**TRƯỜNG ĐẠI HỌC QUỐC TẾ HỒNG BÀNG**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

**🙢🕮🙠**



**BÁO CÁO ĐỒ ÁN**

**MÔN HỌC  
ĐỒ ÁN CHUYÊN NGÀNH**

Đề tài: Hand Detection

**Giảng viên hướng dẫn: Mai Trung Thành**

**Sinh viên thực hiện: Nguyễn Minh Vũ**

**Mã số sinh viên: 2211110063**

**TP. Hồ Chí Minh, 2024**

LỜI CẢM ƠN

Để hoàn thành đề tài này, em đã nhận được sự hướng dẫn, giúp đỡ và góp ý nhiệt tình của quý thầy cô trường Đại Học Quốc Tế Hồng Bàng và thầy Mai Trung Thành.

Em xin gửi lời biết ơn sâu sắc đến thầy Mai Trung Thành đã dành nhiều thời gian và tâm huyết hướng dẫn nghiên cứu và giúp em hoàn thành môn học.

Em cũng xin chân thành cảm ơn đến quý thầy cô trường Đại học Quốc Tế Hồng Bàng, đặc biệt là những thầy cô đã tận tình dạy bảo cho em suốt thời gian học tập tại trường.

Em đã có nhiều cố gắng hoàn thiện dự án bằng tất cả năng lực của mình, tuy nhiên không thể tránh khỏi nhiều thiếu sót, rất mong nhận được những đóng góp quý báu của quý thầy cô và các bạn.

TP.HCM, ... Tháng ... Năm 2024

Người thực hiện

Nguyễn Minh Vũ

TRANG CAM KẾT

Tôi xin cam kết báo cáo thường kỳ này được hoàn thành dựa trên các kết quả thực hiện bài thực hành của tôi và các mã nguồn và kết quả này chưa được dùng cho bất cứ báo cáo của sinh viên nào khác.

*TP.HCM, ngày ... tháng ... năm 2024*

Người thực hiện

Nguyễn Minh Vũ

NHẬN XÉT CỦA GIẢNG VIÊN

TP.HCM, Ngày ... Tháng ... Năm 2024

Chữ ký giảng viên

Contents

[LỜI CẢM ƠN 1](#_Toc180929321)

[TRANG CAM KẾT 2](#_Toc180929322)

[NHẬN XÉT CỦA GIẢNG VIÊN 3](#_Toc180929323)

[1.1. Giới thiệu 7](#_Toc180929324)

[1.1.1. Bối cảnh và lý do chọn đề tài 7](#_Toc180929325)

[1.1.2. Mục tiêu 7](#_Toc180929326)

[1.1.3. Phạm vi nghiêm cứu 7](#_Toc180929327)

[1.2. Cơ sở lý thuyết 8](#_Toc180929328)

[1.2.1. Theo dõi bàn tay và nhận dạng cử chỉ 8](#_Toc180929329)

[1.2.2. Kết quả thu được tương ứng 8](#_Toc180929330)

[1.2.3. Thư viện MediaPipe 8](#_Toc180929331)

[1.2.4. Xử lý hình ảnh với OpenCV 9](#_Toc180929332)

[1.2.5. Ứng dụng vào phần cứng 10](#_Toc180929333)

[1.3. Phương pháp thực hiện 11](#_Toc180929334)

[1.3.1. Sơ đồ tổng quát của hệ thống 11](#_Toc180929335)

[1.3.2. Lựa chọn về cấu hình module 12](#_Toc180929336)

[1.3.3. Mô hình 13](#_Toc180929337)

[1.3.4. Cài đặt chương trình chính 14](#_Toc180929338)

[1.3.5. Các bước thực hiện chi tiết 19](#_Toc180929339)

[1.4. Kết quả 33](#_Toc180929340)

[1.4.1. Kết quả nhận diện bàn tay 33](#_Toc180929341)

[1.4.2. Nhận diện tay trái và tay phải 33](#_Toc180929342)

[1.4.3. Nhận diện cử chỉ tay 33](#_Toc180929343)

[1.4.4. Độ chính xác 33](#_Toc180929344)

[1.4.5. Phần cứng 33](#_Toc180929345)

[1.5. Đánh giá và kết luận 34](#_Toc180929346)

[1.5.1. Đánh giá 34](#_Toc180929347)

[1.5.2. Kết luận 34](#_Toc180929348)

[1.6. Hướng phát triển 34](#_Toc180929349)

[1.6.1. Mở rộng hệ thống nhận diện cử chỉ phức tạp hơn 34](#_Toc180929350)

[1.6.2. Tích hợp các hệ thống điều khiển 34](#_Toc180929351)

[1.6.3. Cải thiện khả năng hoạt động trong điều kiện thực tế 34](#_Toc180929352)

[1.6.4. Phát triển thêm sự nhận diện cánh tay 34](#_Toc180929353)

[1.6.5. Phát triển phần cứng 34](#_Toc180929354)

[KẾT LUẬN 36](#_Toc180929355)

Tài liệu tham khảo

1. Python Machine Learning <https://www.w3schools.com/python/python_ml_getting_started.asp>
2. AI ROBOT ARM sử dụng Python Arduino OpenCV CVZone - Murtaza's Workshop - Robotics and AI <https://www.youtube.com/watch?v=7KV5489rL3c>
3. Hand Tracking - Murtaza's Workshop - Robotics and AI <https://www.youtube.com/watch?v=p5Z_GGRCI5s>
4. Deep Learning Cơ Bản – Nguyễn Thanh Tuấn

[Sách Deep Learning cơ bản - v2.pdf](Sách%20Deep%20Learning%20cơ%20bản%20-%20v2.pdf)

1. How to Build Your Own Biomimetic Bionic Hand - Will Cogley <https://www.youtube.com/watch?v=8w88HwbYPWI>
2. OpenCV - Python Tutorial

<https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html>

1. Lập trình hệ thống nhúng – Lê Văn Hạnh
2. SoTay\_ARDUINO – tdhshop.com.vn
3. Arduino For Beginners
4. Tutorials for Arduino – Funduino Service Team
5. Programming with Arduino - Hans-Petter Halvorsen

## Giới thiệu

### Bối cảnh và lý do chọn đề tài

Trong thời đại công nghệ phát triển, việc tương tác giữa người và máy tính đang trở thành một lĩnh vực quan trọng. Một trong những phương pháp tương tác tiên tiến là thông qua các cử chỉ tay thay vì sử dụng chuột, bàn phím hoặc màn hình cảm ứng. Cử chỉ tay là cách tự nhiên để con người giao tiếp với nhau và nếu có thể được nhận dạng, máy tính sẽ mở ra nhiều ứng dụng mới như điều khiển từ xa, hỗ trợ người khuyết tật, hoặc thậm chí trong lĩnh vực thực tế ảo (AR/VR).

