# Face detection and recognition algorithm in MATLAB

Course TNM034 - ’Advanced image processing’ at LiU

Samuel Svensson
samsv787@student.liu.se
Linköpings university
Norrköping, Sweden

Johan Forslund
johfo522@student.liu.se
Linköpings university
Norrköping, Sweden

Fredrik Johansson
frejo851@student.liu.se
Linköpings university
Norrköping, Sweden

Pontus Arnesson
ponar842@student.liu.se
Linköpings university
Norrköping, Sweden

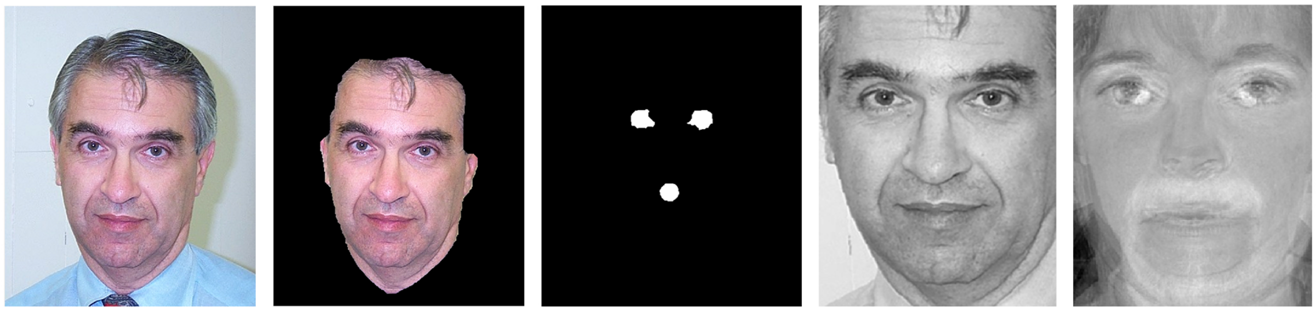

**Figure 1: Face recognition segmentation and normalization pipeline**

## ABSTRACT

This report describes the theory and process of implementing a face recognition algorithm using the computing software MATLAB. Several image processing techniques and morphological operations are used to detect and extract face features such as eigenfaces. This information is then used for matching faces against a given database. Alternative methods are also evaluated and discussed.

## 1 INTRODUCTION

The ability to detect familiar faces is a natural process for humans and usually happens instantly. The human face has a lot of interesting characteristics such as its ridge-like and downwards pointing nose, or the arrangement of the eyes, mouth and chin. Furthermore, a face can have a wide range of different types of facial expressions which can further complicate things when trying to detect faces computationally. Different emotions will result in a variety of options for where the eyes and mouths are actually located, which is an essential task in getting a reasonable result. But there are several factors that may interfere with recognizing a human face from an image. Squinting, or even closed eyes together with clenched lips might yield worse detection rates as well as facing away from the camera.

The ability to do this computationally has become an area of great interest because of its practical use-cases. Many people today have access to a smartphone which has face detecting capabilities to lock it. Because of this, unlocking it quickly becomes a necessity to prevent it from becoming a nuisance. Modern face detection algorithms have been studied by scientists and larger tech-corporations for years, and achieving similar results can be difficult.

In this report we propose a face recognition pipeline consisting of light compensation, normalization, extraction of eye and mouth maps and feature extraction for each image to detect if they match a correlating face in a given database.

This report is organized as follows: The aim of the project is described in section 2. section 3 describes illumination compensation, skin detection, eye/mouth mapping and eigenfaces. Results are presented in section 4. Section 5 discusses alternative methods and finally section 6 draws the conclusions.

## 2 AIM OF PROJECT

Using prior image processing and analysis knowledge, this project report aims to present the theory and implementation techniques used for a face detection algorithm. The aim is to be able to recognize a given image of a face and match it to a database and then return its corresponding identification number, if the face is present in the database. The algorithm should also be able to reject face candidates if no matching face is detected in the database.

