TNM034 - Facial Recognition

Carl Englund
caren083@student.liu.se
Linköping University

Erik Sandrén
erila135@student.liu.se
Linköping University

Klas Eskilson
klaes950@student.liu.se
Linköping University

***Abstract*—Face recognition is a very difficult subject to master. This report will describe an implementation which was developed during a few weeks. The results which was gathered and a discussion of what could have been improved is provided.**

I. Introduction

The art of detecting faces and recognizing them is a part of our everyday life. Facial recognition is used as an authentication system for our phones and laptops. With the increasing personal information stored on these devices, the need for the security and reliance of the recognition has to be increased as well. We also expect the recognition to work with any conditions in regard of lighting conditions and orientation. Within the research area of image processing, these implication of these obstacles can be eliminated or their effect decreased.

This project applied some of these techniques in the implementation of a facial recognition system with mixed result in regards of false rejection rate (FRR) and false acceptance rate (FAR). FRR measures the rate of which a face in the database is not recognized while FAR measures the incorrect of the wrong person in the database or a person not in the database at all.

II. The problem

The aim of this project was to create a facial recognition system that would be able to recognize faces from a database of images. The images could have been modified and transformed in different ways to alter their appearances and compared with the database with successful results. Faces that didn't exist in the database would also be tested against the system. If the system found a face, it would respond with the id of that person in the database. The system would also report if a face wasn't recognized as being part of the database.

Three image databases were given. One of the databases (DB1) was supposed to be used as the reference database – these were the images the system was supposed to be able to recognise. Another database contained images of persons not in the database (DB0). The third (DB2) contained images of persons in the reference database, but where certain parameters were changed; the light might be brighter or darker, the background could be cluttered, the person out of focus and the white balance could be different from the first database as well as the person's facial expression. This database constitutes a great challenge, since these images are different from those in the first database. In order to allow these images to be recognised, the normalization process needs to be thorough.

III. Methodology

For correct recognition of a face the system had to do pre-processes of the image as well as normalization of the face's orientation and position. The face was then analyzed with Eigenfaces to compare with the database. Some of the steps described below used an adaptive thresholding during detection. The decision to do this was based on the will to not dismiss images and faces in the early pre-process steps but rather dismiss it during the comparison with the Eigenfaces.

A brief overlook of the needed steps, which are also described below, for the detection can be seen in figure 1.

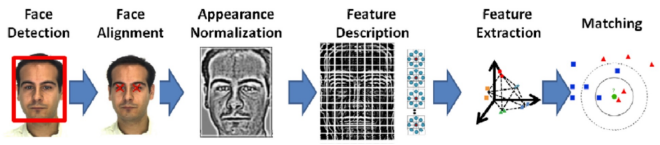

Figure 1: Overlook of the whole face detection and recognition process.

A. Lighting compensation

Since the method of detecting a face in an image is based on color, the potential difference in color temperature have to be eliminated. The method used in this project was *Gray world assumption*. The *Gray world assumption* method makes the assumption that the average value of the images *RGB* channels should be equal. The smallest average of the channels is determined and then all the values in the image is scaled after this average to eliminate any difference in white balance in images.

B. Skin detection

The first part of the recognition was to detect faces within an image. The method used in this project was based on the implementation described by [1]. The image's color space was first converted from *RGB* (Red, Green, Blue) into *YCbCr* (*Y* being the brightness, *Cb* the blue chroma and *Cr* the red chroma). The chroma channels were then transformed into a value based on the brightness value of the pixels. This transformation was done to be able to study the chroma values of each pixel in a 2D space. The criteria for skin color was approximated to an ellipse in the space of all chroma values that can be represented. The image was turned into a binary image with only pixels within the ellipse set to white. This mask would later be used as a mask to only show the skin

areas, see figure 2. The last step of the skin detection was to dismiss all skin areas except the biggest in the image and fills any holes in that area. The image could then be cropped to this area to only contain the face part of the image.

Figure 2: The face mask applied to the original image.

