ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

…☼…

BÁO CÁO BÀI TẬP LỚN

MÁY HỌC CƠ BẢN VÀ ỨNG DỤNG

ĐỀ TÀI:

ĐIỀU KHIỂN CON TRỎ CHUỘT BẰNG CỬ ĐỘNG KHUÔN MẶT

Giảng viên hướng dẫn: Thầy Nguyễn Khánh Lợi

Thành phố Hồ Chí Minh – 2023

PHÂN CÔNG CÔNG VIỆC

Sinh viên thực hiện	MSSV	Công việc	Đánh giá	Ký tên
Phạm Huỳnh Quốc Thiện	2012100	Viết báo cáo, nghiên cứu lý thuyết, code	100%	
Nguyễn Thái Toàn	1912228	Viết báo cáo, nghiên cứu lý thuyết, code	100%	

TÓM TẮT ĐỀ TÀI

Mục tiêu đề tài là tạo một chương trình sử dụng camera máy tính để thu thập video thời gian thực trong đó có chứa mặt người. Nhận diện khuôn mặt người dùng thư viện dlib và tập train shape_predictor_68_face_landmarks. Từ đó thực hiện các thao tác di chuyển con trỏ chuột màn hình tương ứng với động tác trên khuôn mặt.

MỤC LỤC

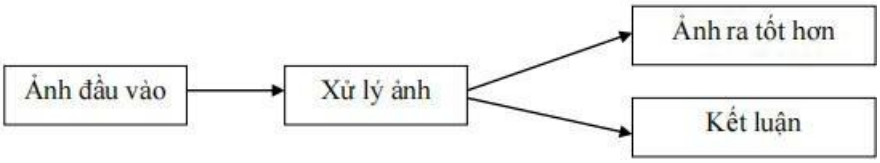
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VỀ XỬ LÝ ẢNH

1. Giới thiệu về xử lý ảnh

Xử lý ảnh là một lĩnh vực mang tính khoa học và công nghệ. Nó là một ngành khoa học mới mẻ so với nhiều ngành khoa học khác nhưng tốc độ phát triển của nó rất nhanh, kích thích các trung tâm nghiên cứu, ứng dụng, đặc biệt là máy tính chuyên dụng riêng cho nó.

Các phương pháp xử lý ảnh bắt đầu từ các ứng dụng chính: nâng cao chất lượng ảnh và phân tích ảnh. Ứng dụng đầu tiên được biết đến là nâng cao chất lượng ảnh báo được truyền qua cáp từ Luân đôn đến New York từ những năm 1920. Vấn đề nâng cao chất lượng ảnh có liên quan tới phân bố mức sáng và độ phân giải của ảnh. Việc nâng cao chất lượng ảnh được phát triển vào khoảng những năm 1955. Điều này có thể giải thích được vì sau thế chiến thứ hai, máy tính phát triển nhanh tạo điều kiện cho quá trình xử lý ảnh số thuận lợi. Năm 1964, máy tính đã có khả năng xử lý và nâng cao chất lượng ảnh từ mặt trăng và vệ tinh Ranger 7 của Mỹ bao gồm: làm nổi đường biên, lưu ảnh. Từ năm 1964 đến nay, các phương tiện xử lý, nâng cao chất lượng, nhận dạng ảnh phát triển không ngừng. Các phương pháp tri thức nhân tạo như mạng nơron nhân tạo, các thuật toán xử lý hiện đại và cải tiến, các công cụ nén ảnh ngày càng được áp dụng rộng rãi và thu nhiều kết quả khả quan.

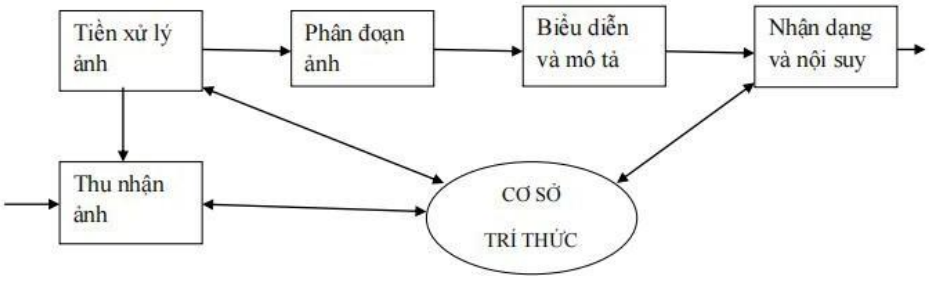
Xử lý ảnh đóng vai trò quan trọng trong tương tác giữa người với máy tính. Quá trình xử lý nhận dạng ảnh là một quá trình gồm các thao tác nhằm biến đổi một ảnh đầu và để cho ra một kết quả hoặc một kết luận.



Hình 1 Quá trình xử lý ảnh

Đầu tiên, ảnh tự nhiên từ thế giới ngoài được thu nhận qua các thiết bị thu (như Camera, máy chụp ảnh). Sau đó, qua Xử lý ảnh thì ảnh được chuyển trực tiếp thành ảnh số tạo thuận lợi cho xử lý tiếp theo. Kết quả của xử lý ảnh có thể là: cho ra một ảnh tốt hơn theo mong muốn của người dùng; Phân tích ảnh để thu được thông tin để phân loại ảnh và nhận biết ảnh; Rút ra những nhận xét, kết luận .v.v…

1.1 Các bước cơ bản trong quá trình xử lý ảnh



Hình 2 Các bước cơ bản trong quá trình xử lý ảnh

1.1.1 Thu nhận ảnh (Image Acquisition)

Ảnh có thể thu nhận qua máy ảnh màu hoặc trắng đen, máy quét ảnh, máy quay,.v.v… Chất lượng một ảnh thu nhận được phụ thuộc vào thiết bị thu, vào môi trường (ánh sáng, phong cảnh). Sau đó, ảnh được chuyển đổi ADC(số hóa ảnh) .Quá trình chuyển đổi ADC(Analog to Digital Converter) để thu nhận dạng số hóa của ảnh.

1.1.2 Tiền xử lý ảnh (Image Processing)

Sau bộ thu nhận, ảnh có thể nhiễu độ tương phản thấp nên cần đưa vào bộ tiền xử lý để nâng cao chất lượng. Chức năng chính của bộ tiền xử lý ảnh là lọc nhiễu, nâng độ

tương phản để làm ảnh rõ hơn, nét hơn. Ảnh sẽ được cải thiện về độ tương phản, khử nhiễu, khôi phục ảnh, nắn chỉnh hình học, ...

- Khử nhiễu: nhiễu có hai loại: nhiễu hệ thống và nhiễu ngẫu nhiên. Đặc trưng của nhiễu hệ thống là tính tuần hoàn nên có thể khử nhiễu bằng việc sử dụng phép biến đổi Fourier và loại bỏ các đỉnh điểm. Nhiễu ngẫu nhiên có thể được khử bằng phương pháp nội suy, lọc trung vị và lọc trung bình.

- Chỉnh mức xám: là chỉnh sửa tính không đồng đều của thiết bị thu nhận hoặc độ tương phản giữa các vùng ảnh.

- Chỉnh tán xạ ảnh: Ảnh thu được từ các thiết bị quang học hay điện tử có thể bị mờ, nhòe ảnh. Phương pháp biến đổi Fourier dựa trên tích chập của ảnh với hàm tán xạ.

1.1.3 Phân đoạn ảnh (Segmentation)

Phân đoạn ảnh là tách một ảnh đầu vào thành các vùng thành phần để biểu diễn phân tích, nhận dạng ảnh. Ví dụ: để nhận dạng chữ (hoặc mã vạch) trên phong bì thư cho mục đích phân loại bưu phẩm, cần chia các câu, chữvề địa chỉ hoặc tên người thành các từ, các chữ, các số(hoặc các vạch) riêng biệt đểnhận dạng. Đây là phần phức tạp khó khăn nhất trong xử lý ảnh và cũng dễ gây lỗi, làm mất độ chính xác của ảnh. Kết quả nhận dạng ảnh phụ thuộc rất nhiều vào công đoạn này. Kết quả của việc phân đoạn ảnh thường là các dữ liệu điểm ảnh thô, hàm chứa biên của một vùng ảnh hoặc tập hợp tất cả các điểm ảnh trong một vùng ảnh đó.

