TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐHQG TP. HCM

KHOA KHOA HỌC MÁY TÍNH

Báo cáo môn Nhập môn thị giác máy tính

NHẬN DIỆN CỬ CHỈ TAY

Lớp: **CS231.L11**

Sinh viên thực hiện: **Trương Văn Nhất – 16521759**

GVHD: **TS Mai Tiến Dũng**

TP. HCM, ngày ... tháng ... năm 2020

MỤC LỤC

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành nhất đến TS Mai Tiến Dũng đã quan tâm giúp đỡ và hướng dẫn rất tận tình cho em trong suốt quá trình học tập và thực hiện đồ án.

Có lẽ kiến thức là vô hạn và sự tiếp nhận kiến thức của em có những hạn chế nhất định. Do đó, trong quá trình làm bài tập và đồ án, chắc chắn không tránh khỏi những thiếu sót. Vì vậy, bản thân em rất mong nhận được những đóng góp ý kiến đến từ thầy để đồ án của em được hoàn thiện hơn.

Cuối cùng, em xin kính chúc thầy sức khỏe, hạnh phúc và thành công trên con đường sự nghiệp giảng dạy của mình.

Em xin chân thành cảm ơn.

I. BÀI TOÁN VÀ LÝ DO CHỌN ĐỀ TÀI

1. Bài toán

- Đề tài thực hiện thiết kế ứng dụng nhận diện cử chỉ bàn tay thông qua Python và OpenCV.
- Đề tài hướng đến mục tiêu nhận diện chính xác cử chỉ bàn tay trong nhiều môi trường khác nhau.
- Trên tất cả, vấn đề lớn nhất mà nhóm đang tập trung giải quyết chính là độ ổn định và độ chính xác của ứng dụng phải được tối ưu nhất có thể và phải đặt lên hàng đầu.

2. Lý do chọn đề tài

- Trong các lĩnh vực đời sống hiện nay, các thành tựu khoa học công nghệ đang được áp dụng mạnh mẽ và không ngừng phát triển. Nhận dạng là một trong những bài toán giúp máy tính hiểu được con người thông qua cử chỉ, hành động, lời nói…Phát hiện cử chỉ bàn tay là một đề tài khá thông dụng, có thể ứng dụng vào điều khiển robot, smarthome,…Có nhiều thuật toán khác nhau để phát hiện cử chỉ bàn tay nhưng trong đề tài lần này nhóm chỉ sử dụng màu sắc để nhận diện cử chỉ bàn tay.
- Nhận diện cử chỉ của con người là một vấn đề quan trọng được đạt ra cho thị giác máy tính. Việc thực hiện lệnh theo cử chỉ của con người (cụ thể ở đây là cử chỉ tay) đặt ra những thử thách lớn trong quá trình phát triển ứng dụng. Đó là làm sao để nhận diện đúng bàn tay con người trong những bối cảnh phức tạp (nhiều điểm nhiễu làm ứng dụng nhận diện nhầm), … Kiến trúc cần thiết để phát hiện đối tượng khác nhau theo một cách quan trọng. Cụ thể, kích thước của vectơ đầu ra không cố định. Ví dụ, nếu có một đối tượng trong hình, thì sẽ có bốn tọa độ xác định hộp giới hạn. Giá trị tĩnh và được xác định trước này hoạt động bằng cách sử dụng các kiến trúc đã đề cập trước đó. Tuy nhiên, khi số lượng đối tượng tăng lên, số lượng tọa độ cũng tăng theo. Đặc biệt là số lượng đối tượng không được biết trước, điều này đòi hỏi phải điều chỉnh trong trang điểm của mạng lưới thần kinh.

