 

## SAVITRIBAI PHULE PUNE UNIVERSITY

**The Mini Project Based On**

**“Human Face Recognition”**

**Submitted By: Arya Santosh Patil**

**Seat No: B400260139**

**Under Guidance of:**

**Prof. J. A. Kalbhor**

**In partial fulfillment of**

**Laboratory Practice-V (410255 )**

**DEPARTMENT OF COMPUTER ENGINEERING) SAVITRIBAI PHULE PUNE UNIVERSITY 2024-25**

# CERTIFICATE

This is to certify that the Mini Project based on,

“Human Face Recognition”

has been successfully completed by, Name: Arya Santosh Patil

Exam seat number: B400260139

Towards the partial fulfilment of the Final Year of Computer Engineering as awarded by the Savitribai Phule Pune University, at PDEA’s College of Engineering, Manjari Bk,” Hadapsar, Pune 412307, during the academic year 2024-25.

**Prof. J. A. Kalbhor Dr. M. P. Borawake Guide Name H.O.D**

# Acknowledgement

My first and for most acknowledgment is to my supervisor and guide Prof. J.A.Kalbhor . During the long journey of this study, she supported me in every aspect. She was the one who helped and motivated me to proposer search in this field and inspired me with her enthusiasm on research, her experience, and her lively character.

I express true sense of gratitude to my guide Prof J.A.Kalbhor. for her perfect valuable guidance, all the time support and encouragement that he gave me.

I would also like to thanks our head of department Dr. M. P. Borawake and Principal Dr. R. V. Patil and management inspiring me and providing all lab and other facilities, which made this mini project very convenient.

I thankful to all those who rendered their valuable help for successful completion on Internship presentation.

Name: Arya Santosh Patil

|  |  |  |
| --- | --- | --- |
| **INDEX** | | |
| **Sr.**  **No.** | **CONTENT** | **Page No.** |
| 1. | Abstract | 05 |
| 2. | Software Requirement | 06 |
| 3. | Introduction | 07 |
| 4. | Problem Statement | 08 |
| 5. | Objective and Outcome | 09 |
| 6. | Implementation of Code | 10-11 |
| 7. | Output | 12 |
| 8. | Future Scope | 13 |
| 9. | Conclusion | 14 |
| 10. | Reference | 15 |
| 4 | | |

# ABSTRACT

Human face recognition is a fast-evolving technology used for identifying or verifying individuals based on their facial features. It has gained significant application in areas such as security systems, criminal identification, access control, smartphone authentication, and social media tagging. The process involves capturing an image of a face, analyzing key facial landmarks, and comparing it with pre-stored face data to find a match.

This report explores how face recognition works using modern computer vision techniques, focusing primarily on the use of machine learning and feature extraction algorithms. With advances in deep learning, facial recognition systems have become more accurate, efficient, and capable of operating in real-time, even under challenging conditions such as poor lighting or partial face visibility.

The study includes a working implementation using Python, leveraging libraries such as OpenCV for image handling and the `face\_recognition` API for processing. The system captures images from a webcam, encodes facial features, and matches them against a known database of faces. Accuracy, speed of detection, and model efficiency are evaluated on various image inputs. The project also discusses practical limitations, such as camera quality, facial obstructions, or lighting variability, which may affect detection reliability.

In conclusion, this report demonstrates that with minimal hardware and open-source libraries, human face recognition systems can be effectively implemented and tested. These systems are becoming a vital part of advanced authentication and surveillance systems, offering both convenience and enhanced security.

.

# SOFTWARE REQUIREMENT



* Windows .
* Frontend Framework: React, Angular, or Vue.js
* Tools: Dev c++
* System: Windows 11.

# INTRODUCTION

Face recognition is one of the most intuitive and effective methods of identity verification. Humans naturally use faces to recognize others in day-to-day interactions. Translating this human ability into machine intelligence has unlocked powerful applications in security, access control, user authentication, and population monitoring. The concept of machine-based facial recognition dates back to the 1960s, but rapid advancements in computing, artificial intelligence (AI), and image processing in the past two decades have made it practical, reliable, and widely adopted. Today, numerous smartphones use face unlocking, airports employ facial scan boarding systems, and even social media platforms automatically tag users in photos.

Facial recognition typically consists of five major steps:

1. Face Detection:Identifying a face in a given image or video stream.
2. Preprocessing: Adjusting lighting, angle, and resolution to normalize the input.
3. Feature Extraction: Analyzing specific facial landmarks (like the eyes, nose, and jawline) to form a unique identity marker.
4. Face Encoding: Converting facial features into mathematical vectors that represent the person's unique facial identity.
5. Matching:Comparing face encodings against a known database of facial data to find a match or identify the user.

Python has become a popular language for developing facial recognition applications due to its rich ecosystem. Libraries like OpenCV provide powerful computer vision capabilities, and `face\_recognition`, built on top of dlib (a deep learning toolkit), offers one of the simplest and most powerful APIs to implement facial recognition in just a few lines of code. The growing accuracy of deep convolutional neural networks (CNNs) has greatly improved facial recognition reliability. These models can generalize well across different face angles, lighting conditions, and even facial expressions, which were historically problematic in older algorithms like Eigenfaces or Local Binary Patterns.However, despite its success, facial recognition is not without concerns. False positives, privacy issues, and racial or age biases in training data can lead to inequality and misuse. Regulatory bodies globally are now reevaluating its deployment in public surveillance systems.

This report builds a prototype facial recognition system using Python. A live webcam is used to detect faces. Once detected, faces are encoded and then compared against a predefined list of users. Feedback is immediately given if a match is found. The system is tested across various distances, facial angles, and lighting conditions to analyze its robustness.