1.1.4 Biểu diễn và mô tả ảnh (Image Representation and Description)

Biểu diễn ảnh: Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của vùng ảnh (ảnh đã phân đoạn) cộng với mã liên kết với các vùng lận cận. Việc biến đổi các số liệu này thành dạng thích hợp là cần thiết cho xử lý tiếp theo bằng máy tính. Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng (Feature Selection) gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được.

Mô tả ảnh: Ảnh sau khi được số hóa sẽ được lưu vào bộ nhớ hoặc chuyển sang các khâu tiếp theo để phân tích ảnh. Nếu lưu trữ ảnh trực tiếp từ ảnh thô thì đòi hỏi dung lượng bộ nhớ rất lớn và không hiệu quả cho các ứng dụng sau này. Thông thường ,các ảnh thô đó được biểu

diễn hay mã hóa lại theo các đặc điểm của ảnh được gọi là các đặc trưng như: biên ảnh, vùng ảnh. Một số phương pháp biển diễn ảnh:

Biểu diễn ảnh bằng mã chạy(Run-length Code): thường biểu diễn cho vùng ảnh và áp dụng cho ảnh nhị phân. Một vùng ảnh R có thể mã hoá đơn giản nhờ một ma trận nhị phân:

Trong đó: U(m, n) là hàm mô tả mức xám ảnh tại tọa độ (m, n).Với cách biểu diễn trên, một

- U(m, n)= 1 , nếu (m, n) ∈ R
- U(m, n)= 0 , nếu (m, n) không ∈ R

vùng ảnh được mô tả bằng một tập các chuỗi số 0 hoặc 1. Giả sử chúng ta mô tả ảnh nhị phân của một vùng ảnh được thể hiện theo toạ độ (x, y) theo các chiều và đặc tả chỉ đối với giá trị"1" khi đó dạng mô tả có thể là: (x, y) r; trong đó (x, y) là toạ độ, r là số lượng các bit có giá trị"1" liên tục theo chiều ngang hoặc dọc.

Biểu diễn ảnh bằng mã xích (Chaine- Code): thường dùng để biển diễn đường biên ảnh. Một đường biên ảnh bất kỳ được chia thành các đoạn nhỏ. Nối các điểm chia, ta có các đoạn thẳng kế tiếp được gán hướng cho đoạn thẳng đó tạo thành một dây xích gồm các đoạn. Các hướng có thể chọn 4, 8, 12, 24,… mỗi hướng được mã hoá theo số thập phân hoặc số nhị phân thành mã của hướng.

Biểu diễn ảnh bằng mã tứ phân(Quad- Tree Code): thường dùng để mã hóa cho các vùng ảnh. Vùng ảnh đầu tiên được chia làm bốn phần thường là bằng nhau. Nếu mỗi vùng đã đồng nhất (chứa toàn điểm đen (1) hay trắng (0)), thì gán cho vùng đó một mã và không chia tiếp. Các vùng không đồng nhất được chia tiếp làm bốn phần theo thủ tục trên cho đến khi tất cả các vùng đều đồng nhất. Các mã phân chia thành các vùng con tạo thành một cây phân chia các vùng đồng nhất.

1.1.5 Nhận dạng và nội suy ảnh (Image Recognition & Interpretation)

Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được lọc (hoặc lưu) từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng. Ví dụ: một loạt chữ số và nét gạch ngang trên phong bì thư có thể được nội suy thành mã điện thoại. Có nhiều cách phân loai ảnh khác nhau về ảnh. Theo lý thuyết về nhận dạng, các mô hình toán học về ảnh được phân theo hai loại nhận dạng ảnh cơ bản:

- Nhận dạng theo tham số

- Nhận dạng theo cấu trúc

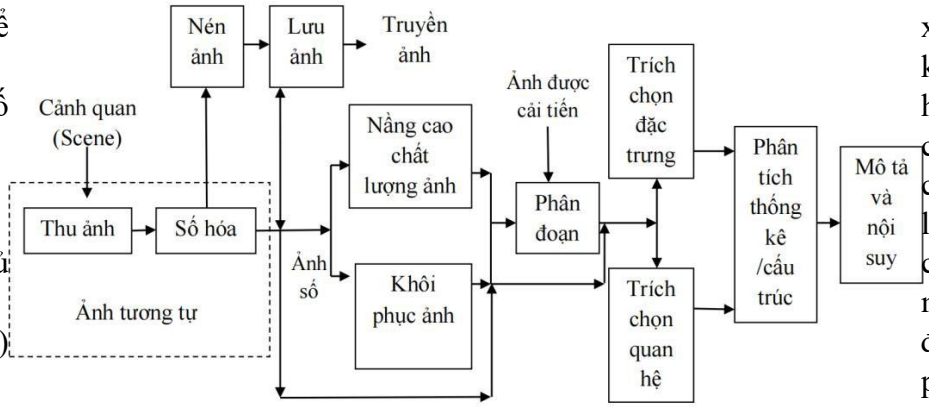
Một số đối tượng nhận dạng khá phổ biến hiện nay đang được áp dụng trong khoa học và công nghệ là: nhận dạng ký tự(chữ in, chữ viết tay, chữ ký điện tử), nhận dạng văn bản (Text), nhận dạng vân tay, nhận dạng mã vạch, nhận dạng mặt người…

1.1.6 Cơ sở tri thức (Knowledge Base)

Ảnh là một đối tượng khá phức tạp về đường nét, độ sáng tối, dung lượng điểm ảnh, môi trường để thu ảnh phong phú kéo theo nhiễu. Trong nhiều khâu xử lý ảnh và phân tích ảnh ngoài việc đơn giản hóa các phương pháp toán học đảm bảo tiện lợi cho xử lý, người ta mong muốn bắt chước quy trình tiếp nhận và xử lý ảnh theo cách của con người. Trong các bước xử

lý ảnh đó, nhiều khâu hiện nay đã xử lý theo các phương pháp trí tuệ con người. Vì vậy, ở đây các cơ sở tri thức được phát huy.

Trong thực tế, các quá trình sử dụng ảnh số không nhất thiết phải qua hết các khâu đó tùy theo đặc điểm ứng dụng. Ảnh sau khi được số hóa được nén, và lưu lai để truyền đi cho các hệ thống khác sử dụng hoặc để xử lý tiếp theo. Mặt khác, ảnh sau khi số hóa có thể bỏ qua công đoạn nâng cao chất lượng(khi ảnh đủ chất lượng theo một yêu cầu nào đó) để chuyển tới khâu phân đoạn hoặc bỏ tiếp khâu phân đoạn chuyển trực tiếp tới khâu trích chọn đặc trưng.



Hình 3 Quá trình xử lý ảnh

Việc trích chọn hiệu quả các đặc điểm giúp cho việc nhận dạng các đối tường ảnh chính xác, với tốc độ tính toán cao và dung lượng lưu trữ được giảm xuống. Các đặc điểm của đối tượng được trích chọn tùy theo mục đích nhận dạng trong quá trình xử lý ảnh. Một số đặc điểm của ảnh:

- Đặc điểm không gian: phân bố mức xám, xác suất, biên độ, điểm uốn, …

- Đặc điểm biến đổi: các đặc điểm được trích chọn bằng việc thực hiện lọc vùng (Zonal filtering). Các mặt nạ đặc điểm(feature mask) thường là các khe hẹp với hình dạng khác nhau(hình chữ nhật, hình vuông, hình tam giác.v.v…).

