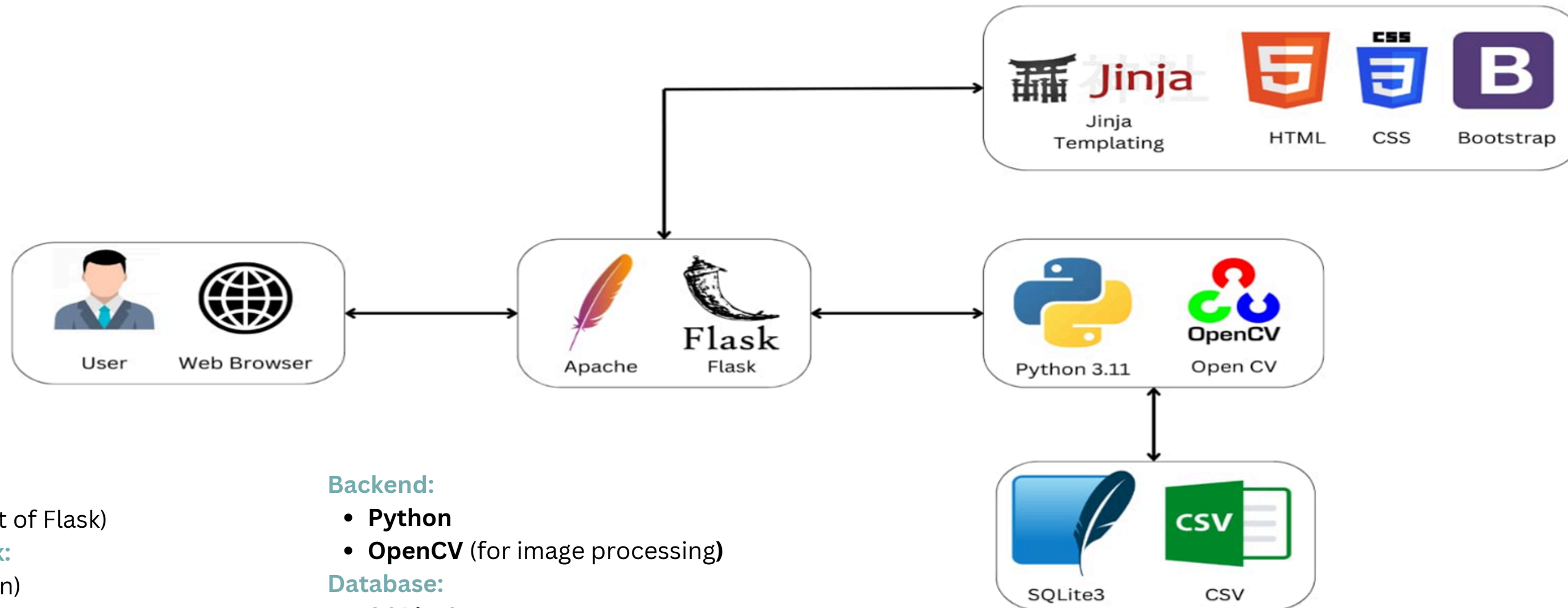


VIDEO-SURVILLANCE-AND-TRACKING-SYSTEM USING FACIAL RECOGNITION

ABSTRACT & INTRODUCTION

- CCTV cameras are widely used for monitoring public and private areas, ensuring constant surveillance.
- When someone goes missing, CCTV footage becomes essential for search teams.
- It provides clear evidence of where the person was last seen, helping to map out their movements and potential routes.
- However, manually going through this footage can be slow and tiresome.
- That's where our project, "The Third Eye," comes in. We're dedicated to simplifying this process, making it quicker and real-time.

SYSTEM ARCHITECTURE



Server:

- **Apache** (Part of Flask)

Web Framework:

- **Flask** (Python)

Frontend:

- **Jinja** (part of Flask)
- **HTML**
- **CSS**
- **Bootstrap 5** (Frontend Framework)

Backend:

- **Python**
- **OpenCV** (for image processing)

Database:

- **SQLite3**
- **CSV files**

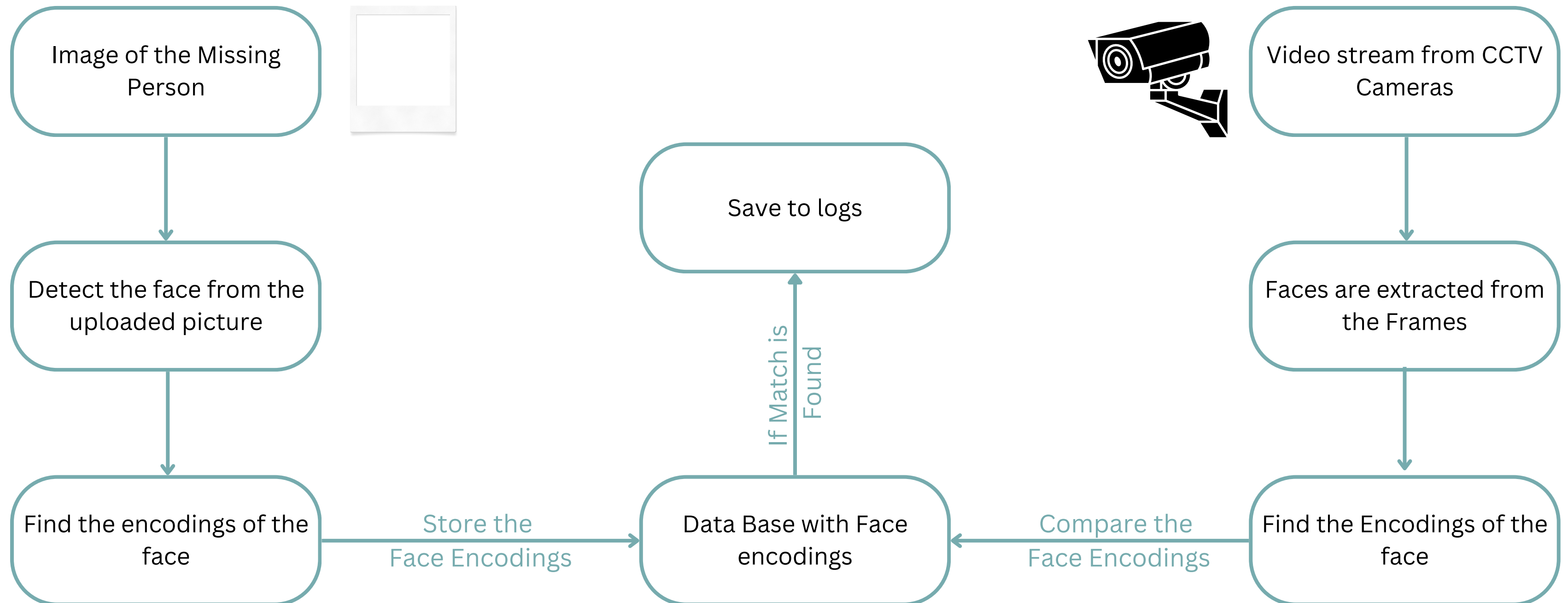
Software Requirements

- Python 3.11
- Flask Web Framework
- Web Browser

Hardware Requirements

- Device capable of accessing a web browser (e.g., Laptop, Phone)
- CCTV Cameras
- Server with a web server such as Gunicorn, uWSGI, or Apache with mod_wsgi

Work Flow

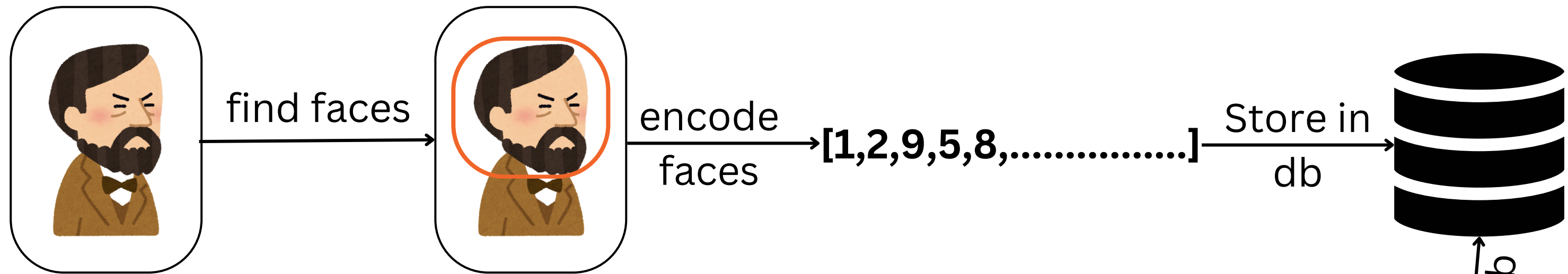


Workflow

- Obtain the CCTV footage, comprising a series of frames, with each frame representing an image captured at a defined rate per second.
- Analyze each frame individually to identify any faces present within it.
- Utilize face detection algorithms to accurately locate and extract facial features from the detected faces.
- Compare the extracted facial features of each detected face against the stored facial data of the missing person within the database.

