Face Recognition

Algot Sandahl *algsa119*
David Robín Karlsson *davka030*
Arvid Magnusson *arvma553*
Viktor Sjögren *viksj950*

December 15, 2020

1 Introduction

The aim of this project is to explore how a face recognition algorithm can be designed and implemented to identify faces in images. The project is limited to images in a data set with *one* face per image.

2 The Data Set

The used data set is a collection of images from Caltech [1]. The images within this set are divided into three categories, *DB0*, *DB1* and *DB2*.

DB1 contains 16 images of faces with mostly small amounts of background detail and good lighting conditions. Consequently, this database is the easiest one to identify images in.

DB2 contains a set of 38 images of faces that are more difficult to detect because of factors such as different lighting conditions, cluttered backgrounds, blurriness or different facial expressions.

DB0 is the smallest set of only four images with varying difficulty with faces which should not be matched to any of the faces in DB1 or DB2, essentially utilized as a control group.

3 Theory

This section describes the theory of the various techniques used in the presented algorithm.

3.1 Face recognition pipeline

There are numerous different methods and strategies that can be used for facial recognition. A lot of modern systems use a deep learning approach that utilize powerful hardware and the enormous amount of data that is available today. This project employs a more classical approach to the problem, using recognition techniques such as PCA together with eigenfaces or fisherfaces. The overall pipeline for the recognition process is illustrated in Figure 1.

3.2 Essential features

In order to detect faces it is important to use key features that make up the foundation of the algorithm. Features such as eyes, ears, nose and mouth to name a few are all distinguishable features, however looking at all at once is proven to not be necessary [2].

Instead, the most important traits can be narrowed down to the eyes and the mouth, which this project has focused on.

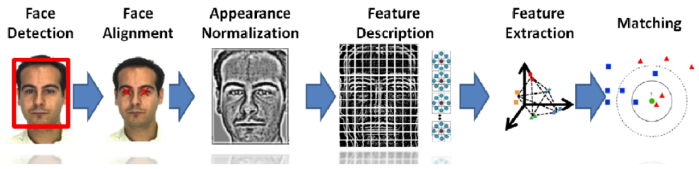


Figure 1: The facial recognition pipeline in the project.

3.3 White Balancing

In photography, the color of objects do not just depend on the color of the object, but also the color of the light and general lighting conditions. This is important for face recognition, not in the sense that the lighting of the faces should look correct, but rather that the dependency on the light source temperature should be removed, since it is not interesting in the detection or identification of a face. This allows for a uniform description of the faces and improves the conditions for the next steps in the pipeline.

Two global white balancing methods were implemented and tested in this project, however only the gray world assumption was finally used.

3.3.1 Gray World

The gray world algorithm is a lighting based method to approximate the light source color [3]. The method assumes that the intensity of the *RGB* channels is the same when looking at the average. This means that the average of the image is presumed to be achromatic.

In order for this criterion to be satisfied, each channel in the image needs to be adjusted. To do this, the mean pixel value is calculated for each of the red, green and blue channel as

$$\begin{aligned} R_{avg} &= \frac{1}{mn}\sum_{x=1}^{n}\sum_{y=1}^{m}R(x,y) \\ G_{avg} &= \frac{1}{mn}\sum_{x=1}^{n}\sum_{y=1}^{m}G(x,y) \\ B_{avg} &= \frac{1}{mn}\sum_{x=1}^{n}\sum_{y=1}^{m}B(x,y) \end{aligned} \tag{1}$$

where n is the number of pixels horizontally and m is the number of pixels vertically. R, G and B is the input image's color

channels. Using these mean values, the relation between red to green and blue to green can be calculated through

$$\hat{\alpha} = \frac{G_{avg}}{R_{avg}} \quad \text{and} \quad \hat{\beta} = \frac{G_{avg}}{B_{avg}} \tag{2}$$

where $\hat{\alpha}$ and $\hat{\beta}$ are the *gains* for the red and blue channel. The gains are then multiplied to the respective red or blue channel of the input image, which will balance out the channels and create a gray world version of the image.

3.3.2 White Patch

The white patch algorithm, also called white world assumption, tries to achieve the same thing as the gray world assumption by making photographs appear in the same color temperature. It does however differ in it's theory and it's final result as it presumes the whitest pixel to be a specular reflection caused by the overcasting light source.

