Virtual Mouse with Computer Vision

Tomson George (C0857730)

Praveen Mahaulpatha (C0860583)

R. B. C. M. W. Thulana Vimukthi Abeywardana (C0861333)

Artificial Intelligence and Machine Learning

AML 1214 2 - Python Programming

Simrandeep Kaur

August 20, 2022

Content

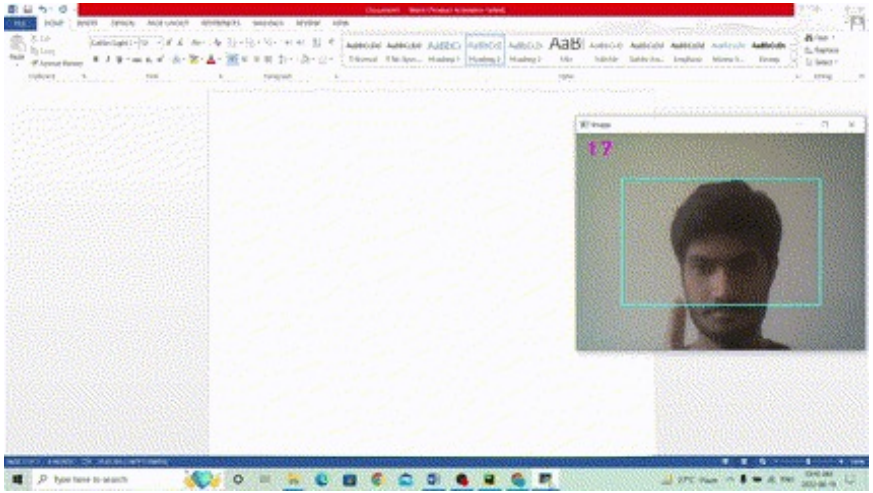
- Introduction
  - Problem Specification
  - Solution Understanding
  - Use Cases
  - SDLC
- Project Scope
  - Module Architecture
  - Project Architecture
  - Technical Specifications
  - Modules
  - Project Challenges
- Conclusion
  - Future Developments

