Facial Recognition

A Project In The Course TNM034 - Adv. Image Processing

Group 2

Daniel Hagstedt
Ludde Jahrl
William Uddmyr

December 14*th*, 2020

Linköping University
Technological faculty

Abstract

This report covers the approach used by our group in an effort to create a face recognition algorithm in *MATLAB*. The project was developed for the course *Advance Image Processing* at Linköping University with an approximate 6 week development timeline. The methodology, implementation and result is described along with a discussion of the result and potential improvements. The result of this course and project was a *MATLAB* algorithm that is able to detect and recognize faces.

Contents

1 Introduction

Facial recognition is a widespread use of bio-metric identification. As image processing is advancing, the use of facial recognition has significantly increased, with uses such as unlocking a digital device or finding missing persons becoming all the more common. There are several methods and techniques to approach the subject of facial recognition. This project used simpler, well known and proven methods for facial detection and recognition.

2 Aim

The aim of this project was to create a algorithm in *MATLAB* that given a set of training images, `Db 1`, would be able to recognize the faces of a set of test images. There were two sets of test images, `Db 0` and `Db 2`. The training set `Db 1` was also used to test the algorithm but was altered using some different operations such as rotation and scaling. The images in `Db 2` due to some parameters were harder to detect and recognize. The images in all of the sets were of varying lighting conditions, orientation and alignment, contained a face but the face did not have to be included in the training set. Meaning that faces not in the training set would be tested against the system.

`Db 2`, would as previously mentioned, be more difficult due to some parameters being more exaggerated as well as some additional parameters not included in `Db 1`. These additional parameters could be more noise in the background of the image, a more varied light source, different facial expressions or simply being of less focus.

The system would, if it were to find a face, return the id of that person in the database, if not, it would report that the face wasn't recognized as a face of the database. The accuracy of the algorithm will be measured in success rate, *false rejection rate* (FRR) and *false acceptance rate* (FAR).

3 Methodology

The facial recognition was segmented into smaller steps and is explained in the following subsections, 3.1 to 3.3. The methods described below assumed that the image contained a mouth, a pair of eyes and that the eyes were located above the mouth.

3.1 Face detection

The face detection is the first step of the face recognition process. To accurately approach the subject proper masking of the skin and mapping of the eyes and mouth are required. The theories behind these steps are explained in detail in the following sections.

3.1.1 Color Correction

The basis of face detection is dependant on color. Since this is the case, the color temperature caused by different lighting conditions needed to be eliminated. This was done to make the lighting conditions more similar. The method implemented in the program to achieve this was *gray world lighting compensation* [1]. *Gray world lighting compensation* was chosen because of its superior results over *white point correction*. *Gray world lighting compensation* assumes that the average of a scene is achromatic and works by finding the average value of the red, green and blue channels and then normalizes each channel R, G and B using the highest average *RGB*

value. Equation 1 shows the average of each channel and how that value $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ is used in equation 2 to create the normalization constant $\hat{A}$, $\hat{B}$ and $\hat{\Gamma}$.

$$\hat{\alpha} = \frac{1}{Red_{avg}}, \quad \hat{\beta} = \frac{1}{Green_{avg}}, \quad \hat{\gamma} = \frac{1}{Blue_{avg}} \tag{1}$$

$$\hat{A} = \frac{\hat{\alpha}}{RGB_{Max}}, \quad \hat{B} = \frac{\hat{\alpha}}{RGB_{Max}}, \quad \hat{\Gamma} = \frac{\hat{\alpha}}{RGB_{Max}} \tag{2}$$

Equations 3 were then used to normalize the red, green and blue channels, shown in equations using $\hat{A}$, $\hat{B}$ and $\hat{\Gamma}$.

$$Red_{channel} = \hat{A} * Red_{channel}, \quad Green_{channel} = \hat{B} * Green_{channel}, \quad Blue_{channel} = \hat{\Gamma} * Blue_{channel} \tag{3}$$

This was done for every image handled by the algorithm to ensure even lighting conditions among images.

