**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN VIỄN THÔNG**

---------------o0o---------------



**BÁO CÁO BÀI TẬP LỚN   
MÔN XỬ LÝ ẢNH**

**ĐIỀU KHIỂN TRÒ CHƠI   
BẰNG CỬ CHỈ TAY**

**GVGD: Võ Tuấn Kiệt**

**SVTH: Nguyễn Văn Minh - 1813098**

**Đào Phú Trọng - 1613766**

**Nguyễn Hữu Nghĩa - 1512157**

**TP. HỒ CHÍ MINH, THÁNG 3 NĂM 2022**

**TÓM TẮT**

Hệ thống nhận dạng cử chỉ nhằm hỗ trợ con người thực hiện các hoạt động một cách đơn giản nhanh chóng. Hệ thống dựa trên sự tương tác giữa người và máy tính bằng cách thu thập cử chỉ ngón tay để xử lý. Hệ thống cho phép con người truyền ý định và mục đích của họ bằng cách sử dụng cử chỉ ngón tay giữa không trung để đưa ra mệnh lệnh. Quy trình hệ thống được chia thành năm phần: thu nhận hình ảnh, trừ nền, phân đoạn tay, trích xuất và nhận dạng cử chỉ. Ngôn ngữ python được sử dụng vì nó cung cấp cú pháp dễ dàng cho phép mã hóa nhanh chóng và cung cấp các thư viện tiêu chuẩn khác nhau. Thư viện nguồn mở OpenCV cũng được sử dụng cho các chức năng khác liên quan đến theo dõi đối tượng và xử lý ảnh. Những điểm yếu của hệ thống sẽ được ghi lại như một phần của công việc trong tương lai của dự án để đạt được sự cải thiện tốt hơn.

**MỤC LỤC**

[I. GIỚI THIỆU 6](#_Toc99631790)

[1.1 Tổng quan 6](#_Toc99631791)

[1.2 Động lực 6](#_Toc99631792)

[1.3 Nhiệm vụ đề tài 6](#_Toc99631793)

[II. LÝ THUYẾT 6](#_Toc99631794)

[2.1 Xử lý ảnh - Thị giác máy tính 6](#_Toc99631795)

[2.1.1 Nhận dạng cử chỉ tay 6](#_Toc99631796)

[2.1.2 Phương pháp tiếp cận 7](#_Toc99631797)

[2.1.3 Trừ nền 8](#_Toc99631798)

[2.1.4 Phân đoạn hình ảnh 9](#_Toc99631799)

[2.1.5 Không gian màu RGB 9](#_Toc99631800)

[2.1.6 Không gian màu HSV 9](#_Toc99631801)

[2.1.7 RGB to HSV 10](#_Toc99631802)

[2.1.8 Image Thresholding 11](#_Toc99631803)

[2.1.9 Biến đổi hình thái 12](#_Toc99631804)

[2.1.10 Làm mịn ảnh 13](#_Toc99631805)

[2.1.11 Phát hiện đường viền 13](#_Toc99631806)

[2.2 Thuật toán học máy KNN 14](#_Toc99631807)

[III. THIẾT KẾ HỆ THỐNG 16](#_Toc99631808)

[3.1 Các yêu cầu về tính năng 16](#_Toc99631809)

[3.2 Block Diagram 17](#_Toc99631810)

[3.2 Sơ đồ khối chi tiết 19](#_Toc99631811)

[3.2.1 Thu nhận hình ảnh 19](#_Toc99631812)

[3.2.2 Trừ nền 20](#_Toc99631813)

[3.2.3 Phân đoạn bàn tay 21](#_Toc99631814)

[3.2.4 Trích xuất tính năng 22](#_Toc99631815)

[3.2.5 Nhận diện cử chỉ 23](#_Toc99631816)

[IV. PHẦM MỀM SỬ DỤNG 24](#_Toc99631817)

[4.1 Phần mềm và Framework 24](#_Toc99631818)

[4.1.1 Google Colab, 24](#_Toc99631819)

[4.1.2 Pycharm 24](#_Toc99631820)

[4.1.2 OpenCV 25](#_Toc99631821)

[4.1.3 Scikit-learn 25](#_Toc99631822)

[4.2 Ngôn ngữ lập trình 26](#_Toc99631823)

[4.2.1 Python 26](#_Toc99631824)

[V. CÁC BƯỚC THỰC HIỆN VÀ KẾT QUẢ 26](#_Toc99631825)

[5.1 Dự đoán cử chỉ tay sử dụng các phép tính toán học. 26](#_Toc99631826)

[5.1.1 Thu thập hình ảnh 26](#_Toc99631827)

[5.1.2 Trừ nền 27](#_Toc99631828)

[5.1.3 Phân đoạn hình ảnh 28](#_Toc99631829)

[5.1.4 Trích xuất tính năng 29](#_Toc99631830)

[5.1.5 Kết quả 30](#_Toc99631831)

[5.2 Dự đoán cử chỉ tay tay sử dụng học máy KNN 31](#_Toc99631832)

[5.2.1 Thu thập dataset 31](#_Toc99631833)

[5.2.2 Xử lý dataset 32](#_Toc99631834)

[5.2.3 Chia dataset thành các tệp Train-Test 32](#_Toc99631835)

[5.2.4 Kết quả 33](#_Toc99631836)

[VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 33](#_Toc99631837)

[6.1 Kết luận 33](#_Toc99631838)

[6.2 Hướng phát triển 34](#_Toc99631839)

[VII. TÀI LIỆU THAM KHẢO 34](#_Toc99631840)

[VIII. PHỤ LỤC 34](#_Toc99631841)

# I. GIỚI THIỆU

## 1.1 Tổng quan

Trước đây, mọi người thường điểu khiển trò chơi bằng cách tác động vật lý lên chuột và bàn phím để thực hiện việc điều khiển nó. Khi họ thực hiện các thao tác này có thể khiến họ mất tập trung vào những thứ khác.

## 1.2 Động lực

Nhằm giúp con người tránh mất tập trung, mong đợi một giải pháp để đơn giản hóa và nâng cao tương tác giữa con người và máy tính cho phép trình điều khiển tương tác với hệ thống giải trí sử dụng cử chỉ tay giữa không trung mà vẫn không mất tập trung.