II. HƯỚNG TIẾP CẬN

1. Không gian màu HSV

- Không gian màu HSV (còn gọi là HSB) là một cách tự nhiên hơn để mô tả màu sắc, dựa trên 3 số liệu.
  - H: (Hue) Vùng màu
  - S: (Saturation) Độ bão hòa màu
  - B (hay V): (Bright hay Value) Độ sáng
- Giả sử ta có một điểm màu có giá trị trong hệ RGB là (R, G, B). ta chuyển sang không gian HSV như sau: Đặt M = Max(R, G, B), m = Min(R, G, B) và C = M - m. Nếu M = R, H' = (G - B)/C mod 6. Nếu M = G, H' = (B - R)/C + 2. Nếu M = B, H' = (R - G)/C + 4. Và H = H'x60. Trong trường hợp C = 0, H = 00 V = M. S = C/V. Trong trường hợp V hoặc C bằng 0, S = 0. Để chuyển từ HSV sang RGB ta làm như sau: Giả sử ta có không gian màu HSV với H = [0, 360], S = [0, 1], V = [0, 1]. Khi đó, ta tính C = VxS. H' = H/60. X = C (1 - |H' mod2 -1|). Ta biểu diễn hệ (R1, G1, B1) như sau.
  - (1, 1, 1) = {
  - (0, 0, 0) nếu H chưa được xác định
  - (C, X, 0) nếu 0 < H' < 1
  - (X, C, 0) nếu 1 < H' < 2
  - (0, C, X) nếu 2 < H' < 3
  - (0, X, C) nếu 3 < H' < 4
  - (X, 0, C) nếu 4 < H' < 5
  - (C, 0, X) nếu 5 < H' < 6}
- Sử dụng hàm HSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV). Với hàm cv2.cvtColor chúng ta dễ dàng chuyển màu BGR sang HSV trong khung hình màu cho video khi được hiển thị.

2. Toán tử contour

- Toán tử contour là “tập các điểm liên tục tạo thành một đường cong (curve) (boundary), và không có khoảng hở trong đường cong đó, đặc điểm chung trong một contour là các

các điểm có cùng, hoặc gần xấp xỉ một giá trị màu, hoặc cùng mật độ. Contour là một công cụ hữu ích được dùng để phân tích hình dạng đối tượng, phát hiện đối tượng và nhận dạng đối tượng". Để tìm contour chính xác, chúng ta cần phải nhị phân hóa bức ảnh (là ảnh nhị phân, không phải ảnh grayscale).

– Cấu trúc Contour trong OpenCV:

contours, hierarchy = cv2.findContours(binaryImage, typeofContour, methodofContour)

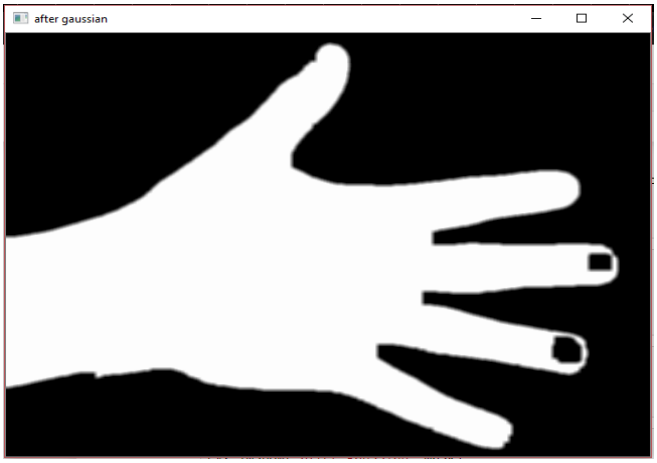
Trong đó:

- **contours**: Danh sách các contour có trong bức ảnh nhị phân. Mỗi một contour được lưu trữ dưới dạng vector các điểm.
- **hierarchy**: Danh sách các vector, chứa mối quan hệ giữa các contour.
- **binaryImage**: Ảnh nhị phân gốc. Một chú ý quan trọng ở đây là sau khi sử dụng hàm findContours thì giá trị của binaryImage cũng thay đổi theo, nên khi sử dụng bạn có thể áp dụng binaryImage.copy() để không làm thay đổi giá trị của binaryImage.
- **typeofContour**: có các dạng sau - RETR_EXTERNAL, RETR_LIST, RETR_CCOMP, RETR_TREE, RETR_FLOODFILL.
- **methodofContour**: Có các phương thức sau - CHAIN_APPROX_NONE, CHAIN_APPROX_SIMPLE, CHAIN_APPROX_TC89_L1, HAIN_APPROX_TC89_KCOS.

