**TRƯỜNG ĐẠI HỌC SÀI GÒN**



**KHOA CÔNG NGHỆ THÔNG TIN**

**MÔN HỌC: CƠ SỞ TRÍ TUỆ NHÂN TẠO**

**BÁO CÁO ĐỒ ÁN:**

**ĐỀ TÀI: ỨNG DỤNG TRÍ TUỆ NHÂN TẠO NHẬN DIỆN KHUÔN MẶT BẰNG NGÔN NGỮ PYTHON**

**Giảng viên: Ths.NCS Phạm Hoàng Vương**

**Thành viên nhóm:**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | Mã số sinh viên |
| 1 | Bùi Quang Minh Hiếu | 3122410112 |
| 2 | Hoàng Đình Hoàn | 3122410123 |
| 3 | Nguyễn Văn Hiếu | 3122410118 |
| 4 | Nguyễn Hoàng Hải | 3122410095 |
| 5 | Đoàn Tuấn Tài | 3122410365 |

Thành phố Hồ Chí Minh, tháng 4, năm 2024

**Nhận xét, đánh giá của giảng viên**

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

**LỜI CẢM ƠN**

Để hoàn thành được bài tiểu luận này, em xin chân thành cảm ơn và gửi lời tri ân sâu sắc đến quý thầy, cô thuộc bộ môn ngôn ngữ lập trình python thuộc khoa Công nghệ thông tin, trường Đại học Sài Gòn đã tạo cơ hội cho em được học tập, nghiên cứu và tích lũy kiến thức để thực hiện bài báo cáo đồ án. Trên hết, em xin được gửi lời cảm ơn chân thành nhất đến giảng viên, ThS.NCS Phạm Hoàng Vương đã tận tình chỉ dẫn và đưa ra nhiều lời khuyên bổ ích giúp em hoàn thành bài báo cáo này một cách tốt nhất.

Do kiến thức của bản thân còn nhiều hạn chế và thiếu kinh nghiệm thực tiễn nên nội dung bài báo cáo khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được những lời góp ý thêm từ quý thầy cô, để chúng em học hỏi thêm được nhiều kỹ năng, kinh nghiệm trong quá trình học tập và sẽ hoàn thành tốt hơn cho những bài báo cáo sắp tới.

Chúng em xin chân thành cảm ơn thầy.

Trân trọng.

**Mục lục**

**Phần 1: Mở đầu……….............................................................................3**

**Phần 2: Cơ sở lý thuyết………………………………………………….5**

1. **Tổng quan về trí tuệ nhân tạo…………………………………………….5**
   1. Khái niệm……………………………………………………………5
   2. Phân loại……………………………………………………………..6
   3. Mặt tích cực và mặt khó khăn của trí tuệ nhân tạo…………………..6
   4. Ứng dụng của trí tuệ nhân tạo……………………………………….7
2. **Tổng quan về đề tài………………………………………………………..8**
   1. Mô tả ứng dụng………………………………………………………8
   2. Lịch sử hình thành và phát triển ứng dụng…………………………..8
3. **Ứng dụng và các công nghệ áp dụng trong ứng dụng…………………..9**
   1. Tổng quan về ngôn ngữ Python……………………………………...9
   2. Thư viện opencv của Python………………………………………...9
   3. Thuật toán LBPH…………………………………………………….9

**Phần 3: Xây dựng và triển khai ứng dụng……………………………12**

1. **Xây dựng ứng dụng………………………………………………………12**
   1. Tải các thư viện…………………………………………………….12
   2. Bước 1\_Face\_Dataset………………………………………...........13
   3. Bước 2\_Face\_Tranning……………………………………………16
   4. Bước 3\_Face\_Recognition…………………………………...........20
   5. Trainer……………………………………………………………..25
   6. haarcascade\_frontalface\_default………………………..........26
2. **Hoạt động của ứng dụng………………………………………………...26**
3. **Hoàn thiện và hướng phát triển ứng dụng……………………………..29**

**Phần 4: Kết luận……………………………………………………….30**

**Tài liệu tham khảo…………………………………………………………...30**

**PHẦN 2: CƠ SỞ LÝ THUYẾT**

1. **Tổng quan về trí tuệ nhân tạo**
   1. **Khái niệm**

Trí tuệ nhân tạo hay trí thông minh nhân tạo (Atificial intelligence hay machine intelligence, thường được viết tắt là AI) là trí tuệ được biểu diễn bởi bất cứ một hệ thống nhân tạo nào.

Trong Công Nghệ Thông Tin, Trí Tuệ Nhân Tạo (TTNT) là một ngành mới, nhưng phát triển rất mạnh mẽ và đem lại nhiều kết quả to lớn.

– **Tri thức** là sự hiểu biết bằng lý thuyết hay thực tế về một chủ đề hay lĩnh vực. Tri thức là tổng của những cái đang biết hiện nay; tri thức là sức mạnh. Những người có tri thức tốt là những chuyên gia (expert).

– **Thông minh (Intelligence):** sự nghiên cứu, sự thu thập thông tin tiêu biểu như: cố gắng học, những ý tưởng xử lý của nộ não con người, bao gồm cả việc nghiên cứu sự vật có ý tưởng, có ý nghĩa, có sự chú ý, nhận dạng, hiểu vấn đề và sáng tạo ra vấn đề.

– **Nhân tạo (Artifcial):** có ý nghĩa là cố gắng sử dụng máy tính để xây dựng những hệ thống nhân tạo bắt chước đặc tính của việc thu thập thông tin một cách thông minh.

– So với chương trình truyền thống (được cấu tạo từ dữ liệu và thuật toán), chương trình trí tuệ nhân tạo được cấu tạo từ hai thành phần là cơ sở tri thức (knowledge base) và động cơ suy diễn (inference engine).

– **Cơ sở tri thức (Knowledge base):** là tập hợp các tri thức liên quan đến vấn đề mà chương trình quan tâm giải quyết. Cơ sở tri thức chứa các kiến thức được sử dụng để giải quyết các vấn đề (bài toán) trong trí tuệ nhân tao.

– **Động cơ suy diễn:** là phương pháp vận dụng tri thức trong cơ sở tri thức để giải quyết vấn đề.

– **Thuật toán và thuật giải:** Trong quá trình nghiên cứu giải quyết các vấn đề - bài toán, người ta đã đưa ra những nhận xét như sau:

* + Có nhiều bài toán cho đến nay vẫn chưa tìm ra một cách giải theo kiểu thuật toán và cũng không biết là có tồn tại thuật toán hay không.
  + Có nhiều bài toán đã có thuật toán để giải nhưng không chấp nhận được vì thời gian giải theo thuật toán đó quá lớn hoặc các điều kiện cho thuật toán khó đáp ứng.
  + Có những bài toán được giải theo những cách giải vi phạm thuật toán nhưng vẫn chấp nhận được.
  1. **Các loại trí tuệ nhân tạo**

Vào năm 2020, trí tuệ nhân tạo được chia thành 4 loại riêng biệt. Các loại này được chia gần giống với cách chia hệ thống phân cấp nhu cầu của Maslow.

Bốn loại của trí tuệ nhân tạo đó là: Máy phản ứng, Bộ nhớ hạn chế, Lý thuyết tâm trí, Tự nhận thức.

1. **Công nghệ AI phản ứng (Reactive Machines)**

Máy phản ứng là cấp độ đơn giản nhất của AI. AI sẽ có khả năng phân tích những động thái khả nghi nhất của mình và đối thủ. Sau đó, sẽ đưa ra giải pháp tốt nhất.

1. **Công nghệ AI với bộ nhớ hạn chế (Limited Memory)**

Máy có bộ nhớ hạn chế, có thể giữ lại một số thông tin học được từ việc quan sát các sự kiện hoặc dữ liệu trước đó. AI có thể xây dựng kiến thức bằng cách sử dụng bộ nhớ đó kết hợp với dữ liệu được lập trình sẵn.

1. **Lý thuyết tâm trí (Theory of Mind)**

Con người có những suy nghĩ và cảm xúc, ký ức hoặc các mô hình não khác điều khiển và ảnh hưởng đến hành vi của họ. Dựa trên tâm lý này, các nhà nghiên cứu lý thuyết về tâm trí hy vọng phát triển các máy tính có khả năng bắt chước các mô hình tinh thần của con người. Máy móc có thể hiểu rằng con người và động vật có những suy nghĩ và cảm xúc có thể ảnh hưởng đến hành vi của chính chúng. Lý thuyết về máy móc tâm trí sẽ được yêu cầu sử dụng thông tin thu được từ con người và học hỏi từ nó, sau đó sẽ thông báo bằng cách máy móc giao tiếp hoặc phản ứng với một tình huống khác.

1. **Tự nhận thức (Self-awareness)**

Công nghệ AI này có khả năng tự nhận thức về bản thân, có ý thức và hành xử như con người. Thậm chí, chúng còn có thể bộc lộ cảm xúc cũng như hiểu được những cảm xúc của con người. Đây được xem là bước phát triển cao nhất của công nghệ AI và đến thời điểm hiện tại, công nghệ này vẫn chưa khả thi.