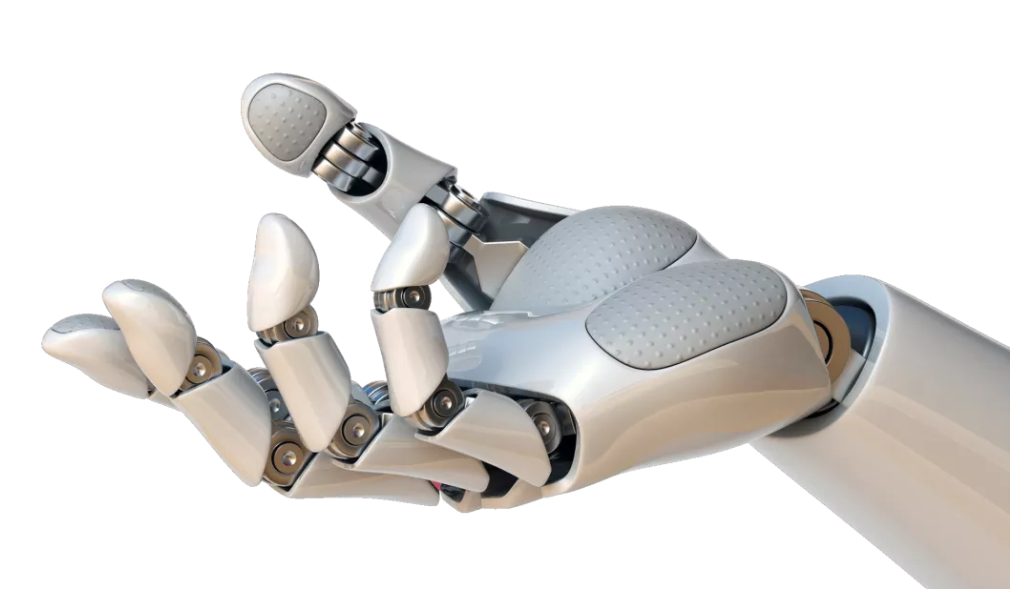
Hiện nay, hầu hết sự tuơng tác giữa con người với công nghệ thực tế ảo đều thông qua các thiết bị hỗ trợ như găng tay cảm biến, … . Vậy nên để có thể tương tác với công nghệ ảo mà không cần thiết bị hỗ trợ thì cần phải có một công nghệ thay thế.

Hand detection là công nghệ được phát triển để theo dõi chuyển động và mô phỏng lại mô hình của bàn tay trong không gian. Việc này được thực hiện bằng các phân tích các hình ảnh thông qua video. Và sau đó bàn tay được mô phỏng lại bằng mô hình 3D.

### Mục tiêu

Mục tiêu của đồ án là xây dựng một hệ thống theo dõi và nhận diện cử chỉ tay trong thời gian thực bằng cách sử dụng camera và các thuật toán xử lý ảnh. Hệ thống này có thể nhận diện vị trí các ngón tay và lòng bàn tay, từ đó xác định các cử chỉ tay khác nhau.

Bên cạnh đó ứng dụng và phát triển phần cứng có tính năng nhận diện và phản hồi thao tác tay.



### Phạm vi nghiêm cứu

* Nghiên cứu cách phát hiện và theo dõi bàn tay trong video từ camera.
* Ứng dụng thư viện **MediaPipe** để theo dõi cử chỉ tay.
* Xây dựng hệ thống nhận diện cử chỉ tay và phản hồi lại bằng cách hiển thị hình ảnh tương ứng hoặc thực hiện các thao tác điều khiển.
* Xây dựng, sáng tạo, lắp ráp phần cứng tích hợp các tính năng trong ứng dụng phần mềm.
* Sử dụng các công nghệ chip, board mạch (Arduino) để tích hợp điều khiển phần cứng.

## Cơ sở lý thuyết

### Theo dõi bàn tay và nhận dạng cử chỉ

Theo dõi bàn tay (Hand Tracking) là quá trình nhận dạng và theo dõi các điểm chính trên bàn tay (landmarks), bao gồm các khớp ngón tay và lòng bàn tay. Khi đã theo dõi được các điểm này, ta có thể xác định trạng thái của bàn tay (mở, đóng, giơ ngón, nắm đấm, v.v.).

### Kết quả thu được tương ứng

Thiết kế chỉnh sửa hình ảnh minh họa cho bàn tay.

Các trường hợp của cả 2 bàn tay về phương diện xoay qua trái hoặc qua phải tùy góc độ ( bé hơn 120 độ).

Xuất kết quả thu được lên màn hình máy sao cho tương ứng với các trường hợp người dùng yêu cầu.

### Thư viện MediaPipe

MediaPipe là một thư viện mã nguồn mở của Google, hỗ trợ việc xây dựng các giải pháp thị giác máy tính và học máy theo thời gian thực. MediaPipe giúp lập trình viên xử lý dữ liệu hình ảnh và video để nhận diện và theo dõi đối tượng, phân tích tư thế, nhận diện khuôn mặt, theo dõi bàn tay, và nhiều ứng dụng khác trong lĩnh vực thị giác máy tính. Một số đặc điểm nổi bật của MediaPipe:

1. Đa nền tảng: MediaPipe hỗ trợ nhiều nền tảng như Android, iOS, và cả trên web thông qua JavaScript.

2. Nhiều mô-đun có sẵn: Cung cấp các mô-đun như Face Detection, Pose Estimation, Hands, Holistic (để nhận diện toàn bộ cơ thể), Objectron (nhận diện 3D), v.v.

3. Xử lý thời gian thực: Được thiết kế tối ưu hóa để xử lý các mô hình thị giác máy tính với hiệu suất cao.

4. Dễ tích hợp: MediaPipe có thể tích hợp với các thư viện học sâu khác như TensorFlow để tăng cường khả năng và tính chính xác của mô hình.

MediaPipe rất phù hợp cho các ứng dụng trong lĩnh vực thực tế tăng cường, thể thao, sức khỏe, và tương tác người-máy trong môi trường yêu cầu nhận diện theo thời gian thực.

* Trong trường hợp này, MediaPipe cung cấp một mô hình sẵn có để phát hiện bàn tay và trả về 21 điểm mốc chính (landmarks) trên bàn tay.

### Xử lý hình ảnh với OpenCV

**OpenCV (Open Source Computer Vision Library)** là một thư viện mã nguồn mở được xây dựng để cung cấp một cơ sở hạ tầng chung cho các ứng dụng thị giác máy tính và để đẩy nhanh việc sử dụng nhận thức máy trong các sản phẩm thương mại. Là một sản phẩm được cấp phép BSD, OpenCV giúp các doanh nghiệp dễ dàng sử dụng và sửa đổi mã, cung cấp nhiều công cụ xử lý ảnh mạnh mẽ. OpenCV được sử dụng trong đồ án để thu thập dữ liệu từ camera, hiển thị kết quả và vẽ lên các hình ảnh.

1. Đọc và hiển thị hình ảnh

- Đọc hình ảnh: Sử dụng cv2.imread() để đọc hình ảnh từ tệp.