C. Eye map

When mapping eyes of a person a methodology presented in [1] was used. The input image was converted into *YCbCr* color space. This was done since observations showed that high Cb values and low Cr values were found around the eyes. Two different eye maps were calculated, one for the chrominance part of the eye and one for the luminance part. To calculate an eye map for the chrominance part equation 1 was used. For the luminance part morphological operations was used to smooth out small and irrelevant details. Equation 2 below was used to calculate the luminance eye map.

$$EyeMapC = \frac{1}{3}(C_b^2 + (\widetilde{C}_r)^2 + (C_b/C_r)) \tag{1}$$

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1} \tag{2}$$

To combine both eye maps elementwise multiplication was used. The combined eye map is then dilated and masked in order to isolate the eyes. The result of the eye detection can be seen in figure 3.

D. Mouth map

The mouth map was calculated in a similar way to the eye map. By measuring the different regions of the face the mouth can be detected based on the fact that it usually has a strong red component in the facial region. The equation for the map can be seen equations 3 and 4. Similarly to the eye map the input image was converted into the YCbCr color space and then processed.

$$MouthMap = C_r^2 \cdot (C_r^2 - \eta \cdot C_r/C_b)^2 \tag{3}$$

Figure 3: The result of the eye map.

$$\eta = 0.95 \cdot \frac{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)^2}{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)/C_b(x,y)} \tag{4}$$

E. Face alignment

With the eye map and mouth map the important facial points could be found. The maps were thresholded to create a binary mask. The maps were studied to find the centroid of each white area in the mask. The different eye and mouth candidates were then compared to each other to find the best face candidate. The face was then warped to a normalized state with specific positions for the eyes and mouth.

F. LogAbout

The purpose of the LogAbout method was to normalize the illumination of the faces. This was to improve comparison in images where the the face had a wide range of differently illuminated areas. The method was implemented using the method suggested in [2]. The input image was sent through a high pass filter that can be seen in equation 5. When the filter had been applied to the image it was transformed using a logarithmic function. The equation used for the logarithmic function can be seen in equation 6.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{5}$$

$$g(x,y) = a + \frac{ln(f(x,y)+1)}{b \cdot lnc} \tag{6}$$

G. Histogram equalization

Similar to LogAbout, the purpose of histogram equalization was to normalize the illumination of the faces. This method redistributes the image's intensity evenly throughout a larger spectrum in the histogram compared to before the equalization. This resulted in an improved contrast in the image. [3] In theory, this could strengthen the distinguishing facial features

of the images, but could also potentially increase the risk of failing to recognise new features introduced to faces over time.

H. Eigenfaces

Storing and comparing a potentially large image database requires a lot of memory, since each normalized image needs to be stored in a database and then compared to the current image. If no action is taken to compensate for this, a situation where the program's memory requirements exceeds what is available is likely to arise. To avoid this, and to lower the data size to a manageable amount, a method called principal component analysis (PCA) was used. In practice the number of dimensions in the high-dimensional dataset, that the image database constitutes, is reduced.

This dimension reduction was performed by focusing on the difference between the images. Correlated variables are not interesting when the aim is to distinguish variance and make decisions upon differences. What was interesting however was the uncorrelated variables. By calculating the mean image out of all images in the database and subtracting this mean from a candidate image, the unique features of the image were highlighted. This was used to calculate the covariance matrices for the images, which in turn was used to calculate eigenvalues and eigenvectors. These eigenvalues and eigenvectors were then used to match candidate images with the images in the database. The resulting images can be seen in figure 4.

