Name: Andrew Hayes
Student ID: 21321503

Name: Cian Elberse
Student ID: 21406512

CT404

Lab Assignment 3: Faces

1 Execution Steps

To run the code, we first installed the necessary package ***image***:

```
1  pkg install -forge image
```

Listing 1: Installation of the ***image*** package

We also replaced the test image used in Step 10 with a test image of our own as we did not have access to the file:

```
165 % Path to a new test image that is not part of the training dataset
166 new_test_img_path = 'C:\Users\waqarsqureshi\OneDrive - National University of Ireland, Galway\Waqar -
    ↪  NUIG\NUIG - Teaching\2024-2025\CT - Digital Image Processing\Lecture\Lab-tesk-03\face.jpg';
167 new_test_img = imread(new_test_img_path);
```

Listing 2: Original filepath used for `new_test_img_path`

```
165 new_test_img_path = '../data/michaeld.jpg';
```

Listing 3: Replacement filepath used for `new_test_img_path`

Figure 1: Replacement test image used (Michael D. Higgins)

Figure 2: Replacement test image and the best match to it found

2 Effect of Training Images on Accuracy

With the original number of training images per person (9), the testing accuracy achieved was 90%.

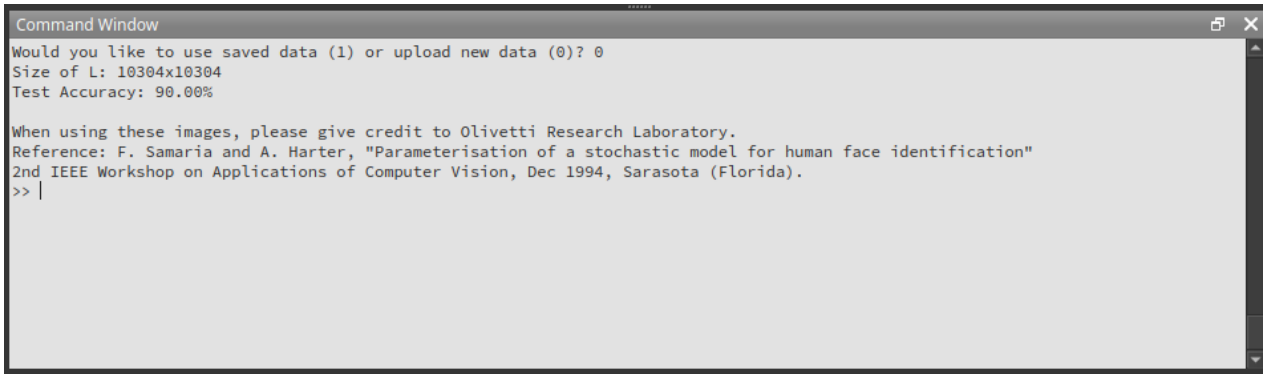

Figure 3: Testing accuracy achieved with original number of training images per person (9)

To reduce the number of training images used per person, we replaced the following line of code:

```
16 train_images_per_subject = 9;  % Number of images per subject for training
```

Listing 4: Original value of `train_images_per_subject`

```
16 train_images_per_subject = 5;  % Number of images per subject for training
```

Listing 5: Replacement value of `train_images_per_subject`

With the reduced number of training images per person (5), the testing accuracy achieved was 85%.