3.1.2 Color Space and Face Mask

Face recognition is heavily dependent on detecting and segmenting the skin region of the face. Due to most images being in the common *RGB* (red, green and blue) color space it can prove to be very challenging to detect skin as the three components represents luminance as well as color. Luminance is the the intensity of light emitted from a surface. As lighting conditions change from face to face the color space needs some channels to be luminance invariant. The color space chosen for this implementation was *YCbCr* (*Y* being luminance, *Cb* the blue chroma and *Cr* the red chroma). Skin was detected by checking if a pixel was within a certain span of luminance and chroma values. The implementation was based on *Face Detection in Color Images*, Hsu et. al. [3].

The face mask was implemented as a binary matrix that represented the skin region. The skin mask was then enhanced with morphological operations to remove potential noise in the background of the image that was erroneously labeled as skin. Lastly, only the largest singular area of the binary image was retained as to remove any false skin candidates. Any holes in the resulting mask were then filled.

3.1.3 Mouth Map

To map the mouth, a mask similar to the one implemented for the entire skin region was made in the *YCbCr* color space. The mouth mask utilised that human lips are generally more of a red hue than most other facial components. A mouth mask was then made utilising equations 4 and 5 [2]. Within the equations 4 and 5, η is the amount of pixels in the skin mask and the matrices C_r/C_b and C_r^2 are the chroma values for their individual channels normalized.

$$\eta = 0.95 * \frac{\frac{1}{n}\Sigma_{(x,y)} C_r(x,y)^2}{\frac{1}{n}\Sigma_{(x,y)} C_r(x,y)/C_b(x,y)} \tag{4}$$

$$MouthMap = C_r^2 * (C_r^2 - \eta * \frac{C_r}{C_b})^2 \tag{5}$$

3.1.4 Eye Map

The implementation used to map the eyes was based on the methodology of the paper which is described in a report written by Hsu et. al. [3]. The eye mapping also followed the same method of binary image masking

utilised in 3.1.2 and 3.1.3. Three methods of masking were used in tandem along with some morphological operations and logical trial-and-error. The three methods of masking used was *illumination-based-*, *color-based-* and *edge-density-based masking*. These methods are explained in the following sections along with their combined implementation.

3.1.4.1 Illumination-Based Masking

The *illumination-based masking* used equations 6 and 7 to find the eyes of the face in *YCbCr* color space. The conversion into *YCbCr* color space was done to isolate the eyes using the chroma and luminance components. Eyes showed low values of the red chroma and high values of the blue chroma. Equation 6 calculated the chrominance masking *Eye Map C*. The luminance eye map, *Eye Map L* used morphological operations to smooth and remove mislabeled eyes; the operation is shown in equation 7. These two masked were then combined using entry-wise multiplication to produce the *Illumination-Based Masking*.

$$EyeMapC = \frac{1}{3}((C_b)^2 + (\bar{C_r})^2 + \frac{C_b}{C_r}) \tag{6}$$

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1} \tag{7}$$

3.1.4.2 Color-Based Masking

The *color-based masking* assumes the eyes are the darkest regions of the face. The image was converted from a *RGB* color space to a *gray scale* color space the whites of the eyes will became the darkest part of the face. A low threshold was applied and the image was thresholded to create the second binary *eye mask*.

3.1.4.3 Edge-Density-Based Masking

The *edge-density-based mask* was created by converting the *RGB* image to a *gray scale* image and creating a mask using the edges detected in the new image with the sobel edge detection method. To further improve the result the binary mask was also dilated and eroded.

3.1.4.4 Combination of Eye Masks

After applying each of the methods described above, the remaining blobs were needed to satisfy four conditions in order to not be eliminated, as described in Hsu et. al. [3]. The *eye mask*s created in 3.1.4.1, 3.1.4.2 and 3.1.4.3 were then improved using the position of the mouth calculated in 3.1.3. The position of the mouth was used to remove sections that erroneously were labeled as eyes by examining the position of the areas. The masking area was removed if it did not satisfy some positional conditions, such as positions beneath the mouth not being possible eye candidates.

After applying the rules derived from the mouth position, the three methods were combined into a single *eye mask* using the logical operations *and* and *or*. Finally every area but the two largest ones were removed.

