EE 424 Final Project

Face detection using Eigenfaces and PCA

By: Travis Gillham and Joe Veal

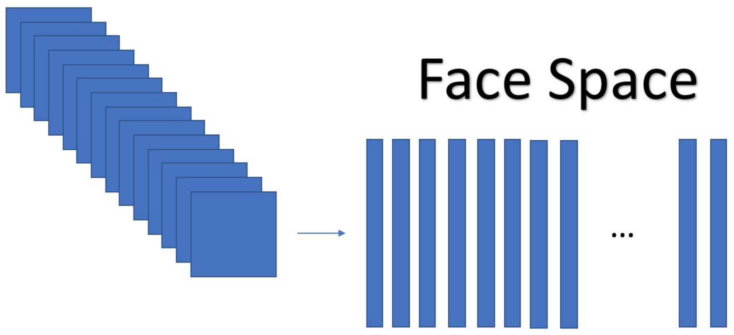
Introduction

For our project, we studied the PCA algorithm. This algorithm uses linear algebra techniques to compress and analyze images. In this report, we will the describe the theory of the PCA algorithm.This will include a mathematical description the linear algebra techniques to be used, as well as an explanation of how the linear algebra help accomplish the task at hand. We will also cover the implementation of a facial recognition system using this method. Then, we will discuss how the algorithm can be implemented as a computer program.

PCA Algorithm

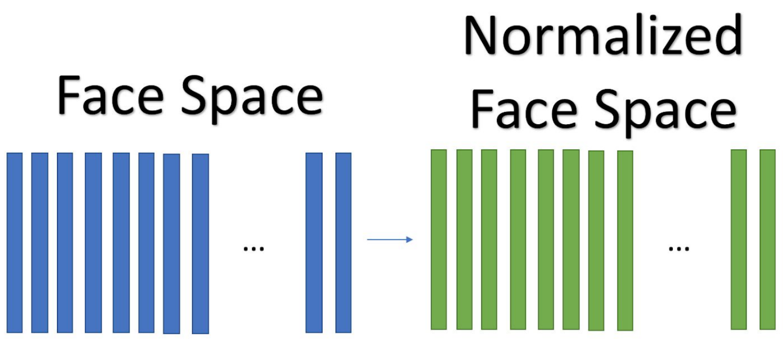
This section will cover the steps to implement the PCA algorithm mathematically. It will cover topics such as the use of eigenvectors and Euclidean Distance. It will talk about the linear algebra multiplication and transpose techniques to compress the images that are being processed.

The first step to implement the pca algorithm is to gather column vectors of data and place them into a large two matrix constructed of column vectors of data. Operation on this matrix will allow for the processing of all of our training data at once.



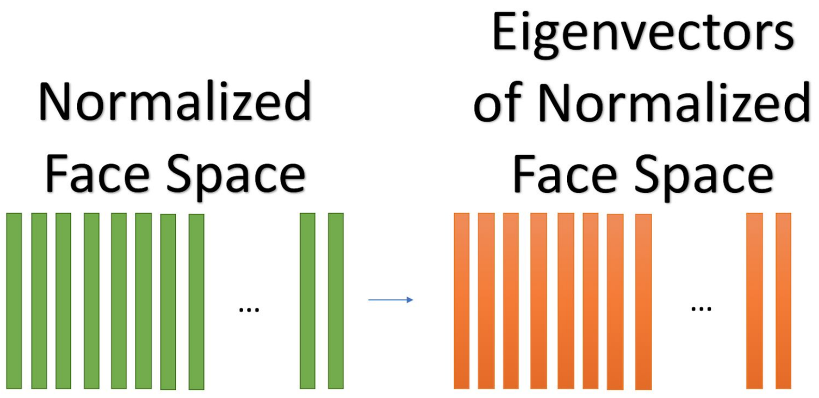
The dataset is now ready to be analyzed. The first step in the algorithm is to calculate the averages of each data point. To do this, the sum of each row of the dataset matrix is taken and divided by the number of columns. This operation will result in a column vector that will represent the average of the dataset.

The next step of the algorithm is to subtract the average of the data set from each column in the dataset matrix. This will result in a matrix that consists of the parts of the data set that are different from the average.