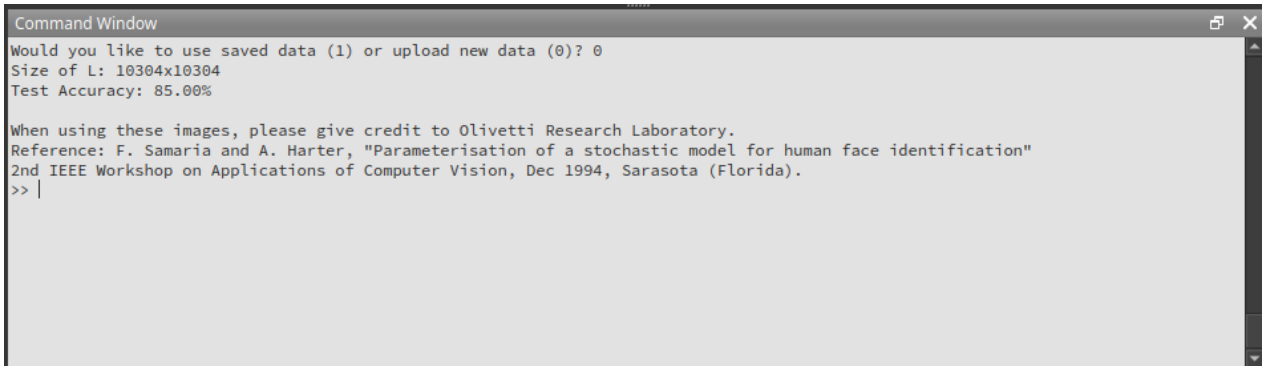

Figure 4: Testing accuracy achieved with reduced number of training images per person (5)

The reason why the testing accuracy decreased is likely due to reduced feature representation & dimensional coverage: fewer training images result in less variability, making it more difficult for the program to capture a comprehensive representation of each face. The eigenfaces algorithm relies on variance across the training images, and reducing the number of images will reduce the variance captured in the principal components, leading to reduced power to discriminate between different faces. With fewer examples, there is also a risk of overfitting wherein the mode may fit to the training data too closely and fail to generalise to unseen testing images.

3 Effect of the Number of Eigenfaces

We experimented with different numbers of eigenfaces including 40, 200, & 300 by assigning different values to `K` in the following line of code:

```
16 K = 200;                    % Number of eigenvectors (Eigenfaces) to use
```

