**Mini Project Report on**



**Face Recognition using OpenCV**



**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name**  **University Roll No.**

**Ayush Kumar Singh 2021671**

***Under the Mentorship of***

**Mr. Rishi Kumar**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**July-2024**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Face Recognition using OpenCV”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Mr. Rishi Kumar, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name- Ayush Kumar Singh University Roll no.- 2021671

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction   * 1. Introduction   2. Background   3. Project Brief | **1**  **1**  **2**  **2** |
| Chapter 2 | Literature Survey | **4** |
| Chapter 3 | Methodology  3.1 Training the Face Recognizer  3.1.1 Directory Setup  3.1.2 Loading Haar Cascade  3.1.3 Creating LBPH Recognizer  3.1.4 Data Preparation  3.1.5 Training the Recognizer  3.2 Real-time Face Recognition  3.2.1 Loading Classifiers  3.2.2 Loading Label Mappings  3.2.2 Cap. Videos & Detecting Faces | **6**  **6**  **6**  **6**  **6**  **7**  **8**  **8**  **8**  **8**  **9** |
| Chapter 4 | Result and Discussion | **10** |
| Chapter 5 | Conclusion and Future Work | **11** |
|  | References | **12** |

**Chapter 1**

**Introduction**

In the following sections, a brief introduction and the problem statement for the work have been included.

* 1. **Introduction**

I chose to do a project based on Face Recognition using openCV for this mini project.

Every day, we are increasingly dependent on technology to carry out even the most basic of our actions. Face Recognition helps us in many ways, be it sorting photos in our mobile phone gallery by recognizing pictures with their face in them or unlocking a phone with a mere glance to adding biometric information in the form of face images in the country’s unique ID database (Aadhaar) as an acceptable biometric input for verification.

Face recognition has become a critical component in numerous modern applications, ranging from security systems and surveillance to identity verification and human-computer interaction. The advent of advanced computer vision techniques and the availability of powerful libraries like OpenCV has significantly facilitated the development of robust and efficient face recognition systems. This report aims to provide a comprehensive overview of building a face recognition system using OpenCV, detailing the process from training the face recognizer to implementing real-time face detection and recognition.

It also shows the practical implementation of face recognition using OpenCV with Python. The inspiration behind this is to make an available, easy-to-use, and accurate face recognition system that can be used in various projects with an increasing need for security and personalized user experiences, this technology offers a reliable solution for identifying and verifying different people.

The project aims to implement Facial Recognition on faces that the script can be trained for. The input is taken from a webcam and the recognized faces are displayed along with their name in real time. This project can be implemented on a larger scale to develop a biometric attendance system which can save the time-consuming process of a manual attendance system.

**1.2 Background**

At the beginning of the project, I had no previous experience with using the OpenCV library in my coding, so a large portion of my time was dedicated to investigating and understanding, as well as looking at alternative ways to figure out how to solve the problem.

Face recognition technology has a long history, with significant advancements over the past few decades. Early methods relied on geometric features of the face, such as the distances between the eyes, nose, and mouth. These methods were limited by their inability to handle variations in lighting, facial expressions, and poses. With the advent of machine learning and computer vision, more sophisticated techniques have been developed, leading to significant improvements in accuracy and robustness.

OpenCV, initially released in 2000, has become one of the most widely used libraries for computer vision applications. It provides a comprehensive set of tools for image and video processing, making it an ideal choice for implementing face recognition systems. OpenCV supports various face detection and recognition algorithms, including Haar Cascades, Local Binary Patterns Histograms (LBPH), and deep learning-based methods. This project leverages OpenCV's capabilities to create a face recognition system that is both efficient and effective

**1.3 Project Brief**

The project is divided into two main components:

* **Training a Face Recognizer:** Using images to train a model capable of recognizing different faces.
* **Real-Time Face Recognition:** Utilizing a webcam to detect and recognize faces in real time.

The training phase involves preparing a dataset of face images, detecting faces within these images, and training a recognizer model. The real-time recognition phase involves capturing video from a webcam, detecting faces in the video stream, and using the trained model to recognize these faces.