- Đặc điểm biên và đường biên: là đặc trưng cho đường biên của đối tượng và trích chon các thuộc tính bất biến được dùng khi nhận dạng đối tượng. Nhờ sử dụng các phương pháp toán tử Laplace, toán tử Gradient, toán tử La bàn, toán tử chéo không(zero crossing).v.v…

1.2 Một số vấn đề cơ bản trong xử lý ảnh

1.2.1 Điểm ảnh (Picture Element)

Gốc của ảnh(ảnh tự nhiên) là ảnh liên tục về không gian và độ sáng. Để xử lý bằng máy tính thì ảnh cần phải được số hoá. Số hoá ảnh là sự biến đổi gần đúng một ảnh liên tục thành một tập điểm phù hợp với ảnh thật về vị trí(không gian) và độ sáng(mức xám). Khoảng cách giữa các điểm ảnh đó được thiết lập sao cho mắt người không phân biệt được ranh giới giữa chúng. Mỗi một điểm như vậy gọi là điểm ảnh (PEL: Picture Element) hay gọi tắt là Pixel. Trong khuôn khổ ảnh hai chiều, mỗi pixel ứng với cặp tọa độ (x, y).

Định nghĩa: Điểm ảnh(Pixel) là một phần tử của ảnh số tại toạ độ(x, y) với độ xám hoặc màu nhất định. Kích thước và khoảng cách giữa các điểm ảnh đó được chọn thích hợp sao cho mắt người cảm nhận sự liên tục về không gian và mức xám(hoặc màu) của ảnh số gần như ảnh thật. Mỗi phần tử trong ma trận được gọi là một phần tử ảnh.

1.2.2 Độ phân giải của ảnh (Resolution)

Độ phân giải(Resolution) của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị. Khoảng cách giữa các điểm ảnh phải được chọn sao cho mắt người vẫn thấy được sự liên tục của ảnh. Việc lựa chọn khoảng cách thích hợp tạo nên một mật độ phân giải và được phân bố theo trục x và y trong không gian hai chiều.

1.2.3 Mức xám của ảnh

Một điểm ảnh(pixel) có hai đặc trưng cơ bản là vị trí (x, y) của điểm ảnh và độ xám của nó. Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại điểm đó. Các thang giá trị mức xám thông thường: 16, 32, 64, 128, 256 (Mức 256 là mức thông dụng. Lý do từ kỹ thuật máy tính dùng 1 byte (8 bit) để biểu diễn mức xám: Mức xám dùng 1 byte để biểu diễn: $2^8 = 256$ mức, tức là từ 0 đến 255).

Ảnh đen trắng (ảnh xám): là ảnh có hai màu đen, trắng(không chứa màu khác) với mức xám ở các điểm ảnh có thể khác nhau.

Ảnh nhị phân: ảnh chỉ có 2 mức đen trắng phân biệt. tức dùng 1 bit mô tả 2^1 mức khác nhau. Nói cách khác mỗi điểm ảnh của ảnh nhị phân chỉ có thể là 0 hoặc 1.

Ảnh màu: trong khuôn khổ lý thuyết ba màu(Red, Blue, Green) để tạo nên thế giới màu, người ta thường dùng 3 byte để mô tảmức màu, khi đó các giá trị màu: $2^8*3= 2^{24} \approx$16,7 triệu màu.

1.2.4 Định nghĩa ảnh số

Ảnh số là tập hợp các điểm ảnh với mức xám phù hợp dùng để mô tả ảnh gần với ảnh thật.

1.2.5 Quan hệ giữa các điểm ảnh

Một ảnh số giả sử được biểu diễn bằng hàm f(x, y).Tập con các điểm ảnh là S, cặp điểm ảnh có quan hệ với nhau ký hiệu là p, q.

- Các lân cận của điểm ảnh (Image Neighbors):

Giả sử có điểm ảnh p tại toạ độ (x, y), p có 4 điểm lân cận gần nhất theo chiều đứng và ngang (có thể coi như lân cận 4 hướng chính: Đông, Tây, Nam, Bắc).

	Đông ← x → Tây		
Nam ↕	(x-1, y-1)	(x, y-1)	(x+1,y-1)
y	(x-1, y)	(x, y)	(x+1, y)
Bắc	(x-1, y+1)	(x, y+1)	(x+1, y+1)

trong đó: số 1 là giá trị logic; $N_4(p)$ tập 4 điểm lân cận của p.

Các lân cận chéo: Các điểm lân cận chéo NP(p) (Có thể coi lân cận chéo là 4 hướng: Đông-Nam, Đông-Bắc, Tây-Nam, Tây-Bắc)

$N_P(p)$ = { (x+1, y+1);(x+1, y-1);(x-1, y+1);(x-1, y-1)}

Tập kết hợp: $N_8(p) = N_4(p) + N_P(p)$ là tập hợp 8 lân cận của điểm ảnh p. Chú ý: Nếu (x, y) nằm ở biên (mép) ảnh; một số điểm sẽ nằm ngoài ảnh.

- Các mối liên kết điểm ảnh:

Các mối liên kết được sử dụng để xác định giới hạn(Boundaries) của đối tượng vật thể hoặc xác định vùng trong một ảnh. Một liên kết được đặc trưng bởi tính liền kề giữa các điểm và mức xám của chúng. Giả sử V là tập các giá trị mức xám. Một ảnh có các giá trị cường độ sáng từ thang mức xám từ 32 đến 64 được mô tả như sau : V={ 32, 33, … , 63, 64}. Có 3 loại liên kết:

Liên kết 4: Hai điểm ảnh p và q được nói là liên kết 4 với các giá trị cường độ sáng V nếu q nằm trong một các lân cận của p, tức q thuộc $N_4(p)$.

Liên kết 8: Hai điểm ảnh pvà qnằm trong một các lân cận 8 của p, tức q thuộc $N_8(p)$.

Liên kết m(liên kết hỗn hợp): Hai điểm ảnh p và q với các giá trị cường độ sáng V được nói là liên kết m nếu: q thuộc $N_4(p)$ hoặc q thuộc $N_P(p)$.

- Đo khoảng cách giữa các điểm ảnh:

Định nghĩa: Khoảng cách D (p, q) giữa hai điểm ảnh p toạ độ (x, y), q toạ độ (s, t) là hàm khoảng cách(Distance) hoặc Metric nếu:

1. D(p, q) ≥ 0 (Với D (p, q)=0 nếu và chỉ nếu p=q).

2. D(p, q) = D (q, p).

3. D(p, z) ≤ D (p, q) + D (q, z); z là một điểm ảnh khác.

Khoảng cách Euclide: Khoảng cách Euclide giữa hai điểm ảnh p(x, y)và q(s, t) được định nghĩa như sau:

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{1/2}$$

Khoảng cách khối: Khoảng cách D_4(p, q) được gọi là khoảng cách khối đồ thị (CityBlock Distance) và được xác định như sau:

$$D_4(p, q) = | x - s | + | y - t |$$

Giá trị khoảng cách giữa các điểm ảnh r: giá trị bán kính r giữa điểm ảnh từ tâm điểm ảnh đến tâm điểm ảnh q khác.

Khoảng cách D_8(p, q) còn gọi là khoảng cách bàn cờ (Chess-Board Distance) giữa điểm ảnh p, q được xác định như sau:

$$D_8(p,q) = \max (| x - s |, | y - t |)$$

1.2.6 Biến đổi ảnh (Image Transform)

Trong xử lý ảnh do số điểm ảnh lớn các tính toán nhiều(độ phức tạp tính toán cao) đòi hỏi dung lượng bộ nhớ lớn, thời gian tính toán lâu. Các phương pháp khoa học kinh điển áp dụng cho xử lý ảnh hầu hết khó khả thi. Người ta sử dụng các phép toán tương đương hoặc biến đổi sang miền xử lý khác để dễ tính toán. Sau khi xử lý dễ dàng hơn được thực hiện, dùng biến đổi ngược để đưa về miền xác định ban đầu, các biến đổi thường gặp trong xử lý ảnh gồm:

- Biến đổi Fourier, Cosin, Sin…

- Biến đổi(mô tả) ảnh bằng tích chập, tích Kronecker (theo xử lý số tín hiệu). Các biến đổi khác như KL(Karhumen Loeve), Hadamard.

