01\_face\_dataset

Soạn

cam = cv2.VideoCapture(0) : Khởi tạo camera máy tính

cam.set(3, 640) *# set video width* : 3 là định danh thuộc tính chiều rộng của camera, 640 là thuộc tính pixel của cameracam.set(4, 480) *# set video height* : 4 là định thanh thuộc tính chiều cao của camera, 480 là thuộc tính pixel của ảnh

* Kiểm soát độ phân giải của camera 480\*640

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

face\_detector: là một lớp đối tượng của

**CascadeClassifier : là thuật toán huấn luyện trong đó đọc các ảnh vào**

“**cv2.CascadeClassifier**”được sử dụng để phát hiện các đặc điểm của khuôn mặt trong camera.

haarcascade\_frontalface\_default.xml : là đường dẫn tới tệp XML chứa thông tin cần thiết cho việc phát hiện khuôn mặt, là một bộ phân loại được cung cấp sẵn trong OpenCV, chứa các thông số và mẫu để nhận diện khuôn mặt trong các hình ảnh.chứa các mẫu dữ liệu học máy mục đích huấn luyện cho máy biết đâu là mặt, đâu không phải là mặt từ những dữ liệu có khuôn mặt và không có khuôn mặt

face\_id = input('\n enter user id end press <return> ==> ') : Nhập id khuôn mặt cần nhận dạng sau khi chạy code.

ret, img = cam.read()  
img = cv2.flip(img, 1) *# flip video image vertically*

* Đọc ảnh và lật ảnh theo chiều ngang

**Haar cascades**: Về cơ bản là sử dụng các đặc trưng loại Haar và sau đó sử dụng thật nhiều đặc trưng đó qua nhiều lượt (cascade) để tạo thành một cỗ máy nhận diện hoàn chỉnh

sử dụng bộ lọc Haar- CNN: mạng nơ ron tích chập. Ở CNN, bộ lọc chiếm toàn bộ cửa sổ trượt, trong khi ở đặc trưng Haar, bộ lọc chỉ chiếm một phần trong cửa sổ trượt thôi

Điều này được thực hiện bằng phương pháp

Classifie: phân loại

cv2::CascadeClassifier::detectMultiScale, trả về các hình chữ nhật biên cho các khuôn mặt được phát hiện (tức là x, y, w, h). Nó có hai tham số là scaleFactor và minNeighbors. ScaleFactor xác định yếu tố tăng kích thước cửa sổ ban đầu bắt đầu từ kích thước "minSize" và sau khi kiểm tra tất cả các cửa sổ có kích thước đó, cửa sổ được thu nhỏ bởi "scaleFactor" và kích thước cửa sổ tăng lên "maxSize". Nếu "scaleFactor" lớn, (ví dụ: 2.0), sẽ có ít bước hơn, do đó việc phát hiện sẽ nhanh hơn, nhưng chúng tôi có thể bỏ lỡ các đối tượng có kích thước nằm giữa hai thang đo được thử nghiệm. (hệ số tỷ lệ mặc định là 1.3). Các giá trị của "minNeighbors" càng cao, số lượng dương tính giả sẽ ít hơn và sẽ ít lỗi hơn về mặt phát hiện sai khuôn mặt. Tuy nhiên, cũng có khả năng bỏ sót một số dấu vết khuôn mặt không rõ ràng.

02\_face\_trainning

**Local Binary Patterns Historgam:**

* **Chia hình ảnh thành các ô nhỏ (vùng)**: Hình ảnh được chia thành các ô nhỏ hơn. Mỗi ô thường là một vùng hình vuông có kích thước nhất định.
* **Tính toán các giá trị nhị phân (binary patterns) trong mỗi ô**: Đối với mỗi ô, LBPH so sánh giá trị của các pixel trong ô với giá trị trung tâm của nó. Nếu giá trị pixel lớn hơn hoặc bằng giá trị trung tâm, nó được gán là 1; ngược lại, nó được gán là 0. Sau đó, chuỗi các giá trị nhị phân này được chuyển thành một số nguyên.
* **Tính toán histogram của các giá trị nhị phân**: Các giá trị nguyên được tính được sử dụng để tạo histogram cho mỗi ô.
* **So sánh histogram của các ô**: Cuối cùng, các histogram của các ô được so sánh với nhau để nhận diện đặc điểm của khuôn mặt hoặc đối tượng.

**compute\_lbp\_pixel** để tính toán Local Binary Patterns (LBP) của một pixel trong ảnh.

Hàm này thực hiện các bước sau để tính toán LBP của pixel tại vị trí **(x, y)**:

Lấy giá trị của pixel tại vị trí **(x, y)** làm giá trị trung tâm (**center**).

So sánh giá trị của các pixel xung quanh với giá trị trung tâm (**center**), nếu giá trị của pixel xung quanh lớn hơn hoặc bằng **center**, ta gán bit 1, ngược lại gán bit 0.

Kết hợp các bit đã tính được từ các pixel xung quanh để tạo thành một chuỗi nhị phân đại diện cho LBP của pixel tại vị trí **(x, y)**.

Trả về giá trị của chuỗi nhị phân này.

Ví dụ: Nếu giá trị của các pixel xung quanh **(x, y)** lớn hơn hoặc bằng giá trị trung tâm, ta gán bit 1, ngược lại gán bit 0, ta sẽ thu được một số nguyên đại diện cho LBP của pixel tại vị trí **(x, y)**.

Hàm **compute\_lbp\_histogram** thực hiện các bước sau:

Khởi tạo một histogram **hist** có 256 bin (vì có 256 giá trị LBP có thể có).

Duyệt qua từng pixel trong vùng ảnh được xác định bởi **x**, **y**, **width**, và **height**.

Tính toán giá trị LBP cho mỗi pixel bằng cách sử dụng hàm **compute\_lbp\_pixel**.

Tăng giá trị của bin tương ứng với giá trị LBP được tính toán.

Trả về histogram **hist** đã tính toán.

Histogram này sẽ biểu diễn phân bố của các giá trị LBP trong vùng ảnh đã chọn

1. **cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)**: Vẽ một hộp giới hạn xung quanh khuôn mặt được phát hiện bằng cách sử dụng hàm **rectangle** từ thư viện OpenCV (**cv2**). Điều này sẽ vẽ một hộp màu xanh lá cây với độ dày là 2 pixel.