The next step is to calculate the covariance of the difference matrix. This creates a problem that occurs when dealing with large columns of data. When the covariance is calculated, the resulting matrix will be of a enormous size. One way to prevent this error is to take the transpose of the difference matrix before calculating the covariance. This is known as dimensionality reduction, and will make the covariance a more efficient operation.

After the covariance matrix has been found, the eigenvectors of the covariance matrix must be calculated. The number of eigenvectors will be the same as the number of columns in the covariance matrix.



Not all of the eigenvectors are necessary for the algorithm, so eigenvectors corresponding to eigenvalues with small magnitudes are sorted out before any further calculations. Once the sorting is completed, the matrix of eigenvectors must be multiplied by the difference matrix.

The next step is to find the projection of each column of the difference vector onto the facespace. The facespace is the name for the eigenface of the data. The facespace will allow us to find a weight vector, which will be used to compute the distance from the data to be tested against.

One step that can be completed in parallel with the step above is the preparation of the test data. The test data must be evaluated as a column vector. As done in previous steps, average data must be subtracted from the the test data column vector.

Next, the projection of the test data difference vector must be projected onto the face space just as done in previous steps. At this point all of the data has been prepared and is ready to be evaluated.

To evaluate the data that has been prepared, the Euclidean distance between the projected test data column vector and the other column vectors must be calculated. The analysis of the Euclidean distance will determine which set of data is the matches the test data. The smallest Euclidean distance will result in the matching data.

With this now in our bag of knowledge we were able to now try and implement this with our project idea of facial recognition.

Implementation for Facial Recognition

This section will describe how we used the algorithm described above to implement a facial recognition system. We used JPEG images as our data set. We used a dataset that we downloaded from the internet, that contained uniformly sized datasets. Our implementation can be broken down into three steps. The first step is to create a database that can be represented in matrix form. The second step is to find the average image and differance data. This step will also contain the dimensionality reduction as well as the eigenvector calculation. The final step is to analyze the test image and compare its Euclidean distance to the images in the database. The result of this system determines if the user is in fact in the database, and which user is accessing the system.

To implement the database, we used python to read JPEG images and convert them to grayscale. We then reshaped the resulting array to a column vector with the length equal to the number of pixels in the image. We then horizontally concatenate the column vectors to create one matrix that contained all of the image data. This database represents the faces of users whose faces will be approved.

The next function we implemented was called Eigenface Core. In this section we first found the average image using methods described in the algorithm section. The average image was stored as a column vector. We then created a new matrix that was the same size as the database matrix. This matrix contained the difference of each matrix from the average image. This matrix was obtained by subtracting the average image column vector from each column of the database matrix.

$$\Phi_i = \Gamma_i - \Psi$$

$$\begin{gathered} Where, \\ \Phi_i \rightarrow Normalized\ Face\ Vector \\ \Gamma_i \rightarrow Face\ Column\ Vector \\ \Psi \rightarrow Average\ Face\ Column\ Vector \end{gathered}$$

Within the Eigenface Core function, we also compressed the size of the data by using a dimensionality reduction technique. To reduce the dimension, we simply took the transpose of the difference matrix before we took its covariance. Once the data size was manageable, we calculated the eigenvectors and values. We also sorted these eigenvectors to remove any unnecessary information. We followed all of the steps outlined in the algorithm to create this function.

$$\begin{gathered} A = [\Phi_1, \Phi_2, \Phi_3, ..., \Phi_n] \\ C = A^T A \end{gathered}$$