Trong trường hợp này tương tác cử chỉ tay có thể là sự thay thế tuyệt vời cho việc tương tác vật lý. Bằng cách sử dụng hình ảnh cụ thể các thuật toán và kỹ thuật xử lý, cử chỉ tay được phát hiện nhận dạng và tạo ra câu lệnh tương ứng đầu vào để điều khiển.

## 1.3 Nhiệm vụ đề tài

Đề tài áp dụng các kỹ thuật và thuật toán để xử lý việc nhận dạng cử chỉ ngón tay.

# II. LÝ THUYẾT

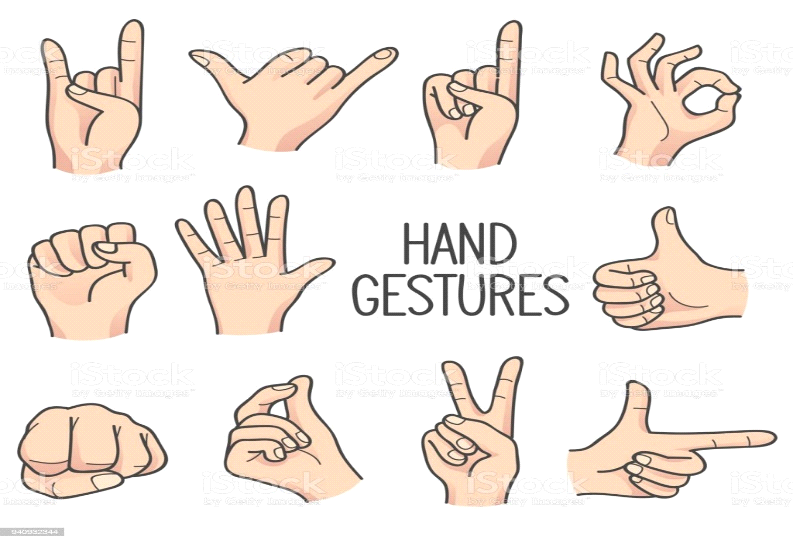
## 2.1 Xử lý ảnh - Thị giác máy tính

Thị giác máy tính và xử lý ảnh là một quy trình liên quan đến việc cách máy tính tự động trích xuất, phân tích và hiểu thông tin từ một hình ảnh hoặc chuỗi video. Sự hiểu biết có thể đạt được nhờ phát triển cơ sở lý thuyết và thuật toán và nó tìm cách tự động hóa các nhiệm vụ có thể được thực hiện bởi hệ thống

   
Hình 2.1 Computer Vision

### 2.1.1 Nhận dạng cử chỉ tay

Cử chỉ là một hình thức giao tiếp không lời liên quan đến con người tương tác hằng ngày và nó hỗ trợ việc truyền đạt ý nghĩa của một hành động hay lời nói đặc biệt với những người có vấn đề về nói hoặc nghe.



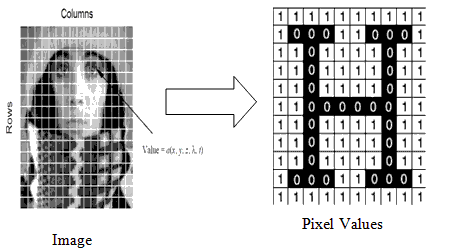
Hình 2.2 Hand gestures

Nhận dạng cử chỉ tay đề cập tới việc theo dõi và ghi nhận các chuyển động của con người xảy ra ở một hoặc hai bàn tay. Kết quả mong muốn nhận dạng được hành động hoặc kí hiệu tay đang thể hiện là gì.

### 2.1.2 Phương pháp tiếp cận

Tiếp cận dựa trên việc xử lý ảnh sao cho có thể phát hiện được sự xuất hiện của bàn tay bằng cách trích xuất các đặc điểm như hình ảnh, màu sắc, diện tích, đường viền, luồng, pixel. Cách tiếp cận này thường nhạy cảm vì có nhiều yếu tố tác động đến như nền, màu da .. v.v Xem xét các yếu tố như thời gian xử lý, độ trễ, giới hạn tài nguyên…

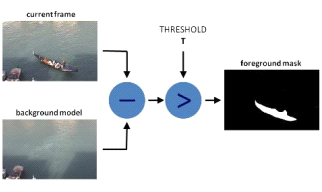
Để thực hiện nhận dạng cử chỉ, xử lý ảnh là bước quan trọng trong xử lý chuỗi video thô trước khi phân tích thông tin hình ảnh và trích xuất thông tin cần thiết để xử lý thêm. Nó bao gồm một số quy trình trong đó bao gồm trừ nền, phân đoạn hình ảnh và trích xuất các tính năng.



Hình 2.1.3: Xử lý ảnh

### 2.1.3 Trừ nền

Trừ nền là một kỹ thuật xử lý hình ảnh phổ biển để loại bỏ nền không cần thiết, nhiễu liên quan đến hình ảnh và tạo ra mặt nạ tiền ảnh là khu vực quan tâm yêu cầu được trích xuất từ hình ảnh. Đối với dự án này BackGroundSubtractorMOG2 từ thư viện OpenCV sẽ được triển khai cho quá trình trừ nền trong việc trích xuất mặt nạ tiền cảnh là bàn tay để được sử dụng cho các bước tiếp theo



Hình 2.1.4: Phương pháp trừ nền

### 2.1.4 Phân đoạn hình ảnh

Phân đoạn hình ảnh là quá trình phân tách giữa các vùng quan tâm và các phân đoạn không mong muốn ở dạng pixel để làm cho hình ảnh có ý nghĩa và dễ dàng hơn khi được phân tích. Đối với dự án này thì phần được tách ra khỏi hình ảnh là vùng bàn tay.

