Face Recognition using PCA

Applied Mathematics Homework

Report by:
Savinien Bonheur and Albert Clerigues

January 6, 2016

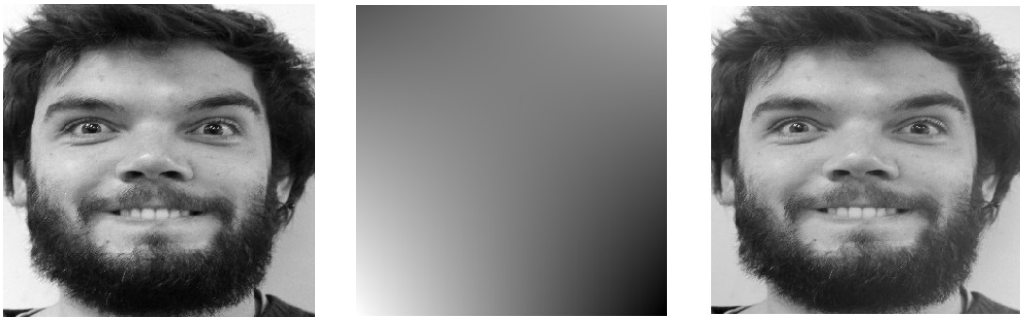
1 Data Normalization

The first step to classify faces based on pixel information is to make sure the training samples and the ones to classify are as similar and standard as possible. For this matter we will make two normalization steps:

1. Light correction, to compensate for non-uniform lightning of pixels in the image and between the samples.
2. Feature mapping, to make sure the pixel data is aligned and corresponds to each feature.

1.1 Light Correction

To perform light correction we make a simplified light intensity image based on approximating a linear model to fit the original image light intensity. Once we have this simplified model we subtract it from the original model to get a light corrected image. The last step is to normalize the contrast range to the [0-255] uint8 range and round.



(a) Original (b) Fitted model (c) Result

Figure 1: Example of application of light correction

It can be seen that the right of the original face was in shadow and darker than the other, after the correction both sides of the face have approximately the same average intensity.

1.2 Normalized Feature Mapping

In order to have the same physical area of the face coincide with each feature pixel in all images we have to transform the image. For this matter we will select key face features and approximate a transform that maps the original image face with the standard face model.

The output image must be of dimension 64x64 and have the facial features mapped according to the reference face:

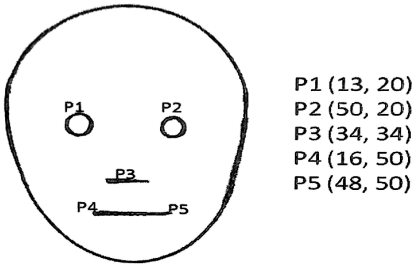


Figure 2: Location of the five facial features in the reference face

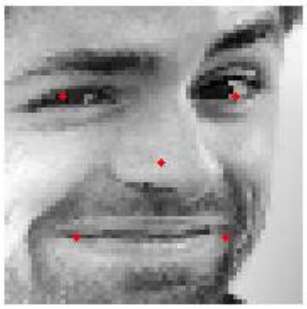
The process of finding such transformation will be implemented using an iterative algorithm that uses SVD to find the Least Linear Squares approximation for the transform. It is not possible to find it by directly solving the linear system since it is overdetermined, having 10 equations and 6 unknowns.

(a) Original Nayeem

(b) Original Dani

(c) Original Hassan

(d) Transformed Nayee



(e) Transformed Dani

(f) Transformed Hassan

Figure 3: Example of feature mapping, standard feature locations are marked in red in the transformed images

2 Face Recognition

2.1 Dimensionality Reduction with PCA

The normalized images contain 4096 pixels which corresponds to 4096 features to compare for the Nearest Neighbour search. We can simplify this problem with PCA reducing the number of features while being less sensitive to noise. By projecting the image into a meaningful base we can get rid of the less distinctive components and use the coefficients that convey more distinctive information for matching the faces.

For this, the images are projected into a base computed with the training data using PCA with *getPCABase(TrainImages)*. The test and train images are projected into the same base and the resulting coefficients are stored.

