Volume-control-using-hand-gesture-using-python-and-openCv

Story

INTRODUCTION

Gesture recognition helps computers to understand human body language. This helps to build a more potent link between humans and machines, rather than just the basic text user interfaces or graphical user interfaces (GUIs). In this project for gesture recognition, the human body's motions are read by computer camera. Thecomputer then makes use of this data as input to handle applications. The objective of this project is to develop an interface which will capture human hand gesture dynamically and will control the volume level.

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pycaw : Python Audio Control Library

Mediapipe is an open-source machine learning library of Google, which has some solutions for face recognition and gesture recognition, and provides encapsulation of python, js and other languages. MediaPipe Hands is a high-fidelity hand and finger tracking solution. It uses machine learning (ML) to infer 21 key 3D hand information from just one frame. We can use it to extract the coordinates of the key points of the hand.

WORKING PRINCIPLE

The camera in our device is used for this project. It detects our hand with points in it so as it can see the distance between our thumb finger tip and index finger tip. The distance between the points 4 and 8 is directly proportional to the volume of device.

METHODOLOGY/APPROACH

- Detect hand landmarks
- Calculate the distance between thumb tip and index finger tip.
- Map the distance of thumb tip and index finger tip with volume range. For my case, distance between thumb tip and index finger tip was within the range of 30 – 350 and the volume range was from -63.5 – 0.0.
- In order to exit press ‘Spacebar'

ADVANTAGES:

- Easy to use
- Hassle free
- Fun to use
- More interactive

DISADVANTAGES:

- Cant be used for long distance
- Sometimes not accurate
- Requires a decent camera
- May be confused by two palms