* 1. **Mặt tích cực và khó khăn của AI – Trí thông minh nhân tạo**
     1. **Tích cực**
* Mạng lưới thần kinh nhân tạo và công nghệ trí tuệ nhân tạo với khả năng học tập sâu đang phát triển nhanh chóng, AI xử lý được lượng lớn dữ liệu nhanh hơn nhiều và đưa ra dự đoán chính xác hơn khả năng của con người.
* Khối lượng dữ liệu khổng lồ được tạo ra hàng ngày sẽ gây khó khăn cho các nhà nghiên cứu, AI sử dụng học máy để có thể lấy những dữ liệu đó và nhanh chóng biến nó thành thông tin có thể thực hiện được.
  + 1. **Khó khăn**
* Việc sử dụng AI là tốn kém rất nhiều khi xử lý một lượng lớn dữ liệu mà lập trình AI yêu cầu.
* Khả năng giải thích sẽ một trở ngại trong việc sử dụng AI trong các lĩnh vực hoạt động theo các yêu cầu phải tuân thủ quy định nghiêm ngặt.
  1. **Ứng dụng của trí tuệ nhân tạo**
     1. **Chăm sóc sức khỏe**
* Các ứng dụng AI có thể cung cấp các thông tin về y khoa đã được cá nhân hóa theo dạng bài đọc. Các trợ lý chăm sóc sức khỏe cá nhân có thể hoạt động như một người hướng dẫn trong cuộc sống, nhắc nhở bạn uống thuốc, tập thể dục hoặc ăn uống lành mạnh hơn.

A belt with a metal buckle

Description automatically generated

*Đai thông minh – Welt dựa vào cảm biến từ tính để cảnh báo khi ăn quá nhiều.*

* + 1. **Sản xuất**
* Với hệ thống mạng định kỳ – loại hệ thống “deep learning” sử dụng chuỗi dữ liệu, AI có thể phân tích dữ liệu IoT từ nhà máy khi được truyền trực tiếp từ thiết bị đã được kết nối nhằm dự báo tải trọng và nhu cầu dự kiến.



*Ứng dụng công nghệ AI thiết kế xe tự lái*

* + 1. **Trong ngành bán lẻ**
* AI cung cấp tiềm năng mua sắm ảo cho người tiêu dùng khi đưa ra các đề xuất mang tính cá nhân hóa và tư vấn lựa chọn mua hàng. Công nghệ quản lý kho và kỹ thuật thiết kế web cũng sẽ được cải thiện với AI.
  + 1. **Ngân hàng**
* Trí tuệ nhân tạo nâng cao tốc độ, tính chính xác và hiệu quả công việc cho con người. Trong các tổ chức tài chính, AI có thể được sử dụng để xác định giao dịch có khả năng gian lận, áp dụng tính năng chấm điểm tín dụng một cách nhanh chóng và chính xác, cũng như tự động hóa các công việc thủ công trong quản lý dữ liệu “nặng”.
  + 1. **Trong ngành Marketing**
* AI trong Marketing đã trở nên phổ biến và bạn có thể tương tác với AI hàng ngày. Một vài trường hợp mà bạn đã tương tác với trí tuệ nhân tạo:
* *Công cụ tìm kiếm Google* sử dụng AI (thuật toán Rankbrain) để xác định kết quả phù hợp nhất cho một tìm kiếm.
* *Email marketing* tự động sử dụng AI để tìm ra những email tiềm năng dựa trên cách tương tác với một doanh nghiệp hoặc trang web.
* *Chatbots* trở nên ngày càng phổ biến hơn, tăng khả năng hỗ trợ khách hàng tức thời và hiệu quả.
* *Sử dụng công nghệ AI trong tìm kiếm giọng nói* trên loa thông minh hoặc điện thoại di động để xác định các tìm kiếm bằng từ khóa dài và truy vấn kết quả một cách tốt nhất.

1. **Tổng quan về đề tài**
   1. **Mô tả ứng dụng**

Công nghệ nhận diện khuôn mặt là một công nghệ sinh trắc học ánh xạ các đặc điểm khuôn mặt của một cá nhân về mặt toán học và lưu trữ dữ liệu dưới dạng faceprint (dấu khuôn mặt). Nói một cách dễ hiểu, nó cho phép nhận dạng một người cụ thể từ ảnh hoặc 1 đoạn video.

* 1. **Lịch sử hình thành và phát triển ứng dụng**

Lịch sử công nghệ nhận diện khuôn mặt đã trải qua nhiều bước tiến hóa từ những khám phá ban đầu đến các ứng dụng phổ biến ngày nay. Dưới đây là một tóm tắt về lịch sử của công nghệ này:

* **Những bước đầu tiên (1960s - 1980s**): Trong những năm đầu tiên, nghiên cứu về nhận diện khuôn mặt chủ yếu tập trung vào việc phát hiện và trích xuất các đặc điểm của khuôn mặt.
* **Các phương pháp thống kê (1990s):** Trong thập kỷ 1990, các phương pháp thống kê, bao gồm việc sử dụng đường biên da và các phân tích không gian màu, được áp dụng để phân tích hình ảnh khuôn mặt
* **Công nghệ nơ-ron nhân tạo (2000s):** Sự phát triển của công nghệ nơ-ron nhân tạo, đặc biệt là các mô hình deep learning, đã đem lại sự đột phá lớn trong lĩnh vực nhận diện khuôn mặt.
* **Ứng dụng thực tiễn (Hiện đại):** Ngày nay, công nghệ nhận diện khuôn mặt đã trở thành một phần không thể thiếu của nhiều ứng dụng thực tế. Các hệ thống nhận diện khuôn mặt được sử dụng rộng rãi trong các lĩnh vực như an ninh, giám sát, nhận dạng và xác minh cá nhân, nhận diện biểu cảm khuôn mặt, và thậm chí trong công nghệ tự lái và thị trường tiêu dùng (ví dụ: nhận diện khuôn mặt để mở khóa điện thoại di động).

1. **Ứng dụng và các công nghệ áp dụng trong ứng dụng**
   1. **Tổng quan về ngôn ngữ Python**

* Python là một trong những ngôn ngữ lập trình được sử dụng phổ biến nhất hiện nay, thích hợp cho người mới bắt đầu bởi vì ngôn ngữ dễ học, sử dụng nhiều thư viện có sẵn hỗ trợ cho người học một cách thật hiệu quả. Nó là một ngôn ngữ lập trình open-source miễn phí với các module hỗ trợ mở rộng và phát triển cộng đồng, dễ dàng tích hợp với các dịch vụ web, cấu trúc dữ liệu thân thiện với user và GUI-based desktop app.
* Nó là một ngôn ngữ lập trình phổ biến cho các ứng dụng machine learning và deep learning. Python tổng hợp nhiều thư viện hỗ trợ cho người học. Thư viện là một tập hợp các mã thường xuyên được sử dụng mà các nhà phát triển có thể bao gồm trong những chương trình Python của họ để không phải lập trình từ đầu. Theo mặc định, Python đi kèm với Thư viện chuẩn, chứa rất nhiều các hàm có thể tái sử dụng. Ngoài ra, hơn 137.000 thư viện Python có sẵn cho các ứng dụng khác nhau, bao gồm phát triển web, khoa học dữ liệu và máy học (ML).
  1. **Thư viện Opencv của Python**
* OpenCV-Python là một thư viện mà các nhà phát triển sử dụng để xử lý hình ảnh cho các ứng dụng thị giác máy tính. Thư viện này cung cấp nhiều hàm cho các tác vụ xử lý hình ảnh như đọc và ghi hình ảnh cùng lúc, xây dựng môi trường 3D từ môi trường 2D cũng như chụp và phân tích hình ảnh từ video.
  1. **Thuật toán LBPH (Local Binary Patterns Histograms)**
* Thuật toán Biểu đồ mô hình nhị phân cục bộ là một cách tiếp cận đơn giản gắn nhãn các pixel của hình ảnh ngưỡng vùng lân cận của mỗi pixel. Nói cách khác, LBPH tóm tắt cấu trúc cục bộ trong một hình ảnh bằng cách so sánh từng pixel với các pixel lân cận của nó và kết quả được chuyển đổi thành một số nhị phân. Nó được định nghĩa lần đầu tiên vào năm 1994 (LBP) và kể từ thời điểm đó, nó đã được coi là một thuật toán mạnh mẽ để phân loại kết cấu.
* Thuật toán này thường tập trung vào việc trích xuất các tính năng cục bộ từ hình ảnh. Ý tưởng cơ bản là không xem toàn bộ hình ảnh như một vectơ chiều cao; nó chỉ tập trung vào các tính năng cục bộ của một đối tượng.