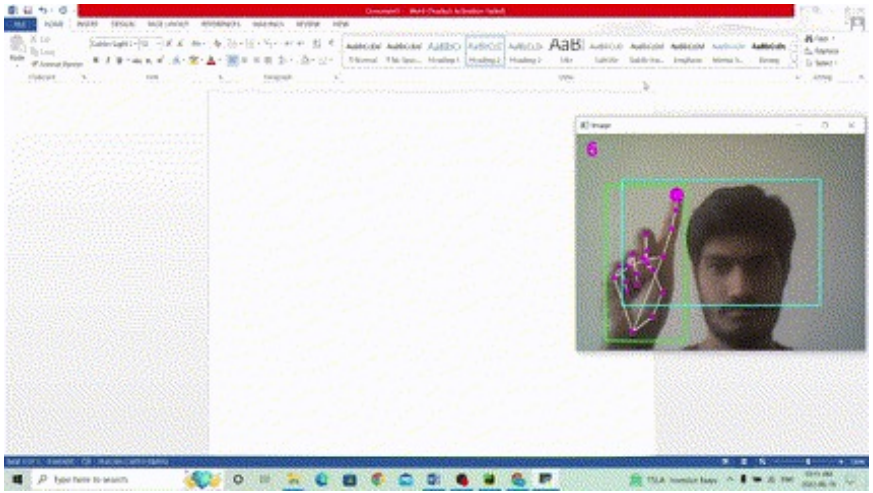
Introduction

Problem Specification

1. Restricted movements
2. Required surface
3. Prone to Injuries
4. Prone to Hardware issues
5. Occupy extra space

Solution Understanding

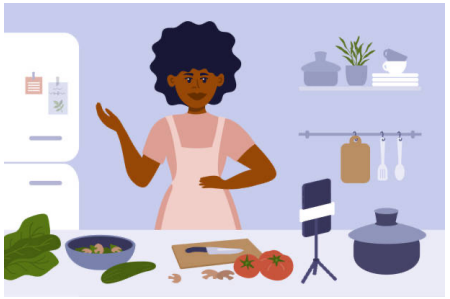




Use Cases

When you are Distant

When you want natural and smoother movements



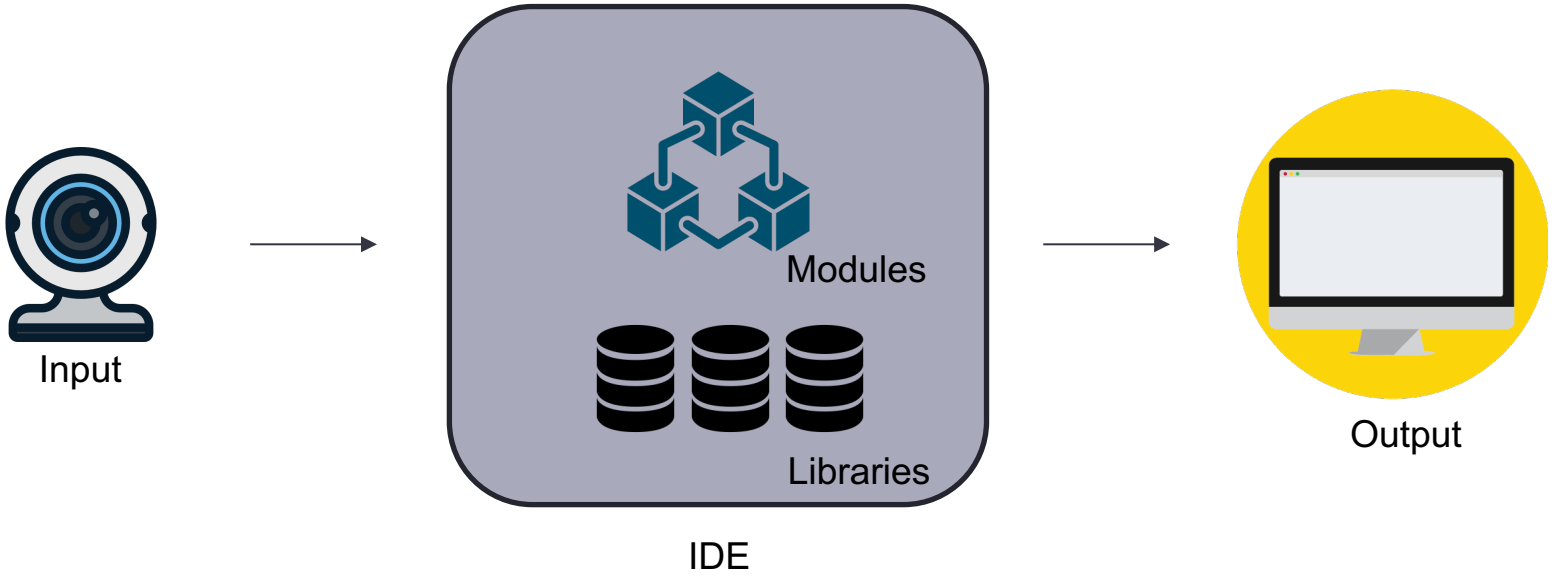
When your hands are busy

SDLC

- Iterative approach
- One feature/module is implemented in one cycle
- Feedback from the previous cycle is take into consideration for next cycle

Project Scope

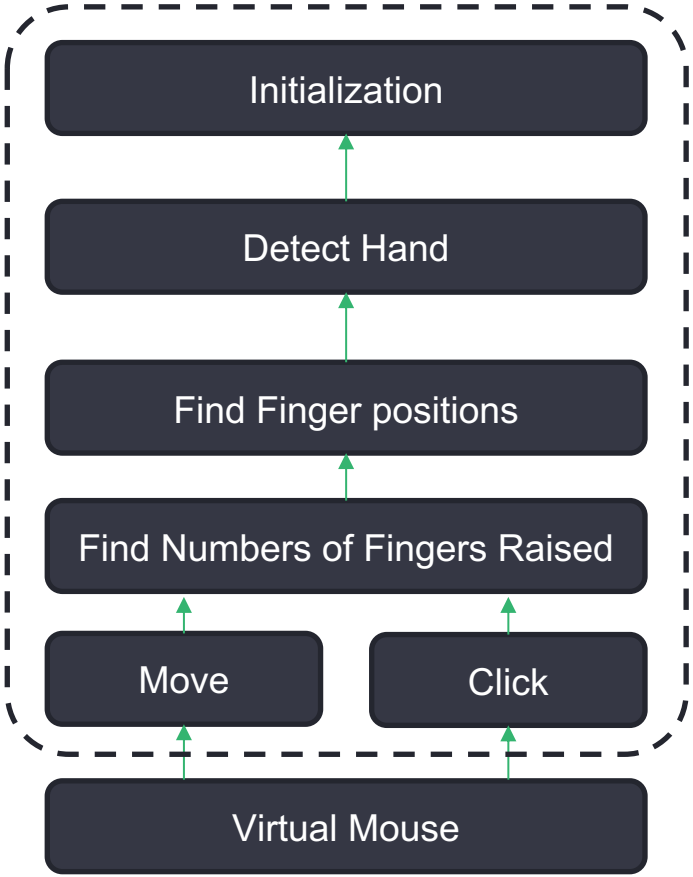
Project Architecture



Technical Specification

Libraries	IDE	Programming Method	Python Version	Other Tools
cv2	Pycharm 	OOP	3.8.9 	Clickup – Project Management
mediapipe				
time				
math				
numpy				
pyautogui				

