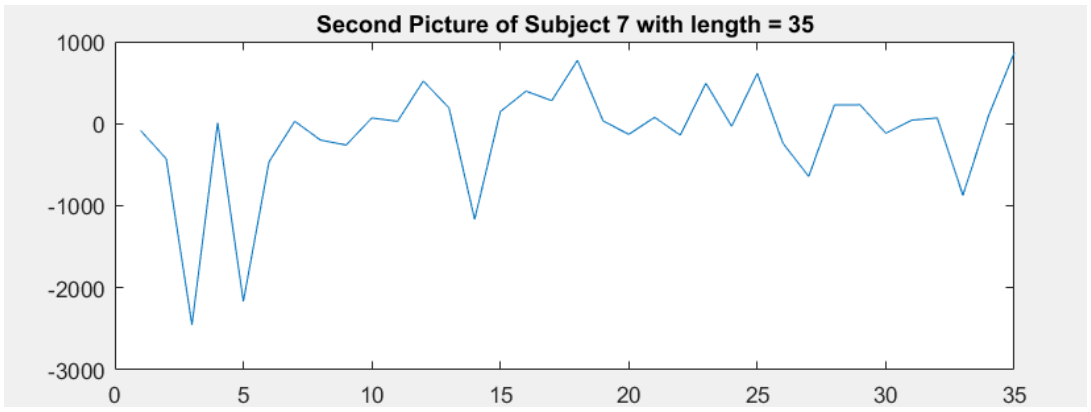


Figure 7. 1-D Feature Vectors of Dimension 35 of Subject 1's Second Image

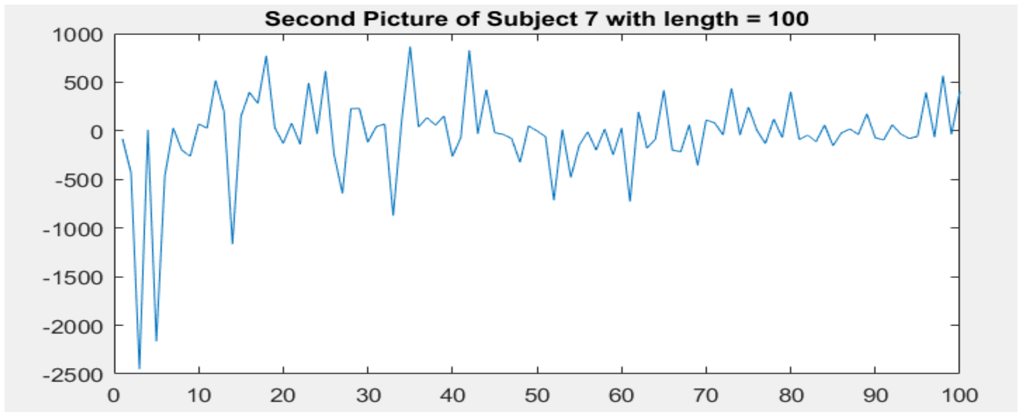


Figure 8. 1-D Feature Vectors of Dimension 100 of Subject 1's Second Image

It can be observed that the DCT coefficients are larger at the beginning of the 1-D feature vector. This is because the beginning of the vector is the upper left-hand area of the image, which shown in part 1 to have the biggest DCT coefficients magnitude. Additionally, the 1-D feature vectors at different lengths are found for subject 9's second image. The results of the plots are represented in Figure 9, 10 and 11, where Figure 9 represents a length of 9, Figure 10 represents a length of 35, and Figure 11 represents a length of 100.