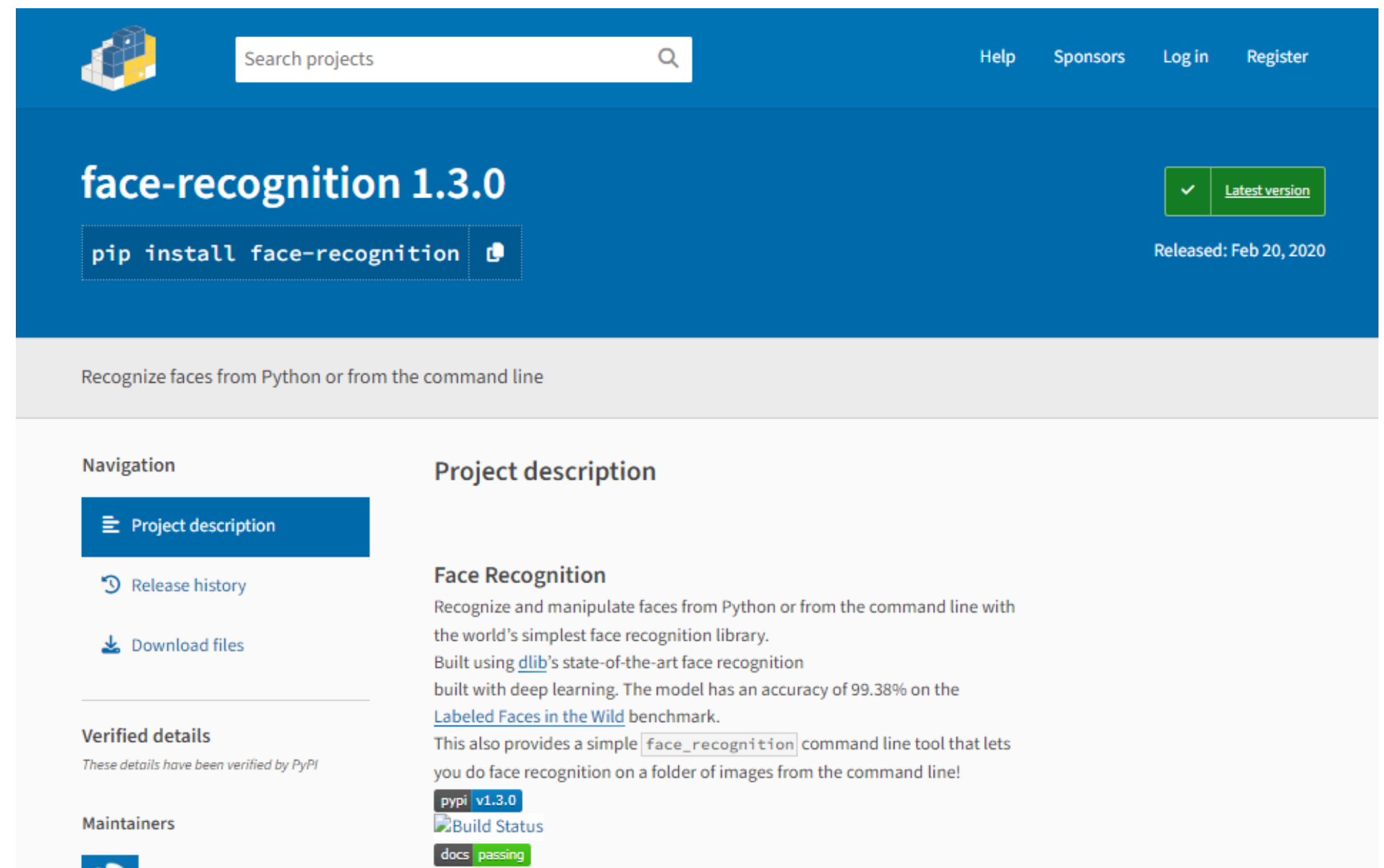
Workflow

- Implement facial recognition techniques to determine the likelihood of a match between the detected faces and the facial data of the missing person.
- Repeat the process for each frame in the CCTV footage to comprehensively search for the missing person across all captured images.
- If the match is found, the timing and location is logged in the database.



face_recognition Library

- The process of extracting Face Encodings relies on the face_recognition library.
- It is constructed using dlib's cutting-edge face recognition technology.
- This model is constructed with deep learning techniques and boasts an impressive accuracy rate of 99.38% on the Labeled Faces in the Wild benchmark dataset.