## 3 METHOD

The method for implementing the proposed face detection algorithm is described in the following subsections. It is assumed that all images available for the project satisfy the following conditions:

- Each image only contain one face
- Both eyes and mouth are clearly visible

- The face has a maximum rotation of 10 degrees in relation to the cameras horizontal axis
- The image is not too over/underexposed such that pixel information is unusable. It should be possible to see a face in the image.

The images used during this project come from the data set 'Faces 1999' from Caltech which contains several images of a person under varying lighting and exposure conditions. [3].

## 3.1 Lighting compensation

Before extracting necessary data from a facial image it is necessary to normalize the image. The first step towards a normalized image is to compensate for bad lighting conditions. In this project *Reference white light compensation* were used which follows the assumption that each image contains at least a bit of "real white" in certain location such as the eyes. The image is then converted to the $YC_bC_r$ color space and the luminance (luma) channel is separated. If there are more than 100 pixels present in the top 5% lighting value in the luma they are set as a reference point. It then adjusts the individual R, G and B-channels such that the mean of the reference white pixels are linearly scaled to 255 (the maximum value).

## 3.2 Skin detection and face masking

The next step of the algorithm is to extract skin tones from the faces. With the skin tone a face mask can be constructed which corresponds to the outline of the face. The mask can then be used to remove insignificant background clutter which might complicate eye and mouth detection.

For skin color detection several different common color spaces can be used to correctly detect faces. In this report, the $YC_bC_r$ color space was chosen since it aims to simulate human vision and is based on opposite color theory (similarly to the HSV color space). By linearly transforming the RGB color space to $YC_bC_r$ and separating to individual luma and chrominance (chroma) channels it is possible to select pixels with values between specific color ranges. This is efficient since many research studies assume that the skin-tone colors of the chroma components are independent of the luma component [6][7][8]. The following chroma value ranges were selected and segmented from the original image.

$$77 < C_b < 200, \quad 134 < C_r < 173 \tag{1}$$

Where $C_b$ and $C_r$ are the normalized blue and red chroma channels respectively.

This operation leaves out the eyes and certain parts where the person's hair are blocking the skin. Therefore morphological opening and closing operations were applied to create a segmented binary image defining where the skin is located.

The binary image is then multiplied to the original image, resulting in a correctly masked image containing only the face. The whole process can be seen in Figure 2.

## 3.3 Eye map

By localizing the eyes from a facial image it is possible to determine if the image is slightly rotated or scaled. The proposed algorithm uses an illumination-based approach based on Hsu, Mottaleb and Jain's work at IEEE [2]. The method works by separating the blue

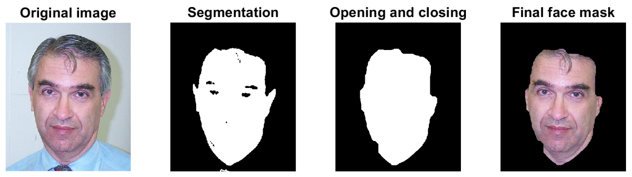

**Figure 2: Skin detection and face mask creation**

and red chroma channels based on the fact that the area around the eyes contain high $C_b$ and low $C_r$ values [4]. This is calculated as follows

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

where $C_b$ and $C_r$ are the normalized blue and red chroma channels respectively and $(\tilde{C_r})^2$ is the inverse of the red chroma channel $(1 - C_r)$.

It is possible to emphasize darker and brighter areas in the face with the help of the luma channel. Since the eyes are usually observed to be in a darker area it is possible to use the morphological operations erosion and dilation to construct another eye map seen below.

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1} \tag{3}$$

where $Y(x,y)$ is the luma component and $g_\sigma(x,y)$ is the structuring element used when dilating ($\oplus$) and eroding ($\ominus$).