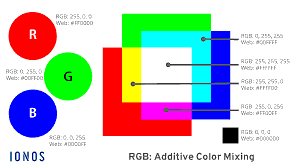
A picture containing graphical user interface

Description automatically generated

Hình 2.5: Phân đoạn hình ảnh

### 2.1.5 Không gian màu RGB

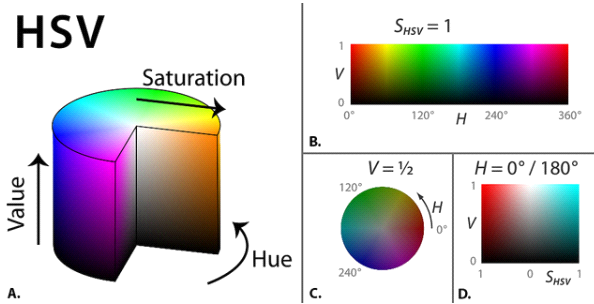
RGB đại diện cho màu đỏ, xanh lục và xanh lam là một trong những không gian màu phổ biển được hầu hết các thiết bị chụp ảnh sử dụng để lưu trữ và xử lý dữ liệu hình ảnh kỹ thuật số. Nó không thích hợp cho việc phát hiện và phân tích dựa trên màu sắc. Do đó nó phải được chuyển sang một không gian màu khác là HSV



Hình 2.1.6: Không gian màu RGB

### 2.1.6 Không gian màu HSV

Không gian màu HSV xác định phần màu (Hue) về mức độ xám trong màu sắc, (Saturation) và giá trị độ sáng của nó (Value). Đó là một không gian màu đơn giản hơn bao gồm ít màu hơn pixel phù hợp hơn để đại diện cho hình ảnh trong việc phát hiện màu da.



Hình 2.7: HSV Color Space

### 2.1.7 RGB to HSV

Công thức chuyển đổi RGB sang HSV :

Các giá trị R, G, B được chia cho 255 để thay đổi phạm vi trừ 0 .. 255 thành 0..1”

R’ = R/255

G’ = G/255

B’ = B/255

Cmax = max(R’, G’, B’)

Cmin = min(R’, G’, B’)



Tính toán Hue:



Tính toán độ bão hòa :



Tính toán giá trị

V= Cmax

Bên trên là lý thuyết khi chuyển đổi màu RGB sang HSV. Đề tài sử dụng trực tiếp các thư viện chuyển RGB2HSV của OpenCV.

### 2.1.8 Image Thresholding

Ngưỡng đề cập đến việc gán giá trị pixel thành màu đen hoặc màu trắng dựa trên ngưỡng. Thư viện OpenCV đã cung cấp hàm ngưỡng để giải quyết vấn đề này.



Hình 2.8: Image Thresholding

Bánh xe màu HSV đôi khi xuất hiện dưới dạng hình nón hoặc hình trụ, nhưng luôn có ba thành phần sau:

**Hue** là phần màu của mô hình, được biểu thị bằng một số từ 0 đến 360 độ:

Màu đỏ rơi vào khoảng từ 0 đến 60 độ.

Màu vàng rơi vào khoảng từ 61 đến 120 độ.

Màu xanh lá cây rơi vào khoảng từ 121 đến 180 độ.

Màu lục lam rơi vào khoảng từ 181 đến 240 độ.

Màu xanh lam rơi vào khoảng từ 241 đến 300 độ.

Màu đỏ tươi rơi vào khoảng từ 301 đến 360 độ.

**Độ bão hòa** mô tả lượng màu xám trong một màu cụ thể, từ 0 đến 100 phần trăm. Giảm thành phần này về 0 sẽ tạo ra nhiều màu xám hơn và tạo ra hiệu ứng mờ dần. Đôi khi, độ bão hòa xuất hiện dưới dạng phạm vi từ 0 đến 1, trong đó 0 là màu xám và 1 là màu chính.

**Giá trị (hoặc Độ sáng)** hoạt động cùng với độ bão hòa và mô tả độ sáng hoặc cường độ của màu, từ 0 đến 100 phần trăm, trong đó 0 là màu đen hoàn toàn và 100 là sáng nhất và hiển thị nhiều màu nhất.

### 2.1.9 Biến đổi hình thái

Hình ảnh lấy ngưỡng phải được làm mịn và lọc để giảm nhiễu.

Nó sẽ ảnh hưởng đến việc phát hiện đối tượng và sự công nhận. Thư viện OpenCV cũng đã có một số hàm để thực hiện việc đó như erosion, dilation và opening sẽ được sử dụng trong đề tài này.

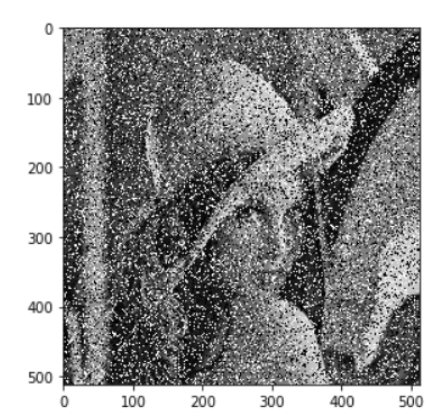


Hình 2.9: Erosion và Dilation



Hình 2.10: Opening

### 2.1.10 Làm mịn ảnh



Hình bên trên là một ví dụ minh họa cho ảnh bị nhiễu. Một cách trực quan, bạn có thể định nghĩa điểm nhiễu là những điểm đốm trắng đen khiếh cho hình khó nhìn. Nếu để ý kĩ hơn, bạn có thể thấy, các điểm nhiễu này có mức sáng (intensity) chênh lệch rõ ràng so với những điểm xung quanh (điểm trắng nổi lên giữa khu vực màu đen, etc.)

Để loại bỏ những điểm nhiễu này, hay nói cách khác, loại bỏ những điểm có mức sáng chênh lệch lớn bất thường, Làm mờ ảnh (blurring) là một trong những phương pháp phổ biến nhất được áp dụng.

Vậy làm thế nào để làm mờ ảnh ? Làm thế nào để làm mờ những điểm có độ sáng bất thường trên ảnh ? Biện pháp đơn giản nhất chính là thay thế những điểm có độ sáng bất thường bằng trung bình độ sáng của những điểm lân cận.

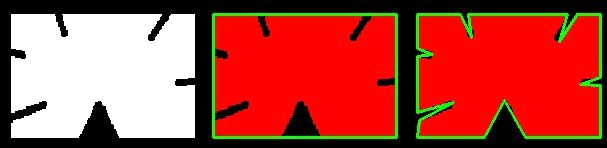
Với tư tưởng trên, chúng ta sẽ tính tích chập của ảnh với ma trận nhân (kernel) có giá trị

blur\_normalized

Khi nhân tích chập với ma trận này, giá trị đầu ra của điểm được tính sẽ bằng trung bình cộng của 8 điểm xung quanh và giá trị của chính điểm đó.

### 2.1.11 Phát hiện đường viền

Đường viền của một đối tượng hình ảnh đề cập đến việc kết nối tất cả các đường bao liên tục chỉ bởi một đường cong. Trong xử lý ảnh đó là một yếu tố quan trọng để phát hiện và nhận dạng đối tượng. Nó có thể được thực hiện bằng cách sử dụng hàm cv2.findcontour trong thư viện OpenCV và tìm đường viền tối đa là hình ảnh bàn tay ta.



