Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor Thesis

Face recognition using Haar Cascades and Eigenfaces

Denis Kotsyuk

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Denis Kotsyuk

Systems Engineering and Informatics
Informatics

Thesis title

Face recognition using Haar Cascades and Eigenfaces

Objectives of thesis

The objectives of the thesis are:

To describe and demonstrate how the face detection and face recognition work, using haar-like features and eigenfaces method.

To develop an application which detects and recognizes human faces using these methods.

To calculate and compare the relative success rates of face recognition. The influence of additional face features will be assessed too.

Methodology

The methodology of the thesis is based on analysis of technical and scientific sources related to face detection and face recognition, in particular haar-like features and eigenfaces. Based on the synthesis of the gained knowledge, a C# application prototype will be implemented using Emgu CV libraries, in order to collect and compute data derived from images of human faces. A comparative analysis based on the relative success rates will be conducted to identify differences between face recognition algorithms. The influence of additional face features (e.g. glasses, facial hair) on algorithm performance will be evaluated.

The proposed extent of the thesis

40-50 pages

Keywords

Face recognition, Haar Cascades, Eigenfaces

Recommended information sources

LÉLIS BAGGIO, Daniel, Emami SHERVIN, David MILLÁN ESCRIVÁ, Ievgen KHVEDCHENIA, Mahmood NAUREEN, Saragih JASONL a Shilkrot ROY. Mastering OpenCV with Practical Computer Vision Projects. 1. Birmingham: Packt Publishing, 2012. ISBN 9781849517829.
SHI, Shin. Emgu CV Essentials. 1. Birmingham: Packt Publishing, 2013. ISBN 9781783559527.
TANAKA, James, ed. Face Recognition: The Effects of Race, Gender, Age and Species. 1. New York: Taylor&Francis, 2015. ISBN 978-1-317-65095-9.

Expected date of thesis defence

2019/20 WS – FEM (February 2020)

The Bachelor Thesis Supervisor

Ing. Petr Hanzlík, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 25. 11. 2019	Electronic approval: 25. 11. 2019
Ing. Martin Pelikán, Ph.D.	**Ing. Martin Pelikán, Ph.D.**
Head of department	Dean

Prague on 18. 11. 2020

Declaration

I declare that I have worked on my bachelor thesis titled "Face recognition using Haar Cascades and Eigenfaces" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 30.11.2020 ______________________________

Acknowledgement

I would like to thank Ing. Petr Hanzlík, Ph.D., for his advice and support during my work on this thesis.

Face recognition using Haar Cascades and Eigenfaces

Abstract

This thesis focuses on one of the most popular robust face detection methods based on the Viola and Jones framework, that is used on frontal face images, and the face recognition algorithm based on the Principal Component Analysis using the Eigenface method. The theory of these methods and the steps for creating an application that implements these methods are explained. The design of a face recognition system is presented, including image acquisition, grayscale conversion, face detection, face extraction, application of Principal Component Analysis, and face recognition. The process of fine-tuning parameters of the face detector to improve detection accuracy is described and the optimal values of these parameters are selected to minimize the influence of the detection algorithm on the performance of the face recognition algorithm.

As a result, a desktop application is designed for face recognition that allows to measure the performance of the model. The classification performance of resulting model is assessed with the confusion matrix and studied in relationship with gender and partial facial occlusions such as glasses and beards.

Keywords: Face detection, face recognition, Viola-Jones framework, Principal Component Analysis, Eigenface method, C#, EmguCV.

Rozpoznávání tváře pomocí Haarových kaskád a metody Eigenface

Abstrakt

Tato práce se zaměřuje na jednu z nejpopulárnějších metod detekce obličeje založených na rámci Violy a Jonese, která se používá pro obrázky tváře z čelního pohledu, a algoritmus rozpoznávání obličeje založený na analýze hlavních komponent pomocí metody Eigenface. Je vysvětlena teorie těchto metod a kroky pro vytvoření aplikace, která tyto metody implementuje. Je představen návrh systému pro rozpoznání obličeje, včetně pořizování obrázků, převod barevné fotografie do stupňů šedi, detekce obličeje, extrakce obličeje, aplikaci analýzy hlavních komponent, a výsledného rozpoznání obličeje. Je popsán proces nastavení parametrů detektoru obličeje pro zlepšení přesnosti detekce a jsou vybrány optimální hodnoty těchto parametrů tak, aby se minimalizoval vliv detekčního algoritmu na výkon algoritmu rozpoznávání obličeje.

Výsledkem je desktopové aplikace pro rozpoznávání tváře, která umožňuje měřit výkon modelu. Přesnost klasifikace výsledného modelu je posouzena pomocí matice záměny a studována ve vztahu k pohlaví a částečným okluzím obličeje, jako jsou brýle a vousy.

Klíčová slova: Detekce obličeje, rozpoznávání obličeje, rámec Viola-Jones, analýza hlavních komponent, metoda Eigenface, C #, EmguCV.

Table of content

List of pictures

List of tables

List of abbreviations

EmguCV – a wrapper to the OpenCV image processing library.
OpenCV – Open Source Computer Vision Library.
PAC – Probably Approximately Correct.
PC – Principal Component.
PCA – Principal Component Analysis.
RGB – Color model (Red, Green, Blue).
BGR – Color model (Blue, Green, Red).
JPEG – an image file format of the Joint Photographic Experts Group.
YML – a file extension of the YAML Ain't Markup Language.
XML – Extensible Markup Language.

1 Introduction

Human face detection and face recognition are very fascinating aspects of computer vision field that are in high demand these days. Even though the research in this area can be traced back to 1960s, the achieved results were far from satisfactory until only recently. The problem of machine vision has proved to be more complex than originally assumed and is still not solved to this day.

„The first great revelation was that the problems are difficult. Of course, these days this fact is a commonplace. But in the 1960s almost no one realized that machine vision was difficult.“ (1)

Nevertheless, a rapid progress has been made in the field of face recognition in recent years. Due to the fast and steady increase in computational power of the machines - the machine learning processes required less time for training and evaluating face recognition models. This contributed to the rate of improvements made in the field. Additionally, a global rapid spread of the Internet expanded the collaboration between researchers around the world and provided the venue where multiple image databases could be collected and shared.

Face recognition finds a large variety of applications in the real world. For example, it is used in law enforcement activities, security surveillance services, biometric identification, social media and social networking services such as Facebook, e-commerce, mobile applications. In future the face recognition systems could also be implemented in banking for account withdrawals or even for registering attendance at workplaces, schools, or universities.

There exist many face detection and face recognition algorithms and techniques, and depending on the different conditions and requirements, some may be more suited for a particular task than others. This thesis covers a robust face detection algorithm based on the Viola-Jones framework and a face recognition algorithm based on the Eigenface method of applying Principal Component Analysis to images of faces in frontal view.

2 Objectives and Methodology

2.1 Objectives

The main goal of the thesis is to describe and demonstrate how the face detection and the face recognition work, using Haar-like features within the Viola-Jones framework and the Eigenface method of applying Principal Component Analysis to images. The next goal is to develop an application which detects and recognizes human faces using these methods.
The final goal is to calculate and compare the relative success rates of the face recognition model. The influence of additional face features will be assessed too.

2.2 Methodology

The methodology of the thesis is based on analysis of technical and scientific sources related to face detection and face recognition, in particular Haar-like features and Eigenfaces. Based on the synthesis of the gained knowledge, a C# application will be implemented using Emgu CV libraries, in order to collect and compute data derived from images of human faces. A comparative analysis based on the relative success rates will be conducted to identify differences between face recognition settings. The influence of additional face features (e.g. glasses, facial hair) on algorithm performance will be evaluated.

3 Literature Review

3.1 Computer Vision

How do you teach a machine to “see”, to have a “vision” on the human level? How can a machine learn to “understand” images? Such questions have been in computer scientist’s minds for decades and figuring out answers to these questions required vast knowledge collected from multiple scientific fields and an enormous amount of work put together.
Computer vision is a discipline that intersects with multiple other scientific fields in pursuit of integration and automation of a broad collection of processes used for vision perception. The aim of computer vision is to construct valid, explicit descriptions of physical objects derived from the images. Such image understanding should not be confused with image processing, which is all about transformations of images into images. (2) But image processing is often a useful complement for image understanding (3).
Typical tasks of computer vision include processes of acquiring data from the images, analysing the gathered data and categorizing it. One of applications of computer vision is face recognition.

3.2 Face Detection

Face detection is the starting point of any type of a facial recognition system and its success is crucial to the face recognition processes and results. To be able to recognize faces, we must first find them. Face detection algorithms are therefore designed to locate human faces in images.
Techniques to detect faces from a single image can be classified into four categories suggested by Yang, Kriegman and Ahuja (4):

1. **Knowledge-based methods.** Such methods rely on the set of rules, which are derived from a human knowledge of detecting faces, where facial features have certain relationships with each other, based on the symmetry, position and distance. The drawback of this approach is the difficulty of composing such set of rules. If the rules are too generalized or too strict - both these factors may

lead to a lower percentage of detection rate, so it is not so easy to find the right balance. Additionally, this approach does not do well if the face poses change, since it so heavily relies on the mentioned feature relationship measurements.

2. **Feature invariant approaches.** These methods are based on the extraction of facial features like lips, nose, eyebrows, eyes and building a statistical model describing their relationships in different lighting conditions and poses. Such observations become a training set of data for the classifiers, which then determine the presence of a face in an input image. This approach is based on the idea of moving away from a human knowledge of faces and human assumptions. Factors like illumination and shadows can affect the reliability of these methods.

3. **Template matching methods.** This approach is based on the creation of the predefined templates, which are then compared to input images. Templates are created from a set of subtemplates of facial features like nose, eyes, mouth or face outlines. Such templates are defined by functions and face detection is done through correlation between new input images and predefined templates.

4. **Appearance-based methods.** Where template matching methods are predefined by specialists through functions, the appearance-based methods let the machine learning figure out itself what features, and elements combined form the human face along with the use of statistical analysis. The appearance-based methods are the best choice for human face detection in present days, they are showing the best results in detection rate.

3.2.1 Viola-Jones framework

For the purposes of this work the frontal face detection framework proposed in 2001 by Paul Viola and Michael Jones (5) has been selected. The framework provides high detection rates and processing speed, which enables it to detect faces in real-time from video sources or multiple input images very fast (6). Although training of the classifiers takes a long time, after such task is complete - these classifiers can be reused in other new applications, created by anyone else, greatly decreasing the overall computational time of face detection programs. Therefore, this algorithm is well liked and favoured among many other algorithms.

The Viola-Jones framework is comprised of the following four main components:

- Haar-like features
- Integral image
- Adaboost
- Cascade of classifiers

3.2.2 Haar-like Features

Human faces have eyes, eyebrows, lips, nose, and edges that are generally darker or lighter in comparison to the surrounding skin under some source of light due to our face structure. The Viola-Jones framework uses so called Haar-like features of rectangular shapes, consisting of black and white regions, in order to determine the presence of the facial features. The main principle is based on adding and subtracting pixel intensities of different adjacent regions of the input image (previously converted to a greyscale image) and comparing the results with ideal values taken from the pre-set Haar-like features (7).

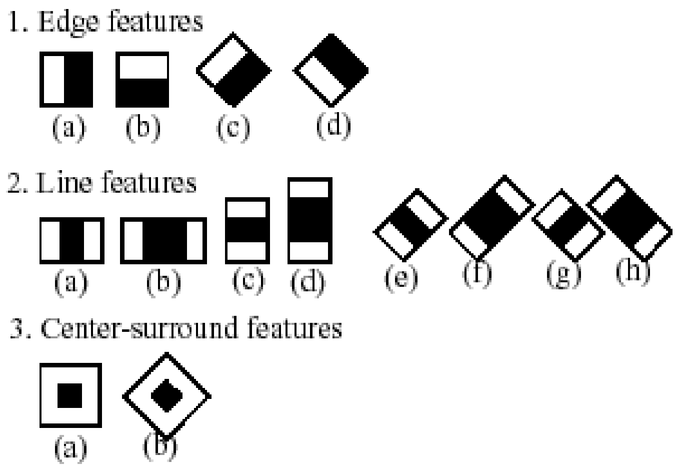
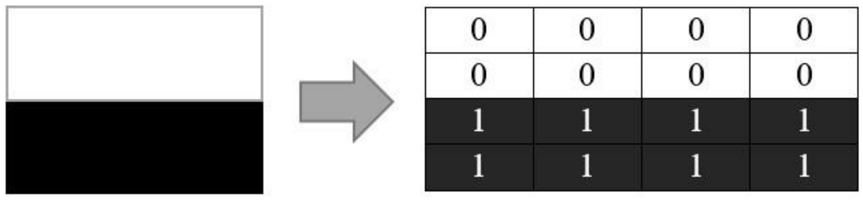


Figure 1 - Types of Haar-like features (8).

