**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**

**Đề tài số 20: Xây dựng hệ thống chấm công nhận diện khuôn mặt dựa trên open cv**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20210339 | Nguyễn Hải Long | DCCNTT12.10.1 |
| **2** | 20210139 | Nông Minh Hoài | DCCNTT12.10.1 |
| **3** | 20210147 | Vũ Đình Danh | DCCNTT12.10.1 |
| **4** | 20210254 | Lò Quang Trung | DCCNTT12.10.1 |

**Bắc Ninh, năm 2024**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÀI TẬP LỚN**

**HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**

**Đề tài số 20: Xây dựng hệ thống chấm công nhận diện khuôn mặt dựa trên open cv**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20210339 | Nguyễn Hải Long | DCCNTT12.10.1 |
| **2** | 20210139 | Nông Minh Hoài | DCCNTT12.10.1 |
| **3** | 20210147 | Vũ Đình Danh | DCCNTT12.10.1 |
| **4** | 20210254 | Lò Quang Trung | DCCNTT12.10.1 |

|  |  |
| --- | --- |
|  |  |

**Bắc Ninh, năm 2024**

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á  **KHOA CÔNG NGHỆ THÔNG TIN** | **KỲ THI KẾT THÚC HỌC PHẦN**  **HỌC KỲ 1, NĂM HỌC 2024** – **2025** |

|  |  |
| --- | --- |
| **PHIẾU CHẤM THI BÀI TẬP LỚN KẾT THÚC HỌC PHẦN**  **Mã đề thi: 20**  **Tên học phần: Xử lý ảnh và thị giác máy tính**  **Lớp Tín chỉ:**  **XATGMT.03.K12.01.LH.C04.1** | |
| **Cán bộ chấm thi 1**  *(Ký và ghi rõ họ tên)*  Lương Thị Hồng Lan | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ tên)* |

| **TT** | **TIÊU CHÍ** | **THANG ĐIỂM** | **Nguyễn Hải Long** | **Nông Minh Hoài** | **Vũ Đình Danh** | **Lò Quang Trung** |
| --- | --- | --- | --- | --- | --- | --- |
| 20210339 | 20210139 | 20210147 | 20210254 |
| **1** | **Nội dung báo cáo trên Word đầy đủ** | **3.5** |  |  |  |  |
| 1.1 | Có bố cục rõ ràng (mục lục, phần mở đầu, nội dung chính, kết luận). | 0,5 |  |  |  |  |
| 1.2 | Nội dung phân tích rõ ràng, logic. | 0,5 |  |  |  |  |
| 1.3 | Có dẫn chứng, số liệu minh họa đầy đủ. | 0,5 |  |  |  |  |
| 1.4 | Ngôn ngữ và trình bày chuẩn, không lỗi chính tả. | 0,5 |  |  |  |  |
| 1.5 | Có trích dẫn tài liệu tham khảo đúng quy cách. | 0,5 |  |  |  |  |
| 1.6 | Được trình bày chuyên nghiệp (canh lề, font chữ, khoảng cách dòng hợp lý). | 0,5 |  |  |  |  |
| 1.7 | Tài liệu đầy đủ, bám sát yêu cầu của đề bài. | 0,5 |  |  |  |  |
| **2** | **Nội dung thuyết trình đầy đủ** | **1.0** |  |  |  |  |
| 2.1 | Trình bày tự tin, phát âm rõ ràng, mạch lạc. | 0,5 |  |  |  |  |
| 2.2 | Nội dung thuyết trình đúng trọng tâm, không lan man. | 0,5 |  |  |  |  |
| **3** | **Slides báo cáo đầy đủ nội dung + Hỏi đáp** | **3.0** |  |  |  |  |
| 3.1 | Slides có bố cục rõ ràng (mở đầu, nội dung, kết luận). | 0,5 |  |  |  |  |
| 3.2 | Thiết kế slides đẹp, chuyên nghiệp (màu sắc, hình ảnh minh họa). | 0,5 |  |  |  |  |
| 3.3 | Nội dung trên slides ngắn gọn, dễ hiểu, súc tích. | 0,5 |  |  |  |  |
| 3.4 | Nội dung slides phù hợp với nội dung báo cáo. | 0,5 |  |  |  |  |
| 3.5 | Trả lời câu hỏi đầy đủ, chính xác. | 0,5 |  |  |  |  |
| 3.6 | Trả lời câu hỏi tự tin, thuyết phục. | 0,5 |  |  |  |  |
| **4** | **Code đầy đủ** | **2.5** |  |  |  |  |
| 1.1 | Code được trình bày rõ ràng, có chú thích đầy đủ. | 0,5 |  |  |  |  |
| 1.2 | Code chạy đúng, không lỗi. | 0,5 |  |  |  |  |
| 1.3 | Code tối ưu, không dư thừa. | 0,5 |  |  |  |  |
| 1.4 | Đáp ứng đầy đủ các yêu cầu chức năng theo đề bài. | 0,5 |  |  |  |  |
| 1.5 | Có tính sáng tạo hoặc cải thiện so với yêu cầu. | 0,5 |  |  |  |  |
| **TỔNG ĐIỂM BẰNG SỐ:** | | **10** |  |  |  |  |
| **TỔNG ĐIỂM BẰNG CHỮ:** | | *Mười tròn* |  |  |  |  |

**Lời nói đầu**

Trong thời đại công nghệ 4.0 hiện nay, các ứng dụng tự động hóa đang ngày càng trở nên phổ biến và đóng vai trò quan trọng trong nhiều lĩnh vực. Trong đó, ứng dụng chấm công là một trong những công cụ thiết yếu giúp quản lý thời gian làm việc của nhân viên một cách hiệu quả và chính xác. Tuy nhiên, các phương pháp chấm công truyền thống như thẻ từ, dấu vân tay, hay mật khẩu đều có những hạn chế nhất định về độ bảo mật và sự tiện lợi.

Để khắc phục những vấn đề này, công nghệ nhận diện khuôn mặt đã được áp dụng rộng rãi trong các ứng dụng chấm công, mang lại giải pháp vừa an toàn, vừa thuận tiện. Nhận diện khuôn mặt không chỉ giúp xác thực danh tính nhanh chóng mà còn giảm thiểu việc gian lận chấm công, tiết kiệm thời gian và nguồn lực cho doanh nghiệp.

Với nền tảng OpenCV – một thư viện mạnh mẽ và phổ biến trong lĩnh vực xử lý ảnh và nhận diện khuôn mặt, đề tài này hướng đến xây dựng một ứng dụng chấm công tự động bằng công nghệ nhận diện khuôn mặt. Ứng dụng sẽ sử dụng các thuật toán học máy để phát hiện và nhận diện khuôn mặt người dùng, từ đó ghi nhận thời gian vào và ra của nhân viên, giúp cải thiện hiệu quả quản lý nhân sự.

Trong khuôn khổ đề tài này, em sẽ triển khai các kỹ thuật nhận diện khuôn mặt cơ bản như Haar Cascades và các mô hình học sâu để tối ưu hóa độ chính xác và hiệu suất của hệ thống. Em hy vọng rằng ứng dụng này sẽ đóng góp vào việc cải tiến quy trình quản lý nhân sự, đồng thời tạo ra một công cụ hữu ích cho các doanh nghiệp trong việc chấm công tự động.