$$C \rightarrow Covariance\ Matrix$$

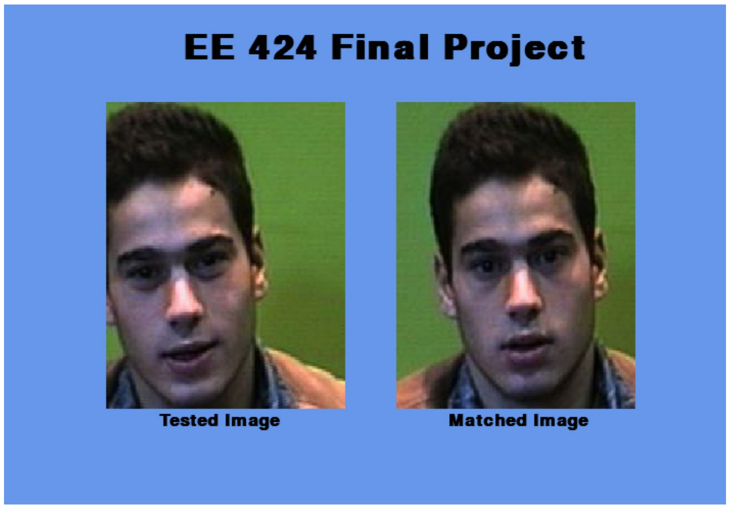
The previous function left us with the following information. A column vector that contained the average image, a difference matrix, and a matrix of eigenvectors. The next step is to compare the data of approved faces with the data from the test image. To do this, we had to create a column vector of the pixels of the test image and subtract the average image from it. We continued to follow the algorithm by finding the projection of the difference of the images from the average onto the Eigenvector matrix. Once we had obtained these projections from the test image and the database images, we compared the Euclidean distance from the test image to each of the database images. The smallest Euclidean distance corresponds to the matching face.

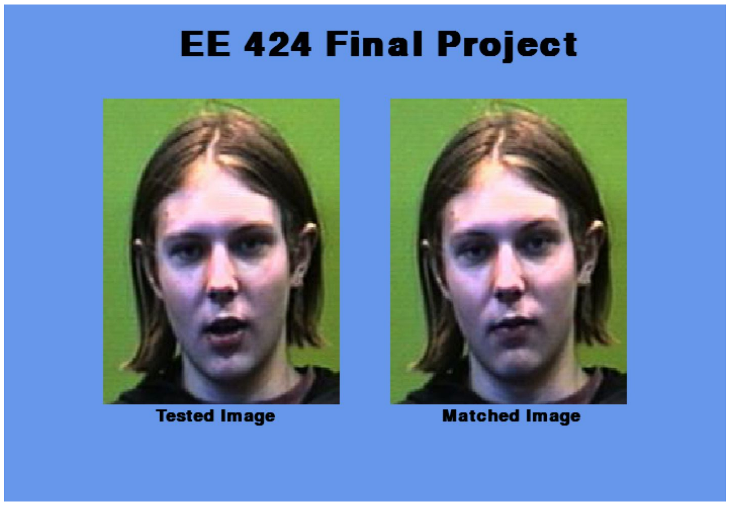
Conclusion

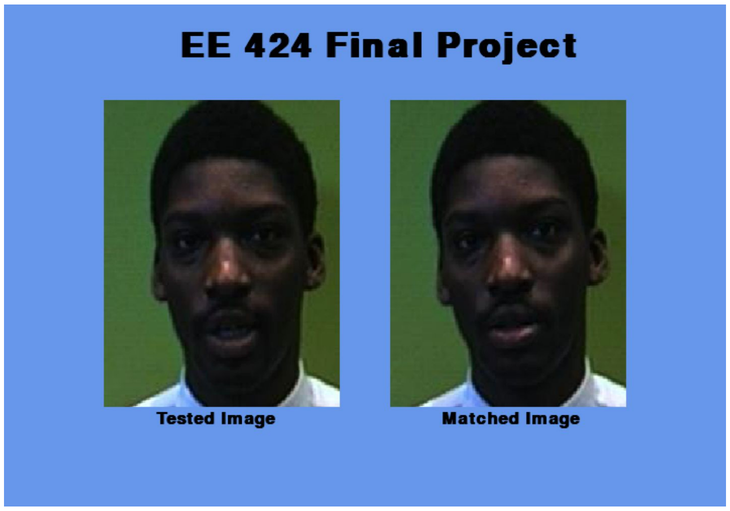
Overall this project was extremely fun as we got to learn, way more than we expected. This will come to be a huge benefit for us in the future as we now have a better understanding of python and PCA which are two things that going into this project neither of us knew. For this project we also used Slack for communication this was also very helpful as we got to share code and ideas in a way that we didn't have to be sitting next to each other to do it. Slack proved to be for us very beneficial. Completing this project was a little bitter sweet. On one hand it was nice to be able to finish the project and get it done with. On the other hand it was also a little saddening to do so, we both put in numerous hours, way more than expected, to finish this project. To see how far we have came from the being of this project and even from the being of the semester this was definitely something that we didn't want to quit working on. Like it was mention though both of us are very proud with how the project turned out!