Hình 2.11: Đường viền

## 2.2 Thuật toán học máy KNN

Thuật toán K-láng giềng gần nhất (KNN) là một loại thuật toán ML có giám sát có thể được sử dụng cho cả phân loại cũng như các bài toán dự đoán hồi quy. Tuy nhiên, nó chủ yếu được sử dụng để phân loại các vấn đề dự đoán trong công nghiệp. Hai thuộc tính sau đây sẽ xác định tốt KNN:

● Thuật toán lười học - KNN là một thuật toán lười học vì nó không có giai đoạn huấn luyện chuyên biệt và sử dụng tất cả dữ liệu để huấn luyện trong khi phân loại.

● Thuật toán học không tham số - KNN cũng là một thuật toán học không tham số vì nó không giả định bất kỳ điều gì về dữ liệu cơ bản

Thuật toán KNN cho rằng những dữ liệu tương tự nhau sẽ tồn tại gần nhau trong một không gian, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất. Việc tìm khoảng cách giữa 2 điểm củng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp.

Diagram, schematic

Description automatically generated

Minh hoạ nguyên lý hoạt động của thuật toán KNN

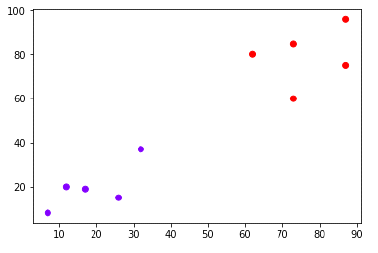
Text, letter

Description automatically generated Khoảng cách trong không gian vector – trong không gian một chiều, khoảng cách giữa hai điểm là trị tuyệt đối giữa hiệu giá trị của hai điểm đó. Trong không gian nhiều chiều, khoảng cách giữa hai điểm có thể được định nghĩa bằng nhiều hàm số khác nhau, trong đó độ dài đường thằng nổi hai điểm chỉ là một trường hợp đặc biệt trong đó. Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính:

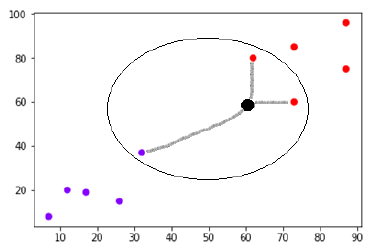
Ví dụ:

Sau đây là một ví dụ để hiểu khái niệm K và hoạt động của thuật toán KNN –

Giả sử chúng ta có một tập dữ liệu có thể được vẽ như sau:



Bây giờ, chúng ta cần phân loại điểm dữ liệu mới với chấm đen (tại điểm 60,60) thành lớp màu xanh lam hoặc màu đỏ. Chúng tôi giả sử K = 3 tức là nó sẽ tìm thấy ba điểm dữ liệu gần nhất. Nó được hiển thị trong sơ đồ tiếp theo –



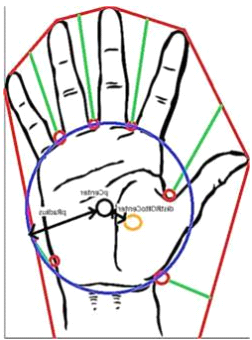
Chúng ta có thể thấy trong biểu đồ trên ba láng giềng gần nhất của điểm dữ liệu với chấm đen. Trong số ba người đó, hai trong số họ nằm trong lớp Đỏ do đó chấm đen cũng sẽ được chỉ định trong lớp đỏ.

# III. THIẾT KẾ HỆ THỐNG

## 3.1 Các yêu cầu về tính năng

Hệ thống phải đạt các yêu cầu sau:

* Chuyển video theo thời gian thực từ webcam thành hình ảnh để dự đoán cử chỉ
* Phát hiện và trích xuất được hình ảnh bàn tay.
* Chuyển đổi hình ảnh từ màu RGB sang màu HSV
* Loại bỏ nền ảnh và tối ưu việc khử nhiễu
* Thực hiện dự đoán cử chỉ bàn tay, bằng phương pháp toán học hoặc học máy.
* Từ kết quả dự đoán xuất ra phím nhấn ảo để điều khiển trò chơi

 Graphical user interface

Description automatically generated

Hình 3.1: Hình ảnh minh hoạ phương pháp toán học và học máy.

## 3.2 Block Diagram

Sơ đồ 3.1 biểu diễn sơ về các bước thực thi hệ thống. Hệ thống có thể được chia thành năm giai đoạn đó là :

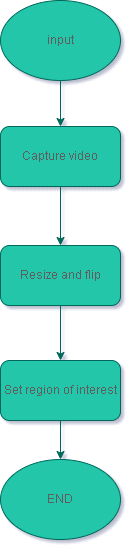
* Thu nhận hình ảnh
* Thực hiện phương pháp trừ nền để thu tiền cảnh
* Tiến hành phân đoạn hình ảnh
* Thưc hiện việc trích xuất tính năng.
* Và cuối cùng là nhận diện cử chỉ và đưa ra tín hiệu thông báo gì đó.

Mỗi giai đoạn cần phải hoàn thành trước khi bước sang giai đoạn tiếp theo. Thu nhận hình ảnh là giai đoạn nắm bắt bàn tay của người dùng làm đầu vào và chuẩn bị cho giai đoạn tiếp theo. Bốn giai đoạn tiếp theo là giai đoạn xử lý cốt lõi của hệ thống.