The screenshot shows the PyPI project page for 'face_recognition' version 1.3.0. The page has a blue header with a search bar and navigation links (Help, Sponsors, Log in, Register). The main content area is white with a blue sidebar on the left. The sidebar contains a 'Navigation' section with links to 'Project description' (highlighted), 'Release history', and 'Download files'. Below this is a 'Verified details' section with the text 'These details have been verified by PyPI'. The main content area features a 'Project description' section with the title 'Face Recognition' and a paragraph describing the library's capabilities and accuracy. At the bottom of the main content area, there are badges for 'pypi v1.3.0', 'Build Status', and 'docs passing'.

Search projects

Help Sponsors Log in Register

face_recognition 1.3.0

✓ Latest version

Released: Feb 20, 2020

`pip install face_recognition`

Recognize faces from Python or from the command line

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Maintainers

Project description

Face Recognition

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using [dlib](#)'s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the [Labeled Faces in the Wild](#) benchmark. This also provides a simple `face_recognition` command line tool that lets you do face recognition on a folder of images from the command line!

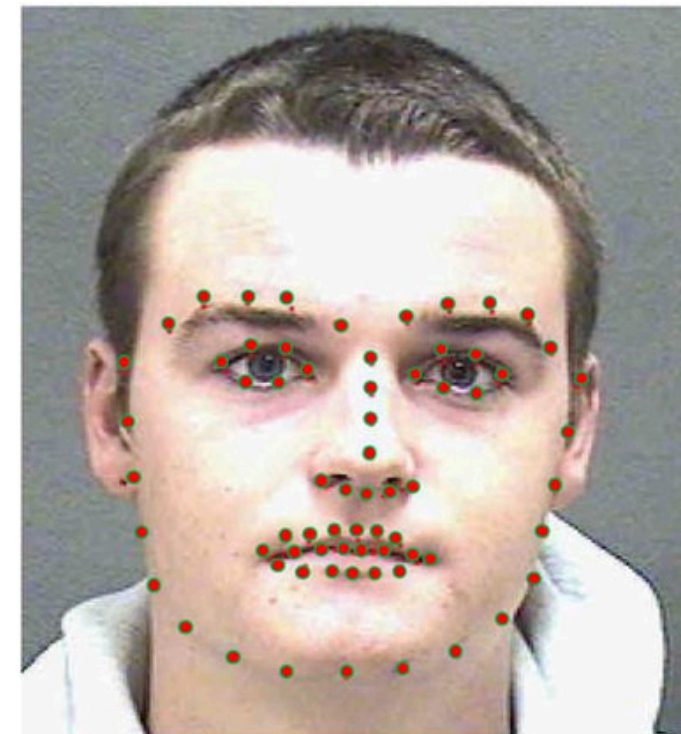
pypi v1.3.0

Build Status

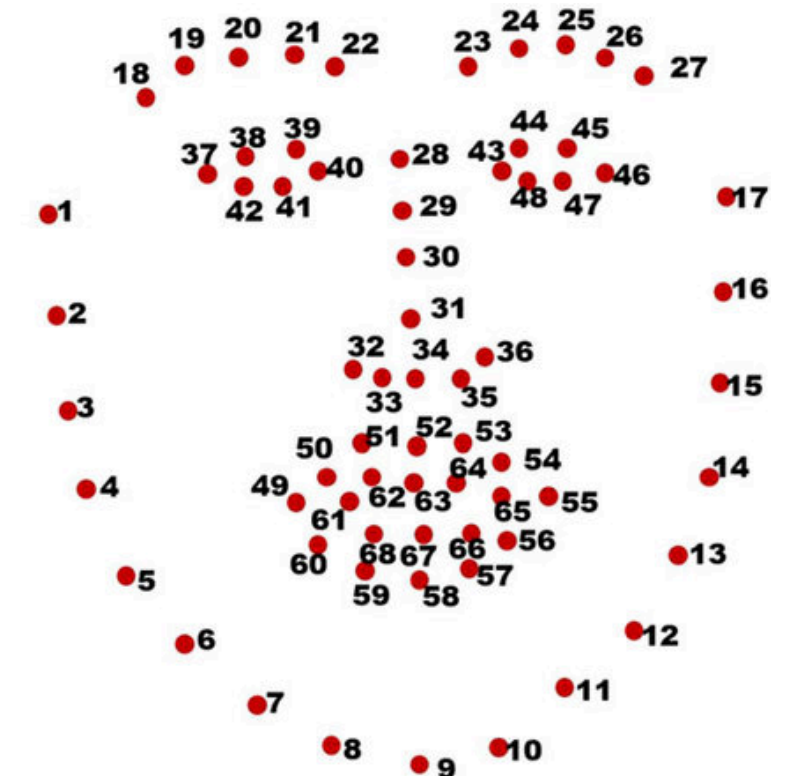
docs passing

Dlib

- Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.