This algorithm does similar steps as the gray world algorithm, however the mean is not computed for each channel as in equation (1). Instead the max values are calculated for the input image. Furthermore, very similarly to the gray world assumption the gain for two of the channels, red and green is calculated using the obtained max values instead of the mean values.

Lastly, in the same way as the gray world algorithm the gains are then multiplied to the input image's red and blue channel to obtain the white world color corrected image.

3.4 Face mask

The process of identifying facial features can be thrown off by background clutter that is similar to human eyes or mouths. Therefore, a face mask isolating human faces from the background is utilized. The face mask is constructed by comparing pixel values to a database of skin color samples as detailed in [2]. Since the RGB color space doesn't make any distinction between lightness and color the images are transformed into the YC_bC_r color space which allows observations of skin color tones in blue and red channels. To ensure that the skin tone values extracted from the images are independent from the luminance channel, a slightly modified transform from appendix A of [2] is applied to the chrominance channels as

$$C_i'(Y) = \begin{cases} (C_i(Y) - \overline{C_i}(Y)) \cdot \frac{W_{c_i}}{W_{c_i}(Y)} + \overline{C_{i,K_h}} & \text{if } Y < K_l \text{ or } K_h < Y \\ C_i(Y) & \text{otherwise} \end{cases} \tag{3}$$

where i defines either the blue or red chrominance channel, $W_{c_b} = 46.97, W_{c_r} = 38.76, K_l = 125, K_h = 188$ are constants defining cluster averages, low and high luminance threshold values respectively. $\overline{C_{i,K_h}}$ is a displacement constant used to accentuate the result of the transformation, its value was obtained heuristically. All other constants in equation (3) and subsequent equations (4), (5) and (6) are detailed in [2]. As evident in equation (3) above, the chrominance channels' values are only transformed when the luminance value Y satisfies the thresholds $Y < K_l$ and $Y > K_h$ implying that relatively lighter and darker regions of the image are the targets of this transform.

Figure 2: An image transformed to the YC_bC_r color space.

Figure 3: An YC_bC_r image where the chrominance channels have been transformed to be independent of the luminance channel.

Each chrominance channel's center $\overline{C_i}(Y)$ are defined as

$$\overline{C_b}(Y)) = \begin{cases} 108 + \frac{10(K_l - Y)}{K_l - Y_{min}} & \text{if } Y < K_l \\ 108 + \frac{10(Y - K_h)}{Y_{max} - K_h} & \text{if } K_h < Y \end{cases} \tag{4}$$

$$\overline{C_r}(Y)) = \begin{cases} 154 - \frac{10(K_l - Y)}{K_l - Y_{min}} & \text{if } Y < K_l \\ 154 + \frac{22(Y - K_h)}{Y_{max} - K_h} & \text{if } K_h < Y \end{cases} \tag{5}$$

where Y_{max}, Y_{min} always are 235 and 16 respectively due to the nature of the RGB to YC_bC_r transform.

Each chrominance channel's cluster $W_{c_i}(Y)$ are defined as

$$W_{c_i}(Y) = \begin{cases} WLc_i + \frac{(Y - Y_{min})(Wc_i - WLc_i)}{K_l - Y_{min}} & \text{if } Y < K_l \\ WHc_i + \frac{(Y_{max} - Y)(Wc_i - WHc_i)}{Y_{max} - K_h} & \text{if } K_h < Y \end{cases} \tag{6}$$

where $WLC_b = 23, WHC_b = 14$ and $WLC_r = 20, WHC_b = 10$. As seen in figures 2 and 3 the transformation described in

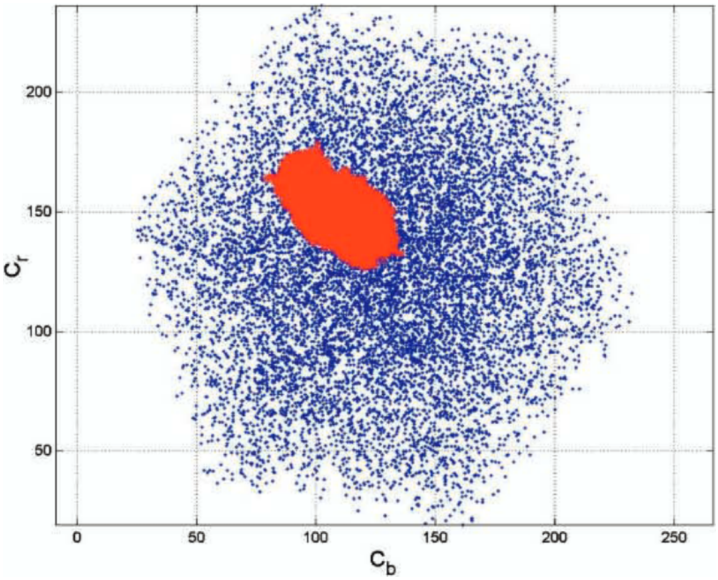


Figure 4: The skin tone color cluster in the blue-red chrominance subspace.