A diagram of a square with numbers and a red dotted line

Description automatically generated

* Trong hình ảnh trên, lấy một pixel làm trung tâm và ngưỡng đối tượng của nó. Nếu cường độ của pixel trung tâm lớn hơn bằng với hàng xóm của nó, thì biểu thị nó bằng 1 và nếu không thì biểu thị nó bằng 0.
* Hãy hiểu các bước của thuật toán:

1. Chọn các tham số: LBPH chấp nhận bốn tham số:
2. **Radius**: Nó đại diện cho bán kính xung quanh pixel trung tâm. Nó thường được đặt thành 1. Nó được sử dụng để xây dựng mẫu nhị phân cục bộ tròn.
3. **Neighbors**: Số lượng điểm mẫu để xây dựng mẫu nhị phân hình tròn.
4. **Grid X**: Số ô theo hướng ngang. Càng biểu thị nhiều ô và lưới mịn hơn, thì kích thước của vectơ đặc trưng thu được càng cao.
5. **Grid Y**: Số ô theo hướng dọc. Càng biểu thị nhiều ô và lưới mịn hơn, thì kích thước của vectơ đặc trưng thu được càng cao.
6. **Huấn luyện thuật toán**: Bước đầu tiên là huấn luyện thuật toán. Nó yêu cầu một tập dữ liệu với hình ảnh khuôn mặt của người mà chúng tôi muốn nhận dạng. Một ID duy nhất (có thể là số hoặc tên của người) phải được cung cấp cùng với mỗi hình ảnh. Sau đó, thuật toán sử dụng thông tin này để nhận dạng hình ảnh đầu vào và cung cấp cho bạn kết quả đầu ra. Hình ảnh của một người cụ thể phải có cùng một ID. Hãy hiểu tính toán LBPH trong bước tiếp theo.
7. **Sử dụng phép toán LBP**: Trong bước này, phép tính LBP được sử dụng để tạo ra một hình ảnh trung gian mô tả hình ảnh gốc một cách cụ thể thông qua việc làm nổi bật đặc điểm khuôn mặt. Các tham số radius và neighbors được sử dụng trong khái niệm cửa sổ trượt.

A diagram of a number

Description automatically generated

* Để hiểu một cách cụ thể hơn, chúng ta hãy chia nó thành nhiều bước nhỏ:
* Giả sử hình ảnh khuôn mặt đầu vào là thang độ xám.
* Chúng ta có thể lấy một phần của hình ảnh này dưới dạng cửa sổ 3×3 pixel.
* Chúng ta có thể sử dụng ma trận 3×3 chứa cường độ của mỗi pixel (0-255).
* Sau đó, chúng ta cần lấy giá trị trung tâm của ma trận sẽ được sử dụng làm ngưỡng.
* Giá trị này sẽ được sử dụng để xác định các giá trị mới từ 8 hàng xóm.
* Đối với mỗi lân cận của giá trị trung tâm (ngưỡng), chúng tôi đặt một giá trị nhị phân mới. Giá trị 1 được đặt cho giá trị bằng hoặc cao hơn ngưỡng và 0 cho giá trị thấp hơn ngưỡng.
* Bây giờ ma trận sẽ chỉ bao gồm các giá trị nhị phân (bỏ qua giá trị trung tâm). Chúng ta cần quan tâm đến từng giá trị nhị phân từ mỗi vị trí từ dòng ma trận từng dòng thành các giá trị nhị phân mới (10001101). Có những cách tiếp cận khác để nối các giá trị nhị phân (theo chiều kim đồng hồ), nhưng kết quả cuối cùng sẽ giống nhau.
* Chúng tôi chuyển đổi giá trị nhị phân này thành giá trị thập phân và đặt nó thành giá trị trung tâm của ma trận, là một pixel từ ảnh gốc.
* Sau khi hoàn thành thủ tục LBP, chúng tôi nhận được hình ảnh mới, thể hiện các đặc điểm tốt hơn của hình ảnh gốc

A graph of a number of dots

Description automatically generated with medium confidence

1. Trích xuất biểu đồ từ hình ảnh: Hình ảnh được tạo ở bước cuối cùng, chúng ta có thể sử dụng các tham số Grid X và Grid Y để chia hình ảnh thành nhiều lưới, chúng ta hãy xem xét hình ảnh sau:

A diagram of a person's face

Description automatically generated

* Chúng tôi có một hình ảnh ở thang độ xám; mỗi biểu đồ (từ mỗi lưới) sẽ chỉ chứa 256 vị trí biểu thị sự xuất hiện của mỗi cường độ pixel.
* Cần phải tạo một biểu đồ mới lớn hơn bằng cách nối từng biểu đồ.

1. Thực hiện nhận dạng khuôn mặt: Bây giờ, thuật toán đã được đào tạo bài bản. Biểu đồ trích xuất được sử dụng để đại diện cho từng hình ảnh từ tập dữ liệu đào tạo. Đối với hình ảnh mới, chúng tôi thực hiện lại các bước và tạo biểu đồ mới. Để tìm hình ảnh phù hợp với hình ảnh đã cho, chúng ta chỉ cần so khớp hai biểu đồ và trả về hình ảnh có biểu đồ gần nhất.

* Có nhiều cách tiếp cận khác nhau để so sánh các biểu đồ (tính toán khoảng cách giữa hai biểu đồ), ví dụ: Khoảng cách Euclide, chi-square, giá trị tuyệt đối, v.v. Chúng ta có thể sử dụng khoảng cách Euclid dựa trên công thức sau:

A black text with a square and a square formula

Description automatically generated with medium confidence

* Thuật toán sẽ trả về ID dưới dạng đầu ra từ hình ảnh có biểu đồ gần nhất. Thuật toán cũng sẽ trả về khoảng cách tính toán có thể được gọi là đo độ tin cậy. Nếu độ tin cậy thấp hơn giá trị ngưỡng, điều đó có nghĩa là thuật toán đã nhận dạng khuôn mặt thành công.

Chúng tôi đã thảo luận về nhận diện khuôn mặt và nhận dạng khuôn mặt. Thuật toán thác nước được sử dụng để nhận diện khuôn mặt. Có nhiều thuật toán khác nhau để nhận dạng khuôn mặt, nhưng LBPH là thuật toán dễ dàng và phổ biến trong số đó. Nó thường tập trung vào các tính năng cục bộ trong hình ảnh

**PHẦN 3: XÂY DỰNG VÀ TRIỂN KHAI ỨNG DỤNG**

1. **Xây dựng ứng dụng**
   1. **Tải các thư viện về máy**
      1. Tải thư viện Opencv-python

****

cv2: Thư viện OpenCV, được sử dụng rộng rãi cho các công việc thị giác máy tính.

* + 1. Tải thư viện numpy

A black background with white text

Description automatically generated

numpy: Một thư viện cho tính toán số trong Python. Thường được sử dụng trong xử lý ảnh cho các phép toán mảng hiệu quả.

* + 1. Tải thư viện PIL của thư viện Image

A black background with white text

Description automatically generated

PIL: Thư viện Python Imaging, cung cấp các khả năng xử lý ảnh. Thường được sử dụng để mở, chỉnh sửa và lưu nhiều định dạng tệp ảnh khác nhau.

* + 1. Thư viện os đã có sẵn khi cài đặt python về máy.

os: Module os cung cấp một cách di động để tương tác với hệ điều hành. Thường được sử dụng cho các hoạt động tệp như đọc thư mục, kiểm tra sự tồn tại của tệp, v.v.

* 1. **Bước 1\_Face\_Dataset**

Import 2 thư viện cv2 (opencv-python) và os vào file.

A black rectangle with white text

Description automatically generated

A black rectangle with white text

Description automatically generated

Truyền 0 làm đối số cho **cv2.VideoCapture().** Sau đó, nó thiết lập độ phân giải của video được chụp thành 640x480 pixel bằng cách sử dụng phương thức **set().** Đối số đầu tiên 3 là cho chiều rộng và đối số thứ hai 4 là cho chiều cao.

Trong OpenCV, hàm **set()** được sử dụng để thiết lập các thuộc tính của đối tượng video capture, trong đó 3 đại diện cho thuộc tính CV\_CAP\_PROP\_FRAME\_WIDTH (chiều rộng khung hình) và 4 đại diện cho thuộc tính CV\_CAP\_PROP\_FRAME\_HEIGHT (chiều cao khung hình). Đây là mã hiển thị các giá trị của thuộc tính CV\_CAP\_PROP\_\*:

* CV\_CAP\_PROP\_FRAME\_WIDTH: ID của thuộc tính cho chiều rộng khung hình của video.
* CV\_CAP\_PROP\_FRAME\_HEIGHT: ID của thuộc tính cho chiều cao khung hình của video.

A screenshot of a computer code

Description automatically generated

Dòng mã này tạo một đối tượng CascadeClassifier trong OpenCV để phát hiện khuôn mặt trong hình ảnh. Đối số **'haarcascade\_frontalface\_default.xml'** là đường dẫn đến tệp XML chứa các thông số cần thiết cho việc nhận dạng khuôn mặt. Tệp XML này chứa thông tin về cấu trúc của một bộ lọc Haar được sử dụng để phát hiện khuôn mặt.

Bộ lọc Haar là một trong những kỹ thuật phổ biến được sử dụng trong thị giác máy tính để phát hiện các đối tượng trong hình ảnh. Bằng cách sử dụng các bộ lọc Haar và các thông số từ tệp XML, CascadeClassifier có thể phát hiện các đối tượng được đào tạo, như khuôn mặt, trong hình ảnh hoặc video.

Nguồn: <https://hanam88.com/kho-tai-lieu/63/233/phat-hien-khuon-mat-nguoi-trong-anh-su-dung-thu-vien-haar-cascade-voi-opencv-python.html>

A computer screen shot of a program code

Description automatically generated

**face\_id = input(‘\n nhập id người dùng cuối < 1 – n >: ‘):** nhập id tương ứng list\_name lưu trữ trong dataset.

**cam.read():** Đọc một khung hình từ camera được mở trước đó và lưu trữ nó trong biến img. Hàm này trả về một giá trị boolean (ret) cho biết liệu việc đọc khung hình đã thành công hay không.

- **cv2.flip(img, 1):** Lật hình ảnh img theo chiều ngang (trục y). Điều này thường được sử dụng để tránh hình ảnh bị ngược.