* **Thu nhận ảnh**
* Để chụp chuỗi video hình ảnh bàn tay người dùng làm đầu vào hệ thống
* Sau đó khung hình ảnh được thay đổi kích thước, lật và xác định ROI cho quá trình tiếp theo trong việc trích xuất thông tin hữu ích
* **Trừ nền**
* Để xử lý chuỗi video để trích xuất bàn tay của người dùng và loại bỏ nền và noise không cần thiết
* Trừ nền, chuyển đổi không gian màu, ngưỡng và biến đổi hình thái sẽ được thực hiện để chuẩn bị hình ảnh nhị phân của bàn tay người dùng mà không có đối tượng không cần thiết và noise từ nền.
* **Phân đoạn ảnh**
* Để có được đường viền tối đa của bàn tay
* Đường viền bàn tay thu được từ hình ảnh bàn tay nhị phân và lấy đường viền lớn nhất cho giai đoạn tiếp theo.
* **Trích xuất tính năng**
* Để thu thập tính năng của bàn tay như là thông tin hữu tích để phân tích và quyết định ý nghĩa của cử chỉ để thực hiện chức năng cụ thể.
* Đó là quá trình chuyển đổi hình ảnh thành một tập hợp các tính năng như trung tâm lòng bàn tay, convex hull, đầu ngón tay, diện tích bàn tay, góc ngón tay ..v .v
* **Nhận dạng cử chỉ**
* Áp dụng để nhận diện cử chỉ bàn tay và hiển thị
* Cử chỉ sẽ được xác định bằng diện tích, tỉ lệ diện tích, số ngón tay, góc của ngón tay.
* **Điều khiển trò chơi**
* Áp dụng kết quả dự đoán cử chỉ ngón tay để xuất ra tín hiệu nút nhấn ảo để thực hiện điều khiển trò chơi.

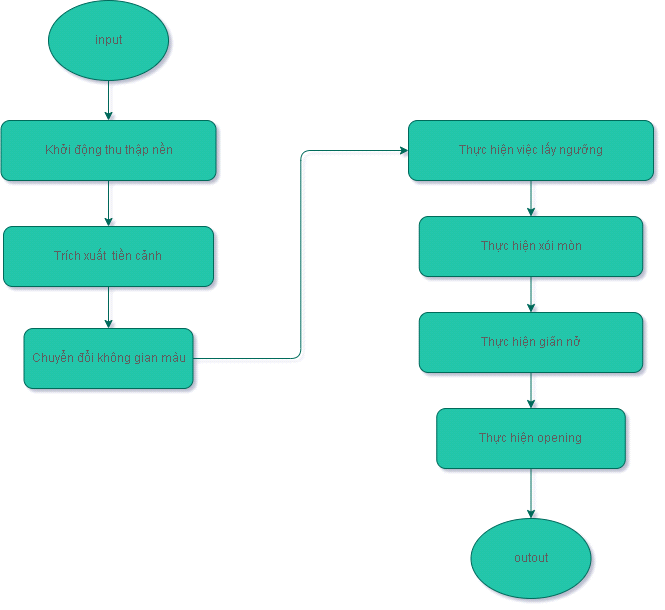
## 3.2 Sơ đồ khối chi tiết

### 3.2.1 Thu nhận hình ảnh



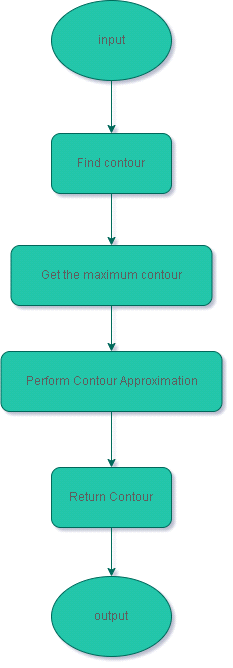
Hình 3.2 Sơ đồ khối chi tiết thu nhận ảnh video

### 3.2.2 Trừ nền



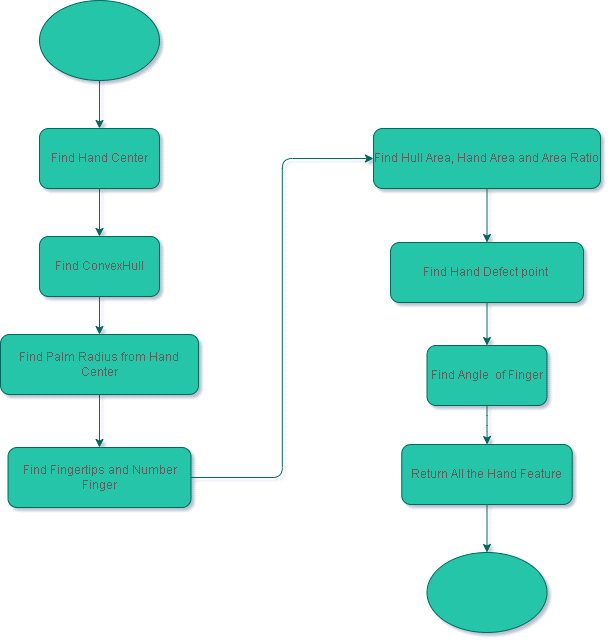
Hình 3.3: Sơ đồ khối chi tiết trừ nền

### 3.2.3 Phân đoạn bàn tay



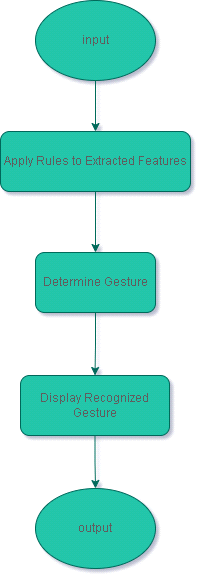
Hình 3.4 Sơ đồ khối chi tiết Phân đoạn bàn tay

### 3.2.4 Trích xuất tính năng



Hình 3.5 Sơ đồ khối trích xuất tính năng bàn tay

### 3.2.5 Nhận diện cử chỉ



Hình 3.6 Sơ đồ khối nhận diện cử chỉ

# IV. PHẦM MỀM SỬ DỤNG

## 4.1 Phần mềm và Framework

### 4.1.1 Google Colab,

Với A.I , DeepLearning, Google không chỉ có những sản phẩm dự án đã và đang phát triển. Google còn có công cụ giúp cho các developer phát triển, train model, một cách dễ dàng nhất. Đó chính là Google Colab.

Colaboratory hay còn gọi là Google Colab, là một sản phẩm từ Google Research, nó cho phép chạy các dòng code python thông qua trình duyệt, đặc biệt phù hợp với Data analysis, machine learning và giáo dục. Colab không cần yêu cầu cài đặt hay cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt, bạn có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao và cả GPUs và cả TPUs đều được cung cấp cho bạn.



Colab cung cấp nhiều loại GPU, thường là Nvidia K80s, T4s, P4s and P100s, tuy nhiên người dùng không thể chọn loại GPU trong Colab, GPU trong Colab thay đổi theo thời gian. Vì là dịch vụ miễn phí, nên Colab sẽ có những thứ tự ưu tiên trong việc sử dụng tài nguyên hệ thống, cũng như giới hạn thời gian sử dụng, thời gian sử dụng tối đa lên tới 12 giờ.

