STUDENT ATTENDANCE SYSTEM USING FACE RECOGNITION

Arlind Sylaj

Follow this and additional works at: https://knowledgecenter.ubt-uni.net/etd
STUDENT ATTENDANCE SYSTEM USING FACE RECOGNITION

Bachelor Degree

Arlind Sylaj

December / 2019
Prishtinë
Arlind Sylaj

STUDENT ATTENDANCE SYSTEM USING FACE RECOGNITION

Mentor: Dr. Sc. Bertan Karahoda

December / 2019

This paper has been compiled and submitted to meet the partial requirements for the Bachelor Degree
ABSTRACT

Since traditional students’ attendance system has been known to be time consuming, not accurate and a hard process to follow, this thesis will try to provide a method for solving this problem.

Facial recognition system has started to become widespread over this past decade and the main interest in many industries. Law enforcement agencies are using face recognition to keep communities safer. Retailers are preventing crime and violence. Airports are improving travelers’ convenience and security. And mobile phone companies are using face recognition to provide consumers with new layers of biometric security.

In this thesis another usage of face recognition system is proposed. Using some Computer Vision algorithms, one will try to develop a face recognition system for managing students’ attendance. Taking pictures from laptops camera a data set of 150 pictures will be created, 50 for each student and the system will be trained. One technique for face detection will be proposed based on Haar features and another one for face recognition specifically Local binary patterns histograms. Results from validation data set will also be provided.
ACKNOWLEDGEMENT

The completion of this paper was achieved thanks to the help of many individuals, whom I would like to thank with all my sincerity. My deepest gratitude goes for my professor Bertan Karahoda. His willingness to dedicate his time, suggestions and advices are indeed profoundly appreciated. Without his guidance, critiques and recommendations, this paper would have been unattainable. I extend a great appreciation to the staff of University for Business and Technology, for always providing me with assistance, for their help and support through my years of studies. They have never failed to show generosity and company whenever it was needed. Their contribution will never be forgotten. Above all I want to thank my family for making all this possible. Thank you for your never-ending love, support and motivation. Every success is dedicated to you. Also, a special thanks to my dearest friend, Valona. Thank you for being with me this entire time.

December, 2019

Prishtinë
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a) Two-rectangle feature; b) three-rectangle feature; c) four-rectangle feature</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>Haar features</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>The structure of the Viola–Jones cascade classifier</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Eigenfaces from AT&amp;T Laboratories Cambridge</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>The first component of PCA mixed more than of LDA.</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>LBP operator</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>Extracting the Histograms</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>Software Graphical User Interface</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>Text file to save students name</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>Image data set</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>Train function</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Predict function</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>Detecting student face</td>
<td>27</td>
</tr>
<tr>
<td>14</td>
<td>Attendance sheet</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>Date configuration</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>Adding student as present</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>Predicted confidence for validation set</td>
<td>29</td>
</tr>
<tr>
<td>18</td>
<td>Predicted ID for validation set</td>
<td>29</td>
</tr>
</tbody>
</table>
LIST OF ABREVIATIONS

PCA - Principal Component Analysis
LDA - Linear Discriminant Analysis
SVM - Support Vector Machine
RGB - Red Green Blue
RL - Reinforcement Learning
OpenCV - Open Source Computer Vision
LBPH - Local Binary Pattern Histogram
RFID - Radio-frequency Identification
GUI - Graphical User Interface
XML - Extensible Markup Language
## CONTENTS

**ABSTRACT** ........................................................................................................... III

**ACKNOWLEDGEMENT** ...................................................................................... IV

1 **INTRODUCTION** .......................................................................................... 1

2 **LITERATURE REVIEW** .................................................................................. 2

   2.1 Digital Image processing ............................................................................. 2
   
   2.1.1 Image data types ..................................................................................... 3

   2.2 Machine Learning ......................................................................................... 4

   2.2.1 Problems .................................................................................................. 5

   2.3 Computer Vision ............................................................................................ 8

   2.3.1 Face detection .......................................................................................... 9

   2.3.1.1 Haar cascade ....................................................................................... 11

   2.3.2 Face recognition ...................................................................................... 16

   2.3.2.1 Eigenface .............................................................................................. 16

   2.3.2.2 Fisherface ............................................................................................. 18

   2.3.2.3 Local binary pattern histogram ............................................................. 19

3 **PROBLEM STATEMENT** ................................................................................. 21

4 **METHODOLOGY** .......................................................................................... 22

5 **RESULTS** ...................................................................................................... 23

6 **DISCUSSIONS AND CONCLUSIONS** ......................................................... 30

7 **REFERENCES** .............................................................................................. 31
1 INTRODUCTION

