**TABLE OF CONTENTS**

* **Acknowledgement I**
* **Certificate II**
* **List of Tables III**
* **List of Figures IV**
* **List of Abbreviations V**
* **Abstract VI**

**1. Introduction**

1.1 About Face Detection System

1.2 Aims & Objectives of the work

1.3 Background

**2. Literature Background**

2.1 What is Face Detection with all concept?

2.2 Machine Leaning

2.3 Python for machine learning

**3. Proposed Work**

3.1 Work Flow Diagram

3.2 Algorithm

3.3 Code

**4. Results & Discussion**

**5. Conclusion & Future Scope**

**6. References**

**CHAPTER-1**

**INTRODUCTION**

* 1. **About face Detection System:**

Building such a system from scratch using the Scala programming language helped achieve a better understanding of the field as well as its advantages and disadvantages compared to other biometric authentication methods. After some research, the decision to do face detection using an OpenCV library for Scala and face recognition using Microsoft’s Face API was unavoidable due to not having a system that could reliably do both detection and recognition in the project’s circumstances.

**1.2 Project Aims and Objectives:**

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

The goal of this work is to provide an easier human-machine interaction routine when user authentication is needed through face detection . With the aid of a regular web camera, a machine is able to detect a person’s face; a custom login screen with the ability to filter user access based on the users’ facial features will be developed. The objectives of this thesis are to provide a set of detection algorithms that can be later packaged in an easily portable framework amongst the different processor architectures we see in machines (computers) today. These algorithms must provide at least a 95% successful detection rate, out of which less than 3% of the detected faces are false positives.

Face Detection where a photo is searched to find a face, then the image is processed to crop and extract the person’s face for easier recognition.

Eigenfaces is considered the simplest method of accurate face recognition, but many other (much more complicated) methods or combinations of multiple methods are slightly more accurate. Most resources on face recognition are for basic Neural Networks, which usually don't work as well as Eigenfaces does. And unfortunately there are only some basic explanations for better type of face recognition than Eigenfaces, such as recognition from video and other techniques at the Face Recognition Homepage [4] or 3D Face Recognition Wikipedia page [5] and Active Appearance Models page [6]. But for other techniques, you should read some recent computer vision research papers from CVPR and other computer vision conferences. Most computer vision or machine vision conferences include new advances in face detection and face recognition that give slightly better accuracy. So for example you can look for the CVPR10 and CVPR09 conferences

Intel's open-source computer-vision library can greatly simplify computer vision programming. It includes advanced capabilities - face detection, face tracking, face recognition, Kalman filtering, and a variety of artificialintelligence (AI) methods - in ready-touse form. In addition, it provides many basic computer-vision algorithms via its lower-level APIs. OpenCV has the advantage of being a multi-platform framework; it supports both Windows and Linux, and more recently, Mac OS X. OpenCV has so many capabilities it can seem overwhelming at first. A good understanding of how these methods work is the key to getting good results when using OpenCV. Fortunately, only a select few need to be known beforehand to get started. OpenCV's functionality that will be used for facial recognition is contained within several modules.

A Haar wavelet is a mathematical fiction that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognise signals with sudden transformations. An example is shown in figure 1. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are introduced [10] for object detection as shown in figure 1. To analyse an image using Haar cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values pass a given threshold, it is considered a match. Face detection on a human face is performed by matching a combination of different Haar-like-features. For example, forehead, eyebrows and eyes contrast as well as the nose with eyes as shown below in figure A single classifier is not accurate enough.

1.3 Background of Project:

1.3.1 Face Detection :-

In computer vision, one essential problem we are trying to figure out is to automatically detect objects in an image without human intervention. Face detection can be thought of as such a problem where we detect human faces in an image. There may be slight differences in the faces of humans but overall, it is safe to say that there are certain features that are associated with all the human faces. There are various face detection algorithms but Viola-Jones Algorithm is one of the oldest methods that is also used today and we will use the same later in the article. You can go through the Viola-Jones Algorithm after completing this article as I’ll link it at the end of this article.

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. However, face detection can have very useful applications. The most successful application of face detection would probably be photo taking. When you take a photo of your friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly.

Now that we are successful in making such algorithms that can detect faces, can we also recognise whose faces are they?

Face recognition is a method of identifying or verifying the identity of an individual using their face. There are various algorithms that can do face recognition but their accuracy might vary. Here I am going to describe how we do face recognition using deep learning.

So now let us understand how we recognise faces using deep learning. We make use of face embedding in which each face is converted into a vector and this technique is called deep metric learning. Let me further divide this process into three simple steps for easy understanding:

1. **Face Detection.**
2. **Face Detection With Open-CV(Python)**
3. **Face Detection With Machine Learning**

**PROBLEMS ENCOUNTERED AND THEIR SOLUTIONS** - It only supports a few programming languages Even though OpenCV offers more than 3000 optimized algorithms, it only offers them for a small number of programming languages, namely C/C++, Python and Java for Android. However, wrapper libraries have been developed for other languages to encourage adoption by a wider audience. SOLUTION: An organization named Bytedeco, who constantly work on adapting C/C++ libraries to Java, have made an OpenCV library for Java and was later adapted to work with Scala. This library was used for having face detection work in real-time on a webcam feed. FACE RECOGNITION WITH MICROSOFT’S FACE API The recognition part of the project was made using Microsoft’s Face API. The reason is that Microsoft’s API offers the ability to create, delete, and update a face list, which represents a group of pictures that must have only one face in them, that can be used to compare a face from outside the list against all the faces in the list and find a match. Therefore, these face lists can be used to their full potential in such an environment

**API LIMITATIONS** However, there are a few restrictions when using Microsoft’s Face API: a free account can make 30.000 calls to the API per month and 20 per minute, whereas paid accounts can make 10 calls per second; only 64 face lists are allowed in one subscription; a face list cannot have more than 1000 faces; once a face has been added to a face list, the user receives the ID that was associated to that face, but there is no way to physically see what face/picture is represented by that ID anymore

**OTHER POTENTIAL SOLUTIONS**:

• A solution that avoids the use of folders would be to upload the pictures in an online database, but the downside is that storing a massive number of pictures in a database negatively impacts its performance and maintainability.

• A different solution is to upload the pictures on a photo-management website or on cloud storage. The main disadvantage of this approach would be uploading confidential information and/or pictures of the students on a website that is not guaranteed to be

**1.3.2** PYTHON :-

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. *Guido Van Rossum* is known as the founder of Python programming.Python is a simple, general purpose, high level, and object-oriented programming language. Python is an interpreted scripting language also.

Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

### Example

print("Hello, World!")

**Python** is a general purpose, dynamic, [high-level](https://www.javatpoint.com/classification-of-programming-languages), and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

## **Python 2 vs. Python 3**

In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version. However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses **print** as a statement and used as print "something" to print some string on the console. On the other hand, Python 3 uses **print** as a function and used as print("something") to print something on the console.
2. Python 2 uses the function raw\_input() to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the int() function in Python. On the other hand, Python 3 uses input() function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int(), str(), etc.).
3. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
4. Python 3 doesn't contain the xrange() function of Python 2. The xrange() is the variant of range() function which returns a xrange object that works similar to Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.
5. There is also a small change made in Exception handling in Python 3. It defines a keyword **as** which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

## **Python History**

Python was invented by **Guido van Rossum** in 1991 at CWI in Netherland. The idea of Python programming language has taken from the ABC programming language or we can say that ABC is a predecessor of Python language.

There is also a fact behind the choosing name Python. Guido van Rossum was a fan of the popular BBC comedy show of that time, **"Monty Python's Flying Circus"**. So he decided to pick the name **Python** for his newly created programming language.

Python has the vast community across the world and releases its version within the short period.

## **Where is Python used?**

Python is a general-purpose, popular programming language and it is used in almost every technical field. The various areas of Python use are given below.