3.1.4.5 Trial-and-Error

If two separate areas describing the eyes were not found in the subsection described earlier,3.1.4.1 to 3.1.4.4, a threshold for pixel values accepted, were gradually lowered in the earlier stages of the implementation until two areas were detected in the final stage.

3.2 Face Alignment

The face recognition implemented within this algorithm was dependent on face alignment such that the eyes of all the images were to be aligned fully. This could only be achieved after the operations completed in 3.1 To be able to align every image such that both eyes overlap three distinct operations were needed. If the face is slightly tilted or if the eyes do not align due to different nasal bridge widths then complications will occur in later stages of the facial recognition. The details are further explained in 3.3.

3.2.1 Rotation

By utilising the position of the eyes extracted in section 3.1.4 and assuming facial symmetry of the eyes the angle α (in degrees) which the face is tilted was determinable. By using equation 8 the angle α which the image is needed to be rotated with was calculated. In equation 8 x_2 and y_2 is the position of the right eye and x_1 and y_1 is the position of the left eye which serves as the rotation point for the image.

$$\alpha = -\frac{180}{\pi} * arctan(\frac{(y_2 - y_1)}{(x_2 - x_1)}) \tag{8}$$

3.2.2 Stretching

Similarly to how 3.2.1 was implemented, the position of the eyes were calculated as in section 3.1.4 after the rotation.The new position of the eyes was used to stretch the image to align the eyes. By using the average width between the eyes as a scaling factor and stretching every image using this factor, all eyes were then aligned without significant distortions. The average width was calculated in equation 9 where *n* is the amount of images in the training set and ($x_2 - x_1$) describes the width for the *n*:th image. The average width is then used to calculate the *Scale Factor* in equation 10 which is then multiplied by the width of the image to align the eyes over all sets of images.

$$Width_{avg} = \frac{1}{n}\sum_{n=0}^{n}(x_2 - x_1) \tag{9}$$

$$ScaleFactor = \frac{Width_{avg}}{(x_2 - x_1)} \tag{10}$$

3.2.3 Cropping

As the dimensions of the images were not uniform a cropping using the eyes as point of reference were needed to ensure alignment. Using the left eye as point of reference all of the images were cropped uniformly. The dimension were chosen such that most of the facial features of the images was retained but not an excessive amount of unnecessary data remained. Acceptable losses of the features were the hairline, ears and the lower part of the jaw.

3.3 Creating eigenfaces

Face recognition may potentially require the need to store large data sets and require heavy computational power and storage. Large amounts of data may be discarded by using *Principal Component Analysis*, a dimensionality reduction algorithm henceforth abbreviated *PCA*. *PCA* is a linear orthogonal transformation that changes the transformed data's dimensions to be orthogonal; that is to say they become independent and without covariance or correlation. Instead of storing additional data and comparing high-dimensional images in our data set the

PCA algorithm reduces the dimensions of the data set exponentially by comparing the variance of the images. The former methods discussed in 3.1 and 3.2 need to be implemented properly for this algorithm to function accordingly. The normalised training set also needs comprise the entire representative. The upcoming methods regarding eigenfaces were based on the paper by Turk et. Al. [4].

3.3.1 Flattening the image

Images in the training set has formerly only been represented as a high-dimensional matrix. For the *PCA* to be utilized each image was transformed and treated as a vector in a high- dimensional space. This transformation is shown in equation 11 were I_i was a *gray scale* image represented as matrix and x_i was the image represented as a vector.

$$I_i = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1c} \\ a_{21} & a_{22} & ... & a_{2c} \\ ... & & & \\ a_{r1} & a_{r2} & ... & a_{rc} \end{bmatrix} \rightarrow x_i = \begin{bmatrix} a_{11} \\ a_{21} \\ ... \\ a_{r1} \\ a_{r2} \\ ... \\ a_{rc} \end{bmatrix} \tag{11}$$

3.3.2 Mean Face

Equation 12 was the equation used to determine μ where μ represents the *mean face* vector for the training set. *M* was the amount of faces in the training set and x_i was an image of the training set represented as a vector.