- **cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY):** Chuyển đổi hình ảnh img từ không gian màu BGR (Blue-Green-Red) sang không gian màu xám. Hình ảnh xám thường được sử dụng cho việc nhận dạng đối tượng vì nó chỉ chứa thông tin về cường độ sáng, không chứa thông tin màu sắc.

**- face\_detector.detectMultiScale(gray, 1.3, 5):** Sử dụng bộ phân loại cascade (face\_detector) để phát hiện khuôn mặt trong ảnh xám. Phương thức **detectMultiScale()** sẽ trả về một danh sách các hình chữ nhật (x, y, w, h) mô tả vị trí và kích thước của các khuôn mặt được phát hiện trong ảnh.

**x, y, w,** và **h** là các thông số mô tả vị trí và kích thước của đối tượng được phát hiện, thường là khuôn mặt trong trường hợp của bạn.

* **x**: là tọa độ x của góc trái trên của hình chữ nhật giới hạn khuôn mặt.
* **y**: là tọa độ y của góc trái trên của hình chữ nhật giới hạn khuôn mặt.
* **w**: là chiều rộng của hình chữ nhật giới hạn khuôn mặt.
* **h**: là chiều cao của hình chữ nhật giới hạn khuôn mặt.

Vòng lặp for được sử dụng để lặp qua mỗi khuôn mặt được phát hiện trong danh sách **faces**

Hàm **cv2.rectangle()** được sử dụng để vẽ một hình chữ nhật xung quanh khuôn mặt. Các đối số của hàm này là: hình ảnh gốc (img), tọa độ của góc trái trên ((x,y)), tọa độ của góc phải dưới ((x+w,y+h)), màu sắc của hình chữ nhật (trong trường hợp này là màu xanh dương (255,0,0)), và độ dày của đường viền (trong trường hợp này là 2).

Biến **count** được tăng lên 1 sau mỗi lần lặp để đếm số lượng khuôn mặt đã được chụp.

Hàm **cv2.imwrite()** được sử dụng để lưu hình ảnh của khuôn mặt vào thư mục **dataset**. Tên tệp được tạo ra dựa trên số thứ tự của khuôn mặt và **face\_id.**

Cuối cùng, hàm **cv2.imshow()** được sử dụng để hiển thị hình ảnh được chụp trong cửa sổ với tiêu đề "**camera dataser**".

**cv2.waitKey(100):** Chờ 100ms để xem nếu có sự kiện từ bàn phím. Nếu không có, nó sẽ trả về -1. Nếu có, nó sẽ trả về mã ASCII của phím được nhấn.

**k = cv2.waitKey(100) & 0xff:** Lấy các byte cuối cùng từ giá trị trả về (sử dụng & 0xff) để đảm bảo rằng giá trị là trong khoảng từ 0 đến 255.

**if k == 27:** Nếu phím ESC được nhấn (mã ASCII là 27), chương trình sẽ thoát khỏi vòng lặp while.

**elif count >= 30:** Nếu số lượng khuôn mặt đã chụp (count) đạt hoặc vượt quá 30, chương trình cũng sẽ thoát khỏi vòng lặp while.

A black screen with text

Description automatically generated

**cam.release():** Đóng kết nối với camera, giải phóng bộ nhớ và tài nguyên hệ thống mà camera đang sử dụng. Điều này giúp giải phóng tài nguyên và ngừng truy cập vào camera.

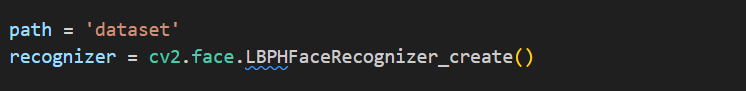
**cv2.destroyAllWindows():** Đóng tất cả các cửa sổ hiển thị bởi OpenCV. Điều này giúp giải phóng tài nguyên hệ thống và kết thúc chương trình một cách sạch sẽ.

* 1. **Bước 2\_Face\_Tranning.py**

**A black screen with white text

Description automatically generated**

Import cv2, numpy, PIL và os vào file.



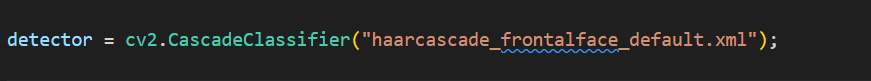
**path = 'dataset'**: Đường dẫn tới thư mục chứa dữ liệu huấn luyện. Trong trường hợp này, giả sử rằng các hình ảnh của các khuôn mặt đã được chụp và được lưu trữ trong thư mục có tên 'dataset'.

**recognizer = cv2.face.LBPHFaceRecognizer\_create():** Khởi tạo một đối tượng nhận dạng khuôn mặt sử dụng thuật toán LBPH. Đối tượng này sẽ được sử dụng để huấn luyện trên các hình ảnh của các khuôn mặt trong thư mục 'dataset' và sau đó sẽ được sử dụng để nhận dạng khuôn mặt trong thời gian thực.

**- Thuật toán LBPH (Local Binary Patterns Histograms)** là một trong những thuật toán phổ biến trong lĩnh vực nhận dạng khuôn mặt. Nó hoạt động bằng cách xác định các mẫu nhị phân cục bộ trong ảnh và tạo ra các histogram đặc trưng cho mỗi khuôn mặt. Sau đó, những histogram này được sử dụng để huấn luyện một mô hình nhận dạng khuôn mặt.

- Sau khi tạo đối tượng nhận dạng khuôn mặt bằng LBPH, bạn có thể sử dụng nó để huấn luyện trên dữ liệu ảnh khuôn mặt và sau đó sử dụng mô hình đã huấn luyện để nhận dạng khuôn mặt trong thời gian thực.

Nguồn: <https://websitehcm.com/face-recognition-va-face-detection-trong-opencv/>



**detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml");**

Tạo một đối tượng bộ phân loại Cascade để nhận diện khuôn mặt. Tệp XML **"haarcascade\_frontalface\_default.xml"**

chứa các thông số cần thiết để phát hiện khuôn mặt trong hình ảnh.

Bộ phân loại cascade là một phương pháp phổ biến được sử dụng để phát hiện các đối tượng trong hình ảnh. Nó hoạt động bằng cách áp dụng một loạt các bộ lọc đặc biệt lên hình ảnh và kiểm tra xem có mặt các đặc điểm đặc trưng của đối tượng cần phát hiện không.

Sau khi tạo đối tượng CascadeClassifier và cấu hình nó để phát hiện khuôn mặt, bạn có thể sử dụng nó để phát hiện các khuôn mặt trong hình ảnh hoặc video bằng cách gọi phương thức **detectMultiScale()** trên hình ảnh đó.

A computer screen shot of a program code

Description automatically generated

* **os.listdir(path):** Trả về một danh sách chứa tất cả các tên tệp trong thư mục được chỉ định bởi biến path.

**- [os.path.join(path,f) for f in os.listdir(path)]:** Dùng một list comprehension để tạo một danh sách mới với đường dẫn đầy đủ của mỗi tệp ảnh trong thư mục. Hàm **os.path.join()** được sử dụng để kết hợp đường dẫn của thư mục với tên tệp ảnh.

**- faceSamples=[]:** Một danh sách được khởi tạo để lưu trữ các mẫu khuôn mặt từ các tệp ảnh.

**- ids = []:** Một danh sách khác được khởi tạo để lưu trữ các ID tương ứng với mỗi khuôn mặt. ID có thể là số duy nhất hoặc nhãn đại diện cho mỗi người.

**- Image.open(imagePath):** Mở tệp ảnh tại đường dẫn imagePath bằng thư viện PIL (Python Imaging Library).

**- .convert('L'):** Chuyển đổi ảnh sang chế độ ảnh xám. Điều này làm giảm kích thước dữ liệu và đơn giản hóa quá trình xử lý.

**- PIL\_img:** Là biến chứa ảnh sau khi đã được mở và chuyển đổi.

**- p.array(PIL\_img, 'uint8'):** Chuyển đổi ảnh từ đối tượng PIL thành một mảng NumPy, sử dụng kiểu dữ liệu unsigned integer 8-bit ('uint8'). Điều này chuyển đổi ảnh từ định dạng PIL sang một mảng NumPy có thể được sử dụng cho việc xử lý và huấn luyện mô hình nhận dạng khuôn mặt.

**id = int(os.path.split(imagePath)[-1].split(".")[1])**

**os.path.split(imagePath)[-1] được** sử dụng để lấy phần cuối cùng của đường dẫn tệp ảnh, nghĩa là tên tệp ảnh.

**os.path.split(imagePath):** Phân tách đường dẫn của tệp ảnh thành một tuple gồm hai phần: đường dẫn thư mục và tên tệp.

[-1]: Chọn phần tử cuối cùng của tuple, nghĩa là tên tệp ảnh.

Ví dụ: nếu imagePath là dataset/User.1.1.jpg, thì os.path.split(imagePath)[-1] sẽ trả về 'User.1.1.jpg'. Điều này giúp lấy tên của tệp ảnh từ đường dẫn đầy đủ.

**- .split("."):** Phân tách tên tệp ảnh bằng dấu chấm, tạo ra một danh sách các phần của tên tệp.

- [1]: Chọn phần thứ hai của danh sách, chứa ID.

Giả sử tên tệp ảnh là 'User.1.1.jpg':

Sau khi được phân tách bằng dấu chấm, chúng ta có một danh sách gồm ['User', '1', '1', 'jpg'].