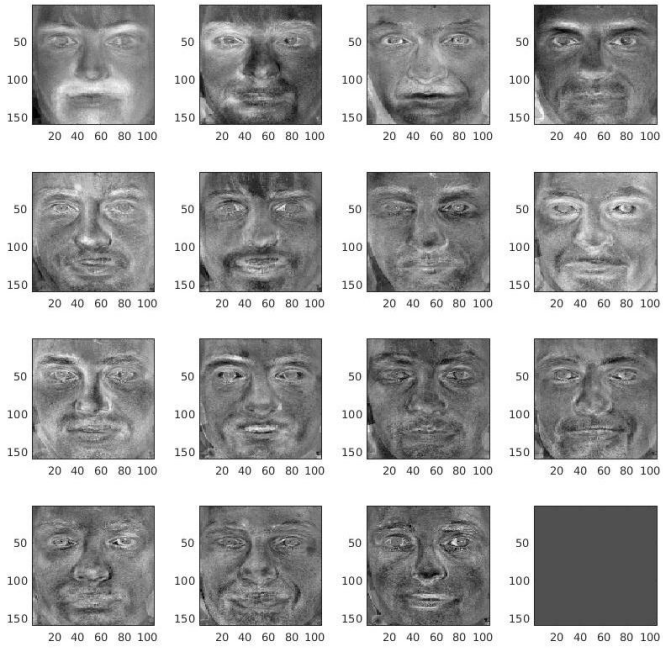

Figure 4: Eigenfaces in the database.

I. False negatives and false positives

Naturally, the program errors in some cases. There are two cases that deserves to be mentioned here; false negatives and false positives. False negatives are when the system does not match a face it should know. This can occur because the image is discarded in the thresholding of the summed error, when the system isn't satisfyingly certain of the result. False positives is when the program simply guesses wrong, but with such a certainty that it passes the thresholding that should stop it. The result amount of false negatives and false positives can be studied with two rates, *false rejection rate* (FRR) and *false acceptance rate* (FAR). FRR measures the rate in which a face in the database is not recognized and rejected, while FAR measures the rate of an unknown face mismatched with a face in the database.

IV. Implementations

The program was implemented in Matlab. It consists of one main function that accepts an RGB image as its only input variable, and returns the number corresponding to the correct face in the image database, if found. If however no match is found or the difference between the candidate image and the image stored in the database is to high, the program returns zero.

Initially, the images are color-corrected using the gray world-assumption method. The largest RGB value was detected and used to normalize every channel. The image's color-space was then converted to YCbCr using Matlab's built-in function. The chroma channels of the image were then transformed through an implementation of the methods and equations described above. When the chroma channels had been transformed, the mask for the face as well as the mouth and eye map was retrieved as described above. Between each step in the progress, the values were normalized to ensure that precision loss was minimal when values were rounded.

After completing the steps for retrieving eyes and mouth a triangle between the two eyes and the mouth was formed. To do this each eye centroid and mouth centroid was saved in two separate lists. The system then evaluated each mouth centroid with two eye centroids and gave this combination a score based on the assumptions that

- eye centroids must be above mouth centroid,
- less difference in height of the eye centroids are preferred and
- equal distance from the eye centroids to the mouth centroid was preferred.

The combinations were then sorted after the sum of these scores in ascending order and the first combination was the proposed face.

This helps later on to normalize all faces by performing a rotation of the faces. This was done by using Matlabs built in imwarp function. The pivot points for the imwarp were the left and right eye as well as the mouth. The threshold value of the masks was set to be adaptive and decrease if the system could not find a mouth or two sets of eyes.

In order to then further normalize the face, histogram equalization was used to make sure all images had a somewhat even illumination. The result of this equalization can be seen below in figure 5.

Figure 5: *Before and after histogram equalization*

To create the database, each normalized image were reshaped to a single vector and all images were then stored in a matrix with each row being an image. The PCA of this matrix was performed with Matlab's Single value decomposition function to retrieve the Eigenfaces sorted in ascending order of the Eigenvalue. The dot product of each normalized image and each Eigenfaces would result in weight values. These values describe how much an Eigenface contributed to recreate the image. The Eigenfaces and the weight values was stored to be used during the recognition process.

When an image was to be recognized, the Eigenfaces' weight values was retrieved from the image, a process that is described above. When the values for the candidate image had been calculated, the error for each image in the reference database of Eigenface values were calculated using the euclidian distance. The ID of the image in the reference database that had the smallest euclidian error compared to the candidate image gets returned if the error also passed the threshold of the maximum allowed error. This threshold value was found by taking the mean value of the error from all the false positives that the system returned during testing. This ID was what the program returned.

