Kingdom of Saudi Arabia
Ministry of Education
Qassim University
College of Computer
Information Technology Dept.

المملكة العربية السعودية
وزارة التعليم
جامعة القصيم
كلية الحاسب
قسم تقنية المعلومات

Face Detection Technique

Students:

Alaa Alromaih	**381211262**
Ameera Alorf	**391202254**
Reema Alsmaeil	**391202186**
Ruba Almuziraee	**391202532**
Wesam Alqaadan	**391202510**

Supervisor:

Dr.Waleed Albattah

A project report submitted in partial fulfilment of the requirements for B.Sc. degree in Information Technology.

Qassim-Saudi Arabia
1444 (2022/2023)

Certificate

It is certified that the project report has been prepared and written under my direct supervision and guidance. The project report is approved for submission for evaluation.

Dr.Waleed Albattah

Dedication

This work is dedicated to everyone who has taught us during our educational journey, to everyone who has supported and walked beside us, to our families and friends who have assisted us, and to every student seeking knowledge.

Alaa, Ameera, Reema, Ruba and Wesam

Acknowledgement

First and foremost, we thank Allah for giving us the ability and power to do this project. It was the long-awaited moment that we always wished to reach and conclude this project. During this time, many people around us provide us who provide us all with support and encouragement to do this work. First of all, our supervisor, Dr. Waleed Albattah, we would like to thank him for all his supervision, support, and efforts that he provided for our team. Also, we would like to thank our families, especially our parents, for their support and encouragement throughout our studies. Last but not least, we would like to thank each member of our team for their efforts.

Alaa, Ameera, Reema, Ruba and Wesam

Abstract

Face Detection Technique can detect faces in images and achieve high performance on detection rates. Face Detection technologies and applications have recently developed and improved as a result of advancements in computer software and hardware. Nowadays, it is employed in every part of our lives. Face Detection technologies are being employed in a wide range of fields, including access control, law enforcement, entertainment, personal safety, and many more. Based on OpenCV and a machine learning software library that is mainly oriented at computer vision. This library consists of more than 2500 algorithms which contain machine learning tools for classification and clustering, image processing, and vision algorithms. OpenCV library has several built-in pre-trained classifiers developed individually for detecting faces, eyes, and smiles. In this project, we choose the Haar cascade Classifier rather than the Local Binary Pattern (LBP) as one of the OpenCV algorithms. LBP features to concentrate on detection hit rate and detection speed. As a result, we selected the Haar cascade Classifier because we aimed for accuracy in detecting faces in images rather than the LBP speed rates. This algorithm is the most efficient and reliable for the implementation phase of face detection systems. It has been a success that we have accomplished most of what we had planned at the beginning of the project. Finally, we verified that the system worked as planned by testing it.

Keywords: Face detection, OpenCV, Machine Learning, Haar cascade, LBP.

Table of Contents

List of Figures

List of Tables

Abbreviations

API Application Programming Interface

COLOR FERET database color Facial Recognition Technology Database

FN False Negative

FP False Positive

HAAR Haar Cascades.

IDE Integrated Development Environment

KNN Nearest Neighbor

LBP Local Binary Pattern

LDA Linear Discriminate Analysis

ML Machine Learning

OpenCV Open Source Computer Vision Library

PCA Principle Component Analysis

RAD Rapid Application Development

SVM Support Vector Machine

TN True Negative

TP True Positive

Chapter 1

INTRODUCTION

INTRODUCTION

1.1 Introduction

Finding the locations and dimensions of all objects in an image that fall into a given class is the main task of a face detection technique. Face detection is frequently misused. However, face detection means being used in various applications that identify human faces in digital images [1]. Face detection uses algorithms to separate faces from all the other features that may be present in an image. Different approaches are used during the facial detection process. Each process has its advantages and disadvantages, as well as some constraints that set it apart from other approaches [2].
This chapter introduces information about face detection and demonstrates the scope of this project, the problem specification, motivation, and the project's key goals, objectives, and study plan.

1.2 Problem Specification and Motivation

Through our look at the research related to our project on the problems facing the face detection process, the main problem that caused concern for sensitive systems that need high accuracy in the face detection process, such as security systems, is how accurate the detection is. Not its speed. Still, there is scope to research this subject and find more efficient implementation methods for sensitive systems that we can count on. The motivation behind this project is to provide or develop a system that will use the system that would detect the person's face, focusing on high-accuracy implementation utilizing the tool in the Open Source Computer Vision Library(OpenCV) called Open Face and the Python programming language in the Machine Learning(ML) domain [3].

1.3 Goals and Objectives

It is possible to achieve the project goal by achieving the following objectives:

- Display appropriate studies of face detection methods using OpenCV classifiers. It will help to understand the face detection methods, their merits and demerits, and the problems they face.
- A suitable system is to be proposed, keeping in mind the requirement that it be accurate.
- Detect people's faces by using OpenCV.
- Face detection using a pre-trained classifier for faces and eyes called (HAAR) Cascade.
- Verify and implement the developed system by analyzing existing studies and solutions.

1.4 Study Scope

The purpose of the study focuses on automatic human face detection from images in surveillance and biometric applications that are being employed in a wide range of fields, including access control, law enforcement, entertainment, personal safety, and many more [4].

1.5 Study Plan and Schedule

In this section, we will show the work breakdown structure followed by a Gantt chart as shown in Table 1.1 and Figure 1.1

Table 1.1: Work breakdown structure

WBS	TASK	START DATE	END DATE
1	Graduation Project	28/8/2022	*/*/2022
1.1	Phase 1	28/8/2022	*/*/2022
1.1.1	Research	28/8/2022	3/9/2022
1.1.2	Read the overview of the subject	4/9/2022	10/9/2022
1.1.3	Search and read related work	4/9/2022	10/9/2022
1.1.4	Summarizes related works	4/9/2022	10/9/2022
1.1.5	Defining the scope	11/9/2022	17/9/2022
1.1.6	Defining Problem domain	11/9/2022	17/9/2022
1.1.7	Write a literature review and background	18/9/2022	13/10/2022
1.1.8	Defining the gap	9/10/2022	13/10/2022
1.1.9	Exploring methodological approaches	16/10/2022	22/10/2022
1.1.10	Methodology	23/10/2022	5/11/2022

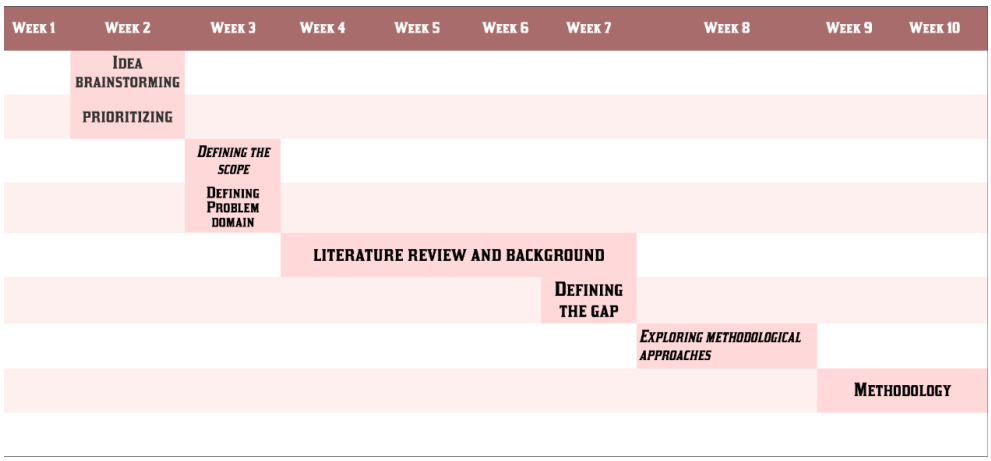

Figure 1.1: Gantt Chart

1.6 Organizing the Chapters

The project document contains six chapters:

- **Chapter 1** This chapter introduces the main points of the project, including the problem specification and motivation, goals and objectives, study scope, study plan and schedule, and the organization of the chapters.
- **Chapter 2** This chapter presents the literature review and discusses the previous related work, the project background, and the proposed system.
- **Chapter 3** This chapter presents the methodology used, which focuses on gathering and analyzing data as well as gathering, analyzing, and organizing needs.

