**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN**

**-----\*\*\*-----**

**A large blue clock on the side

Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN**

**BỘ MÔN: THỊ GIÁC MÁY TÍNH**

***Đề tài:***

***XÂY DỰNG CHƯƠNG TRÌNH NHẬN DẠNG MẶT NGƯỜI SỬ DỤNG PHƯƠNG PHÁP TRÍCH CHỌN ĐẶC TRƯNG MẪU NHỊ PHÂN CỤC BỘ LBP (LOCAL BINARY PATTERNS)***

**GVHD: TS Nguyễn Hữu Tuân**

**Sinh viên thực hiện: Đặng Việt Hoàng – Mã SV: 73341**

***Hải Phòng, tháng 12 năm 2020***

**TRƯỜNG ĐẠI HỌC HÀNG HẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----\*\*\*-----**

**BÀI TẬP LỚN**

**HỌC PHẦN: THỊ GIÁC MÁY TÍNH**

**Đề tài:**

**XÂY DỰNG CHƯƠNG TRÌNH NHẬN DẠNG MẶT NGƯỜI SỬ DỤNG PHƯƠNG PHÁP TRÍCH CHỌN ĐẶC TRƯNG MẪU NHỊ PHÂN CỤC BỘ LBP (LOCAL BINARY PATTERNS)**

1. **Mục đích**

Áp dụng những kiến thức đã được học trong học phần “Thị giác máy tính”, xây dựng được phần mềm nhận dạng mặt người bằng phương pháp LBP

1. **Công việc cần thực hiện**

* Tìm hiểu về đề tài.
* Khảo sát, thu thập thông tin, xử lý nghiệp vụ.
* Làm báo cáo và đánh giá.

1. **Yêu cầu**

Kết quả làm bài tập lớn: Báo cáo bài tập lớn.

Báo cáo bài tập lớn được trình bày theo mẫu, báo cáo bản mềm nộp qua email và bản cứng nộp trực tiếp.

***Hải Phòng, tháng 12 năm 2020***

**NGƯỜI HƯỚNG DẪN**

**TS. Nguyễn Hữu Tuân**

MỤC LỤC

[CHƯƠNG 1: GIỚI THIỆU 1](#_Toc58519093)

[1.1. LÍ DO CHỌN ĐỀ TÀI 1](#_Toc58519094)

[1.2. MỤC ĐÍCH 2](#_Toc58519095)

[1.3. NGỮ CẢNH ÁP DỤNG, PHẠM VI NGHIÊN CỨU 2](#_Toc58519096)

[1.4. YÊU CẦU BÀI TOÁN 3](#_Toc58519097)

[1.5. BỐ CỤC BÁO CÁO 3](#_Toc58519098)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH THIẾT KẾ HỆ THỐNG 4](#_Toc58519099)

[2.1. MÔ HÌNH TỔNG QUAN 4](#_Toc58519100)

[2.2. MÔ HÌNH CHI TIẾT VÀ PHÂN TÍCH 4](#_Toc58519101)

[2.2.1. Mô hình chi tiết 4](#_Toc58519102)

[2.2.2. Phân tích 5](#_Toc58519103)

[CHƯƠNG 3: CÀI ĐẶT VÀ ĐÁNH GIÁ HỆ THỐNG 9](#_Toc58519104)

[3.1. NGÔN NGỮ LẬP TRÌNH, THƯ VIỆN SỬ DỤNG, NỀN TẢNG SỬ DỤNG 9](#_Toc58519105)

[3.1.1. Ngôn ngữ lập trình: Python 3.8.3 9](#_Toc58519106)

[3.1.2. Thư viện sử dụng: OpenCV 10](#_Toc58519107)

[3.1.3. Nền tảng lập trình: Anaconda – IDE: Spyder 11](#_Toc58519108)

[3.2. CÀI ĐẶT HỆ THỐNG 12](#_Toc58519109)

[3.2.1. Hàm thu thập data: 12](#_Toc58519110)

[3.2.2. Training: 14](#_Toc58519111)

[3.2.3. Nhận dạng mặt người: 16](#_Toc58519112)

[3.2.4. Kết quả nghiệm thu 18](#_Toc58519113)

[3.3. ĐÁNH GIÁ HỆ THỐNG – KẾT LUẬN 19](#_Toc58519114)

[3.3.1. Ưu điểm: 19](#_Toc58519115)

[3.3.2. Nhược điểm: 19](#_Toc58519116)

[3.3.3. Hướng phát triển 19](#_Toc58519117)

# 

# CHƯƠNG 1: GIỚI THIỆU

## LÍ DO CHỌN ĐỀ TÀI

Trong những năm gần đây, các nghiên cứu và ứng dụng của trí tuệ nhận tạo (Artificial Intellegence) và học máy (Machine Learning) thu hút được sự quan tâm của không chỉ là những nhà khoa học, mà môi trường học tập trên trường, lớp cũng một bước tìm hiểu về chúng. Một trong những lĩnh vực liên quan tới công nghệ tri thức mà hiện nay được ứng dụng rất nhiều vào trong cuộc sống là Pattern Recognition hay còn gọi là nhận dạng mẫu. Ta có thể kể đến các hệ thống nhận dạng phổ biến hiện nay như: nhận dạng chữ viết, nhận dạng chữ ký, nhận dạng vân tay, nhận dạng chuyển động của mắt, nhận dạng mặt người, nhận dạng các văn bản thông dụng,..