Mục lục

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT** 8](#_Toc184653247)

[**1.1.Bài toán nhận dạng** 8](#_Toc184653248)

[**1.2.Các kỹ thuật trong bài toán nhận dạng** 9](#_Toc184653251)

[**1.2.1. PCA (Principal Component Analysis)** 9](#_Toc184653252)

[**1.2.2. EigenFace** 11](#_Toc184653253)

[**1.2.3.Face Detection** 12](#_Toc184653254)

[**1.2.4. AdaBoost** 15](#_Toc184653255)

[**1.3.Ngôn ngữ lập trình và thư viện sử dung** 17](#_Toc184653256)

[**1.3.1. Python** 17](#_Toc184653257)

[**1.3.2. Các thư viện** 18](#_Toc184653258)

[**CHƯƠNG 2: XÂY DỰNG HỆ THỐNG** 20](#_Toc184653259)

[**2.1. Mô tả bài toán** 20](#_Toc184653260)

[**2.2. Xây dựng hệ thống** 21](#_Toc184653261)

[**2.2.1.OpenCV** 21](#_Toc184653262)

[**2.2.2. Thuật toán Haar Cascade** 22](#_Toc184653263)

[**2.2.3. Thuật toán LBPH (Local Binary Pattern Histogram)** 23](#_Toc184653264)

[**2.2.4.Thuật toán nhận diện khuôn mặt (Face Recognition)** 24](#_Toc184653265)

[**CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM** 25](#_Toc184653266)

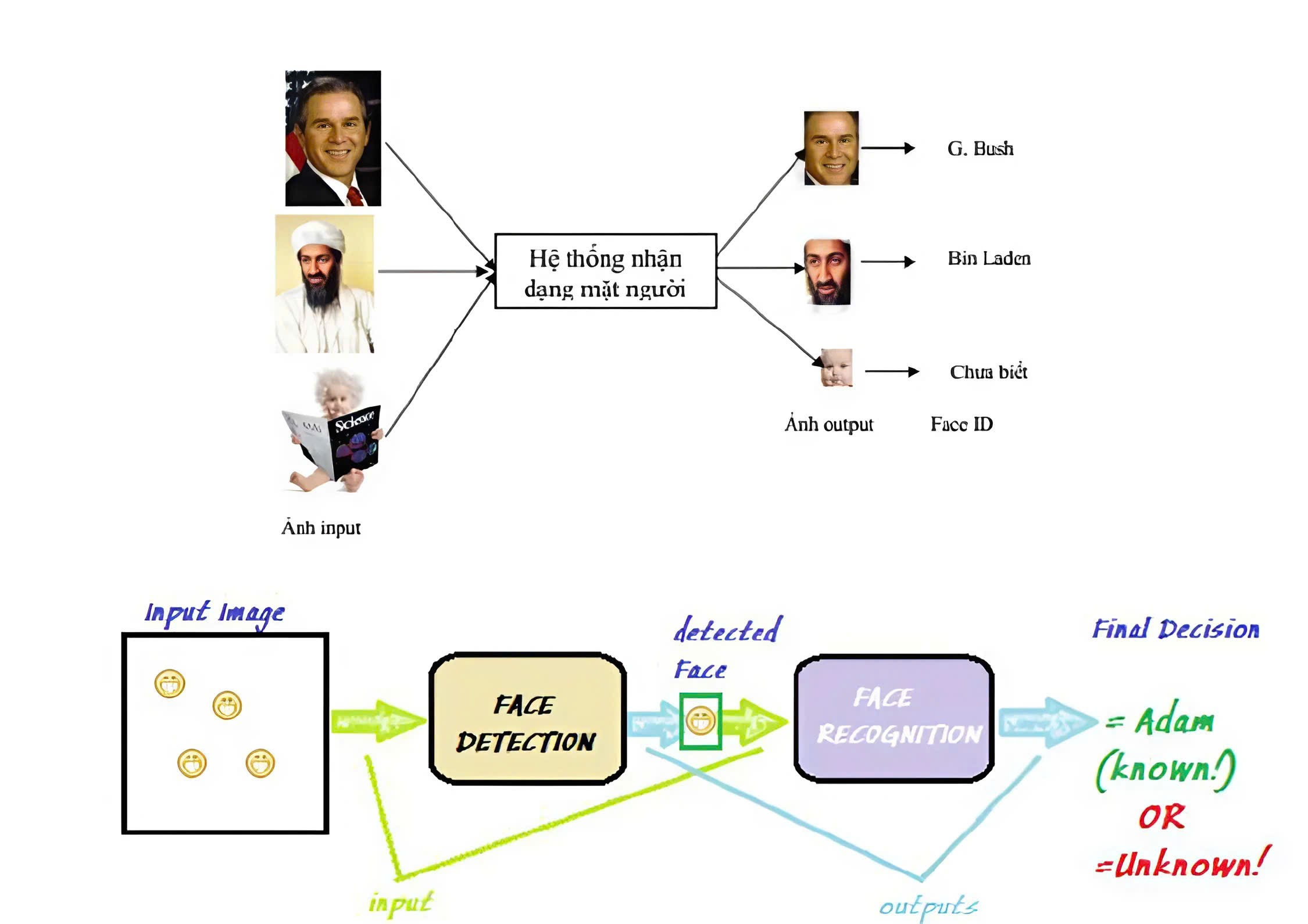
[**3.1. Mô tả dữ liệu** 25](#_Toc184653267)

[**3.2. Kết quả** 29](#_Toc184653268)

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

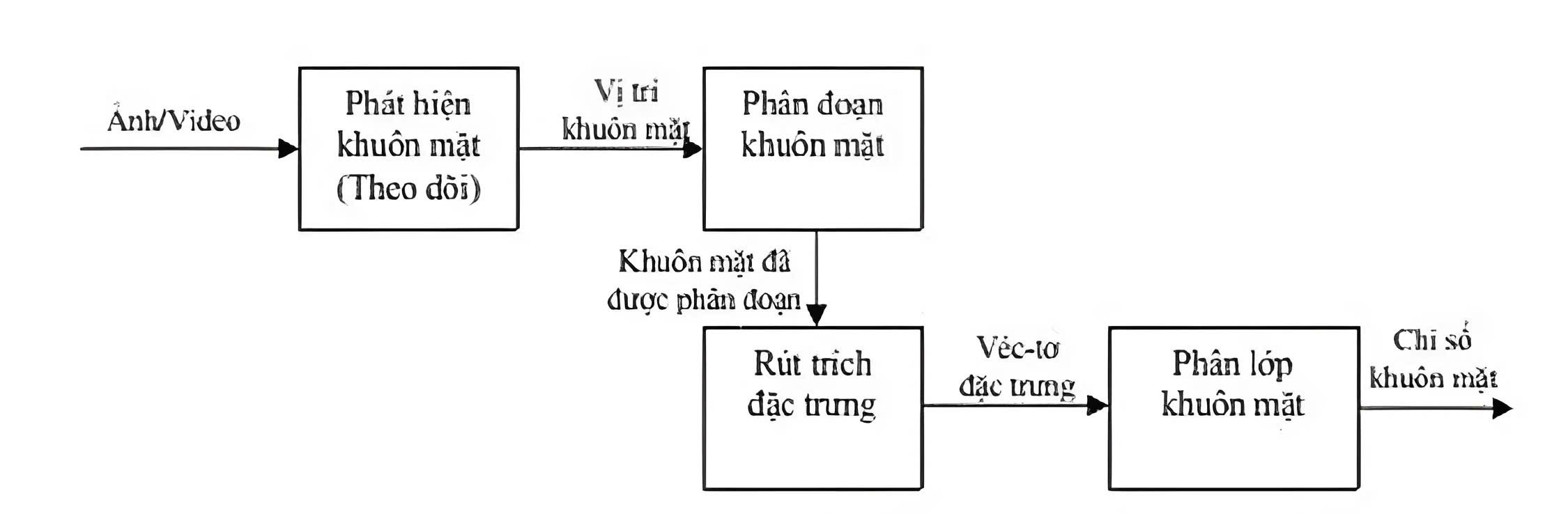
## **1.1.Bài toán nhận dạng**

# Hệ thống nhận dạng mặt người là một hệ thống nhận vào là một ảnh hoặc một đoạn video. Qua xử lý, tính toán hệ thống xác định được vị trí mặt người (nếu có) trong ảnh và xác định là người nào trong số những người mà hệ thống đã được biết (qua quá trình học) hoặc là người lạ.