- Hiển thị hình ảnh: Sử dụng cv2.imshow() để hiển thị hình ảnh trong cửa sổ.

2. Chuyển đổi hệ màu

- OpenCV cung cấp nhiều tùy chọn chuyển đổi hệ màu, ví dụ chuyển đổi từ RGB sang grayscale, HSV, YUV, v.v.

- Chuyển sang ảnh xám: Sử dụng cv2.cvtColor() với tham số cv2.COLOR\_BGR2GRAY.

3. Xử lý lọc ảnh

- Lọc Gauss: Làm mờ hình ảnh để giảm nhiễu. Sử dụng cv2.GaussianBlur().

4. Phát hiện cạnh

- Canny Edge Detection: Một trong các phương pháp phổ biến để phát hiện cạnh trong ảnh. Dùng cv2.Canny() để phát hiện cạnh.

5. Phát hiện và vẽ hình dạng

- Phát hiện hình dạng: Tìm các biên và hình dạng sử dụng cv2.findContours(), sau đó vẽ lại chúng bằng cv2.drawContours().

6. Biến đổi hình học

- Resize: Thay đổi kích thước ảnh với cv2.resize().

- Rotate: Xoay ảnh dùng cv2.getRotationMatrix2D() và cv2.warpAffine().

7. Áp dụng mô hình AI

- OpenCV hỗ trợ tích hợp với các mô hình học sâu để phát hiện đối tượng, phân loại, nhận diện khuôn mặt, v.v. Bạn có thể sử dụng các mô hình được huấn luyện trước hoặc tùy chỉnh theo nhu cầu của dự án.

* OpenCV cung cấp các công cụ mạnh mẽ giúp lập trình viên thao tác với hình ảnh hiệu quả và thuận tiện, là nền tảng cho các ứng dụng từ nhận diện khuôn mặt, theo dõi đối tượng, đến các dự án thực tế tăng cường.

### Ứng dụng vào phần cứng

Khi ứng dụng OpenCV và các công nghệ xử lý hình ảnh vào phần cứng như bàn tay robot, mục tiêu thường là giúp bàn tay robot nhận diện và tương tác với môi trường xung quanh một cách tự động và chính xác. Một số ứng dụng phổ biến của OpenCV và thị giác máy tính trong điều khiển bàn tay robot bao gồm:

**1. Phát hiện và nhận diện đối tượng**

* **Xác định vị trí và loại đối tượng**: Bàn tay robot cần biết vị trí chính xác của đối tượng để có thể bắt lấy hoặc thao tác với nó. Sử dụng các phương pháp phát hiện đối tượng trong OpenCV, chẳng hạn như **YOLO** hoặc **SSD** (tích hợp với các framework AI như TensorFlow hoặc PyTorch), để nhận diện đối tượng như cốc, hộp, công cụ, v.v.
* **Định vị đối tượng**: Sau khi phát hiện đối tượng, xác định tọa độ của nó trong không gian 2D hoặc 3D để robot biết hướng và khoảng cách di chuyển bàn tay.

**2. Theo dõi và nhận diện cử chỉ tay**

* **Nhận diện cử chỉ tay**: Sử dụng mô-đun Hands của MediaPipe để theo dõi và phân tích cử chỉ tay. Thông tin này có thể dùng để điều khiển bàn tay robot qua các cử chỉ, ví dụ: ngón cái giơ lên có thể kích hoạt lệnh mở bàn tay, trong khi nắm tay lại có thể kích hoạt lệnh nắm chặt.
* **Theo dõi bàn tay trong thời gian thực**: Giúp bàn tay robot điều chỉnh vị trí khi đối tượng di chuyển.

**3. Điều khiển bàn tay robot**

* **Chuyển đổi tọa độ từ hình ảnh sang hệ trục của robot**: Tọa độ đối tượng từ camera thường ở dạng 2D, nhưng bàn tay robot cần tọa độ 3D. Có thể sử dụng kỹ thuật **triangulation** hoặc tích hợp với camera chiều sâu (Depth camera) để lấy thông tin 3D.
* **Điều khiển động cơ**: Sử dụng tọa độ đã chuyển đổi để điều khiển động cơ của bàn tay robot. Thông qua bộ điều khiển (thường dùng PID Controller), bàn tay robot có thể thực hiện các thao tác cầm, nắm, hoặc di chuyển đối tượng.

**4. Phản hồi haptic và điều chỉnh lực**

* **Phản hồi lực**: Tích hợp cảm biến lực ở các ngón tay của bàn tay robot giúp kiểm soát lực khi cầm nắm, tránh làm vỡ hoặc làm hư hại đối tượng.
* **Điều chỉnh lực theo đối tượng**: Bàn tay robot có thể tự động điều chỉnh lực dựa vào loại đối tượng và thông tin từ các cảm biến, ví dụ nắm chặt hơn với vật cứng và nhẹ nhàng hơn với vật dễ vỡ.

**5. Ứng dụng AI trong học cử động**

* **Học máy để tinh chỉnh động tác**: Dữ liệu về hình ảnh và cử động có thể được sử dụng để huấn luyện các mô hình học máy giúp bàn tay robot tinh chỉnh động tác, cải thiện độ chính xác và hiệu suất trong các tác vụ như lấy vật thể nhỏ hoặc đặt vật vào vị trí chính xác.
* **Thuật toán học tăng cường (Reinforcement Learning)**: Một số robot sử dụng học tăng cường để tự học cách điều chỉnh vị trí và lực để cầm, nắm, hoặc đặt đối tượng.

## Phương pháp thực hiện

### Sơ đồ tổng quát của hệ thống

1. Khối Cảm biến và Thu thập Dữ liệu

- Camera: Thu nhận hình ảnh hoặc video từ môi trường xung quanh. Thường sử dụng camera RGB hoặc camera chiều sâu (depth camera) để cung cấp thông tin không gian 3D.

2. Khối Xử lý Hình ảnh và Phân tích

- Xử lý hình ảnh ban đầu (Pre-processing): Lọc và chuyển đổi hình ảnh từ camera, ví dụ chuyển sang ảnh xám (grayscale), làm mờ (blurring) hoặc phát hiện cạnh (edge detection).

- Phát hiện đối tượng và phân tích cử chỉ: Sử dụng các thuật toán như YOLO, MediaPipe hoặc mô hình học sâu để nhận diện đối tượng và phân tích cử chỉ tay, từ đó xác định các vị trí và cử động mong muốn.

- Triangulation hoặc sử dụng Depth Camera: Chuyển tọa độ từ hình ảnh sang hệ trục tọa độ của robot, cho phép xác định vị trí 3D của đối tượng hoặc tay người.

3. Khối Điều khiển Robot

- Tính toán tọa độ và điều khiển động cơ: Chuyển đổi vị trí và góc độ từ hệ tọa độ camera sang hệ tọa độ robot để điều khiển động cơ của bàn tay.