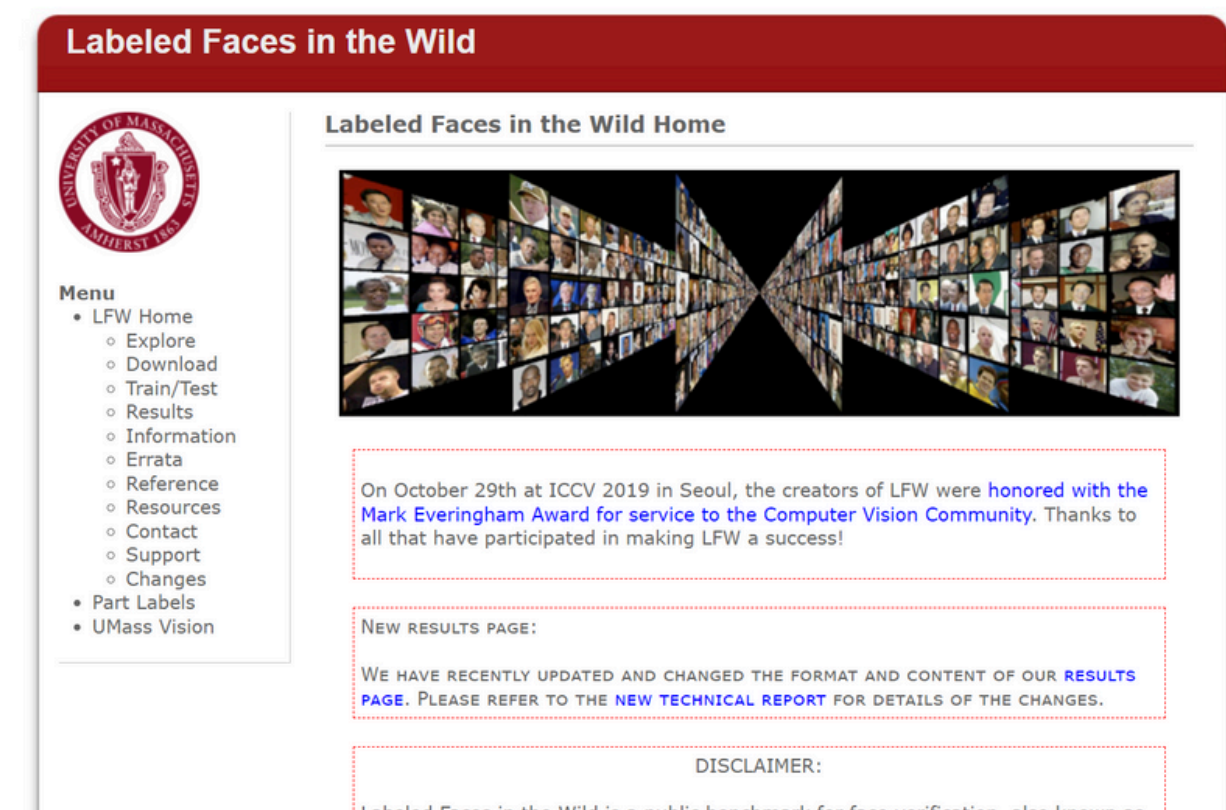
(a)



(b)

Labeled Faces in the Wild

- Labeled Faces in the Wild is created and maintained by University of Massachusetts.
- Labeled Faces in the Wild is a public benchmark for face verification, also known as pair matching. No matter what the performance of an algorithm on LFW, it should not be used to conclude that an algorithm is suitable for any commercial purpose.



- When a missing persons image is added, the face_recognition library is used to encode the image and store it in database
- The Application generates frames from the camera using openv, and face recognition has been integrated.
- The face_recognition library is utilized to perform face detection and recognition.
- For each frame captured, faces are detected using the face_recognition function and encoded using the face_recognition function.

- The encoded faces are compared with the known faces stored in the database to identify any matches
- Once a match is found, a rectangle is drawn around the recognized face, and the database is updated accordingly, marking the person as found.

Scope for further improvements

- Utilize Python's multithreading concept to create separate threads for each camera, enhancing efficiency and reducing memory consumption.
- Implement validation during the upload of a missing person's picture to ensure that the image contains only a single person.
- Verify the picture to confirm whether it contains a single individual before proceeding with the upload process.

Thank You