- **Chapter 4** This chapter describes our system's implementation, including its steps and procedures, as well as system testing.
- **Chapter 5** This chapter discusses the final results of the system, including the major findings and discussions related to the proposed work.
- **Chapter 6** This chapter includes the conclusions and future work.

Chapter 2

LITERATURE REVIEW

LITERATURE REVIEW

2.1 Introduction

In this chapter, we will discuss some background information on the subject. This includes ML, Computer Vision, OpenCV, and Face Detection. It also shows several related works. Furthermore, we briefly explain the proposed work.

2.2 Background

2.2.1 Machine Learning

Machine Learning (ML) is made possible by the accessibility and availability of a massive volume of data gathered from sensors and the internet. The concept of ML proves and spreads the truth that a computer can get better over time [5]. Salim's [5] study has found three types of ML strategies in computer vision: supervised, unsupervised, and semi-supervised. Supervised learning is what we are going to use in this project. The supervised learning of a deep convolutional neural network recognizes faces with a large set of face images [6]. Many different methodologies and many other benchmarks are used to assess them. This inspired us to choose and compare the main state-of-the-art methods for automatic face detection in a succinct but trustworthy manner [7]. We know that such a representation is sensitive to changes in expression, illumination, and poses. The complexities of calculation and storage and the procedure for generating the examples are very complicated. The requirements of these systems may include the addition of ML capability to enable a machine to evolve through a learning process and perform tasks that are difficult or impossible to complete by more conventional algorithmic means [8]. In this project, we use ML for face detection by using OpenCV.

2.2.2 OpenCV Algorithm

The abbreviation "OpenCV" stands for "open source computer vision." It is a massive open-source library for computer vision, ML, and image processing. OpenCV is compatible with a broad range of programming languages, including Python [9]. ML Algorithm is used in OpenCV for searching faces within pictures. Because faces are all seen as complex, they cannot be determined by a simple test. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called Classifiers [10]. We use OpenCV in our project because it furnishes us with Two pre-prepared and fit to be utilized for face detection classifiers, HAAR Classifier and LBP Classifier. These classifiers can determine if an image has a face or not [11]. OpenCV already contains pre-trained sample sets of classifiers for faces, eyes, smiles, etc.., to detect faces in the image, so when using it, we don't need to use another dataset. In this project, we learned to detect faces quickly, and we recommend using OpenCV because you can easily set up your machine and execute your program within milliseconds.

2.2.3 Computer Vision Pattern Face Detection

By organizing conferences, workshops, group discussions, experimentation, and real-world implementation, Western countries have shown great interest in the fields of ML, computer vision, and pattern recognition [5]. One of the visual tasks that people can easily perform is face detection. However, this task is challenging in the context of computer vision. The issue can be described in general terms as follows: Find and localize an unknown number (if any) of faces in an image. Segmenting, extracting, and maybe verifying faces and possible facial features from an uncontrolled background is the problem's solution. A face detection system should

be capable of acting as a visual front-end processor regardless of lighting, orientation, or camera distance [12]. Making machines that can see is the goal of computer vision research and technology. It focuses on the theory, creation, and use of algorithms that can analyze visual data automatically to identify objects, track and recover their shape, and spatial layout [13].

Chellappa et al.[14] discovered that learning how their visual system functions and converting these mechanisms into real systems is of tremendous importance to those who develop computer vision algorithms and system designers. One of the conclusions reached by Chellappa et al.[14] was that developers of face recognition algorithms and systems should be aware of pertinent psycho-physical and physiological studies. However, they should exercise caution and only use relevant or applicable from a practical implementation point of view.

2.2.4 Face Detection In Image

Face detection is the premise of almost everyone's face application, and its purpose is to locate the position of a face from an image [15]. The main objective of face detection is to determine if there are any faces in the image. This may seem like a simple task for humans, but for computers, it is challenging, making it one of the most studied research topics in recent years [16]. Since 2002, with the open source framework known as OpenCV, face detection can be performed fairly easily and reliably. This framework has an in-built face detector that works in roughly 90-95% clear photos of a person looking directly at the camera. However, detecting a person's face when viewed from an angle can be challenging, necessitating 3D head pose estimation in some cases. Additionally, lack of proper brightness in an image, increased contrast in shadows on the face, blurry photos, people wearing glasses, and other factors can make it very difficult to identify faces [17]. The HAAR classifier is used by this system to identify faces in images [18]. One of the most popular algorithms for facial detection is "HAAR". It is a fast algorithm that provides high accuracy at a lower computational cost [19] and an efficient object detection method commonly implemented using OpenCV. It is a machine learning-based approach where a cascade function is trained using a large number of both positive and negative images [9]. Figure 2.1 shows the view of the HAAR classifier.

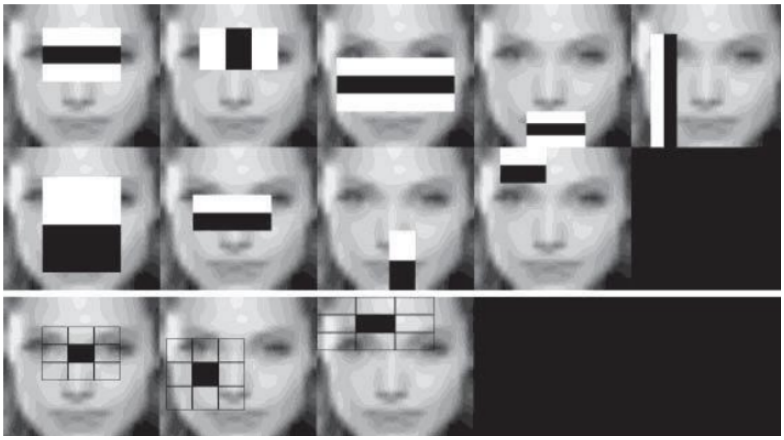

Figure 2.1: View of HAAR classifier

2.2.5 Challenges in Face Detection

Face detection challenges cause the accuracy and detection rate of face detection to decline. Complex backgrounds, too many faces in an image, strange emotions, illuminations, low resolution, face occlusion, skin tone, distance, and orientation are some of these difficulties [16]. 2.2 shows the various categories of challenges for face detection.