- Một số công cụ xác suất thông kê cũng được sử dụng trong xử lý ảnh.

1.2.7 Nén ảnh

Ảnh dù ở dạng nào vẫn chiếm không gian bộ nhớ rất lớn. Khi mô tả ảnh người ta đã đưa kỹ thuật nén ảnh vào. Các giai đoạn nén ảnh có thể chia ra thế hệ 1, thế hệ 2. Hiện nay, các chuẩn MPEG được dùng với ảnh đang phát huy hiệu quả.

2. Lấy mẫu và lượng tử hóa

Một ảnh g(x, y) ghi được từ Camera là ảnh liên tục tạo nên mặt phẳng hai chiều. Ảnh cần chuyển sang dạng thích hợp để xử lí bằng máy tính. Phương pháp biến đổi một ảnh (hay một hàm) liên tục trong không gian cũng như theo giá trị thành dạng số rời rạc được gọi là số hoá ảnh. Việc biến đổi này có thể gồm hai bước:

Bước 1: Đo giá trị trên các khoảng không gian gọi là lấy mẫu.

Bước 2: Ánh xạ cường độ(hoặc giá trị) đo được thành một số hữu hạn các mức rời rạc gọi là lượng tử hoá.

2.1 Lấy mẫu

Lấy mẫu là một quá trình, qua đó ảnh được tạo nên trên một vùng có tính liên tục được chuyển thành các giá trị rời rạc theo tọa độ nguyên. Quá trình này gồm 2 lựa chọn: Một là khoảng lấy mẫu, hai là cách thể hiện dạng mẫu. Lựa chọn thứ nhất được đảm bảo nhờ lý

thuyết lấy mẫu của Shannon. Lựa chọn thứ hai liên quan đến độ đo (Metric) được dùng trong miền rời rạc.

- Khoảng lấy mẫu (Sampling Interval):

Ảnh lấy mẫu có thể được mô tả như việc lựa chọn một tập các vị trí lấy mẫu trong không gian hai chiều liên tục. Đầu tiên mô tả qua quá trình lấy mẫu một chiều với việc sử dụng hàm delta:

$$\delta(x - x_0) = \begin{cases} 0, & khi\ x \neq 0 \\ \infty, & khi\ x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(x - x_0)dx = \int_{x_0-}^{x_0+} \delta(x - x_0)dx = 1$$

Hàm răng lược là chuỗi các xung răng lược từ (-∞, +∞) với các khoảng Δx:

$$Comb(x) = \sum_{r=-\infty}^{\infty} \delta(x - r\Delta x)$$

với r là số nguyên, Δx là khoảng lấy mẫu.

Khoảng lấy mẫu(Sampling Interval) Δx là một tham số cần phải được chọn đủ nhỏ, thích hợp, nếu không tín hiệu thật không thể khôi phục lại được từ tín hiệu lấy mẫu.

- Định lý lấy mẫu Shannon:

Giả sử g(x) là một hàm giới hạn giải (Band Limited Function) và biến đổi Fourier của nó là G(ωx) = 0 đối với các giá trị $W_x > \omega_x$. Khi đó g(x) có thể được khôi phục lại từ các mẫu được tạo tại các khoảng Δx đều đặn. Tức là

$$\Delta x \leq \frac{1}{2\omega_x}$$

Định lý lẫy mẫu của Shannon có thể mở rộng cho không gian hai chiều. Hàm răng lược hai chiều khi đó được xác định:

$$Comb(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x - i\Delta x)(y - j\,\Delta y)$$

Khi đó: $\Delta x \leq \frac{1}{2\omega_x}; \Delta y \leq \frac{1}{2\omega_y}$

- Các dạng lấy mẫu (Tesselation):

Dạng lẫy mẫu điểm ảnh là cách sắp xếp các điểm ảnh trong không gian hai chiều. Một số dạng lấy mẫu điểm ảnh là dạng chữ nhật, tam giác, lục giác…

2.2 Lượng tử hóa

Lượng tử hoá là ánh xạ từ các số thực mô tả giá trị lấy mẫu thành một giải hữu hạn các số thực. Nói cách khác, đó là quá trình số hoá biên độ.

Các giá trị lấy mẫu Z là một tập các số thực từ giá trị Z_{min} đến lớn nhất Z_{max}. Mỗi một số trong các giá trị mẫu Z cần phải biến đổi thành một tập hữu hạn số bit để máy tính lưu trữ hoặc xử lý.

Giả sử Z là một giá trị lấy mẫu (số thực) tại vị trí nào đó của mặt phẳng ảnh, và $Z_{min}<=Z'<=Z_{max}$ và giả sử chúng ta muốn lượng hoá giá trị đó thành một trong các mức rời rạc: l_1, l_2, … l_n tương ứng với Z_{min} đến Z_{max}. Khi đó, quá trình lượng hoá có thể thực hiện bằng cách chia toàn bộ miền vào (Z_{max} - Z_{min}) thành l khoảng, mỗi khoảng là Δl và khoảng thứ I được đặt tại điểm giữa các khoảng liền kề l_i. họ các giá trị z được thực hiện và mô tả bằng l_i theo quá trình trên đây, khi đó sai số của quá trình lấy mẫu có thể được xác định theo: $e_q = l_i - Z$.

CHƯƠNG 2: NHẬN DIỆN KHUÔN MẶT

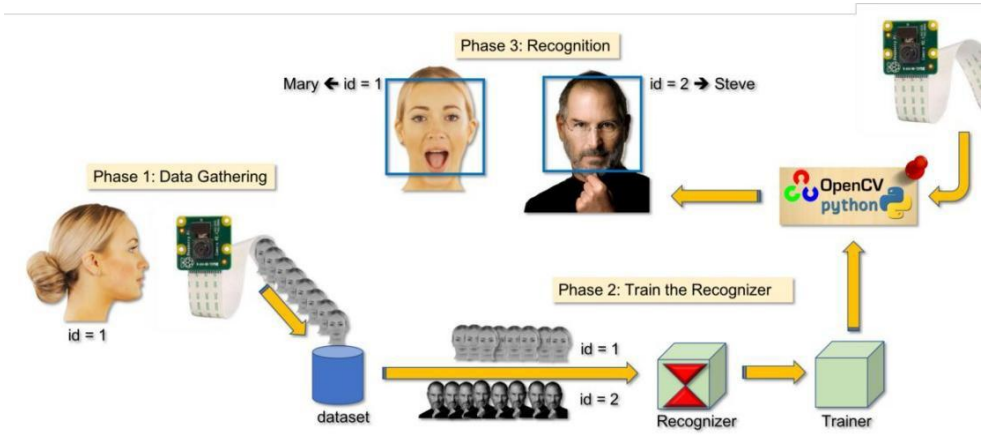
1. Mô hình nhận diện khuôn mặt

1.1 Nhận diện khuôn mặt

Nhận dạng khuôn mặt là một kỹ thuật để xác định hoặc xác minh khuôn mặt từ hình ảnh kỹ thuật số hoặc khung video. Một con người có thể nhanh chóng xác định các khuôn mặt mà không cần nỗ lực nhiều. Đó là một nhiệm vụ dễ dàng đối với chúng tôi, nhưng nó là một nhiệm vụ khó khăn đối với máy tính. Có nhiều độ phức tạp khác nhau, chẳng hạn như độ phân giải thấp, độ che phủ, sự thay đổi về độ chiếu sáng, v.v. Những yếu tố này ảnh hưởng lớn đến độ chính xác của máy tính để nhận dạng khuôn mặt hiệu quả hơn. Trước tiên, cần phải hiểu sự khác biệt giữa nhận diện khuôn mặt và nhận dạng khuôn mặt.