Let us use an example from Figure 1, more specifically the Edge feature of type (b) - it consists of white and black vertical regions. We can represent these regions by pixel intensity values, where each pixel intensity value of the black region is set to 1 and each pixel intensity value of the white region is set to 0. Therefore, this Haar-like feature can be expressed by an array of numbers, where each element of the array represents a pixel density value, as shown in Figure 2.



0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1

Figure 2 - Example of a Haar-like feature pixel intensities(authors work)

The value of the feature is then calculated as a difference between the averages of the black and white regions:

$$\Delta = \frac{1}{n} \sum_{dark\ region}^{n} I(x) - \frac{1}{n} \sum_{light\ region}^{n} I(x)$$

where $0 < \Delta < 1$.

In an ideal situation the value of a facial feature equals to 1 ($\Delta = 1$). In real life though, we can’t get such ideal results from the input images of human faces, after they are converted to grayscale images. Grayscale images can still have up to 256 shades of gray. Therefore, with real images, the closer the feature value is to 1 - the more likely it is a facial feature, and on the other hand the closer the value of the feature is to 0 - the less likely it is a facial feature. To accept any feature as a facial feature the value of this feature would have to exceed a certain pre-set threshold.

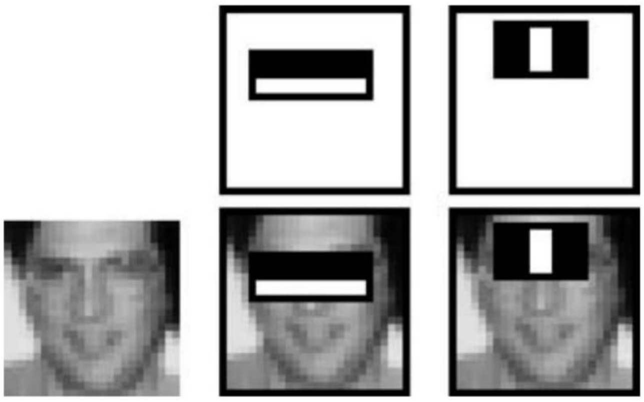


Figure 3 - Example of overlaying Haar-features on the face image (5)

The base resolution of the detector is 24 x 24 pixels and the amount of all possible features in an input image within the detector sub-window is 160000 (7). This is because areas of different sizes are evaluated all around the image, shifting through the image pixel by pixel. Such computational processes are lengthy and if we want to use images of much higher resolutions - it would take considerably longer time to compute the values of all features in an image.

In order to reduce the heaviness of the computations - the use of intermediate representations for the image called the integral image are implemented (7).

3.2.3 Integral Image

The calculation speed of the values of all features for the purposes of detection of the facial features can be improved by using an algorithm called **integral image**, which was introduced by Paul Viola and Michael Jones in 2001 (5). The concept of the integral image is very similar to an algorithm of using a table of summed up areas, described in an earlier work of Franklin Crow on texture mapping in 1984 (9). The summed-area table (or the integral image) is constructed as a representation of the original grayscale image, where the value of any chosen point ***(x, y)*** of the summed-area table is the sum of all pixel intensity values to the left and above of the point ***(x, y)*** of the original image, inclusively:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Where ***i(x, y)*** is the original grayscale image and ***ii(x, y)*** is the integral image.

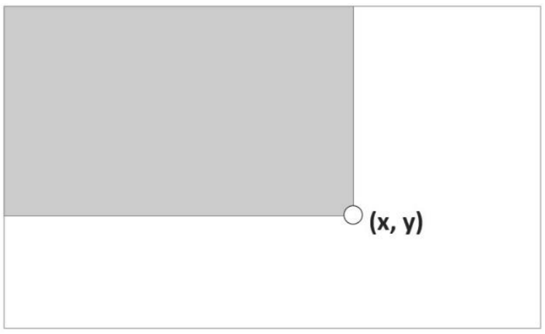


Figure 4 - Integral image as sum of pixels to the left and above of point (x, y) (7).

The reasoning for creating such a representation of an image, is that it is calculated only once and then with its help we can get the sum of pixel intensities of any desired rectangular area within an image in few easy steps. As shown in Figure 5, if our rectangular sum of interest is D, then it is calculated by using the values at locations 1, 2, 3, 4 of the integral image. Then the rectangular sum of D is simply: 4 + 1 – 2 – 3.

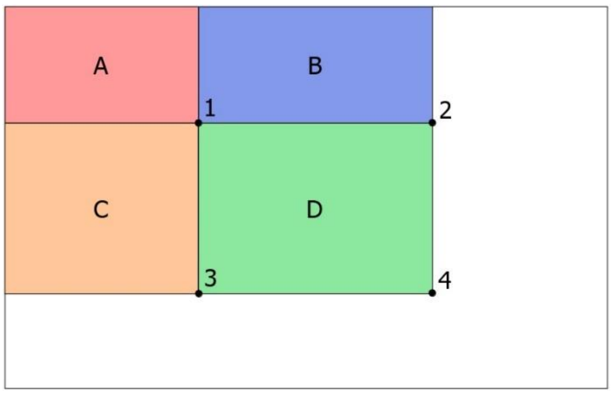


Figure 5 - Example of calculating the sum of pixels withing the chosen rectangle (7).

This approach enables us to retrieve the sums of lighter and darker regions of the image much quicker and more efficiently. Therefore, the integral image simplifies the calculation of all feature values and thus reduces the overall computation time. Unfortunately, the amount of all possible features evaluated for each detector sub-window of the image remains the same and it is rather large, as mentioned earlier it exceeds 160000. Additionally, many of these identified features may not be much relevant at all in determining a presence of a face. But the cascade of classifiers, implemented by Viola and Jones, using the Adaboost algorithm - solved the problem of excessiveness of the features and at the same time increased the performance and accuracy of simple learning algorithms by combining them together.

3.2.4 PAC learning model

In 1984, Valiant (10) introduced a computational, distribution-free learning model, known as ***PAC*** (***Probably Approximately Correct***), which became fundamental for many further studies of the computational learning models and for creations of new learning algorithms. In his work he described “learning” as an outside area of an explicit form of programming, where an algorithm itself acquires skills to perform certain tasks without the human influence by direct instructions. In this learning model, the ability to identify a previously unknown ***concept***, otherwise known as a target ***rule***, is confined to realization whether this concept is true or false for given set of examples.

Such classification that contains only two elements is binary, thus:

$$c = \{1, 0\}, c \in C,$$

where ***c*** is the classification of a target concept and ***C*** is a classification set.

The goal of this learning method is in figuring out a hypothesis that would correctly classify new inputs as negative or positive examples of the concept (11).

If an input example $\boldsymbol{x}$ is a positive example of the concept $\boldsymbol{c}$, then:

$$\mathrm{c(x) = 1.}$$

Otherwise, if

$$\mathrm{c(x) = 0,}$$

then it corresponds to $\boldsymbol{x}$ being a negative example of the concept.

A sequence of ***N*** training examples $\boldsymbol{x}$ is picked randomly from the domain $\boldsymbol{X}$, according to unknown fixed distribution $\boldsymbol{M}$. Therefore, the training set is formed as:

$$(x_1, c(x_1)), \dots, (x_N, c(x_N))\ , where\ x_i \in X, c(x_i) \in C, for\ i = 1,2, \dots, N.$$

The learning algorithm then returns a hypothesis, such as

$$h{:}\, X \to [0,1], h \in H,$$

where ***H*** is a hypothesis space, and ***h(x)*** is the random prediction of the binary classification of $\boldsymbol{x}$ to be equal to 1, thus $\boldsymbol{1 - h(x)}$ is the prediction of $\boldsymbol{x}$ to be equal to 0 (12).

Given a set of inputs $\boldsymbol{x_i}$ the ***learner*** returns random classification predictions $\boldsymbol{h(x_i)}$ of these training examples:

$$(x_1, h(x_1)), \dots, (x_N, h(x_N))\ , where\ x_i \in X, h(x_i) \in H, for\ i = 1,2, \dots, N.$$

The ***error*** of the hypothesis is then calculated by:

$$error(h) = E_{x_i \sim M}(|h(x_i) - c(x_i)|),$$

where $\boldsymbol{x_i}$ is selected from the distribution ***M***, and ***E*** is the expected value.

According to Kearns and Vazirani, the PAC model is then considered to be ***learnable*** if the following conditions are satisfied (13):

- Given constant inputs $\boldsymbol{\varepsilon}$ and $\boldsymbol{\delta}$ such as $0 < \varepsilon, \delta < 1/2$, where $\boldsymbol{\varepsilon}$ is called an error parameter and $\boldsymbol{\delta}$ is called the confidence parameter.

- If $\varepsilon \geq error(h)$.

- If a learning algorithm provides a hypothesis $\boldsymbol{h} \in \boldsymbol{C}$ with probability at least $\mathbf{1} - \boldsymbol{\delta}$.

From stated conditions above, it can be noticed that the error of the hypothesis should be below 50%, and that a learning algorithm must provide a hypothesis with probability above 50%. Such PAC learning model is of ***weak learnability***.

3.2.5 Weak and strong learnability

According to Kearns (14), a learning model where a hypothesis performs slightly better than random guessing is referred to as a model of ***weak learnability***. In other words, if a program is guessing between only two options, whether the concept is true or false, then it is no better than a flip of a coin and has a 50% chance to classify the concept correctly. The weak learning algorithm, therefore, tries to improve the hypothesis, so that the correctness of guessing is just slightly better than 50%. On the other hand, a model of ***strong learnability*** is a model that results in high accuracy of correct classification of the concept.

In 1990, Robert Schapire (11) proved the equivalence of weak and strong learnability and introduced a ***hypothesis boosting*** method, which showed how to turn a single ***weak*** learning algorithm into a ***strong*** learning algorithm, hence improving its accuracy by making alterations to a distribution of examples in order to make the learning algorithm focus on the most difficult to classify examples of the training data, thus producing a hypothesis that is making least errors for those hard examples. This approach allowed predictions of new examples of a concept to be more accurate. During the training process, the configuration of a hypothesis did not rely on simple examples.

3.2.6 Classification using AdaBoost

Few years later, Freund and Schapire introduced a new and improved boosting algorithm called ***AdaBoost*** or ***Adaptive Boosting*** (12). The AdaBoost algorithm combines ***all*** weak learners into a strong learner, increasing an overall accuracy of the learning model. The main new feature of this algorithm is that it "***adapts***" to the errors of hypotheses produced by PAC-type weak learning algorithms, by readjusting the weights assigned to all examples. The weight readjustment is done in such a way, that the higher weightage is assigned to the previously misclassified examples and the lower weightage is assigned to the examples of the correct predictions. Therefore, more attention is given to poor predictions.
This approach is different to a hypothesis boosting method, which was mentioned earlier, as it does not alter the distribution of the examples in order to focus only on the hardest to classify examples, rather it uses all of the given examples. Moreover, Adaptive Boosting no longer relies on the pre-set constants of the PAC model such as error parameter $\boldsymbol{\varepsilon}$, it introduces its own parameter $\boldsymbol{\beta}$, which is no longer constant and changes in accordance with the calculated error of the hypothesis during each iteration of the learning process.

Initially each training example has an equal weight ***W*** assigned to it, so that ***W=1/N***, where ***N*** is the number of all training examples. The sum of all weights then equals to 1. The next phase of computations is set in a loop, or a sequence of ***M*** iterations, where each cycle of this loop uses a different training example out of the training set. This phase consists of four stages (12):

Given the set of training examples (x1, y1), (x2, y2), …, (xn, yn),
where *x={0, 1}, x ∈X, y={0, 1}, y ∈Y,* where x
For m = 1, 2, …, M:

- Calculate the probability distribution by normalization of weights:

$$p^m = \frac{W^m}{\sum_{i=1}^{N} W_i^m}$$

- Calculate the hypothesis ***h***

- Calculate the error of the hypothesis:

$$\varepsilon_m = \sum_{i=1}^{N} p_i^m |h_m(x_i) - y_i|,$$

  where $\boldsymbol{y_i}$ is actually $\boldsymbol{c(x_i)}$ of the PAC learning model.

- Assign new weights:

$$w_i^{m+1} = w_i^{m+1} \beta_m^{1-|h_m(x_i)-y_i|}, \text{ where } \beta_m = \frac{\varepsilon_m}{(1-\varepsilon_m)}$$

After the sequence of ***M*** iterations is completed, the final hypothesis is computed as:

$$h(x) = \begin{cases} 1 & if \ \sum_{m=1}^{M} (\log 1/\beta_m) h_m(x) \geq \frac{1}{2} \sum_{m=1}^{M} \log 1/\beta_m \\ 0 & else \end{cases}$$

This model uses a weak learner in each iteration and after a certain number of iterations all weak learners are combined to form one strong learner, as a result of this boosting method - the prediction is much more accurate.