Một hệ thống nhận dạng mặt người thông thường các bước xử lý sau:

* Phát hiện khuôn mặt (Face Dectection)
* Ohaan đoạn khuôn mặt (Face Alignment hay Sementation)
* Trích chọn đặc trưng (Feature Extraction)
* Nhận dạng hay phân lớp

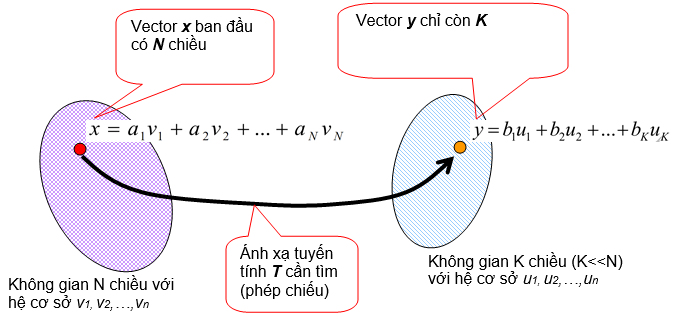


# Phát hiện khuôn mặt dò tìm, định vị những vùng (vị trí) có thể là khuôn mặt xuất hiện trong ảnh hoặc các frame video. Các vùng này sẽ được tách riêng để xử lý. Phân đoạn khuôn mặt sẽ xác định vị trí mắt mũi, miệng và các thành phần khác của khuôn mặt và chuyển kết quả này cho bước trích chọn đặc trưng. Ở bước trích chọn đặc trưng, bằng một phương pháp trích chọn đặc điểm nào đó (mẫu nhị phân cục bộ-Local Binary Pattern-LBP, Gabor wavelets…) sẽ được sử dụng với ảnh mặt để trích xuất các thông tin đặc trưng cho ảnh từ các thông tin về các thành phần trên khuôn mặt, kết quả là mỗi ảnh sẽ được biểu diễn dưới dạng một vector đặc trưng (feature vector). Những vectơ đặc trưng này sẽ là dữ liệu đầu vào cho một mô hình đã được huấn luyện trước để nhận dạng khuôn mặt (Face Recognition) hay phân lớp khuôn mặt (Face Lớpifition), tức là xác định danh tính (identity) hay nhãn của ảnh-đó là ảnh của ai. Ở bước nhận dạng khuôn mặt (Face Recognition), thường thì phương pháp k-láng giềng gần (k-nearest neighbor: kNN) sẽ được sử dụng.

## **1.2.Các kỹ thuật trong bài toán nhận dạng**

### **1.2.1. PCA (Principal Component Analysis)**

PCA là một trong những phương pháp giảm chiều dữ liệu (Dimensionality reduction techniques) phổ biến nhất và được sử dụng trong nhiều lĩnh vực khác nhau. PCA có nhiều ứng dụng như tìm mối tương quan giữa các biến (relationship between observation), trích xuất những thông tin quan trọng từ data, phát hiện và loại bỏ outlier và giảm chiều chiều dữ liệu. Ý tưởng của phương pháp PCA là tìm ra một không gian mới để chiếu data sao cho variation giữ lại nhiều nhất. Tức là “giảm số chiều” của 1 tập vector sao cho vẫn đảm bảo được “tối đa thông tin quan trọng nhất”

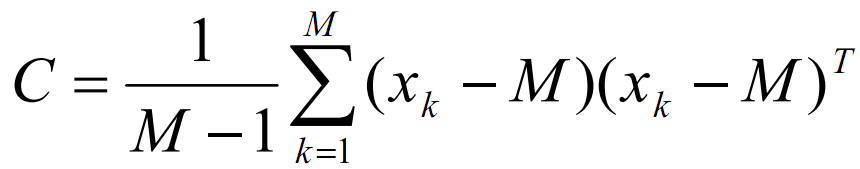


Tất nhiên, khi loại bỏ 1 số thành phần của x để thu được y thì sẽ gây ra sai số (lỗi) !!

🡪 Phương pháp PCA sẽ cố gắng tìm phép biến đổi tuyến tính *T* thỏa: *y = T.x* sao cho trung bình bình phương lỗi (MSE) là bé nhất.

* Cách để tìm được T:

Gọi M là vector trung bình của tất cả các vector x trong tập mẫu.

Gọi ma trận hiệp phương sai của các phần tử x trong tập mẫu là C. C được tính theo công thức sau:

Người ta chứng minh được rằng:

“Nếu T là ma trận *m* hàng, mỗi hàng là 1 vector riêng của C, đồng thời *m* vector riêng này phải ứng với *m* trị riêng lớn nhất. Khi đó T chính là phép biến đổi thỏa MSE nhỏ nhất”.

🡪 Nói tóm lại, phương pháp PCA quy về việc đi tìm trị riêng (eigenvalues) và vector riêng (eigenvectors) của ma trận hiệp phương sai C của tập mẫu X. Sau đó, ta chỉ giữ lại K vector riêng ứng với K trị riêng lớn nhất để làm cơ sở cho không gian mới này

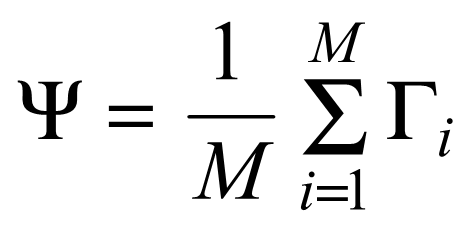
### **1.2.2. EigenFace**

* EigenFace: Hiểu một cách đơn giản là nó dùng PCA là Feature Extraction sau đó mới đưa vào model để training. Mỗi image có chiều chẳng hạn 28x28 = 784 pixel nếu đưa hết vào model thì có một số nhược điểm sau :
  + Thời gian training lâu vì chiều dữ liệu lớn
  + Không phải tất cả các vị trí trên image đều quan trọng
* Vì vậy PCA giúp ta khắc phục các nhược điểm này , nó giúp ta giảm chiều dữ liệu mà vẫn giữ lại được những thông tin quan trọng trên image.
* Các bước thực hiện thuật toán:

Bước 1: Sử dụng các ảnh khuôn mặt *I1, I2, … In*(tập các khuôn mặt huấn luyện) với khuôn mặt phải *chính diện* & tất cả ảnh phải *cùng kích thước.*

Bước 2: Biểu diễn mọi ảnh Ii thành vector 

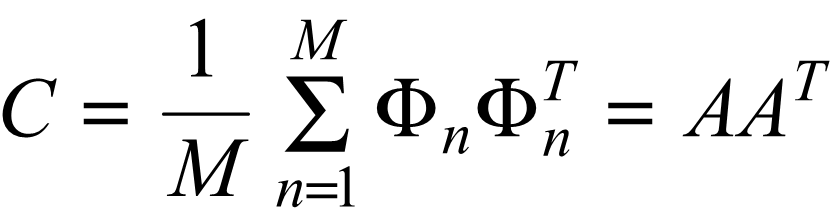
Bước 3: Tính vector khuôn mặt trung bình  theo công thức:



Bước 4: Tính hiệu vector  và



Bước 5: Tính ma trận hiệp phương sai (covariance) C.



C sẽ có kích thước *N2xN2*



Trong đó: A sẽ có kích thước là *N2xM*

Bước 6:

Tính các Eigenvector *ui*(*“vector riêng”*) của ma trận vuông *A.AT* (C có kích thước *N2xN2*).

Bước 7:

Cuối cùng chọn số eigen vector để chiếu data sang không gian mới lấy nó là feature để training.

### **1.2.3.Face Detection**

Để có thể nhận dạng được danh tính khuôn mặt, đầu tiên ta cần phát hiện được vị trí khuôn mặt ở đâu trong bức ảnh hay camera. Để làm được điều này ta sử dụng thuật toán Viola-Jones.

Thuật toán Viola-Jones bao gồm bốn giai đoạn.

Bốn giai đoạn là:

1. Lựa chọn tính năng Haar

2. Tạo một hình ảnh tích hợp

3. Đào tạo Adaboost

4. Phân loại xếp tầng

Trong giai đoạn đầu tiên của thuật toán Viola-Jones, phân loại Cascade dựa trên tính năng Haar. Quá trình phân loại tầng dựa trên tính năng Haar được triển khai theo đó chức năng phân tầng được đào tạo từ nhiều hình ảnh của khuôn mặt (positive images) và hình ảnh không có khuôn mặt (negative images). Tất cả các khuôn mặt của con người có chứa các thuộc tính tương tự. Các tính năng của Haar có thể được sử dụng để phù hợp với các quy tắc này.