Face Detection: Nhận diện khuôn mặt thường được coi là việc tìm kiếm các khuôn mặt (vị trí và kích thước) trong một hình ảnh và có thể trích xuất chúng để sử dụng bởi thuật toán nhận diện khuôn mặt.

Face Recognition: Thuật toán nhận dạng khuôn mặt được sử dụng để tìm các tính năng được mô tả duy nhất trong hình ảnh. Hình ảnh khuôn mặt đã được trích xuất, cắt xén, thay đổi kích thước và thường được chuyển đổi trong thang độ xám.



Các bước tổng quan

Bước 1: Thu thập dữ liệu khuôn mặt (Face Data Gathering).

Bước 2: Huấn luyện mô hình với dữ liệu vừa thu thập (Train the Recognizer). Bước 3: Nhận diện, phân biệt các khuôn mặt với nhau (Recognition).

=> Ở bước 1 (thu thập dữ liệu khuôn mặt) và bước 3 (nhận diện khuôn mặt), điều tiên quyết là phải xác định được vị trí khuôn mặt.

1.2 Xác định vị trí khuôn mặt (Face Detecting)

Nhiệm vụ quan trọng và là nền tảng cho việc nhận diện khuôn mặt (Face Recognition) là Phát hiện khuôn mặt (Face Detecting). Trước hết, phải "bắt" được khuôn mặt (trong Bước 1) để

nhận diện và phân biệt nó với các khuôn mặt khác (trong Bước 3). Và cách thông dụng nhất hiện nay để phát hiện được khuôn mặt là sử dụng bộ phân loại Haar Cascade.

1.3 Giới thiệu về Haar Cascade

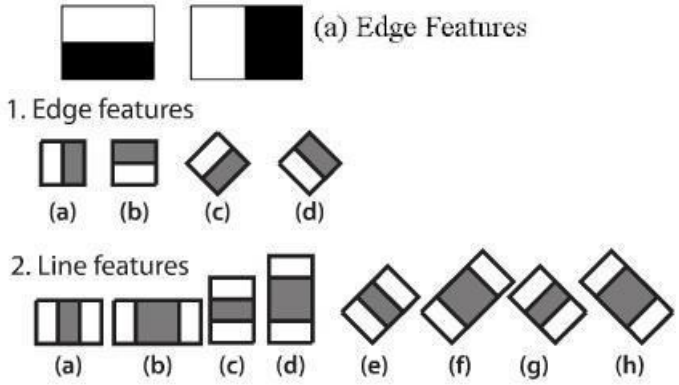
Haar Cascade là một thuật toán được tạo ra dựa trên những tính năng đó để phát hiện đối tượng (có thể là khuôn mặt, mắt, tay, đồ vật,…) được đề xuất vào năm 2001 bởi Paul Viola và Michael Jones trong bài báo của họ với khẳng định "Phát hiện đối tượng một cách nhanh chóng bằng cách sử dụng tầng (Cascade) tăng cường các tính năng đơn giản".

Triển khai ban đầu được sử dụng để phát hiện khuôn mặt chính diện và các đặc điểm như Mắt, Mũi và Miệng. Tuy nhiên, có nhiều đặc trưng Haar được đào tạo trước đó trong GitHub cũng có thể dùng cho các đối tượng khác cũng như cho toàn bộ cơ thể, thân trên, thân dưới, nụ cười và nhiều đồ vật khác.

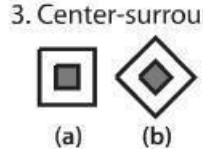
Nói một cách dễ hiểu hơn, Haar Cascade là gì? Là một lớp model có thể giúp chúng ta nhận diện khuôn mặt (Haar Cascade face detection) Haar Cascade sử dụng các tầng Haar và sau đó sử dụng thật nhiều đặc trưng đó qua nhiều lượt (Cascade) và tạo thành một cỗ máy nhận diện khuôn mặt hoàn chỉnh.

1.4 Đặc trưng Haar (Bộ lọc Haar)

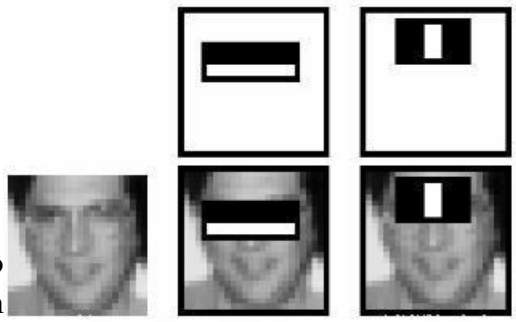
Các ví dụ về đặc trưng Haar được liệt kê ở dưới, trong đó a) là các đặc trưng bắt các cạnh trong ảnh, và b) bắt các đường thẳng trong ảnh. Ngoài ra, còn có các đặc trưng Haar khác, như ví dụ c) về đặc trưng "4 hình vuông" dưới đây



hoặc đặc trưng nằm gọn trong trung tâm một vùng như ví dụ 3 trong ảnh dưới đây:



Tuy nhiên, cách áp dụng các bộ lọc này khác một chút so với các cửa sổ bộ lọc bên CNN. 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 đó được minh hoạ trên ảnh sau:

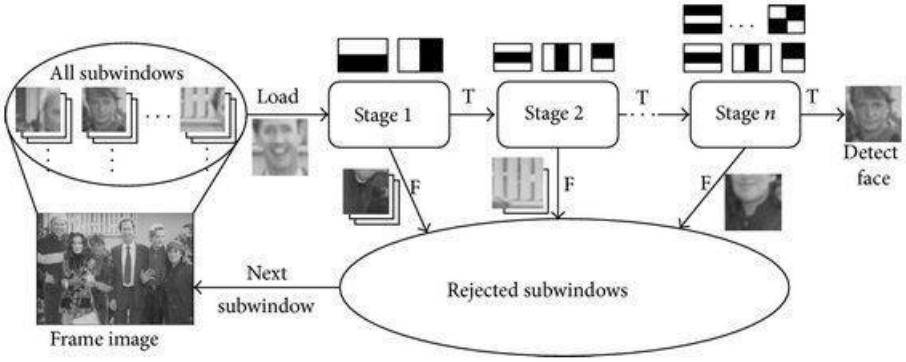
Trong hình trên, cửa sổ  trượt được đặt ngay ngắn vừa gọn để nhìn được toàn bộ ảnh. Có thể nhận ra rằng bộ lọc đầu tiên đang tìm một "cạnh" phân cách giữa mắt/lông mày với mũi, vì ở vị trí này có chênh lệch về màu đáng kể;

Ở bộ lọc sau, mô hình đang tìm đường sống mũi, vì ở đó sẽ có màu sáng hơn so với 2 bên (vì nó cao hơn dễ bắt sáng hơn). Và như đã nói trên, bộ lọc Haar chỉ nhìn cụ thể vào một vùng trong cửa sổ để tìm: trong khuôn mặt thì mũi lúc nào cũng ở chính giữa chứ không ở các góc, nên không cần nhìn các góc khác.

Cấu trúc phân tầng của Haar Cascade

Để hiểu hơn về cách Haar Cascade hoạt động thì ta sẽ đi sâu hơn về từng bước nhận diện khuôn mặt của Haar Cascade:

Trong 6000+ đặc trưng, chia chúng ra thành rất nhiều bước. Trong đó, mỗi lần cửa sổ trượt qua một vùng bước ảnh, từng bước một sẽ được xử lý: nếu bước 1 nhận đó là mặt, chúng ta chuyển qua bước 2; và nếu không thì chúng ta bỏ qua vùng đó và trượt cửa sổ đi chỗ khác. Nếu một vùng pass toàn bộ các bước test mặt đó thì cửa sổ đó có chứa mặt người.



Bước 1: Hình ảnh (đã được gửi đến bộ phân loại) được chia thành các phần nhỏ (hoặc các cửa sổ con như trong hình minh họa).

