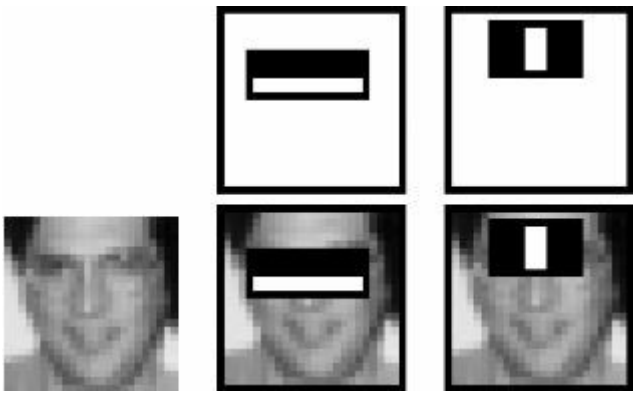
Zack McGinnis
Winter 2016
CS410: Computer Vision
Project 2
Face Detection and Recognition

Background:
This report aims to define and test different methods which accomplish the task of facial detection and recognition within images.

For facial detection within images, cascade classifiers are a popular choice for those who wish to emphasize speed in their detection. In this project, we will explore how detection is accomplished with the Haar-cascades method (also known as the Viola-Jones method). The Haar-cascades detection method is included within the OpenCV library, as well as many different pre-trained XML face classifiers which will assist in the detection process.

Facial recognition is slightly more complex. Of the many algorithms available which are used to accomplish this task, we will inspect and implement three: Eigenfaces, Fisherfaces, and LBPH (Local binary patterns histograms). Luckily, all three algorithms are available in the OpenCV library. All three methods compare the input image (unknown face) with a training set of known faces. In the training set, known faces (10) are provided for each of the 40 people in our training set. When an algorithm is supplied an unknown face, it calls upon the training set to assist in recognition. Each of our three algorithms will accomplish this task in a different manner.

Face Detection – Haar Cascades:
The Haar-cascades method for feature detection is implemented with a pre-trained XML classifier file. This saves us time as we would have had to build and train our own classifier to detect edges, lines, and other features within our images. The XML file "haarcascade_frontalface_alt2" provides our system with the necessary information needed to detect the features of a face from a frontal view. Once an input image is loaded into our program, the entire image is scanned and inspected for the positive features specified in the XML classifier.



This picture shows how an input image is scanned and inspected for specified features in accordance with our classifier.

Face Recognition - Eigenfaces:
The Eigenface approach to facial recognition relies in performing what's called Principle Component Analysis (PCA), finding directions with the greatest variance within the data. This is easily done after each image is converted to grayscale; essentially turning each image into a vector within a vector space. After a PCA is performed on these vectors, we obtain the eigenvectors which make up the basis of the vector space.

The eigenvectors we obtain are very important to us since they usually represent the most prominent features of the data in question (faces). This is especially crucial when attempting to identify an unknown face. In this situation, we can decompose the unknown face and assess which eigenvectors most closely correlate to the image in question. We are then able to compare the similar eigenvectors to our training set, and attempt to find a successful recognition.

Algorithmic implementation:

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a random vector with observations $x_i \in R^d$.

1. The mean (u) is computed for each vector

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. Covariance matrix S is calculated for each vector

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

3. The eigenvalues and eigenvectors are calculated for each vector

$$Sv_i = \lambda_i v_i, i = 1, 2, \ldots, n$$

4. The k principle components are found according to the largest calculated eigenvalues. For any vector x, its k principle components are given by:

$$y = W^T(x - \mu)$$

5. We can then reconstruct the projection by:

$$x = Wy + \mu$$

6. Recognition is then performed by projecting all training samples (and the input image in question) into the PCA subspace.
7. The nearest neighbor (eigenvalues) are found between the projected training samples and the projected input image.

Face Recognition – Fisherfaces:
Where the Eigenface method utilizes PCA to project the image space to a low dimensional subspace and maximize the total scatter across all classes (faces), the Fisherface method uses LDA (Linear Discriminant Analysis) to maximize the ratio of between-class scatter to that of within-class scatter.

The Fisherfaces method attempts to make classes cluster tightly together, while different classes are as far away as possible from each other in the lower-dimensional representation. It is important to note that since we only identified the features to distinguish between subjects, our reconstruction will not be as detailed as in the Eigenfaces method.