Một số thuộc tính mà tất cả các khuôn mặt của con người bao gồm:

1. Má trên sáng hơn vùng mắt.

2. Vùng mắt tối hơn vùng mũi sống.

3. Vị trí và kích thước của sống mũi, miệng và mắt

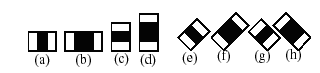
4. Độ dốc định hướng của cường độ điểm ảnh.

Để sử dụng các đặt trưng này vào việc xác định khuôn mặt người, 4 đặt trưng Haar-like cơ bản được mở rộng ra, và được chia làm 3 tập đặc trưng như sau:

1. Đặc trưng cạnh (edge features):

[https://1.bp.blogspot.com/-VfVkuegsiwU/TzI414grUcI/AAAAAAAAAew/yWbLYpIyYeQ/s1600/image002.png](https://1.bp.blogspot.com/-VfVkuegsiwU/TzI414grUcI/AAAAAAAAAew/yWbLYpIyYeQ/s1600/image002.png)

2. Đặc trưng đường (line features):

[](https://2.bp.blogspot.com/-l-KNt0P3waU/TzI5EbCYq9I/AAAAAAAAAe4/vMFZjftJHpY/s1600/Haarlike-2.png)

3. Đặc trưng xung quanh tâm (center-surround features):

[https://4.bp.blogspot.com/-J3I13L81GDw/TzI5PBONVRI/AAAAAAAAAfA/lcj_rZ2k8e4/s1600/Haarlike-3.png](https://4.bp.blogspot.com/-J3I13L81GDw/TzI5PBONVRI/AAAAAAAAAfA/lcj_rZ2k8e4/s1600/Haarlike-3.png)

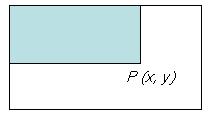
Hình 2. 1 Các đặc trưng mở rộng của đặc trưng Haar-like cơ sở

Dùng các đặc trưng trên, ta có thể tính được giá trị của đặc trưng Haar-like là sự chênh lệch giữa tổng của các pixel của các vùng đen và các vùng trắng như trong công thức sau:

f(x) = Tổngvùng đen(các mức xám của pixel) - Tổngvùng trắng(các mức xám của pixel)

Sử dụng giá trị này, so sánh với các giá trị của các giá trị pixel thô, các đặc trưng Haar-like có thể tăng/giảm sự thay đổi in-class/out-of-class (bên trong hay bên ngoài lớp khuôn mặt người), do đó sẽ làm cho bộ phân loại dễ hơn.

Như vậy ta có thể thấy rằng, để tính các giá trị của đặc trưng Haar-like, ta phải tính tổng của các vùng pixel trên ảnh. Nhưng để tính toán các giá trị của các đặc trưng Haar-like cho tất cả các vị trí trên ảnh đòi hỏi chi phí tính toán khá lớn, không đáp ứng được cho các ứng dụng đòi hỏi tính run-time. Do đó Viola và Jones đưa ra một khái niệm gọi là Integral Image, là một mảng 2 chiều với kích thước bằng với kích của ảnh cần tính các đặc trưng Haar-like, với mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên (dòng-1) và bên trái (cột-1) của nó. Bắt đầu từ vị trí trên, bên trái đến vị trí dưới, phải của ảnh, việc tính toán này đơn thuần chỉ đựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh.

[](https://2.bp.blogspot.com/-n-BAaDMOCvo/TzI5fkMRsFI/AAAAAAAAAfI/PlaaeVry_Fw/s1600/IntegralImag-1-new.gif.jpg)

[https://2.bp.blogspot.com/-_sc_-nfrj1s/TzI5hpdtXFI/AAAAAAAAAfQ/LAntPDhSXCI/s1600/IntegralImag-2-new.gif.jpg](https://2.bp.blogspot.com/-_sc_-nfrj1s/TzI5hpdtXFI/AAAAAAAAAfQ/LAntPDhSXCI/s1600/IntegralImag-2-new.gif.jpg)

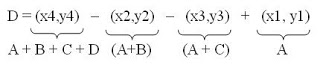
Hình 2. 2 Cách tính Integral Image của ảnh

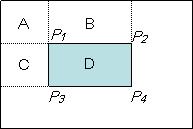
Sau khi đã tính được Integral Image, việc tính tổng các giá trị mức xám của một vùng bất kỳ nào đó trên ảnh thực hiện rất đơn giản theo cách sau:

Giả sử ta cần tính tổng các giá trị mức xám của vùng D như trong hình 4, ta có thể tính như sau:

D = A + B + C + D – (A+B) – (A+C) + A

Với A + B + C + D chính là giá trị tại điểm P4 trên Integral Image, tương tự như vậy A+B là giá trị tại điểm P2, A+C là giá trị tại điểm P3, và A là giá trị tại điểm P1. Vậy ta có thể viết lại biểu thức tính D ở trên như sau:

[](https://1.bp.blogspot.com/-ofY9IDi0KD8/TzI5w8Sh8uI/AAAAAAAAAfY/nWPGqJmTo8s/s1600/IntegralImag-3.gif.jpg)

[](https://4.bp.blogspot.com/-LfhdqoYrXm4/TzI5xRcjMMI/AAAAAAAAAfc/75EyC1f1U18/s1600/sumgraylevel.gif.jpg)

Hình 2. 3 Ví dụ cách tính nhanh các giá trị mức xám của vùng D trên ảnh

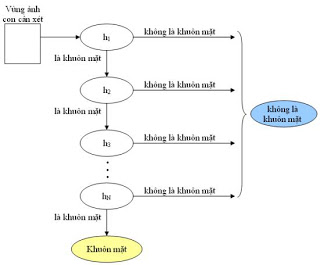
Tiếp theo, để chọn các đặc trưng Haar-like dùng cho việc thiết lập ngưỡng, Viola và Jones sử dụng một phương pháp máy học được gọi là AdaBoost. AdaBoost sẽ kết hợp các bộ phân loại yếu để tạo thành một bộ phân loại mạnh. Với bộ phân loại yếu chỉ cho ra câu trả lời chính xác chỉ hơn viện đoán một cách ngẫn nhiên một chút, còn bộ phân loại mạnh có thể đưa ra câu trả lời chính xác trên 60%.

### **1.2.4. AdaBoost**

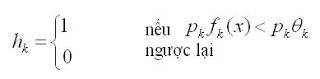
AdaBoost là một bộ phân loại mạnh phi tuyến phức dựa trên hướng tiếp cận boosting được Freund và Schapire đưa ra vào năm 1995 [2]. Adaboost cũng hoạt động trên nguyên tắc kết hợp tuyến tính các weak classifiers để hình thành một strong classifier.

Là một cải tiến của tiếp cận boosting, AdaBoost sử dụng thêm khái niệm trọng số (weight) để đánh dấu các mẫu khó nhận dạng. Trong quá trình huấn luyện, cứ mỗi weak classifiers được xây dựng, thuật toán sẽ tiến hành cập nhật lại trọng số để chuẩn bị cho việc xây dựng weak classifier kế tiếp: tăng trọng số của các mẫu bị nhận dạng sai và giảm trọng số của các mẫu được nhận dạng đúng bởi weak classifier vừa xây dựng. Bằng cách này weak classifer sau có thể tập trung vào các mẫu mà các weak classifiers trước nó làm chưa tốt. Sau cùng, các weak classifers sẽ được kết hợp tùy theo mức độ tốt của chúng để tạo nên strong classifier.