Bước 2: Chúng tôi đặt N không có bộ dò theo cách xếp tầng trong đó mỗi bộ phát hiện sự kết hợp của các loại đặc trưng khác nhau từ các hình ảnh (ví dụ: đường thẳng, cạnh, hình tròn, hình vuông) được truyền qua. Giả sử khi việc trích xuất đối tượng địa lý được thực hiện, mỗi phần phụ được gán một giá trị tin cậy.

Bước 3: Hình ảnh (hoặc hình ảnh phụ) có độ tin cậy cao nhất được phát hiện dưới dạng khuôn mặt và được gửi đến bộ tích lũy trong khi phần còn lại bị từ chối. Do đó, Cascade tìm nạp khung hình / hình ảnh tiếp theo nếu còn lại và bắt đầu lại quá trình.

1.5. Trích xuất đặc trưng khuôn mặt

Ở bước này, chúng ta sẽ cần một mô hình để trích xuất đặc trưng từ ảnh khuôn mặt (Feature Extractor). Những đặc trưng được trích xuất từ mô hình này còn được gọi là Face Embeddings hay Face Encodings, chúng ta có thể hiểu là cách biểu diễn khác của ảnh khuôn mặt giúp cho việc nhận dạng khuôn mặt dễ dàng hơn (thay vì những điểm ảnh đơn thuần). Face Embeddings có một đặc tính rất thú vị: những khuôn mặt giống nhau sẽ có Face Embeddings "gần" nhau hơn (dựa trên khoảng cách Euclid).

2. Thách thức trong nhận diện khuôn mặt

Hệ thống nhận dạng khuôn mặt ngày nay rất cần thiết và nó đã trải qua một chặng đường dài. Việc sử dụng nó là cần thiết trong một số ứng dụng, chẳng hạn như Hệ thống truy xuất ảnh, giám sát, xác thực/truy cập, điều khiển, v.v. Nhưng có một vài thách thức liên tục xảy ra trong hệ thống nhận dạng hình ảnh hoặc khuôn mặt.

Những thách thức này cần được vượt qua để tạo ra các hệ thống nhận dạng khuôn mặt hiệu quả hơn. Sau đây là những thách thức ảnh hưởng đến khả năng của Hệ thống nhận dạng khuôn mặt.

Sự chiếu sáng

Sự chiếu sáng đóng một vai trò thiết yếu trong quá trình nhận dạng hình ảnh. Nếu có một chút thay đổi trong điều kiện ánh sáng, nó sẽ ảnh hưởng lớn đến kết quả của nó. Đó là ánh sáng thay đổi, và sau đó kết quả có thể khác nhau đối với cùng một đối tượng gây ra ánh sáng thấp hoặc cao.

Background

Nền của đối tượng cũng đóng một vai trò quan trọng trong việc Nhận diện khuôn mặt. Kết quả có thể không giống ngoài trời so với những gì được tạo ra trong nhà bởi vì yếu tố – ảnh hưởng đến hiệu suất của nó thay đổi ngay sau khi địa điểm thay đổi.

Dáng bộ

Hệ thống nhận dạng khuôn mặt rất nhạy cảm với các biến thể tư thế. Sự chuyển động của đầu hoặc các vị trí khác nhau của máy ảnh có thể gây ra thay đổi cấu trúc khuôn mặt và nó sẽ tạo ra kết quả sai.

Biến số

Biến số có nghĩa là những thứ thường có ngoài khuôn mặt như râu, ria mép, các phụ kiện (kính bảo hộ, mũ lưỡi trai, mặt nạ, v.v.) cũng ảnh hưởng đến ước tính của hệ thống nhận dạng khuôn mặt.

Biểu cảm

Một yếu tố quan trọng khác cần được ghi nhớ là biểu cản khác nhau của cùng một cá nhân. Thay đổi biểu cảm trên khuôn mặt có thể tạo ra một kết quả khác cho cùng một cá nhân.

CHƯƠNG 3: GIỚI THIỆU VỀ FACIAL LANDMARKS VÀ DLIBS

1. Giới thiệu về Facial Landmarks

Xác định facial landmark là một bài toán con của bài toán dự đoán hình dạng (shape prediction). Vậy bài toán dự đoán hình dạng là gì? Đó chính là việc chúng ta phải xác định được những điểm chính tạo nên hình dạng của đối tượng trong một bức ảnh. Trong bài toán xác định facial landmark, chúng ta sẽ phải xác định được những điểm chính trong bức ảnh tạo nên hình dạng khuôn mặt người. Facial landmark là đầu vào cho nhiều bài toán khác như dự đoán tư thế đầu, tráo đổi khuôn mặt, phát hiện nháy mắt, xoay chỉnh lại khuôn mặt và điển hình là công nghệ nhận dạng khuôn mặt FaceID được Apple trang bị trên iphone X trở về sau.

Quá trình chung phát hiện facial landmarks gồm 2 bước:

Bước 1 - Detect face: Định vị vị trí khuôn mặt trong ảnh (detect face ROI). Một số methods như Haar cascades, HOG + Linear SVM object detector, pre-trained DL-bases model… Không phụ thuộc vào methods, cuối cùng chúng ta sẽ nhận face bounding box (các tọa độ).

Bước 2 - Detect the key facial structures trên face ROI (tìm được ở bước 1). Có nhiều facial landmark detectors nhưng tất cả các methods đều cố gắng localize và label các facial regions:

- Mouth (miệng)
- Right eyebrow (lông mày phải)
- Left eyebrow (lông mày trái)
- Right eye (mắt phải)
- Left eye (mắt trái)
- Nose (mũi)
- Jaw (quai hàm)

2. Tìm hiểu dlib's facial landmark detector

The pre-trained facial landmark detector trong dlib được sử dụng để ước lượng 68 (x, y) - coordinates tương ứng với tọa độ các facial landmarks trên khuôn mặt.

Chỉ số của 68 tọa độ có thể được biểu diễn như hình dưới (tuy nhiên ở trong Python nó sẽ được đánh số từ 0 đến 67).

Từ đó có thể nhìn thấy tọa độ của các vùng trên khuôn mặt như sau:

- Mouth (miệng): [49;68]
- Right eyebrow (lông mày phải): [18;22]
- Left eyebrow (lông mày trái): [23;27]
- Right eye (mắt phải): [37;42]
- Left eye (mắt trái): [43;48]
- Nose (mũi): [28;36]
- Jaw (quai hàm): [1;17]

Hình 4 Tọa độ 68 điểm được đánh dấu trên khuôn mặt

CHƯƠNG 4: THIẾT KẾ VỀ THỰC HIỆN ĐỀ TÀI

1. Phương pháp thực hiện

Sau khi phát hiện khuôn mặt của chủ thể và xác định 68 điểm chuẩn trên phần trước khuôn mặt, ta lần lượt xác định các chuyển động trên khuôn mặt dựa vào khoảng cách giữa các điểm. Sau đó thực hiện các thao tác *di chuyển, click, và tăng tốc độ di chuyển con trỏ chuột*:

- Đối với vùng mắt, ta chọn khoảng cách giữa hai điển 28 và 29 làm giá trị tham chiếu vì khoảng cách giữa hai điểm này ít thay đổi theo tỉ lệ khuôn mặt khi di chuyển khuôn mặt xa gần và theo các hướng. Vì tập train chỉ có thể trả về kết quả hai mắt đóng hoặc mở cùng lúc, do đó ta chỉ cần xác định khi nào mắt trái (hoặc mắt phải) đóng lại, bằng cách tính tỉ lệ khoảng cách giữa cặp điểm 38 - 40 so với khoảng cách tham chiếu. Khi mắt đóng thì thực hiện *click con trỏ chuột trên màn hình*.

- Đối với vùng miệng, ta chọn khoảng cách giữa hai điểm 62 - 66 để đoán khi nào miệng được mở ra. Khi miệng mở ra sẽ *tăng tốc độ di chuột (default 20, max 80, step 20)*.