Một trong những bài toán được quan tâm đông đảo cho đến thời điểm này là bài toán nhận dạng mặt người (Face Recognition). Khuôn mặt đóng vai trò quan trọng trong quá trình giao tiếp giữa người với người và cũng mang một lượng thông tin giàu có, chẳng hạn như qua mặt người, ta có thể xác định được giới tính, tuổi tác, chủng tộc, cảm xúc,… Nhận dạng khuôn mặt không phải là bài toán mới nhưng nó vẫn là một thách thức lớn vì một bài toán nhận dạng mặt người chứa nhiều các bài toán khác như: phát hiện mặt người (Face Detection), đánh dấu mặt người (Facial Landmarking), trích rút đặc trưng (Feature Extraction), gán nhãn, phân lớp (Classification). Ngoài ra, trong thực tế, ảnh/ video chứa khuôn mặt có thể chứa đựng nhiều vấn đề như: độ sáng, độ nhòe/mờ, độ nhiễu, độ phân giải, góc quay ảnh,...

Trong thực tế, nhận dạng khuôn mặt người (Face Recognition) là một hướng nghiên cứu được rất nhiều nhà khoa học lớn quan tâm. Các ông lớn như Microsoft, Google, Apple, Facebook đều có các trung tâm sinh trắc học và nghiên cứu về nhận dạng khuôn mặt người là một trong lĩnh vực nghiên cứu chính cho đến nay.

Cùng với sự phát triển của lĩnh vực thị giác máy tính (Computer Vision) và học máy (Machine Learning), có rất nhiều các hệ thống nhận dạng khuôn mặt khác nhau đã được phát triển. Kết quả nhận dạng của từng hệ thống cũng rất khác nhau vì tùy thuộc vào dữ liệu thử. Trong khuôn khổ bài tập lớn này, chúng ta sẽ tìm hiểu về phương pháp trích chọn đặc trưng mẫu nhị phân cục bộ (LBP) để áp dụng vào bài toán nhận dạng mặt người.

Local Binary Patterns (hay còn viết tắt là LBP) là một phương pháp rút trích đặc trưng trong xử lý ảnh. Đặc trưng được rút trích sẽ tiếp tục được tiến hành chọn lọc (feature selection) thu gọn thành vector đặc trưng. Vector đặc trưng này sau đó có thể dùng để đưa vào mô hình học máy để học / phân loại.

LBP là một đặc trưng rất phổ biến trong các bài toán liên quan đến ảnh khuôn mặt nói chung và nhận dạng khuôn mặt nói riêng. Điểm nổi trội trong phương pháp này là khả năng bất biến với độ sáng – một trong những thách thức lớn nhất của bài toán liên quan đến ảnh mặt người).

## MỤC ĐÍCH

* Nghiên cứu cách thức xây dựng một hệ thống nhận dạng nói chung và nhận dạng khuôn mặt nói riêng.
* Nghiên cứu về đặc trưng rất thông dụng trong các bài toán liên quan đến ảnh khuôn mặt nói chung và nhận dạng khuôn mặt nói riêng: Local Binary Pattern (LBP).

## NGỮ CẢNH ÁP DỤNG, PHẠM VI NGHIÊN CỨU

* Điều kiện nghiên cứu:
  + Ánh sáng đều, không có chiếu sáng, không có ánh sáng mạnh hay quá tối
  + Góc ảnh: trực diện hoặc gần trực diện
  + Không bị che khuất, nạp data riêng lẻ cá nhân, tránh sai xót trong việc nhận dạng nhiều khuôn mặt cùng lúc.
  + Ảnh chất lượng cao (nếu có thể)
* Phạm vi nghiên cứu: Phát hiện khuân mặt mỗi người thông qua video,ảnh. Một cá nhân sử dụng máy tính, với hình ảnh hoặc video chuyển động được thu lại bằng webcam của máy tính hoặc từ một video và hình ảnh có sẵn mặt người.

