Facial Restoration

Using Facial Recognition

Christopher McFee

25 May 2016

**Abstract**

Face recognition seems to be a simple task for most humans. Experiments have shown that even a one to three day old baby is able to distinguish between known faces. But how hard is it for a computer? It turns out we know little about how humans recognize faces. Are inner features (eyes, nose, mouth) or outer features (head shape, hairline) used to successfully recognize the faces of our peers and family? How do we as humans analyze an image and how does our brain decode this? It has been shown by the neurophysiologists David Hubel and Torsten Wiesel that our brain has specialized nerve cells that respond to local features of a face or object. We now know as humans that our brains use “local features” such as lines, edges, angles, and facial movement to recognize faces.

In Hubel and Wiesel’s study, they deduce that since we do not see the world as scattered pieces, our visual cortex must somehow combine the different sources of information into useful patterns. Therefore, it would make sense that computerized automatic facial recognition is all about extracting those meaningful local features from an image, putting them into a useful representation and performing some kind of classification on them to differentiate.

Local features, as mentioned previously, include lines, edges, angles, and movement. We know from our previous studies in the field of image processing how such things as edge and corner detection work and thus can apply our knowledge of those aspects of image processing to facial recognition.

Face recognition based on the geometric features of a face is probably the most intuitive approach to face recognition. One of the first automated face recognition systems used marker points (position of eyes, ears, nose, etc.) to build a feature vector using such things as the distance between the points, points and the angle between them). The recognition was performed by calculating the Euclidean distance between feature vectors of a probe and a reference image. Such a method is robust against changes in illumination by its nature, but has a huge drawback: the accurate registration of the marker points is complicated, even with state of the art algorithms. A 22-dimensional feature vector was used and experiments on large datasets have shown that geometrical features alone may not carry enough information for face recognition.

So, this project does not use a method like this. Instead, I used a model to train with hundreds of images. The model took in all the various features to reconstruct one of the subject faces. For understanding the math behind this, a good understanding of linear algebra and statistics is required.

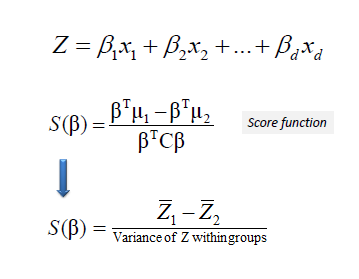
The Eigenfaces method takes a holistic approach to face recognition. See, a facial image is a point from a high-dimensional image space and a lower-dimensional representation must be found, where classification is actually possible. The lower-dimensional subspace is found with Principal Component Analysis, which identifies the axes with maximum variance.

While this kind of transformation is optimal from a reconstruction standpoint, it does not take any class labels into account. Imagine a situation where the variance is generated from external sources, let it be light or other sources. In most cases, the external source I’m talking about here is light. The axes with maximum variance do not necessarily contain any discriminative information at all; hence, a classification becomes impossible. So, a classic-specific projection with Linear Discriminant Analysis was applied. Hence, the other algorithm, Fisherfaces becomes more useful when the variance in the images is higher. In Fisherfaces, we use Linear Discriminant Analysis exclusively.

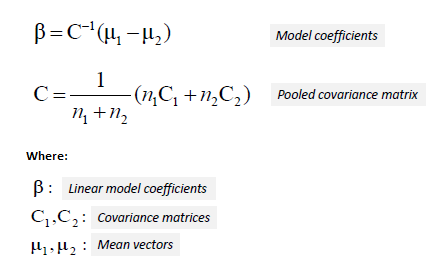
Principal Component Analysis finds the axes with the most variation, but in the presence of light this method is not useful. Linear Discriminant Analysis, which is used in facial recognition, as well as Fisherfaces is a more useful method. Linear Discriminant Analysis (LDA), is the generalization of Ronald Fisher’s linear discriminant, and is used to search for linear combinations of variables which best explain the data.

LDA is a generalized algorithm based on the famous mathematician/statistician Sir Ronald Fisher’s work (the one my program uses is slightly different, but you should get the idea).

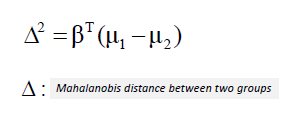
LDA is based upon the concept of searching for a linear combination of variables (predictors) that best separates two classes (targets). To capture the notion of separability, Fisher defined the following score function:



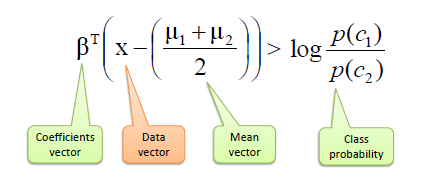
Given the score function, the problem is to estimate the linear coefficients that maximize the score which can be solved by the following equations.