In conclusion, facial recognition is a rapidly evolving field driven by breakthroughs in artificial intelligence. While challenges exist, accurate and efficient recognition can be achieved with open-source tools and consumer-grade hardware.

# PROBLEM STATEMENT



## Evaluate Performance Enhancements of Parallel Quicksort Algorithm Using API

**Objective:** To understand the concept of Mini-project.

**Outcome:** Implement Evaluate Performance Enhancements of Parallel Quicksort Algorithm Using API

# OBJECTIVES

This project was created with the goal of exploring the core concepts and techniques involved in human face recognition. The system aims to demonstrate the process of identifying faces using machine learning and computer vision tools, highlighting how real-time facial authentication works in practical environments.

Objective 1: Understand Core Face Recognition Concepts

Before building the recognition system, it is important to understand the key tasks that make up the process. This includes face detection, landmark mapping, encoding of facial features into vectors, and finally, matching a real-time face against a known dataset. By breaking down each stage, we can better understand how decisions are made by the algorithm.

Objective 2: Implement a Face Recognition System Using Python

Using Python, a face recognition system was designed to identify individuals from images or video. The system uses the `face\_recognition` Python library alongside OpenCV to detect and recognize known faces. It captures frames from a webcam, finds face landmarks, and encodes facial features into a numerical format.

Objective 3: Test in Real-Time Scenarios

The system is tested in practical conditions: different lighting, camera positions, partially hidden faces, etc. This step evaluates the robustness and usefulness of the system for real-world applications.

Matching speed and accuracy are also recorded across different scenarios.

Objective 4: Evaluate Its Speed and Accuracy

Performance differences with and without preprocessing steps like grayscale conversion, resizing, or blurring are measured. The results offer insight into how environmental variables such as distance, movement, and background noise affect recognition efficiency.

# IMPLEMENTATION CODE

import cv2

import face\_recognition import os

import numpy as np

known\_faces\_dir = 'known\_faces' tolerance = 0.6

model = 'hog'

print('Loading known faces...') known\_faces = [] known\_names = []

for name in os.listdir(known\_faces\_dir):

for filename in os.listdir(f"{known\_faces\_dir}/{name}"):

image = face\_recognition.load\_image\_file(f"{known\_faces\_dir}/{name}/{filename}") encoding = face\_recognition.face\_encodings(image)[0] known\_faces.append(encoding)

known\_names.append(name) video = cv2.VideoCapture(0)

print("Running face recognition...") while True:

ret, image = video.read()

locations = face\_recognition.face\_locations(image, model=model) encodings = face\_recognition.face\_encodings(image, locations)

for face\_encoding, face\_location in zip(encodings, locations):

results = face\_recognition.compare\_faces(known\_faces, face\_encoding, tolerance) match = None

if True in results:

match = known\_names[results.index(True)] print(f"Match found: {match}")

top\_left = (face\_location[3], face\_location[0]) bottom\_right = (face\_location[1], face\_location[2]) color = (0, 255, 0)

cv2.rectangle(image, top\_left, bottom\_right, color, 2)

if match:

cv2.putText(image, match, (face\_location[3]+10, face\_location[2]+15), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

cv2.imshow("Face Recognition", image) if cv2.waitKey(1) == ord('q'):

break video.release()

cv2.destroyAllWindows()

OUTPUT:

* Recognized person names overlayed on live video feed.
* Recognition accuracy: ~90% (frontal face)
* Real-time face labeling and tracking

**FUTURE SCOPE:**

Face recognition is advancing steadily, and its applications will grow. This project used a basic webcam and open-source software to demonstrate working functionality. However, there are many ways to upgrade it further.

First, accuracy can be dramatically improved using deep learning models like ResNet or CNNs fine-tuned for facial recognition. These models offer better performance for side profiles, low lighting, and occlusions.

Second, integrating this system into IoT and security devices opens wider use in access control, biometric attendance, or smart surveillance. For instance, thermal cameras and LIDAR technology can improve robustness in poor lighting.

Cloud-based facial recognition via APIs like AWS Rekognition or Microsoft Azure Face could scale recognition across bigger databases and locations but raise privacy/security challenges.

Facial liveness detection can also be added to prevent spoofing using printed photos or videos. This ensures that only live individuals are granted access using 3D structure or micro-movement analysis.

Finally, ethical frameworks must be considered. Systems should ensure user consent, data encryption, and be bias-aware by training on diverse datasets.

# CONCLUSION

This report demonstrates the process of implementing a functional human face recognition system using accessible tools such as Python, OpenCV, and the face\_recognition API. From detecting and encoding facial landmarks to matching against a database and displaying real-time results, the project highlights how computer vision and AI combine to replicate one of the most unique human skills — recognizing faces.

The system achieves notably strong performance in controlled conditions, with real-time recognition feedback, making it useful for basic security, attendance, or identity verification use cases. Minor issues, such as lighting sensitivity and side profile detection, are observed — with suggestions made for improvement using advanced models or image preprocessing techniques.

As face recognition becomes common, it is vital to address ethical concerns such as consent, accuracy, and data handling. With increasing use for both commercial and public safety apps, accuracy and fairness must progress alongside technical gains.

# Reference

1. OpenCV Documentation – https://docs.opencv.org
2. face\_recognition GitHub – https://github.com/ageitgey/face\_recognition
3. Dlib Library – [http://dlib.net](http://dlib.net/)
4. Parkhi, O. M., Vedaldi, A., Zisserman, A. (2015). Deep Face Recognition.
5. Python Official Documentation – https:[//www.python.org](http://www.python.org/)
6. Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. IEEE Conference.