## YÊU CẦU BÀI TOÁN

* Xây dựng thành công chương trình nhận diện khuôn mặt sử dụng phương pháp trích chọn đặc trưng LBP.
* Input: các dữ liệu bằng phương tiện hình ảnh/ webcam/ video có hình ảnh mặt người
* Output: phát hiện khuôn mặt, định danh khuôn mặt được khoanh vùng.
* Data training: >=1000 ảnh

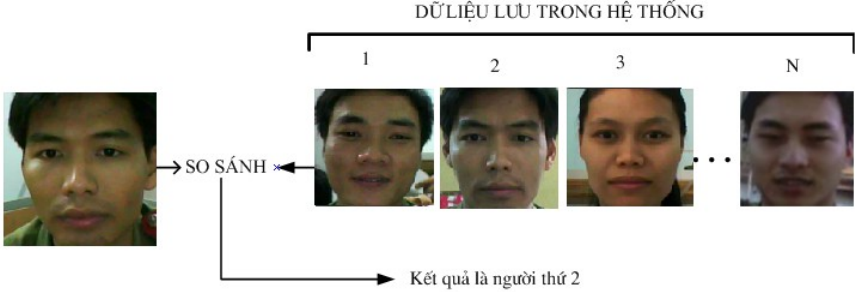
## BỐ CỤC BÁO CÁO

Bố cục báo cáo gồm 3 chương chính

* ***Chương 1: Giới thiệu***
* ***Chương 2: Cơ sở lý thuyết và phân tích thiết kế hệ thống***
* ***Chương 3: Cài đặt và đánh giá hệ thống***

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## MÔ HÌNH TỔNG QUAN

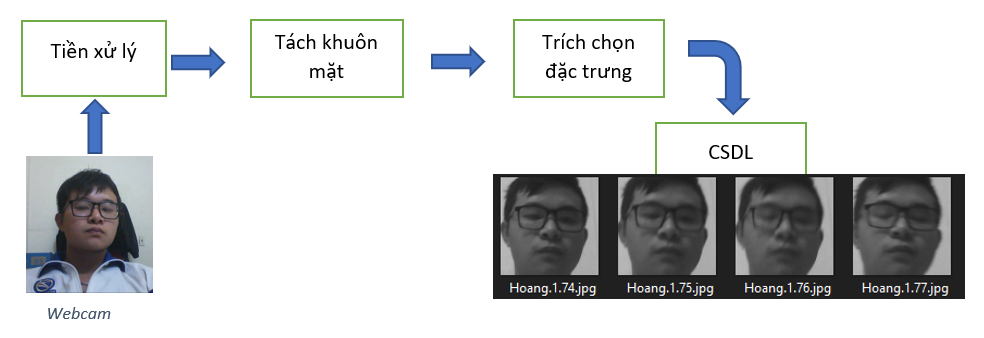


* Input: ảnh/ video chứa mặt người cần nhận dạng
* Output: trả về thông tin người vừa được đưa vào.

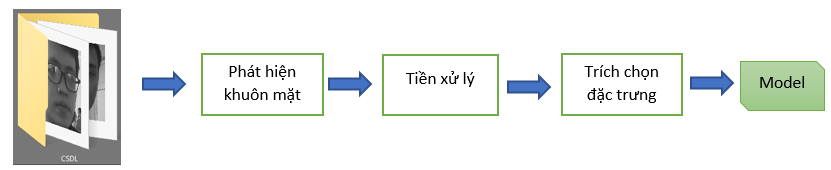
## MÔ HÌNH CHI TIẾT VÀ PHÂN TÍCH

### Mô hình chi tiết

#### Thu thập data

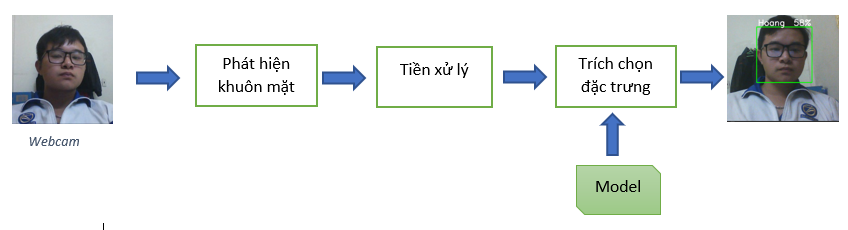


#### Training



Sau khi Training, ta được sản phẩm là model training với các dữ liệu phân lớp từng đối tượng trong cơ sở dữ liệu

#### Nhận dạng



### Phân tích

* Tiền xử lý: Bước này nhằm lọc nhiễu, nâng cao chất lượng của ảnh, video bao gồm: căn chỉnh ảnh, chuẩn hóa ánh sáng cho phù hợp với điều kiện phát hiện ảnh.
* Phát hiện và tách khuôn mặt: Bước này sẽ phát hiện và lấy ra tất cả khuôn mặt trong bức ảnh hoặc video. Tôi sử dụng “lbpcascade detector” thư viện đặc trưng có sẵn để phục vụ công việc phát hiện khuôn mặt.
* Trích chọn đặc trưng LBPH: cách thức hoạt động:
  + Thông tin LBP của pixel tại trung tâm của mỗi khối ảnh sẽ được tính dựa trên thông tin của các pixel lận cận. Có thể tóm tắt các bước tiến hành như sau:

**Bước 1:** Xác định bán kính làm việc.

**Bước 2:** Tính giá trị LBP cho pixel ở trung tâm (xc, yc) khối ảnh dựa trên thông tin của các pixel lân cận:

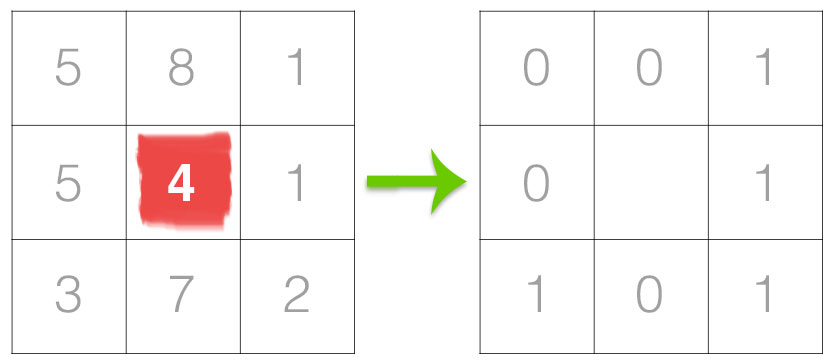
Trong đó, (gp) là giá trị grayscale của các pixel lân cận, (gc) là giá trị grayscale của các trung tâm và (s) là hàm nhị phân được xác định như sau: s(z) = 1 nếu giá trị z ≥0.



**Hình 2.1.3**. Tập hợp các điểm xung quanh Ptt.

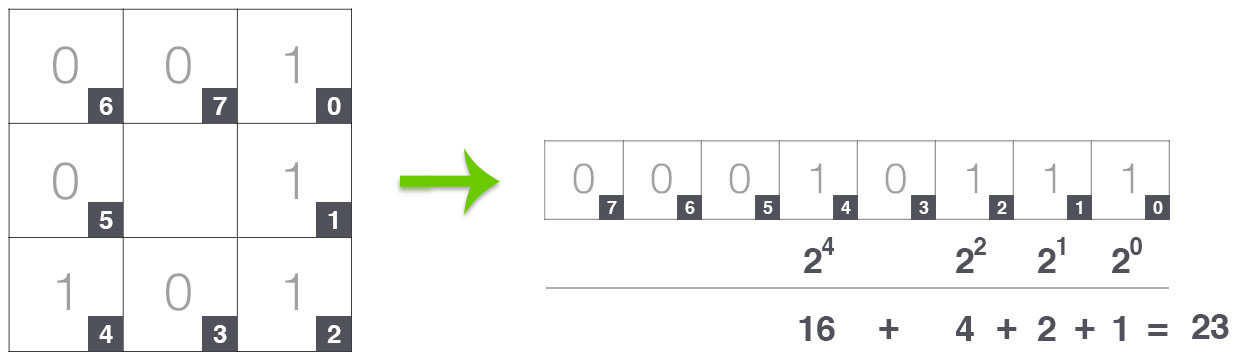
* Các bước tính giá trị LBP với ảnh xám.

**Bước 1 :** chuyển ảnh đầu vào thành ảnh xám. Đối với mỗi điểm ảnh trong ảnh xám, ta chọn một vùng lân cận kích thước r (1,2,4,..) xung quanh điểm ảnh trung tâm. Một giá trị LBP sẽ được tính toán cho điểm ảnh trung tâm này và được lưu trữ trong mảng 2D đầu ra với chiều rộng và chiều cao tương tự như hình ảnh đầu vào.