Figure 5: An image with its calculated face mask applied.

equation (3) clearly separates the skin regions from the rest of the image which made thresholding more robust. By projecting the chrominance channels' values onto any luminance value, the authors of [2] found a region on the resulting 2D-plane in which blue and red skin color tone values formed a distinct cluster. The cluster can be seen in Figure 4, which can roughly be fitted into an ellipse defined as

$$\frac{(C_b - 1.60)^2}{644.65} + \frac{(C_r - 2.41)^2}{196.84} = 1. \tag{7}$$

The ellipse (7) is then used as a threshold to segment the skin regions in the image, resulting the in first makings of a face mask. The segmented binary image is morphologically operated on to remove leftover noise from the segmentation and to create a continuous face mask. The resulting masked face from Figure 2, 3 can be seen in Figure 5.

3.5 Mouth Map

To find the mouth in an image, examining the chrominance and luminance channels is advantageous when considering the fact that mouths are usually more red and less blue than the rest of the face. The mouth map is based on [2] and is formulated as

$$\begin{aligned} MouthMap &= C_r^2 \cdot (C_r^2 - \eta \cdot C_r / C_b)^2, \\ \eta &= 0.95 \frac{\frac{1}{n}\sum C_r(x,y)^2}{\frac{1}{n}\sum C_r(x,y)/C_b(x,y)} \end{aligned} \tag{8}$$

where C_b, C_r are the chrominance channels, the sums are the average values of $C_r(x,y)^2$ and $C_r(x,y)/C_b(x,y)$ respectively and η is the ratio between the these averages. The resulting mouth map is then masked using the face mask described in section 3.4 and contrast stretched afterwards, making sure the highest intensities in the mouth map are centered around mouth. The mouth map is then segmented to remove all but the highest intensities. The binary image is then morphologically operated on to remove leftover noise from the segmentation and to create a continuous mouth map.

3.6 Eye map

When attempting to identify eyes in an image, examining the chrominance and luminance channels is once again advantageous as the eye regions exhibit certain traits which can be used to construct an eye map. In the chrominance channels, the region surrounding the eyes have low values in the red channel but high values in the blue channel. The luminance values of the pixels within the eyes are a mixture of both low and high, corresponding to bright and dark spots. To formulate this, [2] proposes that two separate eye maps be constructed. *EyeMapC* uses the chrominance features of the image and *EyeMapL* the luminance features. These are constructed as

$$EyeMapC = \frac{1}{3}\left\{C_b^2 + \tilde{C_r}^2 + \frac{C_b}{C_r}\right\} \tag{9}$$

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1} \tag{10}$$

where C_b, C_r and Y are the chrominance and luminance channels, $\tilde{C_r}$ is the complement of the red channel, g_σ is a structuring element used for dilation and erosion. These two eye maps are combined with element-wise multiplication and then dilated to accentuate the high intensity eye region. The resulting eye map is then masked using the face mask described in section 3.4 and contrast stretched afterwards, which makes sure the highest intensities in the eye map are centered around the eye regions. The eye map is then segmented to remove all values that are not among the highest intensities, creating a binary eye map. The binary image is then morphologically operated on to remove leftover noise from the segmentation and to create a continuous eye map.

3.7 Face triangle

For the more difficult images the eye map and mouth map will return multiple candidates for the eyes and mouth. In order to find the correct eyes and mouth an eye-mouth triangle is formed for all

possible combinations. All combinations are filtered using rules to cull bad candidates, resulting in coordinates of the centroids for the best eye and mouth candidates. The numerical constraints defining these rules were obtained heuristically.

3.7.1 Mouth candidate filtering

The first step to creating the face triangle is to find the most viable mouth with the following rules:

- The mouth should be in the bottom half of the image.
- The mouth should be wider than it is tall.