University departments are required to monitor students’ attendance during their studies. It is a crucial part of the teaching process, both for the students and for the instructors as well. Traditionally, this has been done by filling the attendance sheet at the beginning of the lecture. Nevertheless, this is time consuming for the instructor to monitor every student during the entire semester, and also students can just write someone else’s name without the student actually being there. To avoid such things and to make it easier for the students and the instructors, engineers have been trying to implement some technological approaches. For instance, in most of the University of Prishtina departments, they have started to use RFID cards for this purpose. Regardless that this was much more practical than the attendance sheets, it did not come without disadvantages. Students can give the card to their colleagues without the professor noticing it. Another disadvantage is that when entering the classroom, students have to form lines in front of the device, which again is time consuming. Face has been known as the only thing that cannot be duplicated. So, to solve the problem this thesis will propose usage of Computer Vision techniques for face detection and face recognition.
2 LITERATURE REVIEW

2.1 Digital Image processing

Today, almost every area in technical attempts is impacted in some way by digital image processing. An image may be defined as a two-dimensional function, \( f(x,y) \), where \( x \) and \( y \) are spatial(plane) coordinates, and the amplitude of \( f \) at any pair of coordinates \((x,y)\) is called the intensity or gray level of the image at the point. When \( x, y \), and the amplitude values of \( f \) are all finite, discrete quantities, we call the image a digital image. The field of digital image processing refers to processing digital images by means of digital computer. A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used to denote the elements of a digital image. (Gonzalez & Woods, 2002)

The size of the 2-D pixel grid together with the data size stored for each individual image pixel determines the spatial resolution and colour quantization of the image. The representational power (or size) of an image is defined by its resolution. The resolution of an image source (e.g. a camera) can be specified in terms of three quantities:

- Spatial Resolution
  The column (C) by row (R) dimensions of the image define the number of pixels used to cover the visual space captured by the image. This relates to the sampling of the image signal and is sometimes referred to as the pixel or digital resolution of the image. It is commonly quoted as \( C \times R \) (e.g. 640 x 480, 800 x 600, 1024 x 768, etc.)

- Temporal resolution
  For a continuous capture system such as video, this is the number of images captured in a given time period. It is commonly quoted in frames per second (fps), where each individual image is referred to as a video frame (e.g. commonly broadcast TV operates at 25 fps; 25–30
fps is suitable for most visual surveillance; higher frame-rate cameras are available for specialist science/engineering capture).

- **Bit resolution**

This defines the number of possible intensity/colour values that a pixel may have and relates to the quantization of the image information. For instance, a binary image has just two colours (black or white), a grey-scale image commonly has 256 different grey levels ranging from black to white whilst for a colour image it depends on the colour range in use. The bit resolution is commonly quoted as the number of binary bits required for storage at a given quantization level, e.g. binary is 2 bits, grey-scale is 8 bit and colour (most commonly) is 24 bits. The range of values a pixel may take is often referred to as the dynamic range of an image (Solomon & Breckon, 2011).

### 2.1.1 Image data types

The choice of image format used can be largely determined by not just the image contents, but also the actual image data type that is required for storage. In addition to the bit resolution of a given image discussed earlier, a number of distinct image types also exist:

- **Binary images** are 2-D arrays that assign one numerical value from the set \{0;1\} to each pixel in the image. These are sometimes referred to as logical images: black corresponds to zero (an ‘off’ or ‘background’ pixel) and white corresponds to one (an ‘on’ or ‘foreground’ pixel). As no other values are permissible, these images can be represented as a simple bit-stream, but in practice they are represented as 8-bit integer images in the common image formats. A fax (or facsimile) image is an example of a binary image.

- **Intensity or grey-scale images** are 2-D arrays that assign one numerical value to each pixel which is representative of the intensity at this point. As discussed previously, the pixel value range is bounded by the bit resolution of the image and such images are stored as N-bit integer images with a given format.

- **RGB or true-colour images** are 3-D arrays that assign three numerical values to each pixel, each value corresponding to the red, green and blue (RGB) image channel
component respectively. Conceptually, we may consider them as three distinct, 2-D planes so that they are of dimension C by R by 3, where R is the number of image rows and C the number of image columns.

- Floating-point store a floating-point number which, within a given range defined by the floating-point precision of the image bit-resolution, represents the intensity. They may (commonly) represent a measurement value other than simple intensity or colour as part of a scientific or medical image.

We can convert from an RGB colour space to a grey-scale image using a simple transform. Grey-scale conversion is the initial step in many image analysis algorithms, as it essentially simplifies (i.e. reduces) the amount of information in the image. Although a grey-scale image contains less information than a colour image, the majority of important, feature-related information is maintained, such as edges, regions, blobs, junctions and so on. Feature detection and processing algorithms then typically operate on the converted grey-scale version of the image (Solomon & Breckon, 2011).

2.2 Machine Learning

To solve a problem on a computer, we need an algorithm. An algorithm is a sequence of instructions that are carried out to transform the input to the output. For example, one can devise an algorithm for sorting. The input is a set of numbers and the output is their ordered list. For the same task, there may be various algorithms and we may be interested in finding the most efficient one, the one requiring the least number of instructions, memory, or both. For some problems, however, we do not have an algorithm. Predicting customer behavior is one; another is differentiating spam emails from legitimate ones. We know what the input is: an email document that in the simplest case is a text message. We know what the output should be: a yes/no output indicating whether the message is spam or not. But we do not know how to transform the input to the output. What is considered spam changes over time and from individual to individual. What we lack in knowledge, we make up for in data. We
can easily compile thousands of messages, some of which we know to be spam and some of which are not, and what we want is to “learn” what constitutes spam from this sample. In other words, we would like the computer (the machine) to extract automatically the algorithm for this task.