Algorithmic implementation:

1. For any vector X with samples taken from c classes:

$$X = \{X_1, X_2, \ldots, X_c\}$$
$$X_i = \{x_1, x_2, \ldots, x_n\}$$

2. Scatter matrices S_B and S_W are then found by:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

3. The total mean (u) is calculated by:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

4. The mean of each class (person) is calculated (u_i):

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

5. The algorithm now finds a projection W which maximizes separability between classes:

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

6. An optimized solution is found by:

$$S_B v_i = \lambda_i S_w v_i$$
$$S_W^{-1} S_B v_i = \lambda_i v_i$$

7. And then rewritten as:

$$W_{pca} = \arg\max_W |W^T S_T W|$$

$$W_{fld} = \arg\max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

Face Recognition - LBPH:

In contrast to the previous two methods, LBPH analyzes each image independently rather than inspecting the whole dataset. The LBPH method characterizes each image in the dataset locally. When an unknown input image is provided, we perform the same analysis on it and compare the result to each of the images in the dataset. By characterizing the local patterns found in specific image locations, we can successfully analyze each image based on these results.

Algorithmic implementation:

1. We can describe the LBP operator as:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

2. Where (x_c,y_c) is the central pixel with intensity i_c, and i_n representing the intensity of a neighbor pixel. With that, our sign function can be defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

3. For any given point (x_c,y_c), its neighbor's position (x_p,y_p), where p is an element of P, is found by:

$$x_p = x_c + R\cos(\frac{2\pi p}{P})$$

$$y_p = y_c - R\sin(\frac{2\pi p}{P})$$

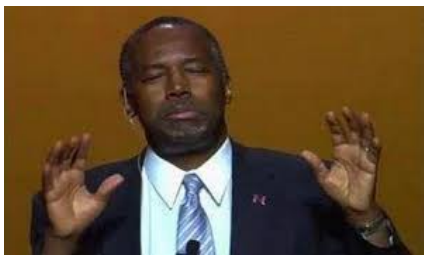
Methods:

Image Database:

Our training set is comprised of images from the AT&T Facedatabase. This database contains 40 seperate people (classes), each with 10 facial images each. This makes for total of 400 images within the database. These images contain various facial expressions, lighting, and facial accessories (glasses).

Inputs:

Experiment #1 – Facial detection: Our facial detector will be asked to detect any discovered faces within the following 3 images:



Experiment #2 – Facial recognition with Eigenfaces:

An eigenface model will be trained on the AT&T database of 399 images except for the image shown below. This image will be used as the input within the trained model, and it will attempt to successfully match it to the correct class. The image belongs to class 13. The model will calculate the first 10 eigenvalues.

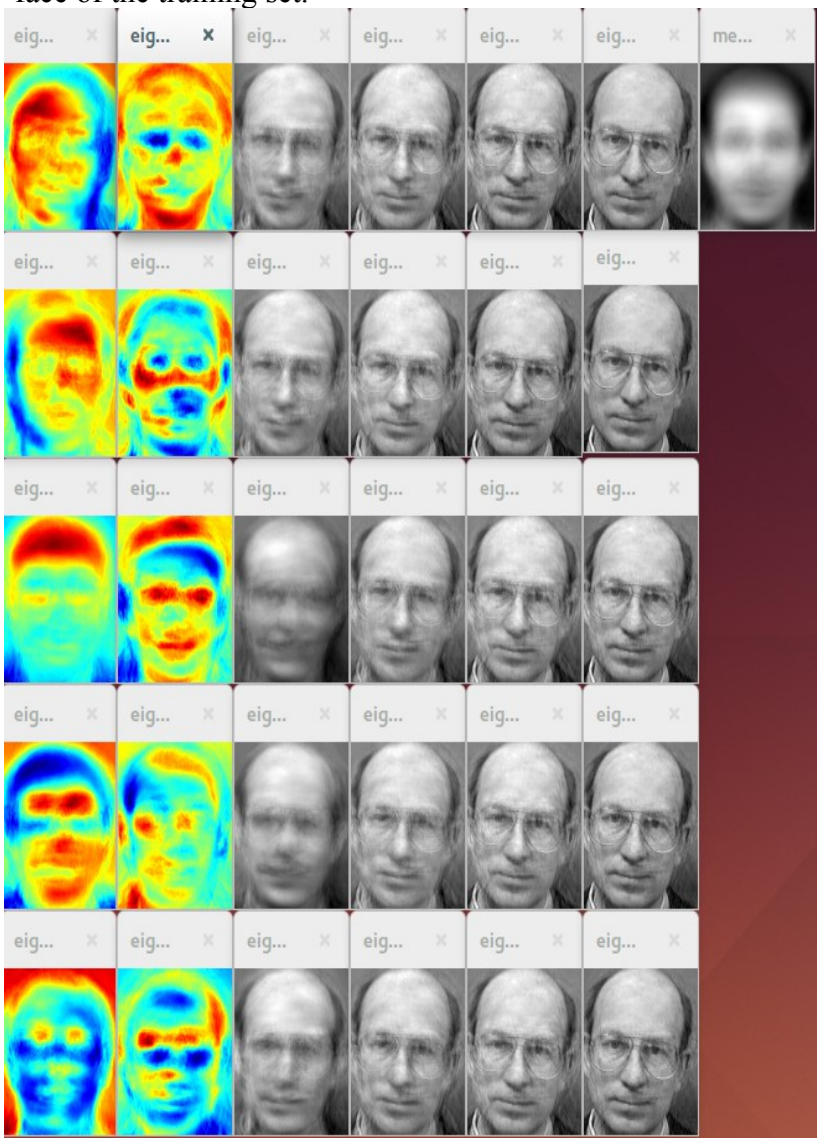
Experiment #3 – Facial recognition with Fisherfaces:
A fisherface model will be trained on the AT&T database of 399 images except for the image shown below. This image will be used as the input within the trained model, and it will attempt to successfully match it to the correct class. The image belongs to class 13. The model will calculate the first 16 eigenvalues.