Results

Below are some images of the results that we were able to get with our GUI. As you can see the code was extremely successful with being able to detect the image we tested it with.


EE 424 Final Project
Tested Image
Matched Image


EE 424 Final Project
Tested Image
Matched Image


EE 424 Final Project
Tested Image
Matched Image

Code

```
"""************************************************************
                   EE 424 Final Project
                   Spring Semester 2018
                    Iowa State University

Group Members:     Joe Veal             javeal@iastate.edu
                   Travis Gillham        tgillham@iastate.edu

Professor:         Aleksandar Dogandzic  ald@iastate.edu
TA:                Anindya Das          abd149@iastate.edu

************************************************************"""

"""************************************************************
                       Project Description
For our project we decided, to try and implement a facial
recognition program using Eigenfaces and PCA. The general idea
of the project is for the most part pretty simple to follow.
Using a GUI a user will select if he/she wants to "Clock In"
or create a new user. If the user selects "Clock In" using a
webcam the users image will be taken. That image will then be
compared with all the other images the webcam has ever taken
and then if a match is found that user will be they granted
access or "Clocked In". If the user selects create  new user.
The GUI will then direct the user to a register key were the
user will enter info and then their picture will be saved to
the database. The following time the user comes he/she will
then be able to "Clock In" with the hopes that the image that
will then be taken at that time will match up with the image
that was taken when the user registered.

************************************************************"""
#IMPORTS
import pygame
import types
import os
import sys
import numpy as np
import scipy as sp
from PIL import Image
from sympy import *
from scipy import linalg as LA
#COLORS
```

```python
BLUE = (100,149,237)                     #Cornflower Blue
BLACK = (0,0,0)                   #Black
def EigenfaceCore(T):

    #average face image
    m = np.sum(T, axis = 1)
    (rows, cols) = T.shape
    m = m/cols
    print("Inside EigenfaceCore")

    #subtracting image from average
    a = np.zeros(shape=(rows,cols))
    for i in range(0,cols):
        a[:,i] = T[:,i] - m

    cov = np.transpose(a) #change matrix multiplication MxM
      #L=A'*A
    L= np.transpose(a).dot(a)

    #change matrix multiplication MxM
    covarianceLMatrix = np.zeros((a.shape[1], a.shape[1]), dtype=np.int32)
    covarianceLMatrix = np.transpose(a).dot(a)
    # Get eigenVectors of L

    [d,v] = np.linalg.eig(covarianceLMatrix) #calculate M eigenvectors
    idx = d.argsort()[::1]
    Eigen_Values = d[idx]
    Eigen_Vectors = v[:,idx]

    np.savetxt('D.txt', Eigen_Values, delimiter='    ,    ')   # X is an array
    np.savetxt('V.txt', Eigen_Vectors, delimiter='    ,    ')   # X is an array

    (vrow,vcol)=np.shape(Eigen_Vectors)

    diagonal_Matrix = np.diagflat(Eigen_Values)

    np.savetxt('Diagonal.txt', diagonal_Matrix, delimiter='    ,    ')
    L_eig_vec = []
    print("vcol:\t\t\t",vcol)
    for j in range(0,vcol):
        if (diagonal_Matrix.item(j,j) > 1):
                #L_eig_vec = np.block((L_eig_vec,v[:,j]))
                L_eig_vec.append(Eigen_Vectors[:,j])
                #L_eig_vec[:,j]=(Eigen_Vectors[:,j])
                #print("Inside EigenfaceCore Loop 2")
    Transpose_L_eig_vec = np.transpose(L_eig_vec)
    #L_eig_vec = np.array_split(L_eig_vec,20)
    #print("L_eig_vec:\t\t\t",L_eig_vec)
    np.savetxt('L_eig_vec.txt', Transpose_L_eig_vec, delimiter='   ,   ')
    np.savetxt('A.txt', a, delimiter='   ,   ')   # X is an array

    eigenfaces=a.dot(Transpose_L_eig_vec)
    np.savetxt('Eigenfaces.txt', eigenfaces, delimiter='   ,   ')   # X is an array