Finally the eye map from both the chroma and luma components are combined into a single final thresholded eye map to receive a binary image mask.

$$EyeMap = EyeMapC \cdot EyeMapL \tag{4}$$

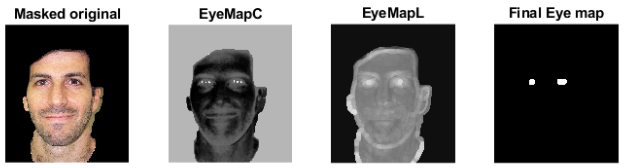

**Figure 3: Eye localization and extraction progress**

This operation might result in an image which contains more than two blobs, that ideally should represent the location of the eyes. Therefore, a set of rules are established to reduce the number of blobs that candidates as eyes. The rules regard the blobs orientation, ratio, solidity and position. Once these rules are applied, the blobs which does not follow the rules, are discarded.

## 3.4 Mouth map

With the assumption that the mouth generally corresponds to the chroma component $C_r$ being greater than the chroma component $C_b$, a mouth map is constructed. Furthermore, the mouth has a relatively low response in the $C_r/C_b$ feature and a high response in $C_r^2$ [1]. The mouth map is defined according to the equation seen in (5) using the help variable $\eta$ seen in (6).

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

$$\eta = 0.95 \cdot \frac{\frac{1}{n} \sum_{(x,y)\in FG} C_r(x,y)^2}{\frac{1}{n} \sum_{(x,y)\in FG} C_r(x,y)/C_b(x,y)} \tag{6}$$

Where $C_r^2$ and $C_r/C_b$ are normalized to the range [0,1], $n$ is the number of pixels within the face mask, $FG$. The parameter $\eta$ is estimated as the ratio of the average $C_r^2$ to the average $C_r/C_b$.

## 3.5 Face normalization

To achieve a sufficient comparison, the faces need to have a uniform size, position, rotation and color. With the combination of the eye map and the mouth map, a face normalization process was implemented.

As mentioned in 3.3 a set of rules help determine which blobs are chosen as eye candidates. After these rules are applied to the eye mask, the face normalization process determine the two most likely eye candidates. This is done by evaluating the best set of eyes for the mouth produced by the mouth map. The position and angle of the eyes relative to each other and the mouth are evaluated, and then the best pair is chosen.

The process constructs a triangle with corners in the center of these candidates. Depending on the rotation and position of the triangle, different operations were then used to align the face to the center of the image.

Sizing and cropping is also necessary to remove irrelevant image data such as background. Therefore, the alignment parameters were adjusted to keep as much facial data as possible as well as removing the irrelevant data. The face normalization process is illustrated in Figure 4.

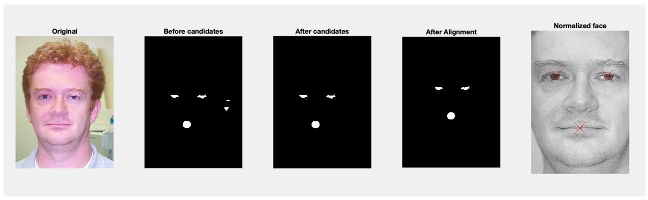

**Figure 4: The process of face normalization**

## 3.6 Eigenfaces

To apply face recognition to the normalized faces, we use eigenfaces. This method is widely used in image recognition and the term eigenfaces comes from the fact that they are composed of eigenvectors. Eigenfaces describe variance of faces in a set of face images, which is a useful metric when doing face recognition. The idea is to apply principal component analysis (PCA) to a large set of face images, which gives the eigenvectors of the covariance matrix that is built from the images. We can then reduce the space dimensionality by only considering the eigenvectors with the highest eigenvalues. This results in a basis of eigenfaces that together characterizes the variation among the face images, see Figure 5.