The objectives of the project are as follows:

1. To develop a face recognition system using OpenCV.
2. To implement a face detection module using Haar Cascades.
3. To train a face recognizer using the LBPH algorithm.
4. To perform real-time face recognition using a webcam.
5. To evaluate the performance of the face recognition system in different scenarios.

The project's scope includes collecting a dataset of face images, implementing the face detection and recognition modules, and testing the system's performance. The project does not cover advanced deep-learning methods or large-scale face recognition systems.

**Chapter 2**

**Literature Survey**

In this chapter, some of the major existing work in these areas has been reviewed.

1. **Eigenfaces [1]**: Proposed by Turk and Pentland in 1991, this method uses principal component analysis (PCA) to reduce the dimensionality of face images and identify significant features for recognition. Eigenfaces represent a set of eigenvectors, which are used to project face images into a lower-dimensional subspace for recognition. Despite its simplicity, the Eigenfaces method has been widely used and serves as a foundational technique in face recognition.
2. **Haar Cascades [2]**: Introduced by Paul Viola and Michael Jones in 2001, this method is widely used for object detection, including face detection, due to its high efficiency and accuracy. Haar Cascades uses a series of simple features, called Haar-like features, to detect objects in images. These features are computed rapidly using an integral image, and a cascade of classifiers is trained to detect faces with high accuracy. Haar Cascades are particularly effective for real-time face detection.
3. **Local Binary Patterns Histograms (LBPH)[3]**: A texture descriptor introduced by Ahonen, Hadid, and Pietikäinen in 2006, used for face recognition due to its robustness against varying lighting conditions. LBPH encodes the local texture information of an image by comparing the intensity of each pixel with its neighboring pixels, resulting in a binary pattern. These patterns are then used to create a histogram that represents the face image. LBPH is known for its simplicity and effectiveness in various lighting conditions.
4. **Deep Learning Models**: Recent advancements include the use of Convolutional Neural Networks (CNNs) for more accurate and robust face recognition. Notable works include DeepFace [4] by Taigman et al. (2014) and FaceNet [5] by Schroff et al. (2015). Deep learning models have revolutionized face recognition by achieving unprecedented accuracy and robustness. These models are trained on large datasets and can learn complex features and representations of faces, making them highly effective in real-world scenarios.
5. **Fisherfaces [6]**: Belhumeur, Hespanha, and Kriegman (1997) introduced Fisherfaces, which use Linear Discriminant Analysis (LDA) to project face images into a lower-dimensional space that maximizes the separation between different classes. Fisherfaces are particularly effective in handling variations in lighting and facial expressions.
6. **SIFT and SURF [7]**: Lowe (1999) and Bay et al. (2006) introduced the Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) algorithms, respectively. These methods extract key points and descriptors from images that are invariant to scale, rotation, and translation. SIFT and SURF have been used for face recognition, particularly in matching and aligning face images.
7. **Dlib [8]**: King (2009) developed Dlib, a modern C++ toolkit that includes machine learning algorithms and tools for creating complex software. Dlib provides state-of-the-art face detection and face recognition methods, including a highly accurate face landmark detector and a deep learning-based face recognition model.

**Chapter 3**

**Methodology**

The project methodology is divided into two primary parts: training the face recognizer and performing real-time face recognition.

##### **3.1 Training the Face Recognizer**

##### The training script is responsible for preparing the dataset, detecting faces, and training the recognizer model.

###### **3.1.1 Directory Setup**

###### Define the base directory where the training images are stored.

###### 

###### **Fig. 3.1.1 Directory Setup**

###### **3.1.2 Loading Haar Cascade for Face Detection**

**Haar Cascade Classifier:**

* **Purpose:** Haar Cascade is a machine-learning object detection algorithm used to identify objects in images or video feeds.
* **Method:** It uses a cascade of classifiers to detect objects by features called Haar-like features, which are derived from the intensity values of pixels.
* **Training:** Requires positive and negative images for training to learn how to distinguish the object from its background.

Initializing the Haar Cascade classifier for face detection.