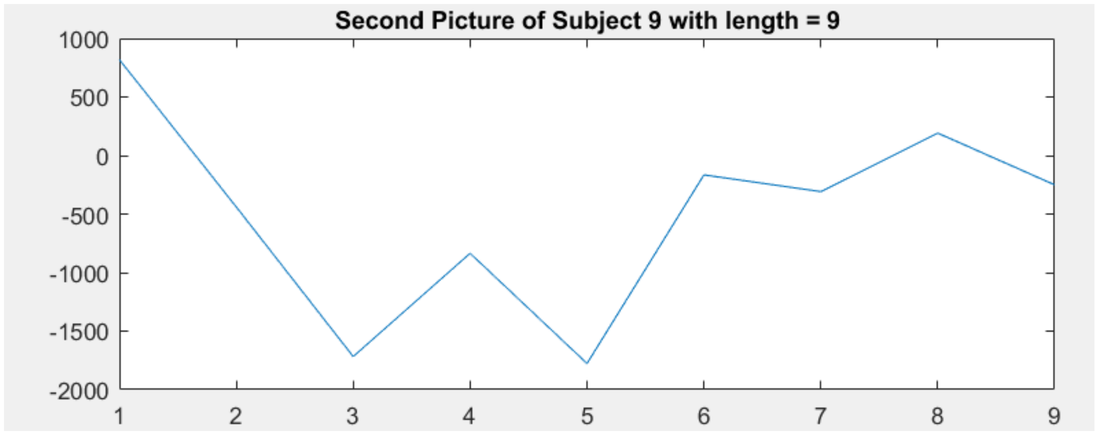


Figure 9. 1-D Feature Vectors of Dimension 9 of Subject 9's Second Image

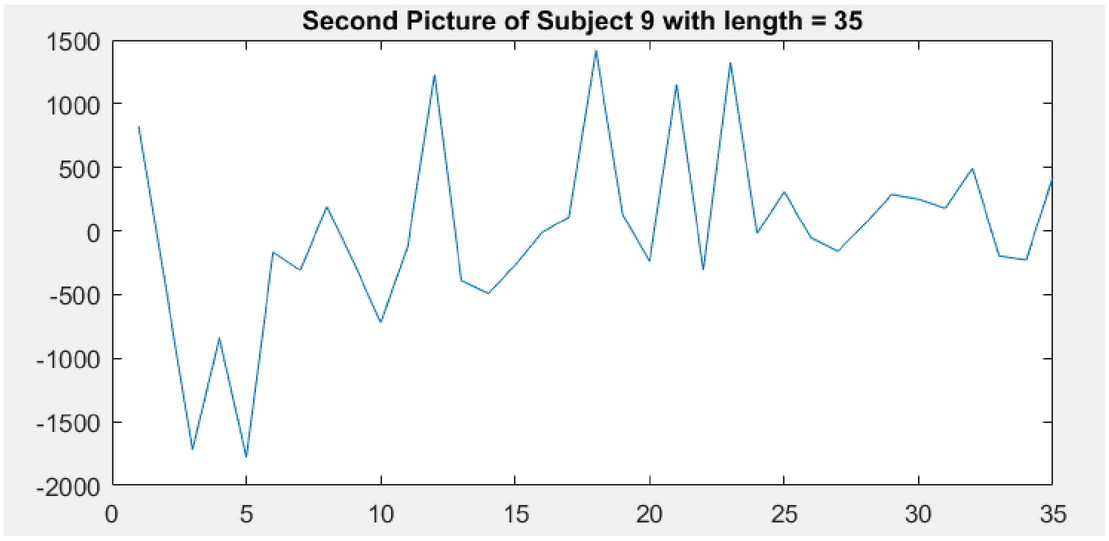


Figure 10. 1-D Feature Vectors of Dimension 35 of Subject 9's Second Image

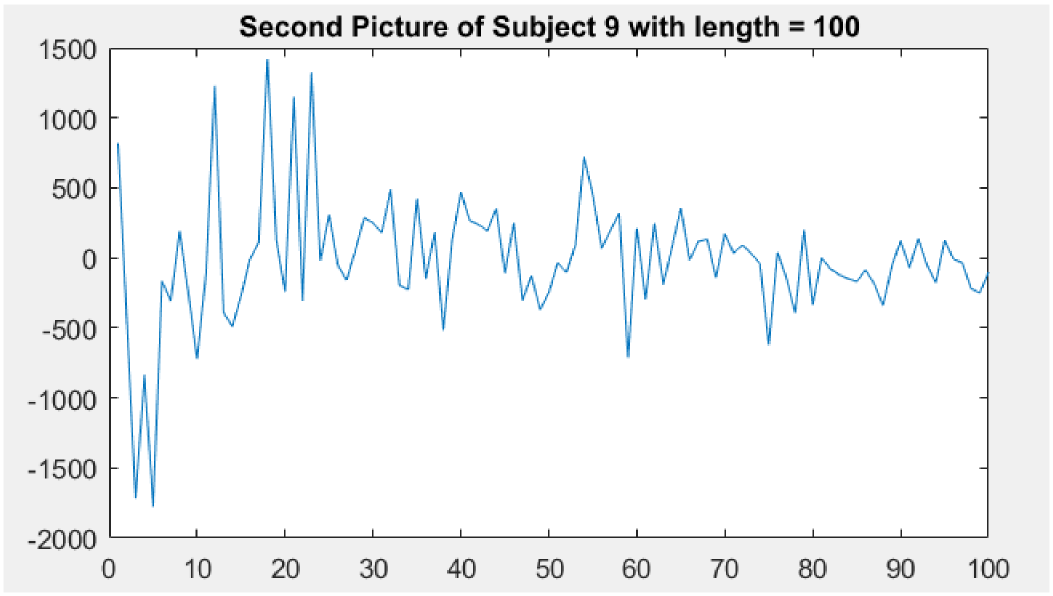


Figure 11. 1-D Feature Vectors of Dimension 100 of Subject 9's Second Image

After observing these plots, it is concluded that: even though the 2 subjects' 1-D feature vectors are similar at the length of 9, they are quite distinct-able at length 35 and especially at the length of 100. The differences can be found in the magnitude and the shape of the 1-D feature vectors. As a result, these 1-D feature vectors can be used to distinguish between 2 different people.

Moreover, despite of the differences between two different subjects' 1-D feature vectors, these vectors do possess some similarities in shape and amplitude. This means that the feature vectors also can be used to find whether a photo contains a face or not.

Part 3 – Training the Face Identification System

In order to train the face identification system, a function called 'face_recog_knn_train.m' is given. This program is used to configure the kNN classifier class that will be written in Part 4. The function takes in first 5 pictures of each subject and processes them as templates to be compared with for the kNN classifier. As to achieve this, the function has 2 parameters: 'subject_range', which is a matrix representing which subject to be used, and 'dct_coef', which set the length for the feature vector used for comparison. First, the function loops through the first 5 pictures of each