Viola và Jones dùng AdaBoost kết hợp các bộ phân loại yếu sử dụng các đặc trưng Haar-like theo mô hình phân tầng (cascade) như sau:

[](https://2.bp.blogspot.com/-p9jOLhjphGU/TzI59Yb21LI/AAAAAAAAAfo/bAfX1oVLDHs/s1600/AdaBoost-0.gif.jpg)

Trong đó, hk là các bộ phân loại yếu, được biểu diễn như sau:

[](https://4.bp.blogspot.com/-bR-KWQLm71A/TzI6JzAt0yI/AAAAAAAAAfw/ktXSzndxG74/s1600/AdaBoost-1.gif.jpg)

x: cửa sổ con cần xét

Ok: ngưỡng (O = teta)

fk: giá trị của đặc trưng Haar-like

pk: hệ số quyết định chiều của phương trình

AdaBoost sẽ kết hợp các bộ phân loại yếu thành bộ phân loại mạnh như sau:

H(x) = sign(a1h1(x) +a2h2(x) + ... + anhn(x)) (a = alpha)

Với: at >= 0 là hệ số chuẩn hoá cho các bộ phân loại yếu

[](https://4.bp.blogspot.com/-sI2m1ndrF74/TzI6VFVv6xI/AAAAAAAAAf4/HisDFfvPDsY/s1600/AdaBoost-2.jpg)

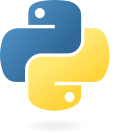
## **1.3.Ngôn ngữ lập trình và thư viện sử dung**

### **1.3.1. Python**

Python là một ngôn ngữ lập trình bậc cao, thông dịch, và đa mục đích, được Guido van Rossum phát hành năm 1991. Với cú pháp đơn giản, dễ đọc và thư viện phong phú, Python đã trở thành lựa chọn phổ biến trong nhiều lĩnh vực như phát triển web, khoa học dữ liệu, trí tuệ nhân tạo, và tự động hóa.

Python hỗ trợ lập trình hướng đối tượng và có khả năng mở rộng nhờ cộng đồng đông đảo và mã nguồn mở. Các framework nổi bật như Django, Flask hỗ trợ phát triển web, trong khi NumPy, pandas, và TensorFlow là những công cụ mạnh mẽ cho khoa học dữ liệu và học máy.

Dù hiệu suất không bằng các ngôn ngữ như C++, Python vẫn được ưa chuộng nhờ sự linh hoạt, khả năng tích hợp tốt, và tài liệu phong phú. Python đã khẳng định vị trí của mình trong cả giáo dục lẫn công nghiệp.



### **1.3.2. Các thư viện**

* Thư viện Tkinter:

Tkinter là một thư viện trong ngôn ngữ lập trình Python được sử dụng để tạo giao diện đồ họa người dùng (GUI). "Tkinter" là viết tắt của "Tk interface,", một toolkit đồ họa cung cấp các công cụ để phát triển giao diện người dùng.

Tkinter là một phần của thư viện tiêu chuẩn của Python và đã được tích hợp sẵn trong hầu hết các cài đặt Python. Điều này giúp cho Tkinter trở thành một lựa chọn phổ biến cho việc phát triển ứng dụng với giao diện đồ họa đơn giản trong Python.

Một số đặc điểm của Tkinter bao gồm khả năng tạo các thành phần giao diện như cửa sổ, nút, ô văn bản, và các widget khác để tương tác với người dùng. Tkinter cung cấp cả các sự kiện và phương thức để xử lý tương tác người dùng và thay đổi trạng thái của ứng dụng.

* Thư viện cv2 (OPENCV)

OpenCV (Open Source Computer Vision Library) là thư viện mã nguồn mở chuyên về xử lý ảnh và thị giác máy tính. Nó hỗ trợ các tác vụ như:

* Phát hiện/nhận diện khuôn mặt.
* Chuyển đổi định dạng màu (RGB, Grayscale...).
* Xử lý nhiễu, phát hiện cạnh (Canny, Sobel...).
* Theo dõi đối tượng, nhận diện chuyển động.
* Áp dụng học máy và AI trong xử lý hình ảnh.

OpenCV được sử dụng rộng rãi trong các ứng dụng AI, robot, và thị giác máy tính

* Thư viện Pandas:

Thư viện xử lý dữ liệu dạng bảng (DataFrame), hỗ trợ đọc/ghi, lọc, nhóm, và phân tích dữ liệu hiệu quả.

* Thư viện NumPy:

Thư viện tính toán số học, cung cấp mảng (array) hiệu suất cao và công cụ xử lý ma trận, đại số tuyến tính.

* Thư viện Pillow:

Thư viện xử lý ảnh, hỗ trợ mở, chỉnh sửa, và lưu ảnh dưới nhiều định dạng phổ biến.

# **CHƯƠNG 2: XÂY DỰNG HỆ THỐNG**

## **2.1. Mô tả bài toán**

Ứng dụng chấm công sử dụng OpenCV là một hệ thống tự động nhận diện khuôn mặt của người dùng để quản lý điểm danh, thay thế cho phương pháp chấm công truyền thống. Hệ thống này hoạt động bằng cách sử dụng camera để nhận diện khuôn mặt của người dùng trong thời gian thực, xác định danh tính và ghi lại thời gian chấm công. Dữ liệu sẽ được lưu trữ và quản lý trong cơ sở dữ liệu (CSV), giúp đơn giản hóa việc theo dõi thời gian vào và ra của nhân viên hoặc học sinh.

Ứng dụng này gồm ba bước chính: thu thập ảnh khuôn mặt, huấn luyện mô hình nhận diện khuôn mặt, và nhận diện khuôn mặt để chấm công. Việc sử dụng OpenCV giúp hệ thống có thể phát hiện và nhận diện khuôn mặt chính xác trong môi trường khác nhau, cải thiện sự tiện lợi và độ chính xác so với các phương pháp chấm công thông thường.

Các chức năng chính

* Thu thập ảnh khuôn mặt (Take Images):
  + Người dùng nhập ID và tên của mình vào các ô nhập liệu.
  + Hệ thống chụp ảnh khuôn mặt của người dùng khi nhận diện được khuôn mặt, lưu trữ vào thư mục "TrainingImage" với tên và ID của người dùng.
* Huấn luyện mô hình nhận diện (Train Images):
  + Dữ liệu ảnh đã thu thập được sẽ được sử dụng để huấn luyện một mô hình nhận diện khuôn mặt sử dụng phương pháp LBPH (Local Binary Pattern Histogram).
  + Mô hình huấn luyện sẽ được lưu trữ trong tệp "Trainner.yml" để sử dụng cho việc nhận diện trong tương lai.
* Nhận diện khuôn mặt và chấm công (Track Images):
  + Hệ thống sử dụng mô hình đã huấn luyện để nhận diện khuôn mặt người dùng trong thời gian thực.
  + Khi nhận diện thành công, thông tin về người dùng, thời gian và ngày hiện tại sẽ được lưu trữ vào tệp CSV dưới dạng danh sách chấm công.
  + Hệ thống cũng sẽ lưu lại những khuôn mặt chưa được nhận diện vào thư mục "ImagesUnknown" để có thể kiểm tra lại sau.
* Quản lý chấm công (Attendance Management):
  + Mỗi lần nhận diện thành công, thông tin của người dùng (ID, tên, thời gian, ngày) sẽ được lưu trữ vào một file CSV.
  + Hệ thống sẽ loại bỏ các bản ghi trùng lặp và lưu lại danh sách chấm công vào một tệp tin CSV, có thể tải xuống hoặc in ra.

## **2.2. Xây dựng hệ thống**

### **2.2.1.OpenCV**

OpenCV trong ứng dụng , với nhiều chức năng mạnh mẽ cho việc xử lý ảnh và nhận diện khuôn mặt. Các tính năng của OpenCV được tích hợp trong toàn bộ quy trình từ phát hiện đến nhận diện khuôn mặt.

\*Phát hiện khuôn mặt với Haar Cascade:

OpenCV cung cấp một số mô hình Haar Cascade đã được huấn luyện sẵn cho việc phát hiện các đối tượng như khuôn mặt. Mô hình này cho phép nhận diện khuôn mặt trong thời gian thực ngay khi người dùng xuất hiện trước camera. Phát hiện khuôn mặt chính xác giúp xác định vị trí khuôn mặt và chuẩn bị cho các bước tiếp theo như nhận diện và điểm danh.