3.2.7 AdaBoost for face detection

In (5) and (7), Viola and Jones were able to successfully implement the AdaBoost learning algorithm for face detection, using smaller number of features. Such task required slight modifications of the original algorithm.

The scanning area by which an image is sequentially scanned through is a rectangular area known as a sub-window, that has a size of 24 by 24 pixels. A sub-window represents a training example $\boldsymbol{x}$, containing features $\boldsymbol{f_j}$. Recalling the original AdaBoost algorithm, which was described in the previous section - only one ***weak*** classifier $\boldsymbol{h^m(x)}$ was trained during each iteration $\boldsymbol{m}$ of the learning process, but here we need to deal with multiple features for every input $\boldsymbol{x}$, that is for every sub-window. A solution was proposed to train a separate weak classifier for each feature and then to select only the “best” one, which produced the lowest classification error.

For every feature $\boldsymbol{f_j}$ - a separate weak classifier $\boldsymbol{h_j}$ is trained as:

$$h_j(x) = \begin{cases} 1 & if\ p_j f_j(x) < p_j \theta_j \\ 0 & else, \end{cases}$$

where $\boldsymbol{p}$ shows the direction of inequality, $\boldsymbol{\theta}$ is a threshold, and $\boldsymbol{x}$ is a sub-window.

This way each weak classifier deals with only a single feature, so the number of all trained weak classifiers during each iteration of the algorithm corresponds to the number of all given features.

The error of each trained weak classifier is then calculated by:

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|,$$

where $\boldsymbol{w_i}$ is normalized weight, $\boldsymbol{y_i}$ = {0, 1} for negative and positive examples.

Only one weak classifier $\boldsymbol{h_m}$ which produces the lowest error $\boldsymbol{\varepsilon_m}$ among all other weak classifiers is then selected. As a result, for each iteration $\boldsymbol{m}$ of the AdaBoost learning process - only one single feature is chosen. After the sequence of $\boldsymbol{M}$ iterations is completed - the final ***strong*** classifier is obtained.

3.2.8 Cascade of classifiers

Even though AdaBoost algorithm can produce highly accurate predicitions, the amount of possible facial features evaluated for each sub-window of an image would still be enormously large. Therefore, such method of detection would not be very efficient in terms of time consumption and resource usage. Viola and Jones proposed a solution to this problem by forming a cascade of ***strong***(boosted) classifiers. It is assumed that most of sub-windows of any image do not contain a face, therefore if they can be discarded quickly without the evaluation of all possible features in them, then the computation time will not be wasted on such examples and would rather be used where it is really needed. For example, the first and simpler classifier may only use a couple of features, but it can prove sufficient enough to immediately reject most of the negative sub-windows, leaving harder tasks for later classifiers. Thus, a cascade or series of classifiers is constructed and arranged in order of increasing complexity, such that for each next classifier the number of used features is increased until pre-set detection rate and false positive rate are achieved. The number of classifiers needed, relies on the overall pre-determined false positive rate, which a cascade of classifiers should produce. All sub-windows are evaluated by this cascade of classifiers. If at any stage any classifier does not produce a positive result within its own pre-set parameters, then the current sub-window under evaluation is rejected completely. (7)

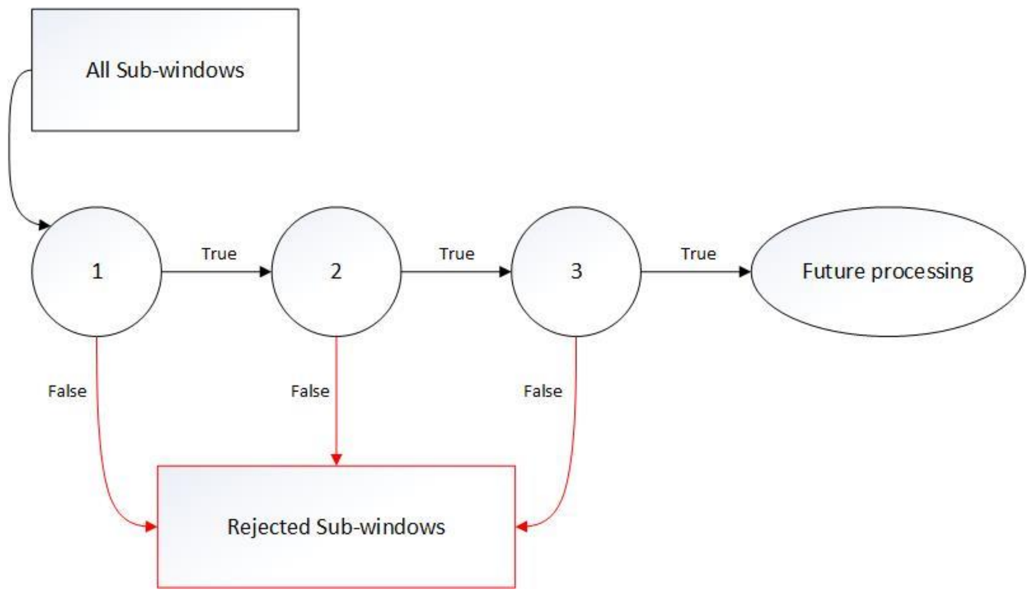


Figure 6 - Cascade of classifiers schematic (authors work), inspired by (7).

3.3 Face Recognition

Patil and Deore have defined face recognition as a visual pattern recognition problem that typically consists of the following sequential processes (15):

1. **Face detection**. In this step the face areas within an image are located by rough estimations. Face detection separates located faces from the rest of the scene. In addition to face detection a face alignment technique can also be implemented to achieve more accurate detection of locations and sizes of the faces.

2. **Feature extraction**. This process provides information about the distinct features of faces that is helpful for distinguishing between faces of different individuals.

3. **Face matching**. The extracted facial features of input faces are compared with a database of already registered facial features. If the face match is found with the acceptable confidence, the identity of the face is produced, otherwise the face is declared as unknown.

The Viola-Jones framework already covers the face detection process. The feature extraction and face matching processes can be performed by the Eigenface method that also uses the Principal Component Analysis to reduce the dimensionality of the data and to explain its variance.

3.3.1 Principal Component Analysis

Principal Component Analysis, or in short PCA, is a dimensionality reduction algorithm, which enables visualization and extraction of the most important information from multi-dimensional datasets, where each variable represents a different dimension. When dealing with large datasets, containing numerous quantitative variables, PCA can be used to reduce the redundancy of the data and explain the variance in the original data by new variables called Principal Components. These components can be obtained by finding out the directions of the largest variations in this data. The maximum possible amount of the Principal Components is equal to the amount of all original variables in any given dataset, but in order to reduce the dimensionality and get rid of the redundancy of the data - we can use fewer Principal Components and still be able to explain most of the variance in the original data. This way we can work with a much smaller dataset of variables derived from a much larger original dataset. (16)

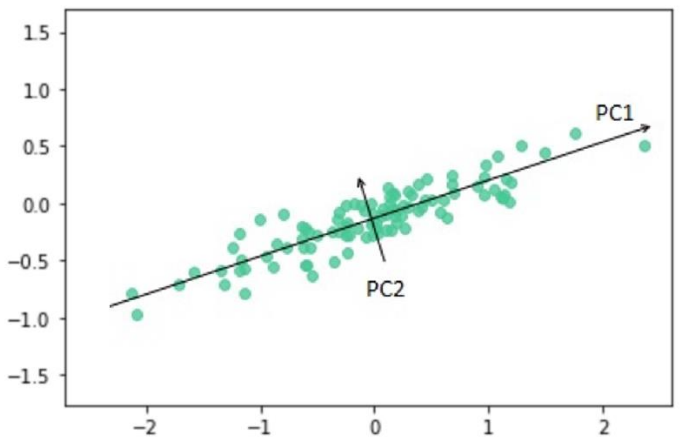


Figure 7 - Example of Principal Component used to find the largest variation of variables in two-dimensional data (authors work).

The Principal Components are in fact the eigenvectors of a covariance matrix, and the eigenvalues are the measurements of variation projection onto the Principal Components. An example of such projection is shown in Figure 8.

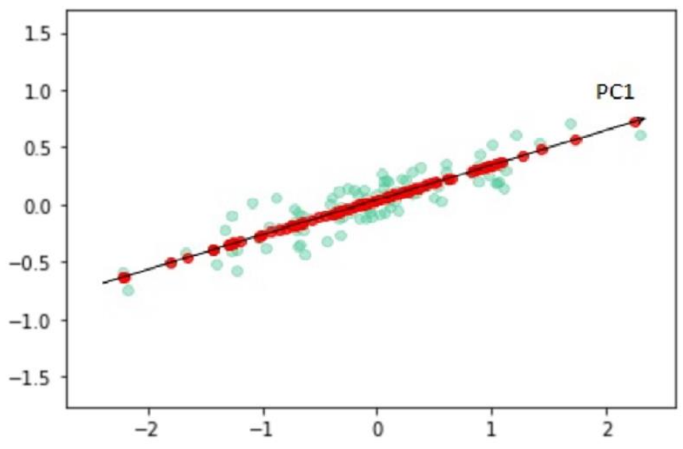


Figure 8 - Example of data projection onto the PC1 axis.

The Principal Component Analysis can be broken down into few stages:

1. Optional Standardization of the variables of a given data.
2. Calculating a covariance matrix from a given data.
3. Obtaining eigenvectors and their corresponding eigenvalues

PCA is used in many fields, including computer vision. Digital images may contain enormous amounts of data that can hinder the computation speed of face recognition algorithms. Hence, the data dimensionality reduction is very useful for such task.

3.3.1.1 Standardizing

„Standardizing the variables may be thought of as an attempt to remove the problem of scale dependence from PCA.“ (17)
Standardization or scaling of the variables in PCA is explained by the desire to measure the variables of the original data in same units of one range, thus removing the problem posed by differencies in variances of the variables of different types. Such differencies in variances can unfairly affect the outcome of the analysis, since some of these variancies may have a stronger influence on the result than others, therefore such analysis may not be as accurate as it could be. For example, if the data consists of two types of measurements, then one type of measurements may have a bigger impact on the outcome of the analysis than the other, if the range of its possible values is much wider. (17)

Standardizing is performed as follows (16):

$$\frac{x-\bar{x}}{s(x)} \quad ,$$

where $\bar{\boldsymbol{x}}$ is the mean of $\boldsymbol{x}$ values and $\boldsymbol{s}$ is a standard deviation.

3.3.1.2 Calculating a covariance matrix

Recalling that PCA is performed on multi-dimensional datasets, let such data be defined as:

$$X = [x_1, x_2, \dots, x_n],$$

where $\boldsymbol{n}$ is the number of all $\boldsymbol{m}$ dimensional vectors $\boldsymbol{x_i}$, for $\boldsymbol{i} = 1,2,\dots,\text{n}$.

The covariance matrix ***C*** of the given data is then calculated as:

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T.$$

This covariance matrix is a square matrix of size ***m*** by ***m***, where the mean $\boldsymbol{\bar{x}}$ is calculated by:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

3.3.1.3 Eigenvectors and eigenvalues

In matrix analysis (18), an ***eigenvalue*** is defined as a scalar $\boldsymbol{\lambda}$ that fulfils the equation:

$$Ax = \lambda x, \quad v \neq 0,$$

where ***A*** is a square matrix and ***x*** is a non-zero vector. If such condition is met, then a non-zero vector ***x*** is called an ***eigenvector***. Each eigenvector ***x*** of the matrix ***A*** is associated with an eigenvalue $\boldsymbol{\lambda}$.

The equation can be also rewritten as:

$$(\lambda I - A)x = 0,$$

where ***I*** is the identity matrix.