* Data Science
* Date Mining
* Desktop Applications
* Console-based Applications
* Mobile Applications
* Software Development
* Artificial Intelligence
* Web Applications
* Enterprise Applications
* 3D CAD Applications
* Machine Learning
* Computer Vision or Image Processing Applications.
* Speech Recognitions

## **Python Basic Syntax**

There is no use of curly braces or semicolon in Python programming language. It is English-like language. But Python uses the indentation to define a block of code. Indentation is nothing but adding whitespace before the statement when it is needed. **For example -**

1. def func():
2. statement 1
3. statement 2
4. …………………
5. …………………
6. statement N

In the above example, the statements that are same level to right belong to the function. Generally, we can use four whitespaces to define indentation.

## **Python First Program**

Unlike the other programming languages, Python provides the facility to execute the code using few lines. **For example** - Suppose we want to print the **"Hello World"** program in Java; it will take three lines to print it.

1. **public** **class** HelloWorld {
2. **public** **static** **void** main(String[] args){
3. // Prints "Hello, World" to the terminal window.
4. System.out.println("Hello World");
5. }
6. }

On the other hand, we can do this using one statement in Python.

1. print("Hello World")

Both programs will print the same result, but it takes only one statement without using a semicolon or curly braces in Python

## **Python Popular Frameworks and Libraries**

Python has wide range of libraries and frameworks widely used in various fields such as machine learning, artificial intelligence, web applications, etc. We define some popular frameworks and libraries of Python as follows.

* **Web development (Server-side) -** Django Flask, Pyramid, CherryPy
* **GUIs based applications -** Tk, PyGTK, PyQt, PyJs, etc.
* **Machine Learning -** TensorFlow, PyTorch, **Scikit-learn**, Matplotlib, Scipy, etc.
* **Mathematics -** Numpy, Pandas, etc.

## **Python print() Function**

The **print()** function displays the given object to the standard output device (screen) or to the text stream file.

Unlike the other programming languages, Python **print()** function is most unique and versatile function.

The syntax of **print()** function is given below.

1. print(\*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

Let's explain its parameters one by one.

* **objects -** An object is nothing but a statement that to be printed. The \* sign represents that there can be multiple statements.
* **sep -** The **sep** parameter separates the print values. Default values is ' '.
* **end -** The **end** is printed at last in the statement.
* **file -** It must be an object with a write(string) method.
* **flush -** The stream or file is forcibly flushed if it is true. By default, its value is false.

Let's understand the following example.

### Example - 1: Return a value

1. print("Welcome to Face detection")
3. a = 10
4. # Two objects are passed in print() function
5. print("a =", a)
7. b = a
8. # Three objects are passed in print function
9. print('a =', a, '= b')

**Output:**

Welcome to Face detection.

a = 10

a = 10 = b

As we can see in the above output, the multiple objects can be printed in the single **print()** statement. We just need to use comma (,) to separate with each other.

## **Python Install**

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

## **The Python Command Line**

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py":

C:\Users\Your Name>py

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\*Your Name*>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\*Your Name*>python helloworld.py

The output should read:

Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

**1.3.3** MACHINE LEARNING :-

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for **building mathematical models and making predictions using historical data or information**. Currently, it is being used for various tasks such as **image recognition**, **speech recognition**, **email filtering**, **Facebook auto-tagging**, **recommender system**, and many more.

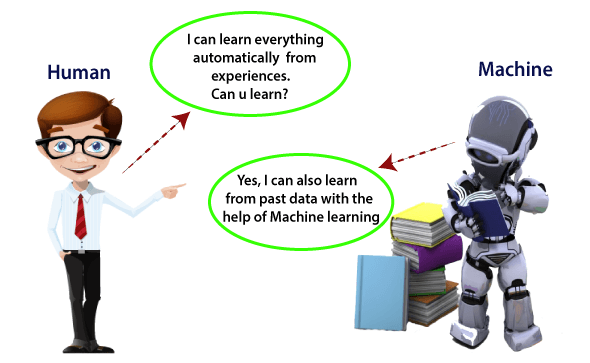
This machine learning tutorial gives you an introduction to machine learning along with the wide range of machine learning techniques such as **Supervised**, **Unsupervised**, and **Reinforcement** learning. You will learn about regression and classification models, clustering methods, hidden Markov models, and various sequential models.

## **What is Machine Learning**

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning**. Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**. We can define it in a summarized way as:

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.



**Fig: 1. Ability & Performance of ML to improve its Capacity to giving more data**

## **How does Machine Learning work**

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



**Fig: 2. Simple Mechanism of a ML**

## **Features of Machine Learning:**

* Machine learning uses data to detect various patterns in a given dataset.
* It can learn from past data and improve automatically.
* It is a data-driven technology.
* Machine learning is much similar to data mining as it also deals with the huge amount of the data.

## **Need for Machine Learning**

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.

We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases, Currently, machine learning is used in **self-driving cars**, **cyber fraud detection**, **face recognition**, and **friend suggestion by Facebook**, etc. Various top companies such as Netflix and Amazon have build machine learning models that are using a vast amount of data to analyze the user interest and recommend product accordingly.

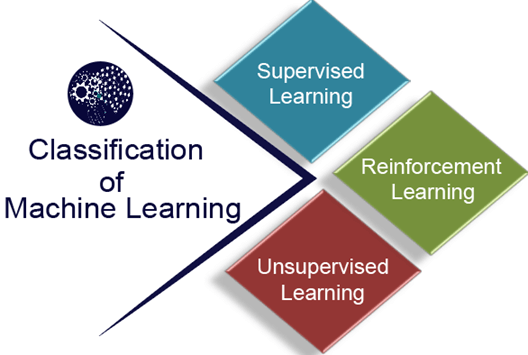
**Following are some key points which show the importance of Machine Learning:**

* Rapid increment in the production of data
* Solving complex problems, which are difficult for a human
* Decision making in various sector including finance
* Finding hidden patterns and extracting useful information from data.

## **Classification of Machine Learning**

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**



**Fig: 3. Classification of a ML**

### 1) Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is **spam filtering**.

Supervised learning can be grouped further in two categories of algorithms:

* **Classification**
* **Regression**

### 2) Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:

* **Clustering**
* **Association**

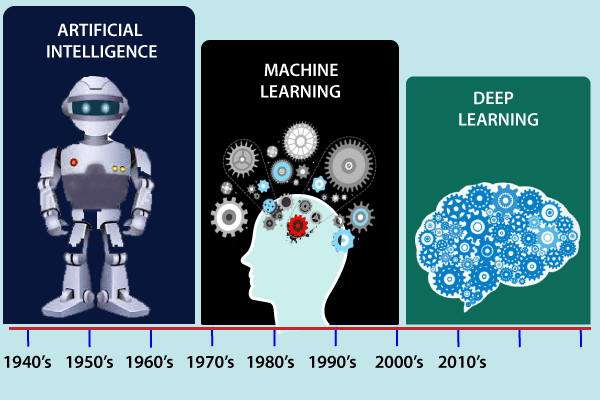
### 3) Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

## **History of Machine Learning**

Before some years (about 40-50 years), machine learning was science fiction, but today it is the part of our daily life. Machine learning is making our day to day life easy from **self-driving cars** to **Amazon virtual assistant "Alexa"**. However, the idea behind machine learning is so old and has a long history. Below some milestones are given which have occurred in the history of machine learning:



## **Fig: 4. History of a ML**

## **The early history of Machine Learning (Pre-1940):**

* **1834:** In 1834, Charles Babbage, the father of the computer, conceived a device that could be programmed with punch cards. However, the machine was never built, but all modern computers rely on its logical structure.
* **1936:** In 1936, Alan Turing gave a theory that how a machine can determine and execute a set of instructions.

## **The era of stored program computers:**

* **1940:** In 1940, the first manually operated computer, "ENIAC" was invented, which was the first electronic general-purpose computer. After that stored program computer such as EDSAC in 1949 and EDVAC in 1951 were invented.
* **1943:** In 1943, a human neural network was modeled with an electrical circuit. In 1950, the scientists started applying their idea to work and analyzed how human neurons might work.