###### 

**Fig. 3.1.2 Loading Haar Cascade**

###### **3.1.3 Creating the LBPH Recognizer**

**LBPH (Local Binary Patterns Histograms) Recognizer:**

* **Purpose:** LBPH is a texture-based face recognition algorithm.
* **Method:** It extracts local binary patterns from an image and uses histograms of these patterns to represent the face.
* **Training:** Requires a set of images of the faces to be recognized for training.
* **Recognition:** Compares the LBPH histograms of faces to recognize or classify faces based on similarity.

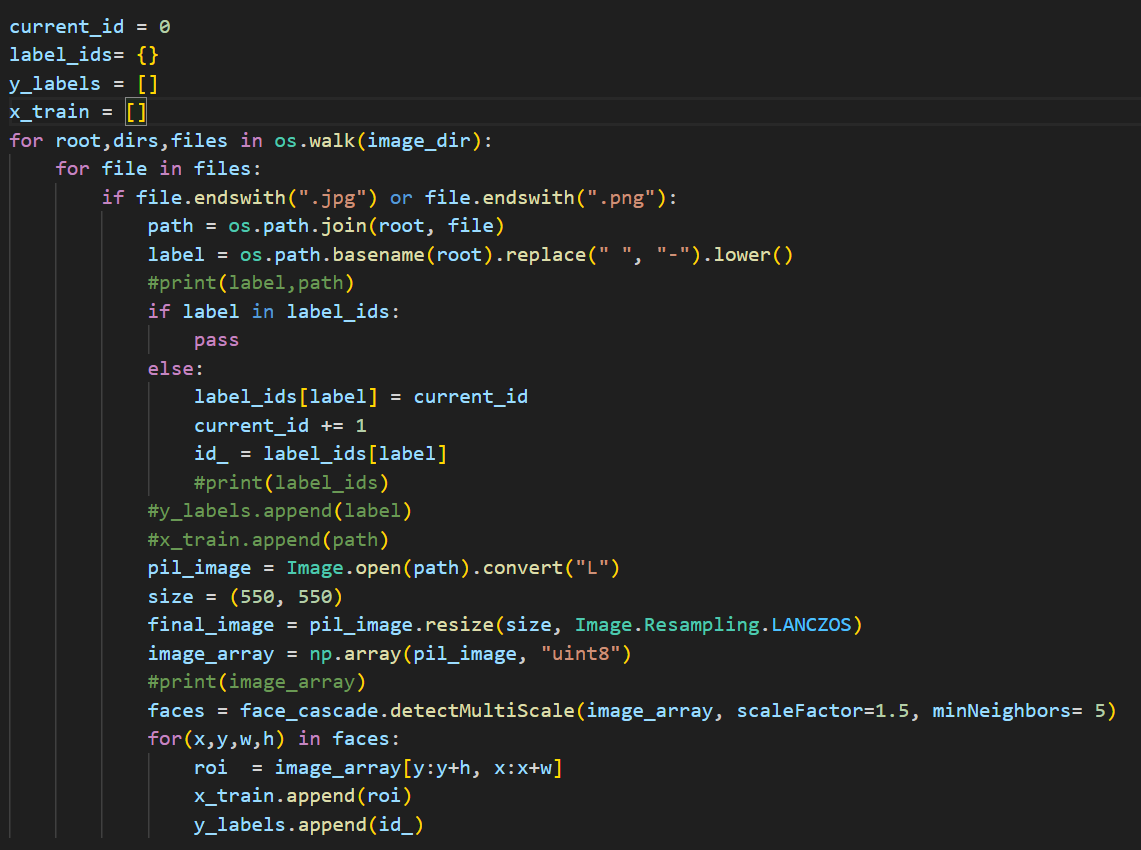
Initialize the Local Binary Patterns Histograms (LBPH) face recognizer.

###### 

**Fig. 3.1.3 LBPH Recognizer**

###### **3.1.4 Data Preparation**

Iterate through the image files, convert them to grayscale, detect faces, and extract the face regions (Regions of Interest, or ROIs).



**Fig. 3.1.4 Data Preparation**

###### **3.1.5 Training the Recognizer**

Collect the face regions and corresponding labels, train the LBPH recognizer, and save the trained model and label mappings.

