**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**NGÀNH CÔNG NGHỆ KỸ THUẬT MÁY TÍNH**

****

**ĐỒ ÁN 1**

**PHÂN TÍCH VÀ XÂY DỰNG THIẾT BỊ KIỂM SOÁT RA VÀO DỰA TRÊN NHẬN DIỆN KHUÔN MẶT**

**Sinh viên thực hiện: Nguyễn Hữu Tài**

**MSSV: 21119375**

**Giảng viên hướng dẫn: TS. Phạm Văn Khoa**

**Thành phố Hồ Chí Minh – Tháng 7/2024**

**MỤC LỤC**

PHẦN MỞ ĐẦU 2

1. Lý do chọn đề tài 2

2. Mục tiêu và đối tượng nghiên cứu 2

3. Công cụ và phần mềm hỗ trợ 2

4. Kết cấu của báo cáo 3

PHẦN 1: PHÂN TÍCH THƯ VIỆN FACE\_RECOGNITION TRONG PYTHON 4

1.1. Phát hiện khuôn mặt – Face Detector 4

*1.1.1. Phân tích mô hình thuật toán HOG – Histogram of Oriented Gradient* 5

*1.1.2. Đánh giá mô hình thuật toán HOG* 6

*1.1.3. Phương pháp sử dụng cho việc phát hiện khuôn mặt* 7

1.2. Nhận diện các đặc điểm trên khuôn mặt 9

1.3. Mã hóa khuôn mặt 10

1.4. So sánh khuôn mặt 12

PHẦN 2: XÂY DỰNG MÔ HÌNH NHẬN DIỆN KHUÔN MẶT 13

PHẦN 3: ĐÁNH GIÁ MÔ HÌNH 16

3.1. Đánh giá về tốc độ 16

3.2. Đánh giá về hiệu năng 19

PHẦN 4: TRIỂN KHAI HỆ THỐNG TỪ MÔ HÌNH 21

4.1. Sơ đồ khối 21

4.2. Lưu đồ xây dựng Face Recognition Device 22

4.3. Thiết kế giao diện 23

4.4. Hoàn thành sản phẩm 24

KẾT LUẬN 26

TÀI LIỆU THAM KHẢO 27

**PHẦN MỞ ĐẦU**

1. **Lý do chọn đề tài**

Đề tài về hệ thống kiểm soát ra vào dựa trên nhận diện khuôn mặt và là một ứng dụng thực tế được lựa chọn với nhiều lý do. Đầu tiên, nó mang lại giá trị thực tiễn trong việc cải thiện an ninh và quản lý ra vào trong nhiều lĩnh vực, từ cơ quan, công ty đến trường học. Thứ hai, hệ thống này cung cấp sự thuận tiện và hiệu quả cho việc kiểm soát ra vào mà không cần sử dụng các phương tiện truyền thống như thẻ từ, mã PIN hoặc vân tay. Cuối cùng, đề tài này cũng mang lại cơ hội phát triển kỹ năng lập trình, xử lý hình ảnh và sử dụng AI cho người tham gia, từ đó nâng cao khả năng cạnh tranh và hiểu biết trong lĩnh vực công nghệ thông tin. Đây là lý do chính tôi chọn dự án này cho Đồ án 1.

1. **Mục tiêu và đối tượng nghiên cứu**

Mục tiêu**:** Xây dựng một hệ thống nhận diện khuôn mặt chính xác và đáng tin cậy bằng việc kết hợp giữa phần cứng, phần mềm, xử lý dữ liệu, trí tuệ nhân tạo.

Đối tượng nghiên cứu**:** Các sinh viên và kĩ sư quan tâm đến các công nghệ an ninh và nhận diện khuôn mặt, những người muốn áp dụng và phát triển kỹ năng lập trình, xử lý hình ảnh và deep learning trong một dự án thực tế. Bên cạnh đó, các tổ chức hay doanh nghiệp cũng sẽ quan tâm để cải thiện quản lý ra vào và tăng cường bảo mật cho khu vực của họ

1. **Công cụ và phần mềm hỗ trợ**

- Thư viện face\_recognition và OpenCV – Sử dụng thuật toán nhận diện khuôn mặt.

**-** Ngôn ngữ lập trình Python 3.xx – IDE (Visual Studio Code, Pycharm, …)

- Phần mềm thiết kế giao diện: GIMP.

1. **Kết cấu của báo cáo**

Bài báo cáo được chia thành 4 phần chính:

Phần 1: Phân tích thư viện face\_recognition trong Python.

Phần 2: Xây dựng mô hình nhận diện khuôn mặt.

Phần 3: Đánh giá mô hình.

Phần 4: Triển khai mô hình vào hệ thống.

**Phần 1**

**PHÂN TÍCH THƯ VIỆN FACE\_RECOGNITION**

**TRONG PYTHON**

**Giới thiệu:**

Thư viện face\_recognition là một thư viện mã nguồn mở với các mô hình AI đã được huấn luyện cho phép người lập trình có thể sử dụng để nhận diện và xác thực khuôn mặt.

Thư viện bao gồm 4 tính năng chính như sau:

1. Phát hiện khuôn mặt.
2. Nhận diện các đặc điểm trên khuôn mặt.
3. Mã hóa khuôn mặt.
4. So sánh khuôn mặt.
   1. **Phát hiện khuôn mặt – Face Detector**

Chức năng: Phát hiện và trả về vị trí của các khuôn mặt mà mô hình phát hiện được trong một bức ảnh đầu vào.

Tên hàm sử dụng: **face\_detector (Image)**

(với Image là một ảnh đầu vào có định dạng RGB)

Các mô hình thuật toán được sử dụng trong hàm:

* HOG – Histogram of Oriented Gradient.
* SVM – Support Vector Machine.

Người dùng có thể lựa chọn một trong hai mô hình trên để sử dụng cho việc phát hiện khuôn mặt, tuy nhiên ở mỗi mô hình đều có những ưu và nhược điểm riêng. Với HOG việc tính toán các đạo hàm dựa trên CPU nên có thể hoạt động mạnh mẽ trên các thiết bị cấu hình thấp, tuy nhiên SVM thì lại sử dụng trên GPU nên chỉ mạnh khi chạy trên các thiết bị cấu hình cao. Mục tiêu hướng đến của hệ thống là xây dựng một thiết bị nhỏ gọn và đạt hiệu suất nhận diện cao. Vì vậy mô hình HOG sẽ được lựa chọn sử dụng.

