# CS428 (Advanced Operating Systems) / MA500 (Geometric Foundations of Data Analysis – Classical Techniques) – Assignment 2

*Name: Taidgh Murray*

*Student ID: 15315901*

## *Principle Component Analysis*

***"Import the images as vectors and perform principle component analysis"***

I have downloaded & imported the orl_faces database into Python, and have implemented the following PCA algorithm into Python:

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

1.) Compute the Mean

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

2.) Compute the Covariance Matrix

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

3.) Compute the eigenvalues & eigenvectors of $S$

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

4.) Order the eigenvectors descending by their eigenvalue. The $k$ principal components are the eigenvectors corresponding to the $k$ largest eigenvalues

The $k$ principal components of the observed vector $x$ are then given by:

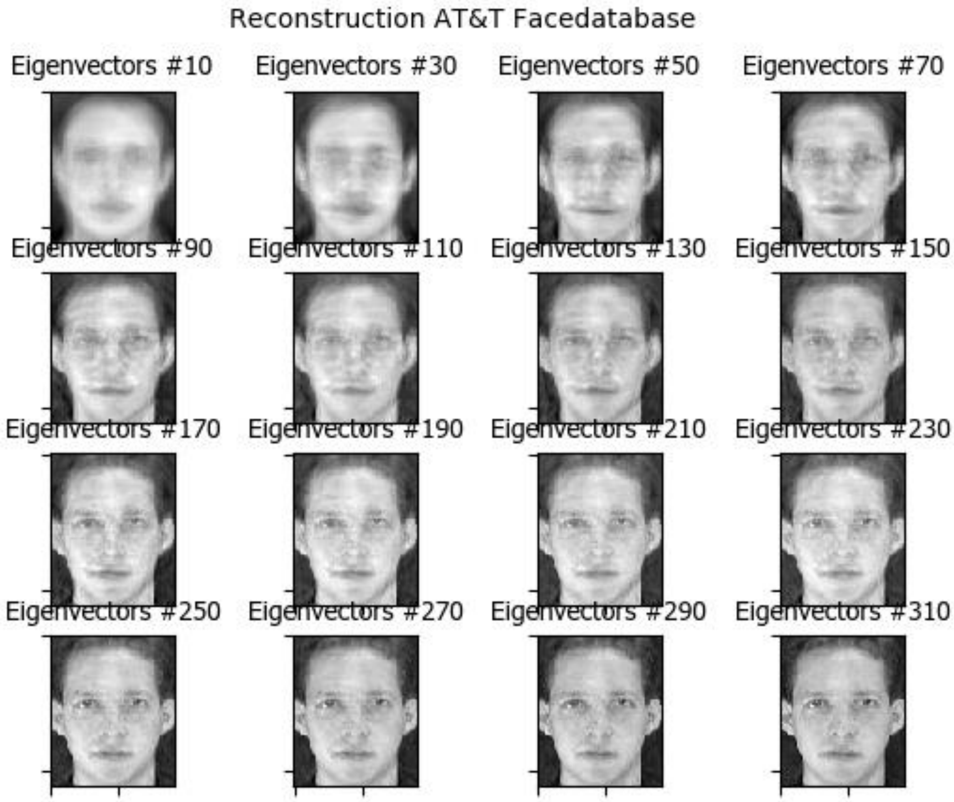
$$y = W^T(x - \mu)$$

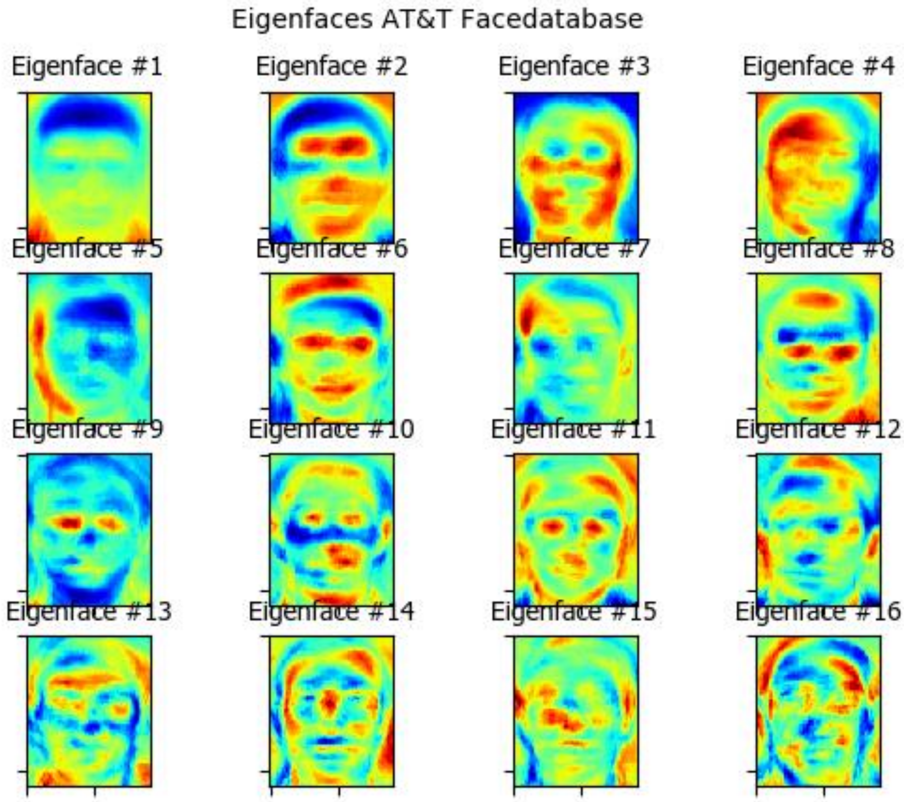