V. Result

All images tested below was retrieved from database one. Different combinations of luminance, rotation and scaling has been performed on the images according to the given boundaries from the examinator. In order to do this we set up a testing program where we could iterate over the different settings. All images were transformed with values where rotation started at -5 degrees, the tone value was decreased by -30%, and the images were scaled down by 10

Enhancements	**Recognized**	**FRR**	**FAR**	**Faces not found**
None	39.6%	18.4%	49.7%	0%
Gray World	44.9%	24.0%	58.2%	0%
Chroma transformation	50%	13.9%	61.1%	0%
Histogram equalization	53.1%	51.2%	40.6%	0%
GW + chroma	56.2%	21.5%	55.6%	9.3%
GW + hist.eq	51.6%	29.2%	50.2%	0%
Chroma + hist.eq	56.0%	24.7%	59.0%	0%
All enhancements	54.9%	46.5%	57.4%	9.3%

Databases	**Number of Images**	**Recognized**	**FRR**	**FAR**	**Fa fo**
DB0	108	-	-	13%	25
DB1	4	-	-	25%	0%
DB1	16	100%	0%	0%	0%
DB1	432	54.9%	46.5%	57.4%	9.3
DB2	38	31.6%	13.3%	75%	10
DB2	1026	28.4%	11.3%	90.0%	18

Image nr.	**Correct matches**	**No matches**	**FRR**	**FAR**
1	6 (22.2%)	-	22.2%	40.7%
2	4 (14.8%)	-	33.3%	51.9%
3	5 (18.5%)	-	37.0%	22.2%
4	3 (11.1%)	-	18.5%	70.4%
5	10 (37.0%)	-	18.5%	37.0%
6	10 (37.0%)	7 (25.9)%	33.3%	3.7%
7	8 (29.6%)	9 (33.3)%	37.0%	-
8	11 (40.7%)	-	59.3%	-
9	1 (3.7%)	-	88.9%	7.4%
10	6 (22.2%)	-	51.9%	25.9%
11	7 (25.9%)	3 (11.1%)	37.0%	25.9%
12	6 (22.2%)	-	44.4%	33.3%
13	5 (18.5%)	-	40.7%	37.0%
14	11 (40.7%)	-	22.2%	37.0%
15	7 (25.9%)	-	18.5%	55.6%
16	2 (7.4%)	-	44.4%	48.2%

Table 1: Images with different types of enhancements, Total number of images 432.

As we can see the different enhancement techniques that are applied to the images help to increase the correction rate a lot. There seems to however be a problem with faces not being found when all enhancements are applied, as well as when Gray world + Chroma transformation is combined.

Table 2: Tests on new images of faces in database(DB2) and face not in database(DB0).

VI. Discussion

The result obtained in our face recognition system with DB1 is dependent on the images being taken in an environment with no distracting background. With this problem in mind there were problems getting any okay result at all on DB2, the database containing the challenging images of persons in the reference database. What happened was that the cluttered background contains skin-like parts that confuse the facial feature detection algorithm, and makes it think that there is skin where there is none. This leads to a situation where the skin detection, whose main purpose is to aid the eye and mouth detection, returns an unreasonably large mask. Within this large mask, there might be several eye and mouth candidates, which combined together creates faulty face candidates. An improvement of the implementation discussed in this paper could therefore be some better face detection method to easier mask out the areas with disturbance.

As can be seen in table 3, some images did not respond to the countermeasures taken against the distortion that the

images were exposed to. Image number 9 only had one correct match out of 27, and in image number 7 there were 9 attempts where no face was found and the program canceled before any matching attempt were made. To understand what is going on in these particular cases, we take a look at the process of the face detection and normalization.

In the images in figure 6 to 9, the face detection process of image 7 is shown, where the image is darkened, rotated and scaled. As can be seen in figure 6, the face mask that is the result of the skin model function only returns a very small part of the whole face. This is the source of the issue – no face is detected since the eyes are excluded by the face mask. This is likely due to the lowered tone value of the image.