Module Architecture



Module 1: Initialization

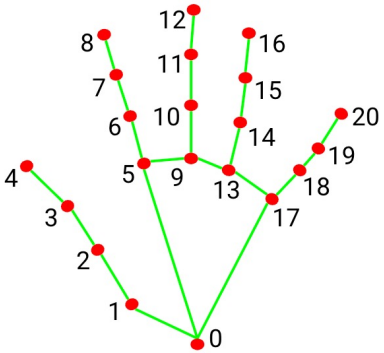
- Passing Default Parameters
  - Static image mode
  - Maximum number of hands
  - Model complexity
  - Minimum detection confidence
  - Minimum tracking confidence

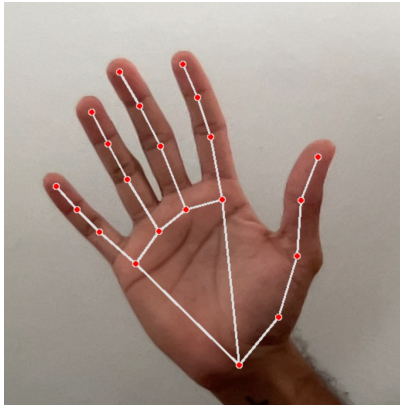
```
8     def __init__(self,static_image_mode=False,max_num_hands=2, modelComplexity=1,min_detection_confidence=0.5,
9                  min_tracking_confidence=0.5):
10
11        self.mode = static_image_mode
12        self.maxHands = max_num_hands
13        self.modelComplexity = modelComplexity
14        self.detectionCon = min_detection_confidence
15        self.trackingCon = min_tracking_confidence
16
17        self.mpHands = mp.solutions.hands  # Formality in using Mediapipe
18        self.hands = self.mpHands.Hands(self.mode,self.maxHands,self.modelComplexity,self.detectionCon,self.trackingCon)
19
20        self.mpDraw = mp.solutions.drawing_utils
21        self.tipIds = [4, 8, 12, 16, 20]
```

- Creating a Hands instance with the defined Parameters

Module 2: Hand Detection

- Input Parameters
  - Webcam image
  - Draw = True
- Convert image to RGB
- Drawing connections between hand points
- Return Parameters
  - Connection drawn image





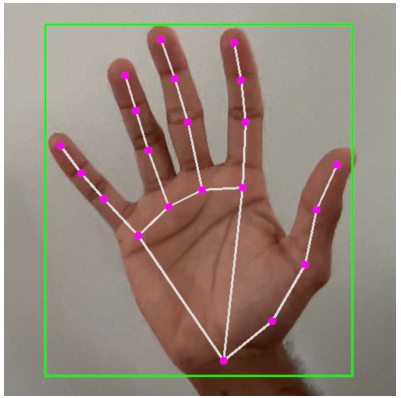
```python
def findHands(self,img,draw=True):

    imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imgRGB)
    # results = hands.process(img)
    # print(results.multi_hand_landmarks)

    if self.results.multi_hand_landmarks:
        for hand in self.results.multi_hand_landmarks:
            if draw:
                self.mpDraw.draw_landmarks(img,hand,self.mpHands.HAND_CONNECTIONS)
    return img
```

Module 3: Find Positions

- Input Parameters
  - Image
  - Hand Number = 0
  - Draw = True
- Draw circles on hand points
- Return Parameters
  - Land marks list with ID and Coordinates

```python
def findPosition(self, img, handNum=0, draw=True):

    self.lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNum]
        for id, lm in enumerate(myHand.landmark):
            # print(id, lm)
            h, w, c = img.shape  # c is number of Channels - RGB (03)
            cx, cy = int(lm.x * w), int(lm.y * h)  # As x & y are ratios of the image
            print(id, cx, cy)
            self.lmList.append([id, cx, cy])
            if draw:
                cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

    return self.lmList
```

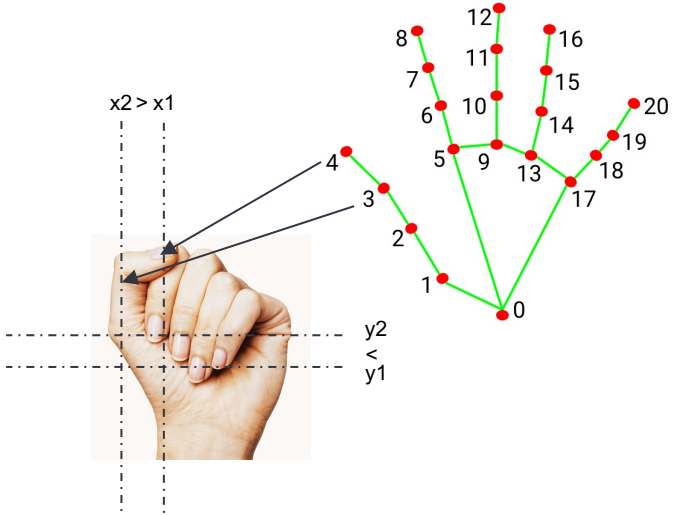
```
[[0, 948, 733], [1, 873, 708], [2, 803, 660], [3, 745, 625], [4, 693, 610], [
[[0, 951, 740], [1, 872, 713], [2, 804, 664], [3, 744, 629], [4, 691, 611], [
```