- Đối với cử động khuôn mặt, ta so sánh tỉ lệ khoảng cách giữa hai bên mặt cho nghiêng trái (13 - 30 / 30 - 2) và nghiêng phải (2- 30 / 30 - 13). Đối với nhìn lên cũng thực hiện so sánh khoảng cách trên các điểm dọc khuôn mặt (8 - 33 / 33 - 27). Đối với nhìn xuống, ta thực hiện so sánh khoảng cách giữa độ rộng mắt mở và khoảng cách từ mắt đến chân mày (23 - 43 / 43 - 47), vì khi cúi xuống mắt ta có xu hướng mở to hơn. Các hướng quay lên, xuống, trái, phải lần lượt thực hiện các thao tác *di chuột lên, xuống, trái phải trên màn hình*.

- Kết quả thể hiện trực tiếp theo thời gian thực trên màn hình máy tính. Sau đó có thể tinh chỉnh để trả về kết quả hợp lý hơn và tương tác mượt hơn mà với người sử dụng.

2. Tóm tắt thư viện và source code

A. Thư viện sử dụng:

- *OpenCV*: Đọc hình ảnh từ camera, tiền xử lý ảnh, gán chữ lên các frame và thể hiện kết quả.

- *Numpy*: Thực hiện các phép tính toán học.

- *Dlib*: Dùng để lấy mô hình train nhân diện mặt trước của khuôn mặt, kết hợp cùng bộ dữ liệu shape_predictor_68_face_landmarks.

- *Time*: Time.sleep cho phép dừng chương trình trong một khoảng thời gian tránh tình trạng nhiều thao tác lặp lại không kiểm soát được.

- *PyautoGUI*: Thực hiện các điều khiển đối với con trỏ chuột trên màn hình.

B. Source code:

```
import cv2
import numpy
import dlib
from time import sleep
import pyautogui

# setup các thông số-----------------------------
speed = 20 # tốc độ di chuyển của chuột
count = 0 # đếm thời gian thực các vòng lặp

# lấy đường dẫn đến file train-------------------
direct = 'C:\\Users\\Lenovo\\Desktop\\BTL ML\\'
datfile = direct + 'shape_predictor_68_face_landmarks.dat'

# khởi tạo biến đọc hình ảnh từ camera ----------
cap = cv2.VideoCapture(0)
_, frame = cap.read()

# khởi tạo các hàm để phát hiện và dự đoán ------
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(datfile)


# hàm tính khoảng cách hai điểm -----------------
def pytago(a, b):
    return numpy.sqrt(pow(a[0] - b[0], 2) + pow(a[1] - b[1], 2))


while True:
    count += 1
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(frame)
    for face in faces:
        landmarks = predictor(gray_frame, face)


        # hàm trả về giá trị tọa độ các điểm
        def dot(x):
            return (landmarks.part(x).x, landmarks.part(x).y)


        def eyes():
            lew = pytago(dot(38), dot(40)) # mắt trái
            rew = pytago(dot(43), dot(47)) # mắt phải
            base = pytago(dot(28), dot(29))
            cv2.putText(frame, 'L: ' + str(round(lew, 2)), (10, 70), cv2.FONT_HERSHEY_DUPLEX,
0.5, (255, 0, 0))
            cv2.putText(frame, 'R: ' + str(round(rew, 2)), (10, 90), cv2.FONT_HERSHEY_DUPLEX,
0.5, (255, 0, 0))
            cv2.putText(frame, 'B: ' + str(round(base, 2)), (10, 110), cv2.FONT_HERSHEY_DUPLEX,
0.5, (255, 0, 0))
            cv2.putText(frame, 'S: ' + str(round(speed, 2)), (10, 150),
cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 0, 0))


            if lew < (base*7.5/18): # phát hiện mắt trái nhắm
                cv2.putText(frame, 'Eyes closed', (230, 20), cv2.FONT_HERSHEY_DUPLEX, 0.5,
(255, 0, 0))
                pyautogui.click() # click màn hình
            else:
                cv2.putText(frame, 'Eyes opened', (230, 20), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,
255, 0))


        eyes()

        # hiển thị số vòng lặp
        cv2.putText(frame, 'Time: ' + str(round(count, 2)), (10, 210), cv2.FONT_HERSHEY_DUPLEX,
0.5, (255, 0, 0))
```

```
        def mouth(speed): # hàm phát hiện chuyển động miệng
            mid = pytago(dot(62), dot(66))
            cv2.putText(frame, 'M: ' + str(round(mid, 2)), (10, 130), cv2.FONT_HERSHEY_DUPLEX,
0.5, (255, 0, 0))
            if mid > 8:
                cv2.putText(frame, 'Talking', (255, 50), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0, 0,
255))
                if speed >= 80:
                    speed = 20
                    sleep(0.2)
                else:
                    speed += 20
                    sleep(0.2)

            return speed


        speed = mouth(speed) # cập nhật tốc độ trỏ chuột


        def face(): # hàm phát hiện hướng di chuyển khuôn mặt
            if pytago(dot(2), dot(30)) / pytago(dot(30), dot(13)) >= 1.5:
                cv2.putText(frame, 'Right look', (355, 110), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,
0, 255))
                pyautogui.move(speed, 0, duration=pyautogui.MINIMUM_DURATION)

            elif pytago(dot(13), dot(30)) / pytago(dot(30), dot(2)) >= 1.5:
                cv2.putText(frame, 'Left look', (155, 110), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,
0, 255))
                pyautogui.move(-speed, 0, duration=pyautogui.MINIMUM_DURATION)
            else:
                if pytago(dot(8), dot(33)) / pytago(dot(33), dot(27)) >= 1.5:
                    cv2.putText(frame, 'Up look', (255, 90), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,
0, 255))
                    pyautogui.move(0, -speed, duration=pyautogui.MINIMUM_DURATION)

                elif pytago(dot(23), dot(43)) / pytago(dot(43), dot(47)) <= 1.5:
                    cv2.putText(frame, 'Down look', (255, 390), cv2.FONT_HERSHEY_DUPLEX, 0.5,
(0, 0, 255))
                    pyautogui.move(0, speed, duration=pyautogui.MINIMUM_DURATION)
                else:
                    cv2.putText(frame, 'Straight look', (255, 110), cv2.FONT_HERSHEY_DUPLEX,
0.5, (0, 0, 255))


        face()

    # hiển thị hình ảnh
    cv2.imshow('App', frame)
    key = cv2.waitKey(1)
    if key == 27:
        break
```

3. Kết quả thực hiện

A. Video kết quả thực hiện được chúng em gắn tại link drive sau (công khai):

Video camera

https://drive.google.com/file/d/14ZUgSucBAo5Moz7VmotGxDoJFkgAd11u/view?usp=sharing

Video full màn hình

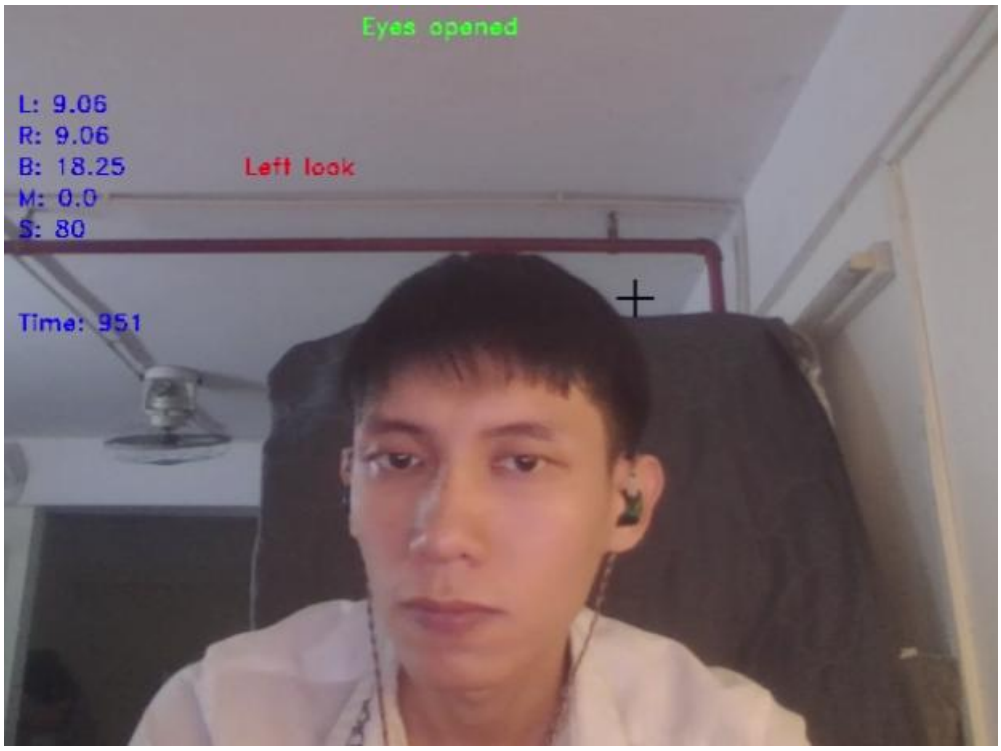
https://drive.google.com/file/d/1Dush-MnzKMJX-ZTjc-swkesYOBOYlLnC/view?usp=sharing