Since an eigenvector must be a nonzero vector, then the equation is satisfied only if the determinant of $(\boldsymbol{\lambda I - A})$ equals to zero:

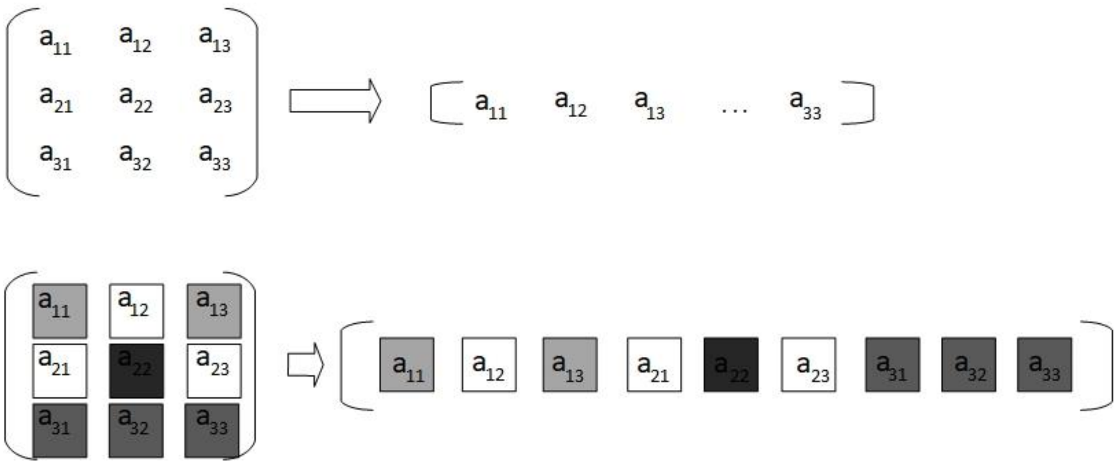
$$p_A(\lambda) = det(\lambda I - A) = 0.$$

Such equation is called the ***characteristic equation***, where $p_A(\lambda)$ is the ***characteristic polynomial***. It is known that the equation provides at most ***n*** solutions, where ***A*** is an ***N*** by ***N*** matrix, hence there are ***n*** eigenvalues and ***n*** associated eigenvectors of the matrix ***A***.

3.3.2 Eigenface method

In (19) Turk and Pentland came up with a new and more computer vision related term called “eigenfaces” when referring to Principal Components of the Principal Component Analysis (also referred to as eigenvectors of the covariance matrix) of the set of face images. In their work, they described how Principal Component Analysis can be used on a set of face images in order to retrieve the eigenfaces which best describe the distribution of these face images, and then how to classify or identify new incoming face images in comparison to the original set of face images. This section describes their eigenface method of applying PCA on face images.

The main idea lies in representing a face image as a vector. For example, if a grayscale image of a face is a 2-dimensional ***M*** by ***M*** array of 8-bit intensity values ranging from 0 to 255 then it can be represented as a vector of dimension $\boldsymbol{M^2}$. In Figure 9, we can observe an example of such transformation.



a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}
a_{11} a_{12} a_{13} … a_{33}

Figure 9 - Example of grayscale image transformation into a vector (authors work)

After transforming all face images into vectors of dimension $\boldsymbol{M^2}$, the common features of the faces shared among the entire set of vectors are removed. Therefore, each face is left with its own distinctive features. The common features are defined as an average face of the set:

$$\Psi = \frac{\sum_{i=1}^{N} \Gamma_i}{N}$$

Where $\boldsymbol{\Psi}$ is the average face, $\boldsymbol{N}$ is the number of all face images and $\boldsymbol{\Gamma_i}$ is a face vector from a set of face vectors $\boldsymbol{\Gamma_1}$, $\boldsymbol{\Gamma_2}\ldots\boldsymbol{\Gamma_N}$.

Distinctive features of each face are then computed as follows:

$$\Phi = \Gamma - \Psi$$

This way we can transform a whole set of face images into a set of vectors with only unique or distinctive features of the faces and then apply Principal Component Analysis. PCA will enable us to reduce the dimensionality of our data, while also will provide us with the most important information from it. It will be up to us then by how many $\boldsymbol{k}$ eigenvectors or Principal Components we want to describe the original data. The eigenvalues themselves will be ranked in the descending order in correspondence to the values of their eigenvectors, thus we will know which eigenvectors are the most important in terms of describing the variation of the data.

To calculate the eigenvectors, we should calculate the covariance matrix:

$$C = AA^T = \frac{\sum_{m=1}^{N} \Phi_m \Phi_m^T}{N} \quad where\ A = [\Phi_1 \Phi_2 \ldots \Phi_N]$$

But it is usually better to calculate the covariance matrix of reduced dimensionality instead:

$$C = A^T A \quad where\ A = [\Phi_1 \Phi_2 \ldots \Phi_N]$$

The reason that the covariance matrix of reduced dimensionality is usually first used, instead of a normal covariance matrix where $\boldsymbol{C=AA^T}$, is due to the need of reduction of the heaviness of the computations. Since matrix $\boldsymbol{A}$ is constructed from the set of $\boldsymbol{N}$ vectors, each of dimension $\boldsymbol{M^2}$, its size is therefore $\boldsymbol{M^2}$ by $\boldsymbol{N}$, thus the covariance matrix then would be of the size $\boldsymbol{M^2}$ by $\boldsymbol{M^2}$, whereas a covariance matrix of reduced dimensionality $\boldsymbol{C=A^TA}$ would be a matrix of the size $\boldsymbol{N}$ by $\boldsymbol{N}$. Such matrix would consist of $\boldsymbol{N}$ eigenvectors, where each eigenvector is a vector of dimension $\boldsymbol{N}$. Hence, through such approach the size of the matrix $\boldsymbol{C}$ is reduced and so is the maximum possible number of the eigenvectors is reduced, given that the number of vectors in the matrix $\boldsymbol{A}$ is less than the number of these vector's dimensions, $\boldsymbol{N<M^2}$.
In simpler terms, the choice of using a covariance matrix of reduced dimensionality can be explained by the fact that usually the number of images in any given set would be lower than the number of pixels within any image of the given set, since all images of the set are of the same size.

The next step then is choosing $\boldsymbol{k}$ eigenvectors $\boldsymbol{v_i}$ which best represent the data from the reduced covariance matrix $\boldsymbol{C=A^TA}$, given that $\boldsymbol{k}$ is less or equal to the number of all face vectors $\boldsymbol{N}$. After obtaining these $\boldsymbol{k}$ eigenvectors we can then convert them back into the eigenvectors $\boldsymbol{u_i}$ of the covariance matrix $\boldsymbol{C=AA^T}$ given that $\boldsymbol{u_i = Av_i}$. Such transformation allows our $\boldsymbol{k}$ best eigenvectors to be returned to the original dimensionality.

The linear combination of the eigenfaces is calculated by:

$$u_i = \sum_{k=1}^{N} v_{ik}\Phi_k \quad where\ i = 1, 2, \dots, N$$

Every face image from the original training set is represented as a linear combination of the weighted best ***k*** eigenfaces:

$$\Omega^T = [w_1 w_2 \dots w_k]$$

$$where\ w_i = u_i^T(\Gamma - \Psi)\ for\ i = 1,2, \dots, k\ .$$

Each ***i***th eigenface provides its own contribution in representation of the original face image. Any new incoming face image can also be represented as a vector $\boldsymbol{\Omega}$, hence it will be projected onto the principal component axes of the training set, and then compared to the previously defined face classes of the training set by calculating the Euclidian distances between their weight vectors $\boldsymbol{\Omega}$:

$$\varepsilon_i = \|\Omega - \Omega_i\|,$$

where $\boldsymbol{\Omega_i}$ is a weight vector of the ***i***th face class of the training set.

A new face will belong to the ***i***th face class where the minimum distance $\boldsymbol{\varepsilon_i}$ was observed, if this distance is lower than some pre-set threshold $\boldsymbol{\theta}$, otherwise a new face will be classified as “unknown”.

4 Practical part

The practical part of the thesis describes the stages of creation and implementation of a desktop application designed to perform face detection and face recognition using the frameworks and the methods studied in the literature review. It starts with the description of the software tools used to create the application, followed by the presentation of the design of the recognition system. Based on the proposed design the stages of implementation of the face recognition system are identified and described, step by step. The characteristics of the database of images used for this work are presented along with the explanation of how this data was organized. After all processes of the system are explained, a simple framework is proposed to fine-tune the detection parameters to minimize the influence of the detection algorithm on the recognition rate.

4.1 Software

The following software has been used to create a desktop application in order to implement and test both the face detection algorithm based on Haar-like features and the face recognition algorithm based on the Eigenfaces method.

4.1.1 Microsoft Visual Studio

Visual Studio is an integrated development environment (IDE) used for software development; it allows development of applications that run on .NET Framework (20). I have used a free Visual Studio Community 2017 version to create a Windows desktop application for face detection and face recognition, written in C# programming language.

4.1.2 EmguCV

EmguCV is a cross platform .NET wrapper for the OpenCV image-processing library. It allows OpenCV functions to be used by .NET compatible languages such as C#, VB, VC++, IronPython. The wrapper can be compiled in Mono and used on Windows, Android, iOS, Windows Phone, Mac OS X and Linux. (21)

Both OpenCV and EmguCV are of open-source type of software. EmguCV provides a set of libraries that allow my desktop application to perform the following functions:

- Detect faces from images using a pre-trained cascade classifier, based on Haar-like features.
- Rescale faces using a built-in interpolation technique.
- Train the face recognition algorithm based on the Eigenfaces method.
- Classify new images with reference to the previously trained data.

The application was using an EmguCV of version 3.4.3.3016.

4.1.3 Haar cascade classifier

In this work, I did not train my own Haar cascade classifier for face detection but used an already existing, pre-trained cascade classifier[1], freely provided by the OpenCV. Such approach allowed me to save some time and to focus more on other aspects of the face recognition system. As mentioned earlier in the literature review, one of the reasons why the Viola-Jones framework based on Haar-like features is still so popular is because once the Haar cascade classifier is trained, it can be repeatedly used by other researchers.

4.2 Designing the system

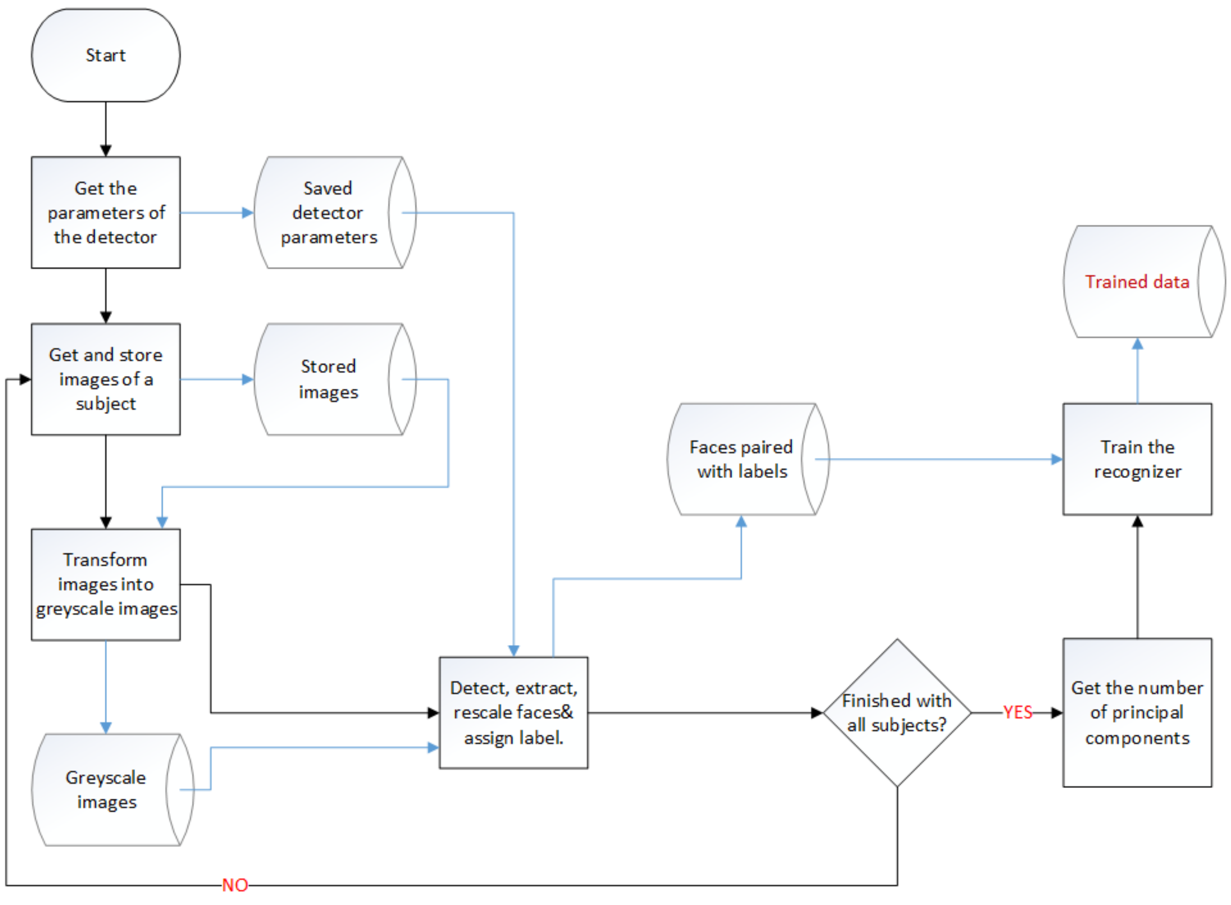
To develop the application that performs face recognition, I have identified the following main sequential parts of the face recognition system:

1. **Training phase**. This phase consists of the series of processes required for using an Eigenface approach of applying PCA on face images obtained from a training set of images.
2. **Testing phase.** The testing phase is composed of the series of processes required for classifying new images obtained from the testing set of images.