Machine learning is not just a database or programming problem; it is also a requirement for artificial intelligence. A system that is in a changing environment should have the ability to learn; otherwise, we would hardly call it intelligent. If the system can learn and adapt to such changes, the system designer need not foresee and provide solutions for all possible situations.

Artificial intelligence takes inspiration from the brain. There are cognitive scientists and neuroscientists whose aim is to understand the functioning of the brain, and toward this aim, they build models of neural networks and make simulation studies. But artificial intelligence is a part of computer science and our aim is to build useful systems, as in any domain of engineering. So, though the brain inspires us, ultimately, we do not care much about the biological plausibility of the algorithms we develop. (Alpaydin, 2016)

2.2.1 Problems

The range of learning problems is clearly large but researchers have identified an ever-growing number of templates which can be used to address a large set of situations. These templates are discussed above:

- **Binary Classification** is probably the most frequently studied problem in machine learning and it has led to a large number of important algorithmic and theoretic developments over the past century. In its simplest form it reduces to the question: given a pattern \( x \) drawn from a domain \( X \), estimate which value an associated binary random variable \( y \in \{ \pm 1 \} \) will assume. For instance, given pictures of apples and oranges, we might want to state whether the object in question is an apple or an orange. Equally well, we might want to predict whether a home owner might default on his loan, given income data, his credit history, or whether a given e-mail is spam.
or ham. The ability to solve this basic problem already allows us to address a large variety of practical settings. (Smola & Vishwanathan, 2008)

- **Multiclass Classification** is the logical extension of binary classification. The main difference is that now \( y \in \{1, \ldots, n\} \) may assume a range of different values. For instance, we might want to classify a document according to the language it was written in (English, French, German, Spanish, Hindi, Japanese, Chinese, . . .). The main difference to before is that the cost of error may heavily depend on the type of error we make. For instance, in the problem of assessing the risk of cancer, it makes a significant difference whether we misclassify an early stage of cancer as healthy (in which case the patient is likely to die) or as an advanced stage of cancer (in which case the patient is likely to be inconvenienced from overly aggressive treatment). (Smola & Vishwanathan, 2008)

- **Structured Estimation** goes beyond simple multiclass estimation by assuming that the labels \( y \) has some additional structure which can be used in the estimation process. For instance, \( y \) might be a path in an ontology, when attempting to classify webpages, \( y \) might be a permutation, when attempting to match objects, to perform collaborative filtering, or to rank documents in a retrieval setting. Equally well, \( y \) might be an annotation of a text, when performing named entity recognition. Each of those problems has its own properties in terms of the set of \( y \) which we might consider admissible, or how to search this space. (Smola & Vishwanathan, 2008)

- **Regression** is another prototypical application. Here the goal is to estimate a real valued variable \( y \in \mathbb{R} \) given a pattern \( x \). For instance, we might want to estimate the value of a stock the next day, the yield of a semiconductor fab given the current process, the iron content of ore given mass spectroscopy measurements, or the heart rate of an athlete, given accelerometer data. One of the key issues in which regression problems differ from each other is the choice of a loss. For instance, when estimating stock values our loss for a put option will be decidedly onesided. On the other hand, a hobby athlete might only care that our estimate of the heart rate matches the actual on average (Smola & Vishwanathan, 2008)
**Novelty Detection** is a rather ill-defined problem. It describes the issue of determining “unusual” observations given a set of past measurements. Clearly, the choice of what is to be considered unusual is very subjective. A commonly accepted notion is that unusual events occur rarely. Hence a possible goal is to design a system which assigns to each observation a rating. (Smola & Vishwanathan, 2008)

Consequently, the field of machine learning has branched into several subfields dealing with different types of learning tasks. Four parameters along which learning paradigms can be classified are:

- **Supervised learning**, the learning element is given the correct (or approximately correct) value of the function for particular inputs, and changes its representation of the function to try to match the information provided by the feedback. More formally, we say an example is a pair \((x, f(x))\), where \(x\) is the input and \(f(jt)\) is the output of the function applied to \(x\). The task of pure inductive inference (or induction) is this: given a collection of examples of \(f\), return a function \(h\) that approximates \(f\). The function \(h\) is called a hypothesis. (Russell & Norvig, 1995)

- **Unsupervised learning** refers to the problem of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning. Unsupervised learning is closely related to the problem of density estimation in statistics. However unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data. Many methods employed in unsupervised learning are based on data mining methods used to preprocess data. Approaches to unsupervised learning include:
• clustering (e.g., k-means, mixture models, k-nearest neighbors, hierarchical clustering),
• blind signal separation using feature extraction techniques for dimensionality reduction (e.g., Principal component analysis, Independent component analysis, Non-negative matrix factorization, Singular value decomposition). (Hinton & Sejnowski, 1999)

