# FACE RECOGNITION WITH REALTIME DATABASE

**A Project Work Synopsis**

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

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE WITH SPECIALIZATION IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**


**Submitted by:**

21BCS6116 – HARI LUNAVATH

21BCS6274 - T KARTHIKEYA

21BCS6323 – VIGNAN KUMAR

21BCS6003 – SHIVA REDDY

**Under the Supervision of:**

Dr Priyanka Kaushik



CHANDIGARH UNIVERSITY

Discover. Learn. Empower.


**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB**


**September, 2022**

# Abstract

The aim of this study is to create a face recognition system based on real-time video. This article basically establishes four guidelines for considering the issue: transmitting the cost of video time in percent to confirm participation, security of video time of real video, accuracy of the system at login, and a real-time video processing system. The effectiveness of AFR technology has increased significantly over the past few years and these systems are now used for security and commercial purposes. Student participation is an important task in the classroom. When done manually, it usually takes a lot of lesson time that can be used for learning.An auto-enrollment system using facial recognition is a suggested solution to an existing problem. The face is an important part of a person. This tutorial shows how to detect and recognize human faces in real time using Arduino UNO. This project uses the open source graphics library OpenCV to define an efficient algorithm. Face Recognition, Face Progression, Face Training, Face Recognition and Interaction Data are the five models that make up our approach. We built a face database to recognize students' faces. The student database contains the faces of all students and is initially used to train the algorithm. The system uses a user-friendly interface to maximize the user experience while collecting student images and engagement throughout training and testing. The program can also use many applications that can use facial recognition for authentication. Using Arduino UNO can reduce product cost and increase performance because it can be connected to any device for now.The system automatically sends messages to the department heads and advisors of the students who are absent, informing them about the student attendance status and updating their student records. The unique thing about our model is that it has poll signing capability; this means that the student must be in the classroom for at least 20 or 30 minutes out of a total of 50 minutes before our signature model for students. Participating in facial recognition can reduce meeting times by up to 60%.

Keywords— Automatic face recognition, attendance system, Arduino UNO, OpenCV, face detection, face preprocessing, face

## Table of Contents

# 1. INTRODUCTION

The use of face recognition system with real time database has become increasingly popular in recent years. The goal of this project is to create a facial recognition system with a real-time database that can reliably record a student's attendance when they are really in the classroom. This chapter introduces the to the need, importance and potential impact on the users. We also provide an overview of the project, its objectives, and the data used in the model.

**1.1 Problem Definition** The current system for keeping attendance is manual. Both professors and students lose a lot of time because of it. If attendance is taken manually, the students must wait longer. While attendance is recorded manually, there are still opportunities for proxies to attend the class.Human error is a constant expense of manual attendance.Any human may be recognised by their face alone. Hence, automating the attendance procedure will boost class productivity.

**1.2 Problem Overview** we want to replace the traditional attendence marking system to smart attendence marking system by using machine learning and our final outcome is going to product Our plan is to take the data of one class and train model with the test data and push into the embedded systems we will connect it to the camera so it can recognise it a nd we give 30 minutes long timming then only it mark present all this information comes to the excel sheet

## 1.3 Hardware Specification

- Arduino Uno Board: You will need an Arduino Uno board, which is a microcontroller board based on the ATmega328P microcontroller.
- Raspberry Pi Camera Module: This camera module is compatible with the Raspberry Pi single-board computer and comes in two versions, one with an 8-megapixel sensor and another with a 12-megapixel sensor. It is relatively inexpensive and offers high-quality images in a compact form factor.
- USB Cable: A USB cable is used to connect the Arduino board to your computer. You will need a USB cable that has a Type-A USB connector on one end and a Type-B USB connector on the other end.

## 1.4 Software Specification

- Face Recognition Algorithm: Select a robust and accurate face recognition algorithm or library that suits your needs. Popular choices include OpenCV, dlib, or face_recognition library. Ensure the algorithm has the capability to detect faces, extract facial features, and compare face encodings.
- Real-Time Video Processing: Implement a video processing module that can capture video frames in real-time from a camera or video stream. This module should interface with the face recognition algorithm to detect and recognize faces in each frame.
- Database Integration: Choose a suitable real-time database system for storing and retrieving face recognition data. Examples include Firebase Realtime Database, MongoDB, or MySQL with appropriate real-time synchronization mechanisms.
- Database Schema: Design the database schema to store relevant information associated with each recognized face, such as unique face IDs, corresponding user IDs or labels, timestamps, and

additional metadata. Ensure efficient indexing and querying capabilities.