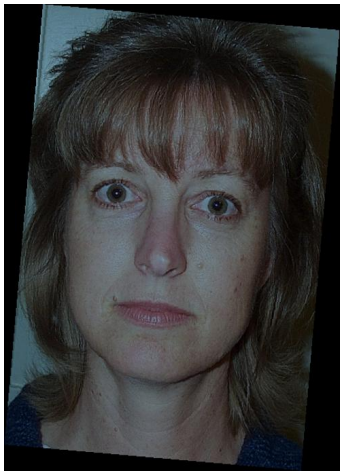

Figure 6: *Image darkened, rotated and scaled*

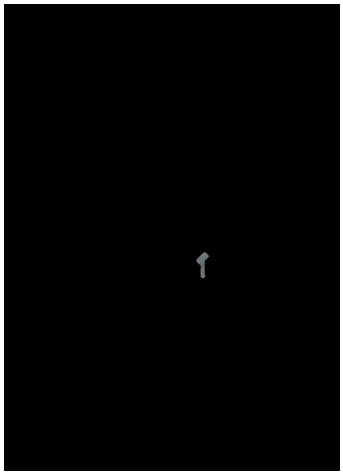

Figure 7: *Image masked*

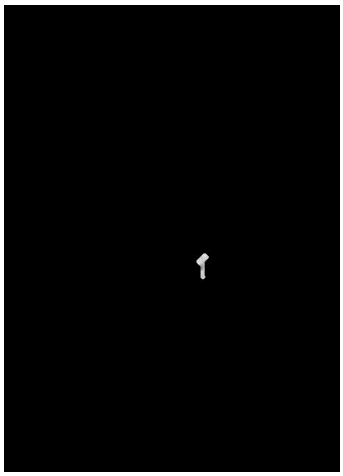

Figure 8: *Eye map of image*

Figure 9: *Mouth map of image*

In the images in figure 10 to 15, we see the completed detection and normalization of a rotated, lightened and scaled version of image 9. The normalization and facial detection works fine on this example, however it was registered as a false positive. The system guessed that this was a picture of another person.