B. Hình ảnh kết quả thực hiện được

- Nhìn thẳng (straight look), mắt mở (eyes opened), tốc độ di chuột 80

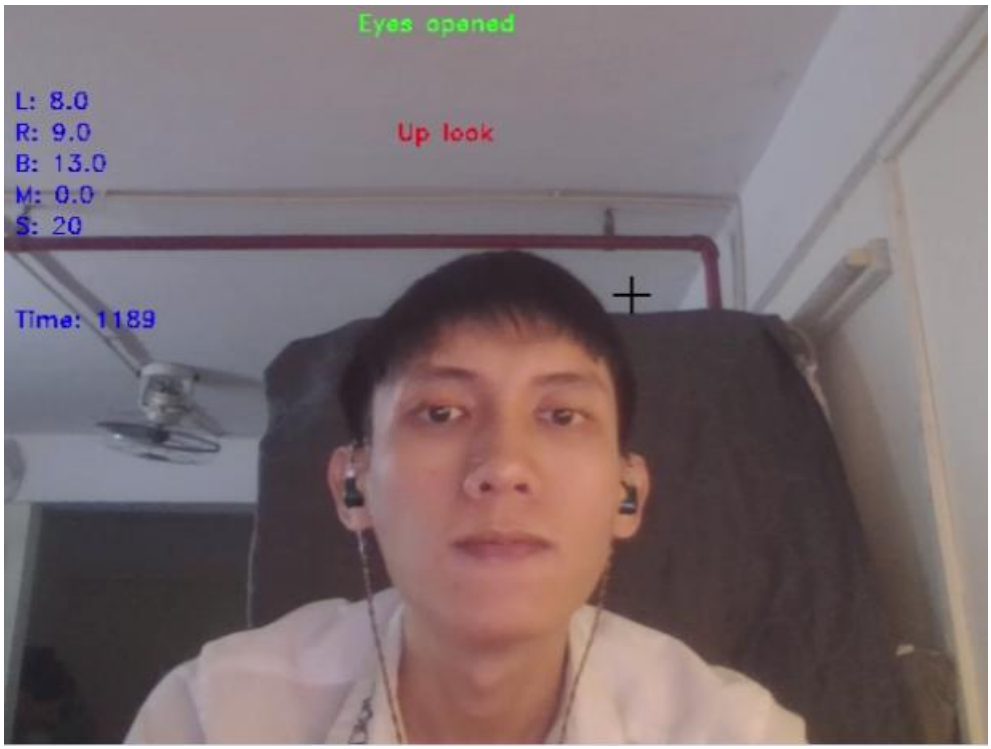
- Nhìn phải (right look), mắt mở (eyes opened), tốc độ di chuột 80

- Nhìn trái (left look), mắt mở (eyes opened), tốc độ di chuột 80

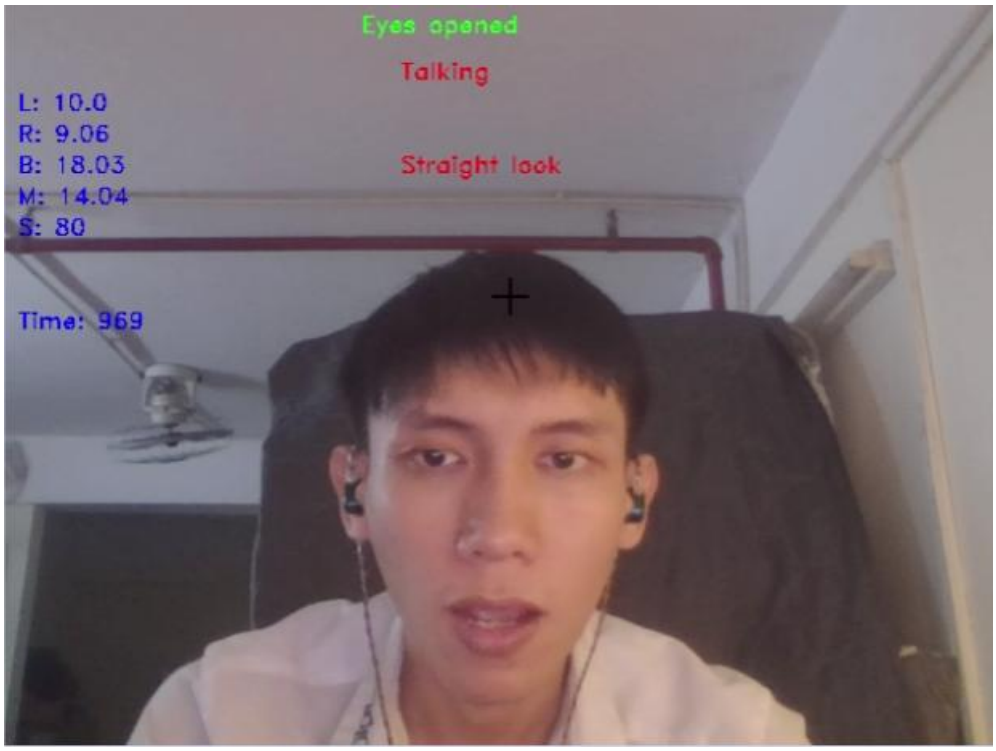


- Nhìn xuống (down look), mắt mở (eyes opened), tốc độ di chuột 20

- Nhìn lên (up look), mắt mở (eyes opened), tốc độ di chuột 20



- Nhìn thẳng (straight look), mắt mở (eyes opened), tốc độ di chuột 80, mở miệng (talking)



- Nhìn thẳng (straight look), mắt đóng(eyes closed), tốc độ di chuột 20

CHƯƠNG 5: KẾT LUÂN

Đề tài không can thiệp sâu vào thuật toán nhận diện khuôn mặt và huấn luyện dữ liệu mà dừng lại ở việc sử dụng mô hình có sẵn cũng như các tính toán đơn giản trên giá trị huấn luyện trả về. Tuy vậy, việc tìm hiểu về mô hình face-landmarks cũng giúp chúng em có cái nhìn sâu hơn về nhận diện khuôn mặt, cùng với việc làm việc trên dữ liệu đầu ra cũng giúp chúng em hình dung được một ứng dụng AI trong thực tế có thể được thực hiện như thế nào.

Các kết quả thực hiện được tuy đáp ứng được yêu cầu đặt ra nhưng vẫn còn nhiều điểm cần phải được tối ưu, cũng như phát triển các thuật toán hợp lý hơn, logic hơn. Đề tài có thể phát triển tiếp tục để thêm nhiều tính năng hơn như kéo thả, cuộn màn hình, hoặc đơn giản là tăng hiệu suất (tốc độ di chuột, độ chính xác cử động, nghiêng mặt nhiều thì chuột đi nhanh hơn, …).