## **Computer machinery and intelligence:**

* **1950:** In 1950, Alan Turing published a seminal paper, "**Computer Machinery and Intelligence**," on the topic of artificial intelligence. **In his paper, he asked, "Can machines think?"**

## **Machine intelligence in Games:**

* **1952:** Arthur Samuel, who was the pioneer of machine learning, created a program that helped an IBM computer to play a checkers game. It performed better more it played.
* **1959:** In 1959, the term "Machine Learning" was first coined by **Arthur Samuel**.

## **The first "AI" winter:**

* The duration of 1974 to 1980 was the tough time for AI and ML researchers, and this duration was called as **AI winter**.
* In this duration, failure of machine translation occurred, and people had reduced their interest from AI, which led to reduced funding by the government to the researches.

## **Machine Learning from theory to reality**

* **1959:** In 1959, the first neural network was applied to a real-world problem to remove echoes over phone lines using an adaptive filter.
* **1985:** In 1985, Terry Sejnowski and Charles Rosenberg invented a neural network **NETtalk**, which was able to teach itself how to correctly pronounce 20,000 words in one week.
* **1997:** The IBM's **Deep blue** intelligent computer won the chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

## **Machine Learning at 21st century**

* **2006:** In the year 2006, computer scientist Geoffrey Hinton has given a new name to neural net research as "**deep learning**," and nowadays, it has become one of the most trending technologies.
* **2012:** In 2012, Google created a deep neural network which learned to recognize the image of humans and cats in YouTube videos.
* **2014:** In 2014, the Chabot "**Eugen Goostman**" cleared the Turing Test. It was the first Chabot who convinced the 33% of human judges that it was not a machine.
* **2014:** **DeepFace** was a deep neural network created by Facebook, and they claimed that it could recognize a person with the same precision as a human can do.
* **2016:** **AlphaGo** beat the world's number second player **Lee sedol** at **Go game**. In 2017 it beat the number one player of this game **Ke Jie**.
* **2017:** In 2017, the Alphabet's Jigsaw team built an intelligent system that was able to learn the **online trolling**. It used to read millions of comments of different websites to learn to stop online trolling.

## **Machine Learning at present:**

Now machine learning has got a great advancement in its research, and it is present everywhere around us, such as **self-driving cars**, **Amazon Alexa**, **Catboats**, **recommender system**, and many more. It includes **Supervised**, **unsupervised**, and **reinforcement learning with clustering**, **classification**, **decision tree**, **SVM algorithms**, etc.

Modern machine learning models can be used for making various predictions, including **weather prediction**, **disease prediction**, **stock market analysis**, etc.

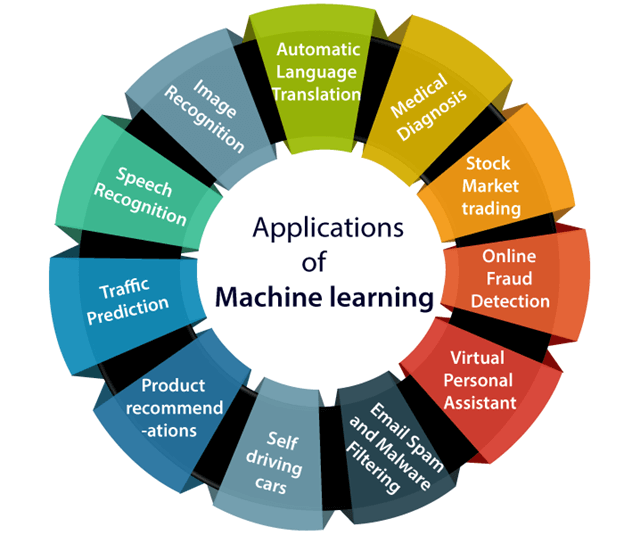
## **Prerequisites**

Before learning machine learning, you must have the basic knowledge of followings so that you can easily understand the concepts of machine learning:

* Fundamental knowledge of probability and linear algebra.
* The ability to code in any computer language, especially in Python language.
* Knowledge of Calculus, especially derivatives of single variable and multivariate functions.

# **Applications of Machine learning**

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:



### Fig: 5. Applications of a ML

### 1. Image Recognition:

Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, **Automatic friend tagging suggestion**:

Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's **face detection** and **recognition algorithm**.

**1.4 Existing Vs Proposed work :**

Feature analysis these algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions varies, and then use these to locate faces. These methods are designed mainly for face localization. Feature searching Viola and Jones presented an approach for object detection which minimizes computation time while achieving high detection accuracy. Viola and Jones proposed a fast and robust method for face detection which is 15 times quicker than existing techniques at the time of release with 95% accuracy. The technique relies on the use of simple Haar-like features that are evaluated quickly through the use of a new image representation. Based on the concept of an integral image it generates a large set of features and uses the boosting algorithm AdaBoost to reduce the over complete set (Zhang et al. 2011). The detector is applied in a scanning fashion and used on gray-scale images, the scanned window that is applied can also be scaled, as well as the features evaluated. This face detection framework is capable of processing images extremely rapidly while achieving high detection rates.

**Local binary pattern (LBP)** technique is very effective to describe the image texture features (Ahonen et al. 2004). LBP has advantages such as high-speed computation and rotation invariance, which facilitates the broad usage in the fields of image retrieval, texture examination, face recognition, image segmentation, etc. Recently, LBP was successfully applied to the detection of moving objects via background subtraction. In LBP, every pixel is assigned a texture value, which can be naturally combined with target for tracking thermo graphic and monochromatic video. Major uniform LBP patterns are used to recognize the key points in the target region and then form a mask for joint color-texture feature selection. **Advantages**: • Effective to describe image texture Feature. • Used in texture analysis, image retrievals, face recognition and image segmentation. • Detection of moving object via background subtraction. • Computationally simple than Haar like features and fast. • The most vital properties of LBP features are tolerance against the monotonic illumination changes and computational simplicity. **Limitations**: • Proposed method is not sensitive to small changes in the face localization. • Using larger local regions increases the errors. • It is insufficient for non-monotonic illumination changes. • Only used for binary and gray images.

**Gabor features based** method An Elastic Bunch Graph Map (EBGM) algorithm that successfully implements face detection system using Gabor filters has been purposed (Sharif et al. 2011). The proposed system applies 40 different Gabor filters on an image. As a result of which 40 images with different angles and orientation are achieved. After that, maximum intensity points in each filtered image are calculated and mark them as fiducial points. The system reduces these points in accordance to distance between them. The next step is calculating the distances between the reduced points using distance formula. At last, the distances are compared with database. If match occurs, it means that the faces in the image are detected.

**Image based approaches:**

Linear sub-space method Eigen faces method Eigenvectors has been used in face recognition, in which a simple neural network is demonstrated to perform face recognition for aligned and normalized face images. Images of faces can be linearly encoded using a modest number of basis images (Kirby and Sirovich 1990). They call the set of optimal basis vectors Eigen pictures since these are simply the eigenvectors of the covariance matrix computed from the vectorized face images in the training set (Hotelling 1933). Experiments on a set of 100 images show that a face image of 91×50 pixels can be effectively encoded using only 50 Eigen faces, while retaining a reasonable likeness (i.e., capturing 95 percent of the variance).Here I am going to add these things for my work , I think now a days it will help us .using same technique mobile phone Face-lock, Online exam ,CCTV etc. works are done .Also we can make some Application by this method to detect our Face.

**PROBLEM FACED**

While developing this project, I faced some minor problems. I had to do a lot of research on various Python modules such as, face recognition, os, shutil, cv2. The “cv2” module was required to capture the face using the webcam. The “os” and “shutil” modules were required for handling files and folders. The “face\_recognition” module was required for the main job, i.e., recognizing the faces. The quality of captured image matters a lot. Hence, a good-quality webcam is recommended for registering a face.