[1] sẽ chọn phần tử thứ hai trong danh sách, nghĩa là '1'.

**- etector.detectMultiScale(img\_numpy**): Sử dụng bộ phân loại cascade đã được tạo (detector) để phát hiện các khuôn mặt trong ảnh (img\_numpy). Kết quả là một danh sách các hình chữ nhật (x, y, w, h) mô tả vị trí và kích thước của các khuôn mặt được phát hiện trong ảnh.

- Cuối cùng, biến id chứa ID của người trong ảnh, và biến faces chứa danh sách các khuôn mặt được phát hiện trong ảnh, mỗi khuôn mặt được mô tả bằng một hình chữ nhật (x, y, w, h).

- Với mỗi hình chữ nhật mô tả khuôn mặt trong danh sách faces, đoạn mã này cắt ảnh từ img\_numpy theo vị trí và kích thước của khuôn mặt, sau đó thêm ảnh đã cắt và ID tương ứng vào faceSamples và ids.

**- img\_numpy[y:y+h, x:x+w]:** Đây là cách cắt ảnh từ mảng **NumPy img\_numpy**. Đoạn mã này chọn phần của ảnh bắt đầu từ tọa độ (x, y) và kết thúc ở tọa độ (x+w, y+h). Điều này tạo ra một phần của ảnh chứa khuôn mặt, với chiều rộng là w và chiều cao là h.

Cuối cùng, đoạn mã trả về faceSamples và ids, chứa danh sách các mẫu khuôn mặt và các ID tương ứng của những người trong ảnh.

**- faceSamples.append(...):** Đoạn mã này thêm phần của ảnh chứa khuôn mặt vào danh sách faceSamples, để sau này có thể được sử dụng cho việc huấn luyện mô hình nhận dạng khuôn mặt.

Kết quả là faceSamples sẽ chứa danh sách các mẫu khuôn mặt được cắt ra từ ảnh gốc, mỗi mẫu khuôn mặt là một phần của ảnh chứa khuôn mặt đó.

A screen shot of a computer

Description automatically generated

**getImagesAndLabels(path):** Gọi hàm **getImagesAndLabels()** để lấy danh sách các mẫu khuôn mặt và các ID tương ứng từ thư mục path.

**recognizer.train(faces, np.array(ids)):** Sử dụng phương thức train() của mô hình nhận dạng khuôn mặt (recognizer) để huấn luyện mô hình với dữ liệu huấn luyện. Đối số faces là danh sách các mẫu khuôn mặt, và **np.array(ids)** là danh sách các ID tương ứng.

Trong đó :

* **faces**: Danh sách các mẫu khuôn mặt, mỗi mẫu là một phần của ảnh chứa khuôn mặt.
* **np.array(ids):** Danh sách các ID tương ứng với mỗi mẫu khuôn mặt. Biến này được chuyển đổi thành một mảng NumPy để có thể sử dụng trong quá trình huấn luyện

Dòng mã này ghi mô hình nhận dạng khuôn mặt đã được huấn luyện vào một tệp YAML có tên là 'trainer.yml'.

* 1. **Bước 3\_Face\_recognition.py**

A black background with a white circle

Description automatically generated with medium confidence

Import thư viện cv2, numpy và os vào file.

A computer screen shot of text

Description automatically generated- Tạo một đối tượng nhận dạng khuôn mặt sử dụng thuật toán **LBPH (Local Binary Patterns Histograms)** thông qua phương thức **cv2.face.LBPHFaceRecognizer\_create().** Sau đó, nó đọc mô hình đã được huấn luyện từ tệp **YAML 'trainer/trainer.yml'** bằng phương thức read().

Khi mô hình được đọc vào bộ nhớ, bạn có thể sử dụng nó để nhận dạng khuôn mặt trong các hình ảnh hoặc video mới. Mô hình đã được huấn luyện sẽ chứa thông tin cần thiết để nhận dạng các khuôn mặt dựa trên các đặc trưng đã học từ dữ liệu huấn luyện trước đó.

Sau đó, bạn có thể sử dụng đối tượng recognizer để thực hiện nhận dạng khuôn mặt trên các ảnh mới hoặc trong thời gian thực, tùy thuộc vào nhu cầu của ứng dụng của bạn.

- Tạo một đối tượng bộ phân loại cascade để phát hiện khuôn mặt trong hình ảnh. Đối tượng được tạo thông qua phương thức **cv2.CascadeClassifier(),** và nó được cấu hình để sử dụng tệp XML **"haarcascade\_frontalface\_default.xml"** như một phần của bộ phân loại.

**cascadePath**: Đường dẫn đến tệp XML chứa thông tin về bộ phân loại cascade được sử dụng để phát hiện khuôn mặt. Trong trường hợp này, đó là **"haarcascade\_frontalface\_default.xml".**

**faceCascade**: Đối tượng bộ phân loại cascade được tạo ra và sẽ được sử dụng để phát hiện khuôn mặt trong hình ảnh.

Sau khi đối tượng **faceCascade** được tạo, bạn có thể sử dụng nó để phát hiện khuôn mặt trong các hình ảnh hoặc video bằng cách gọi phương thức **detectMultiScale().**

A black screen with text on it

Description automatically generated

**font**: Loại phông chữ được sử dụng để hiển thị tên người được nhận dạng lên hình ảnh.

**id**: ID của người được nhận dạng. Trong ví dụ này, được khởi tạo là 0, nhưng có thể thay đổi tùy thuộc vào việc sử dụng.

**names**: Danh sách tên của các người được nhận dạng.

**cam**: Đối tượng VideoCapture để truy cập dữ liệu từ máy ảnh.

**cam.set(3, 640) và cam.set(4, 480):** Thiết lập kích thước khung hình cho máy ảnh. Ở đây, kích thước được đặt là 640x480 pixels.

**minW và minH:** Kích thước tối thiểu của khuôn mặt được chấp nhận. Các kích thước này được tính toán dựa trên kích thước của khung hình từ máy ảnh

A computer screen shot of a code

Description automatically generated **- ret, img = cam.read():** Đọc một khung hình từ máy ảnh và gán nó cho biến img. Biến ret chứa giá trị True nếu việc đọc khung hình thành công, ngược lại là False.

**- gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY):** Chuyển đổi ảnh màu sang ảnh xám. Việc này giúp giảm chi phí tính toán khi xử lý ảnh, đồng thời cũng là bước tiền xử lý quan trọng trước khi nhận dạng khuôn mặt.

**img**: Là ảnh màu ban đầu được đọc từ máy ảnh.

**cv2.COLOR\_BGR2GRAY:** Đây là một hằng số của OpenCV, chỉ định rằng chúng ta muốn chuyển đổi ảnh từ không gian màu BGR (màu mặc định cho ảnh màu trong OpenCV) sang ảnh xám.

Kết quả là biến gray sẽ chứa ảnh xám tương ứng với ảnh màu img

-scaleFactor: Tham số này xác định tỷ lệ thu phóng của ảnh trong quá trình phát hiện khuôn mặt. Giá trị lớn hơn của scaleFactor có nghĩa là việc phát hiện khuôn mặt sẽ diễn ra nhanh hơn, nhưng có thể dẫn đến việc bỏ sót các khuôn mặt nhỏ hoặc đặt sai vị trí của khuôn mặt. Trong trường hợp này, giá trị 1.2 được sử dụng.

Khi scaleFactor được đặt thành một giá trị lớn hơn 1, nó chỉ định tỷ lệ tăng kích thước của cửa sổ phát hiện mỗi lần nó được chuyển đổi qua các tầng của hình ảnh. Một giá trị lớn hơn 1 sẽ làm tăng kích thước cửa sổ phát hiện, dẫn đến việc kiểm tra khu vực lớn hơn của hình ảnh mỗi lần. Điều này có thể giúp tăng tốc độ của thuật toán nhưng cũng có thể dẫn đến việc bỏ sót các khuôn mặt nhỏ hoặc bị che khuất.

Khi scaleFactor được đặt thành một giá trị trong khoảng từ 0 đến 1, nó chỉ định tỷ lệ giảm kích thước của cửa sổ phát hiện mỗi lần nó được chuyển đổi. Một giá trị trong khoảng này sẽ làm giảm kích thước cửa sổ phát hiện, dẫn đến việc kiểm tra khu vực nhỏ hơn của hình ảnh mỗi lần. Điều này có thể làm tăng độ chính xác của việc phát hiện nhưng cũng có thể làm chậm quá trình phát hiện.

Ví dụ, nếu scaleFactor được đặt thành 1.2, cửa sổ phát hiện sẽ được tăng kích thước 20% mỗi lần nó được chuyển đổi qua các tầng của hình ảnh. Điều này có thể làm tăng tốc độ của thuật toán nhưng cũng có thể dẫn đến việc bỏ sót các khuôn mặt nhỏ hoặc bị che khuất.

minNeighbors: Tham số này xác định số lượng neighbors cần phát hiện trong cửa sổ phát hiện trước khi kết luận rằng một khuôn mặt đã được tìm thấy. Giá trị cao hơn của minNeighbors có thể giúp giảm thiểu các phát hiện giả mạo, nhưng cũng có thể làm giảm độ chính xác của kết quả phát hiện. Trong trường hợp này, giá trị 5 được sử dụng.

minSize: Kích thước tối thiểu của khuôn mặt được phát hiện. Chỉ có các khuôn mặt có kích thước lớn hơn giá trị minSize mới được coi là hợp lệ. Trong trường hợp này, kích thước tối thiểu được tính dựa trên kích thước của khung hình từ máy ảnh, với giá trị 10% của chiều rộng và chiều cao tương ứng của khung hình.

**- cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2):** Hàm **cv2.rectangle()** được sử dụng để vẽ một hộp giới hạn trên hình ảnh img. Các tham số của hàm như sau:

**img**: Hình ảnh đích mà chúng ta muốn vẽ hộp giới hạn lên.

**(x,y):** Tọa độ của góc trái trên của hộp giới hạn. Đây là tọa độ của điểm góc trái trên của hộp.

**(x+w,y+h):** Tọa độ của góc phải dưới của hộp giới hạn. Đây là tọa độ của điểm góc phải dưới của hộp.

**(0,255,0):** Màu sắc của hộp giới hạn. Trong trường hợp này, là màu xanh lá cây. Định dạng màu là (B, G, R), với B là mức độ màu xanh, G là mức độ màu lục, và R là mức độ màu đỏ.

2: Độ dày của đường viền của hộp giới hạn. Trong trường hợp này, là 2 pixels.

Kết quả là một hộp giới hạn màu xanh lá cây sẽ được vẽ xung quanh khuôn mặt được phát hiện, với độ dày của đường viền là 2 pixels.

**gray[y:y+h,x:x+w]:** Phần của ảnh xám (gray) chứa khuôn mặt được xác định bởi các tọa độ (x, y, w, h) của hộp giới hạn. Đây là phần ảnh mà chúng ta muốn mô hình nhận dạng và dự đoán ID của khuôn mặt đó.

**recognizer.predict():** Phương thức này của mô hình nhận dạng sẽ dự đoán ID của khuôn mặt dựa trên phần của ảnh đã chọn. Kết quả của dự đoán sẽ trả về hai giá trị: id là ID được dự đoán và confidence là mức độ tự tin của dự đoán. Đối với thuật toán LBPH, confidence là sai số giữa khuôn mặt đã được nhận dạng và mẫu khuôn mặt trong tập dữ liệu huấn luyện. Điều này có nghĩa là mức độ tự tin càng cao thì kết quả dự đoán càng chính xác.

Phương thức **predict()** này được sử dụng để dự đoán ID và tính toán mức độ tự tin của dự đoán, sau đó có thể sử dụng thông tin này để hiển thị kết quả hoặc thực hiện các hành động phù hợp khác trong ứng dụng của bạn.

Giá trị của confidence thường là một số dương, và càng gần 0 thì mô hình càng tự tin vào dự đoán của mình. Nếu confidence gần 0, điều này có nghĩa là mô hình rất tự tin rằng dự đoán của nó là chính xác.

Tuy nhiên, nếu confidence lớn, tức là xa 0, điều này có thể ngụ ý rằng mô hình không chắc chắn về dự đoán của mình. Trong trường hợp này, dự đoán có thể không chính xác hoặc có thể cần được kiểm tra lại.

**cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2):** Vẽ văn bản str(id) (ID của người được nhận dạng) lên hình ảnh img. x+5 và y-5 là tọa độ bắt đầu của văn bản, được dịch chuyển một chút để nằm bên phải và trên của hộp giới hạn. font là font chữ được sử dụng (có thể là một trong các hằng số được xác định trước trong OpenCV). 1 là kích thước của font, và (255,255,255) là màu của văn bản (trong trường hợp này, là màu trắng). 2 là độ dày của văn bản.

**cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1):** Tương tự như trên, nhưng với văn bản str(confidence) (mức độ tự tin của việc nhận dạng khuôn mặt) được hiển thị. (x+5, y+h-5) là tọa độ bắt đầu của văn bản, được dịch chuyển một chút để nằm bên phải và dưới của hộp giới hạn. (255,255,0) là màu của văn bản (trong trường hợp này, là màu vàng). 1 là kích thước của font, và 1 là độ dày của văn bản.

**cv2.waitKey(10):** Hàm này chờ đợi một sự kiện phím nhấn trong một khoảng thời gian nhất định (trong trường hợp này, 10 miligiây). Nó trả về mã ASCII của phím được nhấn.

**k & 0xff:** Đảm bảo rằng chỉ có 8 bit thấp nhất của k được giữ lại, vì trong một số trường hợp, waitKey() có thể trả về giá trị lớn hơn 255.

**if k == 27:** Kiểm tra xem liệu phím được nhấn có phải là phím Esc hay không (với mã ASCII là 27). Nếu phải, chương trình sẽ thoát khỏi vòng lặp while và kết thúc.

A black screen with orange and blue text

Description automatically generated

**am.release():** Giải phóng tài nguyên của thiết bị camera, đảm bảo rằng không có tiến trình nào đang sử dụng camera sau khi chương trình kết thúc.

**cv2.destroyAllWindows():** Đóng tất cả các cửa sổ hình ảnh được tạo ra bởi OpenCV. Điều này đảm bảo rằng tất cả các cửa sổ sẽ được đóng khi chương trình kết thúc, ngăn cản bất kỳ cửa sổ nào vẫn còn mở sau khi chương trình đã kết thúc.

* 1. **Trainer**

Radius: Nó đại diện cho bán kính xung quanh pixel trung tâm. Nó thường được đặt thành 1. Nó được sử dụng để xây dựng mẫu nhị phân cục bộ tròn.

Neighbors: Số lượng điểm mẫu để xây dựng mẫu nhị phân hình tròn.

Grid X: Số ô theo hướng ngang. Càng biểu thị nhiều ô và lưới mịn hơn, thì kích thướccủa vectơ đặc trưng thu được càng cao.

Grid Y: Số ô theo hướng dọc. Càng biểu thị nhiều ô và lưới mịn hơn, thì kích thước của vectơ đặc trưng thu được càng cao.

Trong cấu trúc YAML cung cấp, phần data chứa một dãy số lớn, dường như là một phần của ma trận histogram. Đây có thể là dữ liệu số đã được mã hóa để biểu diễn ma trận histogram của các khuôn mặt đã được huấn luyện.

Đối với mô hình nhận dạng khuôn mặt sử dụng thuật toán LBPH (Local Binary Patterns Histograms), histogram là một phần quan trọng để mô tả đặc trưng của mỗi khuôn mặt. Các giá trị trong histogram biểu thị tần suất xuất hiện của các đặc trưng cụ thể được sử dụng để nhận dạng khuôn mặt.

Dãy số lớn trong phần data có thể là một phần của ma trận histogram, trong đó mỗi số có thể biểu thị giá trị tần suất xuất hiện của một đặc trưng cụ thể trong histogram.

Nguồn: https://websitehcm.com/face-recognition-va-face-detection-trong-opencv/

* 1. **haarcascade\_frontalface\_default**

Tệp haarcascade\_frontalface\_default.xml là một tập hợp các cấu trúc dữ liệu được sử dụng trong thuật toán nhận diện khuôn mặt của OpenCV, cụ thể là trong phần cài đặt của thuật toán phát hiện khuôn mặt sử dụng bộ phân loại cascade.

Trong thuật toán này, "cascade" thường đề cập đến việc sử dụng một chuỗi các bộ lọc để phát hiện các đối tượng trong ảnh. Bộ lọc này được sử dụng để phát hiện các đặc điểm cụ thể của khuôn mặt, như mắt, mũi, miệng, và từ đó xác định vị trí của khuôn mặt trong ảnh.

Tệp haarcascade\_frontalface\_default.xml cụ thể chứa thông tin về các tính năng cụ thể của khuôn mặt được sử dụng để phát hiện khuôn mặt trong các bức ảnh. Các tính năng này được mô tả dưới dạng các "cascade" của bộ lọc, được sử dụng để quét qua ảnh để tìm ra các vùng có khả năng là khuôn mặt.

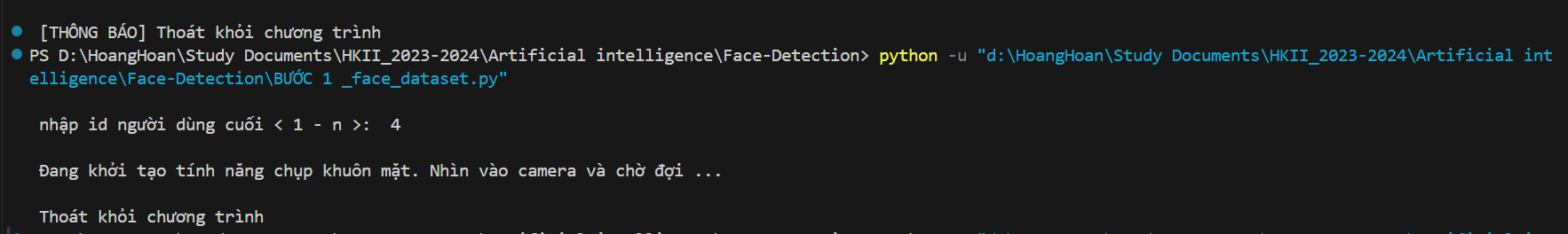
Tệp XML này thường được sử dụng trong quá trình huấn luyện mô hình hoặc trong các ứng dụng thực tế để phát hiện khuôn mặt trong ảnh hoặc video.