$$\mu = \frac{1}{M} \sum_{i=1}^{M} x_i \tag{12}$$

3.3.3 Normalize the Training Set

The *mean face*, μ, calculated in equation 12 was then subtracted from every image in the training set to get the variance of each vector now represented as ϕ. This is shown in equation 13. The vector ϕ is then a vector that describes only the unique features of a face of the training set.

$$\phi_i = x_i - \mu \tag{13}$$

3.3.4 Covariance, Eigenvectors and Eigenvalues

The covariance *C* was calculated as the product of two matrices, *A* and *A* transposed, shown in equation 14. *A* was a matrix composed of the variance of each vector, ϕ, for each image *M* in the training set. A major dimensionality reduction was achieved by the order of multiplication of the matrices *A* and *A* transposed. As *A* is comprised of the vectors ϕ, which has the length of the sum of all pixels in the normalized training set, the covariance *C* only becomes the size of *M* by *M*.

$$C = A^T A, \quad \text{where} \quad A = [\phi_1, \phi_2, ..., \phi_M] \tag{14}$$

From the Covariance *C* of equation 14, the *eigenvectors*, v_i was computed. v_i is a one dimensional vector with the length of *M*. It then follows that *A*, which was the variance of each vector ϕ, multiplied by the *eigenvectors*,

v_i resulted in the *M* largest *eigenvectors* u_i for the $n \times n$ matrix AA^T. This property is shown in equation 15 and was completed for each image *M* in the training set.

$$u_i = Av_i \tag{15}$$

3.3.5 Eigenfaces

The *eigenvectors* u_i from equation 15, were then reshaped into the original dimension of the image from the training set. The result of this is shown in Figure 4.

From the *eigenvectors* of u_i to u_M the *K* best can be chosen to represent the *K*-dimensional sub-space. Each face in the training set could then be represented as a linear combination of the *eigenfaces*. This linear combination is shown in equation 16, were w_j was the weight computed in equation 17 as the product of the *eigenvectors* u_i^T and the variance of each vector ϕ of each image *i* in the training set.

$$Image_i = \mu + \sum_{j=1}^{K} w_j u_j \tag{16}$$

$$w_i = u_i^T \phi_i \tag{17}$$

This resulted is an *K* by one vector Ω_i containing the weights in 17.

$$\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ .. \\ w_K \end{bmatrix} \tag{18}$$

4 Implementation

The following section will explain in detail how the methodology of section 3 was implemented in our *MATLAB* algorithm and how it functions. The algorithm is divided into several components, these components will be described further in the following subsections.

4.1 Creating the database

The database of *eigenfaces*, *weights* and the *mean face* was comprised of the training set of `Db 1` and was used to identify a face sent as a query to the algorithm. The first component of the algorithm to create the database is to detect the faces.

4.1.1 Face Detection

To create the *eigenfaces*, *weights* and the *mean face* which is used in the face recognition; face detection and face alignment is needed. The implementation was based on the methodology covered in 3.1 and 3.2.

A main function was implemented, having an *RGB* image from one of the databases as the sole input variable. The first step was to eliminate the different lightning conditions as described in 3.1. Then, in order to detect face-skin, the background needed to be eliminated from the image. This were done translating to YCbCr color space and eliminating background pixels using threshold values. The threshold values used in the algorithm was

based on experimenting with different values. The value chosen for the luminance was [$Y >= 70$], blue chroma [$Cb >= 110$, $Cb <= 180$] and red chroma [$Cr >= 129$, $Cr <= 185$]. But a threshold was not enough. There were still some holes and cracks in the thresholded face image, which were filled using morphological operations. The different stages in the skin mask component is illustrated in Figure 1.