subject and apply the 'findfeature.m' function to get the feature vectors for each picture. Theses feature vectors, which have the length of dct_coef specified by the user, are then added to a matrix called 'trdata_raw'. To identify which subject these feature vectors belong to, an array called 'trclass' is created. 'trclass' contains the number label corresponding to the number subject in the AT&T database has. After running loop on 40 subjects, the function saves 5 variables (nsubjects, trclass, dct_coef, f_range, and trdata_raw) in a file named 'raw__data.mat'. Here, 'f_range' is the range of subjects being trained, while 'nsubjects' is the number of subjects being trained.

To serve the purpose of this project, the function is run over a subject range of 1 to 40, and a DCT coefficient of 70. The resulting 'raw_data.mat' file is then observed. According to this file, 'dct_coef' equals to 70, as specified by the input parameter, representing the length of the feature vectors. The variable 'f_range' is a 1 x 40 matrix that contains the range of subjects from 1 to 40. The variable 'trdata_raw', which has the 70-length feature vector of each picture, is a 200 x 70 matrix. It has 200 rows because there are 5 pictures for each of the 40 subjects, and each of these 200 pictures has its own feature vector. And as expected, the variable 'trclass' also has 200 rows, as it contains the labels for each feature vectors in 'trdata_raw'. Lastly, the variable 'nsubjects' equals 40, since all 40 subjects in the database is utilized to train the system. This process presents how the feature vectors are configured in the kNN classifier system.