The ideal thing to do would be projecting only the first c components we needed (for face-recognition $c \approx 100$) to save memory later. However, the dimensionality reduction is performed at the time of classification, since we value more being able to change the number of components for classification at any given time than saving some memory by projecting only on the desired number of components. For a final application based on this concept, a number of components c would be fixed and the images projected on the first c eigenvectors of the base before storing.

3 The Implementation

The code provided is divided in three main sections being database loading and handling, GUI, and utility functions.

Database loading and handling includes the next functions:

- `loadDataFolder.m` - Loads the images and features from a given directory.
- `prepareTrainAndTest.m` - Normalizes the Train and Test data sets and gets the PCA projection of both.

The program GUI is implemented and launched in:

- `a_launchPCFace.m` - Loads the Train and Test data sets and launches the GUI.
- `GUI.m` - Handles all the GUI operation.

Finally, the functionality of the face recognition is implemented with:

- `applyImageTransform.m` - Applies the given transform to the image through inverse mapping.
- `computeLMStransform.m` - Computes the approximated transform to map the feature vector to the standard face.
- `light_correction.m` - Applies the light correction algorithm to the given image.
- `normalizeImages.m` - Performs all the previous steps to a set of images and outputs the normalized versions.
- `getPCABase.m` - Obtains the PCA base of eigenvectors for a given set of images.
- `projectPCABase.m` - Projects a set of images into the given base and returns the coefficients
- `matchNN.m` - Performs Nearest Neighbour classification based on the given Training set and the Test image.
- `recogniseFace.m` - Selects only the first k components of the given data sets and uses matchNN to get the nearest neighbour.

3.1 User guide

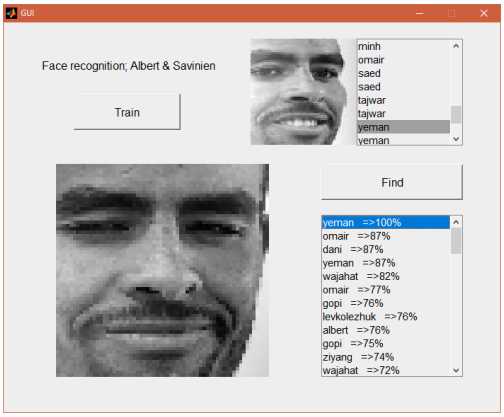
To run the program:

1. Run the script "a_launchPCFace.m" which will load the data sets and launch the GUI.
2. In the GUI, press the button named "Train" to execute the normalization and projection of the loaded data sets.
3. Scroll the top right list of test images and select one which you would like to recognise.

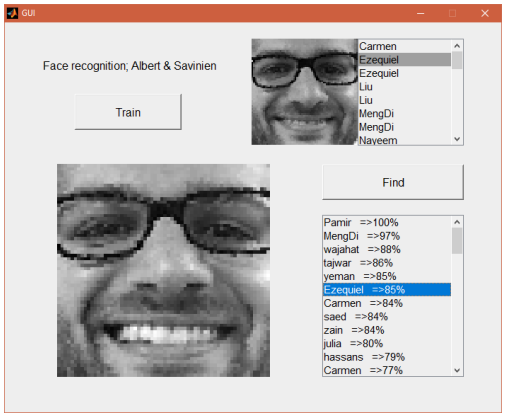
   After that press the "Find" button below it. The lower right list should update with a list of the names of the most probable names along with the certainty score percentage (relative to the smallest and biggest distances).