[1] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

4.2.1 Training phase subsystem

The Figure 10 illustrates the data flow diagram of the prototype application subsystem dedicated for performing a Principal Component Analysis on a training set of images.



Start
Get the parameters of the detector
Saved detector parameters
Get and store images of a subject
Stored images
Transform images into greyscale images
Greyscale images
Detect, extract, rescale faces& assign label.
Faces paired with labels
Finished with all subjects?
YES
NO
Get the number of principal components
Train the recognizer
Trained data

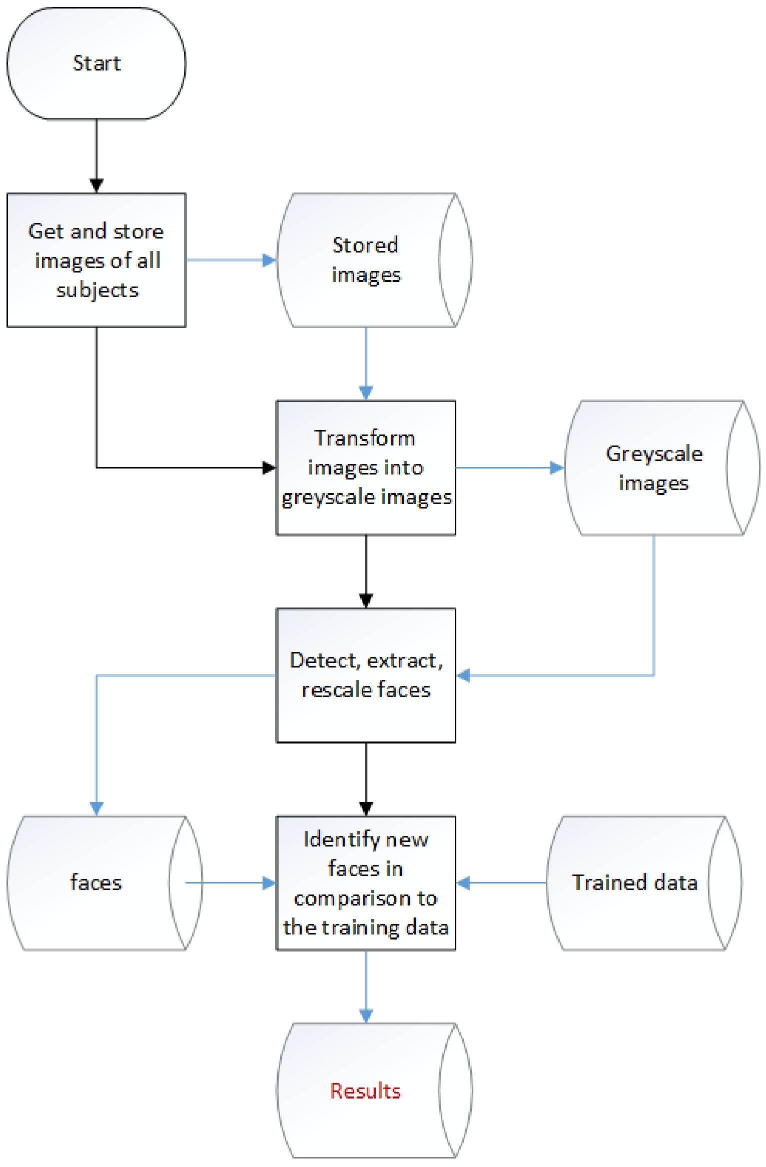
Figure 10 - Dataflow diagram of the training subsystem (authors work).

The training phase consists of the following processes:

1. Setting parameters of a face detector.
2. Acquiring images.
3. Converting acquired images to grayscale.
4. Detecting faces from grayscale images.
5. Extracting and resizing detected faces to specific pre-set pixel height and width.
6. Pairing extracted faces with labels.
7. Training the face recognizer and saving the results to a file.

4.2.2 Testing phase subsystem

The Figure 11 illustrates the data flow diagram of the prototype application subsystem dedicated for testing the face recognition algorithm.



Start
Get and store images of all subjects
Stored images
Transform images into greyscale images
Greyscale images
Detect, extract, rescale faces
faces
Identify new faces in comparison to the training data
Trained data
Results

Figure 11 - Dataflow diagram of the testing subsystem (authors work).

This phase uses a data set specifically prepared for testing the face recognition algorithm. All stages of this testing phase, prior to making predictions of new faces, are the same as in a previously described training phase.

4.2.3 Full face recognition system

Both training and testing phases have certain common sequential stages, such as:

1. Image acquisition.
2. Conversion of images to grayscale representations.
3. Face detection.
4. Face extraction and resampling (rescaling).

Therefore, the two subsystems used the same algorithms created for these mentioned stages, while using different image sets for their cause. After completion of these stages, in the training phase a PCA (with a chosen number of principal components) was applied on a training set of images, and in the testing phase new images were projected onto the PCA space of the training set and classified with respect to the previously defined faces. Figure 12 illustrates the data flow diagram of this face recognition process.

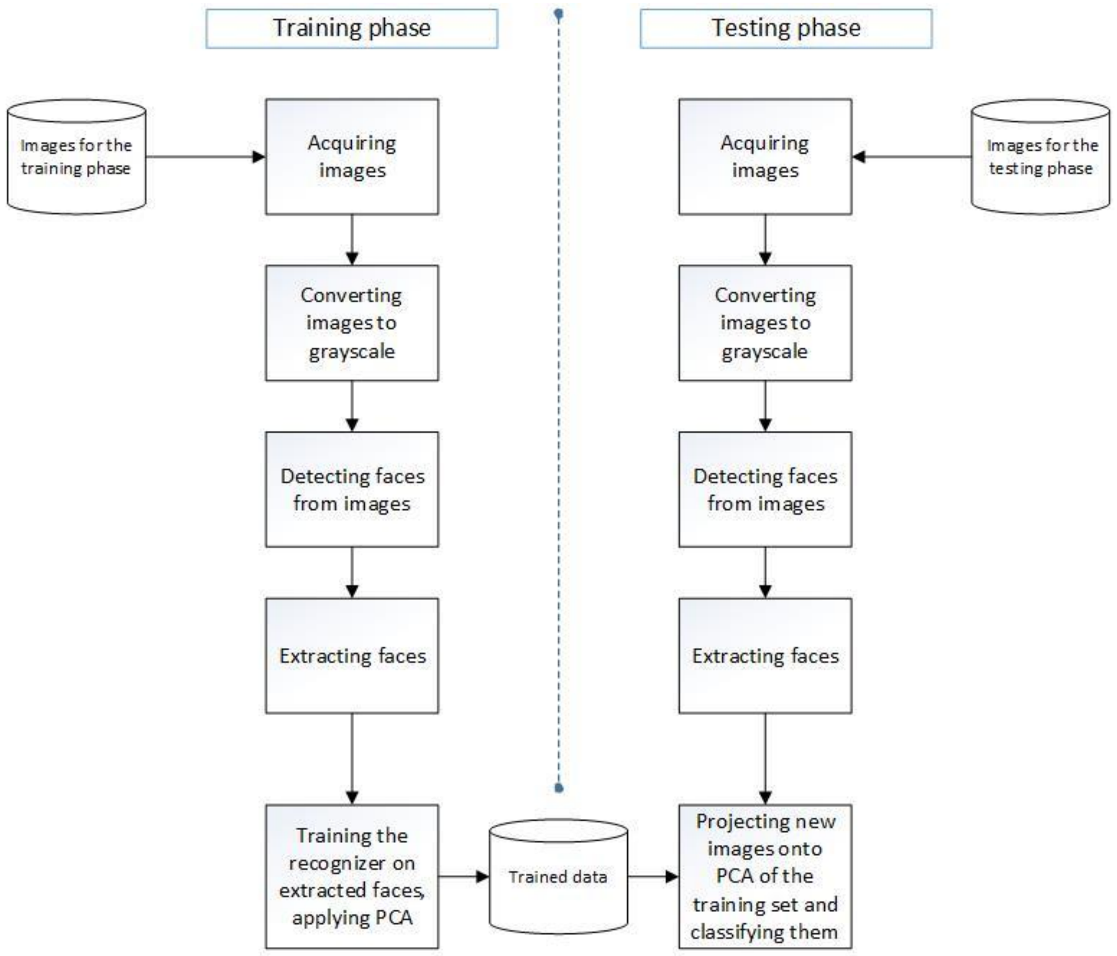

Training phase
Testing phase
Images for the training phase
Acquiring images
Converting images to grayscale
Detecting faces from images
Extracting faces
Training the recognizer on extracted faces, applying PCA
Trained data
Images for the testing phase
Acquiring images
Converting images to grayscale
Detecting faces from images
Extracting faces
Projecting new images onto PCA of the training set and classifying them

Figure 12 - Full recognition system (authors work).

4.3 Acquisition of images

The acquisition of images is the first stage of the training phase and the testing phase of the system. This part will describe the data used for this work and how this data was organized to perform face recognition.

4.3.1 Data set of images

To train and test the face recognizer I have used a part of the database of images **face94**, assembled by Dr Libor Spacek (22). The face94 database provides 20 frontal-face view images for each of the 153 individuals. It mostly consists of pictures of 18 to 20 years old undergraduate students, but it also has images of older people. The database contains images of both male and female genders of different racial origins. There are also some images of bearded male individuals, as well as images of both genders wearing glasses. All pictures of the individuals were taken against a green mono-coloured background under artificial lighting with a mixture of tungsten and fluorescent overhead. All images had a resolution of 180 by 200 pixels and were stored in 24-bit RGB, JPEG format.

This data set appeared to be suitable for my work. According to Turk and Pentland, the background of an image can seriously influence the face recognition performance (19). Since all images from face94 had a green mono-coloured background, I had assumed that image background influence was minimal, hence it would not impact the recognition performance greatly.

4.3.2 Data organization

An entire **face94** data set is unbalanced. For example, the number male images not having any types of occlusions was about 10 times greater than the number of females wearing glasses, or about 5 times greater than images of bearded males. Therefore, it was decided to use only a part of the original data set, and work with equal instances of different classes to measure the performance of face recognition using the Confusion Matrix.

Unfortunately, after downloading the freely provided images for my research purposes, some of the images appeared to be corrupt for almost every individual, therefore not all of them could be used. It was then decided to use only 16 images per individual.

The obtained images of human faces from the data set **face94** have been separated into two categories as follows:

1) Images used for the **training phase:** 11 images per each selected individual were subject to Principal Component Analysis using the Eigenface method.
2) Images used for the **testing phase** were images of the same people, consisting of 5 images per person - they were used to test the performance of the face recognition algorithm. These images were not used as part of the training data and were stored separately.

Nearly 70% of all selected images were used in a training set and about 30% of images were used for testing the face recognition algorithm.

4.4 Converting images to grayscale

Since images taken from the face94 dataset are color images, the conversion of these images into their grayscale image representations is a requirement for performing face detection based on Viola and Jones framework. Moreover, it is also a requirement for performing Principal Component Analysis and classification of the new images, based on the Eigenface method. Therefore, an implementation of grayscale conversion of acquired images is a necessary next step after image acquisition, for both training and testing phases of the proposed face recognition system.

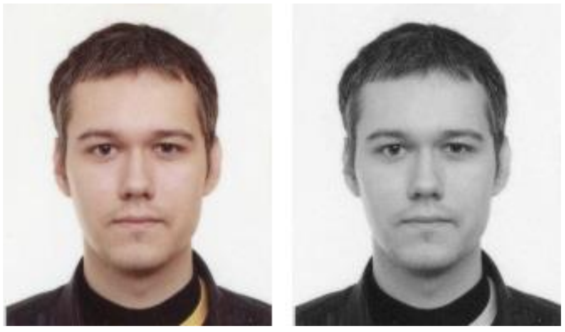


Figure 13 - Example of grayscale conversion (authors work)

In the OpenCV the default color format is BGR (23). Given that EmguCV is a wrapper of the OpenCV, I have decided that my desktop application will be using images of the BGR color model. During some experiments, I have encountered few problems myself, when trying to work with the RGB images. Perhaps it was just my own inability to work with this color model correctly.