- Weird gestures and face identification can be complex since a human face in a picture can make an expression different from the usual.
- Face obscuration face occlusion is when an item obscures the face. It might be anything, like spectacles, a scarf, a hand, hair, a hat, etc. It also lowers the rate of face detection.
- The image's lighting effects might not be consistent. There may be very high light in some regions of the picture and very low illumination in others.
- Complex background When there are many objects in an image, the accuracy, and rate of face detection decrease.
- The image has too many faces, which indicates that there are too many human faces, making face detection difficult.
- Reduced clarity Face recognition may be difficult due to the image's very low resolution.
- Skin hue Skin tone varies according to geography. Chinese people have a distinct skin tone from Africans, who in turn have a different skin tone than Americans, and so forth. Changes in skin tone make it difficult to distinguish faces.
- The distinction of too much space between the camera and a person's face might make it harder to see their face in a photograph.
- Orientation, The angle-pose of the face is called face orientation. Additionally, it lowers facial detection precision and detection rate.

Figure 2.2: Various categories of challenges for face detection

2.3 Related work

Face detection has made significant progress in recent years. Diverse object detection techniques have been proposed to produce robust results. As a result, the face detection technique achieved consistent results, with differences between classifiers for detecting faces in a few proportions. So, in this section, we will show some research related to our project:

The researchers at [20] Face detection is a difficult problem in ML and image processing areas. Three different approaches to facial recognition are presented in this project. This work aims to create new facial recognition technologies with fast running speeds and high recognition accuracy. Attempts are made to build facial recognition systems by mixing various algorithms. Comparisons and evaluations of recognition accuracy and running speed show that for certain

training data and eigenface sizes, PCA + Support Vector Machine(SVM) gets the best recognition performance, which is over 95%. Furthermore, PCA combined with the nearest neighbor (KNN) achieves a balance between recognition accuracy and running speed.

As well as in [21], researchers aim at designing a face recognition approach utilizing an ML algorithm and principal component analysis (PCA). It has been experimented with using linear discriminant analysis (LDA), Naive Bayes, support vector machines, and multilayer perceptrons. Moreover, it has achieved recognition accuracy of 97%&100% by using PCA and LDA.

As described in [22] the authors discussed two methods or algorithms that have been used in face detection: HAAR features and LBP. A comparative study has been conducted between these two algorithms by using three different databases as samples, which are the (COLOR FERET) database, the MIT CBCL database, and the Taarlab database. All three databases provide single-face images of various types of faces. Results have been evaluated based on detection speed for every image and detection hit rate, which leads to accuracy. Based on general results, it can be said that LBP has a 140% overall faster detection speed than HAAR. LBP also detected more faces than HAAR by an extra 4%. From here, we can say that LBP is better than HAAR in the aspect of detection accuracy and detection speed.

Also, in [23] researchers aim to present the comparison of two face detection techniques HAAR and the edification of the LBP for classification. The accuracy of face recognition may be improved by the clear image and right pose. As a result, while the execution time of HAAR is longer than that of LBP, its accuracy is better. This context could help people choose the best algorithm for their work.

2.4 Proposed work

The project reveals the important aspects of different methods concerning the accuracy, speed, and suitability for specific conditions of face detection. The study also provides which methods are most appropriate based on the purpose and where you use this classification.
This study helps generalize the concept of detecting other objects from still images in a succinct but trustworthy manner.

2.5 Summary

Hopefully, this chapter introduced a good summary of our project, starting with a little description of concepts used in our graduation project with the Face Detection Technique, and continuing with related work similar to our project. Finally, we described the proposed system and discussed issues related to it.

Chapter 3

METHODOLOGY

METHODOLOGY

3.1 Introduction

This chapter presents the research methodology. We look forward to using the most effective methodology commensurate with the capabilities of the Face Detection Technique, consisting of how to preprocess the data, how to train the face detector classifier, and to choose the final model, how to Viola-Jones object detection work, how to slide window technique.

3.2 Methodology procedure

We present the methodology for developing a system using Rapid Application Development (RAD). This methodology has become the appropriate choice because quick project turnaround is the main advantage of a RAD approach, which makes it a desirable option for developers working in a hectic environment like software development.
Due to RAD's focus on minimizing the planning stage and maximizing prototype development, this rapid pace is made possible.
We also provide an overview of the proposed method in this project, the system design procedure, and the project requirements.

This study is an experimental study that uses the RAD methodology. As in Figure 3.1

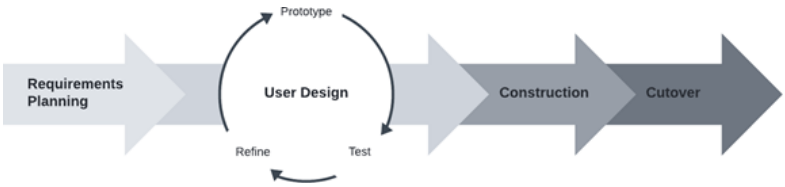

Figure 3.1: RAD methodology

The process can be broken down in a few ways, but in general, RAD follows four main phases:

1. Requirements Planning
This phase includes:

—Researching research to identify issues and gaps in other studies.

—Determine the project's aims and objectives.

—Finalizing the requirements with the acceptance of each team member and supervisor.

2. User Design
In this phase, start development and build up the system design through various prototype iterations, then test the system for any changes to ensure it achieves the project's expectations.

3.Construction
In this phase, the prototypes generated during the design phase are transformed into working models. The steps for this phase include coding, program development, and system testing.

4. Cutover
The last phase is implementation. The final system is released, which includes data conversion, testing, and the changeover to the new system.

3.3 Overview of The Proposed Method

Face detection systems have achieved more popularity due to the invention of real-time applications (fast). Still, much research is going on in face detection to make the algorithm more efficient and accurate. Face detection is not easy because it always includes faces, and it is hard to detect all the variances of a face that appears.

So, we are using OpenCV here. OpenCV is a free and open-source library of software for computer vision, and ML, which is mainly aimed at computer vision [24]. This library has more than 2500 algorithms, which include ML tools for classification and clustering, image processing, and vision algorithms.

OpenCV library has some built-in, pre-trained classifiers for detecting faces, eyes, smiles, etc. The proposed face detection system in this project, depicted in Figure 3.2 that used (HAAR Classifier), is of interest to us because it focuses on accuracy and performs in a concise but reliable manner [25].

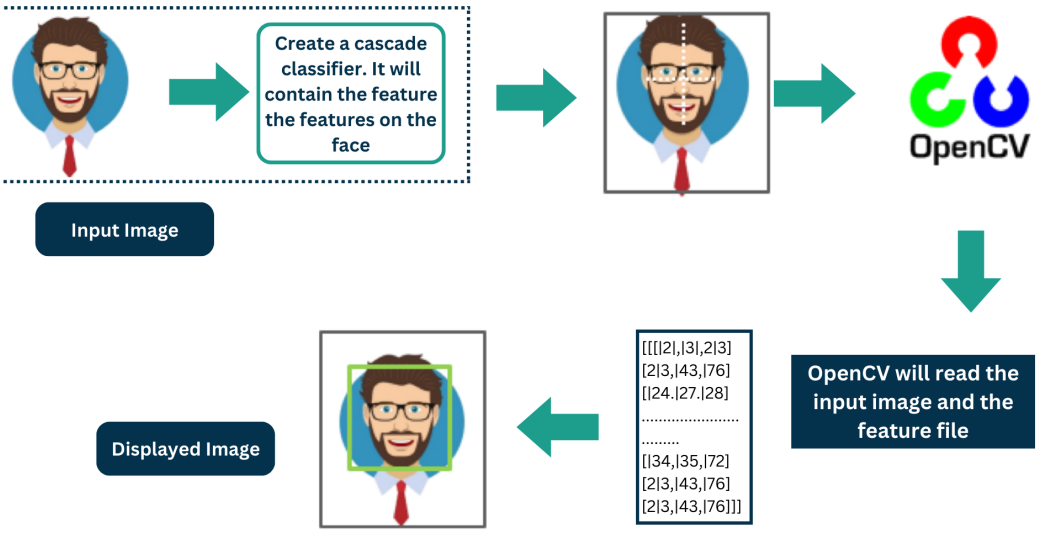

Figure 3.2: Diagram of Proposed face detection System

3.3.1 Face Detection Technique

Face detection is a technique that identifies or locates human faces in digital images. Face detection is commonly used when we take photos with our smartphones, and it rapidly detects faces in the picture. Face detection is not the same as face recognition. Face detection only detects the existence of faces in an image, while facial recognition identifies whose face it is [26]. As shown in Figure 3.3 [27].