- User Interface: Develop a user-friendly interface to display the video stream, recognized faces, and relevant information from the real-time database. This interface should provide options for managing database entries, updating attendance records, and performing administrative tasks.
- Authentication and Security: Implement appropriate security measures to protect sensitive data and ensure secure access to the system. This may include user authentication, encryption of data at rest and in transit, and access control mechanisms.
- Real-Time Updates: Establish a mechanism for real-time updates between the face recognition system and the database. This can be achieved through event-driven programming, database triggers, or real-time synchronization protocols to ensure instant updates of attendance records or changes in the database.

# 2. LITERATURE SURVEY

## 2.1 Existing System

There are several existing face recognition systems that incorporate real-time databases for efficient and accurate identification. These systems utilize facial recognition algorithms to analyze and compare facial features from images or video frames with a database of known faces. Here's a general overview of how such a system might work:

1. Face Detection: The system detects and locates faces in an image or video frame using techniques like Haar cascades, deep learning-based approaches (e.g., convolutional neural networks), or a combination of both.
2. Face Alignment: Once a face is detected, the system aligns the face to a standardized pose or orientation, ensuring consistent positioning for accurate comparison.
3. Feature Extraction: Key facial features are extracted from the aligned face, capturing unique characteristics such as the shape of the eyes, nose, and mouth. This process usually involves encoding the face into a compact numerical representation or feature vector.
4. Database Creation: A database is created to store the precomputed face embeddings (feature vectors) for a set of known individuals. Each person's entry in the database is associated with their identity information.
5. Real-Time Face Recognition: In real-time scenarios, the system captures an image or video frame containing a face. It performs face detection, alignment, and feature extraction on the captured face.
6. Face Matching: The extracted face features are compared against the face embeddings stored in the database using similarity metrics like Euclidean distance or cosine similarity. The system identifies potential matches based on the similarity scores.
7. Identity Verification/Recognition: The system determines the identity of the person by comparing the similarity scores with a predefined threshold. If the highest similarity score exceeds the threshold, it indicates a positive match, and the person's identity is retrieved from the database.
8. Real-Time Database Updates: The system allows for real-time updates to the face database. New faces can be added to the database along with their

corresponding identity information, while existing entries can be modified or removed as necessary.

Some popular frameworks and libraries for building face recognition systems with real-time database integration include OpenCV, Dlib, TensorFlow, and PyTorch. These frameworks provide various face detection and recognition algorithms along with tools for working with databases.

It's important to note that the implementation details may vary depending on the specific system requirements, scale, and constraints. Additionally, ethical considerations and privacy regulations should be taken into account when deploying face recognition systems with real-time database integration.

## 2.2 Proposed System

Real-time face recognition involves detecting and recognizing faces in a video stream in real-time. Here are the main steps involved in real-time face recognition:

1. Face detection: The first step is to detect faces in the video stream. This can be done using techniques like Haar cascades or deep learning-based face detectors like MTCNN, which can detect faces even in challenging lighting conditions and from different angles.

2. Face alignment: Once faces have been detected, they may need to be aligned to a standard pose to ensure that the recognition algorithm can work effectively. This involves adjusting the position, size, and orientation of the face so that it is centered and facing forward.

3. Feature extraction: After the face has been aligned, features are extracted from the face. This is typically done using deep learning-based techniques like face embeddings, which encode the unique features of a face into a compact vector representation.

4. Face recognition: The face embeddings are then compared to a database of known faces to identify the person in the video stream.

This is typically done using techniques like nearest neighbor search or clustering, which can efficiently find the closest matching faces in the database.

5. Update database: If a new face is detected in the video stream that is not already in the database, the user can be prompted to add the new face to the database for future recognition.

6. Real-time feedback: Finally, the results of the face recognition system can be displayed in real-time, such as overlaying the name or information of the recognized person on the video stream.

Real-time face recognition is a challenging task that requires fast and accurate algorithms. It is important to carefully select and optimize each step of the pipeline to ensure high performance and low latency.