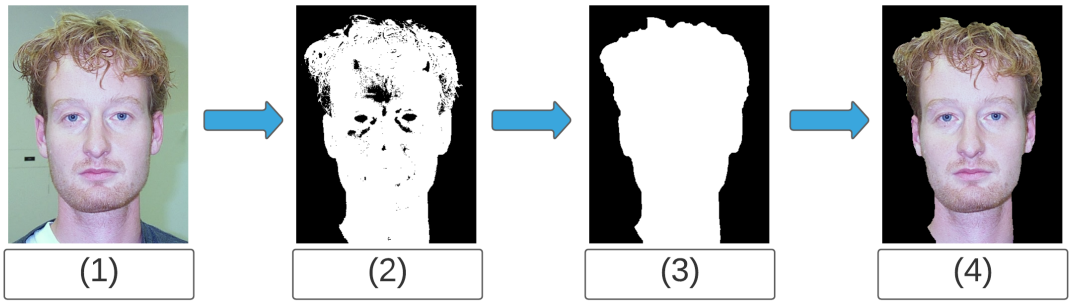

Figure 1: *The skin mask segmented into steps - (1) Shows the color corrected image. (2) Shows the image after YCbCr thresholding. (3) The image after morphological operations. (4) The color corrected image after segmenting the background with use of the image in step (3).*

As can be seen in the images above, all steps includes hair which were not successfully eliminated as the background. Segmenting the image at a specific height in this step is a risk since neither mouth or eyes are found, both features crucial in order to crop the image. Therefore the removal of the hair was postponed until eye positions were found. This is described in later steps.

With the help of the skin mask, described in section 3.1.3, the mouth map was improved by eliminating disturbances in the background component that contained high values of the red chroma that specified the mouth position. After testing later steps and studying the images in the set of images we also took for granted that the mouth is in the lower half of the image. The decision was important to achieve an even more accurate mouth map, without risking for example an eye being classified as the mouth. The different stages in the mouth mapping component is illustrated in Figure 2.

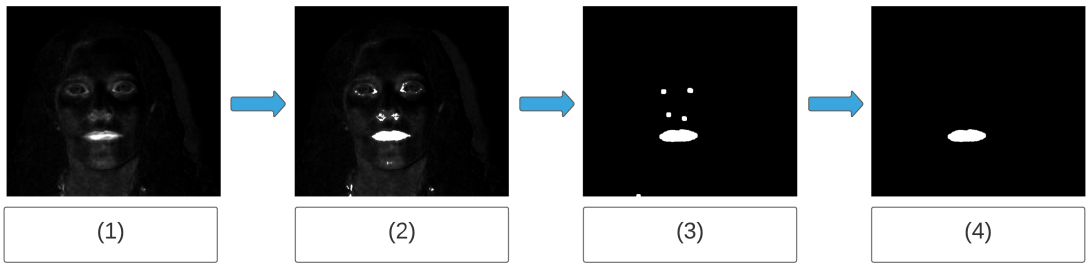

Figure 2: *The mouth mapping segmented into steps - (1) Shows the image after equation 5 had been applied. (2) Shows the image after thresholding. (3) The image after morphological operations. (4) The final result after discarding all objects except the largest one.*

The mapping of the eyes, described in section 3.1.4, was further improved utilizing the skin mask and mouth map. Improbable eye candidates were removed using the skin mask in the same way that they were removed in the mouth mask. The rules mentioned in 3.1.4 also removed blobs not satisfying the following conditions according to Shafi et. al. [3].

- Solidity of the region is greater than 0.5.
- The aspect ratio is between 0.8 and 4.0
- The connected region is not touching the border
- The orientation of the connected component is between -45 and +45 degrees

The mouth position also served to remove several more improbable eye candidates by using a bounding box using the position of the mouth center. As the eyes constituted a rather small portion of the image the eye mask sometimes only would capture one area, some conditions were introduced to counteract this. If none or only one potential eye candidate was found, a threshold for pixel values was lowered until the condition was met. The different stages in the eye mapping component is illustrated in Figure 3.