* + 1. **Phân tích mô hình thuật toán HOG – Histogram of Oriented Gradient**

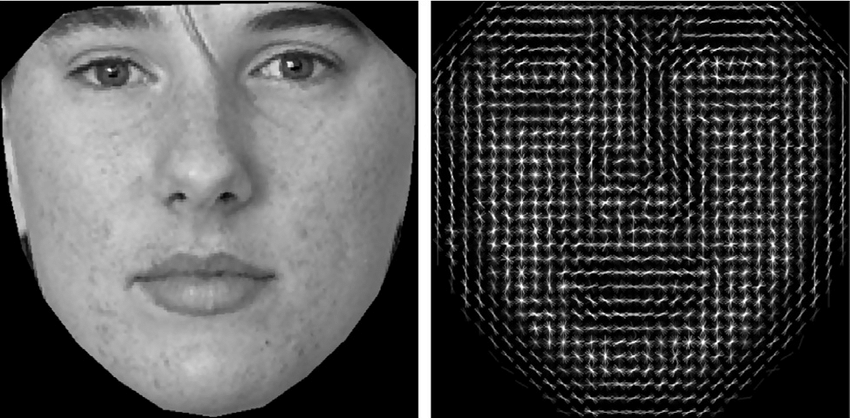
Cách hoạt động:

Thuật toán thu thập các giá trị pixel từ bức ảnh đầu vào và tính toán các đạo hàm theo hướng X và hướng Y tương ứng.

Các đạo hàm theo hướng ngang (X) và dọc (Y) được tính như sau:

Trong đó, là giá trị pixel của ảnh đầu vào. Khi đó giá trị biên độ đạo hàm và giá trị góc tại pixel được tính như sau:

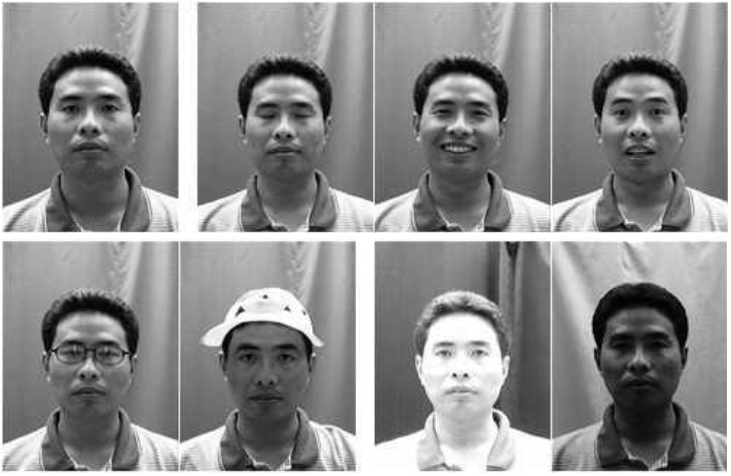
Thuật toán sau khi tính toán đạo hàm và kết quả trả về như ảnh sau:



*Hình 1.1. Kết quả khi tính toán đạo hàm trên ảnh đầu vào*

* + 1. **Đánh giá mô hình thuật toán HOG**

Sử dụng bộ dữ liệu khuôn mặt CAS – PEAL của Học viên Khoa học Trung Quốc với 6992 bức ảnh. Ở ví dụ bên dưới, bức ảnh đầu tiên ở trạng thái bình thường, bức ảnh thứ hai đến bức ảnh thứ tư khuôn mặt biểu bộ thêm các cảm xúc, ảnh thứ năm và thứ sáu khi mang thêm trang phục và hai ảnh cuối độ sáng thay đổi.



*Hình 1.2. Một vài ảnh trong bộ dữ liệu CAS – PEAL*

Cho dữ liệu trên vào mô hình, khi đó kết quả như sau:

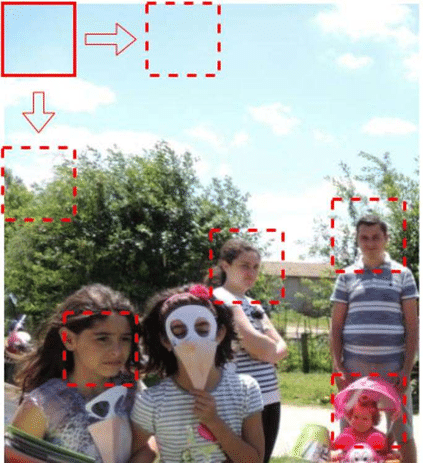
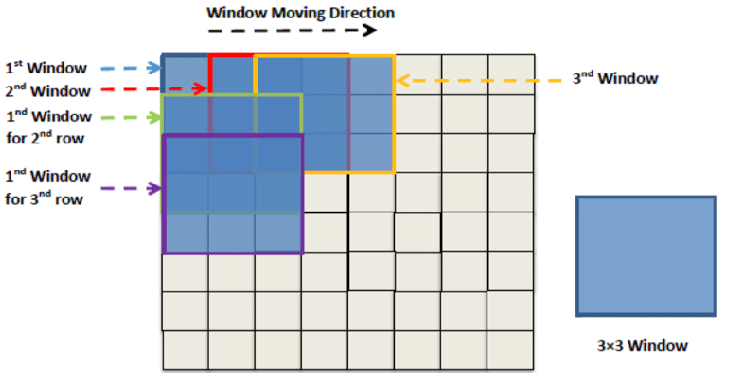
*Bảng 1.1. Kết quả thực hiện trên bộ dữ liệu CAS – PEAL*

|  |  |
| --- | --- |
| **Mức độ nhận diện** | **Tỷ lệ chính xác** |
| Điều kiện thiếu ánh sáng | 66% |
| Có thêm trang phục | 74% |
| Biểu cảm trên khuôn mặt | 84% |

(\*) Số liệu được tham khảo tại Paper “Research on Face Recognition Algorithm Based on HOG Feature” năm 2021 của Yalan Li.

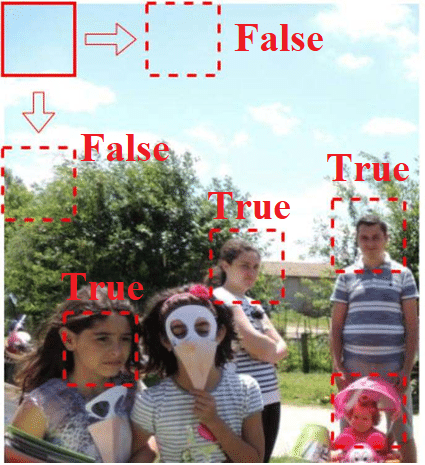
**1.1.3. Phương pháp sử dụng cho việc phát hiện khuôn mặt**

Sử dụng một Sliding Window có kích thước cố định để quét qua ảnh.