Using the **Image<TColor, TDepth>** class[2] of the EmguCV, the program read every image file in a BGR color format and obtained a grayscale representation of each loaded image by performing the following operations written in C#:

```
Image<Bgr, byte> inputImage = new Image<Bgr, byte>(file);
var grayImage = inputImage.Convert<Gray, byte>().Clone();
```

The conversion of acquired images to grayscale was the only “pre-processing” step made prior to face detection. Such proposed model places a certain limitation on the face recognition performance and leaves room for improvement.

4.5 Face detection

After acquired images were converted to their grayscale representations, it was time to detect faces.

At first, a new object of the **Emgu.CV.CascadeClassifier**[3] class was created and loaded with an XML file of the pre-trained Haar Cascade Classifier.

```
CascadeClassifier HaarClassifier = new CascadeClassifier
(System.IO.Path.GetFullPath
(@"../../data/haarcascade_frontalface.xml"));
```

[2] http://www.emgu.com/wiki/index.php/Working_with_Images#Depth_and_Color_as_Generic_Parameter

[3] http://www.emgu.com/wiki/files/3.0.0/document/html/b940bc22-751b-d7bf-e9a4-8798e6f7ccb3.htm

The next step was to call the **DetectMultiScale**[4] method of the earlier mentioned class, that scanned each given image multiple times at different scales to find rectangular regions that were expected to contain human faces, using the pre-trained cascade. Trying to detect face regions in an image at different scales implies that the found face regions may vary in size. For now, the coordinates and size of each found region were stored in the variable called "**faces**".

```
var faces = HaarClassifier.DetectMultiScale(grayImage, scaleFactor, minNeighbors);
```

The variables used by the **DetectMultiScale** function were:

1. **grayImage** – This variable stored a grayscale representation of any given color image, over which the face detection was performed.
2. **scaleFactor** – Scale factor is a parameter that specifies by how much a rectangular search area should be increased between subsequent image scans. As an example, if a scale factor had a value of 1.3, it would mean that on the next scan the rectangular search area would increase by 30%.
3. **minNeighbors** – Minimum neighbors is a parameter for setting up the boundaries on the grouping of the found neighbouring rectangles that make up an object or in this particular case – a face. Since an image is scanned multiple times at different scales and locations during the detection process, this can often lead to multiple detections of the same object. Grouping of these detected areas and finding out their average area that makes up an object is based on a pre-set threshold of the neighbouring rectangles. If the threshold is not bypassed, then such a group of rectangles is not considered to correspond to a valid object of interest.

Some face alignment technique could be implemented here to further improve the face recognition rate. In (24), Ekenel and Stiefelhagen showed that face alignment not only improves face recognition rate for nonoccluded faces, but also can improve the correct face recognition rate in cases of upper and lower face occlusions. As mentioned earlier, partial

[4] http://www.emgu.com/wiki/files/3.0.0/document/html/2b7345cd-2f43-6eb6-a73e-a64382d85d7b.htm

occlusuions like glasses or beards are present in some images of the chosen database. After I have experimented with the face detection parameters and have seen the recognition results, I decided that the chosen approach using this particular set of images would suffice. Most of the extracted face images appeared to be centred.

4.6 Face extraction and resampling

Since the **DetectMultiScale** method only found areas in images that were likely to contain faces, I needed to extract these areas from each image. The faces were extracted from each image using the previously saved information about the coordinates and the size of each found face region in every given image. Recalling that the detected face areas may vary in size, the extracted face images therefore may vary in size too. This could pose a problem for correct classification of the new data. As mentioned earlier in the literature review - in the Eigenface method, the chosen $\boldsymbol{k}$ best eigenvectors $\boldsymbol{v_i}$ of a training set of images can be obtained from calculating the reduced Covariance Matrix, to reduce the computational time, but eventually these eigenvectors will be converted back into the eigenvectors $\boldsymbol{u_i}$ of a Covariance Matrix, and faces will be represented as vectors $\boldsymbol{\Omega}$. When new faces from the testing set will be projected onto the pre-defined face space of the training set, they will also be represented as vectors $\boldsymbol{\Omega}$. Recalling that each weighted eigenface is computed as $\boldsymbol{w_i} = \boldsymbol{u_i^T}(\boldsymbol{\Gamma} - \boldsymbol{\Psi})$, we would need to keep the same dimensionality of the face vectors $\boldsymbol{\Gamma}$, if we wanted to calculate the Euclidian distances correctly. Hence, rescaling all extracted face images to a pre-set size was the next logical procedure. But it needed to be done correctly.

All faces were resized to a pre-set constant width and height in pixels (128 by 150) by an appropriate method.
An appropriate way of scaling images is using image interpolation techniques to try to accomplish the best approximation of pixel intensity values of an image (25). The process of reducing the size of an image by removing pixels is called image downsampling, while inserting pixels to increase the size of an image is called image interpolation or image upsampling (26).

A variety of interpolation techniques is provided by the **Emgu.CV.CvEnum**[5] Namespace. I used an **Inter.Cubic** method that can downsample or upsample an image. This was a perfect choice, since the extracted face images could be smaller or bigger in size than the constant pre-set size.

For every ***i***th input image, the following C# code shows an example of implementation of grayscale conversion, face detection using pre-set parameters, face extraction, and face resizing to a pre-set width and height in pixels using **Inter.Cubic** interpolation and decimation method:

```
var grayImage = inputImage.Convert<Gray, byte>().Clone();
var faces = HaarClassifier.DetectMultiScale
(grayImage, scaleFactor, minNeighbors);
int numFaces = 0;
foreach(var face in faces)
{
        var processedFace = grayImage.Copy(faces[numFaces]).Resize
                        (Width, Height, Emgu.CV.CvEnum.Inter.Cubic);
        numFaces++;
        if (processedFace != null)
        {
        Faces.Add(processedFace);
        Labels.Add(i);
        }
}
```

[5] http://www.emgu.com/wiki/files/3.2.0/document/html/eab4d5b4-81bc-52f9-1545-26230079ca30.htm

4.7 Pairing faces with labels

EmguCV lets us train the Eigenface recognizer by pairing the extracted faces with labels. Labels can only be accepted as an integer value (whole number). Thus, a set of extracted face images, that belonged to each individual (face class) in the training set of images, was given its own unique identifier. Later, in the testing phase, these labels were used to evaluate how well did the model predict the face classes of given new faces.

4.8 Applying PCA

This was the last stage of the training phase. After all grayscale face images from the training set were extracted and rescaled to a constant pre-set size, they became eligible for Principal Component Analysis, since they all consisted of the same number of pixels, with pixel intensity values ranging from 0 to 255, and could be represented as a set of vectors of the same dimensionality. Given the desired number of Principal components(eigenvalues), it was then possible to train the EmguCV face recognizer.

An object of the **EigenFaceRecognizer**[6] class was created with a pre-set number of Principal Components. In the next step, the PCA was applied using the **Train()** method of the previously mentioned class, to which all extracted faces and the corresponding labels were passed. Finally, the results were saved to a YML file.

```
EigenFaceRecognizer FaceRecognition = new EigenFaceRecognizer
                        (num_components, double.PositiveInfinity);
....
FaceRecognition.Train(Faces.ToArray(), Labels.ToArray());
FaceRecognition.Write(YMLPath);
```

[6] http://www.emgu.com/wiki/files/3.0.0/document/html/62795d93-e4b8-3c3a-f11f-42ce97895c0e.htm

The face recognizer was trained multiple times using different numbers of Principal Components. The YML files were later used in the testing phase for classification of new face images, depending on the chosen number of Principal Components.

4.9 Classifying new faces

This was the final step of the testing phase, prior to which the images from the testing set have been already acquired and converted to grayscale, and faces have been detected, extracted, and re-scaled. In this last stage the processed new faces were projected onto the PCA space of the trained data. The algorithm predicted to which of the pre-trained face classes each new face belonged to. Predictions were obtained as label values that have been used earlier to classify the face images of the training set. For each new face, its predicted label value was then compared to the real label value of the face class it belonged to, thus allowing to measure the performance of the face recognition model.

4.10 Classifying face detection results

Accurate face detection is one of the key determinants for successful recognition. To assess the performance of face detector and fine-tune the setup of parameters, a simple framework was proposed to classify face detection results into three basic groups:

1. **Correctly detected**. An outcome, where a detector correctly predicted the presence of a face and correctly located it. If an image of an extracted face contained a full centred face of a person or if it contained both eyes, a nose, and a mouth - such image was considered as a successful prediction made by the face detection algorithm.

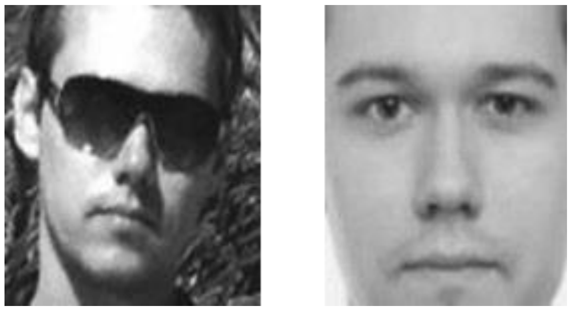


Figure 14 - Example of correctly detected face (authors work).

2. **Incorrectly detected**. An outcome, where a detector incorrectly identified a face. Images containing only a part of the face, lacking all the required facial features to fulfil the conditions of the correct detection, or images not containing a face at all – were considered as incorrectly detected.

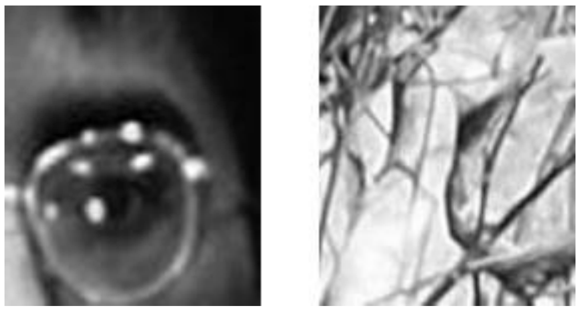


Figure 15 - Example of incorrectly detected face (authors work).

3. **Not detected**. An outcome, where a detector failed to find a face within the given image at all, but the face was present.

I then defined the successful detection rate as follows:

$$successuful\ detection\ rate = \frac{correctly\ detected\ faces}{total\ faces} * 100\%$$

Manual evaluation can be very time consuming, therefore, only 5 images for each of 99 individuals from the face94 database were subjectively tested in such a manner to fine-tune the face detection parameters. A simple algorithm was used by the created desktop application to count the number of all detected and not detected faces. Correct and incorrect detections were then counted manually.

4.11 Setting face detection parameters

The prototype application has been developed with the intent of fine-tuning the **scaling factor** and **minimum neighbors** parameters of the face detector to increase the detection rate. Other parameters were left with their default values during the face detection processes. After observing the influence of the chosen parameters on the detection rate, using images from the face94 dataset, the best combination of both parameters was selected and used in training and testing phases of the face recognition process. It must be said that the results obtained from these tests depended on the set of images collected for this work, hence, the results may differ when using other image databases and therefore, this may affect the choice made on the selection of the optimal values of detection parameters.

4.11.1 Figuring out the optimal face detection parameters

Face recognition rate depends not only on the recognition algorithm itself, but also on the face detection rate. If we can achieve the best results in detection, it can help improve the overall recognition results. For example, if some faces did not get detected, then we would have a lesser sample pool of images of certain people in our training set. After we would apply the PCA on our training set of images, we would have less data corresponding to these certain people's faces, and this could lead to future decline of the recognition rate. Moreover, we would also want to detect the most, if not all new face images in the testing phase.

As mentioned earlier, the **minimum neighbors** parameter sets the boundaries on grouping of the detected rectangular objects as a single rectangular object. In other words, if some objects were detected within a certain set proximity to each other, they would be counted as one object, in our case – a face. The image is scanned multiple times during the face detection process at different scales and the increase of the search area at each iteration is controlled by the **scaling factor**. I have manipulated both of these parameters to achieve the best detection results.

At first, I tried to figure out the optimal **minimum neighbors** parameter for my data set of images. The **scaling factor** was set to 1.4 (the search area was increasing by 40% between each subsequent scan) for obtaining results a bit faster, and it remained a constant value during this first test, while I changed the value of the **minimum neighbors** parameter. For this test, I used a set of 495 images, where each person out of 99 people in the dataset was represented by 5 images. The results of this test can be seen in Table 1 below.

Minimum neighbors	Correct detections	Incorrect detections	Not detected	Total	Correct detection rate
0	5618	84	1	5703	98.51%
1	490	7	5	502	97.61%
2	483	5	12	500	96.6%
3	473	3	22	498	94.98%
4	467	1	29	497	93.96%
5	446	0	49	495	90.1%

Table 1 - Face detection results with different values of the minimum neighbors parameter(authors work).