4. Khối Xử lý Học Máy và Tối Ưu

- Cập nhật mô hình theo thời gian thực: Liên tục cải thiện mô hình dựa trên phản hồi từ cảm biến, từ đó giúp bàn tay robot học cách thao tác linh hoạt hơn.

5. Khối Phản hồi và Hiển thị

- Hiển thị thông tin lên màn hình: Hiển thị hình ảnh và các thông số điều khiển cho người giám sát để dễ dàng theo dõi và điều chỉnh khi cần.

### Lựa chọn về cấu hình module

|  |  |  |  |
| --- | --- | --- | --- |
| Tên lựa chọn | Nội dung mô tả | Khoảng giá trị | Giá trị mặc định |
| mode | Tham số này xác định liệu bộ phát hiện sẽ xử lý ảnh tĩnh (True) hay video (False). Khi True, nó xử lý từng khung hình độc lập, có thể chậm hơn nhưng chính xác hơn cho ảnh tĩnh. | {True, False} | False |
| maxHands | Định nghĩa số lượng tối đa bàn tay có thể phát hiện trong một khung hình. Mặc định là 2, cho phép phát hiện cả hai bàn tay. | > 0 | 2 |
| detectionCon | Độ tự tin tối thiểu để phát hiện bàn tay. Giá trị càng cao thì mô hình càng đòi hỏi độ chính xác cao hơn để xác định bàn tay. | > 0 && < 1 | 0.5 |
| trackCon | Độ tự tin tối thiểu để tiếp tục theo dõi bàn tay sau khi phát hiện. Giá trị cao hơn sẽ khiến mô hình cẩn trọng hơn trong việc tiếp tục theo dõi nếu không chắc chắn. | > 0 && < 1 | 0.5 |
| self.mpHands | Khởi tạo module phát hiện và theo dõi bàn tay của MediaPipe. | | |
| self.hands | Cài đặt mô hình phát hiện bàn tay. Nó sử dụng các tham số được truyền vào | | |
| self.mpDraw | Khởi tạo công cụ vẽ của MediaPipe, giúp vẽ các điểm mốc trên tay và các đường nối giữa các điểm mốc trên hình ảnh. | | |
| img | Ảnh đầu vào (thường là khung hình từ webcam). | | |
| draw | Cờ chỉ định có vẽ điểm mốc lên ảnh hay không. | | |
| handNo | Số thứ tự của bàn tay bạn muốn lấy điểm mốc (nếu phát hiện nhiều bàn tay). Mặc định là 0 (bàn tay đầu tiên). | | |
| lm.x, lm.y | ho tọa độ chuẩn hóa của điểm mốc (từ 0 đến 1), và các giá trị này được nhân với kích thước ảnh (w, h) để chuyển đổi thành tọa độ pixel (cx, cy). | | |
| id | Tọa độ pixel và ID điểm mốc (id) được thêm vào danh sách lmList. | | |
| lmList | được trả về, chứa ID và tọa độ pixel của từng điểm mốc. Danh sách này giúp theo dõi vị trí của bàn tay hoặc sử dụng tọa độ cho các bước xử lý tiếp theo. | | |
| pTime, cTime | dùng để tính FPS (số khung hình trên giây). | | |

### Mô hình

Gói mô hình điểm mốc bàn tay phát hiện bản địa hoá điểm chính của 21 toạ độ khớp tay trong các vùng bàn tay đã phát hiện. Mô hình được huấn luyện về khoảng 30 nghìn hình ảnh thực tế, cũng như một số mô hình bàn tay tổng hợp kết xuất được đặt trên nhiều loại nền.

A screenshot of a computer

Description automatically generated

Gói mô hình điểm mốc kim đồng hồ chứa mô hình phát hiện lòng bàn tay và mô hình phát hiện điểm mốc trên tay. Mô hình phát hiện lòng bàn tay xác định tay trong hình ảnh đầu vào, còn mô hình phát hiện điểm mốc trên bàn tay xác định các dấu vết bàn tay cụ thể trên hình ảnh bàn tay bị cắt do mô hình phát hiện lòng bàn tay xác định.

Vì việc chạy mô hình phát hiện lòng bàn tay tốn nhiều thời gian, nên khi ở chế độ chạy video hoặc phát trực tiếp, Tay Cột mốc sẽ sử dụng hộp giới hạn do mô hình điểm đánh dấu kim đồng xác định trong một khung hình để bản địa hoá vùng tay cho các khung hình tiếp theo. Tay mốc chỉ kích hoạt lại mô hình phát hiện lòng bàn tay nếu mô hình đánh dấu tay không còn xác định sự hiện diện của bàn tay hoặc không theo dõi được tay trong khung hình. Điều này giúp làm giảm số lần Hand Returner kích hoạt mô hình phát hiện lòng bàn tay.

### Cài đặt chương trình chính

* Sử dụng thư viện cv2, time, os, sys, cvzone và module HandTrackingModulee.py

import cv2

import time

import os

import sys

import cvzone.SerialModule

import HandTrackingModulee as htm

import cvzone

* Cài đặt khung hình và bắt lỗi

wCam, hCam = 1280, 720

cap = cv2.VideoCapture(0)

if not cap.isOpened():

    print("Error: Camera not accessible.")

    exit()

cap.set(3, wCam)

cap.set(4, hCam)

* Khởi tạo đường dẫn tới file ảnh và thời gian

folderpath = ""

pTime = 0

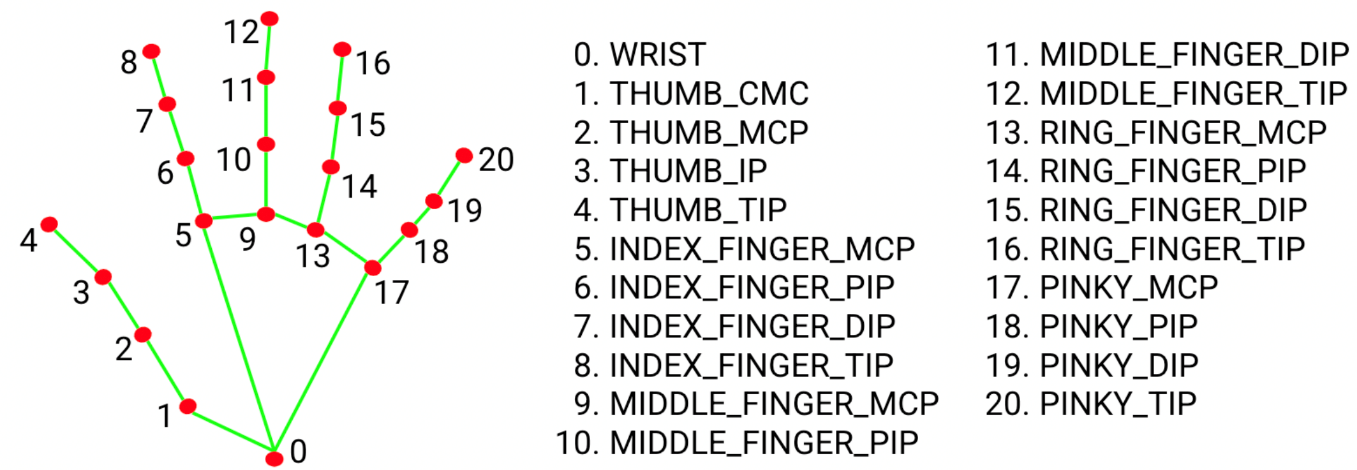
* Khởi tạo kết nối với phần cứng thông quan công COM3

detector = htm.handDetector(detectionCon=0.75)

mySerial = cvzone.SerialModule.SerialObject("COM3", 9600, 1)