Part 4 – Performance Evaluation of the Face Identification System – Clean Data

At last, the kNN classifier system is built with the end goal of executing face identification. The targets for recognition are the last 5 images of 40 subjects in the AT&T database. The system needs to determine which subject each of these images corresponds to. The accuracy of the facial recognizing system is then found to find the optimized parameters.

A function named 'kNN_classifier' is created in MATLAB in order to achieve the given task. This function utilizes the 'raw_data.mat' file created in Part 4 to get the important information such as the matrix 'trdata_raw' and matrix 'trclass'. The function only takes in 1 parameter, 'k', which will determine how many 'k' nearest neighbors are going to compared with the image's feature vector of the 5 last images of each subject. Eventually, the subject of each of these last 5 images will be found and labeled in order to compare with the answer subject.

As to achieve this, after getting the information in the 'raw_data.mat' file, the function first loops thorough images 6 to 10 of each subject to determine their feature vectors. These feature vectors are then utilized to find their norm L2 distance from the training vector in matrix 'trdata_raw'. The resulting distances are saved in an array. Next, the 'k' nearest neighbors of the each of the features vectors are found by choosing the smallest distance away from the training vectors. The label for the 'k' closest training vectors is then stored in an array. If the user input more than 1 'k' nearest neighbors, the subject with the most nearest-neighbors is chosen to be the subject that the feature vector belongs to, using ‘mode’ MATLAB function. If there are more than 1 subjects that have the most nearest-neighbors, the subject that appears first in the array will be chosen as the answer. This method is call “kNN break-even” method. Afterwards, the array contains the experimental answer will have 200 values, corresponding to 200 feature vectors being identified.

In the same function, the experimental array is compared with the correct answer array, which is the variable ‘trclass’. This is done by subtracting the experimental array from the correct array and count the non-zero values in the resulting array. These non-zero values are in the incorrect answers given by the function. By taking the percentage of the remaining correct answers, the accuracy of the system is found and outputted.

The function is then run several times using different parameters as to find the optimize the accuracy of the facial recognition system. The first parameter to change is the length of the 1-D feature vector, and the second parameter is the number of ‘k’ nearest neighbors. Accordingly, the ‘dct_coef’ is changed from 25 to 100 with steps of 15, and ‘k’ is changed from 1 to 7 with steps of 2. The results are then plotted using ‘surf’ MATLAB function.

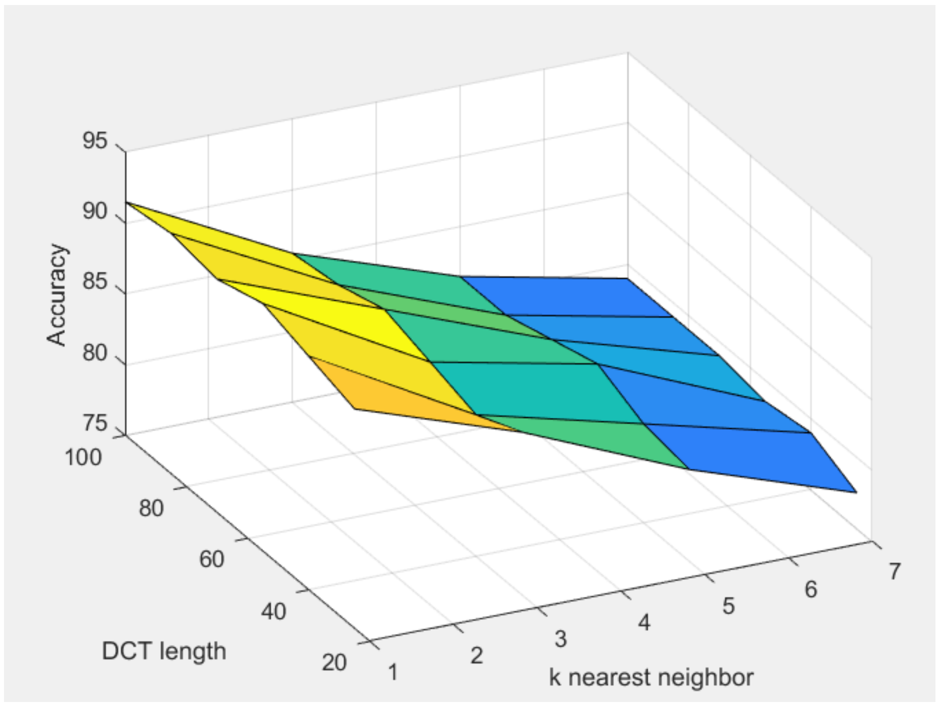
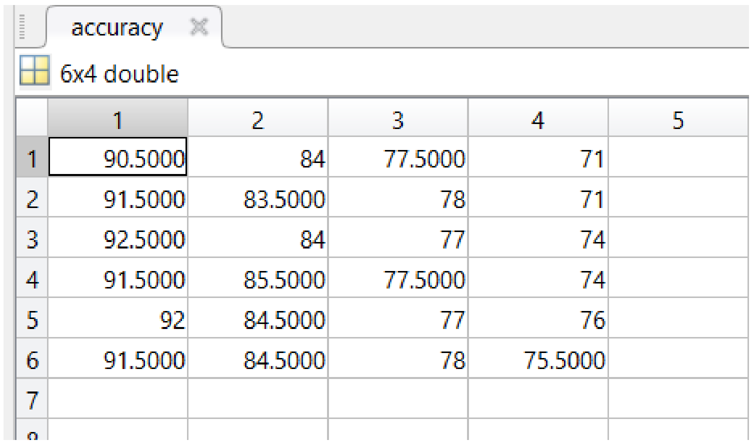


