# Real-time Facial Recognition System Development Report

Date：2023-08-16

Name：Dai Jie Liu

Student Number:20087611

# Table of Contents

**1.Introduction, Architecture & Technologies**

**2.Functional & Non-Functional Requirements**

**3.Implementation**

\* Design

\* Implementation

\* Key Coding Elements

**4.Conclusion**

\*Attempted Modification

**5.References**

## 1. Introduction, Architecture & Technologies

The digital age has ushered in numerous advancements that redefine user identification and security measures. Among these innovations, real-time facial recognition stands as a notable paradigm shift, serving diverse applications from surveillance security to user verification. This document delves into the development intricacies of a real-time facial recognition system constructed on Flask and Flask-SocketIO.

**Technologies:**

**Flask:** An acclaimed Python-based micro web framework. It equips developers with the tools, libraries, and technologies required to craft an interactive web application.

**Flask-SocketIO:** An instrumental extension for Flask that simplifies the implementation of WebSocket, providing a channel for seamless bi-directional communication between server and client.

**Face\_recognition:** An adept Python library, it leans on dlib to offer functionalities to identify and manipulate faces, either through Python or command line.

**OpenCV:** Predominantly aimed at real-time computer vision, OpenCV serves as a comprehensive library for programming functions related to image and video manipulation.

## 2. Functional & Non-Functional Requirements

**Functional:**

1.Stream real-time video from an in-built or external camera.

2.Instantaneously detect and recognize faces in the video stream.

3.For recognized faces, send identification data to the client using WebSocket.

**Non-Functional:**

**1.Performance:** The application must promise brisk response times. The interval from detecting to recognizing a face must be under 2 seconds.

**2.Scalability:** With an ever-increasing database of recognized faces, the system should support easy and quick additions.

**3.Reliability:** The foundation of the system must be robust, ensuring continuous operation sans unexpected crashes or excessive lags.

## 3. Implementation

**Modular Structure:**

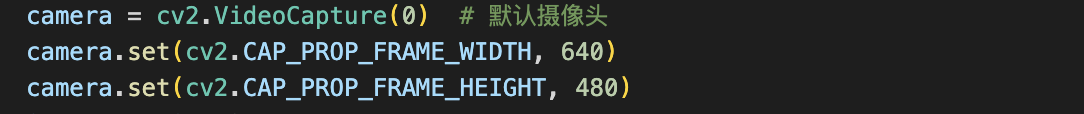
The entire system is architected with a modular approach, ensuring that each functional component (such as video capture, facial recognition, and result transmission) operates independently of one another. This design promotes flexibility, facilitating easier extensions or modifications to specific portions in the future.

**Multi-threading Approach:**

Given the computationally intensive nature of facial recognition, especially in real-time streams, a multi-threading strategy has been employed. This ensures seamless continuity in video streaming and a prompt system response, mitigating any potential delays or lags that could be introduced by the recognition process. By offloading the facial recognition to a separate thread, the application can handle video capture and processing concurrently, optimizing the utilization of computational resources and ensuring smooth user experience.

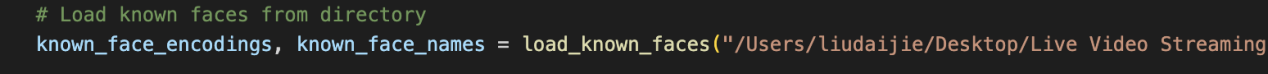
**Implementation:**

**1. Camera Integration and Initialization:**  
Utilizing the VideoCapture function from OpenCV, the system connects to the default camera and configures its resolution. This establishes an appropriate input source for subsequent image processing tasks.