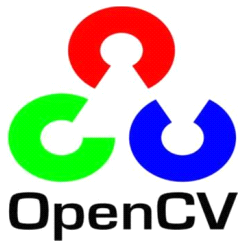
### 4.1.2 Pycharm

PyCharm là môi trường phát triển tích hợp đa nền tảng (IDE) được phát triển bởi Jet Brains và được thiết kế đặc biệt cho Python. PyCharm có mặt trên cả 3 nền tảng Windows, Linux và Mac OS. Pycharm là một giải pháp phù hợp cho Python developers vì IDE này hỗ trợ nhiều extensions, môi trường ảo (Virtual Environment), nhiều tính năng thông minh như bộ code completion, tự động thụt lề, phát hiện văn bản trùng lặp và kiểm tra lỗi. Ngoài ra còn có các tính năng tìm kiếm mã nguồn thông minh để tìm kiếm từng từ một trong nháy mắt.



Đề tài này yêu cầu kết nối và xử lý ảnh thu thập từ video webcam, ngoài ra thực hiện xuất ra nút nhấn ảo để chơi game. Nên ta sẽ xây dựng model trên Colab sau đó tải model về và chạy dự đoán bằng IDE Pycharm.

### 4.1.2 OpenCV



Hình 4.2 OpenCV

OpenCV là một thư viện nguồn mở của các chức năng lập trình chủ yếu nhắm vào thị giác máy tính thời gian thực ban đầu được phát triển bởi Intel. Thư viện đa nền tảng và miễn phí sử dụng theo giấy phép BSD. OpenCV đã cung cấp nhiều chức năng khác nhau liên quan đến theo dõi đối tượng và xử lý ảnh được triển khai trong đề tài này.

### 4.1.3 Scikit-learn

Nếu bạn đang sử dụng Python và đang muốn tìm một thư viện mạnh mẽ mà bạn có thể mang các thuật toán học máy (machine learning) vào trong một hệ thống thì không còn thư viện nào thích hợp hơn scikit-learn. Thư viện này tích hợp rất nhiều thuật toán hiện đại và cổ điển giúp bạn vừa học vừa tiến hành đưa ra các giải pháp hữu ích cho bài toán của bạn một cách đơn giản.



Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

## 4.2 Ngôn ngữ lập trình

### 4.2.1 Python



Hình 4.3 Python

Chương trình được thực hiện bởi ngôn ngữ python là một một ngôn ngữ bậc cao cung cấp nhiều cấu trúc dễ dàng cho phép lập trình nhanh khi so sánh với các ngôn ngữ khác. Đồng thời python cũng cung cấp rất nhiều thư viện tiêu chuẩn giúp thực hiện các cấu trúc phức tạp khá dễ dàng. Trong khi các thuật toán phức tạp và quy trình làm việc linh hoạt của Học máy và AI dễ làm hệ thống trở nên phức tạp thì, tính đơn giản của Python là giải pháp cho phép các lập trình viên viết ra các hệ thống đáng tin cậy hơn

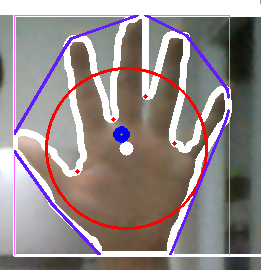
# V. CÁC BƯỚC THỰC HIỆN VÀ KẾT QUẢ

## 5.1 Dự đoán cử chỉ tay sử dụng các phép tính toán học.

* Chương này sẽ tập trung vào các giai đoạn thực hiện của đề tài mô tả thuật toán và tính toán cần thiết để đạt được các mục tiêu của dự án
* Các chức năng OpenCV liên quan đến theo dõi đối tượng và nhận dạng sẽ được sử dụng để hỗ trợ hệ thống
* Về cơ bản hệ thống nhận dạng cử chỉ theo thời gian được thể hiện ở dưới

### 5.1.1 Thu thập hình ảnh

* Chúng ta sẽ sử dụng một hàm của OpenCV để thu thập video từ webcam *cv2.VideoCapture(1)*.
* Nếu video được quay thành công tiếp đến sẽ lật ảnh để tránh chế độ xem gương và resize video theo một kích cỡ để tính toán.
* Tiếp đến thiết lập vùng tính toán để cải thiện hiệu quả hệ thống
* Vùng hình chữ nhật xanh là vùng được quan tâm



Hình 5.1.1 Bàn tay được thu thập từ video

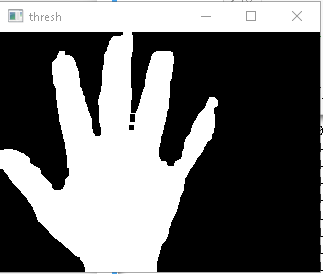
### 5.1.2 Trừ nền

* Ta khởi động bước trừ nền bằng cv2.createBackGroundSubtractorMOG2() với giá trị ngưỡng điều chỉnh và đặt phát hiện bóng thành sai để tránh bóng được công nhận là một phần của đối tượng
* Bước tiếp theo sẽ áp dụng chuỗi video cho trừ nền đã được khởi tạo với learning rate = 0 để trích xuất tiền cảnh từ nền và noise không mong muốn



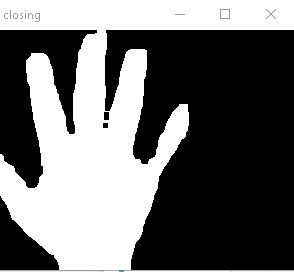
Hình 5.1.2.a Bàn tay nhận được sau khi trừ nền

* Vì các hình ảnh thu thập trong video webcame là không gian màu RGB, không phù hợp để phát hiện và phân tích dựa trên màu sắc. Vì vậy nó phải chuyển đổi sang một không gian màu khác là HSV sử dụng hàm có sẵn của OpenCV là cv2.cvtColor(img,cv2.COLOR\_BGR2HSV).
* Sau đó ta áp dụng bộ lọc CV2.inRange() để trích xuất các pixel hình ảnh nằm trong phạm vi được xác định trước của ngưỡng HSV. Khi đó ta có thể cài đặt phạm vi chặn ngưỡng của HSV bằng cách sử dụng thanh trackbar để điều chỉnh các thông số HSV min max sao cho thu nhập được hình ảnh bàn tay dưới dạng trắng đen một cách tốt nhất



Hình 5.1.2.b Lấy ngưỡng cho bàn tay.