\*Quá trình huấn luyện và nhận diện khuôn mặt với LBPH:

OpenCV giúp đơn giản hóa quá trình huấn luyện mô hình LBPH. Các bước như thu thập ảnh khuôn mặt, lưu trữ dữ liệu, và huấn luyện mô hình nhận diện đều được thực hiện dễ dàng với các hàm có sẵn trong OpenCV. Khi người dùng chụp ảnh khuôn mặt của họ, hệ thống sẽ sử dụng phương pháp LBPHFaceRecognizer của OpenCV để lưu trữ các đặc trưng khuôn mặt vào một mô hình, sau đó sử dụng mô hình này để nhận diện khuôn mặt trong các lần điểm danh sau.

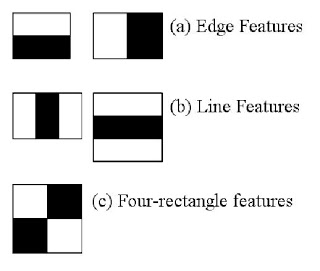
\*Lưu trữ và quản lý thông tin người dùng:

OpenCV cung cấp phương thức để lưu trữ các ảnh khuôn mặt vào một thư mục và sử dụng chúng để huấn luyện mô hình nhận diện. Thông tin người dùng như ID, tên, và thời gian điểm danh cũng được lưu trữ trong các tệp CSV và cơ sở dữ liệu, giúp dễ dàng quản lý và tra cứu thông tin điểm danh.

### **2.2.2. Thuật toán Haar Cascade**

\* Haar Cascade là một trong những phương pháp hiệu quả và phổ biến trong việc phát hiện đối tượng, đặc biệt là trong nhận diện khuôn mặt. Thuật toán này hoạt động dựa trên việc sử dụng các tính năng hình ảnh đặc trưng (như sự khác biệt về độ sáng và cấu trúc) để phát hiện khuôn mặt. Haar Cascade đã được huấn luyện sẵn với hàng nghìn ảnh mặt người và có thể nhận diện khuôn mặt trong các bức ảnh hay video. Mặc dù thuật toán này không cần phần cứng đặc biệt nhưng vẫn có khả năng nhận diện khuôn mặt trong thời gian thực.

Các đặc điểm hình chữ nhật tương tự như nhân được sử dụng để phát hiện các đặc điểm khác nhau của khuôn mặt như mắt và các nốt như trong hình minh họa.



Các tính năng hình chữ nhật được chạy lần lượt trên hình ảnh và tổng số pixel nằm trong phần màu trắng được trừ cho tổng số pixel nằm trong phần màu đen.

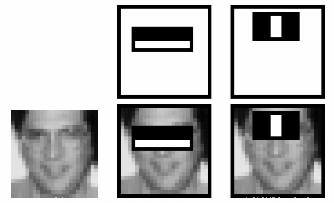
Trong đó

a) Là các bộ lọc bắt các cạnh trong ảnh

b) Bắt các đường thẳng trong ảnh

c) Về đặc trưng 4 hình vuông

Trong hình minh họa bên dưới, đặc điểm hình chữ nhật đầu tiên đang tính toán cho sự khác biệt về cường độ giữa vùng mắt và vùng má trên khuôn mặt. Và đặc điểm hình chữ nhật thứ hai là đo sự chênh lệch về cường độ giữa hai vùng mắt và sống mũi. Bộ lọc Haar chỉ có thể nhìn cụ thể vào một vùng trong cửa sổ để nhận diện.



Vì vậy, sử dụng các đặc điểm hình chữ nhật này trên một hình ảnh, chúng ta có thể tạo ra hàng nghìn điểm đặc trưng cho hình ảnh. Tuy nhiên, việc tính toán tổng số pixel trong các vùng trắng và đen trong toàn bộ ảnh có thể là một hoạt động tốn kém, đặc biệt là đối với các ảnh lớn.

một phương pháp được gọi là ảnh tích phân có thể đạt được tính toán tương tự bằng cách thực hiện các phép toán chỉ trên bốn điểm ảnh. Có thể sử dụng hàng nghìn hình ảnh được gắn nhãn để chuyển đổi chúng thành bản đồ tính năng tầng HAAR và đào tạo nó bằng cách sử dụng mô hình học máy để tạo bộ phân loại nhận diện khuôn mặt.

\* Trong ứng dụng này, thuật toán Haar Cascade được sử dụng để phát hiện khuôn mặt trong video stream từ webcam. Khuôn mặt được phát hiện sẽ được bao quanh bằng một hình chữ nhật, giúp hệ thống xác định rõ vị trí của khuôn mặt trong hình ảnh.

### **2.2.3. Thuật toán LBPH (Local Binary Pattern Histogram)**

\* LBPH là một thuật toán nhận diện khuôn mặt dựa trên việc phân tích các đặc điểm địa phương của bức ảnh. LBPH phân tích từng pixel trong ảnh, so sánh các điểm lân cận và mã hóa chúng thành các mẫu nhị phân. Sau đó, các mẫu này sẽ được chuyển thành các đặc trưng hình học đặc trưng để nhận diện khuôn mặt.

\* Thuật toán này rất hiệu quả trong việc nhận diện khuôn mặt trong điều kiện ánh sáng khác nhau và góc nhìn thay đổi. LBPH là một phương pháp phù hợp cho các ứng dụng chấm công tự động vì nó dễ dàng huấn luyện và có thể nhận diện khuôn mặt nhanh chóng, đặc biệt là khi được kết hợp với các dữ liệu huấn luyện có sẵn.

\* Quá trình nhận diện khuôn mặt sử dụng LBPH được thực hiện qua bước so sánh các đặc trưng của khuôn mặt trong video với các khuôn mặt đã được lưu trong cơ sở dữ liệu. Nếu mức độ khớp là đủ cao, hệ thống sẽ nhận diện thành công và ghi lại thông tin của người dùng.

### **2.2.4.Thuật toán nhận diện khuôn mặt (Face Recognition)**

\* Khi người dùng cung cấp thông tin ID và tên, hệ thống sẽ yêu cầu chụp ảnh khuôn mặt và lưu trữ các bức ảnh này vào thư mục dữ liệu. Những ảnh này sau đó được sử dụng để huấn luyện mô hình nhận diện khuôn mặt.

\* Trong giai đoạn nhận diện, hệ thống sẽ so sánh khuôn mặt hiện tại với các khuôn mặt đã được huấn luyện, và nếu có sự khớp chính xác (dưới ngưỡng độ chính xác nhất định), hệ thống sẽ nhận diện khuôn mặt và xác định người dùng.

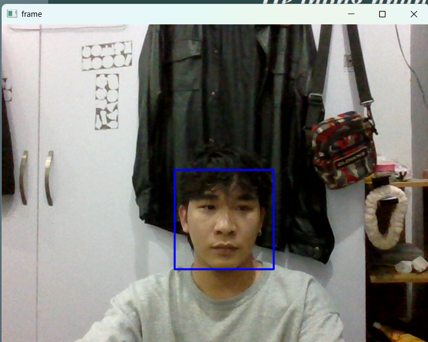
\* Quá trình này giúp tạo ra một hệ thống chấm công tự động, nơi người dùng không cần phải thực hiện thao tác thủ công để điểm danh, mà chỉ cần đứng trước camera để hệ thống tự động nhận diện và ghi lại thông tin.

