

TRƯỜNG ĐẠI HỌC BÁCH KHOA
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



ĐỒ ÁN THIẾT KẾ LUẬN LÝ

Hệ thống mở mật khẩu bằng cử chỉ bàn tay
qua camera

GIÁO VIÊN HƯỚNG DẪN: Trần Thanh Bình
DANH SÁCH THÀNH VIÊN: Lê Quốc Huy 1913516
Lê Thành Long 1913991
Trần Đình Nghĩa 1914325

Thành phố Hồ Chí Minh, 12/2021



Contents

I TỔNG QUAN	2
1 GIỚI THIỆU VỀ ĐỀ TÀI	2
2 MỤC TIÊU VÀ NHIỆM VỤ	2
2.1 MỤC TIÊU CỦA ĐỀ TÀI	2
2.2 NHIỆM VỤ CỦA ĐỀ TÀI	2
3 ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU	2
4 PHƯƠNG PHÁP NGHIÊN CỨU	3
4.1 PHƯƠNG PHÁP TÀI LIỆU	3
4.2 PHƯƠNG PHÁP THỰC NGHIỆM	3
5 Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN	3
5.1 Ý NGHĨA KHOA HỌC	3
5.2 Ý NGHĨA THỰC TIỄN	3
6 CƠ SỞ LÝ THUYẾT	3
6.1 PHẦN MỀM HỖ TRỢ: PYTHON	3
6.2 THƯ VIÊN ĐƯỢC SỬ DỤNG	4
6.2.1 OpenCV	4
6.2.2 Tesseract: Nhận dạng ký tự quang học	5
6.2.3 MediaPipe: Thị giác máy tính	6
6.2.4 Telegram - bot	7
7 CÁCH THÚC HOẠT ĐỘNG VÀ MÔ PHỎNG	8
7.1 Bot_Telegram	8
7.2 HandTrackingModule.py	9
7.3 read_img.py	11
7.4 SetPassWord.py	12
7.5 Final.py	16
II NHẬN XÉT	22
III TÀI LIỆU THAM KHẢO	24



I TỔNG QUAN

1 GIỚI THIỆU VỀ ĐỀ TÀI

Ngày nay cùng với sự phát triển không ngừng của khoa học kĩ thuật, nhiều sản phẩm kĩ thuật số ra đời và đáp ứng được nhiều nhu cầu và lợi ích cho con người, các bộ ứng dụng có khả năng xử lý nhiều hoạt động phức tạp mà chỉ cần một camera nhỏ, nó có thể hoạt động dựa trên những gì nó được lập trình.

Hệ thống này không những góp phần vào kĩ thuật điều khiển từ xa mà còn góp phần to lớn vào việc phát triển các phương thức truyền nhận thông tin. Chính vì vậy mà nhóm thực hiện đề tài **Hệ thống mở cửa từ xa với cử chỉ ngón tay** trong khuôn khổ của môn đồ án thiết kế luận lý do thầy Trần Thanh Bình hướng dẫn.

2 MỤC TIÊU VÀ NHIỆM VỤ

2.1 MỤC TIÊU CỦA ĐỀ TÀI

- Mục tiêu của nhóm khi thực hiện đề tài trước hết là muốn phát huy thành quả của trí tuệ nhân tạo để tạo ra sản phẩm có ích trong đời sống con người chúng ta.

- Ngoài ra, quá trình làm đề tài là khoảng thời gian quý báu để nhóm thực hiện đề tài và có thể tự kiểm tra lại những kiến thức đã học ở trường để giải quyết những vấn đề, yêu cầu mà ban đầu đã đặt ra.

2.2 NHIỆM VỤ CỦA ĐỀ TÀI

- Sử dụng **kiến thức lập trình python** để điều khiển toàn bộ hệ thống.
- Hiển thị khung xương ngón tay.
- Hiển thị được chữ viết trên màn hình.
- So sánh kết quả đã lưu với chữ viết trên màn hình.
- Hiển thị kết quả.

3 ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

- **Đối tượng nghiên cứu:** Hệ thống nhận diện bàn tay và đối chiếu chữ viết
- **Phạm vi đề tài:**
 - Thiết kế ứng dụng đơn giản
 - Sử dụng kiến thức python



4 PHƯƠNG PHÁP NGHIÊN CỨU

4.1 PHƯƠNG PHÁP TÀI LIỆU

- Nghiên cứu về nguyên lý hoạt động của python thông qua các tài liệu trên internet.
- Tìm hiểu kĩ hơn về **thư viện và cách thức hoạt động**.

4.2 PHƯƠNG PHÁP THỰC NGHIỆM

- Mô phỏng sản phẩm trên phần mềm **visual studio code**.

5 Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN

5.1 Ý NGHĨA KHOA HỌC

- Có ý nghĩa quan trọng đối với sự phát triển của từng cá nhân trong nhóm và đối với hệ thống điều khiển từ xa
- Vận dụng một cách tổng hợp những kiến thức đã học để tiến hành thiết kế, lập trình ứng dụng và hoàn thiện vốn hiểu biết của bản thân.