* Sau đó OTSU thresholding sẽ được sử dụng bằng hàm cv2.threshold() để chuyển đổi thành hình ảnh nhị phân chỉ bao gồm màu đen và màu trắng dựa trên giá trị pixel đến giá trị ngưỡng. Các vùng bàn tay phát hiện sẽ được đặt thành màu trắng và vùng không mong muốn sẽ là màu đen.
* Bước cuối cùng của trừ nền ta sẽ chuyển đổi hình thái học bao gồm xói mòn giãn nở để hình ảnh ngưỡng mượt hơn và giảm nhiễu. Thông qua các hàm cv2.erode(), cv2.dialate(), cv2.morphologyEx()



Hình 5.1.2.c Hình ảnh được lọc

### 5.1.3 Phân đoạn hình ảnh

* Hình ảnh được lọc sẽ được áp dùng tìm đường viền bằng cách sử dụng cv2.findContours() và lấy đường viền tối đa trong hình ảnh dựa trên đường bao khu vực.
* Sau đó phép gần đúng đường bao được thực hiện để gần đúng hình dạng đường bao và làm mịn các cạnh của đường viền bằng cách sử dụng cv2.appoxPolyDP() với đường bao con gần đúng nhất.
* Cuối cùng đường viền tay sẽ được vẽ bằng cách sử dụng cv2.drawContours()



Hình 5.1.3 Vẽ đường viền bàn tay

### 5.1.4 Trích xuất tính năng

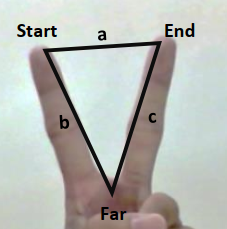
Trích xuất tính năng là một trong phần chính của nhận diện cử chỉ tay bằng xử lý ảnh. Hệ thống giúp chuyển đổi hình ảnh được phân đoạn thành một tập hợp các tính năng để phân tích và xác định ý nghĩa trong cử chỉ.

Đầu tiên là ta có thể tìm tâm bàn tay bằng cách sử dụng hàm cv2.moment() để tìm tâm của lòng bàn tay.



Tiếp đến ta tìm đường viền bao quanh bằng cv2.convexHull() để lấy bán kính lòng bàn tay bằng cách tính khoảng cách tới lòng bàn tay. Scikit-learning đã cung cấp một chức năng để tính khoảng cách từ một điểm đến nhiều điểm. Khi đó ta giả sử bán kính lòng bàn tay bằng 60% khoảng cách tối đa.

Tiếp đó ta có thể xác định các điểm lõm lồi bằng cách sử dụng hàm cv2.convexityDefect() sẽ trả về các điểm tương ứng với các đặc điểm trên đường bao. Ta sẽ sử dụng các đặc điểm này để tính toán.



Hình 5.1.4: Start, End, Far ở ConvexityDefect

Từ các điểm ở trên ta sẽ tính được các khoảng cách a,b,c:







Áp dụng định lý cosin trong tam giác ta tìm được góc ở far tạo bởi b và c:

Đồng thời khoảng cách từ far cho đến đoạn a dựa theo diện tích tam giác trên:

S = (a+b+c)/2

Ar = √[(s\*(s-a) \*(s-b) \*(s-c)]

D = 2\*Ar/a

Bước cuối cùng ta sẽ đếm có một điểm far chỉ khi khi góc far nhỏ hơn 90 độ và khoảng cách từ điểm far đến đoạn a phải lớn hơn bán kính bàn tay. Nghĩa là khi có 2 ngón được đưa lên sẽ có 1 điểm/góc far, tương tự 3 ngón đưa lên sẽ có 2 điểm/góc far.

Diagram

Description automatically generated 

Để phân biệt giữa nắm tay và 1 ngón tay ta đo khoảng cách từ tâm bàn tay cho đến điểm cao nhất để phân biệt đồng thời góc tan tạo bởi ngón tay với tâm một góc bé hơn 90 độ.

### 5.1.5 Kết quả

Mô hình sử dụng xử lý ảnh cơ bản và áp dụng toán học hoạt động hiểu quả. Phân loại rất tốt ở các ngón tay 0-2-3-4-5. Tuy nhiên vẫn còn hay nhầm lẫn giữa 0 ngón tay va 1 ngón tay.

Link video hoạt động của phương pháp toán học:  
 [Qr code

Description automatically generated](https://drive.google.com/file/d/1rD8mMF8929K-H1uZanrnK_3cqblfkVZG/view?usp=sharing)

<https://drive.google.com/file/d/1rD8mMF8929K-H1uZanrnK_3cqblfkVZG/view?usp=sharing>

## 5.2 Dự đoán cử chỉ tay tay sử dụng học máy KNN

Phương án này tiếp cận bài toán dưới ý tưởng sẽ dùng các thuật toán học máy (Machine learning) dạng phân loại (Classification) để dự đoán ra hình ảnh bàn tay đang thể hiện mấy ngón. Ý tưởng này yêu cầu việc thu thập các hình ảnh bàn tay cho 6 class (bao gồm nắm tay, số ngón tay 1-2-3-4-5).

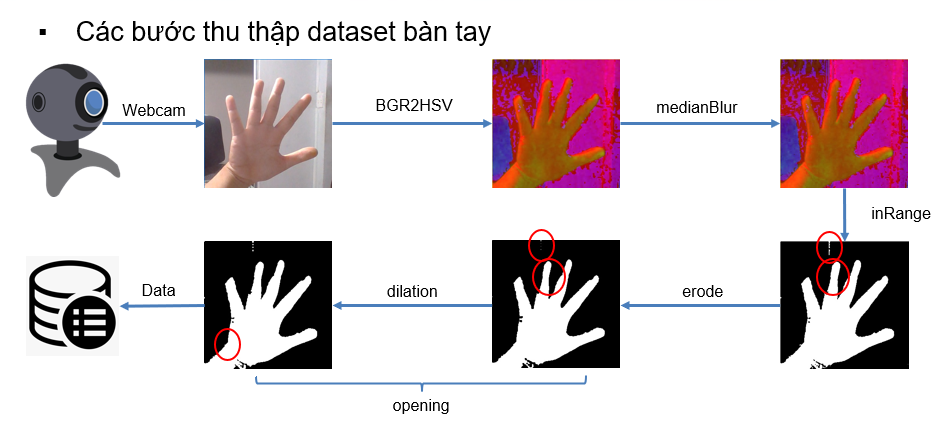
Sau đó tiến hành huấn luyện với các thuật toán Học máy nền tảng như KNN (K-Nearest Neighbor), ngoài ra nếu có thể sẽ khảo sát so sánh với việc sử các Transfer model khác có sẵn như VGG, MobileNet, ResNet.