From the results observed in the Table 1, certain conclusions can be made. The increase of the value of the **minimum neighbors** parameter leads to certain advantages and disadvantages.

Advantages of increasing the value of **minimum neighbors** parameter:

- Decrease of the number of incorrect detections (false positives).

Disadvantages of increasing the value of **minimum neighbors** parameter:

- Decrease of the number of correct face detections (also called true positives).
- Increase of the number of not detected faces (false negative, when an algorithm could not find any face at all, but a face was present).

When the **minimum neighbors** parameter is set to 0, it means that there is no grouping of neighbouring objects at all, which leads to drastic increase of the number of objects detected, yet the majority of those detections may correspond to the same face, but with slightly different positioning and sizing of the rectangular detection area over that face. Additionally, such parameter settings returned the highest number of false positive detections. Therefore, the correct detection rate in this case may be very misleading. While having a higher correct detection rate is a good thing, such repetitive detections of the same object would also lead to multiple extractions of this same object into the training set. Such number of the extracted face images would not only take up more storage space but also would slow down the entire processing time of resizing of the images and of Principal Component Analysis. After analysing the results from the Table 1, it was decided to select the option where the **minimum neighbors** parameter was set to 1.

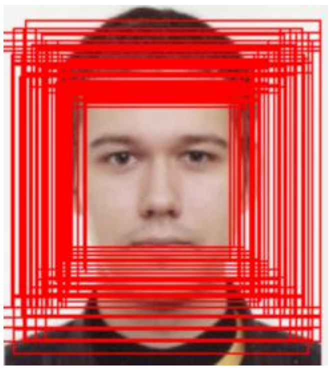


Figure 16 - Example of no grouping of found rectangles (authors work)

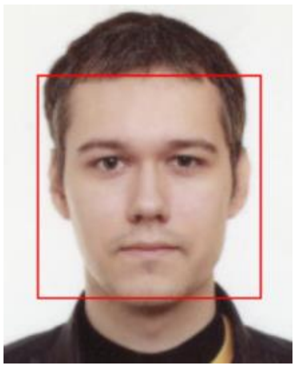


Figure 17 - Example of grouping found rectangles into one (authors work)

Now it is time to test the **scaling factor** in range from 1.1 to 1.6, starting from 1.1 and incrementing it by 0.1, while the **minimum neighbors** parameter will remain to be equal to 1.

Scaling factor	Correct detections	Incorrect detections	Undetected	Total	Correct detection rate
1.1	496	1	0	497	99.8%
1.2	495	0	0	495	100%
1.3	497	0	1	498	99.8%
1.4	490	7	5	502	97.61%
1.5	466	3	28	497	93.76%
1.6	511	2	23	536	95.34%

Table 2 - Face detection results with different values of the scaling factor parameter (authors work).

In the Table 2 we can see the face detection results when fine-tuning the **scaling factor**. It can be noticed, that when we increase this parameter, we get more undetected and incorrectly detected faces. This is due to the fact, that between each subsequent scan the scanning area is substantially increased, enough to sometimes miss the facial features within an image to successfully detect the face. On the other hand, a decision to set the **scaling factor** quite high can be explained by the need to decrease the overall computation time during the face detection process. An interesting observation can be made by comparing the results where the scaling factor was equal to 1.1 and to 1.2. It shows that when the search was performed too thoroughly, it resulted in finding an additional incorrectly predicted face. After analysing the results from the Table 2, it was decided to set the **scaling factor** parameter to 1.2, where a perfect result was observed.

Given the chosen settings of the two parameters a 100% detection rate was achieved. Such great results could probably be explained by the absence of a complex background in images, frontal face views and good lighting conditions.

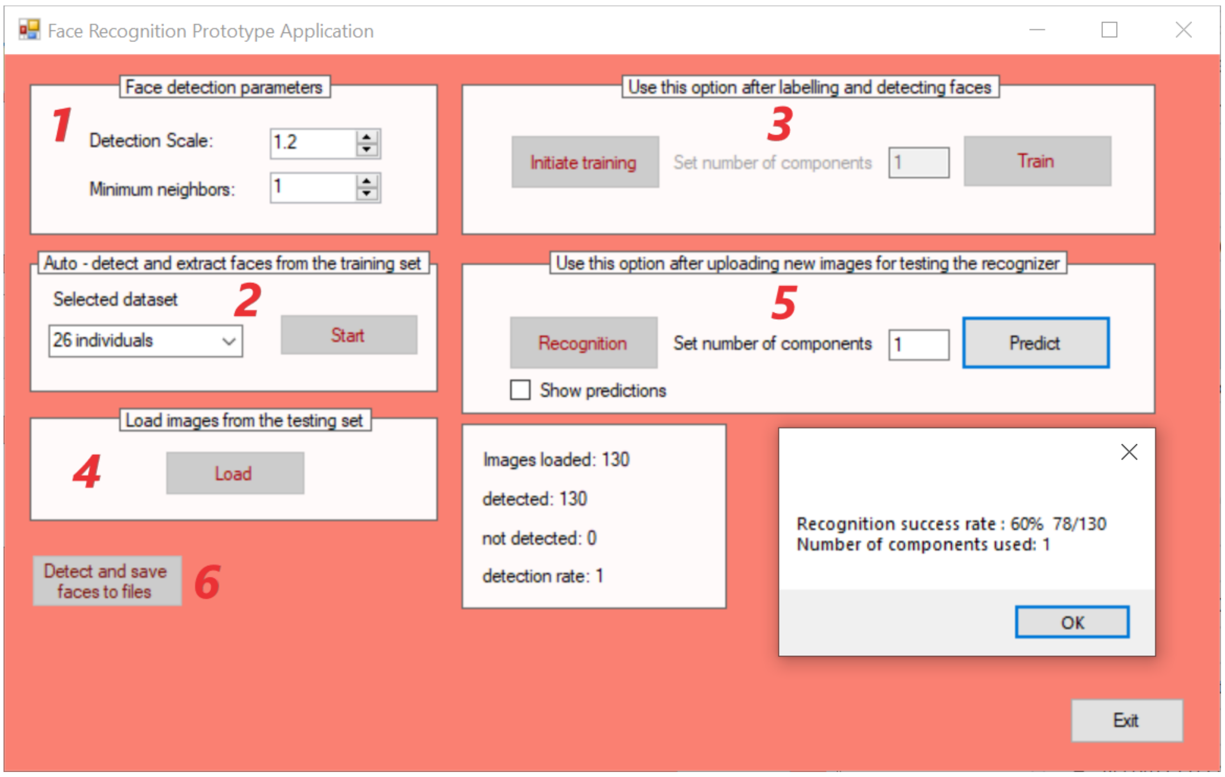
5 Results

5.1 Desktop application

To perform all the experiments a Windows desktop application was developed based on the proposed face recognition design using previously described software tools and methods. Using the face94 database of images of individuals, the application has performed face detection using a pre-trained Haar cascade classifier, with manipulation of such parameters as minimum neighbors and scale factor. The program counted the number of detected and not detected faces and stored the detected face images in JPEG file format for further evaluation. Additionally, the application used the Eigenface method to learn faces from the training set of images and stored the results in YML file format. The program then predicted to which of the known face classes did the new inputs from the testing set of images belong to. The obtained predictions were used to measure the performance of the model.

5.1.1 User interface

The following Figure 18 illustrates the user interface of the developed desktop application.


Face Recognition Prototype Application
Face detection parameters
1
Detection Scale:
Minimum neighbors:
Use this option after labelling and detecting faces
3
Initiate training
Set number of components
Train
Auto - detect and extract faces from the training set
Selected dataset
2
Start
Use this option after uploading new images for testing the recognizer
5
Recognition
Set number of components
Predict
Show predictions
Load images from the testing set
4
Load
Detect and save faces to files
6
OK
Exit

Detection Scale: 1.2
Minimum neighbors: 1
Set number of components 1
Selected dataset: 26 individuals
Set number of components 1

Images loaded: 130
detected: 130
not detected: 0
detection rate: 1

Recognition success rate : 60% 78/130
Number of components used: 1

Figure 18 - User interface of the desktop application (authors work).

The user interface of the application allows us to train and test the recognizer. In Figure 18, the region number 1 lets us set the face detection parameters. After these parameters are set, we can start the training phase or the testing phase. For the training phase, we continue to the region number 2, where our pre-arranged training data will be automatically processed. This automated data processing includes such stages as image acquisition, grayscale conversion, face detection, face extraction, resampling, and assignment of labels. In step 3, we set the number of principal components for the PCA and train the recognizer. To test the recognizer, we need to use the same detection parameters once again, after that we will proceed to the region 4 to load the images from the testing set. Finally, in the region 5, we will set the number of Principal Components and call the face recognizer to predict our testing data. Loaded images will be automatically processed as in the training phase, excluding the processes of automatic image acquisition and pairing faces with labels. "Detect and save" button, in region 6, will detect and save extracted faces from loaded images to separate files for further manual evaluation.

5.2 Overall face recognition rates

In this section the overall face recognition rates are shown regardless of the genders type or occlusion type, using different number of Principal Components. The face recognition rate was calculated as follows:

$$Face\ recognition\ rate = \frac{Total\ number\ of\ correct\ predictions}{Total\ number\ of\ tested\ faces} * 100\%$$

I used a dataset of images of 99 individuals, where each individual was represented by 11 images in the training set and by 5 different images in the testing set. The **minimum neighbors** detection parameter was set to 1 and the **scaling factor** to 1.2.

Number of principal components	Correctly classified individuals	Face recognition rate
1	87/495	17.58%
2	372/495	75.15%
3	469	94.06%
4	481	97.17%
5	488	98.59%
6	489	98.79%
7	489	98.79%
8	490	98.99%
9	491	99.2%
10	491	99.2%
11	495	100%
12	495	100%

Table 3 – Comparison of recognition rates of 99 individuals using different number of Principal Components (authors work).

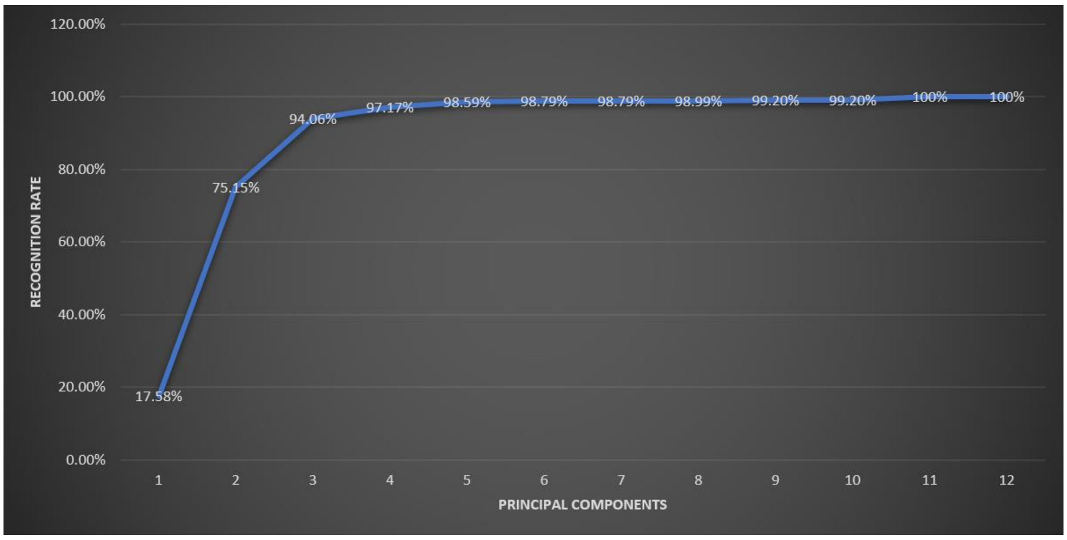


Figure 19 - The recognition rates of 99 individuals using different number of Principal Components (authors work).

5.3 Classification performance

To measure the classification performance of the model, depending on the number of Principal Components used for training and testing the data, I used the confusion matrix and such performance metric as accuracy. The recognition rates of the model were also measured depending on the number of PCs used.
The face recognition rate was calculated as follows:

$$Face\ recognition\ rate = \frac{Total\ number\ of\ correct\ predictions}{Total\ number\ of\ tested\ faces} * 100\%$$

The Confusion Matrix accuracy was calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where

- TP is the number of true positive cases.
- TN is the number of true negative cases.
- FP is the number of false positive cases.
- FN is the number of false negative cases.

5.3.1 Classification performance by gender

For this experiment, I used a dataset of images of 26 individuals, comprised of 13 males and 13 females. The recognizer was trained using 286 images, where each individual of the training set was represented by 11 images. The testing set consisted of 130 images, where each individual was represented by 5 images that have not been used in the training phase. Hence, the split of the data was around 70% for the training phase and 30% for the testing phase. The **minimum neighbors** detection parameter was set to 1 and the **scaling factor** to 1.2. The confusion matrix was created to evaluate the performance of the model based on gender classification.