Experiment #4 – Facial recognition with LBPH:
An eigenface model will be trained on the AT&T database of 399 images except for the image shown below. This image will be used as the input within the trained model, and it will attempt to successfully match it to the correct class. The image belongs to class 13. The method will use a radius of 1, and a neighborhood of 8 neighbors.

Results:

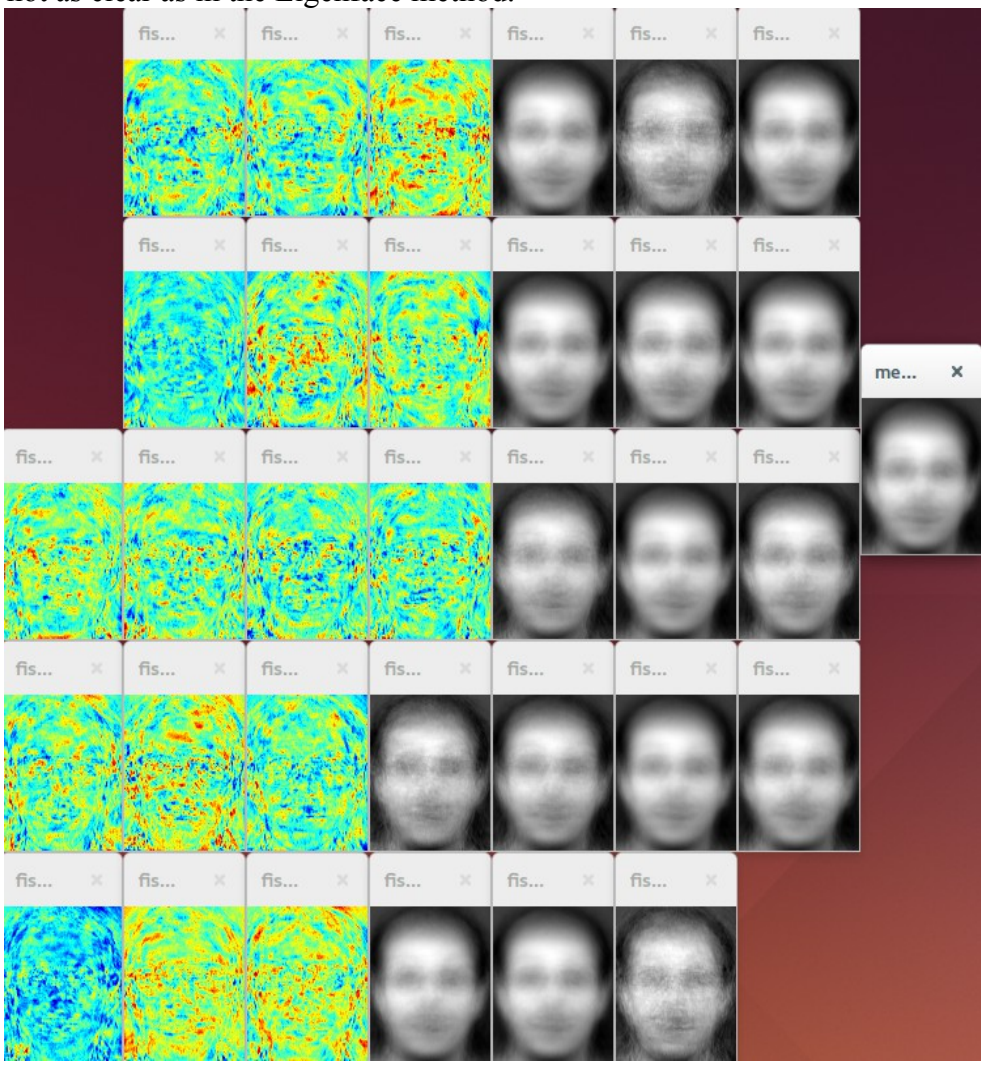
Experiment #1 – Facial detection:

Experiment #2 – Facial recognition with Eigenfaces:
Note: The facial images towards the left are the "eigenfaces" showing the first 10 principle components in the input image. The image to the upper right is the mean face of the training set.



```
Predicted class = 37 / Actual class = 37.
Eigenvalue #0 = 2817234.89109
Eigenvalue #1 = 2065223.71308
Eigenvalue #2 = 1096613.63515
Eigenvalue #3 = 888103.94982
Eigenvalue #4 = 818941.86977
Eigenvalue #5 = 538914.47401
Eigenvalue #6 = 392433.54243
Eigenvalue #7 = 373805.54654
Eigenvalue #8 = 313921.17233
Eigenvalue #9 = 288902.01563
```

Experiment #3 – Facial recognition with Fisherfaces:
Note: The facial images towards the left are the "eigenfaces" showing the first 16 principle components in the input image. The image to the far right is the mean face of the training set. The reconstruction we see of the decomposed input image is not as clear as in the Eigenface method.




```
Predicted class = 37 / Actual class = 37.
Eigenvalue #0 = 473545.43068
Eigenvalue #1 = 10574.09452
Eigenvalue #2 = 1994.04279
Eigenvalue #3 = 1078.95203
Eigenvalue #4 = 722.94469
Eigenvalue #5 = 505.26909
Eigenvalue #6 = 240.57503
Eigenvalue #7 = 200.69655
Eigenvalue #8 = 168.09561
Eigenvalue #9 = 116.97400
Eigenvalue #10 = 83.17700
Eigenvalue #11 = 66.72139
Eigenvalue #12 = 46.44079
Eigenvalue #13 = 44.42637
Eigenvalue #14 = 36.04184
Eigenvalue #15 = 32.85056
```

Experiment #4 – Facial recognition with LBPH:
Note: Our output given tells us our specifications for our neighborhood, as well as the size of the calculated histograms.

```
Predicted class = 37 / Actual class = 37.
Predicted class = -1
Model Information:
        LBPH(radius=1, neighbors=8, grid_x=8, grid_y=8, threshold=0.00)
Size of the histograms: 16384
```

Conclusions:

From the data, we can see some very interesting things. First, and most importantly, all 3 of our recognition models predicted an incorrect class for our input image. Our input image belonged to class 13, and all of our models chose class 37. This tells us that there must be some error in the way that the algorithms classify the training set images. I can only deduce that the error is due to a mislabelling of the classes within the database, or a faulty recognition of the labels by the models.

Additionally, we see some major differences in the reconstruction of the input image between the eigenface and fisherface methods. The eigenface method, using PCA, with its goal of maximizing scatter across all classes, is able to provide a relatively detailed reconstruction of the decomposed input image. The fisherface method, using LDA instead of PCA, aims to maximize between class scatter rather than total scatter. Because of this, reconstruction of the decomposed input image will not provide any helpful information to us.

Some improvements could have been made in my methods and experiments. In the LBPH model, a better representation of the histograms should have been added to better visualize what was occuring during recognition. It is helpful having the size of the histograms, which was included, but having a graph would have been more descriptive for the user.

References:

1. http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
2. http://www.face-rec.org/algorithms/PCA/jcn.pdf
3. http://www.cs.columbia.edu/~belhumeur/journal/fisherface-pami97.pdf