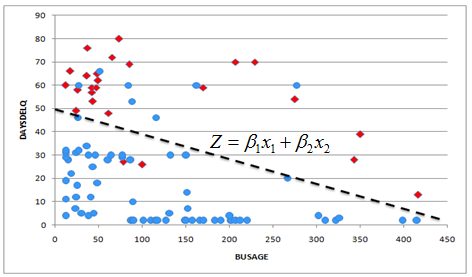
One way of assessing the effectiveness of the discrimination is to calculate the **Mahalanobis distance** between two groups. A distance greater than 3 means that in two averages differ by more than 3 standard deviations. It means that the overlap (probability of misclassification) is quite small.



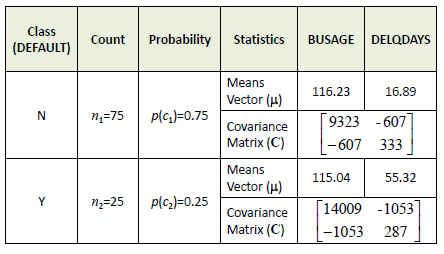
Finally, a new point is classified by projecting it onto the maximally separating direction and classifying it as *C1* if:



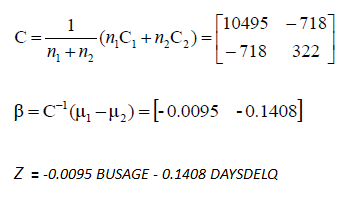
The following is NOT an Image Processing Example (This is a good example because it is far simpler than others and very easy to understand): Suppose we received a [dataset](http://www.saedsayad.com/datasets/credit_scoring_lda.xlsx) from a bank regarding its small business clients who defaulted (red square) and those that did not (blue circle) separated by delinquent days (DAYSDELQ) and number of months in business (BUSAGE). We use LDA to find an optimal linear model that best separates two classes (default and non-default).



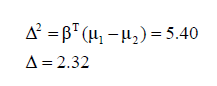
The first step is to calculate the mean (average) vectors, covariance matrices and class probabilities.



Then, we calculate pooled covariance matrix and finally the coefficients of the linear model.



A *Mahalanobis distance* of 2.32 shows a small overlap between two groups which means a good separation between classes by the linear model.



Predictors Contribution: A simple linear correlation between the model scores and predictors can be used to test which predictors contribute significantly to the discriminant function. Correlation varies from -1 to 1, with -1 and 1 meaning the highest contribution but in different directions and 0 means no contribution at all.

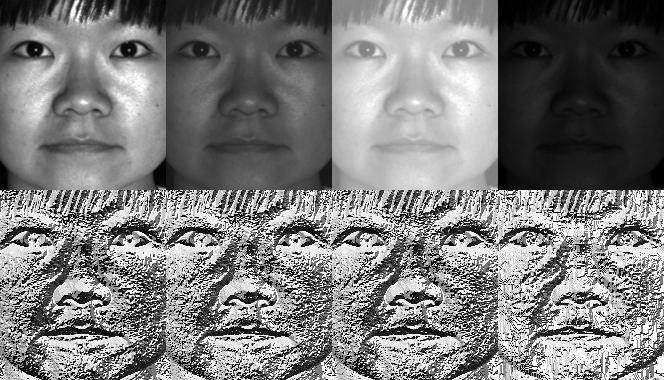
Fisherfaces are defined using class-specific dimensionality reduction. Eigenfaces do not get this type of classification. The notion is basic, that the same classes should cluster close together. Conversely different classes are spread as far apart as possible in the lower-dimensional representation. For example, with 400 pictures sized 100 times 100 pixels, the principal component analysis solves the covariance matrix S equals uppercased x uppercased x to the y power, where the size x equals ten thousand times four hundred. The resulting matrix would be a 10,000 times 10,000 medium. The resulting Eigenvectors are orthogonal. To achieve orthogonal Eigenvectors, they must be normalized to unit length. Principal Component Analysis switches a set of possibly correlated variables into a smaller set of uncorrelated variables.

Recently various methods for a local feature extraction have emerged. To avoid the high-dimensionality of the input data, only local regions of an image are described. The extracted features are (hopefully) more robust against partial occlusion, illumination and small sample size. Algorithms used for a local feature extraction are Gabor Wavelets, Discrete Cosinus Transform, and Local Binary Patterns. It is still an open research question what the best way would be to preserve spatial information when applying a local feature extraction, because spatial information is potentially useful information.