Figure 12. Success rate of kNN classifier with corresponding k nearest neighbors and 1-D feature vector length

Figure 12 shows that varying the length of the 1-D feature vector doesn’t not have a great impact on the accuracy of the kNN classifier. The reason is because the largest magnitude DCT coefficients for each feature vector are located in the beginning of the vector, while the smaller coefficients are located in the end of the vector. As a result, as long as the feature vector contains the largest magnitude DCT coefficients, the system will be able to identify correctly which feature vectors belong to which subject. In contrast, the figure also shows that by increasing the number of nearest-neighbors, we effectively lower the accuracy of the system. This is because finding a bigger number of closest neighbors can bring about a vector being identified as one that has the most vectors near the feature vector being tested, yet does not really have the nearest feature vector. On the other hand, having only one nearest neighbor guarantees that the feature vector that match the trained feature has the most weight in determining which subject the image is of.

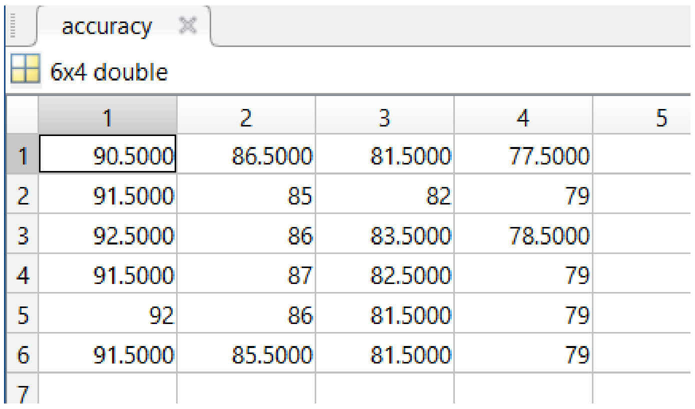
Figure 13 and Figure 14 below will show the accuracy results (represented as tables) to see the differences between the code uses only 'mode' to define kNN and the code uses 'kNN break-even method' mentioned above. In these tables, the column corresponds to number of 'k' (from 1 to 7), and the row corresponds to the DCT length (from 25 to 100).


accuracy

6x4 double

	1	2	3	4	5
1	90.5000	84	77.5000	71	
2	91.5000	83.5000	78	71	
3	92.5000	84	77	74	
4	91.5000	85.5000	77.5000	74	
5	92	84.5000	77	76	
6	91.5000	84.5000	78	75.5000	
7					