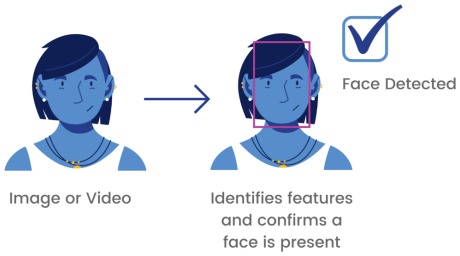

Figure 3.3: face detection

3.3.2 OpenCV classifiers

Face detection is performed by using classifiers. An algorithm known as a classifier decides whether a picture is positive (a face) or negative (not a face). A classifier must be trained on thousands of images that include and do not include faces. OpenCV already has two pre-trained face detection classifiers. Both of these classifiers process images in gray scales as they don't need color information to decide if an image has a face or not, which can readily be used in a program. The two classifiers are:

- **HAAR Cascades Classifier**
- **Local Binary Pattern LBP classifier.** [26]

Figure 3.4It contains a comparison between HAAR and LBP Classifiers based on execution time and Figure 3.5 contains a comparison between HAAR and LBP Classifiers based on accuracy. [23]

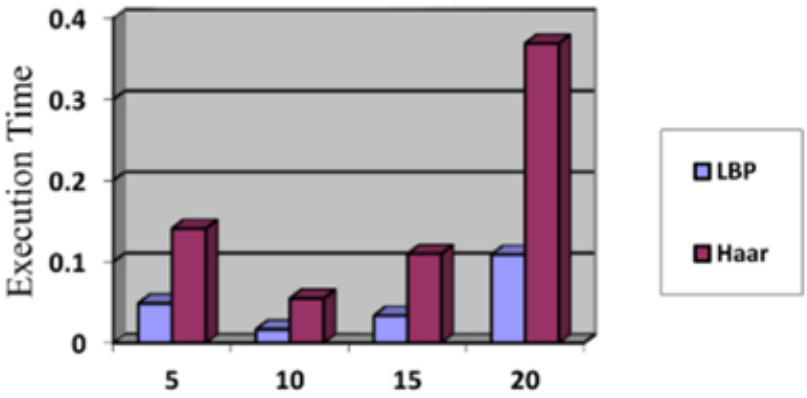

Figure 3.4: Comparison between HAAR and LBP Classifiers based on execution time

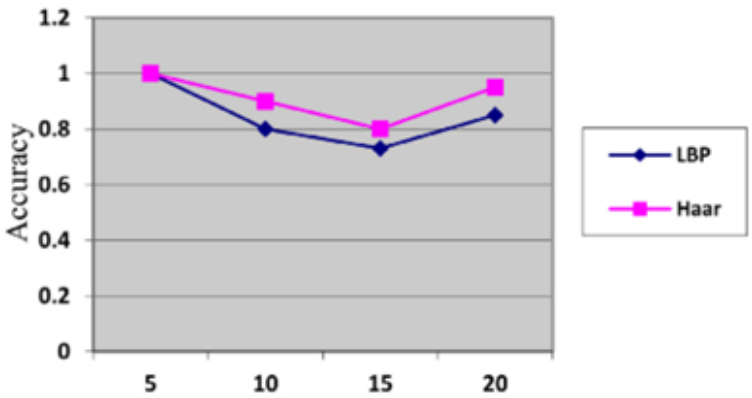

Figure 3.5: Comparison between HAAR and LBP Classifiers based on the accuracy

Each OpenCV face detection classifier has its pros and cons, but the major differences are in accuracy and speed. So, in case more accurate detections are required, the **HAAR classifier** is more suitable for technology such as security systems or high-end stalking. [28] But the **LBP classifier** is faster (a few times faster) but less accurate. (10-20% less than HAAR) [29], it should be used in mobile applications or embedded systems. [28] **In this project**, we propose using the HAAR Classifier because accuracy when face detection in this project is our concern.

3.3.3 HAAR Cascades Classifier

HAAR Cascading is the ML methodology wherever a classifier is trained from a great deal of positive and negative photos. The algorithm is forwarded by Paul Viola and Michael Jones. HAAR feature-based cascade classifiers are the classifiers executed for object detection. This classifier chases the ML procedure within which a cascade operation is inculcated from the images to find items in additional images. Face detection in a picture is with success detected. The exercise is finished by providing positive and negative photos to the classifier.
Then the characteristics are drawn out from the image. Every character is an individual value, which is acquired by subtracting the total of pixels in a white parallelogram from the summation of pixels in a black parallelogram. Which detects the faces of various individuals in several environments. The HAAR-like the feature of any size will be calculated in constant time thanks to integral pictures [23].

3.3.4 HAAR classifier feature

1. Loading the input image using the built-in function cv2.imread(img_path)**, here the passing the image path as an input parameter**
2. Converting it to gray-scale mode and then displaying
3. Loading the HAAR classifier

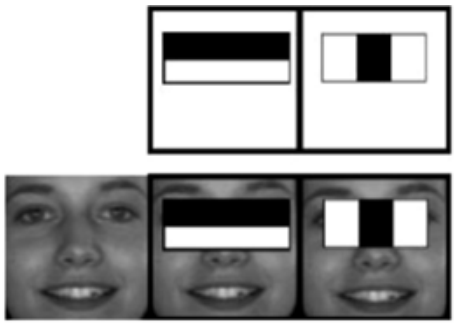

Figure 3.6: HAAR like feature

Figure 3.6 represents the HAAR like feature. It contains an edge feature and a line feature. In the gray-scale image, the white bar represents the pixels that are nearer to the light source.

3.3.5 HAAR calculation

- **HAAR value calculation :**

Pixel value = (Sum of Dark pixels / Number of Dark pixels)
−(Sum of Light pixels / Number of Light pixels)
The HAAR Classifier is an object detection algorithm. To discover the item and spot what it is, the features of the object will be extracted from the image. The HAAR pixel value is

calculated using this equation [23].

- **Accuracy calculation :**

-True positive (TP): It is an actual object of interest that is correctly identified.

The properly classified faces will be calculated as:
True positives rate (TPR) =TP/(TP+(FP))

-False-positives FP: It is a non-object of interest that is falsely identified as the true object.
Where,
TP: True Positive, FP: False Positive
(TN): True Negative, (FN): False Negative
By using this Equation, the Accuracy obtained for the HAAR is 96.24% and for the LBP classifier 94.74% [23]

3.4 HAAR detection if are any face there or not

Making a HAAR Classifier
The algorithm will be explained in four phases:

- Calculating HAAR Features
- Creating Integral Images
- Using Adaboost
- Implementing Cascading Classifiers

It's vital to recollect that this algorithm needs heaps of positive pictures of faces and negative pictures of non-faces to train the classifier, just like different ML models.