* Tạo mảng gồm các điểm đầu của ngón tay tương ướng với Hand landmark

tipIds = [4, 8, 12, 16, 20]



* Tạo dictionary disthand gồm key là mảng gồm 5 phần từ chỉ 1 và 0, value là tên của từng ảnh tương ứng.

dicthand = {

    (1, 0, 0, 0, 0): "/1.png",(0, 1, 0, 0, 0): "/2.png",(0, 0, 1, 0, 0): "/3.png",(0, 0, 0, 1, 0): "/4.png",(0, 0, 0, 0, 1): "/5.png",

    (1, 1, 0, 0, 0): "/6.png",(1, 0, 1, 0, 0): "/7.png",(1, 0, 0, 1, 0): "/8.png",(1, 0, 0, 0, 1): "/9.png",(0, 1, 1, 0, 0): "/10.png",

    (0, 1, 0, 1, 0): "/11.png",(0, 1, 0, 0, 1): "/12.png",(0, 0, 1, 1, 0): "/13.png",(0, 0, 1, 0, 1): "/14.png",(0, 0, 0, 1, 1): "/15.png",

    (1, 1, 1, 0, 0): "/16.png",(1, 1, 0, 1, 0): "/17.png",(1, 1, 0, 0, 1): "/18.png",(1, 0, 1, 1, 0): "/19.png",(1, 0, 1, 0, 1): "/20.png",

    (0, 1, 1, 1, 0): "/21.png",(0, 1, 1, 0, 1): "/22.png",(0, 1, 0, 1, 1): "/23.png",(0, 0, 1, 1, 1): "/24.png",(1, 0, 0, 1, 1): "/25.png",

    (1, 1, 1, 1, 0): "/26.png",(1, 1, 1, 0, 1): "/27.png",(1, 1, 0, 1, 1): "/28.png",(1, 0, 1, 1, 1): "/29.png",(0, 1, 1, 1, 1): "/30.png",

    (1, 1, 1, 1, 1): "/31.png",(0, 0, 0, 0, 0): "/32.png"

}

* Khởi tạo vòng lặp while với giá trị truyền vào là true và gán các biến bằng các hàm đã cài đặt trong module HandTrackingModulee.py

while True:

    success, img = cap.read()

    if not success:

        print("Error: Frame not captured.")

        break

    img = detector.findHands(img)

    lmList = detector.findPosition(img, draw=False)

* Khởi tạo mảng finger[]. Mảng này sẽ chứa 5 phần tử chỉ 1 và 0. 1 tương ứng với ngón mở và 0 tương ứng với ngón đóng.
* Cài đặt câu lệnh if tương ứng với 2 trường hợp là tay trái và tay phải. Tương ứng với từng trường hợp sẽ có các trường hợp xoay bàn tay qua trái hoặc qua phải tùy theo góc độ (<120 độ).

1. Với trường hợp là tay phải:

if (lmList[tipIds[0] - 3][1] > lmList[tipIds[0] - 4][1] and lmList[tipIds[4] - 3][2] < lmList[tipIds[0] - 4][2]) or ((lmList[tipIds[0] - 2][2] < lmList[tipIds[1] - 3][2]) and (lmList[tipIds[0] - 4][1] > lmList[tipIds[1] - 3][1])):

Có 5 trường hợp tương ứng như 0-90 độ trái, 0-90 phải, lớn hơn 90 độ phải, lớn hơn 90 độ trái, và trường hợp mặc định là 0 độ.

+ Lớn hơn 90 độ trái:

if lmList[tipIds[4] - 3][2] >= lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0]][2] > lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right over -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] > lmList[tipIds[id] - 2][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

+ Lớn hơn 90 độ phải:

elif lmList[tipIds[1] - 3][2] >= lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0] - 1][2] > lmList[tipIds[0]][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right over 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] < lmList[tipIds[id] - 2][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

+ Từ 0 đến 90 độ phải:

elif lmList[tipIds[4] - 3][1] > lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:

                    if lmList[tipIds[1] - 3][1] > lmList[tipIds[0]][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

                else:

                    finger.append(0)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

+ Từ 0 đến 90 độ trái:

elif lmList[tipIds[1] - 3][1] < lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[0]][2] > lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

+ Trường hợp mặc định 0 độ:

else:

                if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:

                    finger.append(1)

                else:

                    finger.append(0)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right 0"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

1. Với trường hợp là tay trái: (else)

+ Trường hợp lớn hơn 90 độ phải:

if lmList[tipIds[4] - 3][2] > lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0]][2] > lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left over 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] > lmList[tipIds[id] - 2][1]:

                        finger.append(1)

                    else:

                        finger.append(0)

+ Trường hợp lớn hơn 90 độ trái:

elif lmList[tipIds[1] - 3][2] > lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0]][2] < lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left over -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] > lmList[tipIds[id] - 2][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

+ Trường hợp từ 0 đến 90 độ phải:

elif lmList[tipIds[1] - 3][1] > lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[1] - 3][2] < lmList[tipIds[0]][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

+ Trường hợp từ 0 đến 90 độ trái:

elif lmList[tipIds[4] - 3][1] < lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[0]][2] < lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

+ Trường hợp mặc định 0 độ:

else:

                if lmList[tipIds[0]][1] < lmList[tipIds[0] - 1][1]:

                    finger.append(1)

                else:

                    finger.append(0)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left 0"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

* Xuất mảng thu được và gửi mảng thu được qua cổng COM3 tới mạch Arduino.

print(finger)

        mySerial.sendData(finger)

* Cài đặt vị trí xuất ảnh tương ứng với yêu cầu người dùng ở góc trên bên trái màn hình, bên cạnh đó cài đặt khung ảnh đếm số ngón tay mở.

finger\_tuple = tuple(finger)

        if finger\_tuple in dicthand:

            image\_path = folderpath + dicthand[finger\_tuple]

            overlay\_img = cv2.imread(image\_path)

            if overlay\_img is not None:

                h, w, c = overlay\_img.shape

                img[0:h, 0:w] = overlay\_img

                cv2.rectangle(img, (0, 128), (128, 256), (0, 255, 0), cv2.FILLED)

                cv2.putText(img, str(finger.count(1)), (18, 245), cv2.FONT\_HERSHEY\_PLAIN, 10, (255, 0, 0), 10)

* Cài đặt tốc độ khung hình ứng với thời gian thực.

cTime = time.time()

    fps = 1 / (cTime - pTime)

    pTime = cTime

    cv2.putText(img, f'FPS: {int(fps)}', (950, 70), cv2.FONT\_HERSHEY\_PLAIN, 3, (255, 0, 0), 3)

    cv2.imshow("Image", img)

* Thực hiện chương trình và xuất kết quả.