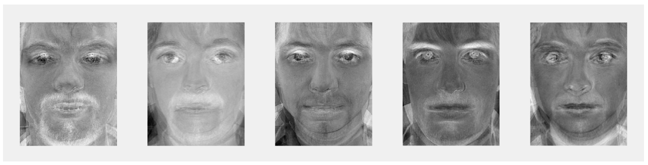

**Figure 5: Eigenfaces with the 5 largest eigenvalues**

The first step in creating the eigenfaces is computing the average face vector $\mu$ by summing the pixels in each face image vector $x$ and then dividing by the number of samples $M$ (7).

$$\mu = \frac{1}{M} \sum_{i=1}^{M} x_i \tag{7}$$

We are then subtracting the average face vector from every face vector (8) which represent the deviation from mean for every face vector.

$$\phi_i = x_i - \mu \tag{8}$$

Every $\phi_i$ is then placed column wise in a matrix $A$ which allows us to compute the covariance matrix $C$ as (9).

$$C = AA^T \tag{9}$$

To find the eigenvectors needed for PCA we have to compute the eigenvectors for $C$, which unfortunately has dimensions $nxn$, where $n = image\_width * image\_height$. This would be a much costly operation, even for relatively small images. Instead we use a shortcut to only compute the eigenvectors with most importance. This is done by switching the matrix multiplication order in (9) as (10).

$$C = A^T A \tag{10}$$

This will create a covariance matrix with dimension only as big as the number of faces used, where the eigenvectors easily can be computed. However, this will create eigenvectors with the same dimension, though we seek eigenvectors that match the dimension of the image vectors. We solve for the $i$ best eigenvectors $u_i$ from (9) by using Equation (11)

$$u_i = Av_i \tag{11}$$

where $v_i$ are the eigenvectors computed from (10). By reshaping the eigenvectors into the resolution of the original images we end up with the eigenfaces for the image set. By linearly weighting and combining the eigenfaces we can reconstruct face images, even faces that are not in the trained set can be approximated. The mentioned weights are used later when trying to match an input image with a face image from the training set.

## 3.7 Finding the correct face

The weights needed to reconstruct a specific face image is stored in a vector $w_j$, computed as Equation (12).

$$w_j = u_j^T \phi_i \tag{12}$$

Each face image in the training set will get assigned a weight vector. The same is valid for the input image that we want to perform face recognition on. To find the most likely match, we choose the face image from the training set that has a weight vector with least euclidean distance from the input image weight vector. However, care has to be taken for images outside of the training set since they should be returned as not found. Therefore, a maximum allowed distance from any image had to be added in order to achieve a match.

#### *3.7.1 Reliability*

There are two main approaches for the reliability of a face recognition program. The first approach is to accept a greater amount of false positives (FAR) for an increased rate in face detection in comparison to the second approach which rejects false positives for increased security (FRR). The first approach tends to result in a higher detection rate, but at the cost of security. During this project the first approach was implemented, but it could easily be changed by adjusting threshold parameters.

## 4 RESULTS

To test our implementation for recognizing faces from facial images a testing program was written which would iterate through every image and return its corresponding identification number. The results would then be compared to the actual correct answer and output the percent of correctly matched faces.

The database of facial images from Caltech (faces 1999) was used for this project. The database was divided in four parts described in Table 1 below.

**Table 1: Database specifications**

| Database | No. of images | Comments |
|---|---|---|
| DB0 | 4 | Images not in database |
| DB1 | 16 | One image per person in good conditions |
| DB1a | 16 | Rotated (max 10 degrees) and scaled images from DB1 |
| DB2 | 38 | Several images per person with varying lighting and color conditions |
| DB3 | 96 | Same as above but more images per person (including DB1 persons) |
| DB4 | 450 | Same as above but more images |

The algorithm was trained on DB4 and the resulting weights were stored in .mat files. These weights were then used for additional training and testing on different databases. See the results in table 2 below.

