**LAGCC FACIAL RECOGNITION SYSTEM**

Elija, Umer, Joe and Mikhael

MAC 125, Section 5243

Over the years there has been an increasing demand for better security systems. One method that is improving, and has gained in popularity is facial recognition technology. Facial recognition technology is computer software that can compare, identify, or verify a digital image of a face from other image sources. This application uses an algorithm which can detect and compare unique features of a person's face.

Face recognition is a versatile technology that has many applications, from security and law enforcement, to social apps and media. Many airports use face recognition for security clearance when entering the airport, verifying passports, or even as boarding passes. Cameras and smartphones implement face detection when capturing photos to adjust focus and improve picture quality. Some popular social apps that use face detection and recognition are Snapchat, and Facebook. Facebook can identify individuals in photos and suggest tagging that individual in the photo. It has even been said that Facebook's recognition software is better at identifying individuals than the government's system.

Another possible use for face recognition is in school environments, in universities, colleges, and campuses. By implementing this type of system, security and safety can be improved. The program can quickly recognize and determine who does, and does not belong in a building or specified area. In addition, the system can further identify specific groups, such as faculty, staff, or students.

However, because face recognition technology is a form of biometrics, there is a lot of debate about whether it should be used or not. There is concern of privacy issues, breach or misuse of the technology. Since this technology only uses a camera, computer, and the program, it is non-invasive to the subjects. In other words, when the application is being used, subjects may not even be aware that they are being observed and checked. Another issue is accuracy. Face recognition is not always accurate, and can be affected by many different factors.

Computer facial recognition works in a few basic steps. First, a database is supplied with a set of images. The database will serve a source for the algorithm to compare against. The next step is training, the algorithm will collect information about unique features of the subject. Finally, when the program is operating, it must detect that a face is present, and determine if a match is found within the database.

**OpenCV Library:**

In our project we implemented libraries from OpenCV. The library itself is based on for applications such as ours. Here is a part of the OpenCV about page which explains some topics about it:

"OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc."

We would use algorithms that would be able to read faces and train the program to be able to recognize test images.

**Problems faced when developing a Facial recognition program:**

When developing a facial recognition application, the complexity is found in how to make a computer understand first what a face is, then later on how to read a face, how to memorize or train a face, and finally how to compare between two different faces. Besides all of this, there are other issues that a programmer may face because facial recognition is not always accurate.

Things such as variations in lighting of an image, variation in facial expressions, angles and distance of a face from the camera, facial accessories such as glasses or even change of facial hair or hairstyles are many of the factors that change within each photo taken of a specific individual. But some of these factors can be eliminated such as the lighting variations or facial expressions of images through the use of better algorithms that give the computer the ability to compare the differences of same or different faces more properly and accurately.

**Algorithms:**

There are a lot of algorithms that are used in facial recognition such as The Trace Transform, the Independent Component Analysis (ICA), Principal Component Analysis (PCA) or the Eigenface Algorithm, Linear Discriminant Analysis (LDA) or Fisherface Algorithm, e.t.c, but in our project, we compare and contrast two algorithms; the Principal Component Analysis or Eigenface Algorithm and the Linear Discriminant Analysis or Fisherface Algorithm. In our project we have selected to use the Fisherface algorithm. The reason will be explained in the Eigenfaces vs Fisherfaces section.

In the Eigenface algorithm we use the principal components analysis to maximize the scatter of all the projected samples. The first step is to obtain the mean of all the provided pictures of a person, once we find the mean we need to find the covariance matrix of the whole set of images with respect to the mean value. The covariance matrix gives us the scatter of the data (how far is each ample image from the next one). After we obtain the covariance matrix we need to obtain the agent vectors and values of the covariant matrix.

Through these agent vectors we can create a W matrix which is a linear transform that allows us to go from spaces defined by x to the spaces defined by y. Also, we need to reduce the dimensions of the images as much as possible for better performance and for that we have to maximize the scatter of all images. We would have to take the biggest agent value which maximizes the scatter hence reducing the dimensionality of the images.

\mu = \frac{1}{n} \sum_{i=1}^{n} x_{i} y = W^{T} (x - \mu)

S = \frac{1}{n} \sum_{i=1}^{n} (x_{i} - \mu) (x_{i} - \mu)^{T}` x = W y + \mu

S v_{i} = \lambda_{i} v_{i}, i=1,2,\ldots,n

**Eigenfaces vs Fisherfaces:**

So, when we maximize the scatter between the classes in the case of the Eigenface algorithm, we are not totally improving the overall performance of the program and this is where the Eigenface falls behind the Fisherface algorithm. Indeed, we get better performance when the algorithm reduces the dimensions but during the process, the algorithm doesn’t try to minimize the scatter located within the classes. The problem with this is that for example, the difference in scatter can come from difference in lighting or illumination in the images so the algorithm will interpret samples with in a class as samples from different classes because of the variations in the illumination. Images of the same class (same person) should always be considered for the same class, not for some other. But this problem is solved by the Fisherface Algorithm.