- **Reinforcement learning** subsumes biological and technical concepts for solving an abstract class of problems that can be described as follows: An agent (e.g., an animal, a robot, or just a computer program) living in an environment is supposed to find an optimal behavioral strategy while perceiving only limited feedback from the environment. The agent receives (not necessarily complete) information on the current state of the environment, can take actions, which may change the state of the environment, and receives reward or punishment signals, which reflect how appropriate the agent’s behavior has been in the past. This reward signal may be sparse, delayed, and noisy. The goal of RL is to find a policy that maximizes the long-term reward. Compared to supervised learning, where training data provide information about the correct behavior in particular situations, the RL problem is more general and thus more difficult, since learning has to be based on considerably less knowledge. (Sutton & Barto, 2015)

### 2.3 Computer Vision

Computer Vision has a dual goal. From the biological science point of view, computer vision aims to come up with computational models of the human visual system. From the engineering point of view, computer vision aims to build autonomous systems which could perform some of the tasks which the human visual system can perform (and even surpass it in many cases). Many vision tasks are related to the extraction of 3D and temporal information from time-varying 2D data such as obtained by one or more television cameras, and more generally the understanding of such dynamic scenes. (Huang, 1996)
Of all the visual tasks we might ask a computer to perform, analyzing a scene and recognizing all of the constituent objects remains the most challenging. While computers excel at accurately reconstructing the 3D shape of a scene from images taken from different views, they cannot name all the objects and animals present in a picture, even at the level of a two-year-old child. There is not even any consensus among researchers on when this level of performance might be achieved. (Szeliski, 2010)

Cameras are everywhere and the number of images uploaded on internet is growing exponentially. We have images on Instagram, videos on YouTube, feeds of security cameras, medical and scientific images. Computer vision is essential because we need to sort through these images and enable computers to understand their content. Computer vision can be used in number of areas such as 3d urban modeling, scene recognition, face detection and recognition, optical character recognition, mobile visual search, self-driving cars, automatic checkout, vision-based interaction, augment reality, virtual reality (Krishna, 2017).

Because the purpose of this thesis is to develop a student attendance system, I will explain in more details only face detection and face recognition.

2.3.1 Face detection

Face detection can be defined as the goal to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face. (Yang, Kriegman, & Ahuja, 2002)

In general, detectors can make two types of errors: false negatives in which faces are missed resulting in low detection rates and false positives in which an image region is declared to be face, but it is not. A fair evaluation should take these factors into consideration since one can tune the parameters of one’s method to increase the detection rates while also increasing the number of false detections. (Yang, Kriegman, & Ahuja, 2002)

The challenges associated with face detection can be attributed to the following factors:

- **Pose.** The images of a face vary due to the relative camera-face pose (frontal, 45-degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded.
- Presence or absence of structural components. Facial features such as beards, mustaches, and glasses may or may not be present and there is a great deal of variability among these components including shape, color, and size.

- Facial expression. The appearance of faces is directly affected by a person’s facial expression.

- Occlusion. Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.

- Image orientation. Face images directly vary for different rotations about the camera’s optical axis.

- Imaging conditions. When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face. (Yang, Kriegman, & Ahuja, 2002)

There are four techniques to detect faces from a single intensity or color image:

1. Knowledge-based methods. These rule-based methods encode human knowledge of what constitutes a typical face. Usually, the rules capture the relationships between facial features. These methods are designed mainly for face localization.

2. Feature invariant approaches. These algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary, and then use these to locate faces. These methods are designed mainly for face localization.

3. Template matching methods. Several standard patterns of a face are stored to describe the face as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection. These methods have been used for both face localization and detection.

4. Appearance-based methods. In contrast to template matching, the models (or templates) are learned from a set of training images which should capture the representative variability of facial appearance. These learned models are then used for detection. These methods are designed mainly for face detection. (Yang, Kriegman, & Ahuja, 2002)
2.3.1.1 Haar cascade

There were many attempts to respond to real-time constraints for object detection. Viola and Jones have come up with a method of rectangular Haar-like features with AdaBoost learning algorithm combined with a cascade of strong classifiers. The proposed object detection application can be deployed in different platforms; it can be deployed on a high-performance platform as well as in mobile platform. It can also be used in surveillance systems with distributed cameras and a back-end server in which the detection takes place. It can also be used in mobile devices equipped with camera and processor. A highly short response time in terms of detection is essential for such systems.

There are three main contributions of this face detection framework:

1. Haar-like feature

Haar feature-based cascade classifiers, classifies images based on the value of simple features. There are many motivations for using features rather than the pixels directly. The simple features used are reminiscent of Haar basis functions which have been used by Papageorgiou et al. (1998). More specifically, we use three kinds of features. The value of a two-rectangle feature is the difference between the sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent (see Fig 1 (a)). A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle (see Fig 1 (b)). Finally, a four-rectangle feature computes the difference between diagonal pairs of rectangles (see Fig 1 (c)).