Linear Binary Patterns Histograms takes an entirely different approach to facial recognition. The idea is to not look at the whole image as a high-dimensional vector, but describe only local features of an object. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. You end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels you’ll end up with 2^8 possible combinations, called Local Binary Patterns or sometimes referred to as LBP codes. The calculations are done in clockwise. The first LBP operator described in literature actually used a fixed 3 x 3 neighborhood just like this:



You'll end up getting picture results like this (taken from Yale database image; not from this project):



That’s the math behind the three algorithms. Now, I can talk about the programming. OpenCV has something called FaceRecognizer class (this is in the contrib module; it’s not a standard part of the typical OpenCV installation; you need to compile OpenCV with the contrib modules loaded when you are running the ./configure script when building OpenCV from source. More info on doing this is available on the OpenCV installation documentation).

FaceRecognizer is a template of the base class BasicFaceRecognizer. There are different FaceRecognizer classes that inherit from the base class each with their own methods. Some are shared with others. The ones we use are EigenFaceRecognizer() and FisherFaceRecognizer()and LBPHFaceRecognizer().

The file being read by the program is a CSV file. CSV is a platform-independent (no extra software required) method of storing data. It is a plain text file which is formatted in a specific way. Because there are so many images in these data sets, I had to write a script to help generate the csv file for each. The script is written in Python and does not require any 3rd party python modules! The CSV files are far too long to write out by hand. I am including this Python script as well as the CSV files I've generated in the source code I am submitting.

Each function first determines the output directory of the images it will create, as well as the input csv file which contains the directory structure. It assigns labels to each subject’s images in the data set, stores them, and then trains the model with each image and label. It then makes predictions based on the ones in the math and outputs them to screen. It outputs to the screen the data for one of the subjects. The LBPH one does not get enough information to write reconstructed images but it still makes predictions and prints those to the screen.

Results: The results vary with each algorithm and data set and displaying them in this report would be far too lengthy, so I will simply include one cherry-picked set of results from using the Eigenfaces method on the Sheffield database. In general, I've had more success with the AT&T datbase than the Sheffield one. However, the eigenfaces function in my program works almost perfectly with any data set given. It has, by far, the highest success rate.

First, I will include the program’s output:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to my Final Project on Facial Reconstruction!

All code written by Christopher McFee

This source code uses many functions from OpenCV documentation on BasicFaceRecognizer

Please see project sources for more information

http://docs.opencv.org/ref/master/dd/d65/classcv\_1\_1face\_1\_1FaceRecognizer.html

http://docs.opencv.org/ref/master/dc/dd7/classcv\_1\_1face\_1\_1BasicFaceRecognizer.html

Related Face Recognizers Are Also included in the OpenCV3 documentation.

FaceRecognizer and BasicFaceRecognizer have changed quite a bit from OpenCV2.

Thus, the old contrib library from OpenCV2 is incompatible with this code.

Please see the OpenCV facerec.hpp file for more information

This code makes heavy use of the OpenCV contrib libraries.

I'd love to thank the original authors and contributors for all of their hard work!

This project would not be possible without them!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This is a facial recognition program that uses

images from two databases, the AT&T face database and the Sheffield University database.

This program uses a basic text menu interface with iostream from the STL.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Facial Recognition can use various algorithms.

Select one. (0: Short Explanation of Each)

1: Eigenfaces

2: Fisherfaces

3: Local Binary Patterns Histograms

(Use numkeys to select algorithm)

Selection: 1

Eigenfaces

...

All set......

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

First, enter the directory where you wish to save the written image files.

By default, the directory is simply the current directory the program is being run in.

Would you like to specify a custom directory? (Y\N):

n

Ok. Using default directory (current one)

Enter file location for csv file containing face db

Example: 'att\_faces/at.txt'

sheffieldcropped/sheffieldcropped.txt

Checking for errors...

No errors detected.

Running through calculations...

Predicted class = 39 / Actual class = 39.

Eigenvalue #0 = 4303631.17766

Eigenvalue #1 = 2197433.47780

Eigenvalue #2 = 1527323.52275

Eigenvalue #3 = 1155134.36964

Eigenvalue #4 = 736316.67188

Eigenvalue #5 = 642592.08244

Eigenvalue #6 = 612650.64471

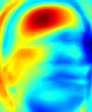
Eigenvalue #7 = 483378.46038

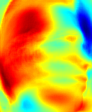
Eigenvalue #8 = 428136.29646

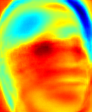
Eigenvalue #9 = 388084.28571

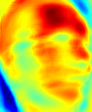
Now, here is the output of the images that were written to the disk:

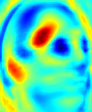
Here are the eigenfaces:

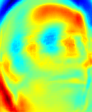


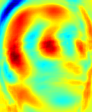


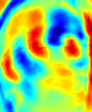


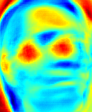


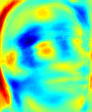












Here are the reconstructed images in order (least constructed to most constructed):









































In conclusion, I set out to have a program recognize faces using these three algorithms and then use the data it learned to redraw the faces. It turns out that LBPH does not gather enough information to do this, but Eigenfaces and Fisherfaces do. If I had time, I would also allow the user to pass thresholds to the functions through std::cin. Right now, the only way to set custom thresholds is to edit the source code directly. I did all the work myself but had some great help from reading the OpenCV documentation relating to the FacialRecognizer class. If it was not for the amazing documentation by the OpenCV team, I would certainly not be able to handle such a complex task on my own in just three weeks. It would be nice to have more time to complete this assignment, as I would have added a lot of very nice extra content as well as a GUI. Specifically, I would have made a file chooser for choosing the face data sets.