**Table 2: Test results where DB1 and DB3 uses 100% of the eigenfaces and DB4 uses 80% of the eigenfaces.**

| Training Data | Test Data | Accuracy |
|---|---|---|
| DB1 | DB3 | 51% |
| DB3 | DB1 | 100% |
| DB3 | DB1a | 100% |
| DB4 | DB1 | 94% |
| DB4 | DB3 | 97% |

The number of eigenfaces that were used during classification had an effect on the detection accuracy. Figure 6 shows the results for different amount of eigenfaces. For this particular test, DB4 was used to train and DB1 was used to test.

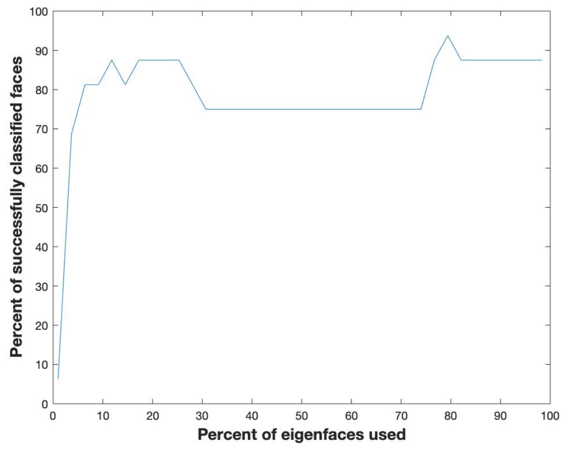

**Figure 6: Detection accuracy using different percentages of total eigenfaces**

## 5 DISCUSSION

The results are based on the training data originating from DB4 which contains a single image per person with forgiving lighting conditions and backgrounds. Due to this there are several factors that can affect the result. It is therefore necessary to discuss alternative methods and ways of countering false positives to obtain optimal results.

## 5.1 Improvements

#### *5.1.1 Light compensation*

There are several different light compensation techniques that can be used for a task like face recognition. The one used in this report is, as stated in 3.1, reference white light compensation. Another appropriate method that could have been used is Gray world light compensation. This method assumes that the average intensity of the R, G and B channels should be equal, and there by scale the channels accordingly. This gives a much more gray image, and can in some cases be more efficient. The downside of this is that an image with a large coloured background would scale the channels in an unwanted manner.

### 5.1.2 *Skin detection and face masking*

The face mask is an important feature in images that for example have multiple faces or are harder to detect in general. In this case, only images are used that contain exactly one face. These images are also within a certain range regarding scale and rotation, which makes the face mask less important.

As mentioned in 3.2, only the $YC_bC_r$ color space is used to detect the skin tone in the color images. However, the HSV color space was also tested to mask out the face, both by itself and combined with $YC_bC_r$. The results were worse in both cases and were therefore discarded as a measure of detecting skin color when using the light compensation at hand.

### 5.1.3 *Eye map*

As mentioned in 3.3 the eye map method uses an illumination-based approach. The paper by Hsu, Mottaleb and Jain's [2] also propose two additional methods; Color-based and edge density based. The color-based method involves taking the original color image and converting it to a histogram equalized grey-level image. It is then thresholded to extract the darkest regions of the image, and if unwanted regions are present a component verification process is used. This is based on the observation that the eyes often are the darkest regions of a face. This approach works well for badly illuminated images as well as images where the person has closed eyes, but not very well for dark skin colors.

The edge density-based approach uses a Sobel edge kernel to process the image and receive a sharp edge mask. It is based on the observation that the eye area has higher edge density than other regions of the face. Two erosions followed by three dilations are then performed on the mask to further enhance the eye map. Hsu proposes a hybrid method where the three detection methods are combined into a single eye map that is defined only where at least two of the methods have detected an eye.

We have experimented with the described hybrid method in this project. However, it was discovered that both the color based and the edge density based method gave poor results when trying to detect the eyes. A reason for why the color based method gave poor results might be the fact that the images that Hsu used in their study had a different lighting setting with higher contrast. For the images used in this report, the eyes could not be determined as the darkest regions. When trying to lower the threshold further, it gave too much noise from other regions of the face. Regarding the edge based method, it was difficult (if not impossible) to find a structuring element for the morphological operations that captured the eyes without including other regions found by the Sobel kernel. An idea for improving this would be to replace the Sobel kernel with some other method to detect the eyes, possibly by using circular Hough transform.