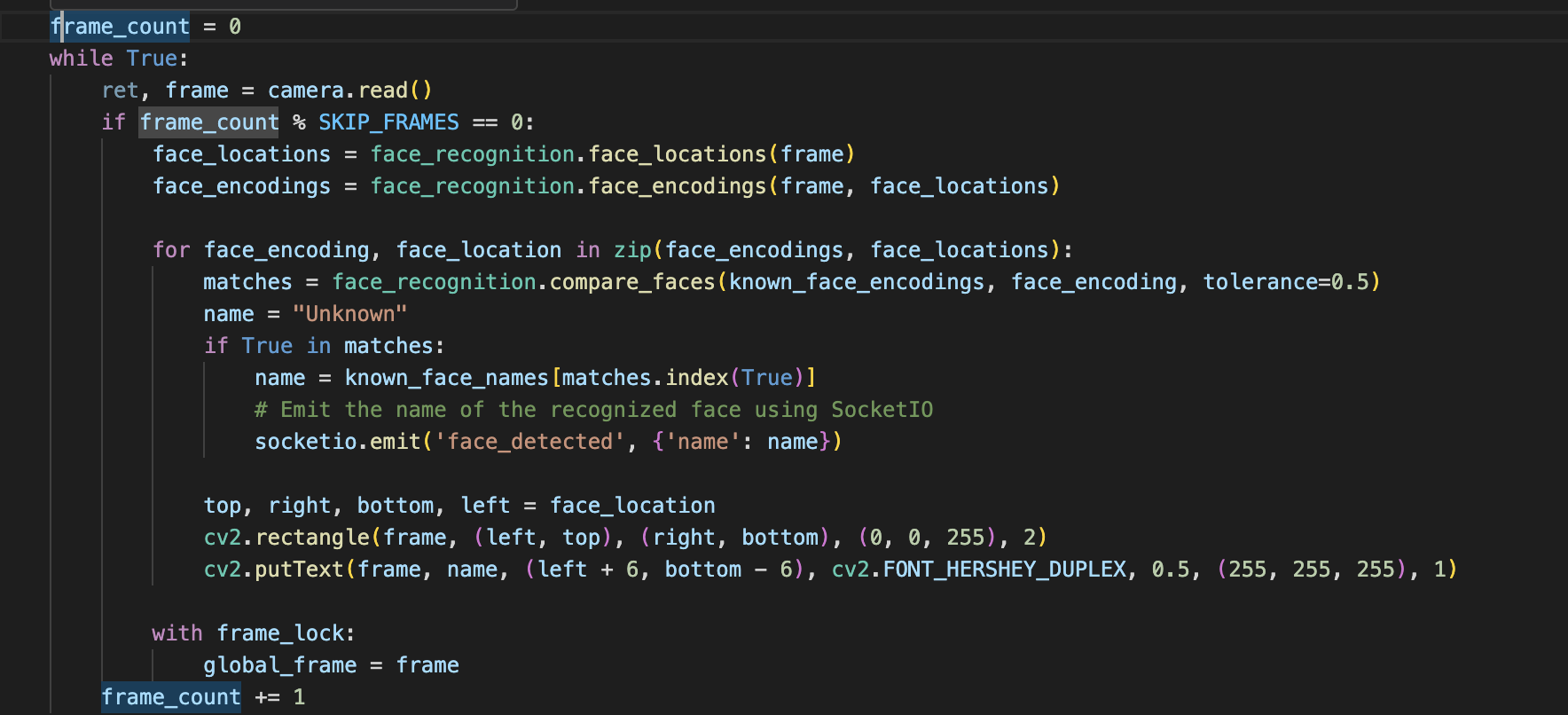
**2. Load known faces:**

The system will iterate through all image files in the specified directory, then use the face\_recognition library to load and encode these images. This provides a reference for the subsequent face comparison process.



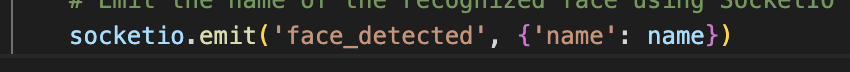
**3. Real-time facial recognition:**

The system reads each frame and performs face recognition every SKIP\_FRAMES frames. This trick reduces the computing power required while ensuring sufficient speed in live streaming.



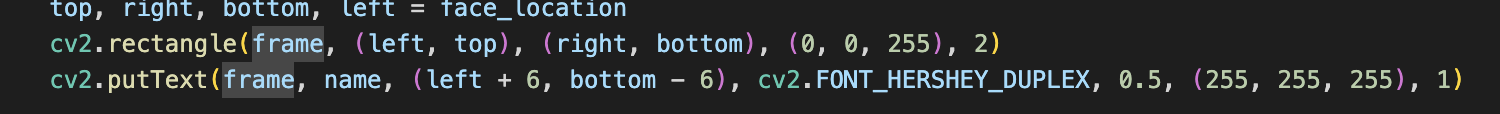
**4. WebSocket communication:**

Using Flask-SocketIO, once a face is recognized, the system sends the name of that face to connected clients in real time. Such real-time feedback is valuable for many applications, such as real-time visitor notification or security monitoring.

****

**5. Face rendering and annotation:**

Whenever a face is recognized in the video stream, a rectangle is drawn around that face with the name displayed below it, whether it's known or not.



These key technical points together constitute an efficient real-time facial recognition system. The design and implementation details of the system ensure its high performance and reliability in practical applications.

**Key Coding Elements:**

The cornerstone of the face recognition process depends on the face\_recognition.face\_encodings method. This function encodes the detected face in the frame and then matches it with the pre-stored encoding using face\_recognition.compare\_faces.

## 4. Conclusions and attempted revisions

The real-time facial recognition system we developed is based on Flask and Flask-Socketio. It is an effective tool for video streaming, real-time facial detection and subsequent recognition. Its lightweight and comprehensive structure makes it a feasible solution for various application scenarios.

During the development stage, a noted modification attempt involves the introduction of SKIP\_FRAMES variables. By skipping certain frames instead of continuously processing each frame, it greatly reduces computing needs and improves the fluidity and performance of the system.

**5. Summary**

This code shows how to combine Flask, Flask-SocketIO, and face\_recognition to create a real-time facial recognition system. The system captures a video stream from a camera and recognizes and notifies clients of known faces in real time.

Flask, 2021. Flask documentation. [online] Available at: [https://flask.palletsprojects.com/en/2.0.x/](https://flask.palletsprojects.com/en/2.0.x/" \t "/Users/liudaijie/Documents\\x/_new) [10/08/2023].

Flask-SocketIO, 2021. Flask-SocketIO documentation. [online] Available at: [https://flask-socketio.readthedocs.io/en/latest/](https://flask-socketio.readthedocs.io/en/latest/" \t "/Users/liudaijie/Documents\\x/_new) [10/08/2023].

Ageitgey, A., 2018. face\_recognition GitHub repository. [online] GitHub. Available at: [https://github.com/ageitgey/face\_recognition](https://github.com/ageitgey/face_recognition" \t "/Users/liudaijie/Documents\\x/_new) [10/08/2023]

Bradski, G., 2000. OpenCV documentation. [online] Available at: [https://docs.opencv.org/4.x/](https://docs.opencv.org/4.x/" \t "/Users/liudaijie/Documents\\x/_new) [10/08/2023].