*Hình 1.3. Phương pháp sử dụng Sliding Window*

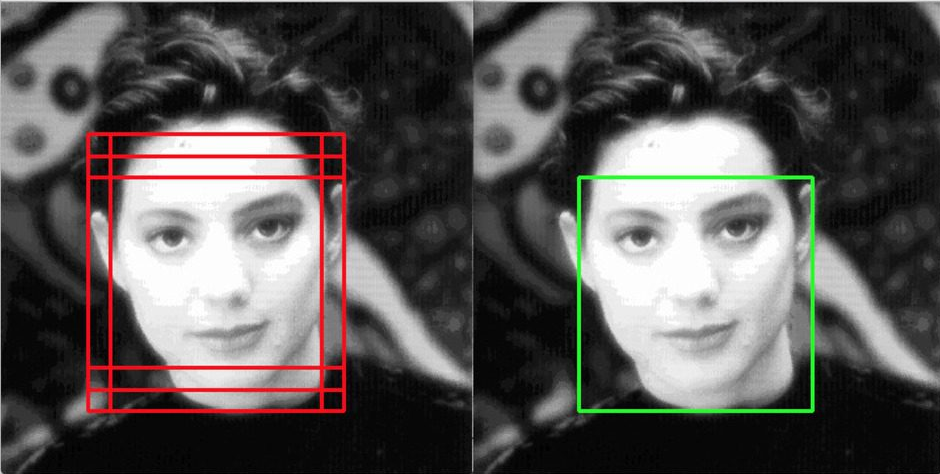
Tại mỗi Window, ảnh được đưa vào mô hình HOG hoặc SVM (gọi bởi hàm **get\_frontal\_detector()**) để kiểm tra và kết quả trả về tại mỗi Window là True hoặc False.



*Hình 1.4. Kết quả trả về của hàm get\_frontal\_detector()*

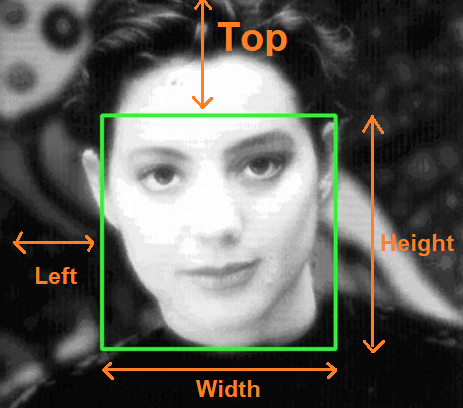
Sau mỗi một lượt quét qua toàn bộ ảnh, quá trình sẽ lặp lại với kích thước ảnh gốc sẽ được thay đổi (resize) để chọn kích thước Window phù hợp nhất với kích thước khuôn mặt.

Tại mỗi khuôn mặt có nhiều vị trí của Window khác nhau, kỹ thuật Non – Maximum Suppression sẽ được sử dụng để lựa chọn Window tối ưu nhất.



*Hình 1.5. Kỹ thuật Non – Maximum Suppression*

Sau khi tối ưu hóa và chọn ra Window phù hợp nhất với mỗi khuôn mặt, kết quả trả về là bộ 4 số (Top, Left, Width, Height) cho vị trí khuôn mặt đã được phát hiện.



*Hình 1.6. Kết quả trả về cuối cùng của hàm phát hiện khuôn mặt*

* 1. **Nhận diện các đặc điểm trên khuôn mặt**

Chức năng: Trả về tọa độ của 68 điểm trên khuôn mặt.

Tên hàm sử dụng: **\_raw\_face\_landmarks (Face\_Image)**

(với Face\_Image là hình ảnh khuôn mặt)

Sơ đồ khối:

[ p[0], p[1], p[2], …, p[67] ]

Point p =

p.x

p.y

[ p[0], p[1], p[2], …, p[67] ]

p.x

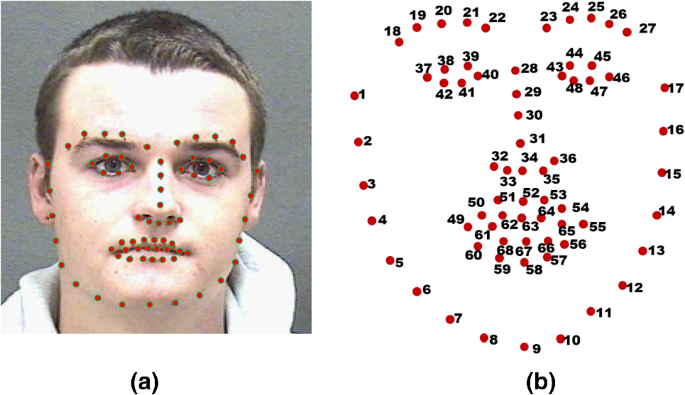
Point p =

p.y

68 points Model

Face Image

Các điểm trả về như sau:



*Hình 1.7. 68 điểm trả về của hàm nhận diện*

Đường viền khuôn mặt: 17 điểm (1-17)

Mi mắt: 10 điểm (18 - 27)

Mũi: 9 điểm (28 - 36)

Mắt: 12 điểm (37 - 48)

Miệng: 20 điểm (49 - 68)

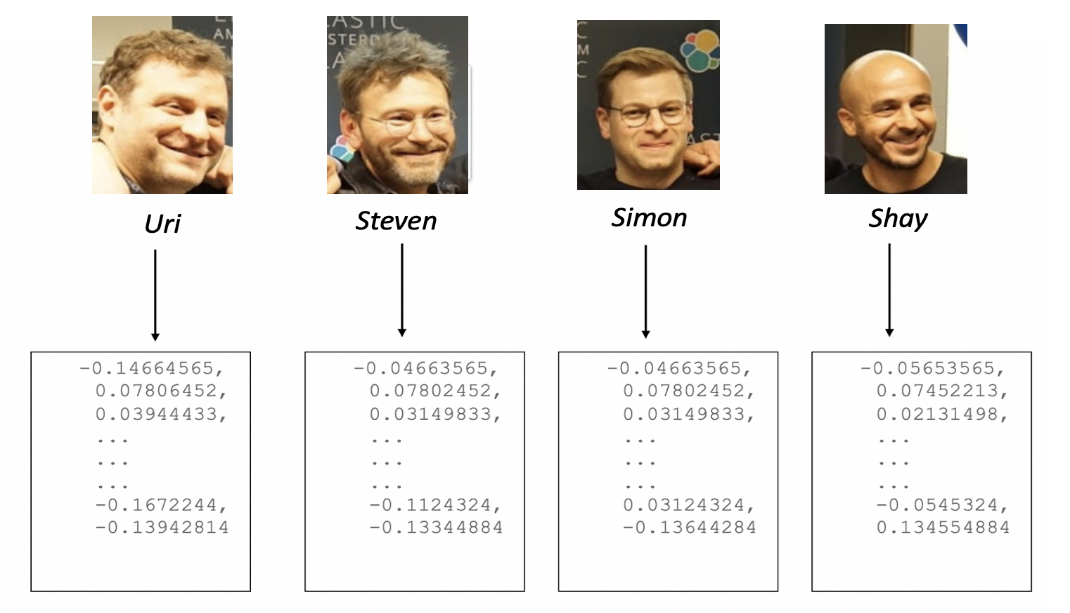
* 1. **Mã hóa khuôn mặt**

Chức năng: Trả về một vector thực (float) có kích thước [1x128] biểu diễn đặc trưng của khuôn mặt.