5.2 Ý NGHĨA THỰC TIỄN

- Đề xuất được các phương pháp thiết kế sản phẩm vào việc mở khóa trong mùa dịch
- Ngoài ra, đề tài còn có thể giúp cho người mới học về kỹ thuật lập trình có thể tiếp thu thêm kiến thức mới về lập trình để ứng dụng vào các nhu cầu của mình trong sinh hoạt đời sống và làm việc.

6 CƠ SỞ LÝ THUYẾT

6.1 PHẦN MỀM HỖ TRỢ: PYTHON

- Giới thiệu
 - Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình và là ngôn ngữ lập trình dễ học; được dùng rộng rãi trong phát triển trí tuệ nhân tạo. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.
 - Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.
- Triết lý thiết kế và tính năng
 - Python là một ngôn ngữ lập trình đa mẫu hình. Lập trình hướng đối tượng và lập trình cấu trúc được hỗ trợ hoàn toàn, và nhiều tính năng của nó cũng hỗ trợ lập trình hàm và lập trình hướng khía cạnh bao gồm siêu lập trình và siêu đối tượng (phương thức thẳn kì).



Các mẫu hình khác cũng được hỗ trợ thông qua các phần mở rộng, bao gồm thiết kế theo hợp đồng và lập trình logic.

- Lý do chọn python

- Đơn giản: Cú pháp đơn giản giúp cho người lập trình dễ dàng đọc và tìm hiểu.
- Tương tác: Chế độ tương tác cho phép người lập trình thử nghiệm tương tác sửa lỗi của các đoạn mã.
- Chất lượng: Thư viện có tiêu chuẩn cao, Python có khối cơ sở dữ liệu khá lớn nhằm cung cấp giao diện cho tất cả các CSDL thương mại lớn.
- Thuận tiện: Python được biên dịch và chạy trên tất cả các nền tảng lớn hiện nay.
- Mở rộng: Với tính năng này, Python cho phép người lập trình có thể thêm hoặc tùy chỉnh các công cụ nhằm tối đa hiệu quả có thể đạt được trong công việc.
- Tốc độ: Python có tốc độ xử lý nhanh hơn so với ngôn ngữ PHP.

6.2 THƯ VIỆN ĐƯỢC SỬ DỤNG

6.2.1 OpenCV

- OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning, và hiện có thêm tính năng tăng tốc GPU cho các hoạt động theo real-time.
- Có trên các giao diện C++, C, Python , Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để hỗ trợ hiệu quả về tính toán và chuyên dùng cho các ứng dụng real-time (thời gian thực). Nếu được viết trên C/C++ tối ưu, thư viện này có thể tận dụng được bộ xử lý đa lõi (multi-core processing).
- OpenCV được sử dụng cho đa dạng nhiều mục đích và ứng dụng khác nhau bao gồm:

- Hình ảnh street view
- Kiểm tra và giám sát tự động
- Robot và xe hơi tự lái
- Phân tích hình ảnh y học
- Tìm kiếm và phục hồi hình ảnh/video
- Phim và cấu trúc 3D từ chuyển động
- Nghệ thuật sắp đặt tương tác



Hình 1: Hình ảnh street view

Theo tính năng và ứng dụng của OpenCV, có thể chia thư viện này thành các nhóm tính năng và module tương ứng như sau:

- Xử lý và hiển thị Hình ảnh Video/ I/O (core, imgproc, highgui)
- Phát hiện các vật thể (objdetect, features2d, nonfree) Geometry-based monocular hoặc stereo computer vision (calib3d, stitching, videostab)....

6.2.2 Tesseract: Nhận dạng ký tự quang học

Là loại công cụ máy tính được tạo ra để chuyển các hình ảnh của chữ viết tay hoặc chữ đánh máy (thường được quét bằng máy scanner) thành các văn bản tài liệu. OCR được hình thành từ một lĩnh vực nghiên cứu về nhận dạng mẫu, trí tuệ nhận tạo và machine vision.

Nhận dạng ký tự quang học (dùng các kỹ thuật quang học chẳng hạn như gương và ống kính) và nhận dạng ký tự số (sử dụng máy quét và các thuật toán máy tính) lúc đầu được xem xét như hai lĩnh vực khác nhau. Bởi vì chỉ có rất ít các ứng dụng tồn tại với các kỹ thuật quang học thực sự, bởi vậy thuật ngữ Nhận dạng ký tự quang học được mở rộng và bao gồm luôn ý nghĩa nhận dạng ký tự số.

Ứng dụng của Tesseract:

- Chuyển đổi tài liệu giấy in thành tài liệu văn bản có thể đọc được bằng máy.
- Nhận dạng văn bản bên trong hình ảnh.....