**Works Cited and Resources Used:**

First, information from the following weblinks:

* <http://www.bytefish.de/blog/pca_lda_with_gnu_octave/>
* http://people.revoledu.com/kardi/tutorial/LDA/
* <http://www.saedsayad.com/lda.htm>
* <http://sebastianraschka.com/Articles/2014_python_lda.html>
* <http://stat.psu.edu/~jiali/course/stat597e/notes2/lda.pdf>
* <https://www.youtube.com/watch?v=moqPyJQHR_s>
* <https://www.youtube.com/watch?v=t-V6y5jwcsA>
* http://www.ics.uci.edu/~welling/classnotes/papers\_class/Fisher-LDA.pdf
* <https://en.wikipedia.org/wiki/Linear_discriminant_analysis>
* https://github.com/bytefish/facerec
* http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec\_api.html#FaceRecognizer : public Algorithm
* <http://docs.opencv.org/ref/master/dc/dd7/classcv_1_1face_1_1BasicFaceRecognizer.html>
* <https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>
* <https://www.sheffield.ac.uk/eee/research/iel/research/face>

**Academic Resources**

|  |  |
| --- | --- |
|  | Ahonen, T., Hadid, A., and Pietikainen, M. Face Recognition with Local Binary Patterns. Computer Vision - ECCV 2004 (2004), 469–481. |

|  |  |
| --- | --- |
|  | Belhumeur, P. N., Hespanha, J., and Kriegman, D. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 7 (1997), 711–720. |

|  |  |
| --- | --- |
|  | Brunelli, R., Poggio, T. Face Recognition through Geometrical Features. European Conference on Computer Vision (ECCV) 1992, S. 792–800. |

|  |  |
| --- | --- |
|  | Duda, Richard O. and Hart, Peter E. and Stork, David G., Pattern Classification (2nd Edition) 2001. |

|  |  |
| --- | --- |
|  | Fisher, R. A. The use of multiple measurements in taxonomic problems. Annals Eugen. 7 (1936), 179–188. |

|  |  |
| --- | --- |
|  | Georghiades, A.S. and Belhumeur, P.N. and Kriegman, D.J., From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 6 (2001), 643-660. |

|  |  |
| --- | --- |
|  | Kanade, T. Picture processing system by computer complex and recognition of human faces. PhD thesis, Kyoto University, November 1973 |

|  |  |
| --- | --- |
|  | Martinez, A and Kak, A. PCA versus LDA IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No.2, pp. 228-233, 2001. |

|  |  |
| --- | --- |
|  | Lee, K., Ho, J., Kriegman, D. Acquiring Linear Subspaces for Face Recognition under Variable Lighting. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27 (2005), Nr. 5 |

|  |  |
| --- | --- |
|  | Messer, K. et al. Performance Characterisation of Face Recognition Algorithms and Their Sensitivity to Severe Illumination Changes. In: In: ICB, 2006, S. 1–11. |

|  |  |
| --- | --- |
|  | Raudys and A.K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioneers. - IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 3 (1991), 252-264. |

|  |  |
| --- | --- |
|  | Tan, X., and Triggs, B. Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE Transactions on Image Processing 19 (2010), 1635–1650. |

|  |  |
| --- | --- |
|  | Turk, M., and Pentland, A. Eigenfaces for recognition. Journal of Cognitive Neuroscience 3 (1991), 71–86. |

|  |  |
| --- | --- |
|  | Chiara Turati, Viola Macchi Cassia, F. S., and Leo, I. Newborns face recognition: Role of inner and outer facial features. Child Development 77, 2 (2006), 297–311. |

|  |  |
| --- | --- |
|  | Wiskott, L., Fellous, J., Krüger, N., Malsburg, C. Face Recognition By Elastic Bunch Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (1997), S. 775–779 |

|  |  |
| --- | --- |
|  | Zhao, W., Chellappa, R., Phillips, P., and Rosenfeld, A. Face recognition: A literature survey. ACM Computing Surveys (CSUR) 35, 4 (2003), 399–458. |

Appendix