![Figure 1. a) Two-rectangle feature; b) three-rectangle feature; c) four-rectangle feature](image-url)
1.1 Integral images

The primary reason for using an integral image is the improved execution speed for computing box filters. Employment of the integral image eliminates computationally expensive multiplications for box filter calculation, reducing it to three addition operations. This allows all box filters to be computed at a constant speed, irrespective of their size; this is a major advantage for computer vision algorithms, especially feature detection techniques which utilize multi-scale analysis. (Clark, Ehsan, Rehman, & McDonald-Maier, 2014)

Using the integral image any rectangular sum can be computed in four array references. Clearly the difference between two rectangular sums can be computed in eight references. Since the two-rectangle features defined above involve adjacent rectangular sums they can be computed in six array references, eight in the case of the three-rectangle features, and nine for four-rectangle features. (Viola & Jones, 2004)

2. Training classifier

Problems in machine learning often suffer from the curse of dimensionality, each sample may consist of a huge number of potential features (for instance, there can be 162,336 Haar features, as used by the Viola–Jones object detection framework, in a 24×24 pixel image window), and evaluating every feature can reduce not only the speed of classifier training and execution, but in fact reduce predictive power, per the Hughes Effect. Unlike neural networks and SVMs, the AdaBoost training process selects only those features known to improve the predictive power of the model, reducing dimensionality and potentially improving execution time as irrelevant features need not be computed. In this system a variant of AdaBoost is used both to select the features and to train the classifier. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple learning algorithm (e.g., it might be used to boost the performance of a simple perceptron). It does this by combining a collection of weak classification functions to form a stronger classifier. In the language of boosting the simple learning algorithm is called a weak learner. The learner is called weak because we do not expect even the best
classification function to classify the training data well (i.e. for a given problem the best perceptron may only classify the training data correctly 51% of the time). In order for the weak learner to be boosted, it is called upon to solve a sequence of learning problems. After the first round of learning, the examples are re-weighted in order to emphasize those which were incorrectly classified by the previous weak classifier. The final strong classifier takes the form of a perceptron, a weighted combination of weak classifiers followed by a threshold.

The key advantage of AdaBoost as a feature selection mechanism, over competitors such as the wrapper method, is the speed of learning. Using AdaBoost a 200-feature classifier can be learned in $O(MNK)$ or about $10^{11}$ operations. One key advantage is that in each round the entire dependence on previously selected features is efficiently and compactly encoded using the example weights. These weights can then be used to evaluate a given weak classifier in constant time.

The weak classifier selection algorithm proceeds as follows. For each feature, the examples are sorted based on feature value. The AdaBoost optimal threshold for that feature can then be computed in a single pass over this sorted list. For each element in the sorted list, four sums are maintained and evaluated: the total sum of positive example weights $T^+$, the total sum of negative example weights $T^-$, the sum of positive weights below the current example $S^+$ and the sum of negative weights below the current example $S^-$. The error for a threshold which splits the range between the current and previous example in the sorted list is:

$$e = \min (S^+ + (T^- - S^-), S^- + (T^+ - S^+)),$$

or the minimum of the error of labeling all examples below the current example negative and labeling the examples above positive versus the error of the converse. These sums are easily updated as the search proceeds.

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks (see Fig. 2). This feature is relatively large in comparison with the detection sub-window, and should be somewhat insensitive to size and location of the face. The
second feature selected relies on the property that the eyes are darker than the bridge of the nose.

In summary the 200-feature classifier provides initial evidence that a boosted classifier constructed from rectangle features is an effective technique for face detection. In terms of detection, these results are compelling but not sufficient for many real-world tasks. In terms of computation, this classifier is very fast, requiring 0.7 seconds to scan a 384 by 288 pixel image. Unfortunately, the most straightforward technique for improving detection performance, adding features to the classifier, directly increases computation time. (Viola & Jones, 2004)

![Figure 2. Haar features](image)

3. Constructing a cascade

The cascade classifier consists of a list of stages, where each stage consists of a list of weak learners. The system detects objects in question by moving a window over the image. Each stage of the classifier labels the specific region defined by the current location of the window as either positive or negative – positive meaning that an object was found or negative means that the specified object was not found in the image. If the labelling yields a negative result, then the classification of this specific region is hereby complete and the location of the window is moved to the next location. If the labelling gives a positive result, then the region moves on to the next stage of classification. The classifier yields a final verdict of positive, when
all the stages, including the last one, yield a result, saying that the object is found in the image.

A true positive means that the object in question is indeed in the image and the classifier labels it as such a positive result. A false positive means that the labelling process falsely determines, that the object is located in the image, although it is not. A false negative occurs when the classifier is unable to detect the actual object from the image and a true negative means that a non-object was correctly classified as not being the object in question. In order to work well, each stage of the cascade must have a low false negative rate, because if the actual object is classified as a non-object, then the classification of that branch stops, with no way to correct the mistake made. However, each stage can have a relatively high false positive rate, because even if the n-th stage classifies the non-object as actually being the object, then this mistake can be fixed in n+1-th and subsequent stages of the classifier (Soo, 2014)

![Figure 3. The structure of the Viola–Jones cascade classifier](image)