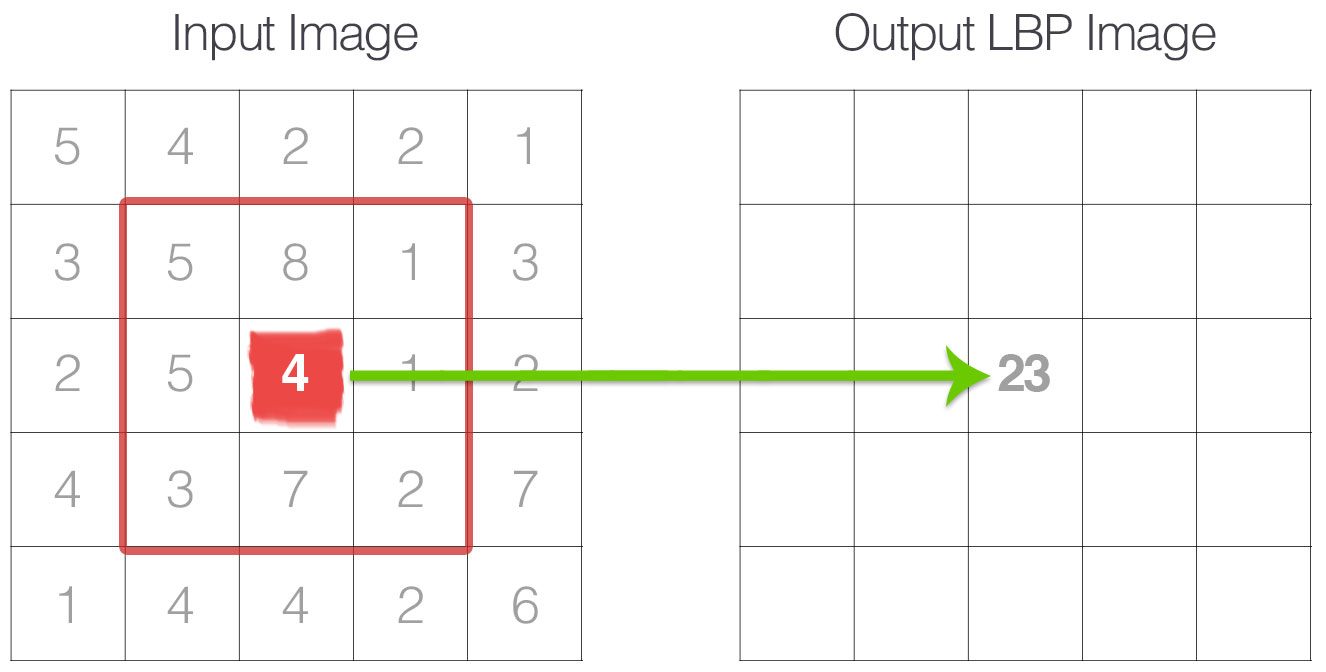
Trong hình trên, chúng tôi lấy điểm ảnh trung tâm (được đánh dấu bằng màu đỏ) và ngưỡng nó so với vùng lân cận của nó với r = 1 và có 8 phần tử (pixel) lân cận. Nếu cường độ của điểm ảnh trung tâm lớn hơn hoặc bằng với điểm ảnh lân cận của nó, thì chúng ta đặt giá trị thành 1; nếu không, ta đặt nó thành 0. Với 8 điểm ảnh xung quanh, chúng tôi có tổng cộng 2 ^ 8 = 256 kết hợp có thể có của mã LBP.

**Bước 2:** Chúng ta cần tính toán giá trị LBP cho điểm ảnh trung tâm. Chúng ta có thể bắt đầu từ bất kỳ pixel lân cận nào và làm việc theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ, nhưng thứ tự của chúng ta phải được giữ nhất quán cho tất cả các điểm ảnh trong hình ảnh của chúng ta và tất cả hình ảnh trong tập dữ liệu. Xét ma trận 3x3 ta có 8 điểm ảnh lân cận mà ta phải thực hiện biến đổi nhị phân. Kết quả nhị phân sau đó được lưu trữ trong một mảng 8-bit, tiếp theo chúng ta chuyển đổi thành số thập phân.

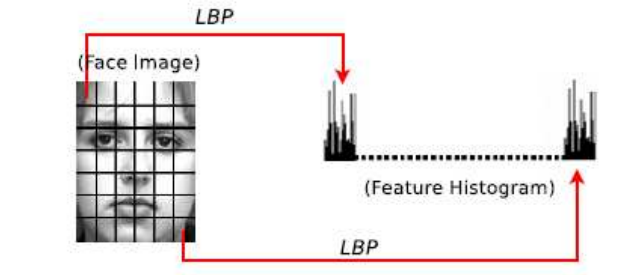


Trong ví dụ này, chúng ta bắt đầu ở điểm trên cùng bên phải và làm việc ngược chiều kim đồng hồ từ đó thu được 1 dãy nhị phân với giá trị 00010111. Sau đó chúng ta có thể chuyển đổi chuỗi nhị phân này thành thập phân được giá trị 23.

Giá trị này được lưu trữ trong mảng LBP như sau :



**Hình 2.1.6:** Giá trị LBP tính toán sau đó được lưu trữ trong một mảng đầu ra với chiều rộng và chiều cao tương tự như hình ảnh ban đầu.



* + Cân bằng histogram

# CHƯƠNG 3: CÀI ĐẶT VÀ ĐÁNH GIÁ HỆ THỐNG

## 3.1. NGÔN NGỮ LẬP TRÌNH, THƯ VIỆN SỬ DỤNG, NỀN TẢNG SỬ DỤNG

### 3.1.1. Ngôn ngữ lập trình: Python 3.8.3

Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics). Python hỗ trợ các module và gói (packages), khuyến khích chương trình module hóa và tái sử dụng mã. Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

Lý do lựa chọn Python:

* + - Ngữ pháp đơn giản, dễ đọc.
    - Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented)
    - Hỗ trợ module và hỗ trợ gói (package)
    - Xử lý lỗi bằng ngoại lệ (Exception)
    - Kiểu dữ liệu động ở mức cao.
    - Có các bộ thư viện chuẩn và các module ngoài, đáp ứng tất cả các nhu cầu lập trình.
    - Có khả năng tương tác với các module khác viết trên C/C++ (Hoặc Java cho Jython, hoặc .Net cho IronPython).
    - Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).
* Python dễ dàng kết nối với các thành phần khác: Python có thể kết nối với các đối tượng COM, .NET (Ironpython, Python for .net), và CORBA, Java… Python cũng được hỗ trợ bởi Internet Communications Engine (ICE) và nhiều công nghệ kết nối khác. Có thể viết các thư viện trên C/C++ để nhúng vào Python và ngược lại.
* Python là ngôn ngữ có khả năng chạy trên nhiều nền tảng: Python có cho mọi hệ điều hành: Windows, Linux/Unix, OS/2, Mac, Amiga, và những hệ điều hành khác. Thậm chí có cả những phiên bản chạy trên .NET, máy ảo Java, và điện thoại di động (Nokia Series 60). Với cùng một mã nguồn sẽ chạy giống nhau trên mọi nền tảng.
* Python rất đơn giản và dễ học: Python có cộng đồng lập trình rất lớn, hệ thống thư viện chuẩn, và cả các thư viện mã nguồn mở được chia sẻ trên mạng.
* Python là ngôn ngữ mã nguồn mở: Cài đặt Python dùng giấy phép nguồn mở nên được sử dụng và phân tối tự do, ngay cả trong việc thương mại. Giấy phép Python được quản lý bởi Python Software Foundation

### 3.1.2. Thư viện sử dụng: OpenCV

OpenCV (*Open Computer Vision*) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV đươc viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực. Opencv có các interface cho C/C++, Python Java vì vậy hỗ trợ được cho Window, Linux, MacOs lẫn Android. Đặc biết thư viện OpenCV rất dễ cài đặt và sử dụng.

OpenCV có rất nhiều ứng dụng:

* Nhận dạng ảnh
* Xử lý hình ảnh
* Phục hồi hình ảnh/video
* Thực tế ảo
* Kiểm tra và giám sát tự động
* Robot và xe hơi tự lái
* Phân tích hình ảnh y tế
* Phim - cấu trúc 3D từ chuyển động
* Nghệ thuật sắp đặt tương tác
* Các ứng dụng khác

Tính năng và các module phổ biến của OpenCV:

* Xử lý và hiển thị Hình ảnh/ Video/ I/O (core, imgproc, highgui)
* Phát hiện các vật thể (objdetect, features2d, nonfree)
* Geometry-based monocular hoặc stereo computer vision (calib3d, stitching, videostab)
* Computational photography (photo, video, superres)
* Machine learning & clustering (ml, flann)
* CUDA acceleration (gpu)

### 3.1.3. Nền tảng lập trình: Anaconda – IDE: Spyder

Anaconda là nền tảng (platform) mã nguồn mở về Khoa học dữ liệu (Data Science) trên Python thông dụng nhất hiện nay. Với hơn 6 triệu người dùng, Anaconda là cách nhanh nhất và dễ nhất để học Khoa học dữ liệu với Python hoặc R trên Windows, Linux và Mac OS X.

IDE Spyder:

* Spyder là một công cụ mã nguồn mở có sự công nhận cao trong thị trường IDE và phù hợp nhất với khoa học dữ liệu. Tên đầy đủ của Spyder là môi trường phát triển Python khoa học. Nó hỗ trợ tất cả các nền tảng quan trọng Linux, Windows và MacOS X.
* Nó cung cấp một tập hợp các tính năng như trình soạn thảo mã cục bộ, trình xem tài liệu, trình thám hiểm biến, bảng điều khiển tích hợp, v.v. và hỗ trợ các mô-đun khoa học như NumPy, SciPy, v.v.
* Tính năng, đặc điểm:
  + Làm nổi bật cú pháp đúng và hoàn thành mã tự động
  + Tích hợp mạnh mẽ với Python console
  + Hoạt động tốt trong chế độ chỉnh sửa đa ngôn ngữ và chế độ hoàn thành mã tự động

## 3.2. CÀI ĐẶT HỆ THỐNG

### 3.2.1. Hàm thu thập data:

- Giao diện thu thập ảnh + training:

+ Button “Lấy ảnh” sẽ gọi hàm take\_img()  
+ Button “Đào tạo” gọi hàm trainimg()

Với mỗi ID, phần mềm sẽ thu nhận tối đa 200 ảnh khuôn mặt. Sau khi có đủ số sinh viên đăng ký. Người quản lý có thể ấn đào tạo để cho máy học.



def take\_img(): #Hàm lấy ảnh

cam = cv2.VideoCapture(0) # bật webcam

#khai báo đối tượng detector dùng để phát hiện khuôn mặt

detector = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

ID = txt.get()

Name = txt2.get()

sampleNum = 0 #biến số đại diện số lượng ảnh thu được

# đọc frame webcam video

while (True):

ret, img = cam.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = detector.detectMultiScale(gray, 1.3, 5)

#vẽ khối hình chữ nhật bao quanh khuôn mặt

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

# tăng biến

sampleNum = sampleNum + 1

# Lưu ảnh vừa cap vào thư mục với định dạng tên: Name.ID.[chỉ số ảnh]

cv2.imwrite("dataset/ " + Name + "." + ID + '.' + str(sampleNum) + ".jpg",

gray[y:y + h, x:x + w])

cv2.imshow('Frame', img)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

elif sampleNum > 200:

break

cam.release()

cv2.destroyAllWindows()

