AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department Of Computer Engineering

A DL PROJECT REPORT

ON

HUMAN FACE RECOGNITION

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERINGAISSMS IOIT

BE Computer Engineering

SUBMITTED BY

STUDENT NAME	ERP No:
Onasvee Banarse	**09**
Kaustubh Kabra	**37**
Akash Mete	**50**
Harsh Shah	**67**

2022 -2023

AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report.
"Human Face Recognition"

Submitted by

STUDENT NAME	ERP No:
Onasvee Banarse	09
Kaustubh Kabra	37
Akash Mete	50
Harsh Shah	67

is a bonafide student at this institute and the work has been carried out by them under the supervision of **Prof. G.J.Navale** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(**Prof. G.J.Navale**) (**Dr. S.N.Zaware**)

Mini-Project Guide Head of Computer Department

Place: Pune Date:

Abstract

This Mini-Project report focuses on Human Face Recognition, which is a field of computer vision that has gained significant attention in recent years. The report presents an overview of the human face recognition process, including facial feature extraction, face detection, and face recognition algorithms.

The report also explores the various techniques used in face recognition, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Convolutional Neural Networks (CNNs). It discusses the advantages and limitations of each technique and presents examples of their application in real-world scenarios.

The mini project includes the implementation of a face recognition system using the OpenCV and face_recognition libraries in Python. The system performs face detection, feature extraction, and recognition on a set of input images, and outputs the results.

The report concludes with a discussion on the future scope of human face recognition and the challenges that need to be addressed to improve its accuracy and efficiency. Overall, this mini-project report provides a comprehensive understanding of human face recognition and its potential applications.

Contents

Introduction

Human face recognition is a fascinating field of study that has attracted significant attention from researchers, engineers, and technologists in recent years. This technology has become ubiquitous in our daily lives, from unlocking our smartphones to enabling security systems and identifying suspects in criminal investigations. The ability to recognize and identify faces is a fundamental human skill, and researchers have been studying this process for decades to understand how it works and how to replicate it in machines.

Human face recognition is a complex process that involves several cognitive and perceptual abilities, including the ability to detect faces, extract facial features, and compare them to stored representations of known faces. The process relies on both structural and surface information, such as the shape of the face, the position of facial features, and the texture of the skin. With recent advancements in computer vision and machine learning, algorithms have been developed that can perform these tasks with increasing accuracy, making human face recognition a vital technology in a wide range of applications, from security and surveillance to social media and entertainment. In this report, we will explore the various techniques used in human face recognition, their strengths and limitations, and their real-world applications.

Software Requirement Specification

Software Used:

- VScode –

  Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

- Python (version 3 or above)-

  Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. The sentiment analysis is performed using python language and packages.

- Jupyter Notebook or Google Colab –

  Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages".

Hardware Requirement

The detailed hardware requirements for the project are:

Item	Description
System	HP OMEN 15 series
Processor	AMD Ryzen 5 4600H
RAM	8 GB
System Type	64-bit operating system, x64-based processor
SSD	256 GB Solid State Drive
HDD	1 TB Hard Disk Drive
Graphics	NVIDIA 4 GB Graphic Card
Operating System	Windows 10 Operating System

Theory

Human face recognition is a process by which a computer system is able to identify and verify the identity of a person from an image or video stream. It is an important technology that has applications in various fields, including security, entertainment, and social media.

Human face recognition involves several steps, including face detection, face alignment, and feature extraction. Face detection is the process of locating faces in an input image or video stream. This is typically done using computer vision techniques, such as Haar cascades or convolutional neural networks (CNNs). Once the faces have been detected, the next step is face alignment, which involves aligning the faces to a standard pose to facilitate feature extraction. This is typically done by detecting the facial landmarks, such as the eyes, nose, and mouth, and transforming the face to a common orientation.

Feature extraction is the process of extracting features from the aligned faces that can be used for recognition. Common feature extraction methods include Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and deep learning-based methods such as Convolutional Neural Networks (CNNs).