Tên hàm sử dụng: **face\_encodings (Face\_Image)**

(với Face\_Image là hình ảnh khuôn mặt)

Khi đưa một bức ảnh khuôn mặt vào hàm, kết quả trả về sẽ được biểu diễn như sau:



*Hình 1.8. Các khuôn mặt được mã hóa*

Hàm sử dụng mô hình CNN – Convolution Neuron Network, nhận vị trí 68 điểm trên khuôn mặt là đầu vào và kết quả trả về là một vector có 128 giá trị.

Sơ đồ khối:

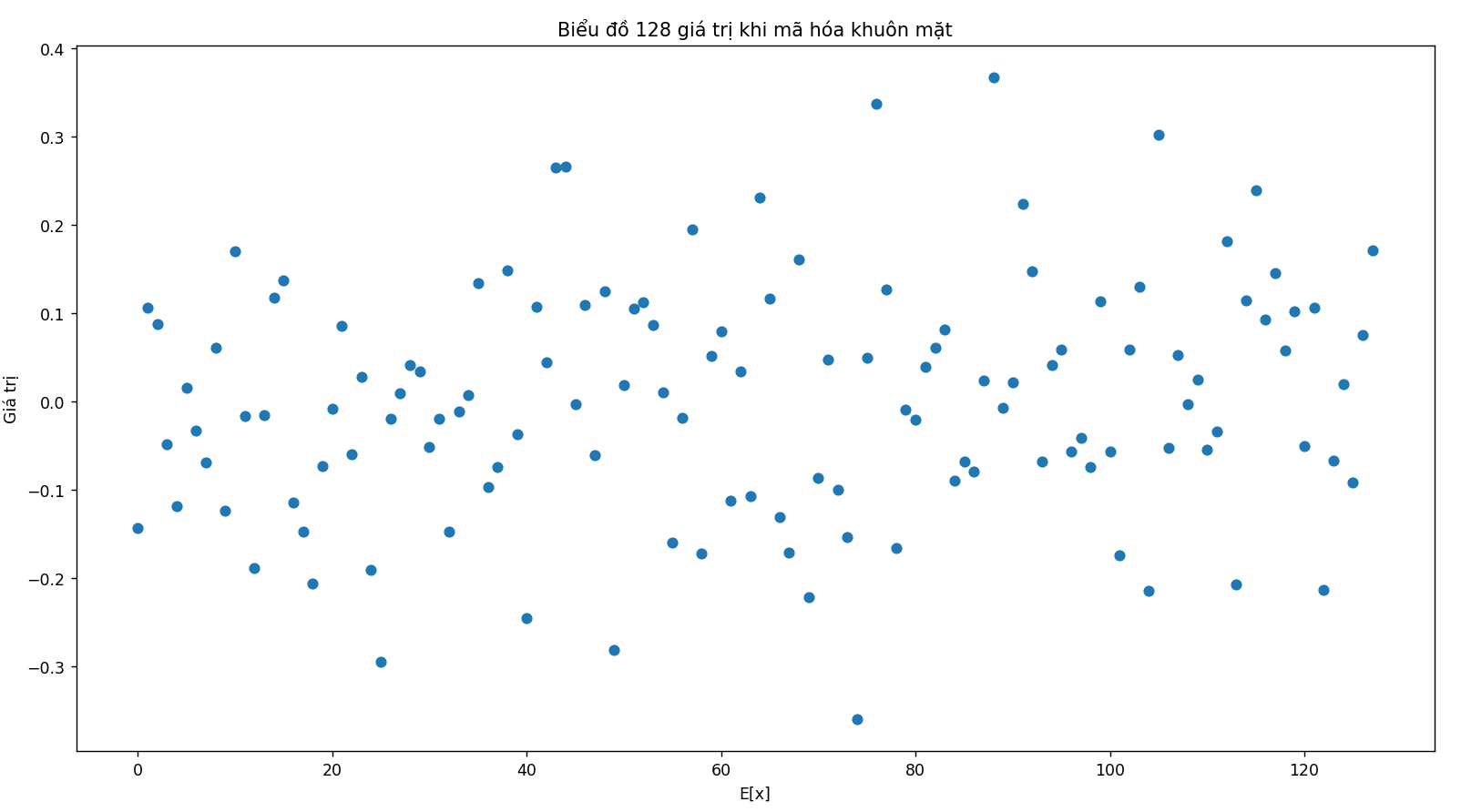
ResNet Model

68 Points Model

Face Image

float [ e1, e2, e3, …, e128 ]

Các giá trị của vector mã hóa một khuôn mặt phân bố như sau:



*Hình 1.9. Đồ thị thể hiện giá trị của 128 phần tử trong một vector mã hóa*

Ở mỗi phần tử là một giá trị thực kiểu float có kích thước 64 bit hay 8 byte. Như vậy với mỗi khuôn mặt khi cần mã hóa ta cần:

8 byte x 128 = 1024 byte hay 1 Kb

* 1. **So sánh khuôn mặt**

Chức năng: Giúp so sánh 2 khuôn mặt đã được mã hóa có phải là cùng một người hay không, kết quả trả về là một số thực có giá trị không âm thể hiện khoảng cách về độ trùng khớp của hai vector khuôn mặt đã mã hóa, càng gần 0 thể hiện độ trùng khớp càng cao và ngược lại.

Tên hàm sử dụng:  **face\_distance (encode1, encode2)**

( với encode1 và encode2 lần lượt là kết quả mã hóa của khuôn mặt 1 và khuôn mặt 2 )

Sơ đồ khối:

E2: [e1, e2, …, e128]

E1: [e1, e2, …, e128]

face\_distance

Cách hoạt động:

Hàm sử dụng công thức khoảng cách Euclidean để đo lường sự khác biệt giữa các Vector đặc trưng.

Công thức cho khoảng cách giữa hai vector A và B có chiều dài n là:

**Phần 2**

**XÂY DỰNG MÔ HÌNH NHẬN DIỆN KHUÔN MẶT**

Mô tả: Xây dựng một thuật toán để nhận diện khuôn mặt từ 2 tấm ảnh, kiểm tra xem 2 khuôn mặt trong ảnh có phải là một người hay không ?

Ta sử dụng kết hợp 4 tính năng chính có sẵn trong thư viện face\_recognition như phân tích phía trên để xây dựng mô hình.