Diagram

Description automatically generated

### 5.2.1 Thu thập dataset

Ta tiến hành sử dụng một file python để tiến hành thu thập các hình ảnh bàn tay với 6 class (bao gồm nắm tay, số ngón tay 1-2-3-4-5). Trước hết từ video webcam ta crop một khung ảnh hình vuông có kích thước 224x224 pixel. Khung ảnh sau khi được trích xuất từ video webcam sẽ được trải qua các bước xử lý tương tự như phương án toán học. Sau đó tất cả hình ảnh này sẽ được đánh số và lưu thành các folder data riêng tương ứng.



Mỗi class sẽ được thu thập 1000 ảnh với các vị trí bàn tay khác nhau để làm dữ liệu training cho mô hình học máy. Do webcam của người làm đề tài có tốc độ bằng 30 fps nên việc cắt ra từng frame ảnh cho 1000 ảnh sẽ tốn khoảng 33 giây.

Link video demo thu thập dataset:  
 [Qr code

Description automatically generated](https://drive.google.com/file/d/1eEfQx032z4YZ2glLQkkVb_jrWFKRA2nP/view?usp=sharing)

<https://drive.google.com/file/d/1eEfQx032z4YZ2glLQkkVb_jrWFKRA2nP/view?usp=sharing>

### 5.2.2 Xử lý dataset

Data sau khi được thu thập sẽ dưới dạng ảnh trắng đen và sẽ resize về 64x64 pixel và được chuyển về dạng ma trận numpy array (n, 64, 64). Label sẽ được chuyển về dạng categorical one hot vector với 6 classes. Do thuật toán KneighborsClassifier ta chỉ áp dụng được với dữ liệu 1 chiều/1 hàng (1-dimesion) nên sau đó ta reshape lại thành ma trận numpy array với (n, 64\*64) = (n, 4096).

### 5.2.3 Chia dataset thành các tệp Train-Test

Dataset sẽ sau khi được thu thập sẽ được chia thành 2 tệp train và tệp test với tỉ lệ 80%-20% (60-20-20% nếu chia thành 3 tệp train-validation-test khi huấn luyện các model DL khác). Phần training data sẽ được dùng huấn luyện model, phần testing data sẽ được dùng để so sánh giữa các model với nhau.

A picture containing bar chart

Description automatically generated

### 5.2.4 Kết quả

Ở đề tài này, sinh viên làm đề tải sử dụng module thuật toán KneighborsClassifier có sẵn trên Sklearn để thực hiện. Với các thông số mặc định và cho n\_neighbor chạy từ 1-10 để tìm ra số lượng k neighbor nào cho kết quả với tệp test tốt nhất. => Chọn được n\_neighbors tốt nhất bằng 2 với độ chính xác trên tệp test là 98,67%. Sau đó tiến hành predict trên tệp test.

import time

start = time.time()

predictions = neigh.predict(d2\_test\_dataset)

end = time.time()

print(classification\_report(y\_test, predictions, digits = 4))

print(end-start)

Table

Description automatically generatedCalendar

Description automatically generated

Classification Report và Confusion matrix của mô hình KNN

Mô hình dự đoán có độ chính xác cao – đạt 98,67%. Độ chính xác giữa các lớp cũng tương đối giống nhau. Chứng tỏ mô hình dự đoán đã hoạt động khá tốt.

Thời gian dự đoán của tệp test với 1200 hình ảnh tiêu tốn 1,68 giây, chỉ tốn 0,0014 giây/hình (server colab). Tuy nhiên model KNN khi tải về khá nặng, lên đến 153 MB.

Link video demo dự đoán của model KNN:  
 [Qr code

Description automatically generated](https://drive.google.com/file/d/1hUuB_Yq617LardyXtdGFVAI9d1JbeJVF/view?usp=sharing)

<https://drive.google.com/file/d/1hUuB_Yq617LardyXtdGFVAI9d1JbeJVF/view?usp=sharing>

# VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 6.1 Kết luận

Trong bài tập lớn này, nhóm thực hiện đã xây dựng thành công 2 mô hình để nhận dạng cử chi tay: sử dụng toán học và học máy. Bao gồm việc hoàn thành các bước sau:

* Thực hiện phát triển và lập trình hệ thống bằng Python
* Kết hợp một số framework và thư viện như OpenCV, numpy, keras, sklearn,…
* Áp dụng các phương pháp xử lý ảnh như chuyển đổi hệ màu, làm mịn, khử nhiễu, phát hiện viền,…
* Thu thập được dataset để training cho mô hình học máy
* Thực hiện nhận dạng bằng phương pháp toán học hoặc học máy và trả về kết quả nhận dạng cử chỉ của tay.
* Sau đó từ kết quả nhận dạng xuất được thành công nút nhấn ảo để tương tác với các trò chơi theo thời gian thực

Hệ thống về cơ bản đáp ứng được các yêu cầu đề ra là nhận dạng tương đối chính xác một vài cử chỉ tay để điều khiển trò chơi.

## 6.2 Hướng phát triển

Trong tương lai hệ thống có thể phát triển bằng cách ứng dụng các thuật toán cao cấp hoặc mạng tích chập như Support Vector Machine hay Convolutionnal Neutral Network hoặc các model thông dụng hiện nay để đạt được hiệu suất tốt.

# VII. TÀI LIỆU THAM KHẢO

1. Vũ Hữu Tiệp, *Machine learning cơ bản*, Ebook, 2018.
2. Nguyễn Thanh Tuấn, *Deep learning cơ bản*, Ebook, 2020.
3. [Online website] - <https://machinelearningcoban.com/>
4. [Online website] - <https://codelearn.io/sharing/xu-ly-anh-voi-opencv-phan-1>
5. [Online website] - <https://github.com/opencv/opencv>
6. [Online website] - <https://towardsdatascience.com/>
7. [Online website] - https://scikit-learn.org/

# VIII. PHỤ LỤC

Link GGDrive video và code demo: <https://drive.google.com/drive/folders/1jnW7GtwuGYOuQSichFwg28J_D7kkD1Gl?usp=sharing>

Qr code

Description automatically generated