No matter what minor problems I faced. I enjoyed creating this piece of Python Project and I am more than happy with my creation.

**CHAPTER-2**

**LITERATURE BACKGROUND**

2.1 What is Face Detection with all concept?

Face detection is the process of automatically locating human faces in visual media (digital images or video). A face that is detected is reported at a position with an associated size and orientation. Once a face is detected, it can be searched for landmarks such as the eyes and nose.

The face API detects faces at a range of different angles, as illustrated below:

| **(a)** | **(b)** |
| --- | --- |
| https://developers.google.com/vision/images/coord.png | https://developers.google.com/vision/images/poseangles.png |

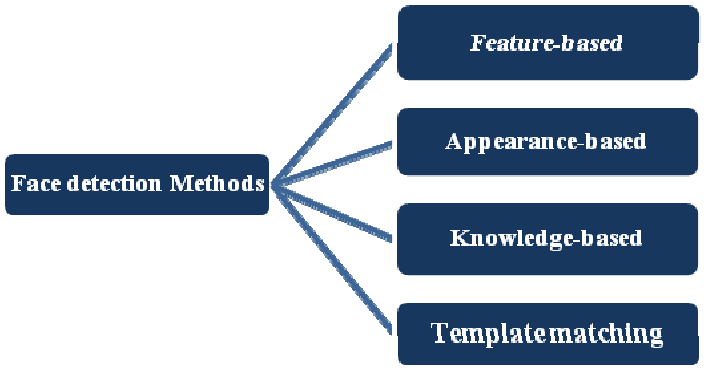
## **Fig: 6. Face Orientation**

Fig. 6. Pose angle estimation. (a) The coordinate system with the image in the XY plane and the Z axis coming out of the figure. (b) Pose angle examples where yEuler Y, rEuler Z.

The **Euler X**, **Euler Y**, and **Euler Z** angles characterize a face’s orientation as shown in Fig. 6. The Face API provides measurement of Euler Y and Euler Z (but not Euler X) for detected faces.

The Euler Z angle of the face is always reported. The Euler Y angle is available only when using the “accurate” mode setting of the face detector (as opposed to the “fast” mode setting, which takes some shortcuts to make detection faster). The Euler X angle is currently not supported

**Face Detection Methods**:-Yan, Kriegman, and Ahuja presented a classification for face detection methods. These methods divided into four categories, and the face detection algorithms could belong to two or more groups. These categories are as follows-



**Fig: 7.Different types of Face Detection Method**

**1. Knowledge-Based:-**

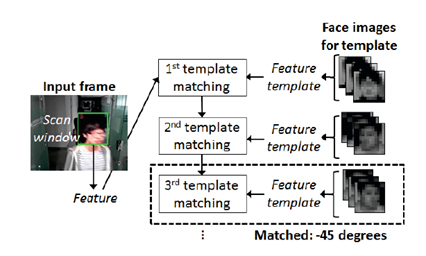
The knowledge-based method depends on the set of rules, and it is based on human knowledge to detect the faces. Ex- A face must have a nose, eyes, and mouth within certain distances and positions with each other. The big problem with these methods is the difficulty in building an appropriate set of rules. There could be many false positive if the rules were too general or too detailed. This approach alone is insufficient and unable to find many faces in multiple images.

**2. Feature-Based:-**

The feature-based method is to locate faces by extracting structural features of the face. It is first trained as a classifier and then used to differentiate between facial and non-facial regions. The idea is to overcome the limits of our instinctive knowledge of faces. This approach divided into several steps and even photos with many faces they report a success rate of 94%.

**4. Template Matching:-**

Template Matching method uses pre-defined or parameterised face templates to locate or detect the faces by the correlation between the templates and input images. Ex- a human face can be divided into eyes, face contour, nose, and mouth. Also, a face model can be built by edges just by using edge detection method. This approach is simple to implement, but it is inadequate for face detection. However, deformable templates have been proposed to deal with these problems.



**Fig: 8.Template Matching**

# **4. Appearance-Based:-**

The appearance-based method depends on a set of delegate training face images to find out face models. The appearance-based approach is better than other ways of performance. In general appearance-based method rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images. This method also used in feature extraction for face recognition.

**How the Face Detection Works:-**

There are many techniques to detect faces, with the help of these techniques, we can identify faces with higher accuracy. These techniques have an almost same procedure for Face Detection such as OpenCV, Neural Networks, Matlab, etc. The face detection work as to detect multiple faces in an image. Here we work on OpenCV for Face Detection, and there are some steps that how face detection operates, which are as follows-

Firstly the image is imported by providing the location of the image. Then the picture is transformed from RGB to Grayscale because it is easy to detect faces in the grayscale.

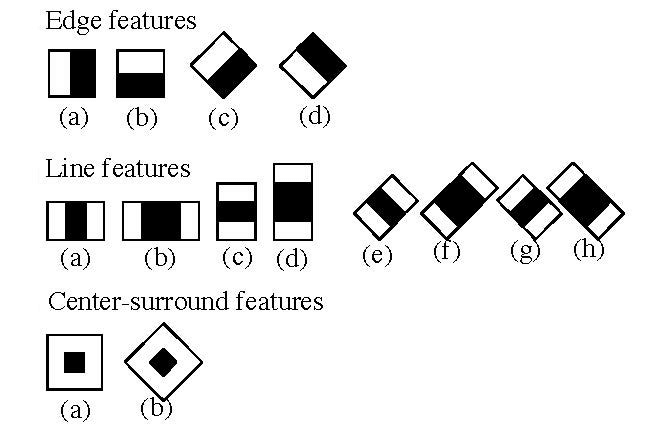


**Fig: 9.Converting RGB image to Grayscale**

After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images done if needed. The next step is image segmentation, which is used for contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture.

The next step is to use Haar-Like features algorithm, which is proposed by Voila and Jones for face detection. This algorithm used for finding the location of the human faces in a frame or image. All human faces shares some universal properties of the human face like the eyes region is darker than its neighbour pixels and nose region is brighter than eye region.

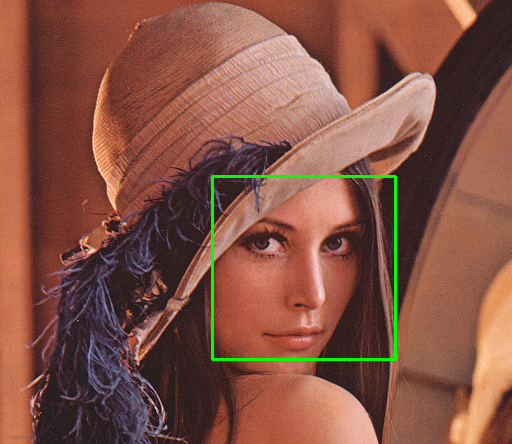




**Fig: 10. Haar-like features for face detection**

The haar-like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, centre detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.

The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.