Hình 2: Nhận dạng văn bản bên trong hình ảnh

6.2.3 MediaPipe: Thị giác máy tính

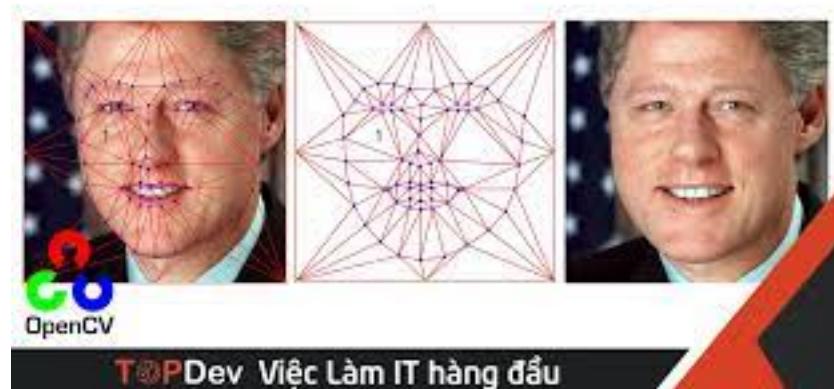
- MediaPipe là một gói Python được tạo sẵn trên PyPI. Nó cũng cung cấp các công cụ để người dùng xây dựng các giải pháp của riêng họ. Gói PythonPipe Python có sẵn trên PyPI cho Linux, macOS và Windows.

- Có thể được định nghĩa là một lĩnh vực trí tuệ nhân tạo đào tạo máy tính để giải thích và hiểu thế giới thị giác. Sử dụng hình ảnh kỹ thuật số từ máy ảnh, video và mô hình học sâu, máy móc có thể xác định và phân loại chính xác các đối tượng.

Thị giác máy tính là một lĩnh vực khoa học liên ngành đề cập đến cách máy tính có thể đạt được hiểu biết cấp cao từ hình ảnh hoặc video kỹ thuật số.

Những điều có thể thực hiện với sự trợ giúp của MediaPipe:

- Xe tự hành
- Nhận diện khuôn mặt và lối khuôn mặt
- Tư thế và phát hiện toàn diện
- Theo dõi và phát hiện đối tượng
- Theo dõi và phát hiện đối tượng
- Đặt ra ước tính
- Dếm ngón tay
- Bộ âm lượng



Hình 3: Nhận diện khuôn mặt và lối khuôn mặt



6.2.4 Telegram - bot

- **Bot Telegram** là giống như một robot có sẵn trong ứng dụng nhằm giúp người dùng tạo lập và quản lý các bot. Nếu bạn lần đầu biết đến Bot API Telegram. Người dùng có thể điều khiển các bot để nhận thông báo và tin tức; tạo các công cụ tùy chỉnh; trải nghiệm trò chơi; tích hợp với các dịch vụ như Gmail Bot, GIF Bot, Wiki Bot,... Đặc biệt là sử dụng Bot Father để thực hiện tạo New Bot. Việc cài đặt Bot không khó như nhiều người suy nghĩ. Bạn có thể dễ dàng thực hiện việc này ngay cả khi bạn sử dụng điện thoại hay máy tính.



- Để tạo virtual environment và cài đặt thư viện python-telegram-bot ta dùng các lệnh sau:

```
virtualenv env
source env/bin/activate
pip install python-telegram-bot
```

- Ta có thể truy cập vào các **HTTP API** và lấy các **chat ID** để thực hiện các lệnh cho bot theo hướng dẫn sau:

<https://core.telegram.org/bots>



7 CÁCH THỨC HOẠT ĐỘNG VÀ MÔ PHỎNG

- Đối tượng cụ thể mà nhóm hướng đến ở trong đồ án lần này đó chính là mở khóa cửa tự động bằng cử chỉ của bàn tay
- Đầu tiên ta tải Tesseract-OCR bằng cách truy cập vào đường dẫn <https://digi.bib.uni-mannheim.de/tesseract/tesseract-ocr-w64-setup-v5.0.0.20211201.exe> sau đó cài đặt và đảm bảo sao cho ta có đường dẫn sau:

```
C:\Program Files\Tesseract-OCR\tesseract.exe
```

- Nhóm tạo ra 4 file lập trình python với các chức năng khác nhau nhưng đồng thời hỗ trợ lẫn nhau có tên lần lượt là **HandTrackingModule.py**; **SetPassWord.py**; **Final.py**; **read_img.py**; **Bot_Telegram**

- Ý tưởng để làm bài đồ án này là tạo ra khung xương của bàn tay rồi sẽ là tọa độ của 1 đầu ngón tay vẽ lên màn hình thông qua camera, khi hình ảnh được vẽ xong thì sẽ lưu thành password (lần 1) và chuyển hình ảnh sang file text khi cần mở password thì sẽ vẽ thêm lần 2 và chuyển sang file text, máy tính sẽ so sánh 2 file text lại với nhau. Khi giống nhau sẽ thông báo và mở cửa, khi khác nhau thì sẽ không mở cửa và thông báo thất bại.

7.1 Bot_Telegram

```
import telegram
import cv2
import os
def bot_message():
    try:
        #Use this token to access the HTTP API:
        notify = telegram.Bot("5012281775:AAH0bp5Wx5tBwRtCQfP4SzoyRI5EdKBjbtw")
        #Read file
        f1 = open("PassLog/Log.txt", 'r+', encoding='UTF-8')
        data1 = f1.read()
        message1 = "{}".format(data1)
        notify.send_message(chat_id="-795707678", text=message1, parse_mode='Markdown')
    except Exception as ex:
        print(ex)

def bot_photo():
    try:
        notify = telegram.Bot("5012281775:AAH0bp5Wx5tBwRtCQfP4SzoyRI5EdKBjbtw")
        notify.send_photo(chat_id="-795707678", photo = open('Fail/human.png', 'rb'),
                          caption = "Someone try to open the door? Do you know this person?")
    except Exception as ex:
        print(ex)
```

