Instituto Superior de Engenharia de Lisboa

Computer Vision and Mixed Reality

Report

Feature Based

Face Detection and Recognition for Augmented Reality

Mestrado em Engenharia Informática e Multimédia

Rodrigo Dias, 45881

Semestre de Verão, 2021/2022

1 Introduction

In this project, we're going to develop algorithms for face detection and face recognition and allowing to include objects in the frame.

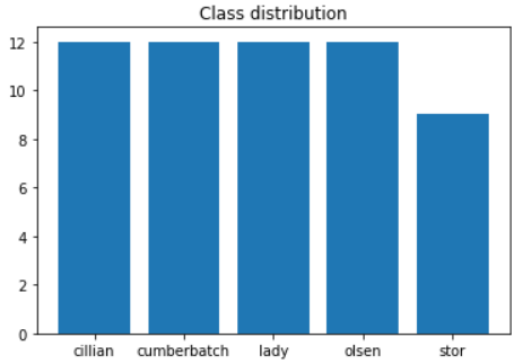
2 Face detection

For the face detection, it was used the Cascade Classifier to detect faces and eyes. It was created a class to handle the face detection, the eye detection and the alignment needed (rotation, scale, and translation). It was needed to calibrate the behavior of the classifier to correctly detect the eyes. Otherwise, there were times where the nose would be considered as an eye for example.

It was tested the DNN face detector too and it too was able to detect the faces with a similar result.

3 Database

It was created a dataset with 57 images total, with 5 different faces. The distributions are shown right.



4 Normalization

To normalize the images, it was used this classifier to find the faces in each image, which returns the part of the image containing the face and then apply the eyes detection. This is to better avoid detecting eyes were there's no face. From that we obtain the coordinates of the eyes in relation to the entire image.

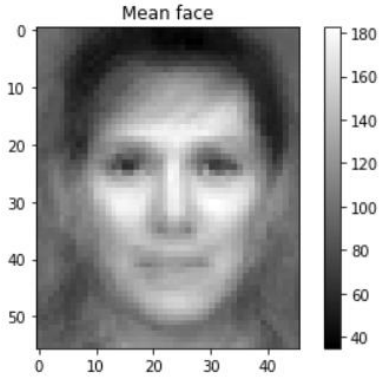
Afterwards, each image is downscaled to 56x46 pixels, converted to grayscale and, having the location of both eyes, we can calculate the angle of rotation and the scale factor and align the face accordingly using *warpAffine*, putting the eyes in line 26, columns 16 and 31, therefore normalizing the entire image.

Doing this to a set of images (57 in total), the dataset was prepared to apply the *eigenfaces* and *fisherfaces* algorithms.

5 Train/Test split

Before applying these algorithms, the dataset was split into training and test sets, having 2 images from each class in the test set. The training set now has 47 images while the test has only 10. Then the training set was randomized to better apply the algorithms and the test set was ordered.

6 Eigenfaces



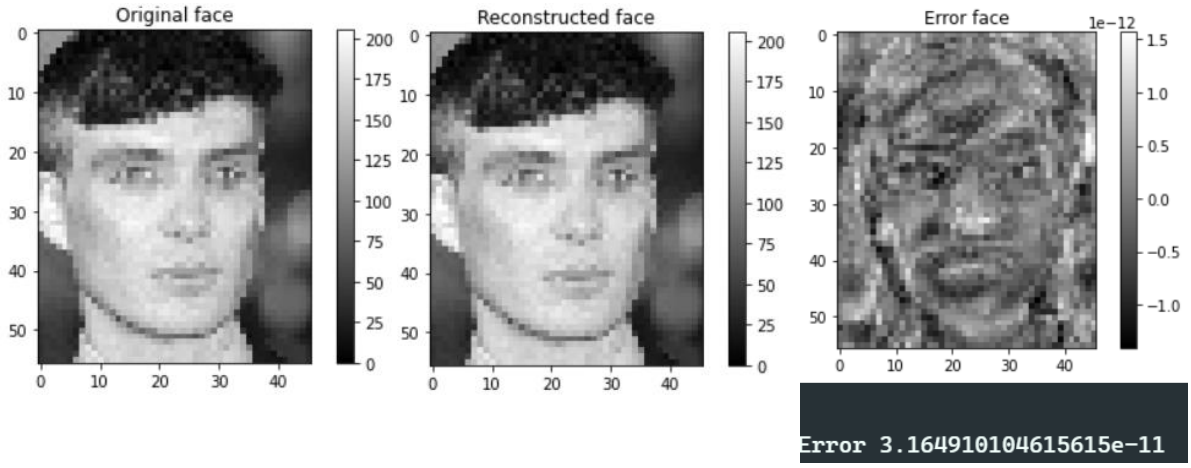
For the eigenfaces algorithm, the first step is to calculate the mean face between all images. This means, we will create another face where each pixel is the mean from the same pixel from all the train images. Then every face will be subtracted this mean face so that it's left with the 'AC components' of the faces. The result of this mean face is shown right.