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

        break

cap.release()

cv2.destroyAllWindows()

### Các bước thực hiện chi tiết

**Bước 1**: **Thu thập hình ảnh từ camera**

Sử dụng OpenCV để đọc hình ảnh từ camera. Các khung hình (frames) sẽ được xử lý lần lượt trong thời gian thực.

**Bước 2**: **Phát hiện bàn tay**

Dùng thư viện MediaPipe để phát hiện bàn tay trong khung hình. MediaPipe trả về tọa độ của 21 điểm trên bàn tay, giúp xác định chính xác vị trí các ngón tay và khớp nối.

**Bước 3**: **Theo dõi vị trí các ngón tay**

Sử dụng các tọa độ landmark từ MediaPipe để theo dõi chuyển động của các ngón tay. Các điểm này được dùng để nhận diện cử chỉ như số ngón tay giơ lên, nắm đấm hoặc mở tay.

**Bước 4**: **Nhận diện tay trái và tay phải**

**A hand with red dots and green lines

Description automatically generatedA close-up of a handprint

Description automatically generated**

Để phân biệt giữa tay trái và tay phải, so sánh vị trí số (1) của ngón cái với điểm gốc bàn tay (0) trên tọa độ của các điểm landmark. Điều này cho phép xác định xem tay nào đang xuất hiện trong khung hình.

* Nếu số (1) nằm bên trái thì đây là trường hợp bàn tay trái, nếu nằm bên phải thì đây là trường hợp bàn tay phải.
* Tuy nhiên đối với máy khi thực hiện chương trình sẽ không nhận biết được bên trái và bên phải như con người vì vậy vấn đề này được giải quyết dựa trên tọa độ điểm.
* Đối với khung hình cũng giống như tọa độ xOy trong không gian. Gốc của khung ảnh nằm phía trên bên trái, trục Ox tương ứng là cạnh trên, Oy tương ứng là cạnh bên trái. Càng về góc dưới bên phải màn hình tọa độ điểm đó càng lớn.
* Vậy nên để biết là tay trái hay tay phải ta so sánh trên trục hoành. Nếu khoảng cách từ trục Oy đến điểm (1) mà nhỏ hơn khoảng cách từ trục Oy đến điểm (0) thì đây là bàn tay trái và ngược lại đối với tay phải.

#Right

lmList[tipIds[0] - 3][1] > lmList[tipIds[0] - 4][1]

**Bước 5**: **Phản hồi cử chỉ**

Khi nhận diện được một cử chỉ tay, hệ thống sẽ thực hiện các hành động tương ứng, ví dụ như hiển thị một ảnh tương ứng hoặc điều khiển một thiết bị.

Đối với một bàn tay ta sẽ chia bàn tay thành 2 phần. Phần gồm ngón cái đến gốc bàn tay (0) và phần 4 ngón còn lại và gốc bàn tay (0).

Khi đã nhận diện được bàn tay trái hoặc bàn tay phải thì tiếp theo ta sẽ nhận diện cách mà các ngón tay hoạt động.

Cả 2 bàn tay sẽ có chung trường hợp là 4 ngón tay khác ngón cái và gốc bàn tay (0).

Nhận xét nếu ngón tay đóng tương ứng với kết quả trả về 0 sau đó thêm vào mảng listHand. Để biết được ngón tay đóng hay mở ta sẽ xét nếu điểm đầu các ngón khác ngón cái so với điểm đầu – 2 của chính ngón tay đó. Nếu khoảng cách từ điểm đầu ngón đến Ox mà nhỏ hơn khoảng cách từ điểm đầu ngón – 2 đến Ox thì ta nói ngón tay mở tương ứng kết quả trả về là 1. Thêm 1 vào listHand và trường hợp còn lại thêm 0.

for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

+ Đối với ngón cái bàn tay phải:

Ta xét khoảng cách điểm đầu ngón (4) đến trục Oy so với điểm đầu ngón – 1 (3). Nếu khoảng cách lớn hơn thì ta thêm 1 vào listHand và ngược lại thêm 0.

if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:

                    finger.append(1)

                else:

                    finger.append(0)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right 0"

+ Đối với ngón cái bàn tay trái:

Ta xét khoảng cách điểm đầu ngón (4) đến trục Oy so với điểm đầu ngón – 1 (3).

Nếu khoảng cách nhỏ hơn thì ta thêm 1 vào listHand và ngược lại thêm 0.

if lmList[tipIds[0]][1] < lmList[tipIds[0] - 1][1]:

                    finger.append(1)

                else:

                    finger.append(0)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left 0"

**Bước 6: Phản hồi cử chỉ chi tiết các trường hợp**

Tuy nhiên các ràng buộc cài đặt trên vẫn còn nhiều hạn chế. Chúng chỉ đáp ứng cho những trường hợp 2 bàn tay đặt thẳng và ngay ngắn.

Các trường hợp khác như nghiêng hay xoay góc bàn tay. Các ràng buộc bên trên không còn đáp ứng được yêu cầu người dùng. Cụ thể như ta xét sự đóng mở của các ngón tay dựa trên khoảng cách đến trục Ox, nhưng khi ta xoay bàn tay đi một góc lớn hơn 45 độ. Các điểm đó dao động ngang hàng hoặc chệch hướng thì đây là sự hạn chế của các câu ràng buộc trên.

Ta nhận thấy rằng tùy yêu cầu của người dùng mà ta xét sự đóng mở sẽ dựa trên khoảng cách đến trục Ox hoặc Oy sẽ phù hợp và đáp ứng với yêu cầu tương ứng.

Vậy nên ta cần cài đặt mở rộng thêm các ràng buộc tương ứng trong từng khoảng giá trị khác nhau.

* 1. Trường hợp bàn tay phải
     + Xoay góc 0 đến 90 độ phải

A diagram of a star

Description automatically generated

elif lmList[tipIds[4] - 3][1] > lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[1] - 3][1] > lmList[tipIds[0]][1]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

Xét khoảng cách từ Oy đến điểm (17) so với khoảng cách Oy đến điểm (0) nếu lớn hơn thì bàn tay đang xoay góc 0-90 độ right. 4 ngón khác ngón cái giữ nguyên. Xét điểm đầu ngón cái nếu khoảng cách từ Oy đến (4) lớn hơn Oy đến (5) thì thêm 1 còn lại thêm 0.