## 2.3 Literature Review Summary **(Minimum 7 articles should refer)**

| Year and Citation | Article/ Author | Tools/ Software | Technique | Source | Evaluation Parameter |
|---|---|---|---|---|---|
| 2001 | Georgiades, A.S., Blumer, P.N. and Kriegman | Programming Languages, Libraries and Frameworks | Image formation process based on illumination cones | IEEE | - |
| 2002 | Viola, P. and Jones | Programming Languages, Libraries and Frameworks | Rapid object detection using a boosted cascade of simple features | IEEE | - |

| | | | | | |
|---|---|---|---|---|---|
| 2002 | Yang, M., Kriegman, D. and Ahuja | Programming Languages, Libraries and Frameworks Survey forms, gui | Detecting faces in images: A survey | IEEE | - |

.

| Year and Citation | Article/ Author | Tools/ Software | Technique | Source | Evaluation Parameter |
|---|---|---|---|---|---|
| 2008 | Turag, P., Chellappan, R., Subrahmanyan, V.S. and Udrea, O | Programming Languages, Libraries and Frameworks | Machine recognition of human faces | IEEE | - |
| 2000 | Phillips, P.J., Moon, H., Rizvi, S.A., Reuss, P.J. and Tyson, J.G. | Programming Languages, Libraries and Frameworks | The FERET evaluation methodology for face recognition algorithms | IEEE | - |
| 2000 | Moghaddam, B., Pentland, A. and Srihari, S.N. | Programming Languages, Libraries and Frameworks Survey forms, gui | A Bayesian framework for face recognition | IEEE | - |
| 2003 | Zhao, W., Chellappan, R., Phillips, P.J. and Rosenfeld | Programming Languages, Libraries and Frameworks Survey forms, gui | A literature survey. ACM Computing Surveys | IEEE | - |

# 3. PROBLEM FORMULATION

Problem Formulation: Face Recognition with Real-time Database Integration

1. Problem Statement: The problem is to develop a face recognition system that integrates real-time database functionality to enable efficient and dynamic face recognition with the ability to track attendance in real-time.
2. Face Recognition: The primary task is to accurately recognize and match faces in real-time video frames captured from a camera or video stream. The system should be able to detect and identify individuals based on their facial features, even in the presence of variations in lighting conditions, poses, and facial expressions.
3. Real-time Database Integration: The system should integrate a real-time database, such as Firebase Realtime Database, to store and retrieve face recognition data. This includes storing known face encodings, associated student IDs or labels, attendance records, and other relevant information.
4. Attendance Tracking: The system should track attendance by recording the time and date when a recognized face is detected. It should update the attendance records in the real-time database in real-time, maintaining accurate attendance counts for each individual.
5. User Interface: The system should provide a user-friendly interface to display the real-time video stream, recognized faces, and relevant information from the real-time database. It should allow administrators to manage database entries, update attendance records, and perform administrative tasks easily.
6. Security and Authentication: The system should incorporate appropriate security measures to ensure secure access to the system and protect

sensitive data. This may include user authentication mechanisms, encryption of data, and access control to prevent unauthorized access.

7. Scalability and Performance: The system should be designed to handle a large number of faces and provide efficient and accurate face recognition in real-time. It should be scalable to accommodate a growing number of individuals and perform face recognition tasks with minimal latency.
8. Reliability and Robustness: The system should be reliable and robust, capable of handling various scenarios and environmental conditions. It should be able to handle variations in lighting, pose, facial expressions, and occlusions while maintaining accurate and consistent face recognition results.
9. Integration with Existing Systems: If applicable, the system should be designed to integrate with other existing systems or databases used for student information management, such as student enrollment records or academic databases.
10. Performance Evaluation: The proposed system should be evaluated and benchmarked using appropriate performance metrics, such as recognition accuracy, processing speed, and database query response time, to ensure its effectiveness and efficiency in real-world scenarios.

By addressing these problem aspects, the proposed system aims to provide a robust and efficient face recognition solution with real-time database integration for applications such as attendance tracking, access control, and identity verification.

## 4. OBJECTIVES

Objectives of Face Recognition with Real-time Database Integration:

11. Accurate Face Recognition: The primary objective is to develop a face recognition system that can accurately identify and match faces in real-time video frames. The system should be able to handle variations in lighting, poses, facial expressions, and occlusions to provide reliable recognition results.
12. Real-time Database Integration: Integrate a real-time database, such as Firebase Realtime Database, to store and retrieve face recognition data in real-time. This includes storing known face encodings, associated student IDs or labels, attendance records, and other relevant information.
13. Real-time Attendance Tracking: Implement a real-time attendance tracking feature that records the time and date when a recognized face is detected. The system should update the attendance records in the real-time database instantly to enable accurate and up-to-date attendance management.
14. Efficient Database Management: Develop efficient database management mechanisms to handle a large number of face recognition entries, attendance records, and associated data. Ensure that the system can handle database queries, updates, and retrieval operations quickly and reliably.
15. User-friendly Interface: Create a user-friendly interface that allows administrators to view the real-time video stream, recognized faces, attendance records, and other relevant information. The interface should provide easy access to database management functionalities and facilitate smooth interaction with the system.
16. Security and Privacy: Implement appropriate security measures to protect the face recognition data and ensure user privacy. This may include encryption of data, user authentication mechanisms, access control, and compliance with data protection regulations.
17. Scalability and Performance: Design the system to be scalable, capable of handling a growing number of individuals and face recognition tasks without compromising performance. Ensure that the system can handle real-time face recognition with minimal latency and provide consistent results.
18. Integration with Existing Systems: If applicable, enable seamless integration with other existing systems or databases used for student information management, such as student enrollment records or academic databases. Ensure data consistency and synchronization between different systems.
19. Evaluation and Validation: Evaluate the performance and accuracy of the face recognition system with real-time database integration using appropriate metrics. Validate the system's effectiveness in real-world scenarios and compare it against existing benchmarks and standards.
20. Reliability and Robustness: Ensure that the system is reliable and robust, capable of handling various scenarios and environmental conditions. Test

the system under different lighting conditions, pose variations, and occlusions to ensure its effectiveness in practical settings.

By achieving these objectives, the face recognition system with real-time database integration aims to provide accurate, efficient, and secure face recognition capabilities for applications such as attendance management, access control, and identity verification.

# 5. METHODOLOGY

Real-time face detection involves real-time face detection and recognition in video streams. The basic steps involved in real-time face recognition are:

- Face recognition. The first step is recognizing faces in the video stream. This can be done using deep learning-based face detectors such as MTCNN, which can detect faces even under complex lighting conditions and from various angles, or using techniques such as Haar Cascade.
- Face Alignment: When a face is detected, it may need to align to a standard pose for the recognition algorithm to work effectively. This includes positioning, sizing and orienting the face so that it is centered and facing forward. 3. Feature extraction . When the faces are aligned, elements are extracted from the faces. This is usually done through deep learning based techniques such as face embedding, which encodes unique facial features into compact vector representations.
- Face recognition. The embedded faces are then compared to a database of known faces to identify people in the video stream. This is usually done through methods such as nearest neighbor search or clustering, which can efficiently find the closest matching face in a database.
- Database update: If a video stream encounters a new face that is not yet in the

database, the user may be prompted to add the new face to the database for later recognition.

Real-time feedback. Finally, the results of the face recognition system can be displayed in real time, such as overlaying names or information about detected faces on a video stream.

Real-time face recognition is a complex task that requires fast and accurate algorithms. Careful selection and optimization of each pipeline stage is critical to ensure high performance and low latency.

# 6.EXPERIMENTAL SETUP

The code you provided appears to be a Python script that uses OpenCV, face recognition, Firebase, and CVZone to implement a face attendance system. The script captures video from a webcam, detects faces in the frames, compares them with known face encodings, and performs attendance tracking based on recognized faces.

Here's a breakdown of the script's main components:

21. Imports: The required libraries/modules are imported, including **cv2** for OpenCV, **pickle** for object serialization, **os** for operating system-related functions, **cvzone** for computer vision utilities, **face_recognition** for face detection and recognition, **datetime** for working with dates and times, **numpy** for numerical operations, and **firebase_admin** for Firebase integration.
22. Firebase Initialization: The Firebase service is initialized using a service account key file and the database and storage URLs are specified.
23. Video Capture: The script opens a video capture object (**cap**) and sets the video resolution.
24. Background and Modes: The script loads a background image and a set of mode images from the specified paths. The background image is resized, and one of the mode images is placed at a specific region on the background image.
25. Encoding File: The script loads a precomputed encoding file (**EncodeFile.p**) containing known face encodings and associated student IDs.
26. Main Loop: The script enters a main loop that captures frames from the video and processes them.
27. Frame Processing: Inside the loop, the current frame is resized and converted to RGB format. Face locations and encodings are then computed using the **face_recognition** module.
28. Overlaying Background Image: The script resizes the current frame to match the background image size and overlays it on the background image.
29. Face Recognition: If any faces are detected in the frame, the script compares their encodings with the known face encodings. If a match is found, the student ID associated with the recognized face is retrieved. The script draws a rectangle around the recognized face on the background image.
30. Data Retrieval: Once a face is recognized, the script retrieves the student information and their corresponding image from Firebase storage. It also calculates the time elapsed since the last attendance and updates the attendance data if the elapsed time exceeds a threshold.