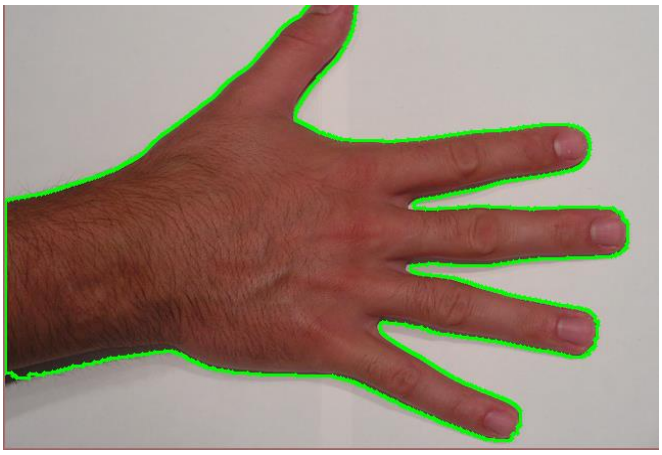
III. THỰC NGHIỆM VÀ ĐÁNH GIÁ

1. Thực nghiệm

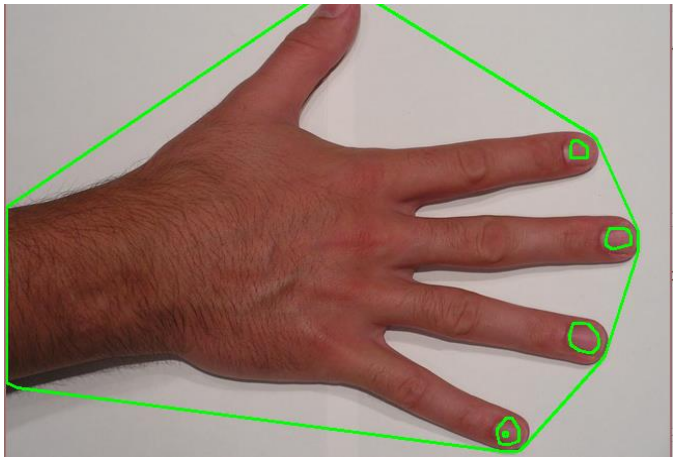
Hình 1: Chuyển ảnh BGR sang ảnh nhị phân



Hình 2: Lọc nhiễu cho đối tượng



Hình 3: Xác định contours



Hình 4: Xác định CONVEX HULL

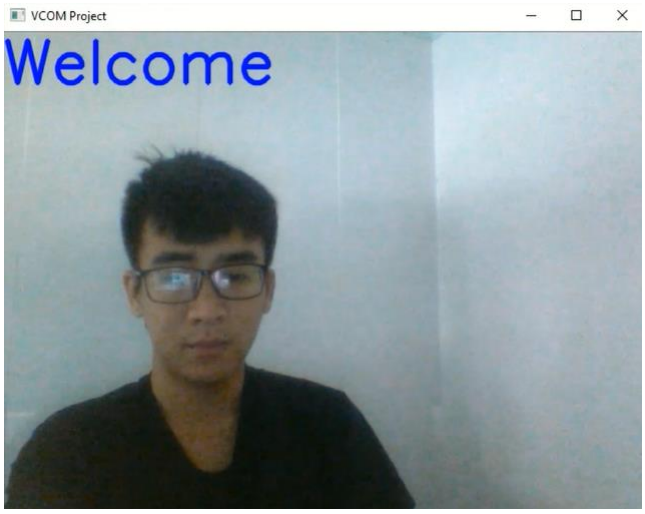
2. Đánh giá

- Trong điều kiện lý tưởng:
  - Nhận diện được các số từ 0-5, OK, Cool (THUMBSUP).
  - Dễ nhầm lẫn giữa số 3 và OK.
- Ưu điểm:
  - Xử lý tốc độ nhanh, khoảng 0.1s.
  - Đơn giản dễ thực hiện.
- Nhược điểm:
  - Lọc nhiễu kém, chỉ nhận được một màu tay đã lấy mẫu trước đó.
  - Dễ bị ảnh hưởng bởi môi trường.
  - Việc tính toán dựa trên diện tích để nhận dạng không tối ưu.
- Hướng phát triển:
  - Áp dụng Machine Learning để nhận diện các cử chỉ như vuốt lên, vuốt xuống, thao tác touch.
  - Cải thiện thuật toán nhận diện được nhiều bàn tay.

