Face Recognition System Using OpenCV and C++

Submitted by: Parastoo Masoodi & Naim Rahman Ifti

# Abstract

This paper presents the design and implementation of a Face Recognition System developed using C++ and OpenCV libraries. The system is capable of detecting and recognizing faces in real-time using classical machine learning techniques. Haar Cascade Classifiers were used for face detection, while the Local Binary Patterns Histogram (LBPH) algorithm was employed for face recognition. The project showcases how lightweight and efficient face recognition systems can be built using classical methods without relying on heavy deep learning architectures. The system can be applied in areas such as security surveillance, user authentication, and attendance monitoring.

# 1. Introduction

Face recognition is a biometric technique that identifies or verifies individuals by analyzing and comparing facial features. It is widely used in applications ranging from smartphone authentication to criminal identification. The goal of this project was to design a real-time face recognition system using C++ and OpenCV, emphasizing both accuracy and computational efficiency. This project focused on classical methods like Haar Cascades and LBPH due to their speed and simplicity, making them ideal for resource-limited environments.

# 2. Background and Related Work

Face detection and recognition have been extensively studied for decades. Early techniques such as Eigenfaces (PCA) and Fisherfaces (LDA) provided the foundation for modern recognition systems. Haar Cascade Classifiers introduced by Viola and Jones revolutionized real-time object detection. Similarly, LBPH provided a fast and reliable method for recognizing faces based on texture features. OpenCV, an open-source computer vision library, offers built-in implementations of these classical algorithms, making it a popular choice for academic and industrial projects.

# 3. System Design

The face recognition system was divided into two main stages:  
  
- Face Detection: Utilized Haar Cascade Classifier (haarcascade\_frontalface\_default.xml) to detect faces from images and video frames. Cropped detected faces and resized them for consistency.  
- Face Recognition: Employed the LBPH algorithm to recognize faces from previously trained images. Provided identity prediction along with a confidence score.

# 4. Implementation Details

## 4.1 Dataset Collection

Multiple face images were collected per user using a webcam. Each image was converted to grayscale, cropped to the face region, and saved inside a structured dataset directory (dataset/) with labels corresponding to each user.

## 4.2 Model Training

The LBPH face recognizer was trained on the collected dataset:  
  
Ptr<LBPHFaceRecognizer> model = LBPHFaceRecognizer::create();  
model->train(images, labels);  
model->save("trainer.yml");  
  
Images were read, converted to grayscale, and associated with their respective user IDs. After training, the model was saved as trainer.yml for future predictions.

## 4.3 Real-Time Recognition

In real-time recognition:  
  
- The webcam captured live frames.  
- Haar Cascade detected faces in each frame.  
- The trained LBPH model predicted the user ID and confidence.  
- Detected faces were labeled with user names or IDs, and the confidence score was displayed.

# 5. Applications

Face recognition systems have widespread applications:  
- Surveillance and Security  
- Mobile Phone Unlocking  
- Attendance Systems  
- Retail and Marketing

# 6. Challenges and Ethical Issues

Despite its advantages, face recognition systems face challenges:  
- Environmental Factors: Variations in lighting, poses, and expressions can reduce accuracy.  
- Privacy Concerns: Unauthorized face tracking can raise significant ethical and legal issues.  
- Bias and Fairness: Systems trained on biased datasets may perform poorly across different demographics.  
- False Matches: Errors in recognition can lead to wrongful identification.

# 7. Conclusion

The Face Recognition System developed using C++ and OpenCV demonstrates that classical methods like Haar Cascades and LBPH are still highly effective for real-time applications. Although deep learning-based approaches have gained popularity, lightweight systems like the one implemented here offer a practical solution when computational resources are limited. Future work may explore integrating deep learning models to improve accuracy further or expanding the dataset to include more varied face samples.

# References

* Emami, S., & Suciu, V. (2012). Facial recognition using OpenCV. Journal of Mobile, Embedded and Distributed Systems, 4(1), 33–38.
* Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1), 71–86.
* Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 711–720.
* Ahonen, T., Hadid, A., & Pietikäinen, M. (2006). Face description with local binary patterns: Application to face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12), 2037–2041.
* Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media.