- **Chức năng:** để hiện thị và thông báo lên group Telegram về trạng thái của cửa bằng cách truy cập vào **HTTP API**



7.2 HandTrackingModule.py

```
import mediapipe as mp
import time
import cv2
class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.detectionCon,
                                       self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw = True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                              self.mpHands.HAND_CONNECTIONS)
        return img
    def findPos(self, img, handNo=0, draw=True):
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                #print(id, lm)
                h, w, c = img.shape
                cx, cy = int(lm.x*w), int(lm.y*h)
                #print(id, cx, cy)\\
                self.lmList.append([id, cx, cy])
            #Draw Dot
            if draw:
                cv2.circle(img, (cx,cy), 10, (140, 228, 109), cv2.FILLED)
        return self.lmList
    def fingerUP(self):
        fingers = []
        #Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0]-1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
        #4 fingers
        for id in range(1,5):
            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id]-2][2]:
                fingers.append(1)
            else:
```



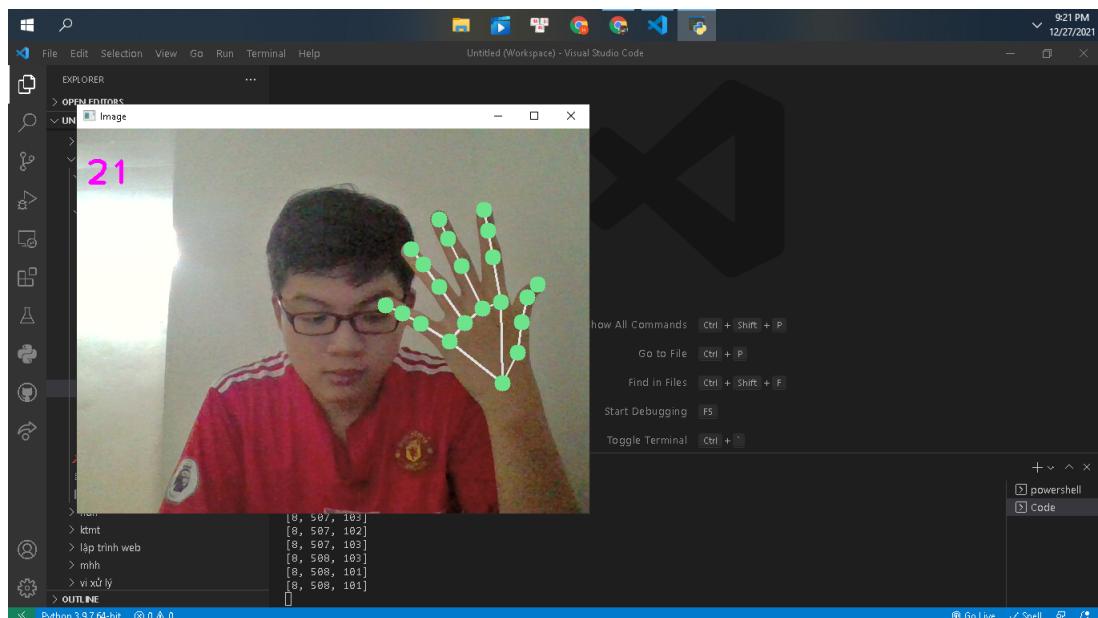
```
fingers.append(0)
return fingers\\

def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = detector.findPos(img)
        if len(lmList)!=0:
            print(lmList[8])

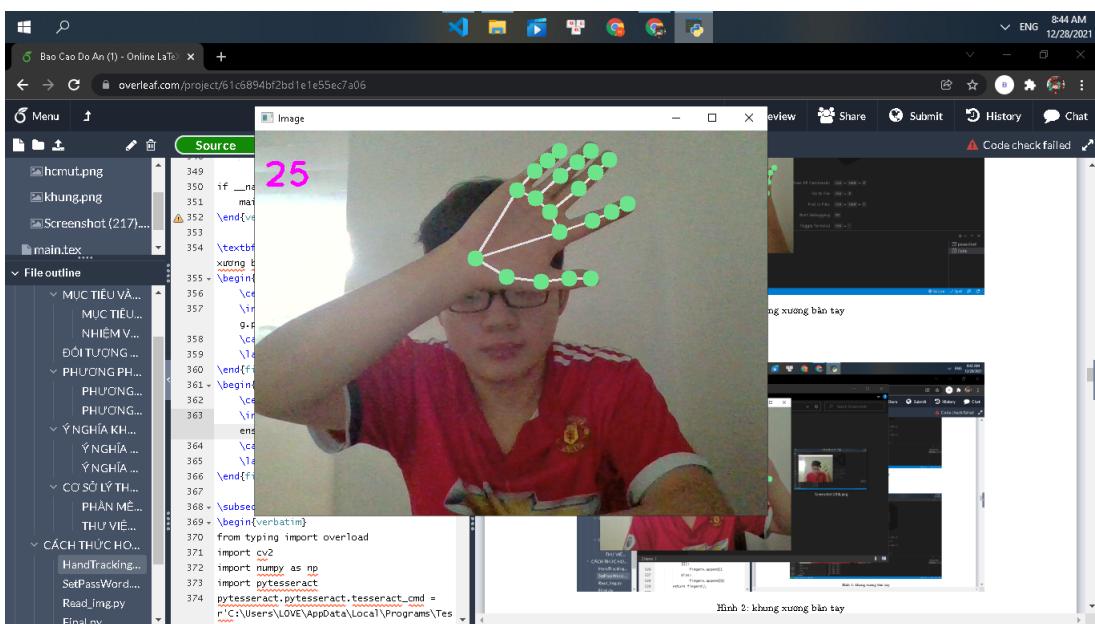
        cTime = time.time()
        fps = 1/(cTime-pTime)
        pTime = cTime
        cv2.putText(img, str(int(fps)), (10,70), cv2.FONT_HERSHEY_PLAIN, 3, (255, 0, 255), 3)
        cv2.imshow("Image", img)
        cv2.waitKey(1)

if __name__ == "__main__":
    main()\\
```