* + - Xoay góc 0 đến 90 độ trái

A diagram of a person's body

Description automatically generated with medium confidence

elif lmList[tipIds[1] - 3][1] < lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[0]][2] > lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

Xét khoảng cách từ Oy đến điểm (5) so với khoảng cách Oy đến điểm (0) nếu nhỏ hơn thì bàn tay đang xoay góc 0-90 độ left. 4 ngón khác ngón cái giữ nguyên. Xét điểm đầu ngón cái nếu khoảng cách từ Ox đến (4) nhỏ hơn Ox đến (2) thì thêm 1 còn lại thêm 0.

* + - Xoay góc lớn hơn 90 độ phải

A close-up of a diagram

Description automatically generated

elif lmList[tipIds[1] - 3][2] >= lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0] - 1][2] > lmList[tipIds[0]][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right over 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] < lmList[tipIds[id] - 2][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

Xét khoảng cách từ Ox đến điểm (5) so với khoảng cách Ox đến điểm (0) nếu lớn hơn thì bàn tay đang xoay góc lớn hơn 90 độ phải. 4 ngón khác ngón cái chuyển từ so sánh từ Ox sang Oy nếu khoảng các từ điểm đầu lớn hơn thì thêm 1 còn lại thêm 0. Xét điểm đầu ngón cái nếu khoảng cách từ Ox đến (4) nhỏ hơn Ox đến (3) thì thêm 0 còn lại thêm 1.

* + - Xoay góc lớn hơn 90 độ trái

A diagram of a tree

Description automatically generated

if lmList[tipIds[4] - 3][2] >= lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0]][2] > lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image right over -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] > lmList[tipIds[id] - 2][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

Xét khoảng cách từ Ox đến điểm (17) so với khoảng cách Ox đến điểm (0) nếu lớn hơn thì bàn tay đang xoay góc lớn hơn 90 độ trái. 4 ngón khác ngón cái chuyển từ so sánh từ Ox sang Oy nếu khoảng các từ điểm đầu lớn hơn thì thêm 0 còn lại thêm 1. Xét điểm đầu ngón cái nếu khoảng cách từ Ox đến (4) nhỏ hơn Ox đến (3) thì thêm 1 còn lại thêm 0.

* 1. Trường hợp bàn tay trái
     + Xoay góc 0 đến 90 độ phải

A diagram of a person's hand

Description automatically generated

elif lmList[tipIds[1] - 3][1] > lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[1] - 3][2] < lmList[tipIds[0]][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

Xét khoảng cách từ Oy đến điểm (5) so với khoảng cách Oy đến điểm (0) nếu lớn hơn thì bàn tay đang xoay góc 0-90 độ right. 4 ngón khác ngón cái giữ nguyên. Xét điểm đầu ngón cái nếu khoảng cách từ Ox đến (4) lớn hơn Ox đến (5) thì thêm 0 còn lại thêm 1.

* + - Xoay góc 0 đến 90 độ trái

A diagram of a hand with red dots

Description automatically generated

#0-90-left

            elif lmList[tipIds[4] - 3][1] < lmList[tipIds[0] - 4][1]:

                #

                if lmList[tipIds[0]][2] < lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                        finger.append(1)

                    else:

                        finger.append(0)

Xét khoảng cách từ Oy đến điểm (17) so với khoảng cách Oy đến điểm (0) nếu bé hơn thì bàn tay đang xoay góc 0-90 độ right. 4 ngón khác ngón cái giữ nguyên. Xét điểm đầu ngón cái nếu khoảng cách từ Oy đến (4) lớn hơn Oy đến (5) thì thêm 0 còn lại thêm 1.

* + - Xoay góc lớn hơn 90 độ phải

A diagram of a tree

Description automatically generated

if lmList[tipIds[4] - 3][2] > lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0]][2] > lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left over 90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] > lmList[tipIds[id] - 2][1]:

                        finger.append(1)

                    else:

                        finger.append(0)

Xét khoảng cách từ Ox đến điểm (17) so với khoảng cách Ox đến điểm (0) nếu lớn hơn thì bàn tay đang xoay góc lớn hơn 90 độ phải. 4 ngón khác ngón cái chuyển từ so sánh từ Ox sang Oy nếu khoảng các từ điểm đầu lớn hơn thì thêm 1 còn lại thêm 0. Xét điểm đầu ngón cái nếu khoảng cách từ Ox đến (4) nhỏ hơn Ox đến (3) thì thêm 1 còn lại thêm 0.

Tuy nhiên ở trường hợp này chương trình sẽ nhận đây là bàn tay phải bởi vì nếu theo ràng buộc của tay phải: Nếu điểm (1) nằm bên phải (0) thì đây là bàn tay phải tức là đã bị trùng với trường hợp bàn tay trái. Vì vậy ta tìm điểm khác nhau giữa 2 trường hợp và thêm nó vào ràng buộc nhận biết bàn tay phải:

if (lmList[tipIds[0] - 3][1] > lmList[tipIds[0] - 4][1] and lmList[tipIds[4] - 3][2] < lmList[tipIds[0] - 4][2]) or ((lmList[tipIds[0] - 2][2] < lmList[tipIds[1] - 3][2]) and (lmList[tipIds[0] - 4][1] > lmList[tipIds[1] - 3][1])):

* + - Xoay góc lớn hơn 90 độ trái

A diagram of a star

Description automatically generated with medium confidence

#>90-left

            elif lmList[tipIds[1] - 3][2] > lmList[tipIds[0] - 4][2]:

                #

                if lmList[tipIds[0]][2] < lmList[tipIds[0] - 1][2]:

                    finger.append(0)

                else:

                    finger.append(1)

                folderpath = "Python Project/Hand-Tracking/Image rotate/Image left over -90"

                for id in range(1, 5):

                    if lmList[tipIds[id]][1] > lmList[tipIds[id] - 2][1]:

                        finger.append(0)

                    else:

                        finger.append(1)

Xét khoảng cách từ Ox đến điểm (5) so với khoảng cách Ox đến điểm (0) nếu lớn hơn thì bàn tay đang xoay góc lớn hơn 90 độ trái. 4 ngón khác ngón cái chuyển từ so sánh từ Ox sang Oy nếu khoảng các từ điểm đầu lớn hơn thì thêm 0 còn lại thêm 1. Xét điểm đầu ngón cái nếu khoảng cách từ Ox đến (4) nhỏ hơn Ox đến (3) thì thêm 0 còn lại thêm 1.

**Bước 7: Lắp ráp phần cứng**

Lắp ráp các phần cứng cho bàn tay robot có các hoạt động cơ bản giống với bàn tay con người.

Sử dụng các thiết bị Servo sg90s, đèn led, điện trở, mạch Arduino Uno để thiết kế phần cứng.

**Bước 8: Cài đặt phần cứng**

Sử dụng Arduino để cài đặt cách tính năng và môi trường cơ bản cần thiết. Sau đó nạp chương trình vào main board Arduino Uno.