3.4.1 Calculating HAAR Features

The first phase is to collect the HAAR features. A **HAAR feature** is essentially calculates what is performed on adjacent rectangular regions at a particular location in a detection window. The calculation entails adding the constituent intensities in each region and adjusting the differences between the sums. Here are some examples of HAAR features below in Figure 3.7.

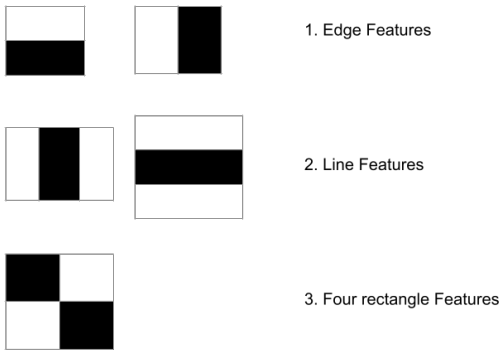

Figure 3.7: Types of HAAR features

These features will be troublesome to see for an outsized image. This is wherever integral images, are because the number of operations reducing using the integral image.

3.4.2 Creating Integral Images

Without going into too much about the mathematics, **integral images** essentially speed up the calculation of these HAAR features. rather than computing at each pixel, it instead creates sub-rectangles and creates array references for each of these sub-rectangles. These are then used to compute the HAAR features, as shown below in Figure 3.8.

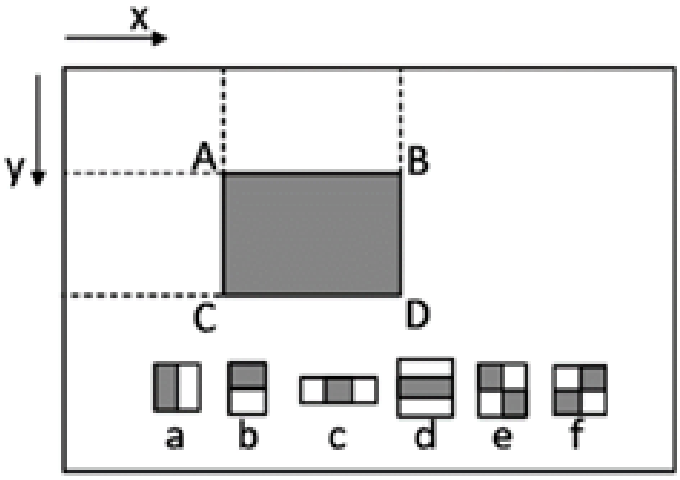

Figure 3.8: Illustration of how an integral image works

It's vital to notice that nearly all of the HAAR features are **irrelevant** when doing object detection because the only important features are those of the object. However, can we confirm the best features representing an object from the hundreds of thousands that exist? This is often the case wherever **Adaboost** comes into play.

3.4.3 Adaboost Training (improve classifier accuracy)

Adaboost primarily chooses the most effective options and trains the classifiers to use them. It uses a combination of "weak classifiers" to create a **"strong classifier"** algorithm that is used to discover objects.
Weak learners are created by moving the window over the input image and then computing HAAR features for each image subsection. This difference is compared to a learned threshold that separates non-objects from objects. Because these are **"weak classifiers"**, A large number of HAAR features are required for accuracy to form a strong classifier; Figure 3.9 depicts a boosting algorithm.

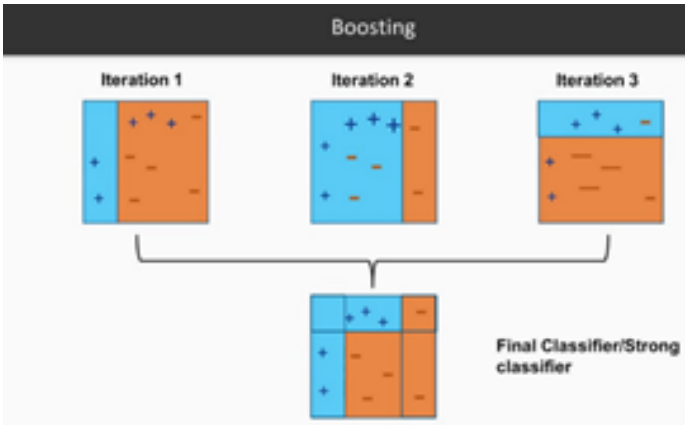

Figure 3.9: Illustration of a boosting algorithm

The last step combines these weak learners into strong learners using cascading classifiers.

3.4.4 Implementing Cascading Classifiers

The cascade classifier is created from a series of phases, where every phase may be an assortment of weak learners. Weak learners square measure trained exploitation boosting, which confers for a highly accurate classifier from the mean prediction of all the weak learners, the classifier Based on this prediction, the classifier either decides to indicate an object -face -was found (positive), or it is automatically rated as (negative). Phases to reject negative samples as fast as possible because most windows don't include anything of interest.
It's vital to **maximize a low false negative rate** because classifying an object as a non-object will so impact your object detection algorithm.

HAAR is one of many algorithms currently employed for object detection from images. One factor to notice concerning HAAR is that it's essential to reduce the false negative rate. Therefore, check that to tune hyperparameters accordingly when training your model [30].

3.5 System Design Procedure

In this section, we illustrate the flowchart of the Proposed system in Figure 3.10.