- Chức năng: nhận diện khung xương bàn tay, dùng các thư viện openCV và mediapipe



Hình 4: Khung xương bàn tay



Hình 5: Khung xương bàn tay

7.3 read_img.py

```
import os
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
import cv2

Path = "Image"
myImage = os.listdir(Path)
core = "password_record.png"
if core in myImage:
    image = cv2.imread('Image/password_record.png', cv2.IMREAD_UNCHANGED)
else:
    image = cv2.imread('Image/blank.png', cv2.IMREAD_UNCHANGED)
#Use Page Segmentation modes and OCR engine modes
cs_config = r'-c tessedit_char_whitelist=ABCDEFGHIJKLMN --psm 7 --oem 3'
image_text = pytesseract.image_to_string(image, lang='eng', config=cs_config)
```

- **Chức năng:** đọc ký tự trên ảnh và chuyển thành string dựa vào các **Page Segmentation modes** và **OCR Engine modes**



Page segmentation modes:

0. Orientation and script detection (OSD) only.
1. Automatic page segmentation with OSD.
2. Automatic page segmentation, but no OSD, or OCR. (not implemented)
3. Fully automatic page segmentation, but no OSD. (Default)
4. Assume a single column of text of variable sizes.
5. Assume a single uniform block of vertically aligned text.
6. Assume a single uniform block of text.
7. Treat the image as a single text line.
8. Treat the image as a single word.
9. Treat the image as a single word in a circle.
10. Treat the image as a single character.
11. Sparse text. Find as much text as possible in no particular order.
12. Sparse text with OSD.
13. Raw line. Treat the image as a single text line, bypassing hacks that are Tesseract-specific.

OCR Engine modes:

0. Legacy engine only.
1. Neural nets LSTM engine only.
2. Legacy + LSTM engines.
3. Default, based on what is available.