**Fig: 11.Successfully detect the face in an image**

**How to Run Face Detector in Real-Time (Webcam):-**Requirement for Running the code- Python, OpenCV, Webcam, Numpy.

**#import libraries**import cv2  
import numpy as np **#import classifier for face and eye detection**  
face\_classifier = cv2.CascadeClassifier(‘Haarcascades/haarcascade\_frontalface\_default.xml’)**# Import Classifier for Face and Eye Detection**face\_classifier = cv2.CascadeClassifier(‘Haarcascades/haarcascade\_frontalface\_default.xml’)  
eye\_classifier = cv2.CascadeClassifier (‘Haarcascades/haarcascade\_eye.xml’)  
def face\_detector (img, size=0.5):**# Convert Image to Grayscale**gray = cv2.cvtColor (img, cv2.COLOR\_BGR2GRAY)  
faces = face\_classifier.detectMultiScale (gray, 1.3, 5)  
If faces is ():  
return img**# Given coordinates to detect face and eyes location from ROI**for (x, y, w, h) in faces  
x = x — 100  
w = w + 100  
y = y — 100  
h = h + 100  
cv2.rectangle (img, (x, y), (x+w, y+h), (255, 0, 0), 2)  
roi\_gray = gray[y: y+h, x: x+w]  
roi\_color = img[y: y+h, x: x+w]  
eyes = eye\_classifier.detectMultiScale (roi\_gray)  
for (ex, ey, ew, eh) in eyes:  
cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)  
roi\_color = cv2.flip (roi\_color, 1)  
return roi\_color**# Webcam setup for Face Detection**cap = cv2.VideoCapture (0)  
while True:  
ret, frame = cap.read ()  
cv2.imshow (‘Our Face Extractor’, face\_detector (frame))  
if cv2.waitKey (1) == 13: #13 is the Enter Key  
break**# When everything done, release the capture**cap.release ()  
cv2.destroyAllWindows ()

**2.2 Machine Leaning:**

**Arthur Samuel**, a pioneer in the field of artificial intelligence and computer gaming, coined the term **“Machine Learning”**. He defined machine learning as – **“Field of study that gives computers the capability to learn without being explicitly programmed”**. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

## **Features of Machine Learning:**

* Machine learning uses data to detect various patterns in a given dataset.
* It can learn from past data and improve automatically.
* It is a data-driven technology.
* Machine learning is much similar to data mining as it also deals with the huge amount of the data.

## **Need for Machine Learning**

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.

We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases, currently, machine learning is used in **self-driving cars**, **cyber fraud detection**, **face recognition**, and **friend suggestion by Facebook**, etc. Various top companies such as Netflix and Amazon have build machine learning models that are using a vast amount of data to analyse the user interest and recommend product accordingly.

**Following are some key points which show the importance of Machine Learning:**

* Rapid increment in the production of data
* Solving complex problems, which are difficult for a human
* Decision making in various sector including finance
* Finding hidden patterns and extracting useful information from data.

## **Classification of Machine Learning**

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

2.3 Python for machine learning:-

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. Python can be used on a server to create web applications.

# **Installing Anaconda and Python**

To learn machine learning, we will use the Python programming language in this tutorial. So, in order to use Python for machine learning, we need to install it in our computer system with compatible **IDEs (Integrated Development Environment)**.

In this topic, we will learn to install Python and an IDE with the help of **Anaconda distribution**.

Anaconda distribution is a free and open-source platform for Python/R programming languages. It can be easily installed on any OS such as Windows, Linux, and MAC OS. It provides more than 1500 Python/R data science packages which are suitable for developing machine learning and deep learning models.

Anaconda distribution provides installation of Python with various IDE's such as **Jupyter Notebook**, **Spyder**, **Anaconda prompt**, **etc**. Hence it is a very convenient packaged solution which you can easily download and install in your computer. It will automatically install Python and some basic IDEs and libraries with it.

# **PYTHON Machine learning Life cycle**

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps, which are given below:

* **Gathering Data**
* **Data preparation**
* **Data Wrangling**
* **Analyse Data**
* **Train the model**
* **Test the model**
* **Deployment**

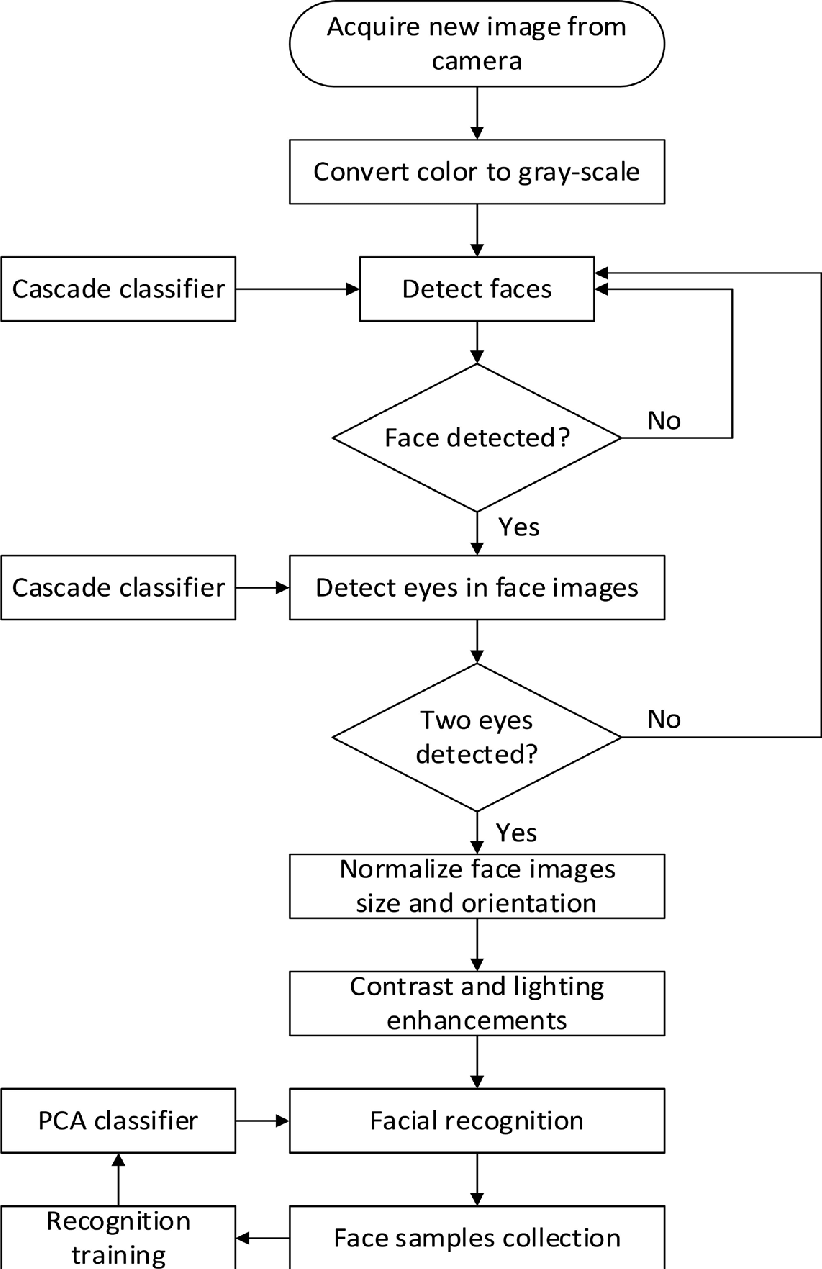
The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

**CHAPTER-3**

**PROPOSED WORK**

3.1 Work Flow Diagram :-



3.2 ALGORITHM :-

The primary aim of **face detection algorithms** is to determine whether there is any **face** in an image or not. ... It is widely used in cameras to identify multiple appearances in the frame Ex- Mobile cameras and DSLR's. Facebook is also using **face detection algorithm** to detect faces in the images and recognize them.

Numerous robust algorithms have been developed and claimed to have accurate performance to tackle face detection and recognition problems. These algorithms or methods are the most successfully and widely used for face detection and recognition.

**Eigenface based algorithm**: Eigenface based approach is the most widely used method for face detection. According to Pavanet al., eigenface is well known due to its simplicity, less sensitive in poses and better performance involving small databases or training sets [19]. This approach utilizes the presence of eyes, nose and mouth on a face and relative distances between these objects. This characteristic feature is known as Eigenfaces in facial domain [20]. This facial feature can be extracted by using a mathematical tool called Principle Component Analysis (PCA). By using PCA, any original image from the training set can be reconstructed by combining the Eigenfaces. Generally, a face is classified as a face by calculating the relative distance of the Eigenfaces.