Module 4: Count Raised Fingers

- No Input Parameters
- Check the X coordinates of Thumb
- Check the Y coordinates for other 4 points
- Return Parameters
  - List of raised/not raised finger tips

```
[0, 1, 0, 0, 0]
[0, 1, 1, 0, 0]
```




```python
def fingersUp(self):
    fingers = []
    # Thumb
    if len(self.lmList) != 0 and (self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]):
        fingers.append(1)
    else:
        fingers.append(0)

    # Fingers
    for id in range(1, 5):

        if len(self.lmList) != 0 and (self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]):
            fingers.append(1)
        else:
            fingers.append(0)
    # print(fingers)
    return fingers
```

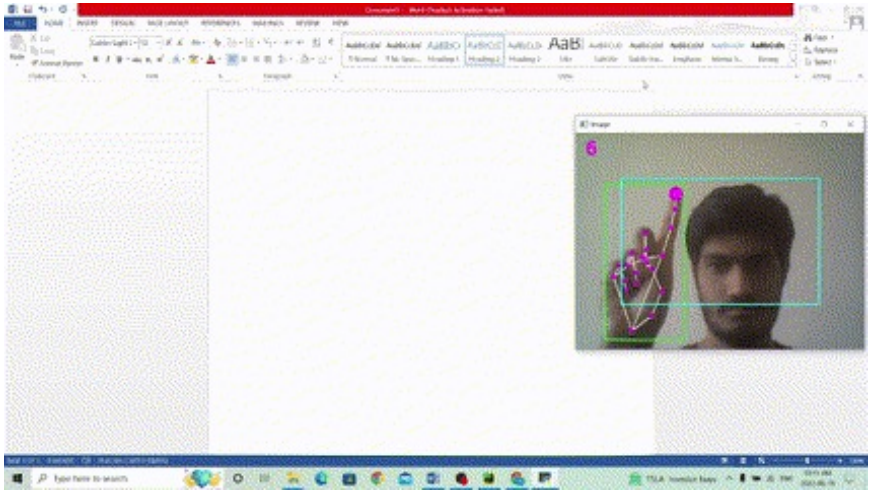
Module 5: Move

- Drawing bounding box
- Convert Coordinates to Ration of Screen
- Smoothen the Values of Coordinates
- Move mouse using “pyautogui” module’s inbuilt functions

```
    # 1. Find the hand Landmarks
    success,img = cap.read()
    img = detector.findHands(img)
    lmList = detector.findPosition(img)

    # 2. Get the tip of the index and middle fingers
    if len(lmList)!=0:
        x1,y1 = lmList[8][1:]
        x2,y2 = lmList[12][1:]
        # print(x1,y1,x2,y2)

    # 3. Check which fingers are up
    fingers = detector.fingersUp()
    # print(fingers)

    cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR), (255, 255, 0), 2)
    # 4. Only Index finger -> Moving Mode
    if fingers[1] == 1 and fingers[2] ==0:
        # 5. Convert Coordinates to ratio of Screen
        x3 = np.interp(x1,(frameR,wCam-frameR),(0,wScr))
        y3 = np.interp(y1, (frameR, hCam-frameR), (0, hScr))

        # 6. Smoothen Values
        clocX = pLocX+(x3-pLocX)/smooth
        clocY = pLoxY+(y3-pLoxY)/smooth

        # 7. Move mouse
        pyautogui.moveTo(wScr-clocX,clocY)
        cv2.circle(img,(x1,y1),15,(255,0,255),thickness=-1)
        pLocX,pLoxY = clocX,clocY
```

Module 6: Click

- Check whether both Index and Middle fingers are raised
- Check whether the distance between them
- If it is less than selected threshold Perform click



```
61      # 8. Both Index and Middle fingers -> Clicking Mode
62      if fingers[1] == 1 and fingers[2] == 1:
63          length, img, lineInfo = detector.findDistance(8, 12, img)
64          if length < 40:
65              pyautogui.click()
```

Project Challenges

- Smoothening Movements of cursor
- Clicking on relatively smaller icons
- Module will not detect only the finger (If the rest of the hand is out of the screen)

Conclusion

Future Developments

- Improved functionality
  - Double Click
  - Move through windows
  - Minimize
- Portable executable

Any Questions?