Listing 6: Assigning the value of `K`

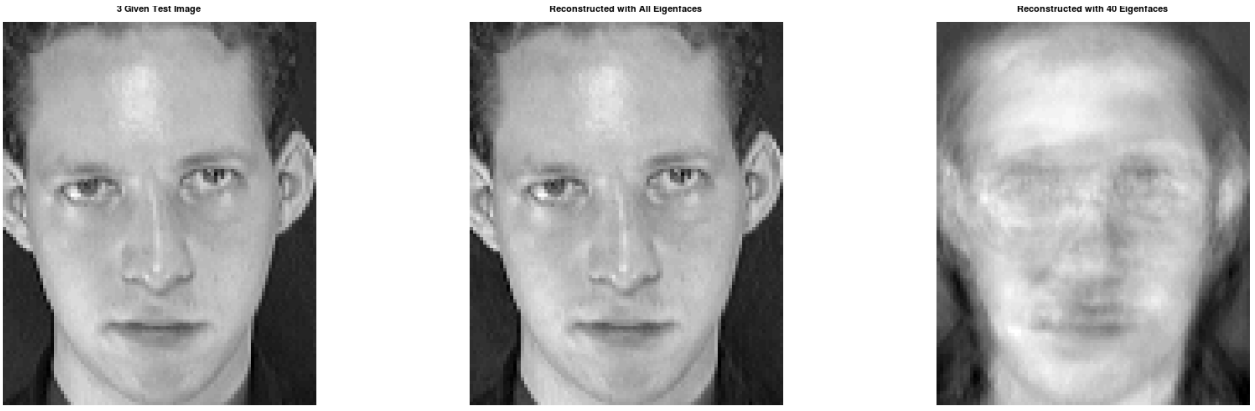

Figure 5: Reconstructed image achieved with `K = 40`

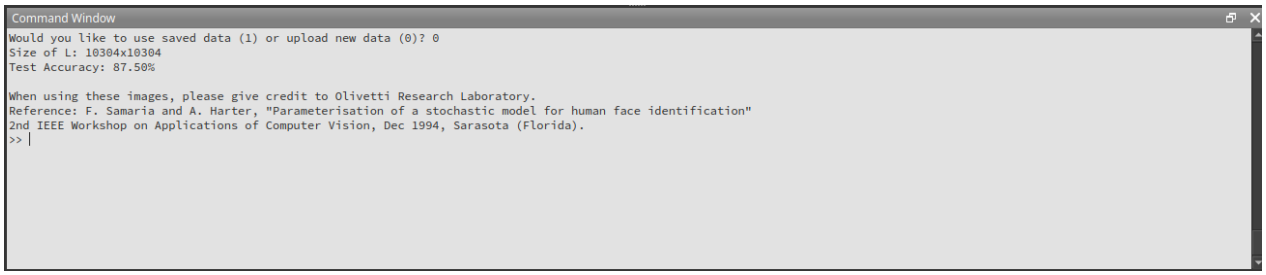

Figure 6: Testing accuracy achieved with `K = 40`

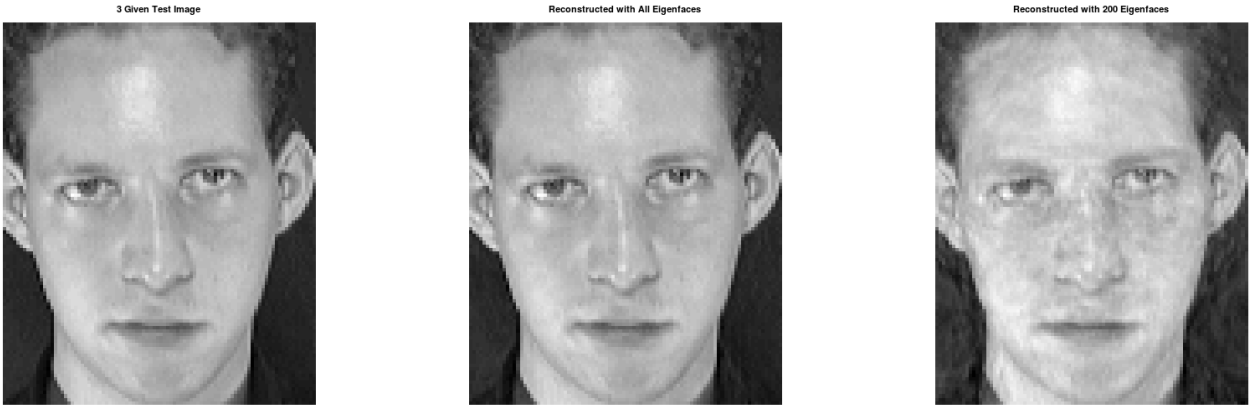

Figure 7: Reconstructed image achieved with `K = 200`

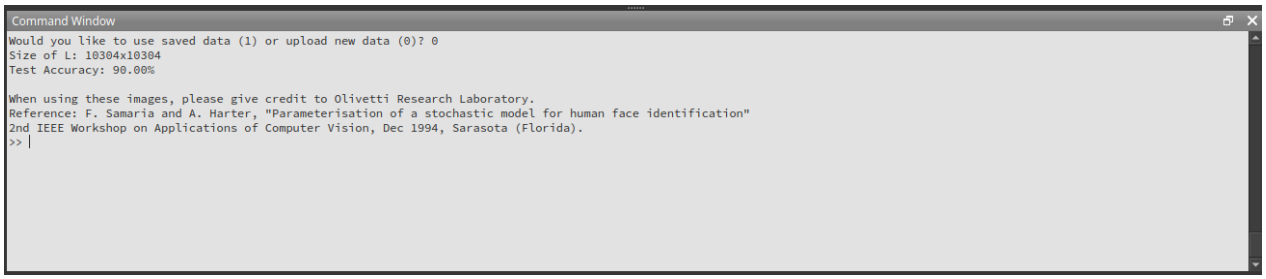

Figure 8: Testing accuracy achieved with `K = 200`

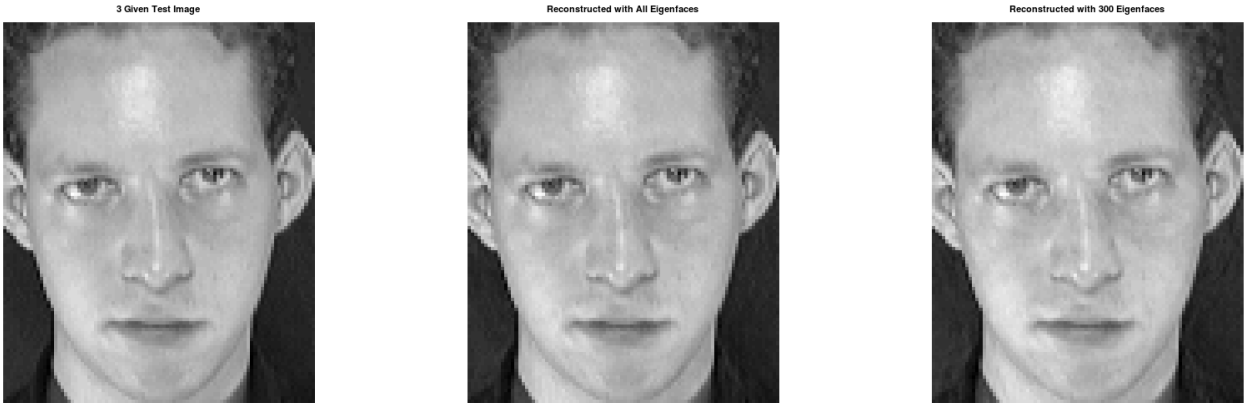

Figure 9: Reconstructed image achieved with `K = 300`

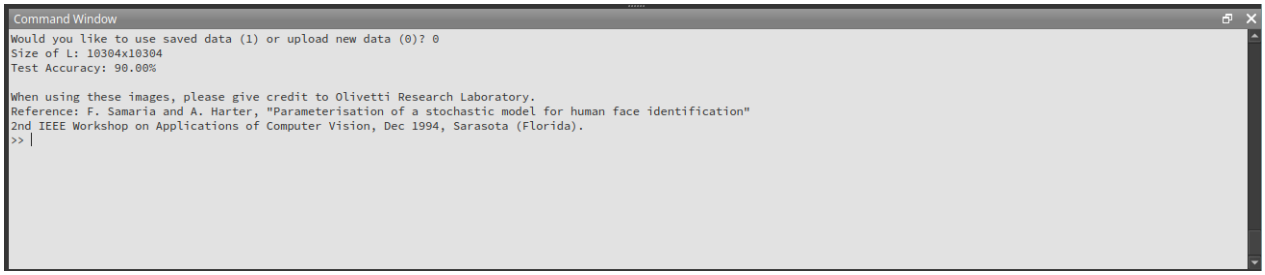

Figure 10: Testing accuracy achieved with `K = 300`

We observed that the quality of the reconstructed images appeared to scale with the number of eigenfaces used: the reconstructed image achieved with `K = 40` was barely recognisable as human face, the reconstructed image achieved with `K = 200` was far better but had quite a lot of blur & noise, and the reconstructed image achieved with `K = 300` was better again with reduced blur & noise. Interestingly, despite the marked improvement in the reconstructed images, the testing accuracy was not greatly affected: `K = 40` achieved a testing accuracy of 87.50%, `K = 200` achieved a slightly improved testing accuracy of 90%, and `K = 300` failed to improve on this and also achieved a testing accuracy of 90%.

The low testing accuracy with `K = 40` can be explained by there being insufficient eigenfaces to capture the all the relevant facial features. We can see that the features were better-captured with `K = 200`, and we observed that the improvement plateaued here as `K = 300` failed to improve the testing accuracy. This indicates that `K = 200` was sufficient to capture the relevant features.

4 Adding Your Friend's Faces

To add the images of the 5 new people to the dataset, we took 10 images of each person and added them to directories in the data directory named `s41` through `s45`. Because each image we had was either JPEG or PNG and the program relies on files being named `%d.pgm` where `%d` is some integer, we had to convert and re-name all the files for each person. As one of us was using a Linux system, we achieved this using a one-line shell script on the command line using a program called ImageMagick:

```
count=1; for file in *; do magick "$file" "$count.pgm"; ((count++)); rm "$file"; done
```

Listing 7: Script used to rename & convert image files

We then realised that the images had to be re-sized to work with the program as well, so we ran the following script to resize the images:

```
for file in *; do magick $file -resize 92x112! $file; done
```

Listing 8: Script used to rename & convert image files

We then edited the code to reflect the new number of subjects:

```
num_subjects = 45;        % Number of subjects in the dataset
```

Listing 9: Updated number of subjects

Figure 11: Best match found for unseen friend picture

The model recognised the new photograph of our friend very well, correctly matching the unseen test image with a training image of the same individual!