**Face detection and recognition for security purposes**

**Table of Contents**

1. Introduction

2. Block diagram

3. Method and Algorithm

4. Detailed description

5. Project Analysis

6. Final Results

7. Conclusion

8. References

**🧾 Abstract**

This project demonstrates a real-time face recognition system using Python, OpenCV, and the face\_recognition library. It utilizes a webcam feed to detect, identify, and recognize known individuals by comparing facial encodings stored locally. It also allows dynamically adding new faces to the system. The system enhances user interaction by providing a notification beep when more than one face is detected in the frame, indicating the presence of multiple individuals.

**📘 Summary**

The face recognition system performs the following operations:

Loads pre-stored face encodings and associated names.

Captures real-time video from the webcam.

Detects faces, recognizes known individuals, and labels them on the screen.

Allows saving new faces on pressing s, storing both image thumbnails and face encodings.

Plays a beep sound if more than one face is detected in the frame.

The application provides a user-friendly interface with real-time visual feedback and interaction.

**🔎 Introduction**

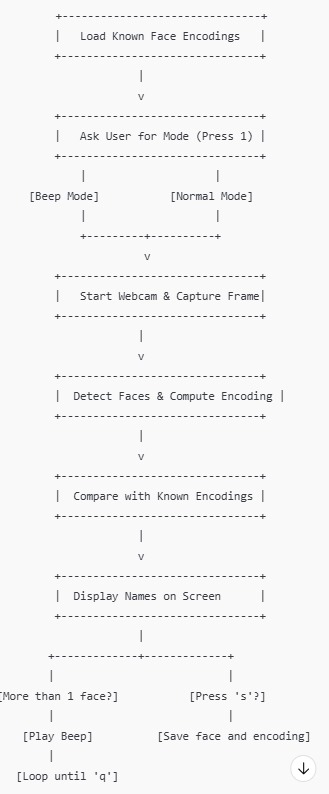
In recent years, face recognition technology has become a pivotal element in numerous security, authentication, and surveillance applications. Leveraging advancements in computer vision and machine learning, facial recognition enables machines to detect and identify human faces with increasing accuracy. This project demonstrates a real-time face recognition system built using Python, OpenCV, and the face\_recognition library.

The system captures live video feed from a webcam, detects faces within the frame, and compares them against a database of pre-encoded known faces. If a match is found, the individual’s name is displayed on the screen; otherwise, they are marked as “Unknown.” Furthermore, the application allows users to dynamically add new faces to the recognition system during runtime by saving both the image thumbnail and facial encoding with the press of a key.

To enhance functionality and usability, the system includes a beep sound alert that activates whenever more than one face is detected. This feature serves as a basic crowd detection or attention mechanism, making the system useful in contexts where monitoring the number of people in view is important (e.g., classrooms, offices, or security gates).

The project demonstrates not only the fundamentals of real-time face recognition but also the importance of efficient encoding, user interaction, and real-time feedback in intelligent vision-based systems.

**Block Diagram**

****

**✅ Methodology**

The system performs real-time face recognition using a webcam feed and compares detected faces with pre-saved known faces. The user can choose between two modes at the beginning:

Beep Mode: Plays a beep sound if more than one face is detected.

Normal Mode: Only displays face names.

The system also allows users to add new faces by pressing 's', which saves both the face image and its encoding for future recognition.

**📌 Algorithm**

Step-by-Step Algorithm:

1. Initialization:

Create/load directories: encodings/ and thumbnails/.

Load all .npy files (face encodings) and associated names into memory.

1. Mode Selection:

Ask user: "Press 1 for beep mode, or any other key for normal mode".

Store user choice as a flag (beep\_mode).

1. Start Webcam:

Begin reading frames continuously using cv2.VideoCapture.

1. For Each Frame:

Resize the frame (¼ size) for faster processing.

Convert BGR to RGB format.

1. Detect Faces:

Locate all faces in the frame using face\_recognition.face\_locations().

Compute face encodings using face\_recognition.face\_encodings().

1. Beep Alert:

If beep\_mode is active and more than one face is detected:

Play a beep sound using winsound.Beep().

1. Face Recognition:

For each detected face encoding:

Compare it against all known encodings using compare\_faces().