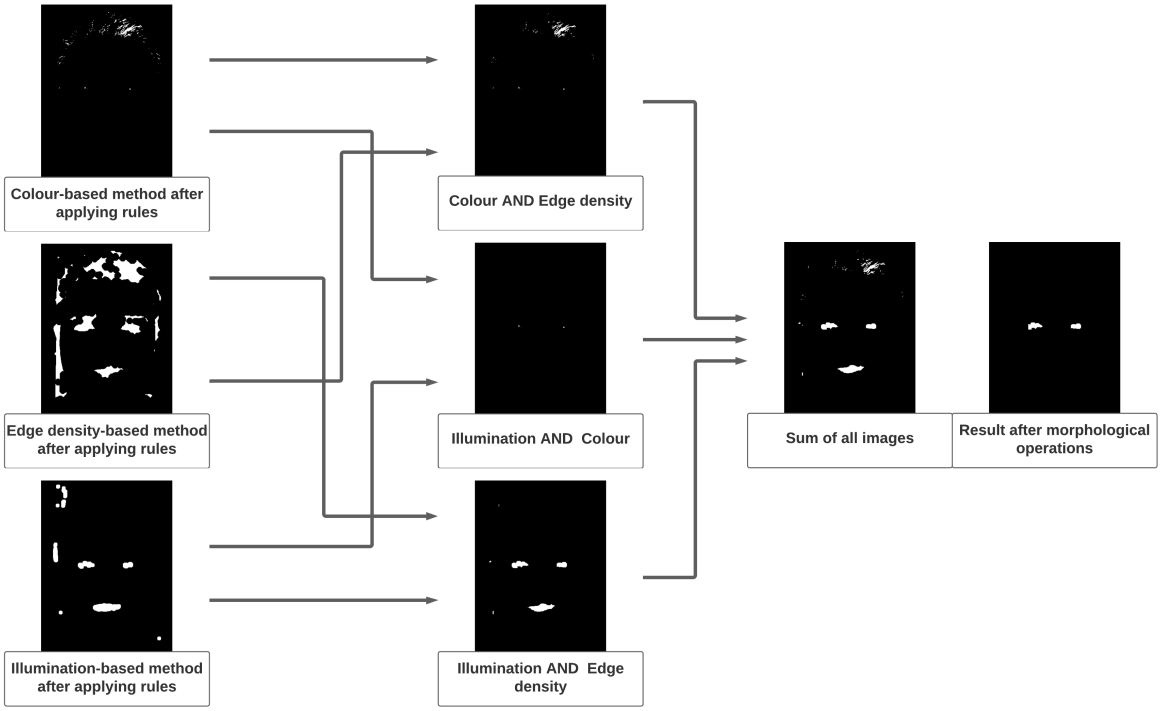

Figure 3: *The eye mapping segmented into steps*

The functions for face alignment, described in section 3.2, then used the eyes positions and rotated, stretched and cropped the image to fit the dimensions and positions needed to compare it to the database of eigenfaces.

4.1.2 Normalization

In order to handle alterations with different tone values in the images, a normalization was applied to the image. A scale factor shown in 19, was also calculated for optimal classification. This factor was multiplied with the normalized image, and resulted in all images having a mean value of 0.74. How this scale factor was discovered, and its importance will be discussed in 6.1.

$$GrayScaleFactor_i = \frac{0.74}{mean(Image_i)} \tag{19}$$

4.1.3 Creating eigenfaces

The creation of the eigenfaces which constituted the database was described in section 3.3 wherein all normalized and aligned image was flattened into a vector which was stored as a matrix where each column described an entire image of the training set. This matrix in turn was then used to perform the principal component analysis. The eigenfaces was then computed by assortment of the eigenvalues and the weights used to verify the face were calculated using the eigenfaces and the dot product of the normalized images. Ultimately, the *eigenfaces*, *weights* and *the mean face* was saved to the database to then be utilized in the next component of the algorithm.