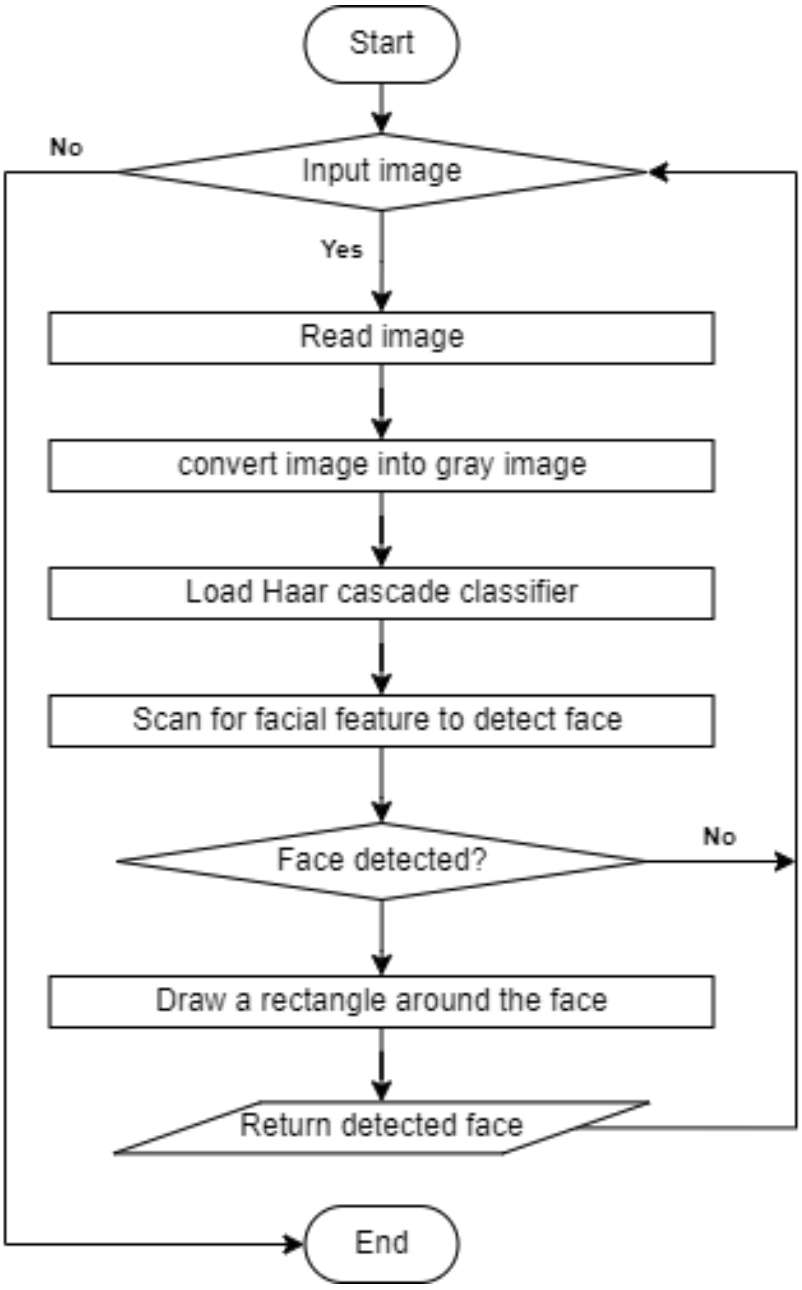

Figure 3.10: flowchart the of system.

In Figure 3.10, our system will be functioning correctly, as we planned. We will speak briefly about how the system will work. First, start the system. Then, using the OpenCV library, you must input an image to read the face image. Once the system acquires the image, it converts the image into grayscale. The LBP feature vectors are extracted, and a histogram of the face image is obtained using the LBPH algorithm, which represents the characteristic of the image. When the face image is converted into a grayscale image. Then, the system will load the HAAR cascade classifier for detection. After that, the system will do the scanning to detect the face. The HAAR classifier is used by this system to identify faces in images, as shown in Figure 2.1 importance of the HAAR Cascade classifier is that the perception precision is higher and the positive rate is lower. This function allows the detection of objects of different sizes in

the input image. As a result, an image with multiple people with different sizes of faces can also be detected. Finally, the system will return the image with the rectangle drawn around the face if the face is detected. If no detected faces are found, it will directly end the system.

3.6 Requirements

In this section, we describe the hardware and software requirements for this project.

3.6.1 Hardware Requirement

- Processor –Core i5 or more
- Hard Disk – 500 GB or more
- Memory – 4GB RAM or more
- Monitor

3.6.2 Software Requirements

- Windows 10 or higher
- python 2.7 or higher
- GitHub (to deploy and share this project)

3.7 Summary

The purpose of this chapter was to suggest an appropriate development methodology to develop a face detection system technology that is capable of not only detecting faces but also being able to face with a high- accuracy that meets our requirements. This chapter compares the two most essential algorithms HAAR Cascade and LBP Classifiers. The conclusion is that the HAAR cascade classifier is more accurate than the LBP classifier. This will help people to choose the best algorithm for their work.

Also, at the end of this chapter, there is a detailed explanation of how the system works through a diagram. Phase 2 of the project involves the implementation and testing of the system.

Chapter 4

IMPLEMENTATION AND TESTING

IMPLEMENTATION AND TESTING

4.1 Introduction

While finalizing the project ideas, we researched various software and algorithms used for face detection techniques to choose the best one for our project.
In this chapter, we will explain the implementation of our system. We will talk about how the system will run. We will provide descriptions of each system tool. Then we will test our system.

4.2 System Implementation

Some more software, like a visual studio and programming languages like C, Java, etc., is famous for coding face detection and recognition. After analyzing, we found that OpenCV provides the most extensive library for face detection and recognition, which makes this process easier. To implement the library gifted by OpenCV, we realized that the programming language Python is the best to use, as it implements the techniques more efficiently and is user-friendly. This project's implementation did not require hardware tools. However, we added a few more dependencies to make the process easier to use[31].

4.3 Implementation Steps

The implementation phase comes after the design phase and after understanding the complete system requirements and specifications. At this phase, we take the current code and assign the configuration to meet certain requirements and functions. The system is ready to be deployed, installed, and running in this phase.

4.4 Implementation Procedure

In this section, we will explain all software configuration requirements you need to have before running the code as "software configuration" and then implement the techniques as "user-friendly" coding. No hardware tools were required in this project's implementation.

4.4.1 Software Configuration

PyCHARM: An Integrated Development Environment (IDE) includes a code editor and a compiler for writing and compiling programs in one or more programming languages. An IDE contains many features that facilitate software development.
Python: It is being used as a programming language, and for the standard Application Programming Interface (API) OpenCV library is utilized.

This project's implementation has a few additional dependencies that we add to streamline the process for the face detection system. Table 4.1 shows the used libraries.

Table 4.1: Used Libraries

Library	Definition
CV2	Cv2 is the module import name for OpenCV, an open-source library used for computer vision. It is a Pre-built CPU-only OpenCV package for Python.
Numpy	Open source library used to perform mathematical operations on arrays.
OS	this module provides a function to interact with the operating system department functionality.

Before starting the coding part, we first installed the above software. Though PyCHARM was optional to install, we installed it to see the results in an organized manner. On the other hand, while implementing the face detection system, to make the computing easier, we used the NumPy array to feed data as input to the OpenCV function, as it contains a robust implementation of N-dimensional.

4.4.2 System Coding

Let us break down our project into easy steps and build an algorithm.

- **Importing OpenCV**
- **Importing an XML file**
- **Importing test Image**
- **Converting the image to greyscale**
- **Detecting Multi-scale faces**
- **Mentioning sides of the rectangle for face detection**
- **Displaying the detected image**

Since we have broken down the System Coding into multiple sections, let's start working on them!

STEP 1 – Importing OpenCV
In the command prompt type (pip install OpenCV-python), using this command, we can install and set up OpenCV to python. OpenCV provides an optimized Computer Vision library, tools, and hardware, and the same will be used in our project.
After creating a new directory and creating a new file named 'Main.py', below in Figure 4.1 showing Importing OpenCV.

```
1    #importing the OpenCV Lib
2    import cv2
```

Figure 4.1: Importing OpenCV

STEP 2 – Importing an XML file
Loading the HAAR Cascade front face file, You can download the file using this link and save it in the project folder– `https://raw.githubusercontent.com/saswatsamal/faceDetection/master/haarcascade_frontalface_default.xml` in Figure 4.2 showing Importing an XML file

```
4    #Loading the haarcascade front face file
5    faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Figure 4.2: Importing an XML file

STEP 3 – Importing the test image
imread() in Figure 4.3 is a method of OpenCV to read the input. Similarly, imshow() is a method to display the processed input in the form of output.

```
7    #Reading the face image
8    img = cv2.imread('a1.jpg')
```

Figure 4.3: Importing the test image

STEP 4 – Converting to grey Scale
This project works on images that are in HAAR and hence we convert the image to greyscale for ease of face detection as shown in Figure 4.4.