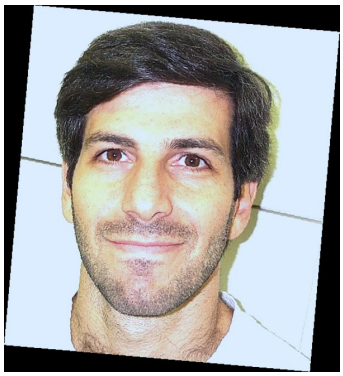

Figure 10: *Image darkened, rotated and scaled*

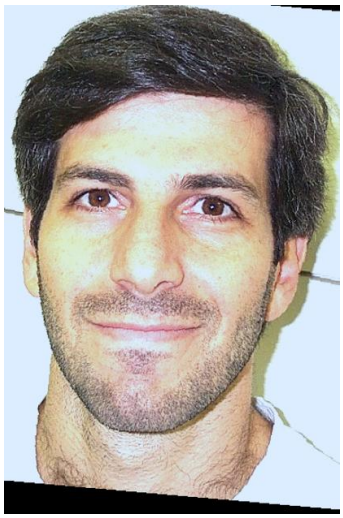

Figure 11: *Image cropped*

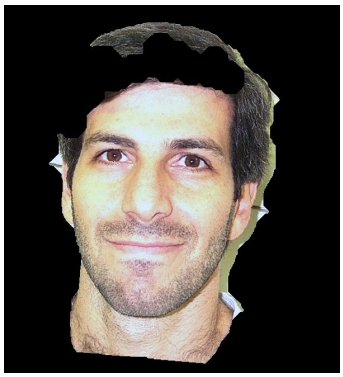

Figure 12: *Image masked*

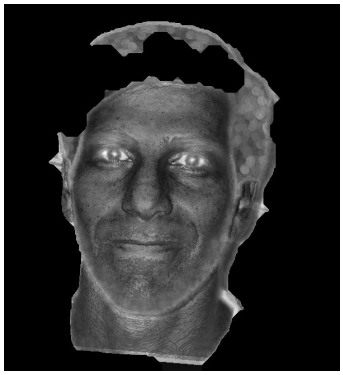

Figure 13: *Eye map of image*

Figure 14: *Mouth map of image*

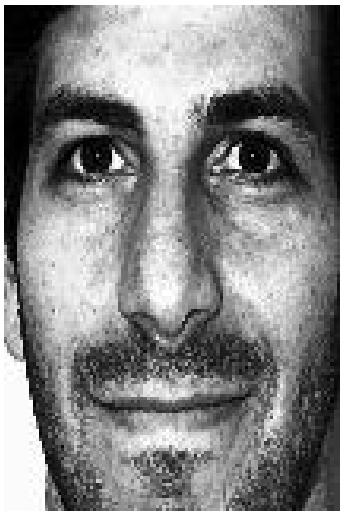

Figure 15: *Image normalized*

A. Fisherfaces

One method that could have been used instead of Eigenfaces to increase the success of our recognition is Fisherfaces. FIsherfaces rely on linear discriminant analysis (LDA), a method that succeed Fisher's linear discriminant method (hence the name Fisherfaces) and that recognises patterns in a set of images. Similar to PCA, this method focus a lot on dimensionality reduction. This method would replace the Eigenfaces state, where the faces are recognised and not detected, of the system discussed here.

However, Fisherfaces is a method that greatly benefits from having several photos of the same class, in our case image of a face. Here, Fisherfaces focuses on maximizing the ratio of the between-class differences and the within-class differences (Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection). Since we do know which class, again; face, each image belong to, we can use images from DB2 to create a better performing Fisherface database. This method can be considered more complex than Eigenfaces and relies on a solid and reliable facial detection system. Also, before this could be implemented, an improved face detection that works well on DB2 would have to be implemented.

According to Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection, Eigenfaces has a higher error-

rate when parameters such as lightning and expression vary. Therefore, the system is likely to benefit from an implementation of Fisherfaces. Other than the increased complexity, there isn't really anything negative with Fisherfaces compared to Eigenfaces.

B. LogAbout

LogAbout was implemented and is available as an extra normalization feature in the program. However it was hard to find values that performed well for the given datasets. The problem that was found was that it was hard to determine what values the constants a, b, and c should have in the equation. In the researched papers for the implementation of LogAbout there were no given constants since every dataset has its own values that fit for it. Therefore attempts were made to determine the constants by trial and error. There were no good results that came out of this and it was settled that the LogAbout method were not to be used in our implementation since it only worsened the final result.

VII. Conclusion

The face recognition process is something that a lot of research and time is put into. As stated in the beginning of this report a face that needs to be recognized could be rotated, have different illumination or a different scale than the reference image. The implementation suggested in this article works for some of the images that have been changed, but it has been proven that it is hard to implement a waterproof system.

Some improvements that could be made have been discussed. Fisherfaces could probably vastly improve the process, we do however not have that many training faces that seems to be necessary for that implementation. Other methods that also could improve our implementation is for example the LogAbout method. This has been implemented but the results we achieved did not improve our implementation. There are reports that show the LogAbout method improving recognition rate however.

Overall the art of face recognition is something very hard to implement in a dynamic way for real life usage.

References

[1] Hsu, R. L., Abdel-Mottaleb, M., and Jain, A. K. *Face detection in color images. Pattern Analysis and Machine Intelligence.* IEEE Transactions, 2002.

[2] Liu, H., Gao, W., Miao, J., and Li, J. *A Novel Method to Compensate Variety of Illumination In Face Detection.* Institute of Computing Technology,Chinese Academy of Sciences, 2002.

[3] Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., and Zuiderveld, K. *Adaptive histogram equalization and its variations.* Computer vision, graphics, and image processing, 1987.