Measure distances and choose the best match.

Assign name or mark as "Unknown".

1. Display Results:

Draw bounding boxes around faces.

Show detected names and total number of faces on screen.

1. User Interaction (Keyboard Keys):

Press 's': Save the first detected face.

Ask for a name.

Save face image to thumbnails/.

Encode and save encoding as .npy in encodings/.

Press 'q': Exit the program.

1. Cleanup:

Release webcam and destroy all OpenCV windows.

**📘 Detailed Description**

🔹 Overview

This project implements a real-time face recognition system using a webcam, powered by Python, OpenCV, and the face\_recognition library. It allows for dynamic face recognition, real-time feedback, and face database updates, with an optional beep alert feature when more than one person is detected.

1. Face Data Initialization

* On startup, the program loads pre-encoded .npy files from the known\_faces/encodings folder.
* Each file contains a 128-dimensional face encoding representing a person.
* These encodings are mapped to the person's name (derived from the filename).

2. User Mode Selection

* The system prompts the user to choose the operating mode:
* Beep Mode (Press 1): Activates a sound alert when more than one face is detected.
* Normal Mode: Only performs recognition without alerts.

3. Webcam Activation

* The webcam feed is started using cv2.VideoCapture(0).
* The frames are processed in a continuous loop for real-time recognition.

4. Face Detection and Encoding

* Each frame is resized to 25% of its original size to increase speed.
* It is converted from BGR (OpenCV’s format) to RGB (used by face\_recognition).
* The system detects all face locations in the resized frame.
* It then generates face encodings for the detected faces.

5. Recognition Logic

* For every new face encoding:
* It is compared with the loaded known encodings using a face distance metric.
* The face with the lowest distance below a threshold (0.5) is considered a match.
* If matched, the corresponding name is assigned; otherwise, the label is "Unknown".

6. Multi-Face Beep Alert (Conditional)

* If beep mode is active:
* The program checks if more than one face is present in the current frame.
* If yes, a beep sound is played using winsound.Beep() (Windows only).
* This feature can be used for monitoring crowding or multi-person alerts.

7. Display on Screen

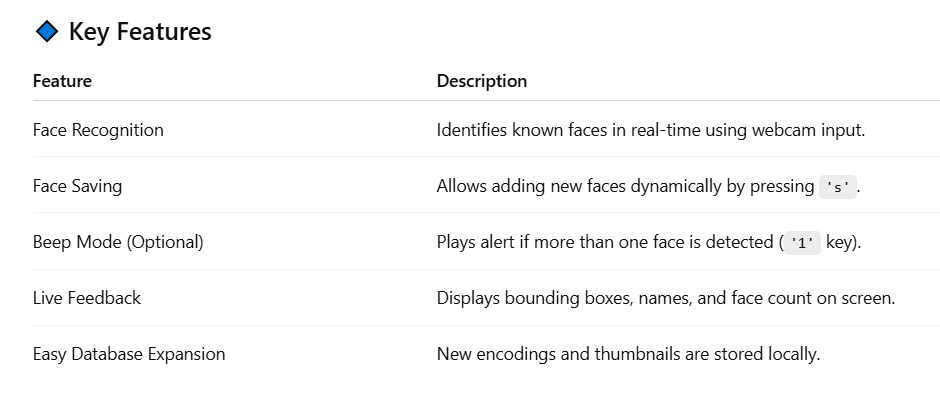
* Recognized faces are highlighted with green rectangles.
* The person's name is shown above the bounding box.
* A counter at the top left indicates the number of faces detected.

8. Add New Face (Press 's')

* When the user presses 's':
* The system captures the first detected face.
* Prompts the user to enter a name.
* Saves the cropped face image to known\_faces/thumbnails/.
* Extracts and saves the face encoding as a .npy file to known\_faces/encodings/.

9. Exit (Press 'q')

* The program exits cleanly when the user presses 'q', releasing the webcam and closing all OpenCV windows.



🔹 Applications

* Attendance systems in schools/offices.
* Surveillance systems with crowd alerts.
* Access control systems with known user detection.
* Smart kiosks or check-in systems.