# **CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM**

## **3.1. Mô tả dữ liệu**

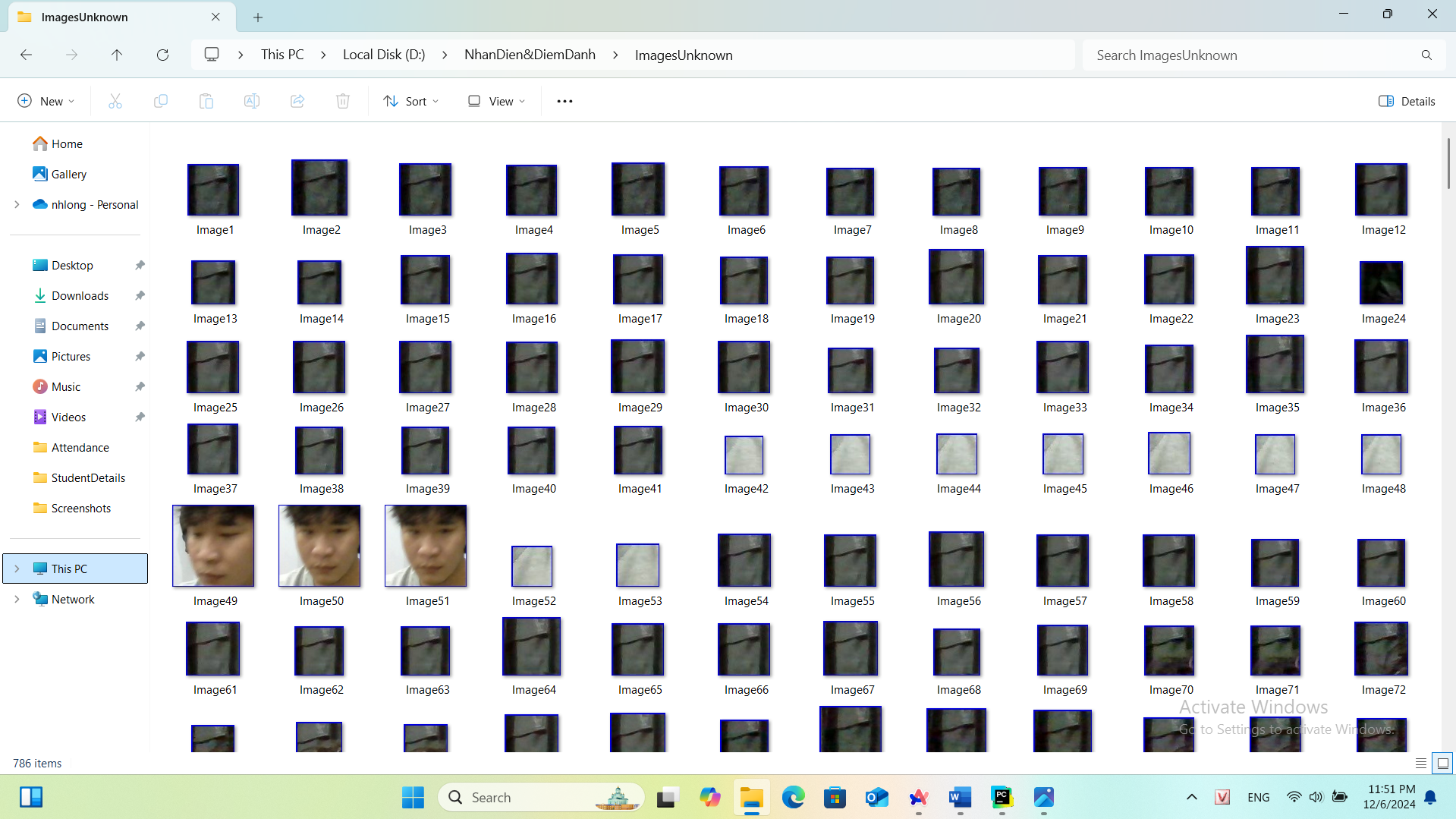
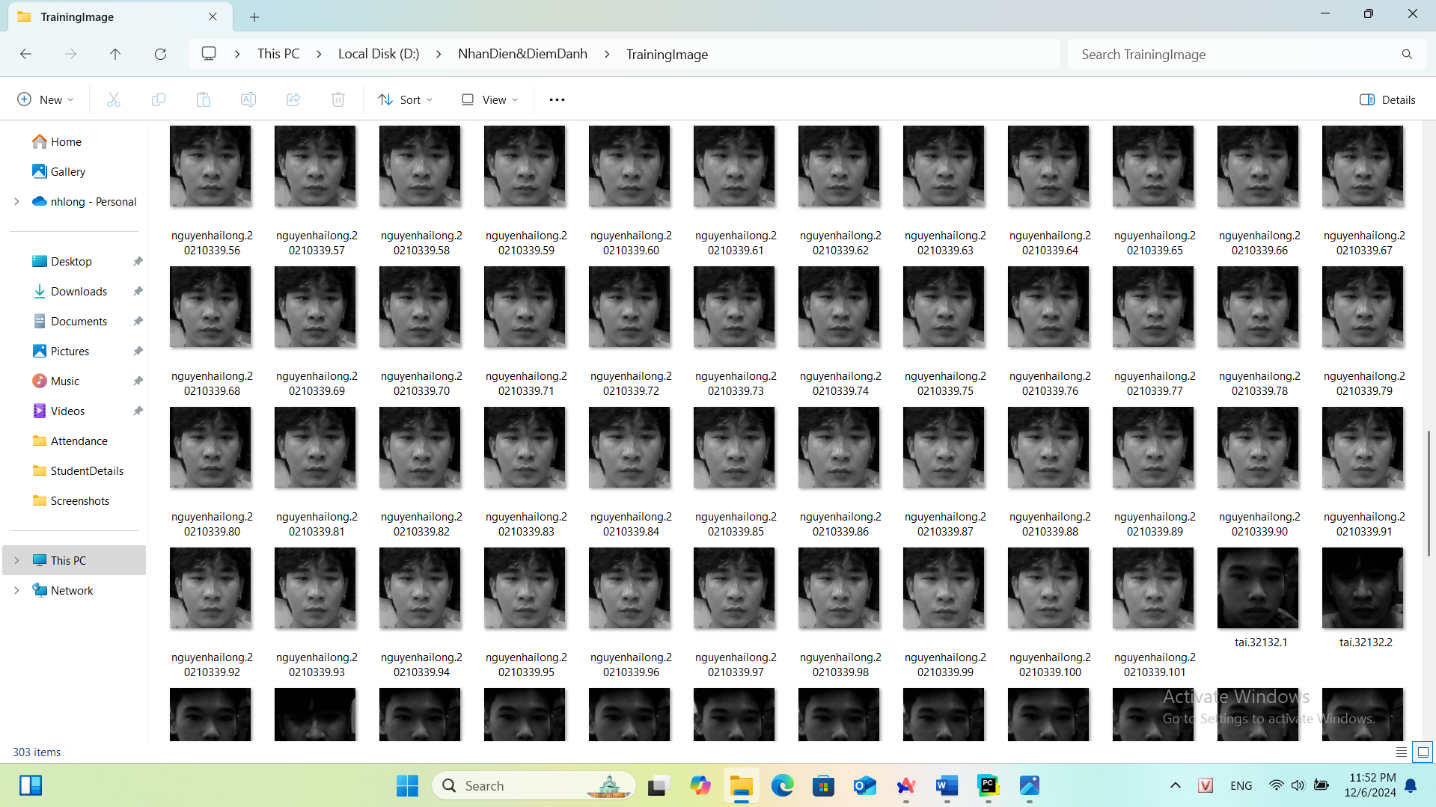
Phát hiện khuôn mặt trong thời gian thực với Haar Cascade:

Khi người dùng đứng trước camera, hệ thống sử dụng thuật toán Haar Cascade để nhận diện khuôn mặt. OpenCV sẽ xử lý video stream và phát hiện các khuôn mặt trong thời gian thực. Khuôn mặt sẽ được đánh dấu bằng một hình chữ nhật, giúp hệ thống xác định vị trí khuôn mặt trong hình ảnh.



Chụp ảnh và huấn luyện mô hình nhận diện khuôn mặt:

Khi người dùng nhập ID và tên, ứng dụng sẽ yêu cầu họ chụp ảnh khuôn mặt. Các ảnh này sẽ được sử dụng để huấn luyện mô hình nhận diện khuôn mặt. OpenCV giúp lưu trữ các ảnh khuôn mặt vào thư mục TrainingImage và sử dụng chúng để huấn luyện mô hình LBPH. Hệ thống sau đó sẽ lưu trữ mô hình huấn luyện vào tệp Trainner.yml để sử dụng trong quá trình nhận diện.