Methods Used for Human Face Recognition:

1. **Principal Component Analysis (PCA):**

   PCA is a statistical method that is commonly used for face recognition. It involves finding the principal components of a set of faces and using these components to represent and recognize new faces.

2. **Linear Discriminant Analysis (LDA):**

   LDA is a supervised machine learning method that is used for feature extraction and classification. It involves finding the features that are most useful for distinguishing between different classes of faces.

3. **Local Binary Patterns (LBP):**

LBP is a texture-based method that extracts features based on the local patterns of pixel intensities in an image.

4. **Deep Learning:**

   Deep learning methods, such as Convolutional Neural Networks (CNNs), have revolutionized face recognition in recent years.

5. **3D Face Recognition:**

   3D face recognition uses three-dimensional models of faces to recognize individuals. It is often used in security applications where a high level of accuracy is required.

6. **Hybrid Approaches:**

   Many face recognition systems use a combination of different methods to achieve better performance.

Python Libraries:

face_recognition lib:

The `face_recognition` library is a popular Python library for face recognition and face detection tasks. It is built on top of the OpenCV library and uses deep learning algorithms for face recognition. The library provides a simple and easy-to-use interface for face recognition tasks, including face detection, face alignment, and face comparison.

1. Face Detection
2. Face Alignment
3. Feature Extraction
4. Face Comparison
5. Simple API

Overall, the `face_recognition` library is a powerful and easy-to-use tool for face recognition tasks in Python. It is widely used in various applications, including security systems, social media, and entertainment.

Code

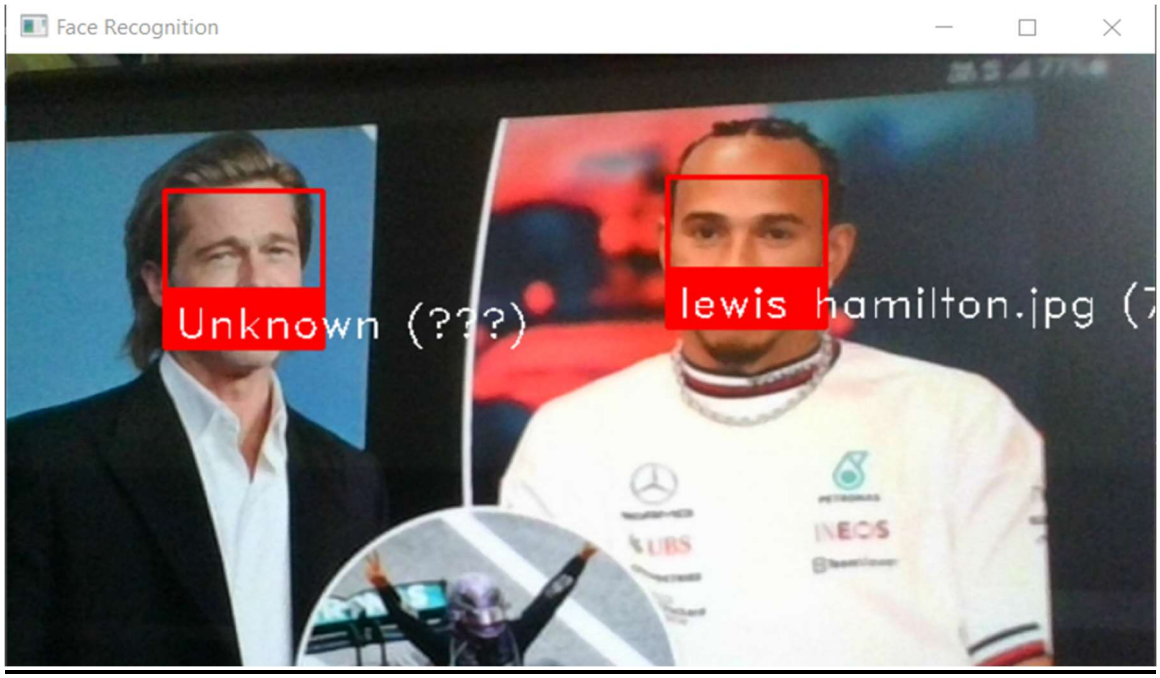
➢ Code:

```
import face_recognition
import os, sys
import cv2
import numpy as np
import math
class FaceRecognition:
    face_locations = []
    face_encodings = []
    face_names = []
    known_face_encodings = []
    known_face_names = []
    process_current_frame = True
    number_of_times_to_upsample = 3
    model = "hog"
    def __init__(self):
        self.encode_faces()
    # noinspection PyCompatibility
    def run_recognition(self):
        video_capture = cv2.VideoCapture(0)


        if not video_capture.isOpened():
            sys.exit('Video source not found...')
        while True:
            ret, frame = video_capture.read()
            # Only process every other frame of video to save time
            if self.process_current_frame:
                # Resize frame of video to 1/4 size for faster face recognition processing
```

```
            small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
            # rgb_small_frame = small_frame[:, :, ::-1]
            rgb_small_frame = np.ascontiguousarray(small_frame[:, :, ::-1])
                # See if the face is a match for the known face(s)
                matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
                name = "Unknown"
                confidence = '???'
                # Calculate the shortest distance to face
                face_distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    name = self.known_face_names[best_match_index]
                    confidence = face_confidence(face_distances[best_match_index])
                self.face_names.append(f'{name} ({confidence})')
        self.process_current_frame = not self.process_current_frame


        # Display the results
        for (top, right, bottom, left), name in zip(self.face_locations, self.face_names):
            # Create the frame with the name
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        # Display the resulting image
        cv2.imshow('Face Recognition', frame)
        # Hit 'q' on the keyboard to quit!
        if cv2.waitKey(1) == ord('q'):
    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()
if __name__ == '__main__':
    fr = FaceRecognition()
    fr.run_recognition()
```