Hình 6: Multiple config option

7.4 SetPassWord.py

```
from typing import overload
import cv2
import numpy as np
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
import os
import HandTrackingModule as htm
from playsound import playsound
import colorama
from colorama import Fore, Back, Style
colorama.init(autoreset=True)

RED = '\033[31m'
GREEN = '\033[32m'
#Size
brush = 15
eraser = 60
#Color
```



```
drawColor = (255, 0, 255) #Pink

#Background Folder
FolderPath = "Pic"
myList = os.listdir(folderPath)
#print(myList)
count = 0
mode = 0
overlayList = []
#Save image in folderPath to overlayList
for imPath in myList:
    image = cv2.imread(f'{FolderPath}/{imPath}')
    overlayList.append(image)
#Setting background
header = overlayList[0]

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
cap.set(3, 1280)
cap.set(4, 720)

detector = htm.handDetector(detectionCon = 0.85)
#Previous position
x0, y0 = 0, 0

imgCanvas = np.zeros((720, 1280, 3), np.uint8)

while True:
    #Show camera
    success, img = cap.read()
    #Flip camera
    img = cv2.flip(img, 1)
    #Draw landmarks
    img = detector.findHands(img)
    lmList = detector.findPos(img, draw=False)

    imgGray = cv2.cvtColor(imgCanvas, cv2.COLOR_BGR2GRAY)
    _, imgInv = cv2.threshold(imgGray, 50, 255, cv2.THRESH_BINARY_INV)
    imgInv = cv2.cvtColor(imgInv, cv2.COLOR_GRAY2BGR)

    img = cv2.bitwise_and(img, imgInv)
    img = cv2.bitwise_or(img, imgCanvas)

    if len(lmList)!=0:

        #Index finger
        x1, y1 = lmList[8][1:]
        #Middle finger
        x2, y2 = lmList[12][1:]

        finger = detector.fingerUP()
        #Drawing Mode
        if finger[1]==True and finger[2]==False:
            cv2.circle(img, (x1,y1), 15, drawColor, cv2.FILLED)
```



```
#print("Drawing Mode")
if x0 == 0 and y0 == 0:
    x0, y0 = x1, y1
#Draw & Eraser line in img:
if mode == 2:
    cv2.line(img, (x0, y0), (x1, y1), drawColor, eraser)
    cv2.line(imgCanvas, (x0, y0), (x1, y1), drawColor, eraser)
elif mode == 1:
    cv2.line(img, (x0, y0), (x1, y1), drawColor, brush)
    cv2.line(imgCanvas, (x0, y0), (x1, y1), drawColor, brush)

x0, y0 = x1, y1
#Selecting Mode
if finger[1] and finger[2]:
    #print("Selection Mode")
    x0, y0 = 0, 0
    if 0 < y1 < 115:
        if 237 < x1 < 352:
            header = overlayList[1]
            drawColor = (255, 0, 255)
            mode = 1

        elif 467 < x1 < 582:
            header = overlayList[2]
            drawColor = (0, 0, 0)
            mode = 2

        elif 697 < x1 < 812:
            header = overlayList[3]
            imgCanvas = cv2.bitwise_xor(imgCanvas, imgCanvas)

        elif 927 < x1 < 1042:
            header = overlayList[0]
            pw_r = 'Image/setpw_record.png'
            img_r = 'Image/set_record.png'
            test_r = 'record.png'
            cv2.imwrite(test_r, imgInv)
            break
            #Export passwork screen to file png

            cv2.rectangle(img, (x1, y1 - 25), (x2, y2 + 25), drawColor, cv2.FILLED)
#Setting header image
img[0:115, 0:1280] = header
#img = cv2.addWeighted(img, 0.5, imgCanvas, 0.5, 0)
cv2.imshow("Console", img)
key = cv2.waitKey(1)
if key == ord('s'):
    break
cv2.destroyAllWindows()

Path = ""
myImage = os.listdir()
core = "record.png"
```



```
if core in myImage:
    imagepw = cv2.imread('record.png', cv2.IMREAD_UNCHANGED)
else:
    imagepw = cv2.imread('empty_pw/blank.png', cv2.IMREAD_UNCHANGED)

custom_config = r'-c tessedit_char_whitelist=ABCDEFGHIJKLMN --psm 7 --oem 3'
imagepw_text = pytesseract.image_to_string(imagepw, lang='eng', config=custom_config)
spw = imagepw_text[:4]
#convert string to list
char = "ABCDEFGHIJKLMN"
chars = list(char)
chars_ps = list(spw)
# check if all ele in spw are in char
check = all(item in chars for item in chars_ps)

if check is True:
    with open("PassLog/setpw.txt", 'w', encoding='utf-8') as f:
        f.write(spw)
    cv2.imwrite('Image/setpw_record.png', imgInv)
    cv2.imwrite('Image/set_record.png', img)
    print(GREEN + ("Create password successfully. The password is: " + spw))
    playsound(f'Audio/voice.mp3')
    os.remove('record.png')
else:
    print(RED + "Can not recognize the password, please try again!")
    playsound(f'Audio/voice_fail.mp3')
```

- Chức năng: Đây là file để cài đặt mật khẩu thông qua camera:

- Nếu nhận diện được mật khẩu thì sẽ thông báo voice "**Created new password successfully, the door is locked**", lưu lại mật khẩu và hình ảnh người đặt mật khẩu.



INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Create password successfully. The password is: 2A3E

Hình 7: Cài đặt mật khẩu thành công. Chụp lại người cài đặt mật khẩu, mật khẩu bây giờ là **2A3E**

- Nếu không nhận diện được thì xuất ra voice "**Can not recognize the password, please try again!**" và thông báo lên màn hình.

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Can not recognize the password, please try again!

Hình 8: Xuất voice và hiện thị cài đặt mật khẩu thất bại

7.5 Final.py

```
from typing import overload
import cv2
import numpy as np
import os
import HandTrackingModule as htm
from playsound import playsound
import pytesseract
from datetime import datetime
import colorama
from colorama import Fore, Back, Style
```



```
colorama.init(autoreset=True)

RED = '\033[31m'
GREEN = '\033[32m'
#Size
brush = 15
eraser = 60
#Color
drawColor = (255, 0, 255) #Pink

#Background Folder
folderPath = "Pic"
myList = os.listdir(folderPath)
#print(myList)
count = 0
mode = 0
overlayList = []
#Save image in folderPath to overlayList
for imPath in myList:
    image = cv2.imread(f'{FolderPath}/{imPath}')
    overlayList.append(image)
#print(len(overlayList))

#Setting background
header = overlayList[0]

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
#Width = 1280
cap.set(3, 1280)
#Height = 720
cap.set(4, 720)

detector = htm.handDetector(detectionCon = 0.85)
#Previous position
x0, y0 = 0, 0

imgCanvas = np.zeros((720, 1280, 3), np.uint8)

while True:
    #Show camera
    success, img = cap.read()
    #Flip camera
    img = cv2.flip(img, 1)
    cv2.imwrite('Fail/human.png', img)
    #Draw landmarks
    img = detector.findHands(img)
    lmList = detector.findPos(img, draw=False)

    imgGray = cv2.cvtColor(imgCanvas, cv2.COLOR_BGR2GRAY)
    _, imgInv = cv2.threshold(imgGray, 50, 255, cv2.THRESH_BINARY_INV)
    imgInv = cv2.cvtColor(imgInv, cv2.COLOR_GRAY2BGR)

    img = cv2.bitwise_and(img, imgInv)
```



```
img = cv2.bitwise_or(img, imgCanvas)

if len(lmList)!=0:
    #print(lmList)

    #Index finger
    x1, y1 = lmList[8][1:]
    #Middle finger
    x2, y2 = lmList[12][1:]

    #Checking finger (Up/Down)
    finger = detector.fingerUP()
    #print(finger)

    #Drawing Mode
    if finger[1]==True and finger[2]==False:
        cv2.circle(img, (x1,y1), 15, drawColor, cv2.FILLED)
        #print("Drawing Mode")
        if x0 == 0 and y0 == 0:
            x0, y0 = x1, y1
        #Draw & Eraser line in img:
        if mode == 2:
            cv2.line(img, (x0, y0), (x1, y1), drawColor, eraser)
            cv2.line(imgCanvas, (x0, y0), (x1, y1), drawColor, eraser)
        elif mode == 1:
            cv2.line(img, (x0, y0), (x1, y1), drawColor, brush)
            cv2.line(imgCanvas, (x0, y0), (x1, y1), drawColor, brush)

    x0, y0 = x1, y1

    #Selecting Mode
    if finger[1] and finger[2]:
        #print("Selection Mode")
        x0, y0 = 0, 0
        if 0 < y1 < 115:
            if 237 < x1 < 352:
                header = overlayList[1]
                drawColor = (255, 0, 255)
                mode = 1

            elif 467 < x1 < 582:
                header = overlayList[2]
                drawColor = (0, 0, 0)
                mode = 2

            elif 697 < x1 < 812:
                header = overlayList[3]
                imgCanvas = cv2.bitwise_xor(imgCanvas, imgCanvas)

            elif 927 < x1 < 1042:
                header = overlayList[0]
                pw_r = 'Image/password_record.png'
                img_r = 'Fail/fail_record.png'