Chấm công tự động:

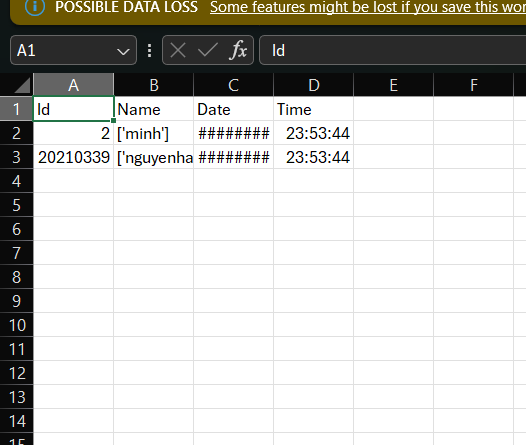
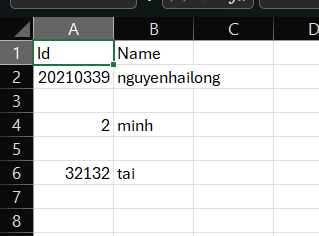
Khi người dùng xuất hiện trước camera, hệ thống sử dụng mô hình LBPH để nhận diện khuôn mặt của họ. Nếu mô hình nhận diện thành công, thông tin của người dùng và thời gian điểm danh sẽ được lưu trữ vào tệp CSV Attendance.csv. Điều này giúp tự động ghi nhận thời gian làm việc mà không cần sự can thiệp của người quản lý.

Thông tin về người dùng (ID, tên, ngày, giờ điểm danh) được hiển thị và lưu trữ, đảm bảo tính chính xác và dễ dàng tra cứu trong tương lai.



Quản lý thông tin điểm danh:

Các thông tin điểm danh sẽ được lưu trong tệp CSV, giúp người quản lý có thể xem lại lịch sử điểm danh của nhân viên hoặc học viên một cách dễ dàng và nhanh chóng. Giao diện người dùng của ứng dụng cũng cho phép tra cứu các thông tin này qua bảng điểm danh được hiển thị trực tiếp trên màn hình.



## **3.2. Kết quả**

Thực nghiệm 1: sử dụng hình ảnh điện thoại



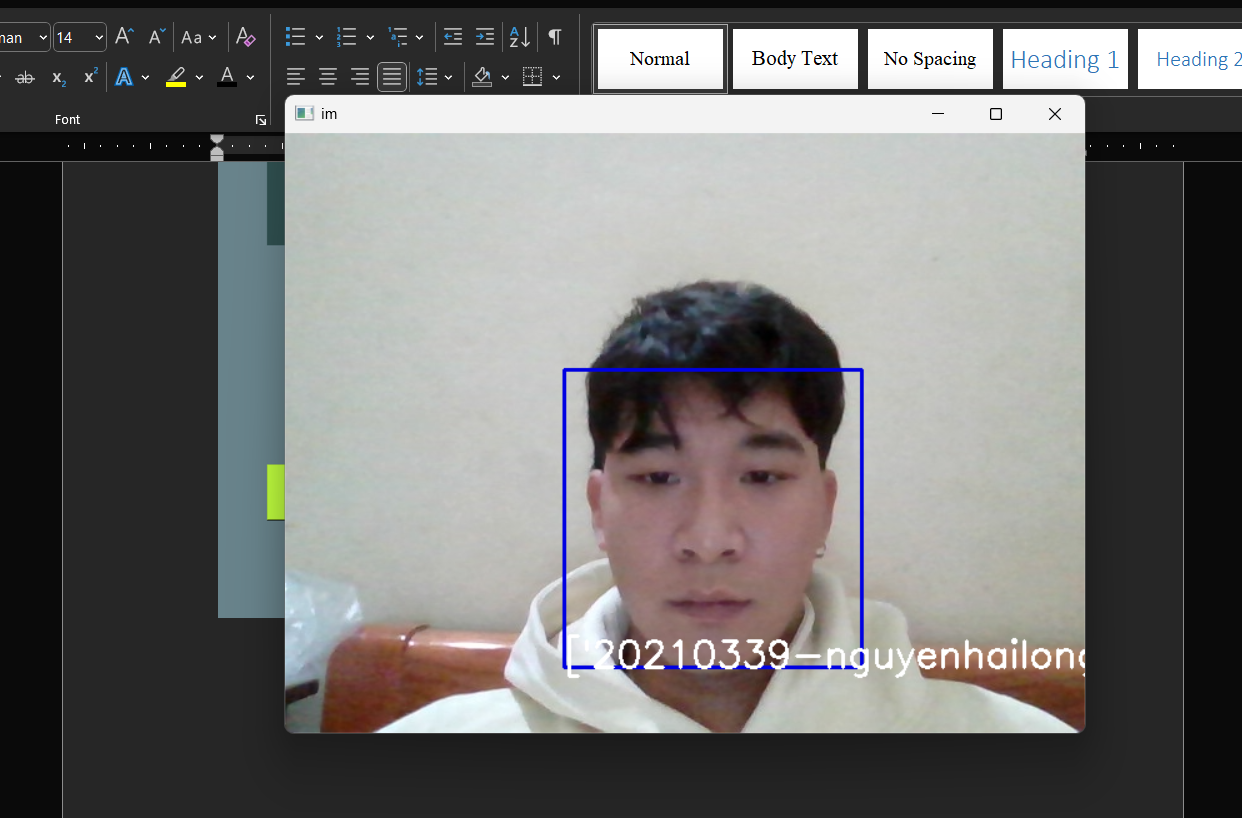
Thực nghiệm 2: 1 người đã được thêm vào danh sách và 1 người chưa được thêm vào danh sách



Thực nghiệm 3: đeo khẩu trang khi điểm danh



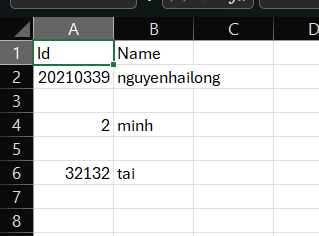
Thực nghiệm 4: 1 người đã được thêm vào danh sách

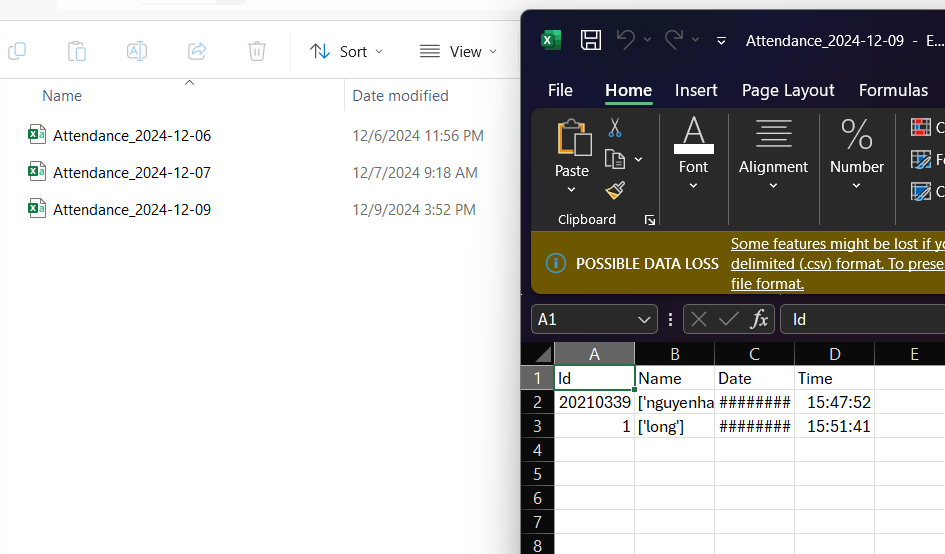


Thực nghiệm 5: 2 người đã được thêm vào danh sách và cùng điểm danh



Kết quả đã được điểm danh và thêm vào danh sách





Tài liệu tham khảo:

<https://docs.opencv.org/>

<https://pyimagesearch.com/>

<https://medium.com/tag/opencv>

<https://dennybritz.com/posts/wildml/understanding-convolutional-neural-networks-for-nlp/>