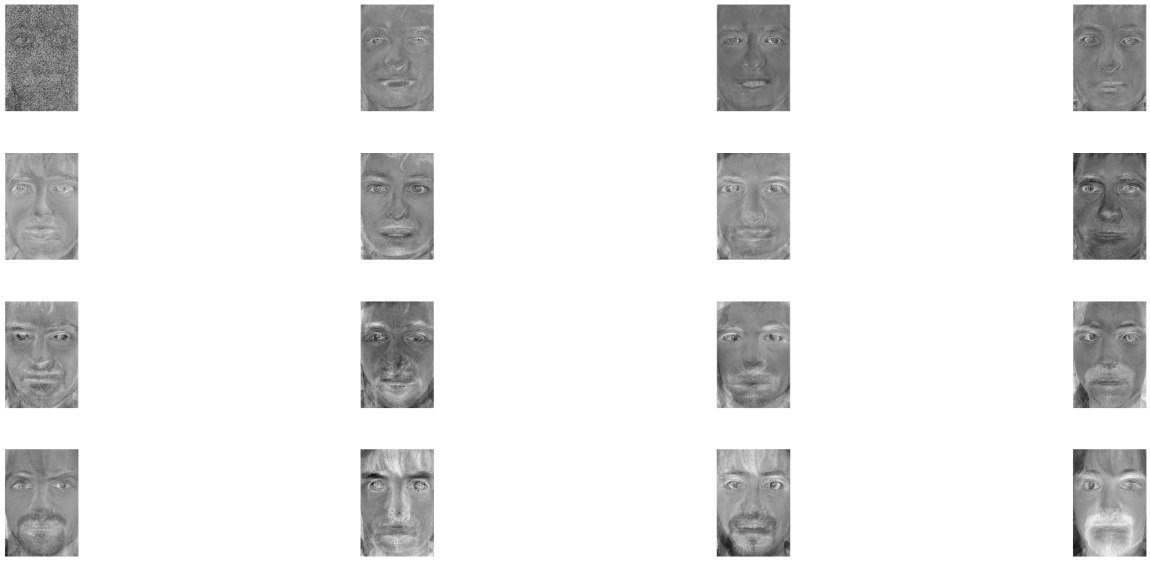

Figure 4: *The unweighted eigenfaces of the training set*

4.2 Face Recognition Using Eigen Faces

The face verification process uses the pre-computed variables *weights*, *the mean face* and the *eigenfaces* to verify the query image. The verification component projects the image onto the subspace which is spanned by the databases eigenfaces. This results in a weight vector. This query weight vector is then compared to the weight vectors of the eigenfaces of the database. The *euclidean distance* is then what determines if the query image is recognized among the faces in the training database. The minimum *euclidean distance* of the query images weight vector and the subspace weight vector determines which face in the database is the best match. The threshold value set in this component in the algorithm was 1500.0. The threshold value determines if the minimum distance is enough to validate the query face. If the query image was determined to be a valid face existing in the database its corresponding ID value was returned as the answer.

5 Result

The following section describes the results of the developed algorithm. The algorithm was first tested on the set of images in `Db 1` and then the same set, with alterations. The alterations were random rotations in the span of

$\pm$ 5 degrees, random scaling in the span of $\pm$ 10% and random tonal shifts in the span of $\pm$ 30% of all three color channels.

The result obtained after 50 iterations per image, without alterations was 100% correct face recognition's. Table 1 below shows the results obtained for each alteration, combination of all three alterations and an average. Note that the results of the combination section can differ from each alteration result, since the span of each alteration were randomly selected for each iteration of the set.

Table 1: Results after alterations on images in `Db 1`

Image nr.	**Random rotation (+-5 deg)**	**Resize (+-10 %)**	**Tone all three channels ($\pm$30 %)**	**Combination**
1	48 (96%)	41 (82%)	50 (100%)	32 (64%)
2	50 (100%)	50 (100%)	48 (96%)	44 (88%)
3	50 (100%)	47 (94%)	50 (100%)	36 (72%)
4	48 (96%)	50 (100%)	50 (100%)	30 (60%)
5	41 (82%)	34 (68%)	50 (100%)	28 (56%)
6	24 (48%)	25 (50%)	50 (100%)	17 (34%)
7	22 (44%)	32 (64%)	42 (84%)	10 (20%)
8	24 (48%)	19 (38%)	44 (88%)	6 (12%)
9	25 (50%)	33 (66%)	44 (88%)	21 (42%)
10	50 (100%)	44 (88%)	33 (66%)	35 (70%)
11	49 (98%)	50 (100%)	50 (100%)	40 (80%)
12	36 (72%)	29 (58%)	42 (84%)	20 (40%)
13	50 (100%)	45 (90%)	31 (62%)	31 (62%)
14	12 (24%)	50 (100%)	37 (74%)	5 (10%)
15	50 (100%)	50 (100%)	50 (100%)	46 (92%)
16	49 (100%)	50 (100%)	50 (100%)	45 (90%)
Average:	78.625%	81.125%	90.125%	55.75%

The majority of the fails come from FRR *false rejection rate*, which had a rate of over 98 %. The FAR *false acceptance rate* very low, with only 13 cases during the 800 tests.