There is a hidden benefit of training a detector as a sequence of classifiers which is that the effective number of negative examples that the final detector sees can be very large. One can imagine training a single large classifier with many features and then trying to speed up its running time by looking at partial sums of features and stopping the computation early if a partial sum is below the appropriate threshold. One drawback of such an approach is that the training set of negative examples would
have to be relatively small (on the order of 10,000 to maybe 100,000 examples) to make training feasible. With the cascaded detector, the final layers of the cascade may effectively look through hundreds of millions of negative examples in order to find a set of 10,000 negative examples that the earlier layers of the cascade fail on. So, the negative training set is much larger and more focused on the hard examples for a cascaded detector. (Viola & Jones, 2004)

### 2.3.2 Face recognition

Face recognition or face identification compares an input image (probe) against a database (gallery) and reports a match, if any. The purpose of face authentication is to verify the claim of the identity of an individual in an input image. The following methods are used to face recognition:

1. Holistic Matching Methods
2. Feature-based (structural) Methods
3. Hybrid Methods

There are three most commonly used algorithms for face recognition, which are also easy to implement using OpenCV:

#### 2.3.2.1 Eigenface

Eigenface is based on PCA that classify images to extract features using a set of images. It is important that the images are in the same lighting condition and the eyes match in each image. Also, images used in this method must contain the same number of pixels and in grayscale. In mathematical terms, we have to find the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images, treating an image as a point (or vector) in a very high dimensional space. The eigenvectors are ordered, each one accounting for a different amount of the variation among the face images. These eigenvectors can be thought of as a set of features that together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that we can display the eigenvector as a sort of ghostly face which we call an eigenface.
Each eigenface deviates from uniform gray where some facial feature differs among the set of training faces; they are a sort of map of the variations between faces. Each individual face can be represented exactly in terms of a linear combination of the eigenfaces. Each face can also be approximated using only the “best” eigenfaces—those that have the largest eigenvalues, and which therefore account for the most variance within the set of face images. The best M eigenfaces span an M-dimensional subspace—“face space”—of all possible images.

This approach to face recognition involves the following initialization operations:

1. Acquire an initial set of face images (the training set).
2. Calculate the eigenfaces from the training set, keeping only the M images that correspond to the highest eigenvalues. These M images define the face space. As new faces are experienced, the eigenfaces can be updated or recalculated.
3. Calculate the corresponding distribution in M-dimensional weight space for each known individual, by projecting their face images onto the “face space.”
These operations can also be performed from time to time whenever there is free excess computational capacity. Having initialized the system, the following steps are then used to recognize new face images:

1. Calculate a set of weights based on the input image and the M eigenfaces by projecting the input image onto each of the eigenfaces.
2. Determine if the image is a face at all (whether known or unknown) by checking to see if the image is sufficiently close to “face space.”
3. If it is a face, classify the weight pattern as either a known person or as unknown.
4. (Optional) Update the eigenfaces and/or weight patterns.
5. (Optional) If the same unknown face is seen several times, calculate its characteristic weight pattern and incorporate into the known faces. (Turk & Pentland, 1991)

If the same face is analysed under different lighting conditions, it will mix the values when distribution is calculated and cannot be effectively classified. This makes different lighting conditions pose a problem in matching the features as they can change dramatically.

2.3.2.2 Fisherface

Fisher’s Linear Discriminant is a “classical” technique in pattern recognition, first developed by Robert Fisher in 1936 for taxonomic classification. Depending upon the features being used, it has been applied in different ways in computer vision and even in face recognition. (Belhumeur, Hespanha, & Kriegman, 1997)

When reducing dimensions, PCA looks at the greatest variance, while LDA, using labels, looks at an interesting dimension such that, when you project to that dimension you maximise the difference between the mean of the classes normalised by their variance. LDA maximises the ratio of the between-class scatter and within-class scatter matrices. Due to this, different lighting conditions in images has a limited effect on the classification process using LDA technique. Eigenface maximises the variations while Fisherface maximises the mean distance between and different classes and minimizes variation within classes. This enables LDA to differentiate between feature classes better than PCA and can be observed in Figure 5.
Furthermore, it takes less amount of space and is the fastest algorithm in this project. Because of these, PCA is more suitable for representation of a set of data while LDA is suitable for classification. (Dinalankara, 2017)

![PCA vs LDA](image)

**Figure 5. The first component of PCA mixed more than of LDA.**

### 2.3.2.3 Local binary pattern histogram

The LBP operator (Figure 6) is one of the best performing texture descriptors and it has been widely used in various applications. It has proven to be highly discriminative and its key advantages, namely its invariance to monotonic gray level changes and computational efficiency, make it suitable for demanding image analysis tasks.

![LBP Operator](image)

**Figure 6. LBP operator**

The LBP operator was originally designed for texture description. The operator assigns a label to every pixel of an image by thresholding the 3x3-neighborhood of each pixel with the center pixel value and considering the result as a binary number. Then the histogram of the labels can be used as a texture descriptor.
On the LBP approach for texture classification, the occurrences of the LBP codes in an image are collected into a histogram. The classification is then performed by computing simple histogram similarities. However, considering a similar approach for facial image representation results in a loss of spatial information and therefore one should codify the texture information while retaining also their locations. One way to achieve this goal is to use the LBP texture descriptors to build several local descriptions of the face and combine them into a global description. Such local descriptions have been gaining interest lately which is understandable given the limitations of the holistic representations. These local feature-based methods are more robust against variations in pose or illumination than holistic methods.