##### 

**Fig. 3.1.5 Training the Recognizer**

##### **3.2 Real-time Face Recognition**

The real-time face recognition script captures video from a webcam, detects faces in the video stream, and recognizes these faces using the trained model. Initially, the system employs object detection techniques such as Haar Cascade to detect faces within each frame of the video feed or image. Once a face is detected, its location is determined through localization methods, marking its bounding box. The next step involves extracting features from the detected face, which can be done using methods such as LBPH (Local Binary Patterns Histograms). These features are then used to create a compact representation of the face, typically in the form of a feature vector. The system compares this vector against a database of known faces to recognize or verify the identity of the individual.

###### **3.2.1 Loading Classifiers and Recognizer**

Load the Haar Cascades for face and eye detection, and the trained LBPH recognizer model.

###### 

**Fig. 3.2.1 Loading Classifiers and Recognizer**

###### **3.2.2 Loading Label Mappings**

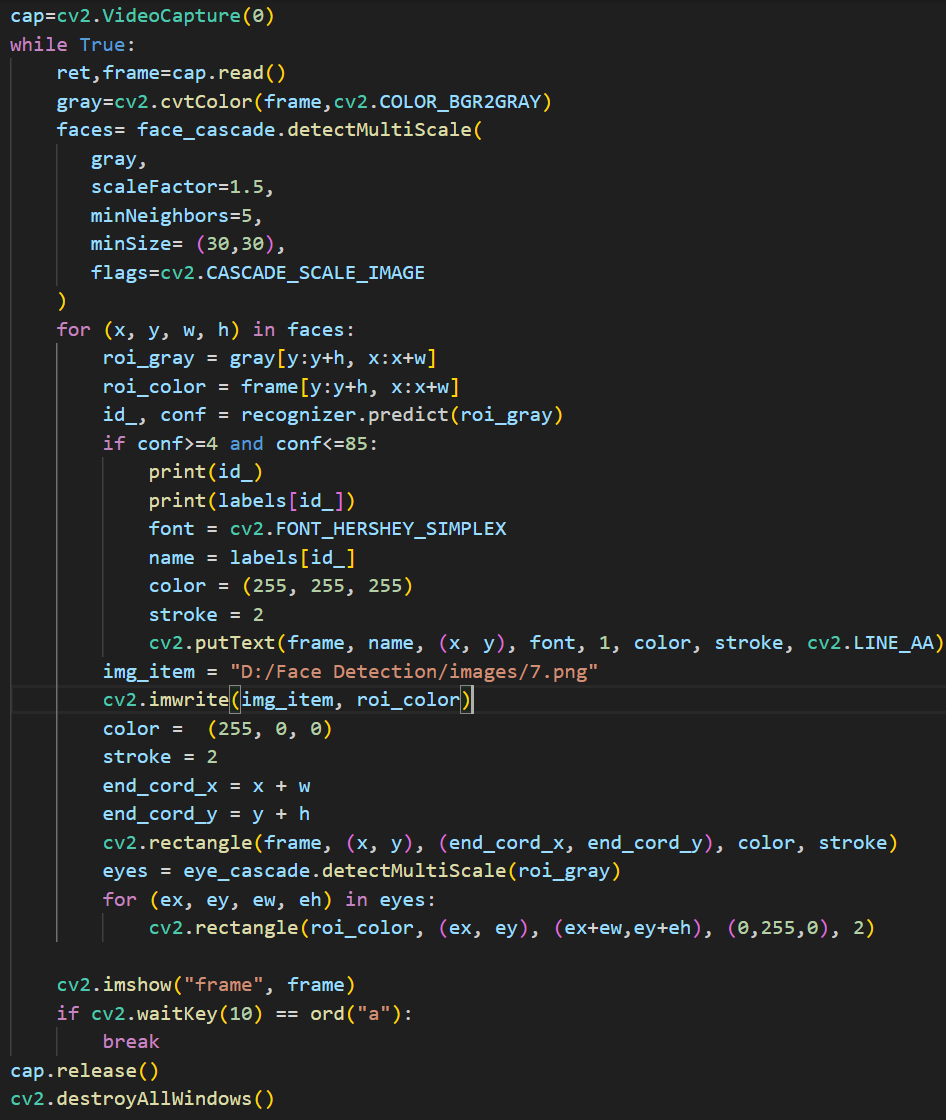
Load the label mappings from the pickle file and create a dictionary for label names.