```
10   #Converting the image into grayscale image
11   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Figure 4.4: Converting to grey Scale

STEP 5 – Detecting Multi-scale faces
This function allows the detection of objects of different sizes in the input image. Hence, an image with multi people of different sizes of the face can also be detected. The parameters of the functions are detectMultiScale (InputArray image, double caleFactor=1.1, in minNeighbors=6), below in Figure 4.5 showing detectMultiScale.

```
13   #Detecting the face
14   face = faceCascade.detectMultiScale(gray, 1.1,4)
```

Figure 4.5: Detecting Multi-scale faces

STEP 6 – Mentioning sides of the rectangle for face detection
This function in Figure 4.6 helps us mention the rectangle's dimensions, thickness, and color that will be visible during face detection.

```
16   #Draw rectangle around the face
17   for(x,y,w,h) in face:
18       cv2.rectangle(img, (x,y),(x+w, y+h), (255,0,0),2)
```

Figure 4.6: Mentioning sides of the rectangle for face detection

STEP 7 – Displaying the detected image
Yay! You have come so far. Now it's time to display the image that has been detected. We also added the feature of closing the image tab only when a key is pressed. The detected image is depicted in Figure 4.7.

```
20    #Display the result with face detected
21    cv2.imshow('img',img)
22    cv2.waitKey(0)
23    cv2.destroyAllWindows()
```

Figure 4.7: Displaying the detected image

4.5 Testing

In this section, we did an initial test for the face detection technique. We used an image of a woman through research on Google, and then we downloaded the same file extension for this project. Then we installed the library in Table 4.1 needed to run testing code in the system, and we will show faces before AND after they are detected using the HAAR classifier in Figures 4.8 and 4.9.

Figure 4.8: before detected using the HAAR Cascade classifier

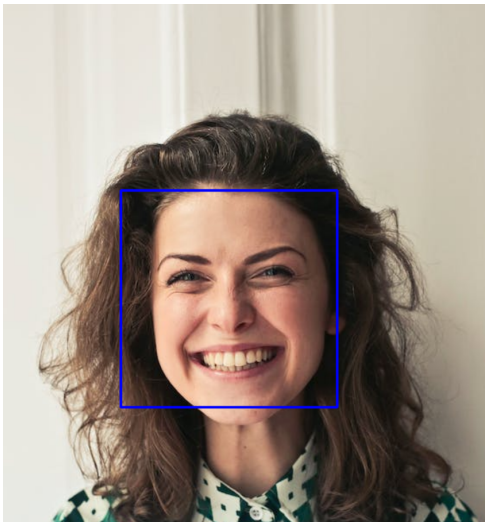

Figure 4.9: after detected using the HAAR Cascade classifier

The testing is performed on the HAAR cascade classifier by using some images. As a result, the HAAR cascade has more accuracy but the time taken by the HAAR cascade classifier is more compared with other classifiers in research. HAAR cascade classifier detects more faces than other classifiers in research. Table 4.2 contains the execution time, the number of faces detected, and the accuracy of the HAAR cascade classifier. Time is measured in seconds, and accuracy is measured in percentages.

Table 4.2: HAAR Cascade classifier performance

No.of faces in an image	**Execution Time (sec)**	**No.of faces detected**	**Accuracy (%)**
5	0.141	5	100
10	0.055	9	90
15	0.11	12	80
20	0.369	19	95

4.6 Summary

This chapter discussed implementation steps and listed the most important libraries used. Then we explained the stages of implementation of our proposed system. The proposed system was tested and presented the performance of the HAAR Cascade classifier.

Chapter 5

RESULTS AND DISCUSSIONS

RESULTS AND DISCUSSIONS

5.1 Introduction

In this chapter, the final results of the system will be discussed and reviewed, and the objectives we have achieved to meet the idea of the project will be determined.

5.2 Major Finding

We have successfully achieved most of what we had planned to do at the beginning of this project, which included searching for a suitable system for the project requirements, choosing the OpenCV library for detection, using a pre-trained HAAR cascade classifier to detect faces and eyes, and as a final step, verifying and implementing the developed system by analyzing existing studies and solutions.
Here in this figure 5.1 are the results of the images that we tested in our code:

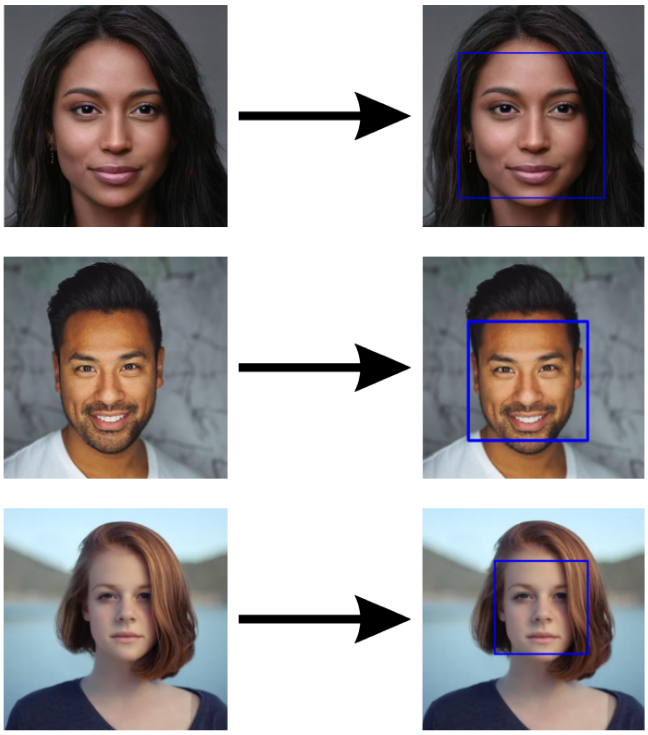

Figure 5.1: Tested faces

As mentioned in Chapter 3, Figure 3.10. Our system will be functioning correctly, as we planned. We will speak briefly about how the system will work.
Firstly, start the system. Then, you must input an image.5.2.

```
7    #Reading the face image
8    img = cv2.imread('testFace.jpg')
```

Figure 5.2: Reading the face image

Secondly, the system will convert the image into a gray-scale image. 5.3

```
10  #Converting the image into grayscale image
11  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Figure 5.3: Converting the image into a grayscale image

Then, the system will load the HAAR cascade classifier for detection. 5.4

```
4  #Loading the haarcascade front face file
5  faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Figure 5.4: Loading HAAR cascade classifier

After that, the system will do the scanning to detect the face.5.5

```
13  #Detecting the face
14  face = faceCascade.detectMultiScale(gray, 1.1,4)
```

Figure 5.5: Detecting the face

If the face is detected, the system will draw a rectangle around the face.5.6

```
16  #Draw rectangle around the face
17  for(x,y,w,h) in face:
18      cv2.rectangle(img, (x,y),(x+w, y+h), (255,0,0),2)
```

Figure 5.6: Draw a rectangle

Finally, the system will return the image with the rectangle drawn around the face.5.7

```
20  #Display the result with face detected
21  cv2.imshow('img',img)
22  cv2.waitKey(0)
23  cv2.destroyAllWindows()
```

Figure 5.7: Display the result

5.3 Discussions Related To Proposed Work

5.3.1 Discussion related to study objectives

In this section, we will review our goals and objectives for this study.

- **Display appropriate studies of face detection methods using OpenCV classifiers.**
  The goal has been achieved in Chapter 3, Section 3.3.2. We investigated various studies

on face detection methods using OpenCV classifiers. Then, we summarized the suitable method we found.

- **A suitable system is to be proposed, keeping in view the requirements that it should be accurate.**
  This goal has been achieved in Chapter 3, Section 3.3. We presented an overview of the work. Keeping in mind the requirements that it be accurate.
- **Detect the faces of people by using OpenCV.**
  The goal has been achieved in Chapter 2, Section 2.2.2. We clarified why we recommend this library.
- **Face detection using a pre-trained classifier for faces and eyes called (HAAR) Cascade.**
  The goal has been achieved in Chapter 3, Section 3.3.3.
- **Verify and implement the developed system by analyzing existing studies and solutions.**
  The goal has been achieved in Chapter 4. We clarified the implementation steps and procedures for putting the developed system into action.