If there are multiple candidates fulfilling this, the last filter is to pick the one with the largest area.

3.7.2 Eye candidate filtering

With the best mouth candidate chosen, it can be used to filter out eye candidates. The filtering is done in pairs of eye candidates with the following rules:

- The eyes must be above the mouth but not too far from it vertically.
- The left and right eye must be on the correct side of the mouth.
- The vertical distance between the eyes cannot be to long.
- They need to be within a reasonable and similar angle to the mouth.
- The area of the candidates needs to be reasonably similar.
- The magnitude of the vector between the eyes and the mouth has to be reasonably similar.

If there still are multiple potential eye pair candidates, the pair with the lowest angle to mouth is chosen.

3.7.3 Pupil identification

After the best eye candidates have been identified, another pass of analysis is done in an attempt to find the coordinates of the subject's pupils which makes normalization of the images, described in the next section, more consistent. The centroid coordinates found in the last steps do not always end up on the pupil of the eye but more often than not on the iris of the eye. To remedy this, the following algorithm is used: The original image is masked by only letting through pixel values in a circle with $r = 3$ (euclidean) around the centroid coordinates. To see if the centroid managed to center right on the pupil, the image is then converted to gray scale and searched for high intensities in an attempt to find the camera flash reflection in the pupil. If it is found, no further computation is necessary. Otherwise, the original image is masked the same way as described earlier but with the radius $r = 13$ (quasi-euclidean) in an attempt to capture the whole eye.

The resulting masked image is then converted to the *HSV* color space which enables searches of specific hues, specifically blue and brown, to identify the iris of the eye. The hue channel is then thresholded to segment out all pixels not contained in the blue $200° < H < 270°$ or brown $5° < H < 55°$ interval. A helpful consequence of filtering in the hue channels is independence from the saturation channel which means that the pupil's reflection of the camera flash could be let through.

The segmented binary image is then regionally analyzed. If a single region is present the centroid of it is assumed to be the pupil. If multiple regions are present the areas of them are compared to find the pupil's flash reflection, which was heuristically found to be at most three pixels large. If such a region does not exist all regions are assumed to be a part of the iris. Thus, the average of these regions' centroids are assumed to be the pupil.

3.8 Face normalization

To enable the usage of the techniques described in the following sections, standardization of all the faces are made. This includes scaling, rotation and changing the color tone to normalize all faces. This is done by utilizing the coordinates of the eye and mouth described in section 3.7.

The scaling normalization sets the image to a predetermined fixed size, while also making sure that the coordinates of the eyes and mouth is set to a fixed position by scaling and rotating the image. The tone values are normalized by mapping the *RGB* values to gray scale values.

3.9 Eigenfaces

The dimensionality reduction technique used is *principal component analysis* (PCA), and the use in face classification as described here is based on [4].

The main idea is to form a lower dimensional feature space that spans the most important variations in the data set. To classify a face it is then projected onto this space and the distance to known faces in this space is used to classify the face.

To facilitate the calculations, each of the M images is reformed to a vector, $x_i \in \mathbb{R}^n$, where n is the number of pixels in each image. Note that n is the same for all all images as a consequence of the normalization step as described in section 3.8.

The first step in calculating the PCA is to form the mean vector, μ, of all x_i. In Figure 6, the mean for *DB1* is visualized. The mean is then subtracted from each vector:

$$\phi_i = x_i - \mu. \tag{11}$$

The next step is to find the covariance matrix

$$C = AA^T, \qquad \text{where} \qquad A = [\phi_1, \phi_2, \ldots, \phi_M] \tag{12}$$

but for computational reasons this is not feasible since C would be a $n \times n$ matrix, and the number of pixels, n, in an image is substantial; it would also mean that n eigenvectors are obtained. It turns out, however, that only $M - 1$ of the eigenvectors carry any meaning (given $M < n$).

From the eigenvectors v_i of A^TA it is given that

$$A^TAv_i = \lambda_i v_i \tag{13}$$

where λ_i are the eigenvalue. By multiplying with A from the left,

$$AA^TAv_i = \lambda_i Av_i \tag{14}$$

Figure 6: The mean vector for *DB1*.