Figure 13. kNN Classifer Using Only MATLAB 'mode' function


accuracy

6x4 double

	1	2	3	4	5
1	90.5000	86.5000	81.5000	77.5000	
2	91.5000	85	82	79	
3	92.5000	86	83.5000	78.5000	
4	91.5000	87	82.5000	79	
5	92	86	81.5000	79	
6	91.5000	85.5000	81.5000	79	
7					

Figure 14. kNN classier using 'break-even' method

As it can be seen from Figure 13 and Figure 14, the accuracy at k = 5 and k = 7 is significantly increased when applying ‘break-even’ method. Therefore, this method is chosen to plot the graph represented in Figure 12 above. Also, it is determined that the optimal parameters values to yield the highest accuracy, which is 92.5%, are k equals 1 and 1D feature vector length of 55. These values can be used for later applications and studies.

Summary and Conclusions

After a series of implementing and experimenting, the objectives of this laboratory are successfully achieved and a complete system of facial identification is developed. The system makes use of DCT and kNN classifier as the main concepts, and an AT&T Laboratories face database is then used to evaluate the system. In this database, there are 400 images in total, in which there are 40 subjects with 10 images per subject. At the beginning, by taking one of the subject's image and plotting its DCT, the concepts about DCT is observed and fully understood. Then, in order to reconstruct the original image, the inverse of the DCT image is used. More importantly, by taking the log of the 2-D DCT image's magnitude, it is found the DCT coefficients with large magnitude are concentrate in the upper left-hand corner of the matrix.

In the second part of the project, a program called "findfeatures.m" is utilized to find the 1-D DCT feature vector of an image. Through analyzing the provided code, the way this program converts an image into a DCT feature vector of a specified dimension is by scanning the image in a 'zigzag' pattern while placing each DCT coefficients into the 1-D array. The program is then applied on several subjects of the AT&T database. It is then seen that while each subject has unique feature vectors, they tend to retain similarities in the shape of the vector when being plotted. Interestingly, the feature vectors of different images of one subject are very similar in such manner.

Next, the program called "face_rcog_knn_train.m", which utilizes the find feature program, is studied and applied so as to find the feature vectors of the first 5 images of each subject and label which subject the feature vectors are representing. The results are then used to train the face recognizing system by outputting the data to the file called "raw_data.mat". At the end of the training, the raw_data file contains several pieces of important information, such as the number of subjects, the range of subjects and feature vectors, the feature vectors and their labels, that is needed in order to implement the face recognizing feature.

Finally, a MATLAB program, which implements the DCT and kNN classifier, is written to identify the last five images of each subject so as to examine the accuracy of the system. In this program, first, each subject's last five pictures are looped through to find and label the training vectors that has the least L2 distance from the vector being recognized. As soon as the nearest neighbor is found, the feature vector is labeled as the subject whose training vector is closest to the vector being examined. Afterwards, the success rate of the system, which depends on variable 'k' and the length of the feature vector, is then found. From the experiment, it is determined that a 'k' of 1 produces the highest success rate, while having more 'k' nearest neighbors reduces the success rate since more errors is added to the system. Similarly, it is found that the length of the feature vector does not have a significant impact on the success rate of the whole system. As long as these feature vectors include the coefficients of greatest magnitude (first 25 samples), the accuracy of the system is guaranteed. Last of all, according to the 3-D plot, the best values which yield the highest accuracy (92.5%) for this system are k equals 1 and length equals 55.

By observing, testing and implementing these tasks, a thorough understanding of how a facial identification system runs is developed. With the fundamental knowledge of the steps required to train and execute a kNN classifier, further understandings of how the facial identification systems are implemented in real world applications is also comprehended. In conclusion, facial recognition is an industry that still has room to grow and advance, since its applications in surveillance, social media and photography are vital.

Reference

[1] Meyer, R. (2015, July 02). Who Owns Your Face? Retrieved November 25, 2017, from https://www.theatlantic.com/technology/archive/2015/07/how-good-facial-recognition-technology-government-regulation/397289/

[2] "What is grayscale? - Definition from WhatIs.Com." *WhatIs.com*, whatis.techtarget.com/definition/grayscale.