Where $W = (v_1, v_2, \ldots, v_k)$. The reconstruction from the PCA basis is given by:

$$x = Wy + \mu$$

***"For n = 10,50,100 and 300 determine how much of the variability of the database is captured by projecting onto P(n)"***

Below you can find an image output for the eigenvalues. The eigenvalues are in a range from 10 to 310, incrementing in 20 every step. Without calculating the variance for each n=10,50,100,300, we can see that the variance becomes almost 0 by n=300.


Reconstruction AT&T Facedatabase
Eigenvectors #10
Eigenvectors #30
Eigenvectors #50
Eigenvectors #70
Eigenvectors #90
Eigenvectors #110
Eigenvectors #130
Eigenvectors #150
Eigenvectors #170
Eigenvectors #190
Eigenvectors #210
Eigenvectors #230
Eigenvectors #250
Eigenvectors #270
Eigenvectors #290
Eigenvectors #310

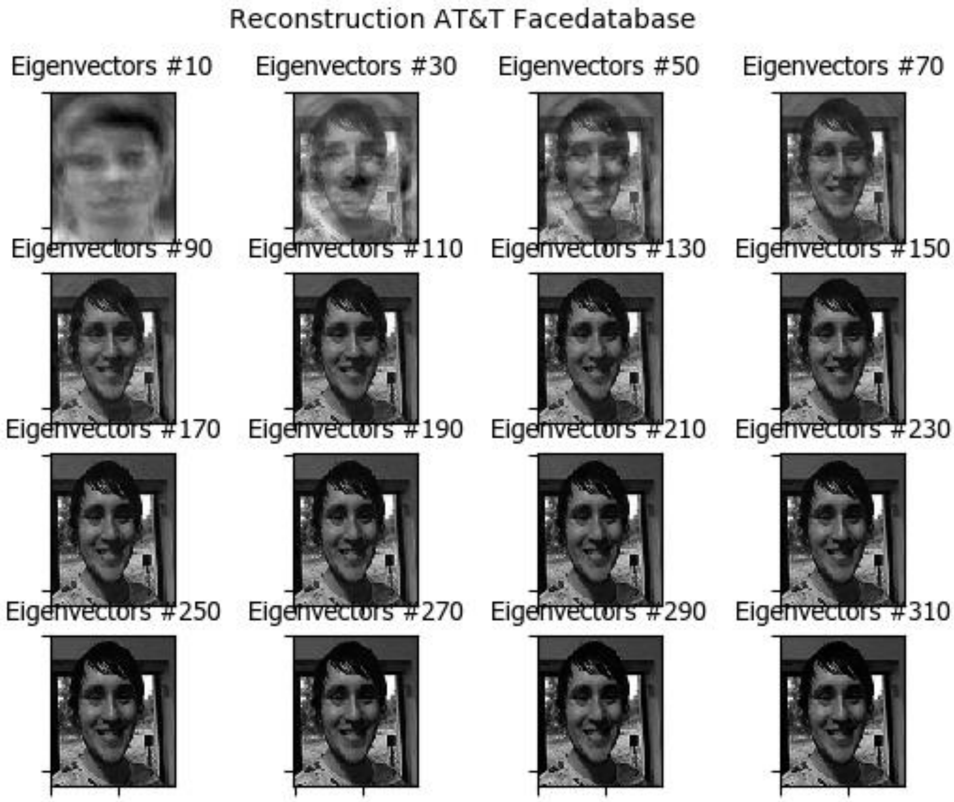

Eigenfaces AT&T Facedatabase
Eigenface #1
Eigenface #2
Eigenface #3
Eigenface #4
Eigenface #5
Eigenface #6
Eigenface #7
Eigenface #8
Eigenface #9
Eigenface #10
Eigenface #11
Eigenface #12
Eigenface #13
Eigenface #14
Eigenface #15
Eigenface #16