5.3.1.1 Limitations of the proposed system

When implementing the proposed system, we noticed some limitations:

- **Slow performance in the face detection process and not very accurate performance.**
  Because the HAAR classifier is relatively slow in nature and predicts many false positives.
- **There is not much improvement it can do to the algorithm itself.**

5.4 Summary

In this chapter, we discussed, documented, and reviewed our study's goals, objectives, and limitations.

Chapter 6

CONCLUSIONS AND FUTURE WORK

CONCLUSIONS AND FUTURE WORK

6.1 Introduction

Face detection technology has come a long way in the last twenty years. Today, machines can automatically verify identity information for secure transactions, surveillance, and security tasks, access control to buildings, etc. In This chapter, the conclusion of the research will be discussed on the Face detection technique using OpenCV and HAAR cascade and closes with the future work of our project.

6.2 Conclusion

A fundamental concept of face detection is described in this report, followed by an implementation using OpenCV classifiers. With this classifier, the HAAR cascade was successfully used to detect faces in photos. Much research and testing have demonstrated that HAAR classifiers are the most effective for face detection. We have implemented and tested a prototype for automatic face detection. According to the test findings, our project's detection system can successfully identify human faces in photos. It will also be an affordable technology that can be used for a variety of devices, including automatic teller machines, vehicle security systems, and attendance tracking.

6.3 Future Work

The disadvantage of the classifiers is that they don't detect the faces of the children. This can be the future implementation. And along with face detection, face recognition may also be implemented.

References

[1] S. Manoharan *et al.*, "Image detection classification and recognition for leak detection in automobiles," *Journal of Innovative Image Processing (JIIP)*, vol. 1, no. 02, pp. 61–70, 2019.

[2] G. Singh and A. K. Goel, "Face detection and recognition system using digital image processing," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. IEEE, 2020, pp. 348–352.

[3] T. Dhawle, U. Ukey, and R. Choudante, "Face detection and recognition using opencv and python," *Int. Res. J. Eng. Technol*, vol. 7, no. 10, 2020.

[4] R. Rathi, M. Choudhary, and B. Chandra, "An application of face recognition system using image processing and neural networks," *International Journal of Computer Technology and Application*, vol. 3, no. 1, pp. 45–49, 2012.

[5] A. I. Khan and S. Al-Habsi, "Machine learning in computer vision," *Procedia Computer Science*, vol. 167, pp. 1444–1451, 2020.

[6] R. Bond, A. Koene, A. Dix, J. Boger, M. D. Mulvenna, M. Galushka, B. W. Bradley, F. Browne, H. Wang, and A. Wong, "Democratisation of usable machine learning in computer vision," *arXiv preprint arXiv:1902.06804*, 2019.

[7] J. Lemley, S. Abdul-Wahid, D. Banik, and R. Andonie, "Comparison of recent machine learning techniques for gender recognition from facial images." *MAICS*, vol. 10, pp. 97–102, 2016.

[8] H. Filali, J. Riffi, A. M. Mahraz, and H. Tairi, "Multiple face detection based on machine learning," pp. 1–8, 2018.

[9] R. T. Hasan and A. B. Sallow, "Face detection and recognition using opencv," *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, pp. 86–97, 2021.

[10] K. Sathursan, "Real Time AI Face Detection With Python for Beginners — medium.com," https://medium.com/swlh/real-time-ai-face-detection-with-python-for-beginners-a2a097636205, [Accessed 18-Dec-2022].

[11] J. Manikandan, S. L. Prathyusha, P. S. Kumar, Y. J. Chandra, and M. U. Hanuman, "Face detection and recognition using open cv based on fisher faces algorithm," *Proceedings of International Journal of Recent Technology and Engineering (IJRTE), ISSN*, pp. 2277–3878, 2020.

[12] E. Hjelmås and B. K. Low, "Face detection: A survey," *Computer vision and image understanding*, vol. 83, no. 3, pp. 236–274, 2001.

[13] R. Cipolla, S. Battiato, G. M. Farinella *et al.*, "Machine learning for computer vision," vol. 5, 2013.

[14] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705–741, 1995.

[15] Y. Lin and H. Xie, "Face gender recognition based on face recognition feature vectors," in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*. IEEE, 2020, pp. 162–166.

[16] A. Kumar, A. Kaur, and M. Kumar, "Face detection techniques: a review," *Artificial Intelligence Review*, vol. 52, no. 2, pp. 927–948, 2019.

[17] S. Emami and V. P. Suciu, "Facial recognition using opencv," *Journal of Mobile, Embedded and Distributed Systems*, vol. 4, no. 1, pp. 38–43, 2012.

[18] G. S. Nagpal, G. Singh, J. Singh, and N. Yadav, "Facial detection and recognition using opencv on raspberry pi zero," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. IEEE, 2018, pp. 945–950.

[19] "Face detection using Cascade Classifier using OpenCV-Python - GeeksforGeeks — geeksforgeeks.org," https://www.geeksforgeeks.org/face-detection-using-cascade-classifier-using-opencv-python/, [Accessed 10-Oct-2022].

[20] J. Chen and W. K. Jenkins, "Facial recognition with pca and machine learning methods," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 973–976.

[21] S. Sharma, M. Bhatt, and P. Sharma, "Face recognition system using machine learning algorithm," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2020, pp. 1162–1168.

[22] K. Kadir, M. K. Kamaruddin, H. Nasir, S. I. Safie, and Z. A. K. Bakti, "A comparative study between lbp and haar-like features for face detection using opencv," in *2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T)*. IEEE, 2014, pp. 335–339.

[23] A. B. Shetty, J. Rebeiro *et al.*, "Facial recognition using haar cascade and lbp classifiers," *Global Transitions Proceedings*, vol. 2, no. 2, pp. 330–335, 2021.

[24] "About - OpenCV — opencv.org," https://opencv.org/about/, [Accessed 29-Oct-2022].

[25] V. HARIRAMANI, "Face Identification using Haar cascade classifier — medium.com," https://medium.com/geeky-bawa/face-identification-using-haar-cascade-classifier-af3468a44814, [Accessed 30-Oct-2022].

[26] "Face detection with python using opencv tutorial | datacamp," https://www.datacamp.com/tutorial/face-detection-python-opencv, (Accessed on 10/30/2022).

[27] "Face recognition vs. face detection | what's the difference?" https://honorlock.com/blog/face-detection-vs-face-recognition-in-online-proctoring/, (Accessed on 11/04/2022).

[28] "Face detection using opencv and python: A beginner's guide - blogs - superdatascience | machine learning | ai | data science career | analytics | success," https://www.superdatascience.com/blogs/opencv-face-detection, (Accessed on 10/30/2022).

[29] "opencv - haar cascades vs. lbp cascades in face detection - stack overflow," https://stackoverflow.com/questions/8791178/haar-cascades-vs-lbp-cascades-in-face-detection, (Accessed on 10/30/2022).

[30] "Haar cascades, explained. a brief introduction into haar… | by aditya mittal | analytics vidhya | medium," https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d, (Accessed on 10/30/2022).

[31] M. Saberin Nipun, R. B. Sulaiman, and A. Kareem, "Efficiency comparison of ai classification algorithms for image detection and recognition in real-time," *arXiv e-prints*, pp. arXiv–2206, 2022.