### 5.1.4 *Face normalization*

Since the mouth map in most cases produce only one mouth candidate, the first one found is always chosen. This is done to make the face normalization faster and more stable, since the eye map is more often incorrect than the mouth map. However, this may lead to the wrong mouth being chosen. This results in a skewed face triangle in some cases.

Aside from choosing the wrong mouth in the mouth map, the face normalization works well, given at least two correct eye candidates. If the face mask or eye map doesn't work properly either blobs mistaken for eyes will be selected, or the image will not be included in the calculations.

If no correct eyes are produced from the eye map and faulty ones are chosen instead, the normalized face is going to be incorrect and create errors in the recognition process.

### 5.1.5 *Eigenfaces*

The method of creating eigenfaces worked well for this set of images. It was relatively easy to detect faces and crop them out which made it straight forward to then create the eigenfaces. This would have been much harder if the images were not as simple, for example if images contained multiple people or covered faces.

The number of eigenfaces used during the classification had an affect on the accuracy, though not as much as one could think. Figure 6 shows that the accuracy is relatively stable independent on the number of eigenfaces used. However, if we choose to few (< 10%) eigenfaces, the accuracy drops significantly as the eigenfaces can no longer represent the faces in a correct manner. At around 80% the plot reaches a peak where the accuracy is maximum.

We had some problem at first to reconstruct the original images using the mean image added to a linear combination of the weights. The problem was that the weight contribution had much bigger values which totally dominated the result. This was solved by normalizing each eigenface vector before applying the weights, something that was not mentioned in most of the reports on this subject.

### 5.1.6 *LDA and Fisherfaces*

As opposed to the unsupervised PCA method, it is possible to use the supervised Linear Discriminant Analysis (LDA) which makes use of both data and class information, which is very effective when comparing variance in classes. This is the main difference between PCA and LDA. They both work by looking for linear combinations of variables which best explain the given data. LDA is a generalization of Fisher's linear discriminant (FLD) which aims to maximize inter-class variance and minimize intra-class variance. A class is composed of several images with varying conditions but of the same person/identity. FLD tries to "shape" the scatter to make it more reliable for classification and recognizing whose face is on the image [9]. The within-class (intra-class) scatter matrix is defined as

$$S_w = \sum_{j=1}^{C} \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T \tag{13}$$

where $C$ is the number of sample images, $i$ is the $i$th sample of every class, $\mu_j$ is the mean of the class $j$, and $n_j$ is the number of classes $j$.

Furthermore, the between-class (inter-class) is defined as

$$S_b = \sum_{j=1}^{C} (\mu_j - \mu)(\mu_j - \mu)^T \tag{14}$$

where $\mu$ represents the mean of all classes.

As described in the paper by P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman [9] it is possible to combine Eigenfaces and PCA together with LDA to compute class-representative Fisherfaces using the following methodology

- Reshape every training image to a vector and organize every vector into a matrix.
- Use PCA on the matrix to obtain eigenfaces.
- Use the eigenfaces and project them into PCA subspace to receive reduced dimensionality feature vectors.
- Perform LDA on these vectors to obtain fisherfaces which represent each class. Maximize inter-class variance and minimize intra-class variance. Use these matrices to solve for eigenvalues and calculate fisherfaces.
- Use fisherfaces to project **training images** into LDA subspace to receive reduced dimensionality feature training vectors.
- Use fisherfaces to project **test images** into LDA subspace to receive reduced dimensionality feature test vectors.
- Use KNN (or any other classifier) using both training and testing feature vectors

This might have been a more direct way to combine both methods and may have yielded better results. We tried this approach but due to time constraints and undesirable results it was decided to simply use the eigenfaces method instead.