#Thông báo lưu trữ thành công!

res = "Hoàn thành lưu trữ ID : " + ID + " - Tên : " + Name Notification.configure(text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))

Notification.place(x=250, y=400)

### 3.2.2. Training:

def trainimg(): #hàm train Img

recognizer = cv2.face.LBPHFaceRecognizer\_create()

global detector

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

try:

global faces,Id

faces, Id = getImagesAndLabels("dataset")

except Exception as e:

l='please make "dataset" folder & put Images'

Notification.configure(text=l, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))

Notification.place(x=350, y=400)

recognizer.train(faces, np.array(Id))

try:

recognizer.save("model/trained\_model2.yml")

except Exception as e:

q='Please make "model" folder'

Notification.configure(text=q, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))

Notification.place(x=350, y=400)

res = "Model Trained" # +",".join(str(f) for f in Id)

Notification.configure(text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))

Notification.place(x=250, y=400)

def getImagesAndLabels(path):

imagePaths = [os.path.join(path, f) for f in os.listdir(path)]

# create empth face list

faceSamples = []

# create empty ID list

Ids = []

# now looping through all the image paths and loading the Ids and the images

for imagePath in imagePaths:

# loading the image and converting it to gray scale

pilImage = Image.open(imagePath).convert('L')

# Now we are converting the PIL image into numpy array

imageNp = np.array(pilImage, 'uint8')

# getting the Id from the image

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

# extract the face from the training image sample

faces = detector.detectMultiScale(imageNp)

# If a face is there then append that in the list as well as Id of it

for (x, y, w, h) in faces:

faceSamples.append(imageNp[y:y + h, x:x + w])

Ids.append(Id)

return faceSamples, Ids

window.grid\_rowconfigure(0, weight=1)

window.grid\_columnconfigure(0, weight=1)

### 3.2.3. Nhận dạng mặt người:

# khởi tạo biến recognizer dùng để ghi thông tin file training

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read('model/trained\_model2.yml')

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath)

font = cv2.FONT\_HERSHEY\_SIMPLEX

id = 0

# danh sách các sinh viên đã đăng ký học phần theo STT danh sách lớp (bắt đầu từ 1)

names = ['None','HoangDang’,'Giap','Kien',’Huong’,’KhacHoang’]

cam = cv2.VideoCapture(0)

while True:

ret, img =cam.read()

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(

gray,

scaleFactor = 1.2,

minNeighbors = 5

)

for(x,y,w,h) in faces:

cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

# If confidence is less them 100 ==> "0" : perfect match

if (confidence < 100):

id = names[id]

print(confidence)

confidence = " {0}%".format(round(100 - confidence))

else:

id = "unknown"

confidence = " {0}%".format(round(100 - confidence))

cv2.putText(img, str(id)+str(confidence), (x+5,y-5), font, 1, (255,255,255), 2 )

cv2.imshow('camera',img)

k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video

if k == 27:

break

cam.release()

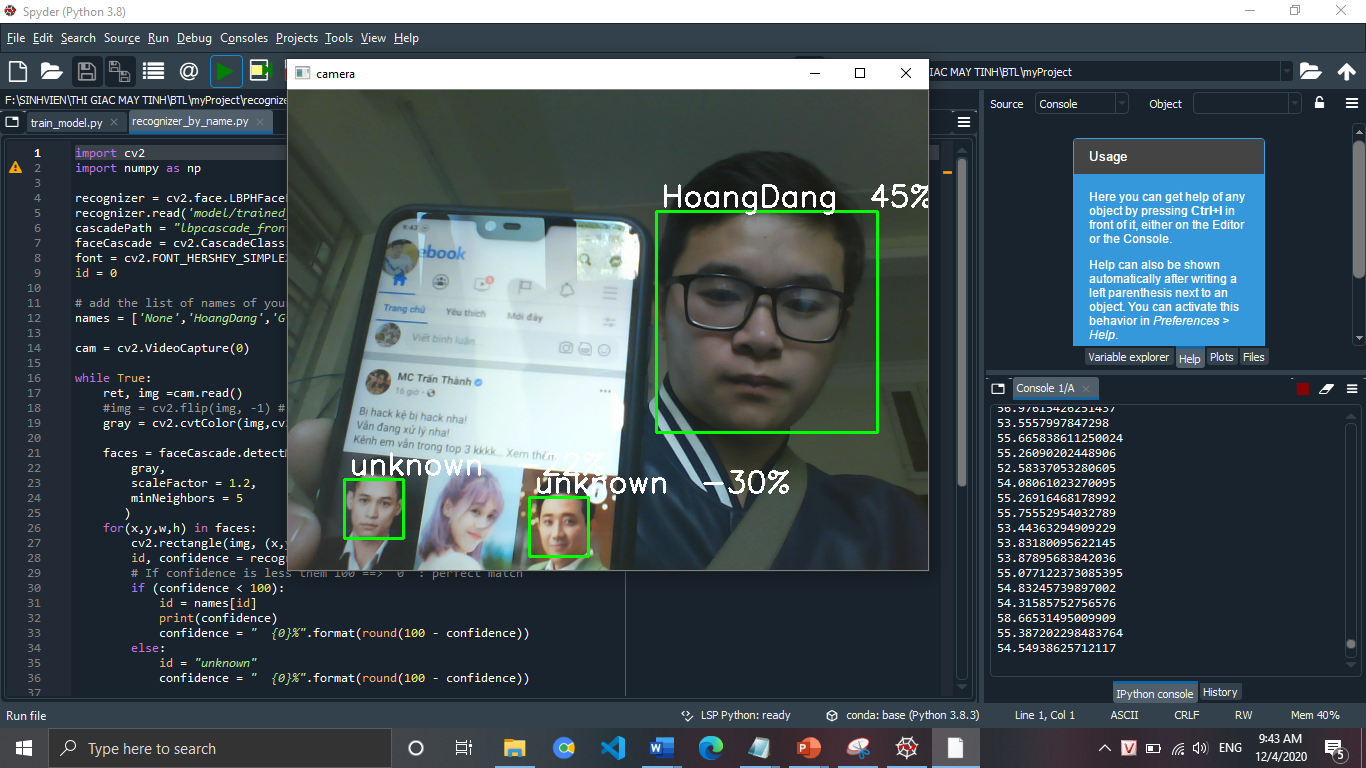
cv2.destroyAllWindows()