**RGB (Red-Green-Blue)**: In RGB colour space, a normalized colour histogram is used to detect the pixels of skin colour of an image and can be further normalized for changes in intensity on dividing by luminance. This localizes and detects the face. However, this colour space is not preferable for colour based detection methods compared to YCbCr or HIS. A survey that has been conducted Vezhnevetset al. reveals that RGB colour space tends to mix the chrominance and the luminance data, high correlation between channels and significant perceptual non-uniformity [29]. These factors

**Skin colour based algorithm:** Skin colour is the most obvious and important features of human faces. Human skin colours are distinguished from different ethnic through the intensity of the skin colour not the chromatic features [22]. One of facial feature methods is involving skin colour based processing method. According to Crowley and Coutaz [23], one of the simplest algorithms for detecting skin pixels is to use skin colour algorithm. Each pixel is classified as skin colour and non-skin colour. This classification is based on its colourcomponent, which is modelled by Gaussian probability density [24]. For an input image, this method utilizes colour space for the skin region as the classification. Threshold is applied to mask the skin region. Finally, a bounding box is drawn to extract the face from the input image. According to Sanjay Kr. Singh et al., skin colour processing method offers a faster processing time than other facial feature methods and orientation invariant [13]. However, Yeong Nam Chaeet al. has a diverse opinion whereby skin colour method is time consuming as it scans the target image linearly which involves a large space of scanning [25]. Hence, they have proposed a novel method using sub-windows scanning instead of the conventional linear scanning. This proposed method works by scanning the image sparsely based the facial colour density by determining the horizontal and vertical intervals.

There are three most popular colour spaces, namely, the RGB [26][27], YCbCr [24] and HIS[28] . From the experiment, the results reveal that this proposed method was successfully detects faces in a shorter period of time compared to the conventional method. The sub-windows scanning method contributes to the less computational time as it skips the sub-windows that do not consist of possible faces

3.3 Code :-

Firstopen Pycharm , then open up a new file, name it  *dataset.py* now follow these steps:-

### ****Step #1: Install Libraries****

First, you should install the required libraries, OpenCV, and NumPy. You can install it easily through:

pip install opencv-python

pip install opencv-contrib-python

For installing NumPy in your system, use the same command as above and replace ‘opencv-python’ with ‘numpy’:

pip install numpy

### ****Step #2: Detect Faces****

Now, you must configure your camera and connect it to your system. The camera should work properly to avoid any issues in face detection.

Before our camera recognizes us, it first has to detect faces. We’ll use the Haar Cascade classifier for face detection. It is primarily an object detection method where you train a cascade function through negative and positive images, after which it becomes able to detect objects in other photos.

In our case, we want our model to detect faces. OpenCV comes with a trainer and a detector, so using the Haar Cascade classifier is relatively more comfortable with this library. You can create your classifier to detect other images as well.

**Here’s the code:**

import cv2  
import numpy as np  
  
face\_classifier = cv2.CascadeClassifier('C:/Users/girijasankar/Desktop/haarcascade\_frontalface\_default.xml')  
  
def face\_extractor(img):  
  
 gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  
 faces = face\_classifier.detectMultiScale(gray,1.3,5)  
  
 if faces is():  
 return None  
  
 for(x,y,w,h) in faces:  
 cropped\_face = img[y:y+h, x:x+w]  
  
 return cropped\_face  
  
  
cap = cv2.VideoCapture(0)  
count = 0  
  
while True:  
 ret, frame = cap.read()  
 if face\_extractor(frame) is not None:  
 count+=1  
 face = cv2.resize(face\_extractor(frame),(200,200))  
 face = cv2.cvtColor(face, cv2.COLOR\_BGR2GRAY)

file\_name\_path = 'C:/Users/girijasankar/Desktop/dataset/'+str(count)+'.jpg'  
  
 cv2.imwrite(file\_name\_path,face)  
  
 cv2.putText(face,str(count),(50,50),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,255,0),2)  
 cv2.imshow('Face Cropper',face)  
 else:  
 print("Face not found")  
 pass  
  
 if cv2.waitKey(1)==13 or count==100:  
 break  
  
cap.release()  
cv2.destroyAllWindows()  
print('Samples Colletion Completed ')

### ****Step #3: Gather Data****

Now that your model can identify faces, you can train it so it would start recognizing whose face is in the picture. To do that, you must provide it with multiple photos of the faces you want it to remember.

That’s why we’ll start with creating our dataset by gathering photos. After collecting the necessary images, add IDs for every person, so the model knows what face to associate with what ID. Start with the images of one person and add at least 10-20. Use different expressions to get the most effective results.

Create a script for adding user IDs to images, so you don’t have to do it manually every time. The script is vital in case you want to use your model for multiple faces.

### ****Step #4: Train****

After creating the dataset of the person’s images, you’d have to train the model. You’d feed the pictures to your OpenCV recognizer, and it will create a file named ‘training.py’ in the end.

In this stage, you only have to provide the model with images and their IDs so the model can get familiar with the ID of every image. After we finish training the model, we can test it.

**Here’s the code:**

import cv2  
import numpy as np  
from os import listdir  
from os.path import isfile, join  
  
data\_path = 'C:/Users/girijasankar/Desktop/dataset/'  
onlyfiles = [f for f in listdir(data\_path) if isfile(join(data\_path,f))]  
  
Training\_Data, Labels = [], []  
  
for i, files in enumerate(onlyfiles):  
 image\_path = data\_path + onlyfiles[i]  
 images = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  
 Training\_Data.append(np.asarray(images, dtype=np.uint8))  
 Labels.append(i)  
  
Labels = np.asarray(Labels, dtype=np.int32)  
  
model = cv2.face.LBPHFaceRecognizer\_create()  
  
model.train(np.asarray(Training\_Data), np.asarray(Labels))  
  
print("Dataset Model Training Completed ")

### ****Step#5: Start Recognition****

Now that you have trained the model, we can start testing the model. In this section, we have added names to the IDs so the model can display the names of the respective users it recognizes.

The model doesn’t recognize a person. It predicts whether the face it detects matches to the face present in its database. Our model displays a percentage of how much the face matches the face present in its database. Its accuracy will depend heavily on the image you’re testing and the pictures you’ve added to your database (the images you trained the model with).create a file called *detection.py* and follow the code-

**Here’s the code:**

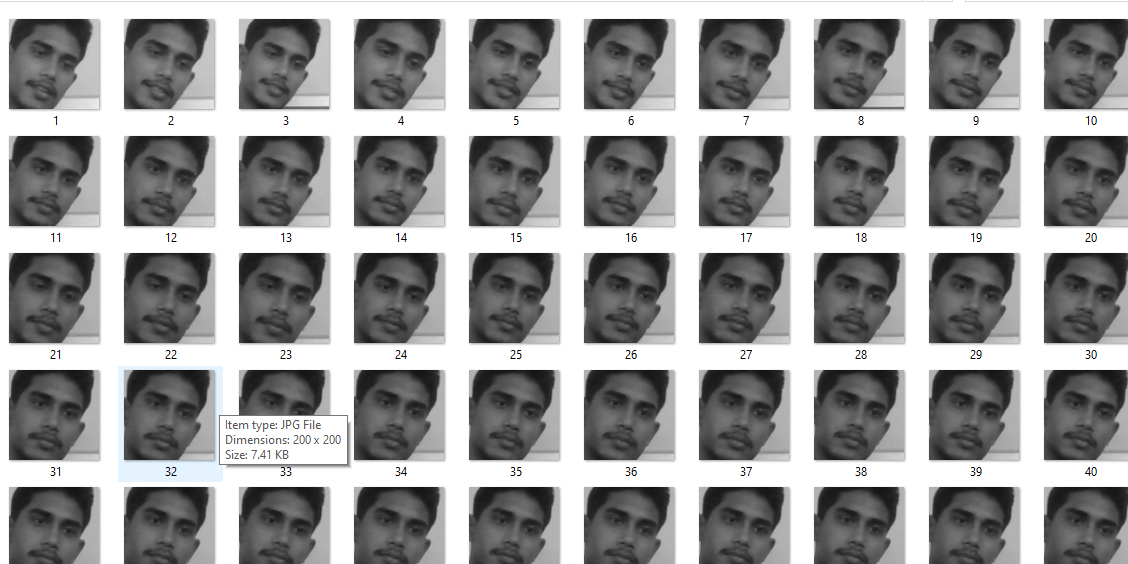
import cv2  
import numpy as np  
from os import listdir  
from os.path import isfile, join  
  
data\_path = 'C:/Users/girijasankar/Desktop/dataset/'  
onlyfiles = [f for f in listdir(data\_path) if isfile(join(data\_path,f))]  
  