## 5.2 Different approaches

The approach used in the project is from an image processing standpoint. There is worth to mention that there are machine learning methods available for face detection and recognition. However, these methods have some drawbacks compared to the image processing method. The machine learning approach usually requires a larger amount of data and training phase. There are also several different methods of normalizing illumination in images, such as the LogAbout method.

### *5.2.1 SQI and LogAbout*

To even out contrast levels in all images, histogram equalization was used. This is the most common illumination normalization approach but it is possible to use illumination-free methods like Self Quotient Image (SQI) and LogAbout. SQI is composition of the reflectance and the lighting of the scene. Furthermore, the lighting can be estimated using a low-pass filter since it is seen as a low frequency component of the image [5].

LogAbout is a technique where you apply a high-pass filter to an image followed by a logarithmic transformation on it

$$g(x, y) = a + \frac{ln(f(x, y) + 1)}{b \cdot ln \cdot c} \tag{15}$$

where $f(x, y)$ is the original image together with $a$, $b$ and $c$ which are parameters that determine the location and shape of the logarithmic transformation.

You can then use both methods by processing the SQI output through a high-pass filter and mapping it using equation. The fusion between SQI and LogAbout was done by using equation 15.

LDA is then used to extract feature vectors from each face and the Euclidean distance is measured, very similarly as described in chapter 3.7.

## 6 CONCLUSION

This project proves that it is doable to construct a face recognition pipeline in a short amount of time with some prior image processing knowledge. The pipeline is general and abstract enough for further development in the form of new features to expand the capabilities of the program.

Designing an efficient face detection algorithm can be very difficult since there are many different factors and requirements of the image for it to successfully detect a face. Multiple faces, closed eyes, tilted head or bad lighting conditions can all contribute to making it more difficult when detecting faces. The use of hybrid methods based on multiple scientific papers compared to single papers tends to increase the robustness of the program but can lead to drawbacks as well. There are also certain normalization methods which tend to favor specific image conditions. Recently, machine-learning based algorithms and methods have been successful in facial detection in images and may pave the way for new opportunities in correctly classifying faces using relevant training data.

## REFERENCES

[1] Rein-Lien Hsu, *Face Detection and Modeling for Recognition* , Michigan State University Department of Computer Science & Engineering , 2002
[2] Hsu, R.L., Abdel-Mottaleb, M. & Jain, A. 2002, *Face detection in color images*, IEEE Transactions on Pattern Analysis and Machine Intelligence , vol. 24, no. 5, pp. 696-706.
[3] Caltech, 2019 California Institute of Technology, hämtad: 2019-12-05 http://www.vision.caltech.edu/archive.html
[4] Muhammad Shafi and Paul W. H. Chung, *A Hybrid Method for Eyes Detection in Facial Images*, IN: Proceedings of World Academy of Science, Engi- neering and Technology International Conference on Computer Science Singa- pore, 32, pp. 99 - 104
[5] Mohammad Shazri and Chai Tong Yuen , *LogAbout Mapping of Self Quotient Image* , Department of Research & Development and Department of Mechatronic & Biomedical Engineering at Universiti Tunku Abdul Rahman (UTAR), Kuala Lumpur, Malaysia
[6] E. Saber and A.M. Tekalp,*Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions*, Pattern Recognition Letters, vol. 19, no. 8, pp. 669–680, June 1998.
[7] M.J. Jones and J.M. Rehg,*Statistical color models with application to skin detection*, Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition, vol. 1, pp. 274–280, June 1999.
[8] B. Menser and M. Brunig, *Locating human faces in color images with complex background*, Intelligent Signal Processing and Communications Systems, pp. 533–536, Dec. 1999.
[9] Peter N. Belhumeur, Jo ao P. Hespanha, and David J. *Kriegman Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection* IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 7, JULY 1997 pp. 714-715