There were 38 images in `Db 2`, with several images of each person in `Db 1`, in different conditions. The result obtained from the algorithm tested for these images is shown in Table 2 below. The set *bl* consist of unsharp - blurred images, *cl* with cluttered background, *ex* of different facial expressions and *il* with different illumination properties.

Table 2: Results of face recognition in `Db 2`

Set	**No. images**	**Correct index found**	**Classified correct**
bl	9	3	1
cl	16	6	1
ex	7	4	0
il	6	0	0

6 Discussion

This section discusses the result of the project and the difficulties of implementation.

6.1 Normalization and scaling factor for gray-levels

At first, only normalization was applied on the images in order to match the gray-levels between them to account for changes in tone values. However, after observing the resulting accuracy of classification, it was clear that only using this method was not sufficient, as the average correct classification was $\approx$ 15-20%. Therefore, additional methods were added as an experiment, such as histogram equalization. The results were still poor for most images, but one of them showed an accuracy of 92% when running 50 tests with varying tone values (±30%). This observation led to the conclusion that there might be an optimal value for the the gray-levels in the images in order to correctly classify them. The mean of the image with high accuracy was equal to 0.74, and by using this information a scaling factor for all other images could be obtained through equation 19. Using this method ensured that all images had the same mean-value, and the results dramatically improved as can be seen in table 1 under "*Tone all three channels* ($\pm$30%)"

6.2 Result of `Db 1`

The facial recognition performed on the altered images in `Db 1`, shown in table 1, yielded decent average results for combinations of alterations. The relative high success rate for the resizing and the tone alterations contrast with the accuracy of the rotation. The speculative reason for the success rate of some faces being so low when using combined alterations is due to poor eye mapping. The eye mapping component of the algorithm is accurate in the sense that it almost always is able to map out the eyes, but it however is not precise in the way it places its pivot points for the rotation component. In order to increase the overall accuracy, this component would likely need to be adjusted. The result could perhaps also be improved if the eigenfaces were created using multiple images of the subjects in `Db 1`. The same can be said for the results in DB2.

6.3 Result of `Db 2`

The results of table 2 illustrates how the algorithm responds when faced with more discord in the pictures. Even though the algorithm at times could detect and index the right face from the images in `Db 2` the euclidean distance was almost always to great to be able to classify the face correctly. If the threshold value for the minimum euclidean distance were to be increased, it would result in faces from `Db 2` being correctly classified, but it would in turn result in more erroneous validations, i.e. the FAR would greatly increase.

7 Conclusion

The methods used in this report provides a decent result for recognizing faces that have been slightly altered. However, for this method to be used in a real setting it would have to be improved greatly. Firstly, it would have to be able to recognize the faces from `Db 2` with a relatively high success rate, as in real life, appearances can vary from day to day. Secondly, it should be practically impossible to have an FAR above zero, as this could have serious consequences. As mentioned in 5, only 13 false acceptances were recorded during testing. This is a low number, but in practice this would not be acceptable.

References

[1] Baozhu Wang, Xiuying Chang & Cuixiang Liu. Skin detection and segmentation in human face in color images. Hebei University of Technology, Cangzhou, VOL. 4, NO. 1, (2011)

[2] R-L Hsu, M. Abdel-Mottaleb, A.K. Jain. Face Detection in Color Images. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 5, (2002)

[3] Shafi, Muhammad & Chung, Paul. (2008). A HYBRID METHOD FOR EYES DETECTION IN FACIAL IMAGES. Proc. of Int. Conf. on Computer Science (WASET 2008). NO. 32.

[4] Matthew Turk & Alex Pentland. Eigenfaces for Recognition. Vision and Modelling Group, The Media Laboratory. Massachusetts Institute of Technology, VOL. 3, NO. 1