Then we calculate that matrix transposed with the normal matrix to obtain a 47x47 matrix (47 images in the training set). This is another way simply to avoid calculating the normal matrix by the transposed one because it would result in a 2576x2576 and its way more intense to calculate.

Having this 47x47 matrix, the eigenvalues and eigenvectors are calculated, and we want to discard the vector with the least value, because its value is close to zero and can be discarded. Technically we could discard more vectors, but the classification would be worse because it would have less information to reconstruct.

Then we multiply the matrix with the AC components of the faces by these vectors and then normalize each one by its norm to get the face subspace. We can confirm that the resulted matrix is orthogonal because when multiplying its transposed version with itself, the result is close to the identity matrix.

In the end we obtain the projection by multiplying the face subspace transposed with the AC components. After that we can see the result of the reconstruction by going backwards and see the results. By human eye we can't see any difference, but when checking the error face, we see that there is some despite being almost zero, as shown below.



To check if the error face is orthogonal, we project the error face on the face subspace and see that the sum of every pixel is close to zero like shown above.

7 Fisherfaces

This algorithm was not developed in time.

8 Classification

For the classification, it's needed to remove the mean face from every test image now and project every image on the face subspace. It was used the matrices from thee eigenfaces algorithm because the fisherfaces wasn't implemented.

With the projection of the train and test faces, for each new test face it's calculated the distance between each vector from the test to see which one is closer. This one would mean that the information of the face is the most similar with a specific one, meaning that one should be from the same person.

That was made for every test face and then calculated the confusion matrix to see the results. We can see that the results are quite good, as shown right.

```
Confusion matrix
[[1 1 0 0 0]
 [0 2 0 0 0]
 [0 0 2 0 0]
 [0 0 0 2 0]
 [0 0 0 0 2]]
```

9 Combine real and virtual objects

In this step it was used the same concepts of the normalization. The idea is to have some objects with a green screen to create a mask around them, so that we can attach them to a real image.

The first step is to create the mask, placing the pixels that have a range of green to 1 and the rest to 0. The idea is to apply this mask to the original image and the inversion of this mask to the object so that we can add them together.

Once the masks are created, it was applied the concept of the normalization. For example, for the glasses, we define the location of the eyes and with that, its possible to get the scale factor and the rotation of the glasses and we just need to know the location and orientation of the eyes on the original image to make the transformation and translation.

Once the masks are in place with the original image, we multiply the masks as referred above and add them to create the final product.

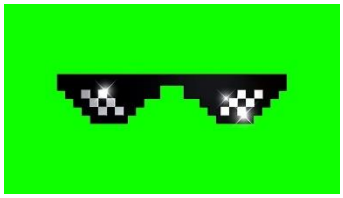


Figure 3 Glasses

Figure 2 Hat

Figure 1 Original image

In this example, there were used two objects, glasses and a hat, and the original image, shown above, to create the final one, shown right.

Figure 4 Result

It wasn't developed but the idea was to identify the person on the screen and apply a different virtual object to each person detected. What would be needed, let's say for each second, grab a frame and apply the classification mentioned above so that the algorithm detects the person and applies its unique object to that face. In this case, the

teacher would be the only one to see its own object because the rest of the dataset doesn’t include any of the students.

10 Conclusion

With this project it was possible to see the performance of the face and eyes detection and see how powerful that can be and how it can be used for everyday use of face recognition or simply for entertainment purposes.