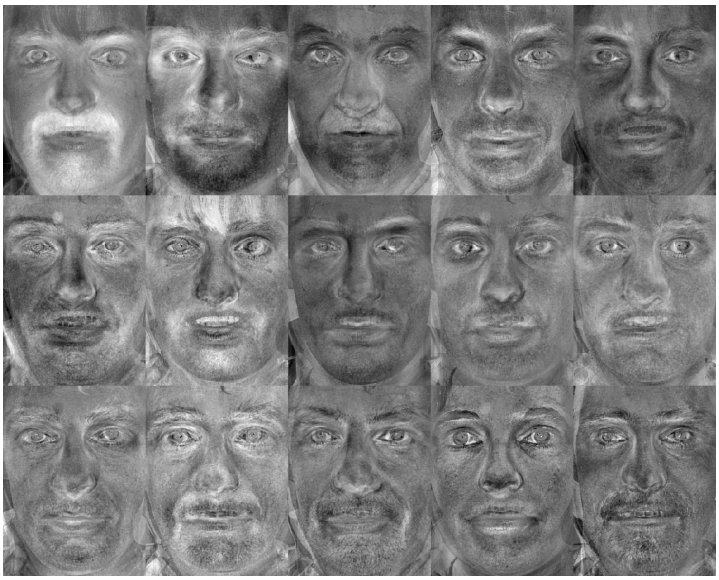


Figure 7: The eigenfaces from *DB1*

which is the eigenvalue equation for $C = AA^T$ where the eigenvectors are given by $u_i = Av_i$. In summary: the first M eigenvectors, $u_1, ..., u_M$, of C are given by

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

where v_i are the eigenvectors of $A^TA \in \mathbb{R}^{M\times M}$. Since M is significantly smaller than n, the task is then well within the bounds of what is feasible, and all useful eigenvectors are obtained.

Using k of these eigenvectors, u_i, a feature basis is formed—these basis vectors are the *eigenfaces* (albeit in a vector form). Figure 7 show the eigenfaces produced when trained on *DB1*.

Classifying a vector x is then simply a task of projecting it onto the feature space, according to

$$\Omega = U_k^T(x-\mu), \qquad U_k = [u_1, ..., u_k] \tag{16}$$

and determine the euclidean norm, ε_i, between Ω and vector number i among the known faces, which is calculated analogously to (16). The recognized face is then the face corresponding to the vector with lowest ε_i, unless ε_i is larger than a given threshold, then the face is classified as unknown.

3.10 Fisherfaces

When using multiple images of the same person to train a model, the class information should preferably be used to optimize the model, but the eigenface method does not take this information into account when spanning the basis vectors. To remedy these drawbacks of the Eigenfaces method, another approach based on Fisher's Linear Discriminant was developed—this method is appropriately called *Fisherfaces*. This method as described in [5] aims to find the matrix, $W_{\text{opt}} = [w_1, ..., w_m]$, of basis vectors, w_i, such that

$$W_{\text{opt}} = \arg\max_W \frac{\left|W^T S_B W\right|}{|W^T S_W W|} \tag{17}$$

where the between-class scatter matrix, S_B, and the within-class scatter matrix, S_W is defined by

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{18}$$

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T \tag{19}$$

where μ is the mean of all images, μ_i is the mean of person number i, x_k is the vectors of class X_i, c is the number of persons/classes and N_i is the number of images of person number i. This means that the basis vectors are in some sense spanned to optimize for distinguishing between different persons.

The solution to (17) is then given by the eigenvalue problem

$$S_B w_i = \lambda_i S_W w_i \tag{20}$$

where w_i are the columns of W_{opt}. Simliarily to the eigenfaces method, there is no more than $c-1$ useful eigenvectors.

The problem is that $S_W \in \mathbb{R}^{n\times n}$ is singular since $\text{rank}(S_W) \leq N - c$. By first using PCA as described in Section 3.9, an $(N-c)$-dimensional space, W_{pca}, is created whose within-class scatter matrix is not singular; (17) can then be applied to this data, giving W_{fld}. Finally W_{opt} is obtained by

$$W_{\text{opt}}^T = W_{\text{fld}}^T W_{\text{pca}}^T. \tag{21}$$

The columns of W_{opt} are visualized in Figure 8. Mathematically, W_{pca} and W_{fld} can be described as

$$W_{\text{pca}} = \arg\max_W \left|W^T S_T W\right|$$

$$W_{\text{fld}} = \arg\max_W \frac{\left|W^T W_{\text{pca}}^T S_B W_{\text{pca}} W\right|}{\left|W^T W_{\text{pca}}^T S_W W_{\text{pca}} W\right|}.$$

The classification is then performed in the same way as in the eigenfaces method except that the projection is given by

$$\Omega = W_{\text{opt}}^T x \tag{22}$$

where x is the vector to project.