```



```
cv2.imwrite(pw_r, imgInv)
break
#Export passwork screen to file png

cv2.rectangle(img, (x1, y1 - 25), (x2, y2 + 25), drawColor, cv2.FILLED)

#Setting header image
img[0:115, 0:1280] = header
#img = cv2.addWeighted(img, 0.5, imgCanvas, 0.5, 0)
cv2.imshow("Console",img)
key = cv2.waitKey(1)
if key == ord('s'):
    break
cv2.destroyAllWindows()
#print(image_text)
from read_img import image_text
with open("PassLog/setpw.txt", 'r', encoding='utf-8') as f:
    setpw = f.read(4)
with open("PassLog/enterpw.txt", 'w', encoding='utf-8') as f:
    f.write(image_text[:4])
with open("PassLog/enterpw.txt", 'r', encoding='utf-8') as f:
    enterpw = f.read(4)

#print(enterpw)

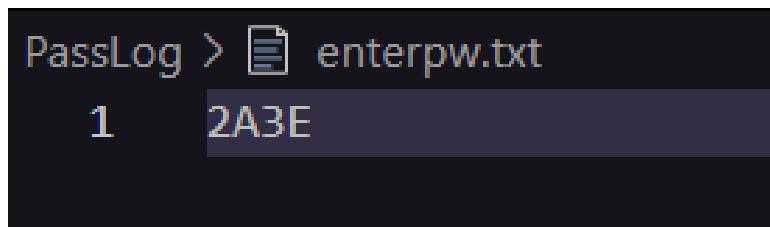
# Print the time and the access history of the door
now = datetime.now()
dt = now.strftime("%d/%m/%Y %H:%M:%S")
from Bot_Telegram import bot_message, bot_photo
bot_message()
if setpw == enterpw:
    print(GREEN + "Welcome home, sir! (^.^)")
    playsound(f'Audio/success.mp3')
    with open("PassLog/Log.txt", 'w+', encoding='utf-8') as f:
        #f.seek(0,0)
        #f.writelines(enterpw + '\n')
        f.writelines(dt + " - The door is opened!" + '\n' + '\n')
    os.remove('Image/password_record.png')
else:
    print(RED + "Wrong password, please try again!")
    playsound(f'Audio/fail.mp3')
    cv2.imwrite('Fail/fail_record.png', img)
    with open("PassLog/Log.txt", 'w+', encoding='utf-8') as f:
        #f.write(enterpw + '\n')
        f.writelines(dt + " - Access denied!" + '\n' + '\n')
    bot_photo()
keys = cv2.waitKey(1)
```

- **Chức năng:** Đây là file để hiện lên màn hình, mật khẩu để đổi chiếu với mật khẩu đã được cài đặt sẵn:

- Nếu đúng thì cửa sẽ mở, xuất ra voice "Welcome home, sir" mở cửa thành công và lưu lại



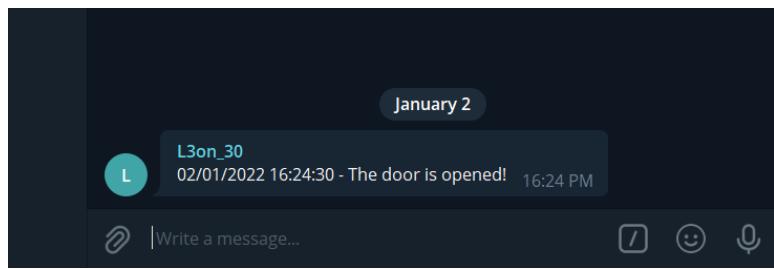
thời gian và trạng thái mở cửa thành công hay thất bại trong "Log.txt"