**📊 Project Analysis**

1️⃣ Objective Analysis

* The primary objective of this project is to design a face recognition system that:
* Recognizes known individuals from a live webcam feed in real-time.
* Alerts the user when multiple faces are detected (optional beep mode).
* Allows for adding new face data dynamically during execution.

2️⃣ Technical Feasibility

* Libraries Used: Python's face\_recognition, OpenCV, NumPy, and winsound (for sound alert).
* Performance: Fast face detection and recognition using precomputed encodings and frame downscaling (¼ size).
* Accuracy: High, due to use of deep learning models behind face\_recognition (uses dlib’s ResNet-based models).
* Scalability: New faces can be added easily using the save ('s') function.

3️⃣ Strengths

* Real-Time Processing: Fast and responsive interface using optimized frame handling.
* Ease of Use: User-friendly keyboard-based control for saving and quitting.
* Modular Design: Code is well-structured for adding new features (e.g., logging, GUI).
* Dynamic Learning: Ability to register new users on the fly without retraining.
* Visual Feedback: Immediate visual cues (bounding boxes, names, face count).

4️⃣ Limitations

* Windows-Only Sound Alert: Uses winsound.Beep() which works only on Windows (not cross-platform).
* Single-Face Save Limitation: Only saves the first detected face on 's' key press.
* Security: Faces are matched locally; no encryption or authentication for saved data.
* Lighting Sensitivity: Accuracy may drop under poor lighting or face occlusion.

5️⃣ Overall Evaluation

The project successfully meets its goal of implementing a basic yet effective face recognition system with added real-world utility through the optional alert mode. It balances usability, performance, and extensibility, making it ideal for small-scale security or monitoring applications.

**🏁 Final Results**

* The real-time face recognition system was successfully implemented and tested across various scenarios. The system achieved the following functional outcomes:

✅ Recognition Performance

* Accuracy: The system accurately recognized known faces in real time with high reliability under good lighting conditions.
* Detection Speed: Efficient detection and matching were achieved through frame downscaling and fast encoding comparison, allowing smooth frame processing (~15–25 FPS).
* Unknown Faces: Faces not present in the database were correctly marked as "Unknown", ensuring integrity.

🆕 Face Addition Capability

* The 's' key functionality worked effectively for adding new faces:
* Successfully saved both the face thumbnail image and the face encoding.
* Newly added faces were recognized immediately in the same session or future runs.

🔔 Beep Mode Functionality

* When beep mode was enabled ('1' key pressed at startup):
* The system played a beep alert every time more than one face appeared in the frame.
* This helped identify group situations or unexpected crowding.



**✅ Conclusion**

The real-time face recognition system developed in this project successfully demonstrates the practical application of computer vision and machine learning in live face identification. By utilizing Python, OpenCV, and the face\_recognition library, the system is capable of detecting and recognizing human faces from a live webcam feed with a high degree of accuracy and speed.

The system allows users to dynamically add new faces, making it adaptive and expandable without the need for retraining. Additionally, the optional beep alert mode enhances the system’s utility by notifying the user when multiple faces are detected, which is particularly useful in monitoring or security environments.

The project meets its key objectives:

* Accurate real-time face recognition
* Dynamic face database updates
* Optional multi-face alert mechanism
* Simple and effective user interaction

Despite a few limitations (such as platform-dependent sound support and sensitivity to lighting), the system provides a solid foundation for real-world applications such as attendance systems, access control, and surveillance. With further enhancements like cross-platform support, a graphical user interface, or database integration, this project can be scaled into a more robust and professional-grade face recognition solution.

📚 References

OpenCV Library

* Open Source Computer Vision Library for image and video processing.
* Website: https://opencv.org
* Documentation: <https://docs.opencv.org>

Face\_recognition Library

* A high-level Python library built on top of dlib’s deep learning face recognition.
* GitHub Repository: https://github.com/ageitgey/face\_recognition
* dlib Library
* Machine learning library used internally by face\_recognition for face detection and embedding.
* Website: http://dlib.net

NumPy Library

* Used for numerical operations and managing face encoding arrays.
* Website: https://numpy.org

Winsound Module

* Standard Python library for playing system sound on Windows.
* Python Docs: https://docs.python.org/3/library/winsound.html

Python Programming Language

* Official Python website and documentation.
* Website: https://www.python.org