4 Implementation

The implementation was done in MATLAB. The interface of the program is one function that takes an RGB-image as input and

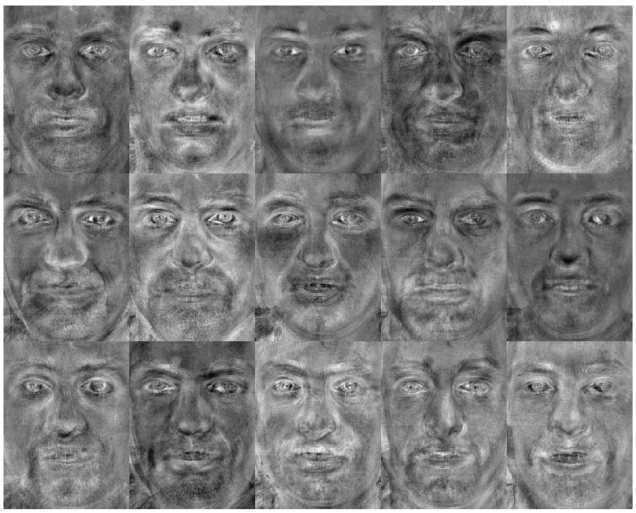
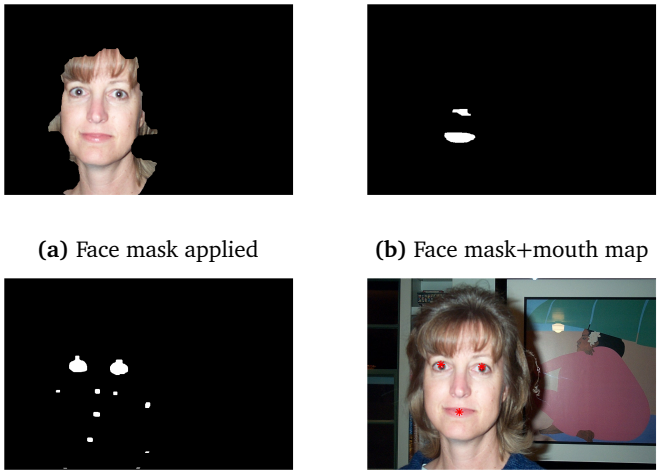


Figure 8: The fisherfaces produced from *DB1* and *DB2*



(a) Face mask applied **(b)** Face mask+mouth map

(c) Face mask+eye map **(d)** After using face triangle

Figure 9: Illustrates the process to extract the mouth and eye coordinates from image.

returns an id corresponding to the known faces; if the face is unknown, zero is returned. The final implementation uses the Fisherfaces approach described in Section 3.10.

4.1 Build the database with training data

The first step of the program is to build the database of Fisherfaces to be used for the recognition part. The images used for the construction needs to first be normalized in order to have homogeneous images. All images in *DB1* and *DB2* were used as training data.

The initial step of the normalization is to color correct the image. For this the gray world-assumption method was used. In order to detect the face in the image it tries to extract the mouth and eyes. The image is first converted to YC_bC_r to utilize the chrominance and luminance features of it. Then the face mask is applied together with the eye map and mouth map to filter out a lot of the background clutter and extract eye and mouth candidates. For more difficult images multiple candidates are found. To pick the correct ones, the face triangle method is used, described in section 3.7. The images in Figure 9 illustrates the result after applying the different masks. As can be seen in Figure 9b and 9c, it leaves multiple areas as potential candidates. That is why the face triangle method is vital to pick the correct ones.

When the eye and mouth coordinates are found the image is normalized around the face, described in 3.8.

To create the database, every normalized image is reshaped to a single column vector. Then they are all stored in a matrix where each column is a normalized image. This matrix is then used to perform the PCA and LDA algorithms to create the fisherfaces.

4.2 Recognition process

For recognizing a face, the program gets an RGB-image as input which goes through the same normalization process as the training data. The image is then projected onto the face subspace as described in Section 3.10. The error is then calculated by taking the difference between the projected image and each image in the database. The face is then classified as the id of the image with least error, as long as it is small enough.