Principal Components = 1	Male	Female
Male	92.31%	7.69%
Female	12.31%	87.69%

Table 4 - Confusion matrix of gender type classification with number of Principal Component set to equal to 1 (authors work).

Principal Components = 2	Male	Female
Male	100%	0%
Female	1.54%	98.46%

Table 5 - Confusion matrix of gender type classification with number of Principal Component set to equal to 2 (authors work).

Principal Components = 3	Male	Female
Male	100%	0%
Female	1.54%	98.46%

Table 6 - Confusion matrix of gender type classification with number of Principal Component set to equal to 3 (authors work).

Principal Components = 4	Male	Female
Male	100%	0%
Female	0%	100%

Table 7 - Confusion matrix of gender type classification with number of Principal Component set to equal to 4 (authors work).

Principal components	Recognition rate	Gender classification accuracy
1	60%	90%
2	92.3%	99.23%
3	99.2%	99.23%
4	100%	100%

Table 8 - Comparison of recognition rates and gender classification accuracies depending on the number of Principal Components (authors work).

5.3.2 Classification performance by occlusion

In this experiment I used images of 12 male individuals. The equal number of individuals was separated into the following four classes:

1. No facial occlusions.
2. Partial upper occlusion (wearing reading glasses).
3. Partial lower occlusion (bearded individuals).
4. Partial double occlusion (bearded individuals wearing glasses).

The recognizer was trained using 132 images and the testing set consisted of 60 images, using the same nearly 70/30 data split as in the previous experiment. The detection parameters also remained the same.

PC=1	No occlusion	Upper occlusion	Lower occlusion	Double occlusion
No occlusion	73.3(3)%	0%	0%	26.6(6)%
Upper occlusion	13.3(3)%	86.6(6)%	0%	0%
Lower occlusion	0%	0%	100%	0%
Double occlusion	6.6(6)%	6.6(6)%	0%	86.6(6)%

Table 9 - Confusion matrix of 4 types of occlusion classes with the number of Principal Component set to equal to 1 (authors work).

PC=2	No occlusion	Upper occlusion	Lower occlusion	Double occlusion
No occlusion	100%	0%	0%	0%
Upper occlusion	0%	100%	0%	0%
Lower occlusion	0%	0%	100%	0%
Double occlusion	0%	0%	0%	100%

Table 10 - Confusion matrix of 4 types of occlusion classes with the number of Principal Component set to equal to 2 (authors work).

Principal components	Recognition rate	Classification accuracy
1	81.66%	86.66%
2	95%	100%
3	100%	100%

Table 11 - Comparison of recognition rates and occlusion classification accuracies depending on the number of Principal Components (authors work).

6 Discussion

Table 3 and Figure 19 demonstrate an overall performance of the model using an unbalanced dataset of images of 99 individuals of different genders, with and without partial facial occlusions. The results from this first experiment show that the model required no less than 11 Principal Components to describe this data in order to achieve the perfect result of 100% face recognition rate.

In the second experiment, a balanced dataset of images of 26 individuals was used for measuring the performance of the model based on gender classification. The results from the Table 8 show that the model required only 4 Principal Components to arrive at 100% recognition rate.

In Table 11, a perfect recognition result is achieved using just 3 PCs, where a balanced dataset of images of 12 individuals was used for measuring the performance of the model based on occlusion classification.

All experiments show that the recognition rate increased with the increase of the number of Principal Components used, until reaching 100% recognition rate. Hence, the more Principal Components or Eigenfaces were chosen, the better the face recognition results were achieved. Additionally, the number of Principal Components required to achieve better results increased with the increase of the size of the dataset. Given that image data can be much more complex than the one used for this thesis - the optimal number of PCs to explain the variance of the data and to achieve good recognition results will differ. In fact, the experiments in this work show that the chosen database is very simplistic.

Since the proposed model was the most basic one and I did not even implement a good amount of image processing techniques that are often used to improve the performance of face recognition models - the achieved results were beyond my own expectations.

Confusion Matrix Tables 4, 5, 6, 7, show the rates at which the actual two gender classes were predicted by the model using from 1 to 4 (incrementing by 1) Principal Components respectively. Table 8 shows a comparison of recognition rates and gender classification accuracies of the model depending on the number of PCs. It shows that the classification performance of the model and the identification rate of each individual were increasing with the increase of number of PCs. Additionally, the results of the experiments demonstrate that prior to achieving a 100% recognition rate, the gender classification accuracy was always higher than the identification rate of the individuals. The performance of the model showed that it could classify the images by gender well.

Confusion Matrix Tables 9, 10 show the rates at which the actual 4 occlusion classes were predicted by the model using from 1 to 2 (incrementing by 1) Principal Components respectively. The results using 3 PCs were not included, since they were identical to the results when using 2 PCs. Table 11 shows a comparison of recognition rates and occlusion classification accuracies depending on the number of PCs. Just like with the gender classification, the performance of the model was increasing with the increase of the number of Principal Components used. Prior to reaching a 100% recognition rate, the model has shown the same performance trend – it performed well, and the classification accuracy was always ahead of the recognition rate.

The experiments revealed that the model has performed very well using the face94 database of images.
Such great results were achieved most likely due to the simplicity of the database of images chosen for this work. These results can probably be explained by the following possible factors:

- Each image had a simple monocoloured background without any additional environmental objects. As mentioned earlier, this factor has been shown by previous researchers to be influential on the recognition rate.

- Images contained the frontal views of human faces with little face angle deviation and little variation of facial expressions.
- The lighting conditions and shadows did not to pose any serious issues.
- The model has learnt faces too well. Further manual and subjective inspection of the database of images used for this thesis led me to believe that perhaps the testing samples did not differ much from the samples used in training the model, due to the nature of how pictures were taken using a camera. Although, all pictures were different, it seems like the pictures of each individual were taken in one session with some pre-set delay in-between. Without knowing the exact facts, I will subjectively assume that the delay was rather short, hence not allowing much deviation in facial expressions and poses, or maybe such was the initial plan of taking pictures of people with little variation in mentioned factors.

All of the factors mentioned above show that the experiments performed in this work cannot be related to the real-world scenario, where the complexity or the variation of the surrounding environment, lighting conditions, facial expressions and face angles is much greater. The results show that the model works for the given database of images and due to the low number of image samples and the simplicity of the dataset - these results cannot be really accepted as statistically significant in drawing conclusions about the population. Here I refer to population as to all possible images of humans taken, in different scenarios, for the purposes of face recognition. For example, cameras in airports, stores, universities, or in ATMs would be taking pictures at different angles, at different times of day and night, with different surroundings. A model that could better fit the real-world scenario would need to include many other techniques that would provide solutions to all these mentioned problems.

7 Conclusion

The literature review has described the Viola-Jones framework for face detection using Haar-like features, and face recognition using the Eigenface method by applying Principal Component Analysis to face images. In the practical part of the thesis, the design of the proposed model and the stages of implementation of the recognition system were demonstrated and described. Following the design of the proposed recognition system a desktop application was created with C# using EmguCV libraries, based on the gained knowledge from the study of related scientific and technical sources. The program has collected, and computed data derived from the database of images of human faces, and successfully performed face detection and face recognition of the given images using the previously studied methods.

The best parameters of the face detection algorithm were selected to minimize its influence on the face recognition rate. The predictions made by the program were then used to evaluate the influence of gender and of such facial features as glasses and beards on the proposed model and measure its performance. The differences between face recognition settings, such as using different number of Principal Components, have been identified and the relative success rates were compared.

The model developed for this work and the algorithms which have been studied – are serving only as a basis of face detection and face recognition, leaving much room for improvements. The model does not provide solutions to the real-world scenarios and can be used only as a starting point in this direction.

8 Future work

The future development of the face recognition model could include implementation of solutions to such problems as different lighting conditions of the images and shadows interfering with correct facial feature identifications and classifications. To solve these problems, it would be needed to investigate the algorithms which help rebalance the light and the dark regions of the images and the algorithms that improve the contrast of the images. Additionally, a variety of human poses from different angles and the surrounding complex environment (which was absent in the data set that used for this work) could greatly challenge both face detection and face recognition algorithms in ability to produce satisfying results. Some of these obstacles could be removed with certain image manipulations like image rotation, straightening, normalization, and face alignment. If in the future work a different database of images would be used, then implementation of these mentioned techniques would probably be required to achieve good results. Moreover, the labelled face images could be also connected to a database where additional personal information of each individual would be stored.

9 References

1. Marr, David. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* s.l. : The MIT Press, 2010. 978-0262514620.

2. Ballard, Dana H. and Brown, Christopher M. *Computer vision.* s.l. : Prentice Hall, 1982. 0131653164.

3. Shapiro, Linda and Stockman, George. *Computer vision.* s.l. : Pearson, 2001. 978-0130307965.

4. *Detecting Faces in Images: A Survey.* Yang, Ming-Hsuan, Kriegman, David J. and Ahuja, Narendra. 1, s.l. : IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2002, Vol. 24.

5. *Rapid object detection using a boosted cascade of simple features.* Viola, Paul and Jones, Michael. s.l. : IEEE, 2001, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Vol. 1. 1063-6919.

6. *A Study of Various Face Detection Methods.* Gupta, Ms. Varsha and Sharma, Mr. Dipesh. 5, s.l. : International Journal of Advanced Research in Computer and Communication Engineering, 2014, Vol. 3. 2278-1021.

7. *Robust Real-Time Face Detection.* Viola, Paul and Jones, Michael J. 2, s.l. : Kluwer Academic Publishers, 2004, Vol. 57. 137-154.

8. Cascade Classification. *docs.opencv.org.* [Online] https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html?highlight=grouprectangle.

9. *Summed-Area Tables for Texture Mapping.* Crow, Franklin C. 3, New York : Association for Computing Machinery, 1984, Computer graphics, Vol. 18, pp. 207–212. 0097-8930.
10. *A Theory of the Learnable.* Valiant, G. L. New York : Association for Computing Machinery, 1984. Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing. pp. 436–445. 0897911334.
11. *The strength of weak learnability.* E., Robert and Schapire. Boston : Kluwer Academic Publishers, 1990, Machine Learning, Vol. 5, pp. 197–227.
12. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.* Freund, Yoav and Schapire, Robert E. 1, s.l. : Elsevier, 1997, Journal of Computer and System Sciences, Vol. 55, pp. 119-139. 0022-0000.
13. Kearns, Michael and Vazirani, Umesh. *An introduction to computational learning theory.* s.l. : The MIT Press, 1994. 9780262111935.
14. Kearns, Michael J. *The Computational Complexity of Machine Learning.* s.l. : MIT Press, 1990. 0262111527.
15. *Video-based face recognition: a survey.* Patil, Shailaja A and Deore, Pramod J. 4, 2012, World journal of science and technology, Vol. 2, pp. 136-139. 2231-2587.
16. Kassambara, Alboukadel. *Practical guide to principal component methods in R.* s.l. : CreateSpace Independent Publishing Platform, 2017. 1975721136.
17. Jolliffe, I.T. *Principal Component Analysis.* 2. s.l. : Springer, 2002. 0387954422.
18. Horn, Roger A. and Johnson, Charles R. *Matrix analysis.* 2. s.l. : Cambridge University Press, 2012. 978-0-521-83940-2.
19. *Eigenfaces for recognition.* Turk, Matthew and Pentland, Alex. 1, 1991, Journal of Cognitive Neuroscience, Vol. 3, pp. 71-86. 1530-8898.
20. Overview of visual studio. *docs.microsoft.com.* [Online] https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.
21. Shi, Shin. *Emgu CV Essentials.* s.l. : Packt Publishing Ltd, 2013. 978-1-78355-952-7.
22. Spacek, Libor. Libor Spacek's Facial Images Databases. [Online] http://cmp.felk.cvut.cz/~spacelib/faces/.
23. Baggio, Daniel Lélis, et al. *Mastering OpenCV with Practical Computer Vision Projects.* s.l. : Packt Publishing, 2012. 978-1-84951-782-9.

24. Ekenel, Hazim and Stiefelhagen, Rainer. *Why is facial occlusion a challenging problem?* 2009. pp. 299-308.

25. *A Review: Image Interpolation Techniques for Image Scaling.* Parsania, Pankaj and V.Virparia, Dr. Paresh. 12, 2014, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2. 2320-9801.

26. *Interpolation-Dependent Image Downsampling.* Zhang, Yongbing, et al. 11, s.l. : IEEE Signal Processing Society, 2011, IEEE Transactions on Image Processing, Vol. 20, pp. 3291-3296.