Các bước thực hiện:

- Ở 2 ảnh đầu vào, sử dụng **face\_detector** để phát hiện chính xác vị trí của khuôn mặt.

- Sau khi phát hiện khuôn mặt, đưa 2 khuôn mặt qua 68 Points Model để nhận diện 68 đặc điểm trên khuôn mặt.

- Sau đó, tiến hành mã hóa 2 khuôn mặt trên từ 68 điểm vừa tìm được trên mỗi khuôn mặt.

- Cuối cùng sử dụng **face\_distance** để tính khoảng cách giữa 2 vector mã hóa. Chọn ngưỡng 0.5 (ngưỡng tối ưu nhất – đã được bên phía OpenCV kiểm tra), nếu khoảng cách nhỏ hơn 0.5 là cùng 1 người và ngược lại.

Sơ đồ khối:

* **Bộ mã hóa**

**[1x128]**

**ENCODE**

ResNet Model

68 Points Model

Face Detector

Image

* **Mô hình nhận diện**

Number

face\_distance

ENCODE

Ảnh khuôn mặt

cần kiểm tra

Image

ENCODE

Ảnh khuôn mặt có sẵn trong bộ dữ liệu

Image

True/False

Lưu đồ:

Đúng

Sai

“2 người khác nhau”

“Cùng 1 người”

d < 0.5

d = face\_distance(E1, E2)

E1 = face\_encodings(Image1)

E1 = face\_encodings(Image1)

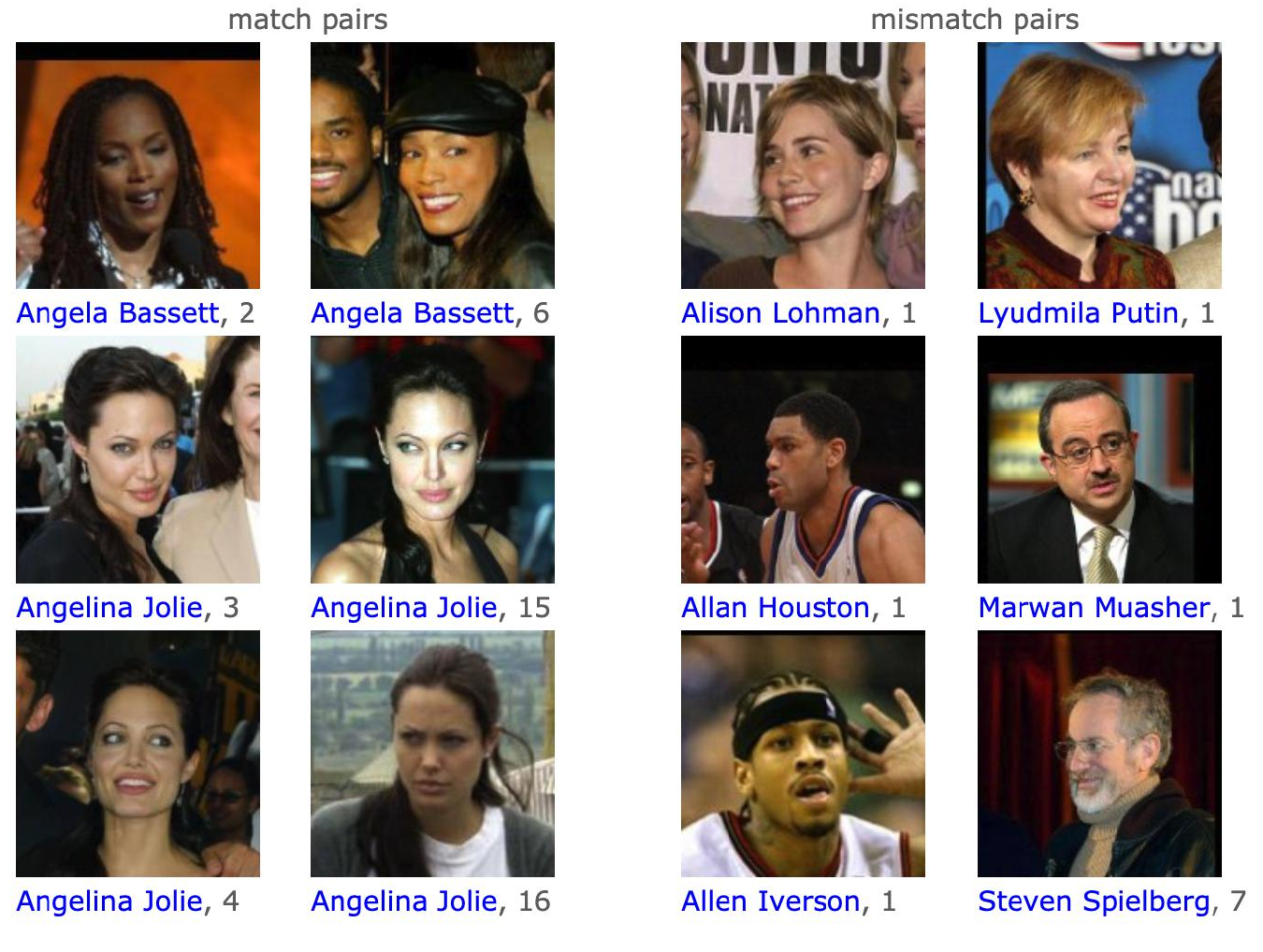
Đọc ảnh 2 – Image2

Đọc ảnh 1 – Image1

**Phần 3**

**ĐÁNH GIÁ MÔ HÌNH**

Ta sử dụng bộ dữ liệu LFW (Labeled Faces in the Wild) gồm 13,233 hình ảnh của 5,749 người khác nhau để đánh giá tốc độ và hiệu năng của mô hình.



*Hình 3.1. Bộ dữ liệu LFW*

**3.1. Đánh giá về tốc độ**

Viết một chương trình để tính thời gian cho một khối xử lý với 1000 ảnh với lưu đồ như sau:

Đúng

i = i + 1

total\_time = end\_time – start\_time

end\_time = time of system

Sai

i < 1000

Process (Image)

Image = /\* Ảnh thứ i trong data \*/

i = 0

start\_time = time of system

Chương trình được chạy trên máy tính có bộ xử lý “**Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz 2.59 GHz**” với 2 nhân và 4 luồng xử lý.

Kết quả

* **Bộ mã hóa**

**[1x128]**

ResNet Model

68 Points Model

Face Detector

Image

**4,372s / 1000 ảnh**

**1,631s / 1000 ảnh**

**1,882s / 1000 ảnh**

Tại hàm face\_detector: 1,882s / 1000 ảnh.

Tại mô hình nhận dạng 68 điểm: 1,631s / 1000 ảnh.