Training\_Data, Labels = [], []  
  
for i, files in enumerate(onlyfiles):  
 image\_path = data\_path + onlyfiles[i]  
 images = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  
 Training\_Data.append(np.asarray(images, dtype=np.uint8))  
 Labels.append(i)  
  
Labels = np.asarray(Labels, dtype=np.int32)  
  
model = cv2.face.LBPHFaceRecognizer\_create()  
  
model.train(np.asarray(Training\_Data), np.asarray(Labels))  
  
print("Dataset Model Training Completed ")  
face\_classifier = cv2.CascadeClassifier('C:/Users/girijasankar/Desktop/haarcascade\_frontalface\_default.xml')  
  
def face\_detector(img, size = 0.5):  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces = face\_classifier.detectMultiScale(gray,1.3,5)  
  
 if faces is():  
 return img,[]  
  
 for(x,y,w,h) in faces:  
 cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,0),2)  
 roi = img[y:y+h, x:x+w]  
 roi = cv2.resize(roi, (200,200))  
  
 return img,roi  
  
cap = cv2.VideoCapture(0)  
while True:  
  
 ret, frame = cap.read()  
  
 image, face = face\_detector(frame)  
  
 try:  
 face = cv2.cvtColor(face, cv2.COLOR\_BGR2GRAY)  
 result = model.predict(face)  
  
 if result[1] < 500:  
 confidence = int(100\*(1-(result[1])/300))  
  
  
  
 if confidence > 82:  
 cv2.putText(image, "Himangshu", (250, 450), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), 2)  
 cv2.imshow('Face Cropper', image)  
  
 else:  
 cv2.putText(image, "Unknown", (250, 450), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0, 0, 255), 2)  
 cv2.imshow('Face Cropper', image)  
  
  
 except:  
 cv2.putText(image, "Face Not Found", (250, 450), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 0, 0), 2)  
 cv2.imshow('Face Cropper', image)  
 pass  
  
 if cv2.waitKey(1)==13:  
 break  
  
  
cap.release()  
cv2.destroyAllWindows()

**CHAPTER-4**

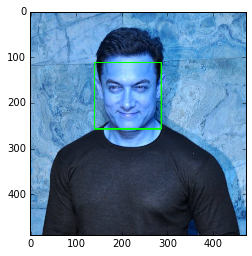
**RESULTS & DISCUSSION**

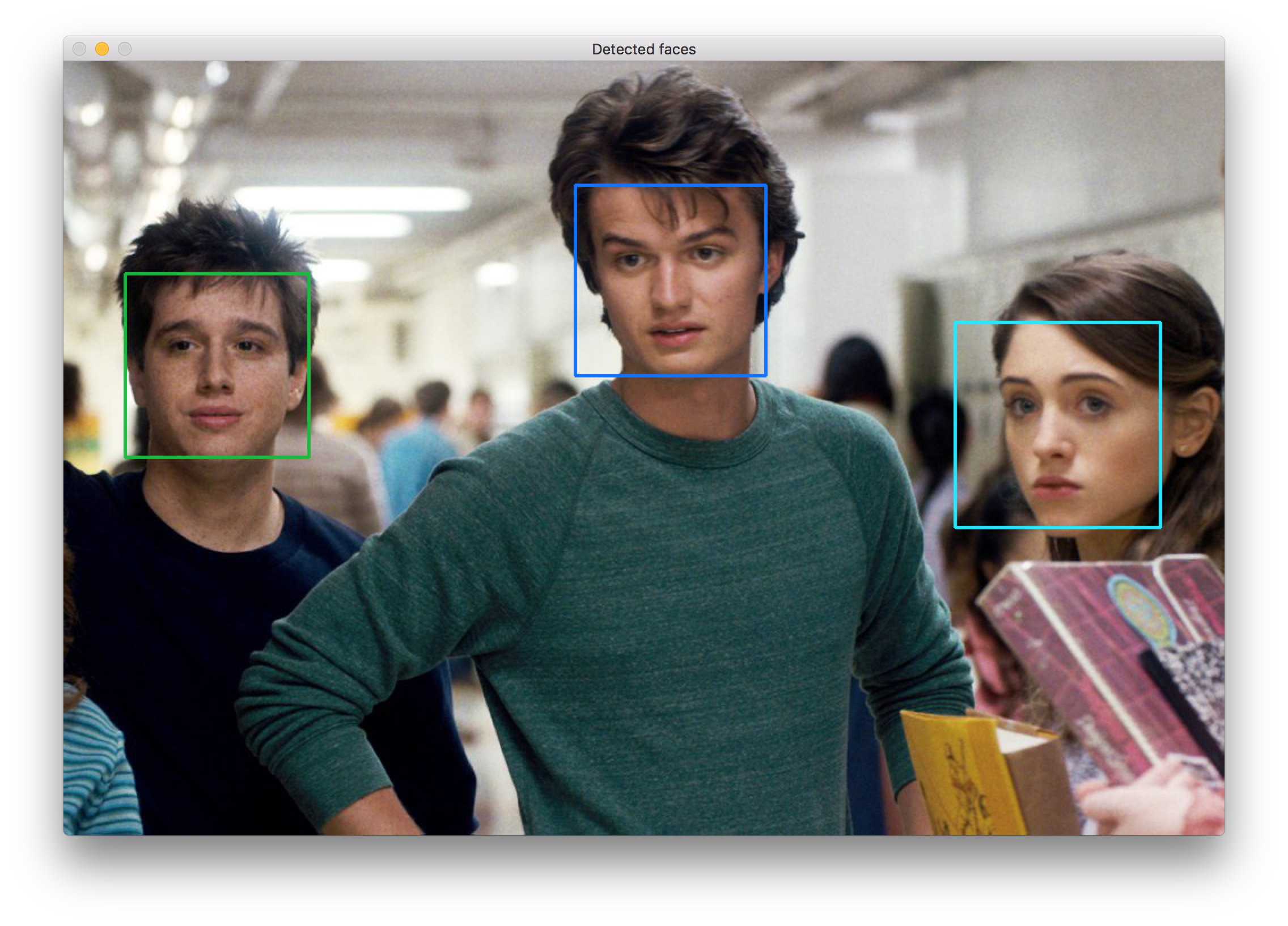
First, we need to install OpenCV package for python. This package can be downloaded from python website or by using pip install command. Then we use two cascade files haarcascade\_frontalface\_default.xml and haarcascade\_eye.xml files which are available under GNU licence and can be used without permission also. The whole code is implemented in python and need a working webcam to capture images or videos.The model is implemented successfully and is able to recognise faces in still images, videos, paintings and webcam captures. For still images and paintings the model is able to recognize the face. Result for different images is shown below.

Taking some Gray scale image at 30 degree angel motion picture within a minute and test it

****

**Fig: 12. View of Face Gray-scale image**

**Fig: 13 View of Face detection Result**

****

**Fig: 14. Multiple Face detection Result**

**Discussion :**

Early attempts on Eigenface and Fisherface was disappointing since the LBPH alone recognised a face. By designing an application to test the algorithms, and after calibration with new data, both algorithms performed well. The tester applications also allowed accurate threshold settings. Another problem was people tilting their head when images taken for the data. This was fixed with an application that identifies the locations of the eyes and rotate the image to correct the off-set. It was noticed that some early-stage images in the data-set different brightness’s. To resolve this, before taking an image, the brightness was averaged to prevent dark images. These changes to the system improved the performance noticeably.