Table 1: Face recognition performance for DB0, DB1, DB2 and modified versions of each DB1 and DB2.

DB	Correct	Accuracy	FR	FRR	FA	FAR
DB0	4/4	100%	0	0%	0	0%
DB1	16/16	100%	0	0%	0	0%
DB1'	434/464	93.53%	30	6.47%	0	0%
DB2	38/38	100%	0	0%	0	0%
DB2'	919/1102	83.39%	182	16.15%	1	0.09%

5 Results

The face recognition performance is presented in Table 1. The databases *DB1′* and *DB2′* contain modified versions of the base images in *DB1* and *DB2*. These modifications include rotation (±5°), scaling (±10%) and tone value changes (±30%). *DB0* is comprised of unknown faces and thus should all be identified as unknown.

6 Discussion

The images tagged with *il* in *DB2* proved difficult to identify. These images were captured with extreme lighting conditions such as heavy back light or overpowering camera flash. The gray world white balancing does not correct for extreme lighting conditions which means that the YC_bC_r luminance independence transformation (3) will distort skin color tones. This leaves the skin color tone values unrecognizable to the threshold ellipse (7) which prevents the construction of a face mask. Since facial feature identification depends on a decent face mask the facial recognition pipeline fails on the first step. To rectify this issue further pre-processing could be implemented in which the image is scanned for characteristics

of adverse lighting conditions and correct for them accordingly. Since white balancing does not work optimally for overexposed images, lighting correction would have to be applied before white balancing.

In the early stages of the project the way of obtaining coordinates for eyes and mouth was much more inaccurate and not as capable of discarding false positives. The main idea for this early prototype was to encapsulate a smaller region in the middle of the image, and only select eye candidates and a mouth candidate that exists within the selected region. The theory was that faces often covered the center part of the image. This did allow the faces within the *DB1* set to correctly obtain the coordinates for the mouth and both of the eyes. However, the region which was selected was determined by fixed pixel values that only worked as intended for the original image size. This meant that once features were added to make identifying faces in *DB2* better the static region selection to work on images in *DB1* proved to not be sufficient if accurate identification in *DB2* was going to be possible.

The primary problem with the prototype method was the image normalization that was introduced to ensure each image is standardized. This manipulated the sizes of the images hence the static pixel values for the region selection did not bear the same meaning. This made it possible and very likely that parts of face to be outside of the selected region making the method either not find eyes, find only one eye or give unreasonable matches for eyes that very clearly were not eyes. The same problem for the mouth was present where either no mouth was found or a false positive was found.

A complete overhaul to the method was necessary where the static selection part was removed and needed to be replaced with a more robust algorithm. This came to be mainly replaced by the presented face mask, face triangle and eye pupils methods.

As expected, the results for *DB1* and *DB2* are excellent, as these are the images used when training the model. To simulate differences, *DB1'* and *DB2'* was used for testing. To better evaluate the fisherface method a bigger set of images should be used, both for training and for testing. This would produce more "real-world" results, as opposed to the artificially modified images in *DB1'* and *DB2'*.

The results in table 1 shows that the implemented face recognition algorithm works relatively well. Because DB1′ and DB2′ have distortion applied to them, the face recognition accuracy represents a more realistic scenario where not all pictures are taken perfectly, e.g a face scanner security device to gain access to some sort of secret information. In this scenario, a false reject rate larger than false acceptance rate is undeniably the preferred outcome considering you would rather have the right people attempt to authenticate themselves multiple times rather than having the wrong person gaining access once. Fortunately, table 1 shows that $FRR \gg FAR$. Unfortunately, one image in DB′ was falsely accepted which could jeopardize secret information. This could be remedied by using a stricter recognition threshold to lower the false acceptances. However, this would come at the cost of accuracy and a larger false reject rate as well although this would be the preferred results in such a scenario.

References

[1] Caltech, *Computational vision: Archive*. [Online]. Available: http://www.vision.caltech.edu/html-files/archive.html.

[2] Rein-Lien Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, 2002. DOI: 10.1109/34.1000242.

[3] J. Su, *Illuminant estimation: Gray world*. [Online]. Available: https://web.stanford.edu/~sujason/ColorBalancing/grayworld.html.

[4] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991. DOI: http://dx.doi.org/10.1162/jocn.1991.3.1.71.

[5] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997. DOI: 10.1109/34.598228.