    print("Before Leaving EigenfaceCore")
    print("Size Of A\t\t",a.shape)
    print("Size Of D\t\t",d.shape)
    print("Size Of V\t\t",v.shape)
    print("Size Of L\t\t",L.shape)
    print("Size Of L_eig_vec\t",Transpose_L_eig_vec.shape)
    print("Size Of m\t\t",m.shape)
    print("Size Of T\t\t",T.shape)
    print("Size Of Eigenface\t\t",eigenfaces.shape)
    print("\n")

    return [m,a,eigenfaces]

def Recognition(TestImage, m, A, Eigenfaces):

        print("Inside Recognition")

        ProjectedImages = []

        (vrow,Train_Number)=np.shape(Eigenfaces)

        TransposeE = np.transpose(Eigenfaces)
        print("Size of TransposeE",TransposeE.shape)
        np.savetxt('TransposeE.txt',TransposeE)
        np.savetxt('Eigenfaces.txt',Eigenfaces)

        TransposeTemp = np.transpose(Eigenfaces)

        for i in range(0,Train_Number):
            temp= TransposeE.dot(A[:,i])
            ProjectedImages.append(temp)


        Transpose_ProjectedImages = np.transpose(ProjectedImages)

        print("Size Of Temp",temp.shape,)

        np.savetxt('ProjectedImages.txt',Transpose_ProjectedImages)

        InputImage = pygame.image.load(TestImage).convert()

        data1 = Image.open(TestImage)

        data1 = np.array(data1)

        (imgrow,imgcol,row)=data1.shape

        temp = np.zeros(shape=(imgrow,imgcol))

        for j in range(1,imgcol):
                temp[:,j]=(data1[:,j,1])

        temp[:,:] = data1[:,:,1]

        TransposeTemp = np.transpose(temp)

        (irow,icol)=np.shape(temp)

        InImage = np.reshape(TransposeTemp, irow * icol)

        Difference = (InImage) - m                    #Centered test image

        ProjectedTestImage = TransposeE.dot(Difference) #Test image feature vector

        Euc_dist = []
        tempa = []
        q= []
        for i in range(0,Train_Number):
                q.append(Transpose_ProjectedImages[:,i])
                Val=np.subtract(ProjectedTestImage,q)
                TransposeVal = np.transpose(Val)
                np.savetxt('Val.txt',TransposeVal)

                tempa.append(np.square((np.linalg.norm(TransposeVal))))

                Euc_dist.append(tempa[i])
        Euc_dist_min = min(Euc_dist)
        print("Euc_dist MIN\t\t",Euc_dist_min)

        Recognized_index = np.argmin(Euc_dist)
        print("Recognition_index \t\t",Recognized_index)
        imageValue='.jpg'
        OutputName = str(Recognized_index) + imageValue
        print (OutputName)

        print("\n")

        return OutputName

def rgb2gray(rgb):

    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]
    gray = 0.2989 * r + 0.5870 * g + 0.1140 * b

    return gray

def CreateDatabase():

        print("Inside CreateDatabase")
        Train_Numbers=0

        face1 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/1.jpg")
        data1 = np.array(face1)
        graydata1 = rgb2gray(data1)
        (drows,dcols) = graydata1.shape
        length = drows*dcols
        graydata1 = graydata1.reshape(length,1)
        Train_Numbers+=1

        face2 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/2.jpg")
        data2 = np.array(face2)
        graydata2 = rgb2gray(data2)
        (drows,dcols) = graydata2.shape
        length = drows*dcols
        graydata2 = graydata2.reshape(length,1)
        Train_Numbers+=1
```

```
	face3 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/3.jpg")
	data3 = np.array(face3)
	graydata3 = rgb2gray(data3)
	(drows,dcols) = graydata3.shape
	length = drows*dcols
	graydata3 = graydata3.reshape(length,1)
	Train_Numbers+=1

	face4 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/4.jpg")
	data4 = np.array(face4)
	graydata4 = rgb2gray(data4)
	(drows,dcols) = graydata4.shape
	length = drows*dcols
	graydata4 = graydata4.reshape(length,1)
	Train_Numbers+=1

	face5 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/5.jpg")
	data5 = np.array(face5)