###### 

**Fig. 3.2.2 Loading Label Mappings**

###### **3.2.3 Capturing Video and Detecting Faces**

Capture video from the webcam, convert frames to grayscale, and detect faces and eyes.

****

**Fig. 3.2.3 Capturing Video and Detecting Faces**

**Chapter 4**

**Result and Discussion**

The face recognition system successfully detects and recognizes faces in real time, as demonstrated by the scripts. The use of Haar Cascades for face detection provides a balance between accuracy and computational efficiency, while the LBPH algorithm offers robustness against varying lighting conditions and facial expressions.

###### **4.1 Training Phase Results:** The training phase involves creating a dataset of face images and labels, detecting faces within these images, and training the LBPH recognizer. The trained model achieves a good balance between accuracy and computational efficiency, making it suitable for real-time applications.

###### **4.2 Real-time Recognition Phase Results:** The real-time recognition phase involves capturing video from a webcam, detecting faces in the video stream, and recognizing these faces using the trained model. The system successfully detects and recognizes faces, annotating the video feed with the names of recognized individuals. The system's performance is robust, but it has limitations, such as sensitivity to lighting conditions, facial expressions, and partial occlusions.

###### **4.3 Limitations and Challenges**

* **Lighting Conditions**: The system's accuracy can be affected by varying lighting conditions, making it challenging to recognize faces in low light or with strong shadows.
* **Facial Expressions**: Changes in facial expressions can affect the system's ability to recognize faces accurately.
* **Partial Occlusions**: The presence of occlusions, such as hats, glasses, or masks, can hinder the system's performance.

**Chapter 5**

**Conclusion and Future Work**

It was an exciting project to work and there are a lot of things that I have learned from it. I was able to study a new topic that was completely new to me. Although I was not able to complete the project I have I desired I have completed the core purpose of the system that is working as I desired. This mini-project provides insights and solutions about Face Recognition and how to address these. This recognition system contributes to creating a digital environment that is safe for public use.

The current implementation of the face recognition system using OpenCV lays a solid foundation for recognizing faces in real time. However, several potential enhancements and improvements can be made to increase the system's robustness, accuracy, and functionality. The following sections outline some possible future enhancements.

In conclusion, there are numerous opportunities to enhance the face recognition system developed in this mini-project. By integrating advanced techniques such as deep learning, improving dataset quality, enhancing real-time performance, expanding functionality, and ensuring security and privacy, the system can be made more robust, accurate, and versatile. These enhancements will enable the face recognition system to be effectively used in a wide range of real-world applications.

**References**

[1] Matthew Turk and Allex Pentland, et al, “Eigenfaces for Recognition”,

https://www.face-rec.org/algorithms/PCA/jcn.pdf

[2] Paul Viola and Michael Jones, et al, “Rapid Object Detection using a Boosted Cascade of Simple Features”, <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>

[3] Ahonen, Hadid, et al, “Face Recognition with Local Binary Patterns”, https://www.researchgate.net/publication/221304831\_Face\_Recognition\_with\_Local\_Binary\_Patterns

[4] Taigman, Yang, et al, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”, https://www.cs.toronto.edu/~ranzato/publications/taigman\_cvpr14.pdf

[5] Schroff, Philbin, et al, “FaceNet: a unified embedding for face recognition and clustering”, https://doi.org/10.1109/CVPR.2015.7298682

[6] Belhumeur, Hespanha, Kriegman, et al, “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection”, https://doi.org/10.1109/34.598228

[7] Lowe, et al, “Object Recognition from Local Scale-Invariant Features”, https://www.cs.ubc.ca/~lowe/papers/iccv99.pdf

[8] Davis E. King, et al, “Dlib-ml: A Machine Learning Toolkit”, https://www.jmlr.org/papers/volume10/king09a/king09a.pdf