Hình 9: Nhận diện chính được chuỗi trên ảnh

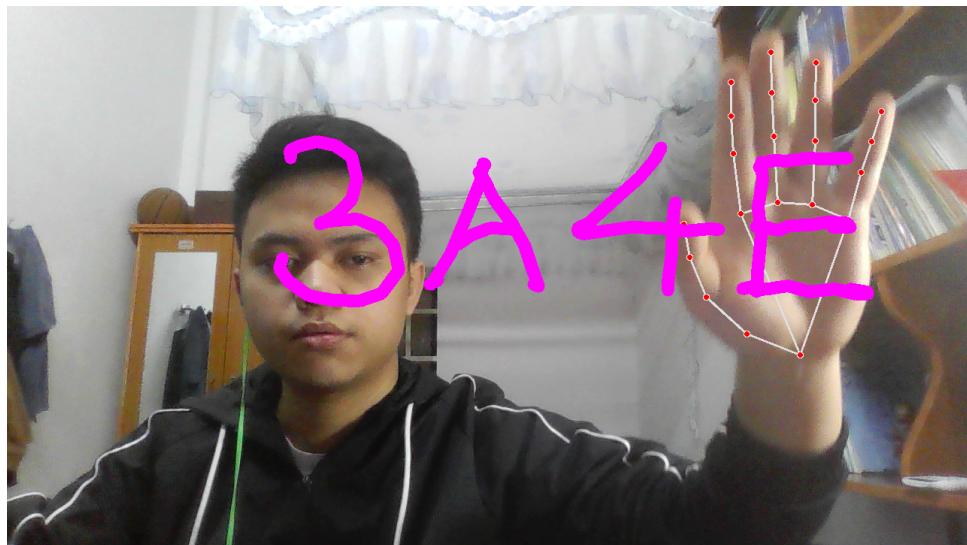
```
INFO: Created TensorFlow Lite XNNPACK delegate
Welcome home, sir! (^.^)
```

Hình 10: Mật khẩu là chính xác. Mở cửa thành công



Hình 11: Bot gửi thời gian và trạng thái mở cửa lên group Telegram

- Nếu sai thì cửa sẽ không mở, xuất ra voice "Wrong password, please try again" và lưu lại hình ảnh người vẽ sai trong "Fail".



```
PassLog > enterpw.txt
1      3A4E|
```

Hình 12: Nhận diện đúng được chuỗi trên ảnh nhưng không phải là mật khẩu đúng

```
INFO: Created TensorFlow Lite XNNPACK delegate
Wrong password, please try again!
```



Hình 13: Mật khẩu là không chính xác, người vẽ sai bị chụp lại và được Bot gửi lên Telegram

II NHẬN XÉT

Trong quá trình thực hiện đề tài, nhóm đã thực sự gặp nhiều khó khăn do ảnh hưởng của đại dịch COVID-19 khiến cho 3 thành viên phải làm việc online nên việc lập trình hệ thống gặp nhiều bất lợi. Tuy vậy, nhóm cuối cùng cũng đã hoàn thành nhiệm vụ.

Nhóm có đăng video clip và cấu trúc lập trình về cách hoạt động của đề tài trên đường link:

Github: <https://github.com/L3on30/UnlockDoorByHandGesture.git>

- Ưu điểm:
 - Tiện lợi, linh hoạt.
 - Tăng khả năng bảo mật, bảo vệ nhà cửa.



- Phù hợp với gia đình đông người ở hoặc văn phòng công ty.
- Sang trọng, hiện đại
- Nhược điểm:
 - Độ chính xác chỉ khoảng 80 đến 90%
 - Chi phí đầu tư và lắp đặt chúng khá tốn kém nên không được sử dụng rộng rãi diện rộng như khóa truyền thống.

Tóm lại, hệ thống đã đạt được những thành công nhất định với sự tự động sẽ làm thay đổi và mở ra kỷ nguyên mới cho công nghệ điều khiển từ xa (từ cơ bản đến phức tạp).

Qua thời gian nghiên cứu thực hiện, chúng em đã phần nào nắm được căn bản nguyên tắc làm việc của Machine learning. Đặc biệt đồ án đã nghiên cứu và giới thiệu được cách sử dụng để chứng minh cho thấy đây là một hệ thống rất hay, cần được tiếp tục nghiên cứu nhiều hơn nữa.

Tuy có rất nhiều ưu điểm nhưng hệ thống này vẫn chưa được chuẩn hóa một cách chính xác cũng như thiếu cơ sở hạ tầng để có thể triển khai rộng rãi. Vì vậy cần thêm nhiều thời gian hơn để nghiên cứu và đầu tư để hệ thống này được đạt được độ chính xác cao hơn.



III TÀI LIỆU THAM KHẢO

- <https://codelearn.io>
- <https://youtu.be/WgtTRwYwrJI>
- <https://stackoverflow.com/questions/44619077/pytesseract-ocr-multiple-config-options>
- <https://nanonets.com/blog/ocr-with-tesseract/>
- <https://core.telegram.org/>