***"Take an image of yourself and store it in the same format as the AT&T images. Display,***

***as an image (rather than a vector), the projection of your original image onto P(n) for***

***n = 10,50,100 and 300"***

Below is the image I am using. I have converted it into a .PGM file, the same type used in the database. I then projected it into the Reconstruction database. Likely due to the image being very different to the database, the projection focuses in on my image very quickly. One would imagine that if I had access to the same conditions as the database photos, the results would be much more in line with previous assignments. I also only included a single photo, whereas the database had 10 photos per entry.



***"Take an image of a friend and determine the distance between the projections of your own image and your friend's image onto P(300). Specify which metric you are using to compute this distance"***

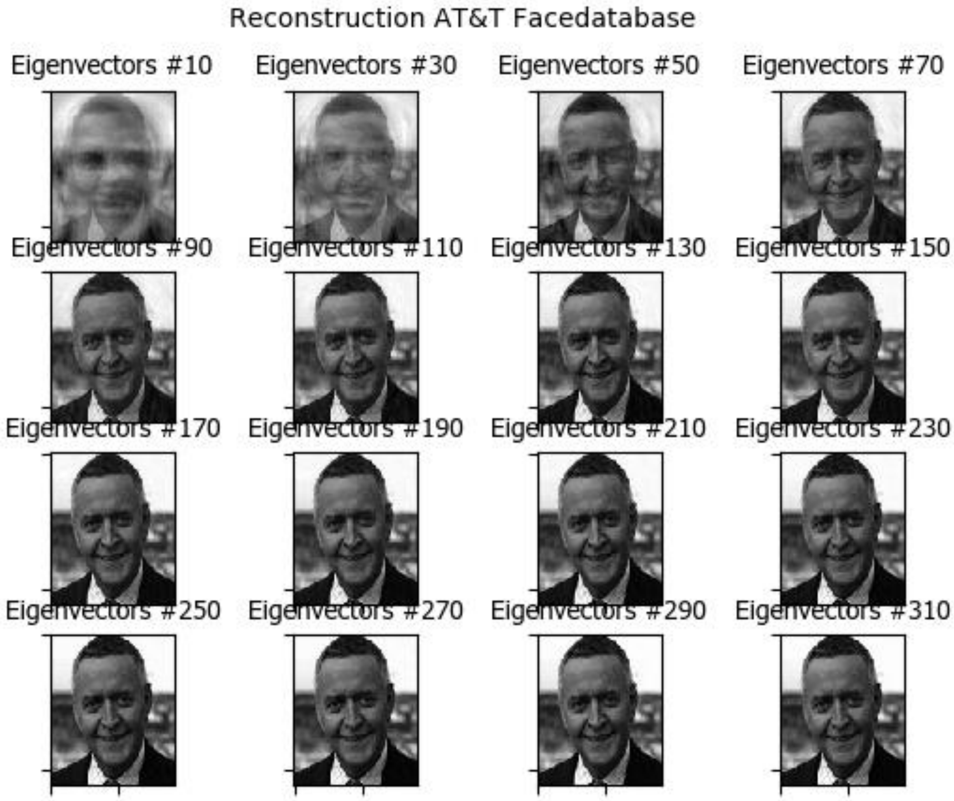
I decided to pick a random image online, as I don't have the time to get a photo of a friend. I found a random 'Frank' on twitter, and performed the same conversion as above.

The metric I'm using is a 1-Nearest Neighbour, using the Euclidean Distance metric, which is:

$$Distance((x, y), (a, b)) = \sqrt{(x-a)^2 + (y-b)^2}$$

Expected = 4 / Predicted = 4

The calculations resulting in the above value is largely abstracted behind the code. Principal Component Analysis was performed on the two images at n=300, and this was the resulting answer was 4.


Reconstruction AT&T Facedatabase
Eigenvectors #10
Eigenvectors #30
Eigenvectors #50
Eigenvectors #70
Eigenvectors #90
Eigenvectors #110
Eigenvectors #130
Eigenvectors #150
Eigenvectors #170
Eigenvectors #190
Eigenvectors #210
Eigenvectors #230
Eigenvectors #250
Eigenvectors #270
Eigenvectors #290
Eigenvectors #310