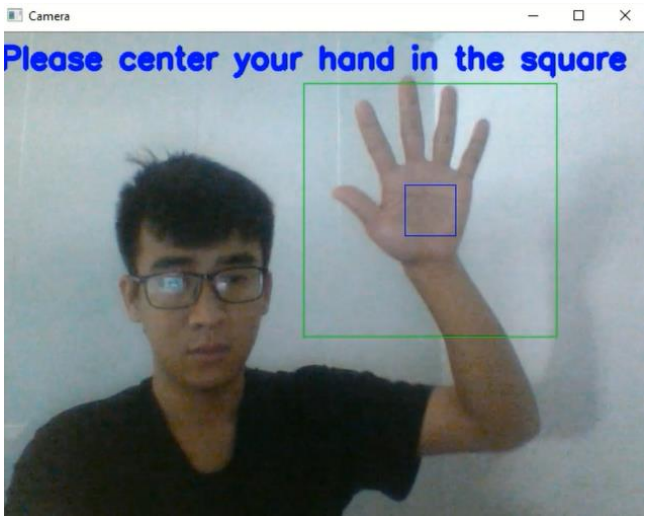
IV. HƯỚNG DẪN SỬ DỤNG

- **Bước 1**:
  - Download source sau đó giải nén source vừa download.
  - Mở phần mềm code của bạn và import source vào (ở đây mình dùng VSCode).
  - Cài đặt các thư viện numpy, opencv, argparse, traceback, ….
- **Bước 2**: Sau khi đã import source thành công, vào file **hand_detection.py** click chuột phải và chọn "**Run Python File in Terminal**", khi chạy xong bạn sẽ được giao diện như Hình 5.
- **Bước 3**: Nhấn phím "v" để vào giao diện lấy mẫu màu ở lòng bàn tay. Khi đó giao diện sẽ như Hình 6.
- **Bước 4**: Đặt tay vào trong khung vuông màu xanh lá sao cho long bàn tay nằm ở vị trí khung vuông màu xanh dương sau đó ấn "space" để lấy mẫu màu bài tay như Hình 6.

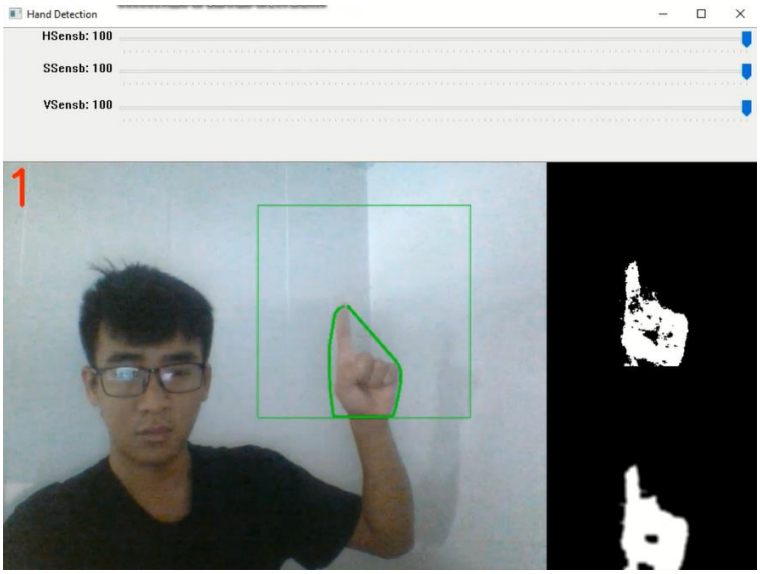
- **Bước 5**: Khi lấu mẫu màu xong ta tiếp tục ấn "space" thêm một lần nữa để vào giao diện chính của chương trình, như Hình 7.
- **Bước 6**: Tại giao diện chính của chương trình, đặt bàn tay vào khung vuông màu xanh lá để tiến hành nhận diện. Kết quả sẽ hiển thị liên tục ở bên trái của giao diện. Tại bước này, bạn có thể điều chỉnh các thanh (HSensb, SSensb, VSensb) như Hình 8 để điều chỉnh độ nhiễu của môi trường xung quanh, giúp tăng kết quả nhận diện.
- **Bước 7**: Ấn phím "q" để kết thúc chương trình.
- **Bước 8**: Xem video "**demo.mp4**" để hiểu rõ hơn.



Hình 5: Giao diện cơ bản



Hình 6: Lấy mẫu màu bàn tay



Hình 7: Giao diện chính của chương trình

Hình 8: Các thanh điều chỉnh độ nhiễu

V. TÀI LIỆU THAM KHẢO

– https://github.com/BlueDi/Hand-Detection