The Fisherface Algorithm is much like the Eigenface algorithm but with some variations. The Algorithm uses the fisher's linear discriminant. This algorithm actually notes down the small variations such as illumination into a linear subspace of the image space. It follows the protocol that classes are convex and therefore, linearly separable. This algorithm also performs dimensionality reduction while retaining the linear separability. It first calculates the variance matrix of all the images with respect to their global mean which is the basic scatter between the classes. In addition, it also calculates the variance matrix which gives the scatter within the classes. Basically, it is the difference between the mean value and the image associated to a class. After that we need to obtain the agent values and vectors for both the scatter outside the class and inside the class. Through this we get two W matrixes. We have to divide the W matrix for the Scatter outside the class by the W matrix for the scatter with the class which will result in a linear transformation. This allows the program to maximize the scatter outside the class while minimizing the scatter within the classes therefore giving the program the ability to differentiate between classes and put as close as possible the images that belong to the same class. Overall Fisherface Algorithm achieves a greater between-class scatter than principal component analysis which leads to a better classification performance. This is why Fisherface is better than the Eigenface.

\begin{align*}
    X & = & \{X_1,X_2,\ldots,X_c\} \\
    X_i & = & \{x_1, x_2, \ldots, x_n\}
\end{align*} Inserting image... \mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j

\begin{align*}
    S_{B} & = & \sum_{i=1}^{c} N_{i} (\mu_i - \mu)(\mu_i - \mu)^{T} \\
    S_{W} & = & \sum_{i=1}^{c} \sum_{x_{j} \in X_{i}} (x_j - \mu_i)(x_j - \mu_i)^{T}
\end{align*} Inserting image...

**Experimental Results:**

The results from the Facial Recognition Algorithm are affected by several factors such as variations in facial expression, different angles, lighting, camera quality, resolution, facial hair, hair styles, hats, glasses and other accessories. All these factors including the implementation of Fisherface over Eigenface algorithm makes all the difference in the results we gained.

More than half of the attempts the algorithm was able to detect faces and check if it the face was recognized from the database or if it wasn't therefore classifying the detected face as an "unidentified" face. The results showed that our algorithm is sensitive and requires more sophistication to be available for commercial or academic use such as college campuses and white-collar industries. Regardless of this matter the program shows and demonstrates potential.

**Sources**

1. Understanding Facial Recognition Software

<https://www.fi.edu/understanding-facial-recognition-software>

2. How Facial Recognition Systems Work

<https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm>

3. Face Recognition using OpenCV and Python: A Beginner’s Guide

<https://www.superdatascience.com/opencv-face-recognition/>

4. Face Recognition with OpenCV

<https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#algorithmic-description>

5. OpenCV Basics

<https://www.youtube.com/playlist?list=PLAp0ZhYvW6XbEveYeefGSuLhaPlFML9gP>

6. Eigenfaces and Fisherfaces

h[ttp://disp.ee.ntu.edu.tw/~pujols/Eigenfaces%20and%20Fisherfaces.pdf](http://disp.ee.ntu.edu.tw/~pujols/Eigenfaces%20and%20Fisherfaces.pdf)

7. Dimensionality Reduction

<http://courses.media.mit.edu/2010fall/mas622j/ProblemSets/slidesPCA.pdf>

8. Facial Recognition May Boost Airport Security But Raises Privacy Worries

<https://www.npr.org/sections/alltechconsidered/2017/06/26/534131967/facial-recognition-may-boost-airport-security-but-raises-privacy-worries>

9. About OpenCV

<https://opencv.org/about.html>

10. Face Recognition using Eigenfaces, Fisherfaces and Laplacianfaces algorithms

<https://www.youtube.com/watch?v=nSrmSwE-qbU>

11. Eigenfaces

<https://www.youtube.com/watch?v=jQOZrXZTXcw>

12. Eigenfaces and Fisherfaces part 1

<https://www.youtube.com/watch?v=xoJ9OtIhnT0>

13. Installing OpenCV 3.3.0 with Visual Studio 2017 on Windows 10

<https://youtu.be/oJ6fh-XLjtg>