Khi bạn sử dụng tệp haarcascade\_frontalface\_default.xml cùng với OpenCV, bạn có thể phát hiện khuôn mặt trong ảnh hoặc video một cách tự động và hiệu quả.

1. **Hoạt động của ứng dụng**

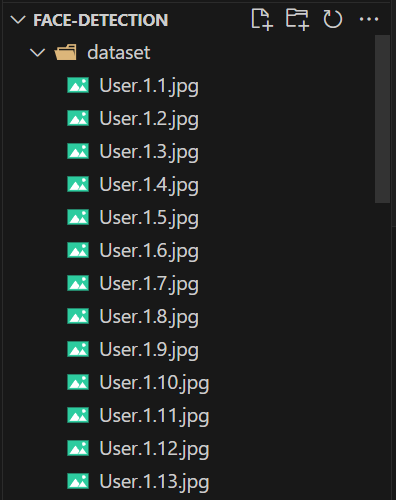
Đầu tiên, chúng ta chạy file Bước Bước 1\_Face\_Dataset để lưu ảnh khuôn mặt của người dùng vào dataset.

Mình sẽ nhập id người dùng tương ứng với listname trong file Bước 3\_Face\_Recognition

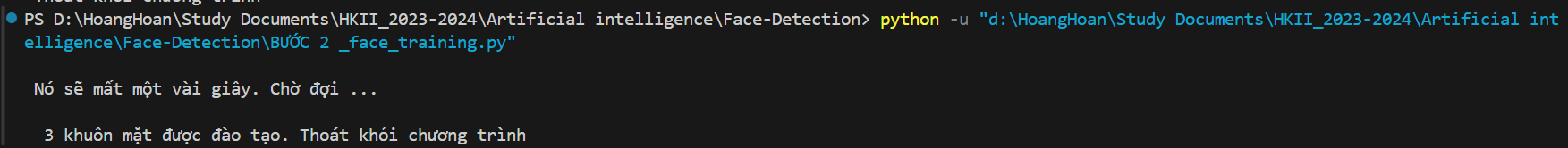


Trình khởi tạo sẽ chụp 50 ảnh khuôn mặt của người dùng để trainning nhận diện khuôn mặt.

Sau khi khởi tạo xong, hệ thống sẽ thoát khỏi chương trình.

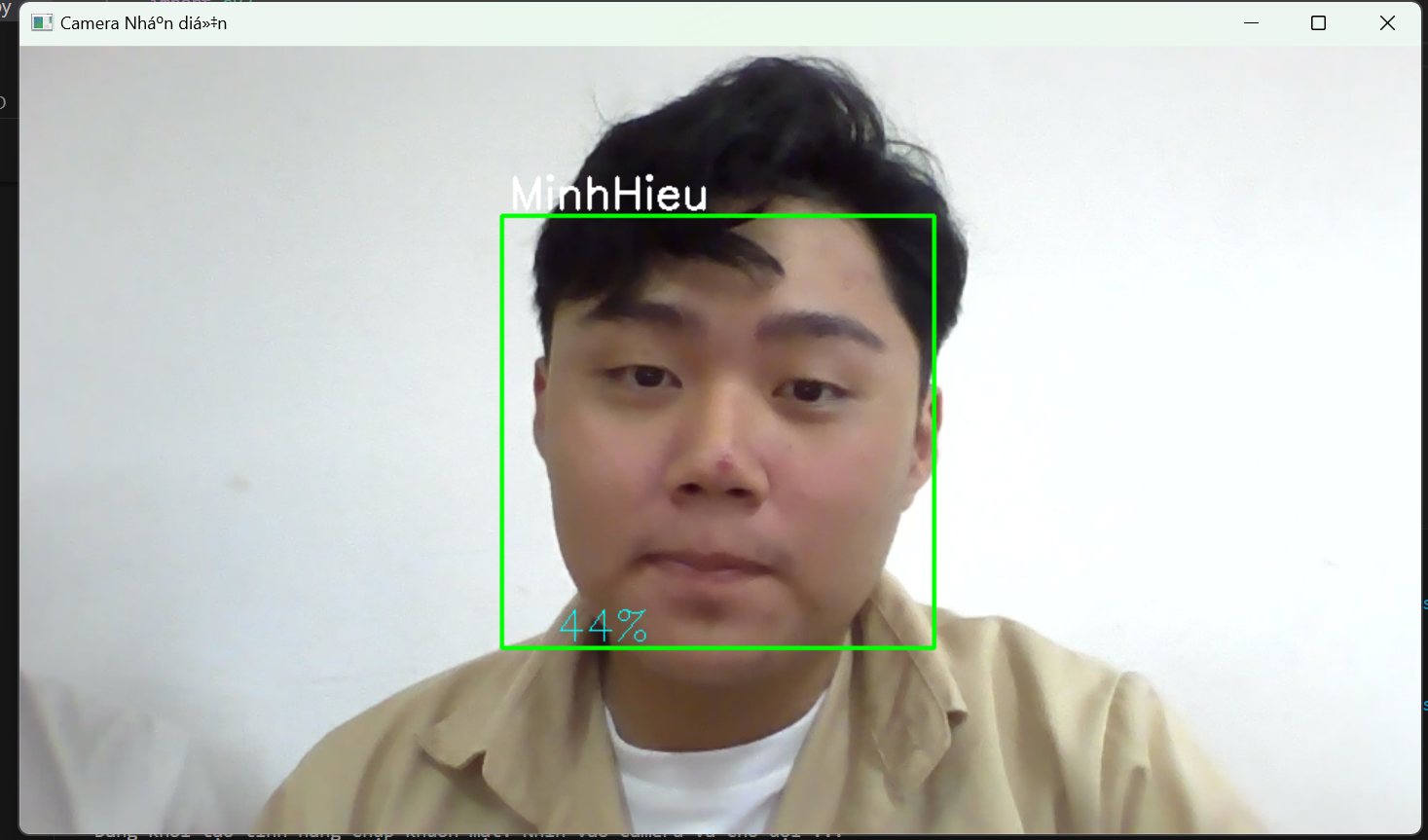


Sau đó, chúng ta sẽ chạy file Bước 2\_Face\_Tranning. Khi này, các khuôn mặt vừa được chụp sẽ được đào tạo nhận diện khuôn mặt cho AI. Sau khi đào tạo xong sẽ thoát chương trình.

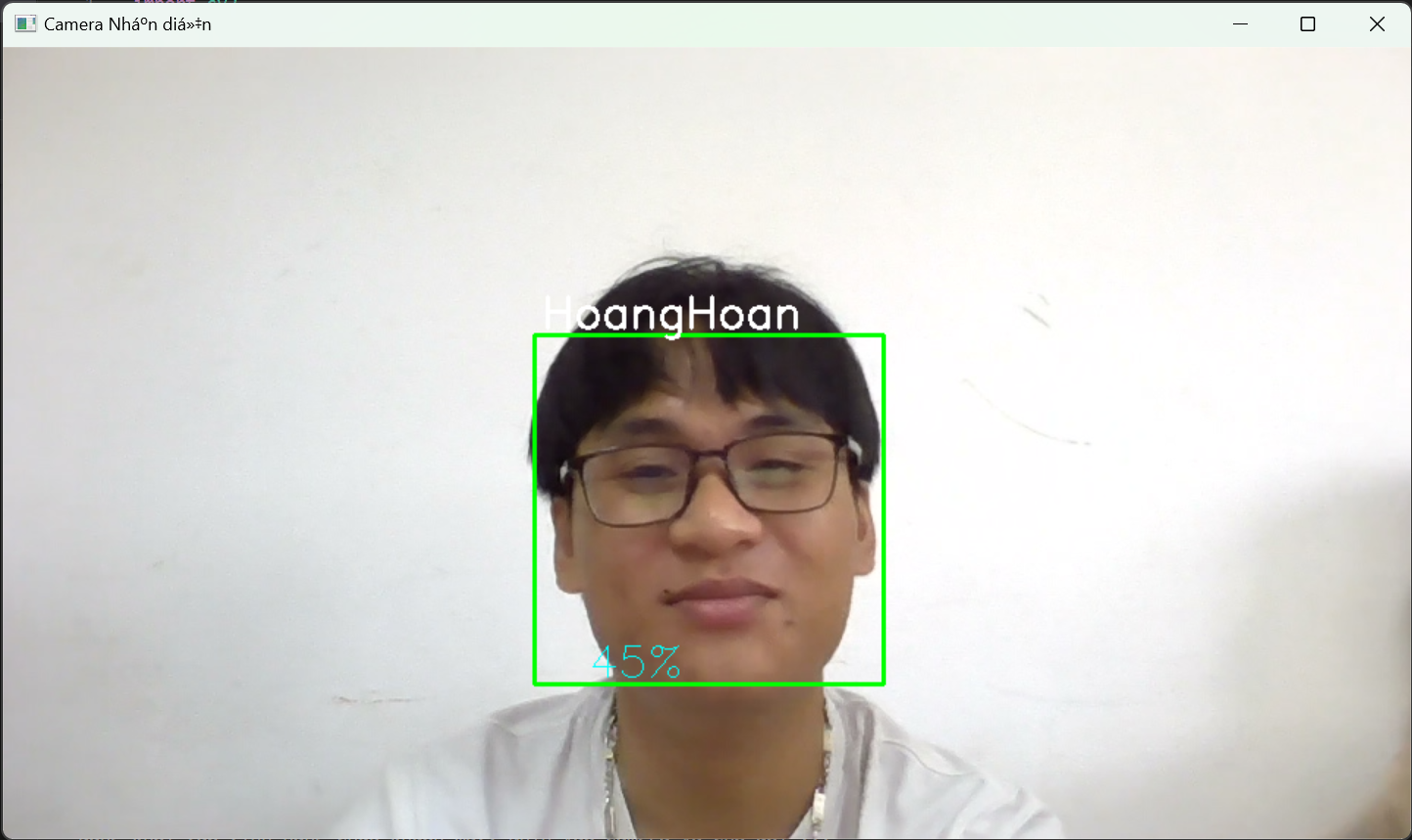


Sau đó, chúng ta sẽ chạy file Bước 3\_Face\_Recognition. Khi này máy sẽ hiện lên một cửa sổ camera để nhận diện khuôn mặt. Yêu cầu người dùng nhìn thẳng vào camera để nhận diện khuôn mặt.