Tại mô hình ResNet: 4,372s / 1000 ảnh.

Như vậy, tổng thời gian để mã hóa là **7,855s / 1000 ảnh.**

* **Mô hình nhận diện**

**7,855s / 1000 ảnh**

Yes/No

Number

face\_distance

ENCODE

Ảnh khuôn mặt có sẵn trong bộ dữ liệu

Image

**5,9292s / 1000 lượt**

**7,855s / 1000 ảnh**

ENCODE

Image

Ảnh khuôn mặt

cần kiểm tra

Tại hàm face\_distance: 5,9292s / 1000 lượt so sánh.

Như vậy toàn bộ mô hình cần 21,6992s / 1000 lượt để nhận diện 2 khuôn mặt có cùng một người hay không.

**3.2. Đánh giá về hiệu năng**

Tạo một bộ dữ liệu mới từ LFW trong đó mỗi dữ liệu là một cặp hình ảnh của 2 khuôn mặt và được đánh label True (nếu 2 ảnh là cùng một người) hoặc False (nếu 2 ảnh là 2 người khác nhau).

Bộ dữ liệu gồm 16,898 cặp ảnh với:

8,662 cặp ảnh là cùng một người.

8,236 cặp ảnh là 2 người khác nhau.



*Hình 3.2. Tạo bộ dữ liệu để so sánh và đánh nhãn từ LFW*

Cho toàn bộ 16,898 cặp ảnh trên qua mô hình nhận diện đã xây dựng ở Phần 2, so sánh kết quả ngõ ra và label ta thu được kết quả như sau:

Cùng một người: đúng 7,425 / 8,662. Tỷ lệ chính xác 85.7192 %.

Hai người khác nhau: đúng 8232 / 8236. Tỷ lệ chính xác 99.9514 %.

Như vậy tỷ lệ nhận diện chính xác của mô hình trên bộ dữ liệu này là 92,1596 %.

Tương tự, tôi cũng tạo một bộ dữ liệu khuôn mặt được lấy từ những người thân xung quanh với 40 cặp ảnh, trong đó:

20 cặp ảnh là cùng một người.

20 cặp ảnh là 2 người khác nhau.

Cho 40 cặp ảnh trên qua mô hình nhận diện khuôn mặt, ta thu được kết quả như sau:

Cùng 1 người: đúng 20/20.

Hai người khác nhau: đúng 17/20. 3 cặp bị sai trong đó có 2 cặp khuôn mặt không quay thẳng vào camera, 1 cặp còn lại là khuôn mặt của 2 anh em ruột.

Như vậy, tỷ lệ nhận diện đúng là 92,5%.

**Nhận xét:**

Với tốc độ xử lý nhanh và độ chính xác khá cao, ta có thể triển khai mô hình nhận diện khuôn mặt ở Phần 2 vào việc ứng dụng – xây dựng hệ thống kiểm soát ra vào hoàn chỉnh bằng nhận diện khuôn mặt.

**Phần 4**

**TRIỂN KHAI MÔ HÌNH**

Tên mô hình: Hệ thống kiểm soát ra vào dựa trên nhận diện khuôn mặt.

Các công cụ sử dụng để xây dựng:

Ngôn ngữ lập trình Python 3.xx.

Thư viện OpenCV, face\_recognition,…

Cơ sở dữ liệu thời gian thực: Firebase.

Công cụ thiết kế giao diện: GIMP.

**4.1. Sơ đồ khối**

Ta thiết kế hệ thống theo sơ đồ khối sau:

Faces Encoded

Faces Encoded

Creat Data

Software

Info Data (ID, Name, …)

Data

Time of in/out door

Realtime Database

Server

Face Recognition Device

Creat Data Software: Phần mềm để tạo dữ liệu khi có người đăng ký khuôn mặt vào hệ thống. Thu thập các thông tin bao gồm ID, Tên, Tuổi, Vị trị,… và hình ảnh khuôn mặt.

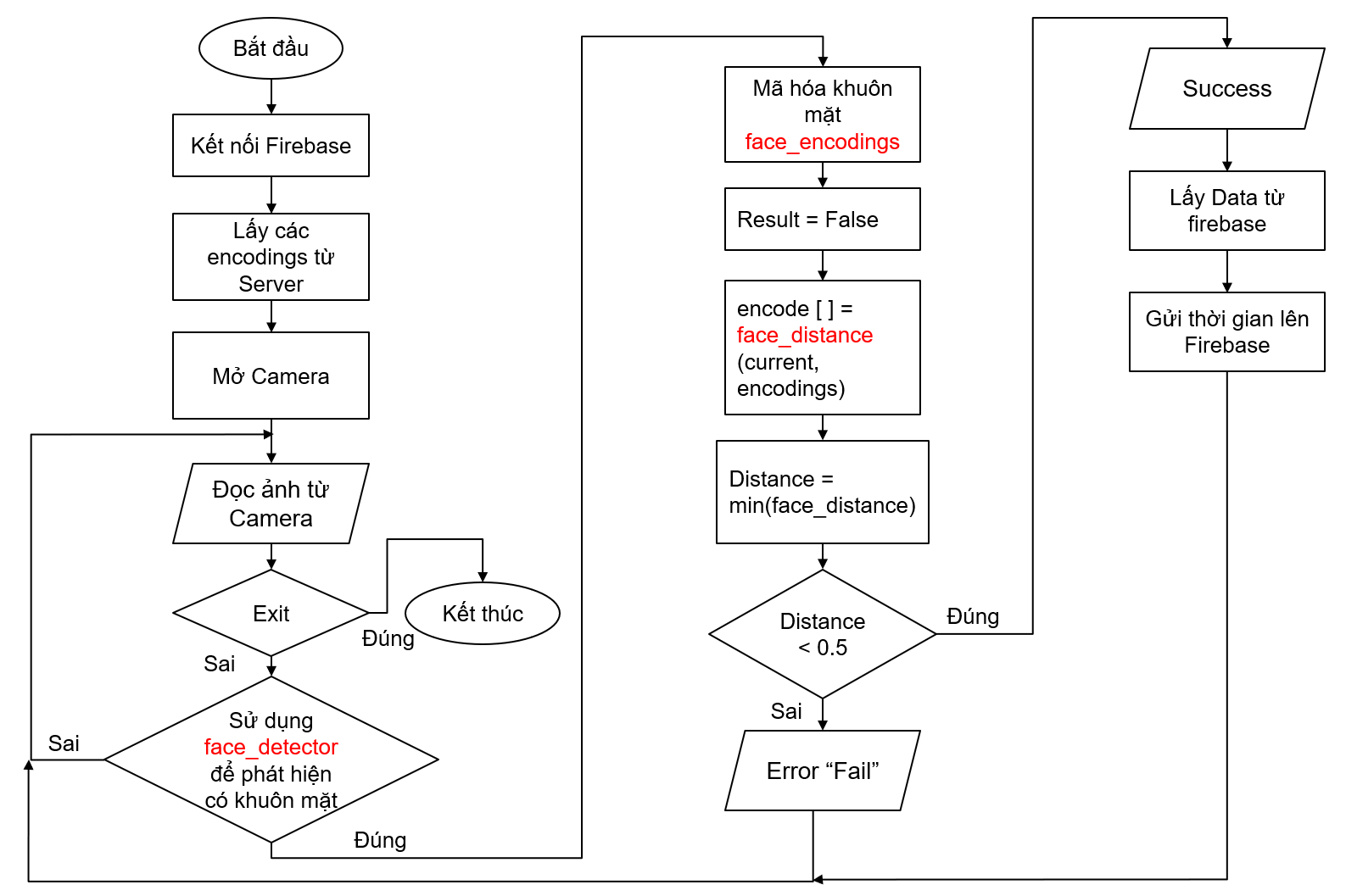
Server: Tại đây lưu trữ toàn bộ các vector mã hóa các khuôn mặt đã đăng ký hệ thống.