### 3.2.4. Kết quả nghiệm thu

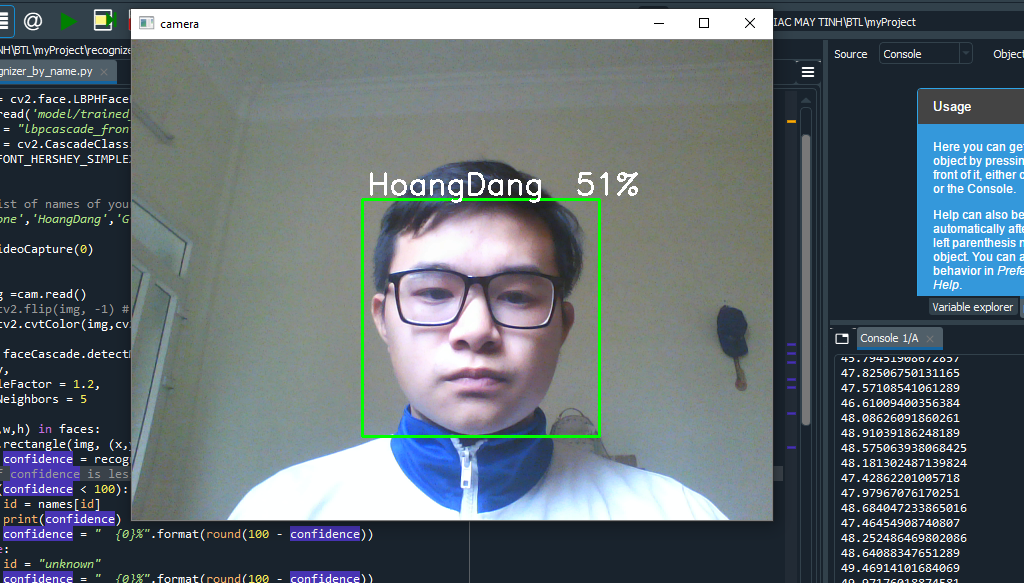
- Số lượng ảnh (data) hiện có là 1200 ảnh tương đương 6 người. Thời gian thiết lập đăng ký ảnh mất khoảng 5-10s tùy thuộc vào điều kiện ánh sáng.

- Thời gian training 1200 ảnh là khoảng 18s.

- Độ chính xác khoảng 95% khi trong điều kiện vừa đủ ánh sáng



Trường hợp chưa thiết lập khuôn mặt



## 3.3. ĐÁNH GIÁ HỆ THỐNG – KẾT LUẬN

### 3.3.1. Ưu điểm:

* Thời gian training khá nhanh (18s cho 1200 ảnh)
* Xác thực khá chính xác
* Code dễ hiểu, không bị dài dòng, rõ ràng.

### 3.3.2. Nhược điểm:

* Vì bước đăng ký ảnh bao gồm nhiều công việc như tách ảnh, trích rút nhị phân, nên nếu trong trường hợp ánh sáng không đủ, có thể gây giật lag, đóng băng phần mềm.
* Vì là một phương pháp nhận diện cũ, nên tôi cũng không đánh giá cao về phương pháp này

### 3.3.3. Hướng phát triển

* Cải thiện tốc độ training, ghi nhận data.
* Phát triển và cài đặt phần mềm điểm danh cho sinh viên đến lớp hoặc chống gian lận trong thi cử.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Nguồn tham khảo:** Github, Youtube, Google.

**Ghé thăm trang Github của tôi:** Github.com/hoangdvhp99/ThiGiacMayTinh