Nhận diện khuôn mặt đầu tiên với id = 1, name = ‘MinhHieu’



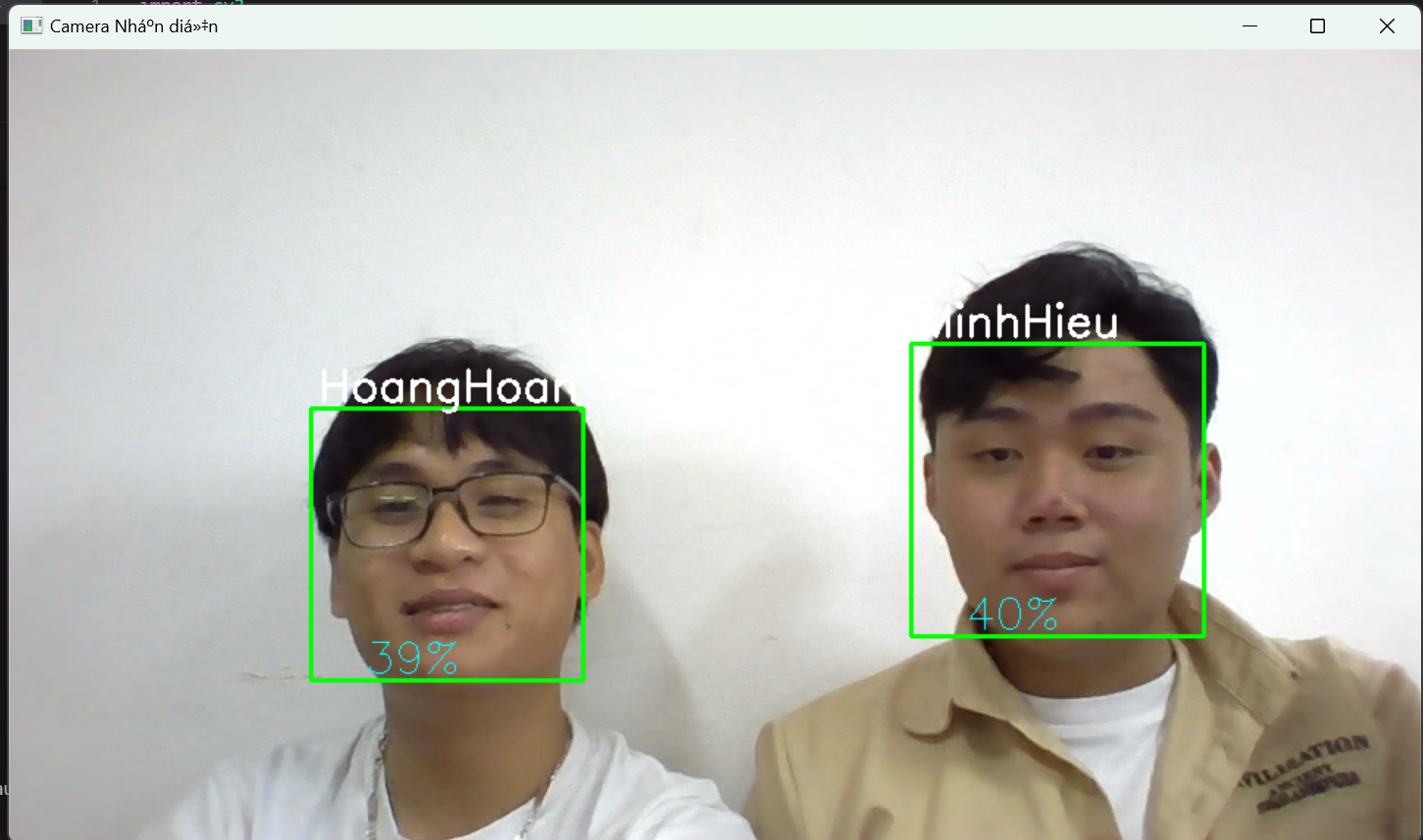
Nhận diện khuôn mặt đầu tiên với id = 2, name = ‘HoangHoan’



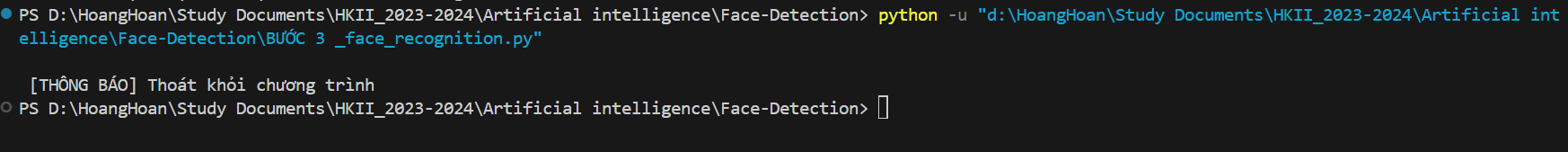
Nhận diện khuôn mặt đầu tiên với id = 4, name = ‘VanHieu’



Hệ thống có thể nhận diện 2 khuôn mặt với id = 1 và id =2, name = ‘MinhHieu’ và name = ‘HoangHoan’



Sau khi chạy xong, mình nhấn phím Esc để thoát chương trình.

****

1. **Hoàn thiện và hướng phát triển ứng dụng**

Sau khi thực thi và hoàn thành ứng dụng nhận diện khuôn mặt, nhóm chúng em đã rút ra được một số kết luận và tổng hợp hướng phát triển tương lai cho ứng dụng như sau:

**Hướng phát triển:**

* **Kiểm thử và sửa lỗi:**
* Tiếp tục kiểm thử để xác định và sửa chữa các lỗi, đảm bảo tính ổn định của ứng dụng.
* **Phát triển ứng dụng:**
* Nâng cao tính bảo mật, nhận diện khuôn mặt với độ chính xác cao.

Trong quá trình thực hiện làm ứng dụng không tránh khỏi những thiếu sót và hạn do thiếu kinh nghiệm và kiến thức, chúng em kinh mong nhận được những ý kiến đóng góp của thầy để đồ án của chúng em được hoàn thiện hơn.

**PHẦN 4: KẾT LUẬN**

Sau khi nghiên cứu và xây dựng ứng dụng nhận diện khuôn mặt bằng ngôn ngữ Python với thư viện opencv-python và thuật toán LBPH, chúng em xin được tổng kết một số điều như sau:

* **Kết quả đạt được:**

+ Thiết kế được một AI đơn giản nhận diện khuôn mặt dựa trên ngôn ngữ Python và các thư viện Opencv-python và thuật toán LBPH.

+ Sử dụng thuật toán và các thư viện có sẵn của Python giúp cho chương trình chạy nhanh và hiệu quả hơn.

+ Chương trình thực thi thành công và nhận diện được khuôn mặt của người dùng.

* **Hạn chế:**

+ Thuật toán cần phải tải thư viện opencv-python, opencv có nhiều phiên bản, có một số phiên bản mới cập nhật không hỗ trợ cv2.face.

+ Do kiến thức lập trình còn hạn chế hầu hết chương trình đều tham khảo dựa trên chương trình có sẵn và các video ở trên nền tảng Youtube.

* **Kết luận:**

Chúng em còn nhiều thiếu sót trong đề tài cũng như trong quá trình thực hiện đề tài. Từ đó chúng em mong nhận được nhiều ý kiến cũng như đóng góp từ thầy để có thể hoàn thiện đề tài và bổ sung thêm kiến thức để có thể hoàn thành tốt các bài báo cáo sau.

Chúng em xin chân thành cảm ơn!

**Tài liệu tham khảo**

# Face Recognition và face Detection trong OpenCV

<https://websitehcm.com/face-recognition-va-face-detection-trong-opencv/>

1. Phát hiện khuôn mặt người trong ảnh sử dụng thư viện Haar Cascade với OpenCV-Python

<https://hanam88.com/kho-tai-lieu/63/233/phat-hien-khuon-mat-nguoi-trong-anh-su-dung-thu-vien-haar-cascade-voi-opencv-python.html>

1. ChatGPT

<https://chat.openai.com/>

# Hướng dẫn sử dụng thư viện PILLOW để xử lý hình ảnh trong Python cho người mới bắt đầu.

<https://viblo.asia/p/huong-dan-su-dung-thu-vien-pillow-de-xu-ly-hinh-anh-trong-python-cho-nguoi-moi-bat-dau-3Q75wm4MZWb>

1. Giáo trình trí tuệ nhân tạo của giảng viên Huỳnh Minh Trí