Output


Face Recognition
ms dhoni.jpg (89.12%)
sachin tendulkar.jpg (69.25%)


Face Recognition
Unknown (???)
lewis hamilton.jpg (

Conclusion

In conclusion, human face recognition is a complex and challenging task that has many applications in various fields, including security, entertainment, and social media. In this mini project, we explored the different methods and techniques used for human face recognition, including PCA, LDA, LBP, deep learning, and hybrid approaches. We also looked at how the `face_recognition` library can be used in Python for face detection, alignment, feature extraction, and comparison.

The `face_recognition` library provides a powerful and easy-to-use tool for face recognition tasks in Python, and it can be used in a variety of applications. However, it is important to note that face recognition technology raises privacy and ethical concerns, and it is important to use it responsibly and in accordance with applicable laws and regulations.

Overall, this mini project introduced the theory and practical implementation of human face recognition, and it highlighted the importance and potential of this technology in various applications.

References

Books:

1. Face Recognition: A Literature Survey by Samarth Bharadwaj and K. R. Ramakrishnan. Published in the International Journal of Computer Applications, Volume 139, Number 11, February 2016.
2. Deep Face Recognition: A Survey by Yan Xu, Wei Chen, Liangjiang Yu, and Junbao Zhuo. Published in the IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 42, Issue 12, December 2020.
3. OpenCV: Computer Vision Projects with Python by Joseph Howse, Prateek Joshi, and Michael Beyeler. Published by Packt Publishing, 2018.
4. Handbook of Face Recognition by Stan Z. Li and Anil K. Jain. Published by Springer, 2011.
5. Face Recognition Using Deep Learning: A Survey by Ayush Jaiswal, Aayushi Mahajan, and Pankaj Mishra. Published in the Journal of Ambient Intelligence and Humanized Computing, Volume 12, pages 4301–4317, 2021.

Websites:

1. OpenCV documentation: https://docs.opencv.org/
2. `face_recognition` library documentation: https://face-recognition.readthedocs.io/en/latest/index.html
3. Face Recognition Homepage: http://www.face-rec.org/
4. PyImageSearch: https://www.pyimagesearch.com/
5. Deep Learning for Computer Vision with Python by Adrian Rosebrock: https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/