31. Displaying Information: The script displays the student's total attendance, name, and ID on the background image. The student's image is resized and centered on the background image.
32. Mode Transition and Reset: After a specific time period, the script transitions to a different mode and resets the counters and related variables.
33. Exiting the Loop: If no faces are detected in the frame, the script resets the mode and counters.
34. Displaying the Result: The final background image with overlays and text is displayed in a window named "Face Attendance."
35. Exiting the Script: The script waits for a key press and exits when a key is pressed.

Please note that in order to run this script successfully, you'll need to make sure you have all the required dependencies installed, such as OpenCV, face_recognition, cvzone, and firebase_admin. You'll also need to provide the necessary files, such as the service account key file, the encoding file, and the background image.

# 7.CONCLUSION

The project's purpose is to create and use of a face recognition system for keeping track of students' attendance at a school have proven to be a significant development. An automated system that makes use of facial recognition technology has successfully replaced the labor-intensive, error-prone, and time-consuming manual attendance tracking approach. During the pilot test, the system proved its capability to efficiently and reliably track student attendance, removing the chance of proxy attendance and enhancing the procedure in general.

The use of the facial recognition system has helped the educational facility in several ways. As a result, administrative tasks have become easier to manage, instructional time has been preserved, and attendance records' correctness and dependability have improved. The system has considerably increased the efficiency and efficacy of the attendance tracking process by utilizing the power of biometric technology, allowing instructors and staff to devote more time and resources to the system's main educational activities.

## Future Work

Even if the system for tracking attendance using facial recognition has been built and put into use with success, there are still ways to expand and improve it. Some potential future work areas include:

Integration with current student information systems: By integrating the face recognition system with the current student information systems used by the educational institution, seamless data transfer, thorough student records management, and a comprehensive picture of attendance and performance are all made possible.

System optimization that is ongoing can improve the precision and effectiveness of the face recognition system. The system can adapt better to changing environmental conditions and student appearances, increasing its reliability. This can be achieved by strengthening algorithms, modernizing hardware components, and utilizing machine learning approaches.

Scalability and wider implementation: Building on the positive results of the pilot test, the face recognition system needs to be assessed and given serious consideration for wider use in other educational settings, such as classrooms and institutions. Trials and assessments in various environments will yield insightful information and feedback for ongoing system adaptation and improvement.

Ethics and legal ramifications: It are important to continually discuss the ethical and legal ramifications of using biometric technologies in educational settings. To preserve the system's moral usage and safeguard the privacy of employees and students, compliance with data protection laws, upholding privacy rights, and routinely assessing rules and procedures are crucial.

The face recognition system can continue to develop into a reliable and widely used solution for attendance tracking in educational institutions by pursuing these future work areas. Its continuing development will increase effectiveness, precision, and general student participation, supporting the institution's dedication to excellence while also advancing educational procedures.

# 8. TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK

## CHAPTER 1: INTRODUCTION

## CHAPTER 2: LITERATURE REVIEW

## CHAPTER 3: OBJECTIVE

## CHAPTER 4: METHODOLOGIES

## CHAPTER 5: EXPERIMENTAL SETUP

## CHAPTER 6: CONCLUSION AND FUTURE SCOPE

# REFERENCES


[1]. Georgiades, A.S., Blumer, P.N. and Kriegman, D.J., 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(6), pp.643-660.

[2]. Viola, P. and Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (Vol. 1, pp. I-511). IEEE

[3]. Yang, M., Kriegman, D. and Ahuja, N., 2002. Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1), pp.34-58.

[4]. 4. Turag, P., Chellappan, R., Subrahmanyan, V.S. and Udrea, O., 2008. Machine recognition of human faces. Springer Science & Business Media.

[5]. Phillips, P.J., Moon, H., Rizvi, S.A., Reuss, P.J. and Tyson, J.G., 2000. The FERET evaluation methodology for face recognition algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(10), pp.1090-1104.

[6]. Moghaddam, B., Pentland, A. and Srihari, S.N., 2000. A Bayesian framework for face recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (Vol. 1, pp. 176-183). IEEE.

[7]. Zhao, W., Chellappan, R., Phillips, P.J. and Rosenfeld, A., 2003. Face recognition: A literature survey. ACM Computing Surveys (CSUR), 35(4), pp.399-458.

[8]. Wang, Y., Huang, T. and Wu, G., 2004. Face recognition using discriminant locality preserving projections. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (Vol. 2, pp. II-209). IEEE

[9]. Tan, X. and Trigs, B., 2010. Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE Transactions on Image Processing, 19(6), pp.1635-1650.

[10]. Hu, J., Lu, J., Tan, Y.P. and Zhou, J., 2014. Discriminative deep metric learning for