	graydata5 = rgb2gray(data5)
	(drows,dcols) = graydata5.shape
	length = drows*dcols
	graydata5 = graydata5.reshape(length,1)
	Train_Numbers+=1

	face6 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/6.jpg")
	data6 = np.array(face6)
	graydata6 = rgb2gray(data6)
	(drows,dcols) = graydata6.shape
	length = drows*dcols
	graydata6 = graydata6.reshape(length,1)
	Train_Numbers+=1

	face7 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/7.jpg")
	data7 = np.array(face7)
	graydata7 = rgb2gray(data7)
	(drows,dcols) = graydata7.shape
	length = drows*dcols
	graydata7 = graydata7.reshape(length,1)
	Train_Numbers+=1

	face8 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/8.jpg")
```

```
        data8 = np.array(face8)
        graydata8 = rgb2gray(data8)
        (drows,dcols) = graydata8.shape
        length = drows*dcols
        graydata8 = graydata8.reshape(length,1)
        Train_Numbers+=1

        face9 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/9.jpg")
        data9 = np.array(face9)
        graydata9 = rgb2gray(data9)
        (drows,dcols) = graydata9.shape
        length = drows*dcols
        graydata9 = graydata9.reshape(length,1)
        Train_Numbers+=1

        face10 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/10.jpg")
        data10 = np.array(face10)
        graydata10 = rgb2gray(data10)
        (drows,dcols) = graydata10.shape
        length = drows*dcols
        graydata10 = graydata10.reshape(length,1)
        Train_Numbers+=1

        face11 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/11.jpg")
        data11 = np.array(face11)
        graydata11 = rgb2gray(data11)
        (drows,dcols) = graydata11.shape
        length = drows*dcols
        graydata11 = graydata11.reshape(length,1)
        Train_Numbers+=1

        face12 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/12.jpg")
        data12 = np.array(face12)
        graydata12 = rgb2gray(data12)
        (drows,dcols) = graydata12.shape
        length = drows*dcols
        graydata12 = graydata12.reshape(length,1)
        Train_Numbers+=1

        face13 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/13.jpg")
        data13 = np.array(face13)
        graydata13 = rgb2gray(data13)
        (drows,dcols) = graydata13.shape
        length = drows*dcols
        graydata13 = graydata13.reshape(length,1)
        Train_Numbers+=1

        face14 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/14.jpg")
        data14 = np.array(face14)
        graydata14 = rgb2gray(data14)
        (drows,dcols) = graydata14.shape
        length = drows*dcols
        graydata14 = graydata14.reshape(length,1)
        Train_Numbers+=1

        face15 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/15.jpg")
        data15 = np.array(face15)
        graydata15 = rgb2gray(data15)
        (drows,dcols) = graydata15.shape
        length = drows*dcols
        graydata15 = graydata15.reshape(length,1)
        Train_Numbers+=1

        face16 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/16.jpg")
        data16 = np.array(face16)
        graydata16 = rgb2gray(data16)
        (drows,dcols) = graydata16.shape
        length = drows*dcols
        graydata16 = graydata16.reshape(length,1)
        Train_Numbers+=1

        face17 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/17.jpg")
        data17 = np.array(face17)
        graydata17 = rgb2gray(data17)
        (drows,dcols) = graydata17.shape
        length = drows*dcols
        graydata17 = graydata17.reshape(length,1)
        Train_Numbers+=1

        face18 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/18.jpg")
        data18 = np.array(face18)
        graydata18 = rgb2gray(data18)
        (drows,dcols) = graydata18.shape
        length = drows*dcols
        graydata18 = graydata18.reshape(length,1)
        Train_Numbers+=1

        face19 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/19.jpg")
        data19 = np.array(face19)