The basic methodology for LBP based face description proposed by Ahonen et al. (2006) is as follows: The facial image is divided into local regions and LBP texture descriptors are extracted from each region independently. The descriptors are then concatenated to form a global description of the face, as shown in Fig. 7.

![Figure 7. Extracting the Histograms](image)

This histogram effectively has a description of the face on three different levels of locality: the LBP labels for the histogram contain information about the patterns on a pixel-level, the labels are summed over a small region to produce information on a regional level and the regional histograms are concatenated to build a global description of the face.

It should be noted that when using the histogram-based methods the regions do not need to be rectangular. Neither do they need to be of the same size or shape, and they do not necessarily have to cover the whole image. It is also possible to have partially overlapping regions. (Pietikäinen, 2010)
3 PROBLEM STATEMENT

Technology has been the solution to many obstacles that mankind has had for a long time. New discoveries from different fields, such as artificial intelligence, machine learning and computer vision, have made researchers inquisitive about how can these systems transform the way we live. These solutions can be from the very mundane ones, like detection of spam emails to those more complex like pattern recognition of any kind. One of the problems that can be solved with computer vision techniques is the students’ attendance. As it has been stated earlier in this thesis, in every university, students’ attendance is monitored every semester of their studies. In Kosovo, in almost every educational institution, this process is accomplished manually, through the attendance sheet, which is signed by the students. The professor and the corresponding department track the sheets of every lecture and by the end of the semester they check and use it as an aid for the evaluation procedure. This process, as any other manual process, requires a lot a time and consideration; for there can be times when the sheet is lost or when the number of students is enormous. Some of the faculties in Kosovo have implemented various types of automatic systems. These systems have mainly used RFID technology, whose costs of implementations are irrelevant with the provided results. But one thing that has not yet been applied in Kosovo is biometric identification. This approach will have low costs of implementations, since one will not have to make and ID card for every student, it would be easy for the IT of the faculties to use, and it would be the answer to all the problems for monitoring the student attendance; all of this, only by a pc small as a Raspberry pi and a camera.
4 METHODOLOGY

In order to accomplish this thesis, a data set with images of people faces had to be created. This data set can be created directly from the application that is developed in Python, using OpenCV and its libraries. 50 images of my face and two different football players are captured and then the algorithm is trained. Before proceeding to training algorithm, images are scaled and converted to grayscale.

For the face detection part, OpenCV provides pretrained Haar cascade models trained by Intel Corporation to detect faces and eyes in an image.

Local Binary Patterns Histograms is the algorithm that is used for face recognition because of results during testing and illumination invariance.

Google Cloud Platform with Google Sheets API is used to create attendance sheet and add students when identified.
5 RESULTS

To make system easier to use, and to allow user easily to interact with program a GUI has been created. This GUI contains two elements: a text field and five buttons.

![Figure 8. Software Graphical User Interface](image)

- **Register**

  Image data set of this program can be created directly when writing student name in text field and clicking “Register” button, which will open computer’s default camera. Using OpenCV function “COLOR_BGR2GRAY” video stream is converted to gray. The average pixel value is calculated and a threshold is used to ensure that images have enough brightness.

  A text file with the name “Names.txt” is created and will store students name starting with their indexis.
If a face is detected using Haar cascade and pixel average value is above threshold, image is saved in folder “dataSet”. 50 images for every student will be taken every 300 milliseconds. The final data set will contain student images with their indexes relative to indexes in text file “Names.txt”.

Figure 9. Text file to save students name

Figure 10. Image data set
- **Train**

If all students are added and are photographed, training process can start by simply clicking “Train” button.

```python
path = 'dataSet'
IDs, FaceList = getImageWithID(path)

#Train
LBPHFace.train(FaceList, IDs)
print('LBPH FACE RECOGNISER COMPLETE')
LBPHFace.save('Recogniser/trainingDataLBPH.xml')
print('FILE HAS BEEN SAVED')
```

**Figure 11. Train function**

Using “train method” a FaceRecognizer with given data and associated labels can be trained. This method takes two parameters:

1. **Src** - The training images, that means the faces you want to learn. The data has to be given as a vector
2. **Labels** - The labels corresponding to the images have to be given either as a vector of int or a Mat type (n-dimensional dense array).

When the process of training is finished, which takes about 2 – 2.5 seconds a “XML” file will be created in folder “Recogniser”. This file will be used later for recognition.

Since the size of the input images for the input layer must be the same with the size of inputs for the output layer, the images are resized all in the same size and converted into grayscale images.

- **Predicting**

Main process of this system is the process of predicting faces of students and sign them in attendance sheet. This process starts by clicking “START” button which will open default camera and start predicting students when they are detected.