* 1. Khai báo các biến và thư viện cần thiết

#include <Servo.h>

#define numOfValsRec 5

#define digitsPerValSec 1

Servo servoThumb;

Servo servoIndex;

Servo servoMiddle;

Servo servoRing;

Servo servoPinky;

int valsRec[numOfValsRec];

int stringLen = numOfValsRec \* digitsPerValSec + 1; //$00000

int count = 0;

bool countstart = false;

String receiveString;

* 1. Trong void setup() cài đặt các servo và đèn led ứng với các cổng Digital tùy chỉnh

void setup() {

  Serial.begin(9600);

  servoThumb.attach(7);

  servoIndex.attach(9);

  servoMiddle.attach(11);

  servoRing.attach(8);

  servoPinky.attach(10);

  pinMode(2, OUTPUT);

  pinMode(3, OUTPUT);

  pinMode(4, OUTPUT);

  pinMode(5, OUTPUT);

  pinMode(6, OUTPUT);

}

* 1. Cài đặt hàm nhận dữ liệu, chia chuỗi, đếm và thêm các phần tử vào chuỗi receiveString. Sau đó gán các phần tử trong chuỗi receiveString vào Mảng valsRec.

void receiveData() {

  while(Serial.available()) {

    char c = Serial.read();

    if (c == '$') {

      countstart = true;

    }

    if (countstart) {

      if (count < stringLen) {

        receiveString = String(receiveString + c);

        count++;

      }

      if (count >= stringLen) {

        for (int i = 0; i < numOfValsRec; i++) {

          int num = (i \* digitsPerValSec) + 1;

          valsRec[i] = receiveString.substring(num, num + digitsPerValSec).toInt();

        }

        receiveString = "";

        count = 0;

        countstart = false;

      }

    }

  }

}

* 1. Hàm void loop() xét các trường hợp phần tử trong mảng trả về là 1 hoặc 0 từ đó servo sẽ quay 1 góc 180 độ, đèn led sẽ sáng.

void loop() {

  receiveData();

  if (valsRec[0] == 1) {

    servoThumb.write(180);

    digitalWrite(2, HIGH)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  else {

    servoThumb.write(0);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  if (valsRec[1] == 1) {

    servoIndex.write(0);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  else {

    servoIndex.write(180);

    digitalWrite(2, LOW)

    digitalWrite(3, HIGH)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  if (valsRec[2] == 1) {

    servoMiddle.write(0);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  else {

    servoMiddle.write(180);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, HIGH)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  if (valsRec[3] == 1) {

    servoRing.write(180);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, HIGH)

    digitalWrite(6, LOW)

  }

  else {

    servoRing.write(0);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  if (valsRec[4] == 1) {

    servoPinky.write(0);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, LOW)

  }

  else {

    servoPinky.write(180);

    digitalWrite(2, LOW)

    digitalWrite(3, LOW)

    digitalWrite(4, LOW)

    digitalWrite(5, LOW)

    digitalWrite(6, HIGH)

  }

}

**Bước 9: Cài đặt kết nối với phần cứng**

Thiết lập kết nối và đảm bảo phần cứng hoạt động ổn định.

detector = htm.handDetector(detectionCon=0.75)

mySerial = cvzone.SerialModule.SerialObject("COM3", 9600, 1)

## Kết quả

### Kết quả nhận diện bàn tay

Hệ thống nhận diện bàn tay và theo dõi chính xác các điểm landmark trong thời gian thực, với độ trễ thấp.

### Nhận diện tay trái và tay phải

Hệ thống có khả năng phân biệt giữa tay trái và tay phải dựa trên vị trí của ngón cái so với bàn tay.

### Nhận diện cử chỉ tay

Hệ thống phản hồi gần như chính xác và đầy đủ tất cả các trường hợp đóng mở ngón tay hay xoay các góc của bàn tay.

### Độ chính xác

* Hệ thống hoạt động ổn định với độ chính xác cao trong điều kiện ánh sáng tốt.
* Một số trường hợp hệ thống gặp khó khăn trong điều kiện ánh sáng yếu hoặc khi bàn tay bị che khuất một phần.

### Phần cứng

* Phần cứng hoạt động tốt các tính năng đã thiết lập.
* Hạn chế:

+ Sai số kỹ thuật khi lắp ráp các linh kiện.

+ Phần cứng hoạt động có phần chưa ổn định.

A close-up of a machine

Description automatically generated Close-up of several wires

Description automatically generated

## Đánh giá và kết luận

### Đánh giá

* Hệ thống và phần cứng đã đáp ứng được yêu cầu cơ bản về phát hiện và theo dõi cử chỉ tay trong thời gian thực.
* Sử dụng MediaPipe giúp tăng tốc độ phát triển và đạt hiệu suất cao.
* Tuy nhiên, tổng thể cần được cải tiến để hoạt động tốt hơn trong môi trường có ánh sáng kém hoặc khi bàn tay bị xoay nghiêng. Hay phần cứng chưa thực sự ổn định.

### Kết luận

Đề tài Hand Tracking là một ứng dụng thú vị và tiềm năng trong lĩnh vực tương tác người-máy. Hệ thống có thể được mở rộng để ứng dụng trong nhiều lĩnh vực như điều khiển thiết bị bằng cử chỉ, tương tác thực tế ảo, hoặc hỗ trợ người khuyết tật.

## Hướng phát triển

### Mở rộng hệ thống nhận diện cử chỉ phức tạp hơn

Hệ thống có thể mở rộng để nhận diện thêm nhiều cử chỉ phức tạp hơn, chẳng hạn như cử chỉ trong ngôn ngữ ký hiệu (Sign Language). Các trường hợp khác của xoay hướng bàn tay.

### Tích hợp các hệ thống điều khiển

Mở rộng khả năng ứng dụng của hệ thống bằng cách tích hợp nó với các thiết bị điều khiển như TV, máy tính, hoặc hệ thống nhà thông minh.

### Cải thiện khả năng hoạt động trong điều kiện thực tế

Cải thiện khả năng nhận diện trong điều kiện ánh sáng yếu hoặc khi bàn tay không hoàn toàn nằm trong khung hình.

### Phát triển thêm sự nhận diện cánh tay

Phát triển sự nhận diện không chỉ là bàn tay và các trường hợp cử chỉ toàn bộ cánh tay phức tạp hơn.

### Phát triển phần cứng

Phát triển và cải tiến phần cứng có khả năng cầm nắm, giảm thiểu sai số kỹ thuật, nâng cấp và thiết kế lại kết cấu bàn tay. Thêm các bộ phận cần thiết hoặc lược bỏ các bộ phận dư thừa.

KẾT LUẬN

Em xin gửi lời biết ơn sâu sắc đến thầy Mai Trung Thành đã dành nhiều thời gian và tâm huyết hướng dẫn nghiên cứu và giúp em hoàn thành môn học.

Em đã có nhiều cố gắng hoàn thiện dự án bằng tất cả năng lực của mình, tuy nhiên không thể tránh khỏi nhiều thiếu sót, rất mong nhận được những đóng góp quý báu của quý thầy cô và các bạn.