Realtime Database: Các thông tin dữ liệu về người dùng sẽ lưu trữ tại đây, đồng thời khi có tín hiệu nhận diện khuôn mặt từ hệ thống thì thời gian người sử dụng sẽ được cập nhật ngay lập tức.

Face Recognition Device: Ứng dụng mô hình thuật toán nhận diện khuôn mặt ở Phần 2, hệ thống quét khuôn mặt khi người dùng sử dụng và thông báo nhận diện thành công hay không.

**4.2. Lưu đồ xây dựng Face Recognition Device**

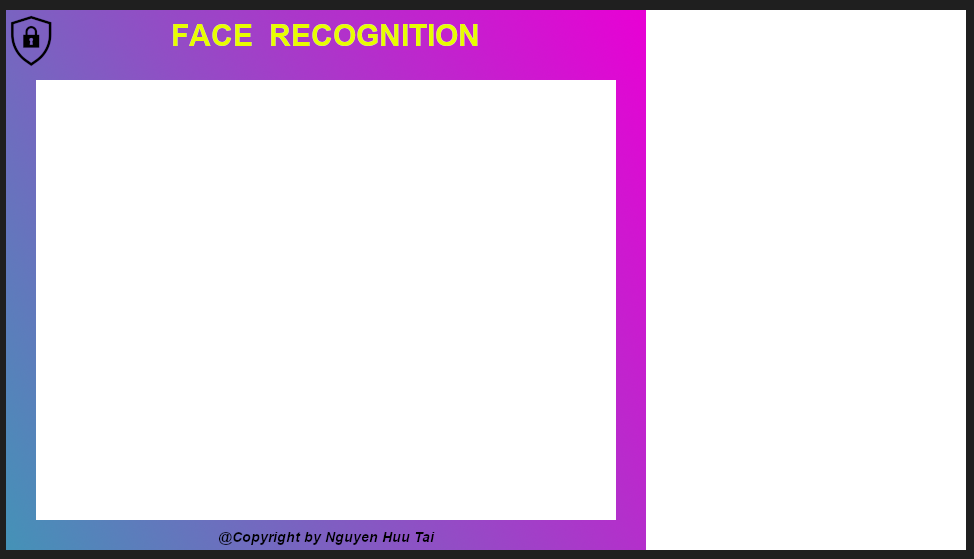
Ta xây dựng thiết bị nhận diện khuôn mặt, liên kết với Server và Firebase theo lưu đồ sau:



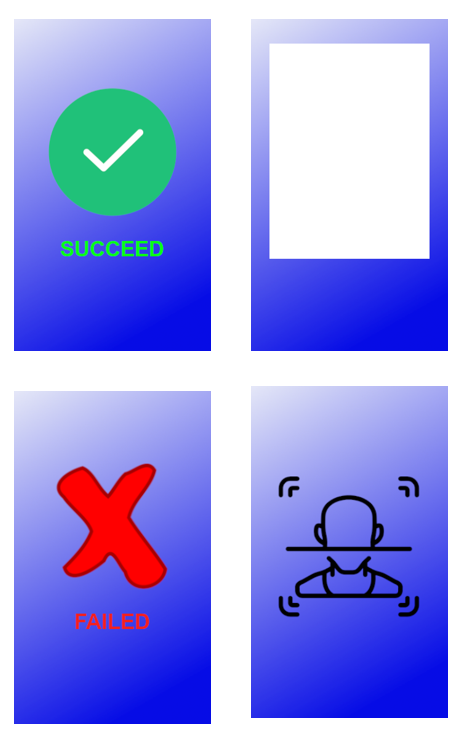
*Hình 4.1. Lưu đồ thiết bị nhận diện khuôn mặt*

**4.3. Thiết kế giao diện**

Sử dụng phần mềm thiết kế GIMP, ta xây dựng các giao diện hiển thị như sau:



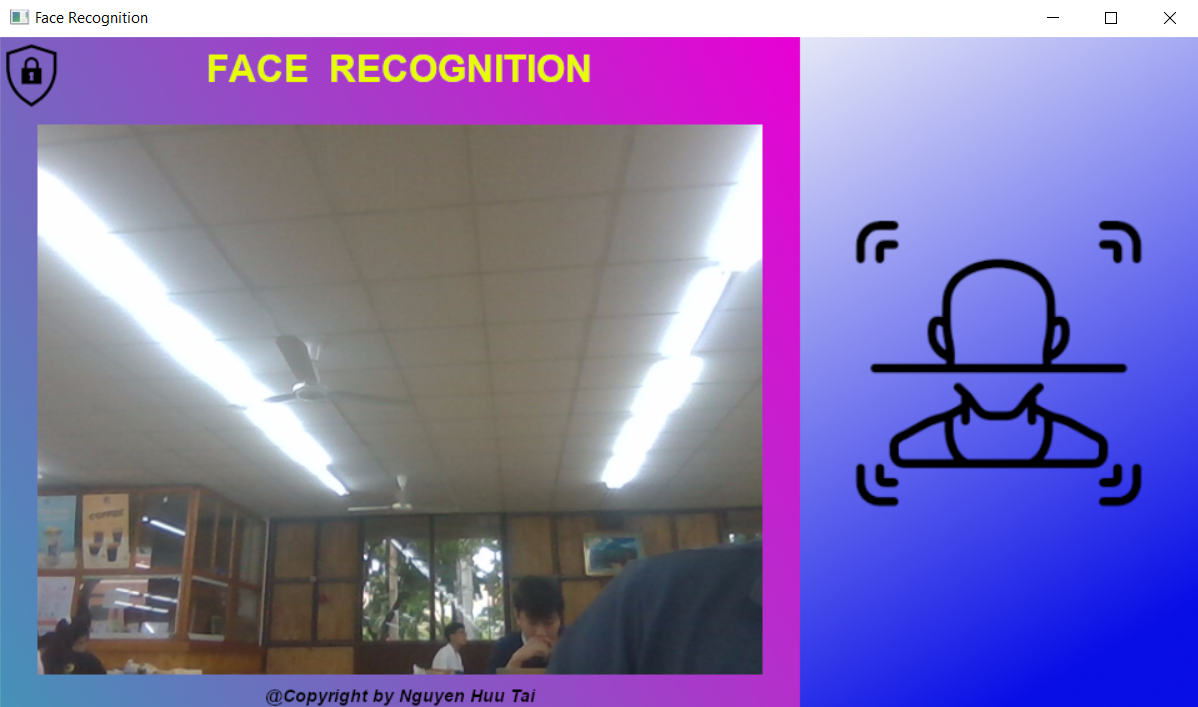
*Hình 4.2. Background của giao diện*



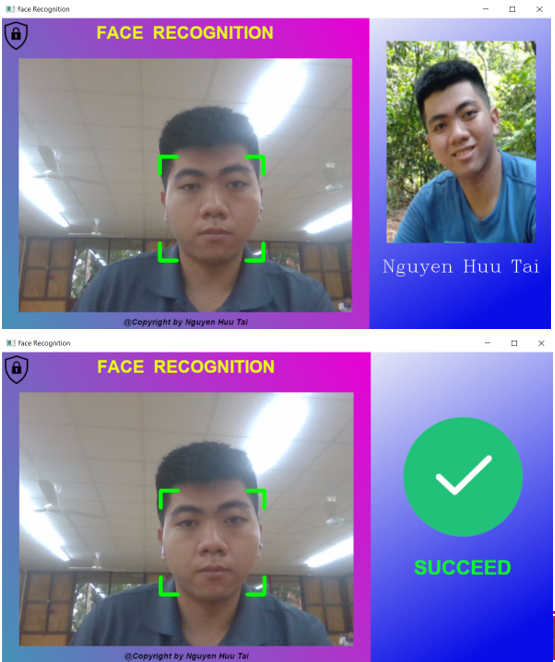
*Hình 4.3. Các chế độ hiển thị*

**4.4. Hoàn thành sản phẩm**

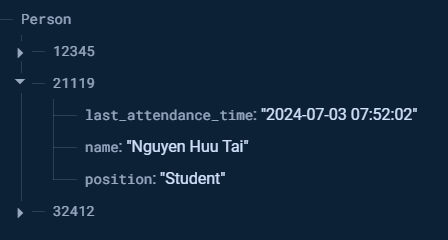
Hoàn thành toàn bộ quá trình trên, ta được hệ thống hoàn chỉnh hoạt động ổn định. Một số hình ảnh hoạt động của hệ thống như sau:



*Hình 4.4. Thiết bị hoạt động khi không có người sử dụng*



*Hình 4.5. Khi có người sử dụng và nhận diện thành công*



*Hình 4.6. Thời gian khi người dùng nhận diện được cập nhật lên Firebase*

**KẾT LUẬN**

Trong đồ án này, tôi đã thành công trong việc triển khai một hệ thống kiểm soát ra vào dựa trên nhận diện khuôn mặt sử dụng ngôn ngữ lập trình Python và thư viện face\_recognition của OpenCV. Hệ thống đã được tích hợp với camera máy tính để quét và nhận diện khuôn mặt, đồng thời cập nhật thời gian sử dụng vào cơ sở dữ liệu lưu trữ trên Firebase khi phát hiện khuôn mặt trùng khớp với khuôn mặt đã đăng ký trước đó.

Qua quá trình thực hiện, tôi đã tiến hành phân tích sâu vào thư viện face\_recognition, bao gồm cách thức hoạt động, các thuật toán và mô hình sử dụng bên trong thư viện. Kết quả phân tích cho thấy thư viện này cung cấp các công cụ mạnh mẽ và dễ sử dụng cho việc nhận diện khuôn mặt, với độ chính xác cao và tốc độ xử lý nhanh.

Hệ thống kiểm soát ra vào của tôi đã chứng minh được tính hiệu quả và khả năng ứng dụng thực tiễn trong các tình huống kiểm soát an ninh và quản lý ra vào. Tuy nhiên, hệ thống mà tôi triển khai trong đồ án này được xây dựng trên máy tính đa dụng cá nhân, không có ý nghĩa trong sử dụng thực tế.

Để cải thiện và nâng cao hiệu quả của hệ thống trong tương lai, tôi đề xuất các hướng nghiên cứu và phát triển tiếp theo như: sử dụng các thuật toán tiên tiến hơn cho nhận diện khuôn mặt, tích hợp thêm các cảm biến phụ trợ để cải thiện chất lượng hình ảnh, và tối ưu hóa hiệu năng của hệ thống để giảm thiểu thời gian xử lý. Bên cạnh đó kích thước của các Model có sẵn với dung lượng rất lớn (HOG: 70MB, 68 Points Model: 100MB, …) cần phải cải thiện giảm kích thước thì mới có thể phát triển lên khi xây dựng trên các thiết bị nhỏ gọn hơn như Raspberry Pi hay ESP.

Nhìn chung, đồ án đã mang lại những kết quả tích cực, không chỉ trong việc triển khai một hệ thống kiểm soát ra vào hiệu quả, mà còn trong việc hiểu rõ hơn về các công nghệ và phương pháp nhận diện khuôn mặt hiện đại.

**TÀI LIỆU THAM KHẢO**

[1] Himanshu Singh, *Practical Machine Learning and Image Processing*, Copyright © 2019 by Himanshu Singh, India.

[2] Tutorials Point, *OpenCV With Python*, Copyright © 2021 by Tutorials Point (I) Pvt. Ltd, India.

[3] Adrian Rosebrock, *Deep Learning for Computer Vision*, PyImageSearch 2017, USA.

[4] OpenCV, *Face Recognition with OpenCV*, opencv.org,<https://docs.opencv.org/4.x/da/d60/tutorial_face_main.html>, 14/3/2024.

[5] Firebase, *Firebase Realtime Database*, firebase.google, <https://firebase.google.com/docs/database>, 14/3/2024.

[6] Yalan Li (2021), *Research on Face Recognition Algorithm Based on HOG Feature*, Journal of Physics: Conference Series, <https://iopscience.iop.org/article/10.1088/1742-6596/1757/1/012099/pdf>, 2/7/2024.

[7] Gary, *LFW (Labeled Faces in the Wild)*, Papers With Code, <https://paperswithcode.com/dataset/lfw>, 2/7/2024.