4 Results and Conclusions

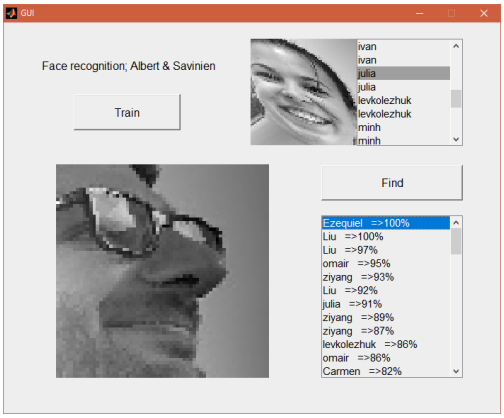
4.1 Results



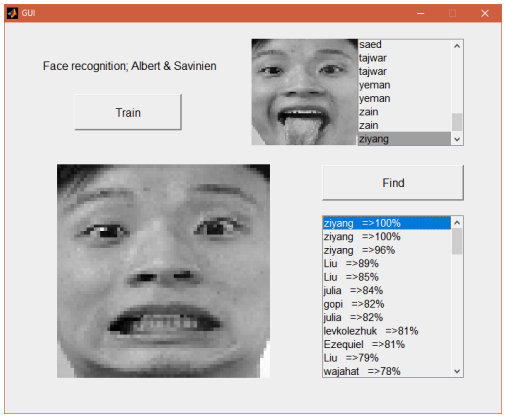
(a) Yeman 1st match



(b) Eze 5th match



(c) Julia bad features



(d) Ziyang 1st match

Figure 4: Example of different classifications with the developed program.

The program successfully recognises a big percentage of faces in the test data set. We measured the accuracy of the classifier according with the formula:

$$\text{Accuracy } (\%) = \left(1 - \frac{\epsilon_k}{N_{\text{test images}}}\right) \cdot 100 \tag{1}$$

Where ϵ_k is the number of classifications that didn't provide the right label in the first k positions. We also check how the precision evolves according to the number of components used for classification.

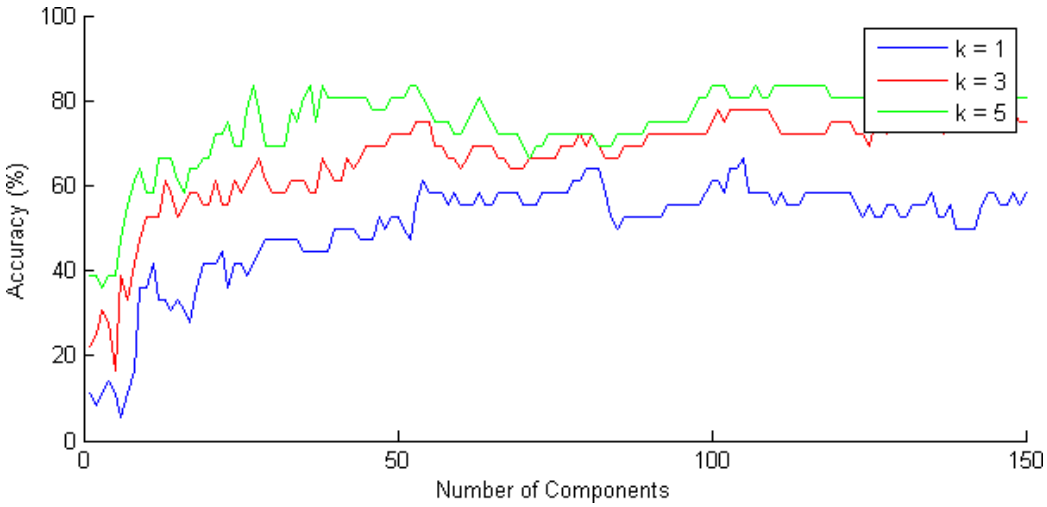


Figure 5: Recognition accuracy (%) for different ϵ_k and number of components

With enough number of components (c > 60) the classifier has an accuracy of around 50% for first match on the test dataset, getting up to 65% for first three matches. The result is satisfactory although it is far from optimal, the main weaknesses of this classifier are its dependency on the quality of data such as the right tagging of features and the similarity of face expression of the image to be recognised with any of the training set.

4.2 Conclusions

Classification of image content using PCA is tricky since you need normalized data such that all the features are uniform throughout the samples (e.g. the pixels in each image must correspond to the same physical location, the lightning conditions similar, good sampling of many angles...).

On the other side, we have also seen how much the problem is reduced from 4096 original features to 60, a reduction to 1.5% of the original size, which translates in less memory to store the databases and less computation time for classification.