This paper concentrates more on the theoretical part as review of the overall of the highlighted algorithms. Based on these features, the strengths and limitations can be analyzed. The reviewed parameters are explained in the following subsection. Please take note that these parameters are not fixed as every case yields different outcomes. Hence, this review is based on previous researches done by other people.

**i. Size and types of database (dataset)**

**ii. Facial expressions variations**

**iii. Illumination tolerance**

**iv. Pose variations**

Most face detection systems attempt to extract a fraction of the whole face, thereby eliminating most of the background and other areas of an individual's head such as hair that are not necessary for the face recognition task. With static images, this is often done by running a across the image. The face detection system then judges if a face is present inside the window (Brunelli and Poggio, 1993). Unfortunately, with static images there is a very large search space of possible locations of a face in an image.

Real-time face detection involves detection of a face from a series of frames from a video capturing device. While the hardware requirements for such a system are far more stringent, from a computer vision stand point, real-time face detection is actually a far simpler process than detecting a face in a static image. This is because unlike most of our surrounding environment, people are continually moving. We walk around, blink, fidget, wave our hands about, etc.

It is process of identifying different parts of human faces like eyes, nose, mouth, etc… this process can be achieved by using MATLAB codeIn this project the author will attempt to detect faces in still images by using image invariants. To do this it would be useful to study the grey scale intensity distribution of an average human face. The following 'average human face' was constructed from a sample of 30 frontal view human faces, of which 12 were from females and 18 from males. A suitably scaled colormap has been used to highlight grey-scale intensity differences. The grey-scale differences, which are invariant across all the sample faces are strikingly apparent. The eye-eyebrow area seem to always contain dark intensity (low) gray-levels while nose forehead and cheeks contain bright intensity (high) grey-levels. After a great deal of experimentation, the researcher found that the following areas of the human face were suitable for a face detection system based on image invariants and a deformable template. The above facial area performs well as a basis for a face template, probably because of the clear divisions of the bright intensity invariant area by the dark intensity invariant regions. Once this pixel area is located by the face detection system, any particular area required can be segmented based on the proportions of the average human face After studying the above images it was subjectively decided by the author to use the following as a basis for dark intensity sensitive and bright intensity sensitive templates. Once these are located in a subject's face, a pixel area 33.3% (of the width of the square window) below this.

The problem of face recognition is all about face detection. This is a fact that seems quite bizarre to new researchers in this area. However, before face recognition is possible, one must be able to reliably find a face and its landmarks. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step. There are two types of face detection problems:

1) Face detection in images and

2) Real-time face detection.

**CHAPTER-5**

**CONCLUSION & FUTURE WORK**

In recent years face detection has achieved considerable attention from researchers in bio metrics, pattern recognition, and computer vision groups. There is countless security, and forensic applications requiring the use of face recognition technologies. As you can see, face detection system is very important in our day to day life. Among the entire sorts of biometric, face detection and recognition system is the most accurate. In this article, we have presented a survey of face detection techniques. It is exciting to see face detection techniques be increasingly used in real-world applications and products. Applications and challenges of face detection also discussed which motivated us to do research in face detection. The most straightforward future direction is to further improve the face detection in presence of some problems like face occlusion and non-uniform illumination. Current research focuses in field of face detection and recognition is the detection of faces in presence of occlusion and non-uniform illumination. A lot of work has been done in face detection, but not in presence of problem of presence of occlusion and non-uniform illumination. If it happens, it will help a lot to face recognition, face expression recognition etc. Currently many companies providing facial biometric in mobile phone for purpose of access. In future it will be used for payments security, healthcare, advertising, criminal identification etc.

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable. The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of Eigen faces that were used for the PCA transform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate. The fully automated frontal view face detection system displayed virtually perfect accuracy and in the

researcher's opinion further work need not be conducted in this area.

That's why face recognition is relatively easy to do in real-time if you are training on someone and then instantly trying to recognize them after, since it will be the same camera, and background will be the same, their expressions will be almost the same, the lighting will be the same, and the direction you are viewing them from will be the same. So you will often get good recognition results at that moment. But once you try to recognize them from a different direction or from a different room or outside or on a different time of the day, it will often give bad results! So it’s important to do a lot of experimentation if you want better results, and if you still can't get good results after trying many things, perhaps you will need a more complicated face recognition algorithm than PCA (Eigenfaces), such as 3D Face Recognition or Active Appearance Models, mentioned below. The proposed model is able to recognize faces correctly but when tried for videos, it takes more time for processing. The advantage of this model is that it is able to recognize blurred images and side face images also which other traditional models are incapable of recognizing in such case. The only drawback is that it fails to recognize eyes with glasses. In future, this can be extended to recognize persons using video capture which will be helpful in getting identities from CCTV cameras that can police to identify the person in no time. It can also be implemented in home security systems as well.

**CHAPTER-6**

**REFERENCES**

[1] Seeing with OpenCV, Article,

[http://www.cognotics.com/opencv/servo \_2007\_series/part\_1/index.html](http://www.cognotics.com/opencv/servo%20_2007_series/part_1/index.html),

Published by Robin Hewitt, 2010

[2] OpenCV Homepage, <http://opencv.willowgarage.com>

[3] Face Recognition Homepage, <http://www.face-rec.org/algorithms/>

[4] Wikipedia, Three-dimensional face recognition, <http://en.wikipedia.org/wiki/Three->dimensional\_face\_recognition

[5] Wikipedia, Active appearance model,

http://en.wikipedia.org/wiki/Active\_appe arance\_model

[6] Computer Vision Papers,

http://www.cvpapers.com/

[7] Wikipedia, Histogram equalization,

http://en.wikipedia.org/wiki/Histogram\_ equalization

[8] Shervin Emami, Rotating or Resizing an Image in OpenCV, http://shervinemami.info/imageTransforms.html

[9] PISSARENKO, D. (2003). EigenfaceBased Facial Recognition.Retrieved from http://http://openbio.sourceforge.net/resourc es/eigenfaces/eigenfaces html/facesOptions.html.

[10] McCOLLUM, P. (1998). An Introduction to Back-Propagation Neural Networks.Retrieved from http://www.seattlerobotics.org/encoder/nov 98/neural.html.

[11] Retrieved on 16 April 2009 from

http://www.xidtech.com/our\_news\_2008.html

[12] <https://github.com/ageitgey>

[13] https://github.com/akshaybahadur21

[14] https://github.com/cmusatyalab

[15] https://github.com/akshaykhadse

[16] https://github.com/didi

[17] https://github.com/krasserm

[18] For python & ML

https://www.javatpoint.com/

[19] For some web design work and Front-end work-

https://www.w3schools.com/

[20] https://realpython.com/face-recognition-with-python/

[21] https://www.upgrad.com/blog/face-detection-project-in-python/

[22] Haralick, R.M. and Shapiro, L.G.. (1992) Computer and Robot Vision, Volume I. Addison-Wesley

[23] Haxby, J.V., Ungerleider, L.G., Horwitz, B., Maisog, J.M., Rapoport, S.I., and Grady, C.L. (1996). Face encoding and recognition in the human brain. Proc. Nat. Acad. Sci. 93: 922 - 927.

[24] Heisele, B. and Poggio, T. (1999) Face Detection. Artificial Intelligence Laboratory. MIT.

[25] Jang., J., Sun, C., and Mizutani, E. (1997) Neuro-Fuzzy and Soft Computing. Prentice Hall.

[26] Johnson, R.A., and Wichern, D.W. (1992) Applied Multivariate Statistical Analysis.

[27] Rein-Lien Hsu, M. Abdel-Mottaleb and A. K. Jain, "Face detection in color images," in IEEETransactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 696-706, May 2002, doi: 10.1109/34.1000242.

[28] E. Osuna, R. Freund and F. Girosit, "Training support vector machines: an application to face detection," Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 1997, pp. 130-136, doi: 10.1109/CVPR.1997.609310.

[29] <http://www.python.org/>

[30] Johnson, R.A., and Wichern, D.W. (1992) Applied Multivariate Statistical Analysis.

Prentice Hall. p356-395