Firstly, face and eye cascades are imported from folder “Haar”, then faces are detected using photo frames from camera converted into gray as first parameter, scale factor of 1.3 which is relatively small step for resizing and minimum neighbor parameter set to 5 neighbors each.
candidate rectangle should have to retain it. This method returns the positions of detected faces as Rect(x, y, w, h). For the process of predicting to start eyes has to be detected and also be inside the rectangle that return face detection method. The face is considered as detected only if eyes are inside face.

Recognizer object created from “LBPHFaceRecognizer_create” takes five parameters:

1. **Radius** - The radius used for building the Circular Local Binary Pattern. This value is set to 2. The greater the radius, the smoother the image but more spatial information you can get.

2. **Neighbors** - The number of sample points to build a Circular Local Binary Pattern from. An appropriate value is to use 8 sample points but the more sample points you include, the higher the computational cost.

3. **Grid_x** - The number of cells in the horizontal direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.

4. **Grid_y** - The number of cells in the vertical direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.

5. **Threshold** - The threshold applied in the prediction. If the distance to the nearest neighbor is larger than the threshold, this method returns -1.

Finally predict method will return student name and confidence. Method “ID2Name” will check if confidence is not above a threshold and the name is one of the students’ name in our textfile.

```python
ID, conf = recognise.predict(gray_face)
NAME = NameFind.ID2Name(ID, conf)
NameFind.DispID(x, y, w, h, NAME, gray)
```

**Figure 12. Predict function**
Figure 13. Detecting student face

- Attendance sheet

Attendance sheet is created beforehand in Google Sheets and has been shared with a specified email provided by Google Sheets API on Google Cloud Platform. In this way we will be able to have access to this sheet. Google Service Account Credentials are imported from “json” file in order to be identified.

Figure 14. Attendance sheet

There are 14 columns totally created, one for student name, another one for total student presence during semester and the left ones for every week during semester.
When the face is detected and the “ID2Name” is called to check for student name, depending on the date of that day the respective column will be filled.

```python
    today = date.today()
    if (str(today) == "2019-12-04"): 
        kolona=2
    if (str(today) == "2019-12-05"): 
        kolona=3
    if (str(today) == "2019-12-06"): 
        kolona=4
    if (str(today) == "2019-12-07"): 
        kolona=5
    if (str(today) == "2019-12-08"): 
        kolona=6
    if (str(today) == "2019-12-09"): 
        kolona=7
    if (str(today) == "2019-12-10"): 
        kolona=8
    if (str(today) == "2019-12-11"): 
        kolona=9
    if (str(today) == "2019-12-12"): 
        kolona=10
    if (str(today) == "2019-12-13"): 
        kolona=11
    if (str(today) == "2019-12-14"): 
        kolona=12
    if (str(today) == "2019-12-15"): 
        kolona=13
```

**Figure 15. Date configuration**

If the predicted face corresponds to a student a string with value “1” is sent to Google Attendance Sheet and a flag is set to make sure the student presence field is filled only once.

```python
    if (Names[ID-1] == "Leo Messi" and lmflag==1):
        sheet = client.open("Attendance").sheet1
        sheet.update_cell(7, kolona, "1")
        lmflag=0
        print("Leo Messi - PRESENT")
```

**Figure 16. Adding student as present**

- **Accuracy**

In order to calculate accuracy and to measure performance of this system, another function is created which takes 150 images from folder “testdataSet”, different from images that it was trained and for each image it requests predicted id and confidence. For each image, result is saved in a text file and accuracy is calculated. The highest accuracy shown is 78.40 %, precision is 100 % and recall is 64 %. 
In Figure 22 can be seen the values of the test set and the predicted confidence that the system made. Because the last 50 images are images of unknown students’ confidence starts to increase.

![Graph of predicted confidence for validation set](image)

**Figure 17.** Predicted confidence for validation set

In Figure 23 can be seen predicted ID values for given test set. First 75 images are images of known students so the predicted ID are 100% correct.

![Graph of predicted ID for validation set](image)

**Figure 18.** Predicted ID for validation set
6 DISCUSSIONS AND CONCLUSIONS

Face recognition system can solve the problem of students’ attendance on educational institutions. This can be done by using some Computer Vision techniques and algorithms first to detect a face then to predict whose face is. This system is done using Haar cascade classifier to detect faces, with a pre trained cascade from Intel, LBPH algorithm from OpenCV to recognize faces, and Google Sheets API for updating student’s attendance sheet. After training algorithms with 50 images for student, process of recognizing can start by using GUI of the software.

Training the system with values of 1 for radius, 1 for neighbors, 15 for threshold the maximum accuracy we can get is 78.40 %.

This accuracy value and also precision value of 100 % show us that this system combined with some other new features can be used to manage students’ attendance.

To increase the value of the accuracy for a better result, usage of adaptive threshold can be proposed. Also system can be trained with 100 pictures for student but this will increase computational cost drastically.

This system can be used as a pilot project including usage of two other algorithms, Eigenfaces and Fisherface and the final result to be combinated result of the three algorithms. This will give the system more reliability.

Furthermore, this system can be improved by using databases for saving students name, students image for updating training files, and saving training files.
7 REFERENCES