        graydata19 = rgb2gray(data19)
        (drows,dcols) = graydata19.shape
        length = drows*dcols
        graydata19 = graydata19.reshape(length,1)
        Train_Numbers+=1

        face20 = Image.open("/Users/tgillham/Desktop/EE 424/Final Project/Eigen
Faces/PCA_based Face Recognition System/TrainDatabase/20.jpg")
        data20 = np.array(face20)
        graydata20 = rgb2gray(data20)
        (drows,dcols) = graydata20.shape
        length = drows*dcols
        graydata20 = graydata20.reshape(length,1)
        Train_Numbers+=1

        #creation of the T matrix
        T = np.zeros(shape = (length,1))

        #filling the T matrix
        T = graydata1
        T = np.hstack((T,graydata2))
        T = np.hstack((T,graydata3))
        T = np.hstack((T,graydata4))
        T = np.hstack((T,graydata5))
        T = np.hstack((T,graydata6))
        T = np.hstack((T,graydata7))
        T = np.hstack((T,graydata8))
        T = np.hstack((T,graydata9))
        T = np.hstack((T,graydata10))
        T = np.hstack((T,graydata11))
        T = np.hstack((T,graydata12))
        T = np.hstack((T,graydata13))
        T = np.hstack((T,graydata14))
        T = np.hstack((T,graydata15))
        T = np.hstack((T,graydata16))
        T = np.hstack((T,graydata17))
        T = np.hstack((T,graydata18))
        T = np.hstack((T,graydata19))
        T = np.hstack((T,graydata20))
        print("Before Leaving CreateDatabase")
        print("dcol\t\t\t",dcols)
        print("drow\t\t\t",drows)
        print("Size Of T\t\t",T.shape)
        print("Size Of Temp\t\t",graydata20.shape)
        print("Train_Numbers\t\t",Train_Numbers)

        print("\n")

        return[T,Train_Numbers]

# --------------Main Program Loop ----------------
TrainDatabasePath=os.path.abspath('/Users/tgillham/Desktop/EE 424/Final
Project/Eigen Faces/PCA_based Face Recognition System/TrainDatabase')
TestDatabasePath=os.path.abspath('/Users/tgillham/Desktop/EE 424/Final
Project/Eigen Faces/PCA_based Face Recognition System/TestDatabase')
#Extension
imageValue='.jpg'

print("\n")

#Get User Input
prompt = ('Enter test image name (a number between 1 to 10):')

#Get Path of Image Picked
TestImageValue = input("What Image? ")
type(TestImageValue)
if TestImageValue == ("new"):
        print("Hi")
        import camera

print("\n")

z=os.path.join(TestDatabasePath,TestImageValue)+imageValue
pygame.init()

#Display Screen
width=718
height=502
screen = pygame.display.set_mode((width, height ))
screen.fill(BLUE)

#FONT
font = pygame.font.SysFont('Calibri', 25, True, False)
font_Title = pygame.font.SysFont('Calibri', 50, True, False)
```

```python
img = pygame.image.load(z).convert()

[T,Train_Numbers]=CreateDatabase()

[m, A, Eigenfaces] = EigenfaceCore(T)
OutputName = Recognition(z, m, A, Eigenfaces)

SelectedImage = os.path.join(TrainDatabasePath, OutputName)

pygame.display.set_caption('Display an image')

#IMAGES
Person1 = pygame.image.load(z).convert()
Person1 = pygame.transform.scale(Person1, (234, 302))

Person2 = pygame.image.load('Person2.png').convert()
Person2 = pygame.transform.scale(Person2, (234, 302))

#GUI TITLE
Title = font_Title.render("EE 424 Final Project", True, BLACK)
screen.blit(Title, [175, 30])

#TEXT UNDER IMAGES
text_Person1 = font.render("Tested Image", True, BLACK)
text_Person1 = pygame.transform.rotate(text_Person1, 0)
screen.blit(text_Person1, [152, 405])
text_Person2 = font.render("Matched Image", True, BLACK)
text_Person2 = pygame.transform.rotate(text_Person2, 0)
screen.blit(text_Person2, [435, 405])

x = 0; # x coordnate of image
y = 0; # y coordinate of image

#ADD IMAGES TO GUI
screen.blit(Person1, ( x+100,y+100)) # paint to screen
screen.blit(Person2, ( x+384,y+100)) # paint to screen

pygame.display.flip() # paint screen one time

running = True

while (running): # loop listening for end of game
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
#loop over, quite pygame
pygame.quit()
```