Implementing Facial Recognition Authentication in Physical Access Control System

Job Adegede (Student), Dr. Christopher Ivancic (Supervisor/PI)
Stephen F. Austin State University, Texas

Abstract

With the rise in data security threats, authentication systems have adopted innovative ways of identifying users like facial recognition. Biometric credentials unlike other forms, relies on the individual's distinctive biological features like fingerprints, iris, face, voice, etc., to verify identity. Facial recognition authentication is still considered a relatively new application with more improvements still anticipated even as it has seen rapid adoption in the last decade. Current developments are mostly based on machine learning algorithms like Opencv, Keras, Tensorflow, Pytorch etc. This effort is focused on the development of facial recognition as the authentication module for a physical access control system. It uses the Opencv pre-trained model and Python's face_recognition library for human face detection and recognition. In a bid to achieve higher detection and recogniton rates, dual Haar cascades - face and eyes, were used in combination with the Histogram of Oriented Gradients transform. Another major point of improvement over existing efforts is the adoption of memory-resident 128-dimensional embeddings vector dictionary structure for stored enrolled user data. This surpasses byte stream storage formats like Python pickle, BSON, JSON etc., by removing security risks associated with serialized/de-serialized structures. The resulting accuracy of the facial authentication subsystem, has produced no false positive identification after over 300 transactions, complemented with responsive electronic lock actuation in response to authentication passes or fails

Introduction

Facial recognition has been increasingly embraced as acceptable authentication medium for both physical and electronic access to resources and assets. The objective of this work is to build a physical access control system that use facial recognition as authentication module. Among the various package options analyzed for use as core engine for the system, opencv was chosen because of its unique advantages like real-time vision support, speed, large user/support community, tested feature-set, and considerable documentation accumulated over the years.
This work aimed for improvement over reviewed existing efforts in 2 ways:
- Employing multiple cascade classifiers
- Remove risk factor associated with byte stream storage formats for enrolled user data.
A review of existing works quickly revealed a direct relationship between false positive outcomes and the number of cascade classifiers applied in the detection process. Applying more classifiers created less opportunity for false positive results as the added eyes dimension created for a more rigorous identification process. The opencv implementation of the Haar Cascades algorithm by Paul Viola and Michael Jones, developed in 2001, was used.
Another improvement opportunity was in the area of object persistence in code. Most facial recognition developments use byte stream-based formats for storing both primitive or custom types, for future use in code. These formats have been frequently associated with insecurities which would be unsuitable for an authentication system. It was in this vein that this implementation opted for the pre-trained Machine Learning (ML) option provided by the face_recognition.face_encodings() function. This improved security by eliminating the need to save encoded data to disk, but rather persisted enrolled user database in code as a 128-dimensional embeddings vector running in memory throughout code execution.

Solution Description

The facial recognition system was designed as the authentication module of a physical access control system. Various components of this implementation includes:

Table 1.: Materials

Hardware	Purpose
i. Raspberry Pi-4B device	System core device – software & hardware I/O module
i. Electronic lock mechanism	Demonstrate physical access control response
i. Raspberry pi module-2 camera	Capture user images and videostream
Software	**Purpose**
i. Opencv library	Object detection, recognition, classification and tracking
i. Face_recognition.py	API for face detection and recognition
i. Haar Cascade classifiers	Human face and eyes detection and classification
i. Imutils library	Image processing: resizing, translation, rotation, etc.
i. Imutils.video	Video streaming
i. Numpy	Numerical operations
i. Tqdm	Progress bar functionality
i. Time	Time-related tasks

Methodology

- ❑ Live image capture and face object detection for authorized user enrollment using VideoStream(usePiCamera=True).start() function and human face and eyes cascade classifiers
- ❑ Eye object classification within face region of interest using eyeCascade.detectMultiScale(roi_face) before storage to dataset repository
- ❑ Saved image processing using Histogram of Oriented Gradients (HOG) and unique feature descriptor generation as a 128-dimensional embedding vector using face_recognition.face_encodings()
- ❑ Persistent video monitoring module on-the-ready for capturing real-time images from new users
- ❑ Comparison of new user image encoding with encoded 128-dimensional embedding vector of enrolled users using face_recognition.compares_faces()
- ❑ Access grant or deny response demonstrated by lock release or arming
- ❑ Screen display of active identification progress bar of verified user's motion in camera focus

Results

The outcome of this work is a concept model of a facial recognition physical access control system running on a Raspberry pi-4 unit and connected to a camera and electronic lock.
Authentication tests so far conducted has not recorded a false positive result after processing over 300 images and counting. More tests are expected carried out in the near future

Discussion

It's noteworthy to mention that this project was initially designed to deliver a Multi-Factor authentication system with voice recognition as primary credential, and facial recognition as secondary. However, upon closer assessments and requirements testing, it came to light that 2 challenges stood in the way which were:
❑**Insufficient computing power**
Voice or speaker recognition systems are currently only available as pre-trained neural network systems which must be further trained using GPUs. The hardware requested by the researcher unfortunately fell short of these requirement and the inadequacy was not revealed at the outset
❑**Considerable time requirement**
To train a voice model required between 300 to 1500 recorded voice samples covering various speech patterns, tones and expressions for each user meant for enrollment
These were the 2 considerations that refocused the project scope towards implementing facial recognition as current hardware and available time were deemed adequate for it.

Conclusions

Existing efforts in this study context. Adopting multiple Haar cascade classifiers improved detection and encoding accuracy.
Security of the system was also significantly improved by employing the ML model available in the **face_recognition.face_encodings()** function, which generated the 128-dimensional embeddings vector data structure persistently resident in memory without the need for storage to disk.
The project has been challenging but it has also opened opportunities to new interesting areas in the computer vision domain for further study

Future Work

From the initial design where the system was intended to perform authentication on voice or speaker recognition as primary credential, it's still desired to have the voice module integrated in the future in order to realize the original objective of this study. With the constraint of time more accurately understood now, a more informed project plan can be made which will improve the chances of success

Contact

Dr. Christopher Ivancic
Stephen F Austin State University,
1936 North street, Nacogdoches 75961
ivanciccp@sfasu.edu

References

1. https://face-recognition.readthedocs.io/en/latest/face_recognition.html
2. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
3. https://github.com/carolinedunn/facial_recognition?tab=readme-ov-file
4. https://ieeexplore.ieee.org/document/990517
5. https://snyk.io/blog/guide-to-python-pickle/
6. https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
7. https://picamera.readthedocs.io/en/release-1.13/api_array.html
8. https://core-electronics.com.au/guides/solenoid-control-with-raspberry-pi-relay/#:~:text=Connect%20the%20Ground%20Pin%20of,either%20to%20open%20or%20close
9. https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/1
10. https://www.geeksforgeeks.org/python-smile-detection-using-opencv/?ref=lbp