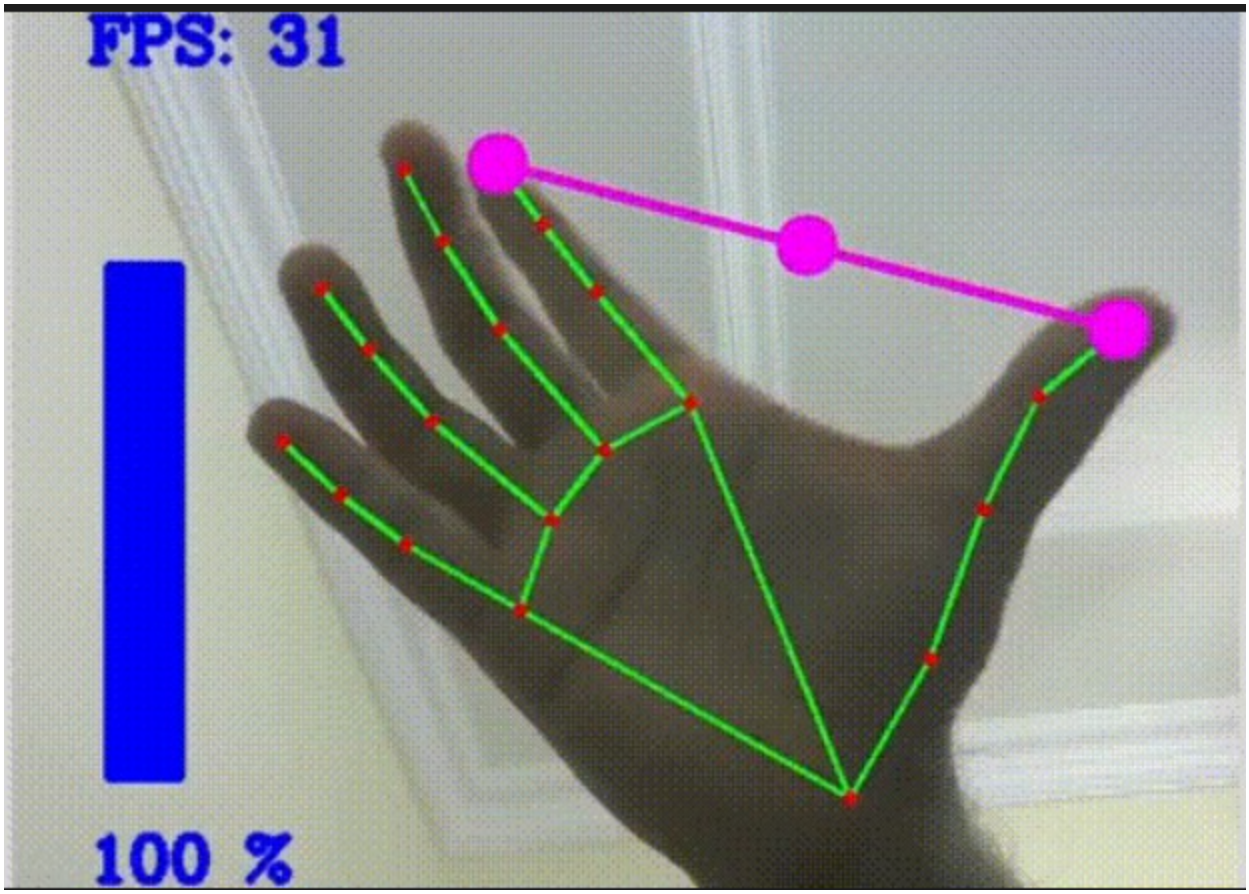
Code

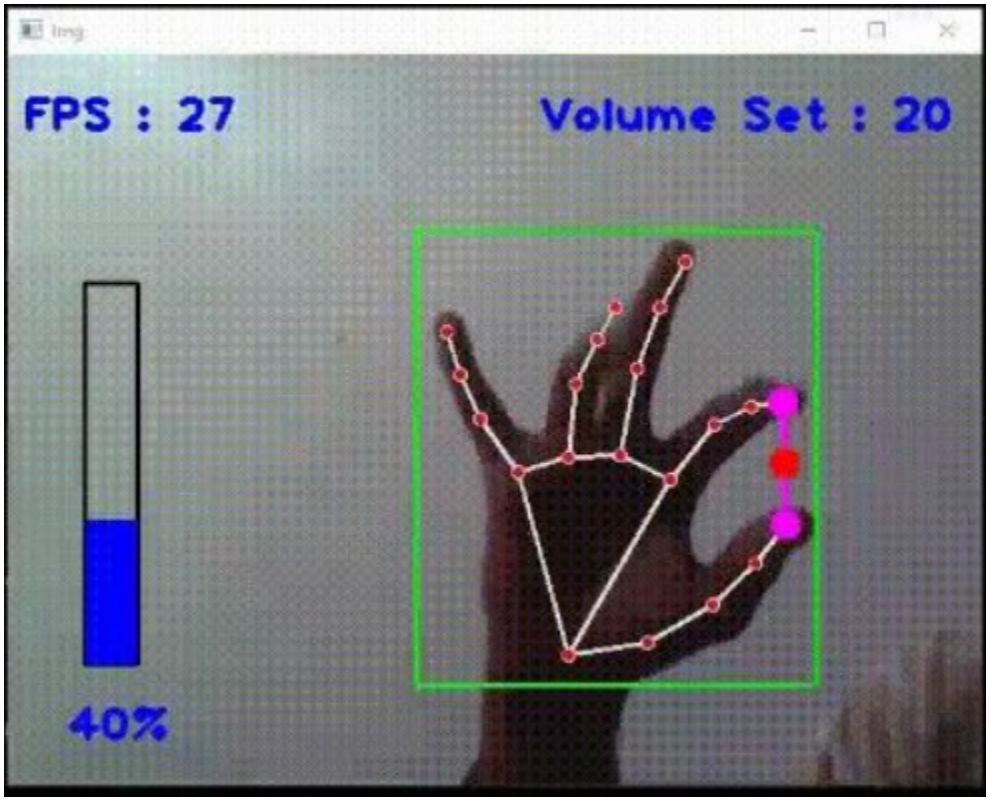
```
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np

cap = cv2.VideoCapture(0) #Checks for camera
```

```
mpHands = mp.solutions.hands #detects hand/finger
hands = mpHands.Hands()   #complete the initialization configuration of hands
mpDraw = mp.solutions.drawing_utils

#To access speaker through the library pycaw
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volbar=400
volper=0

volMin,volMax = volume.GetVolumeRange()[:2]

while True:
    success,img = cap.read() #If camera works capture an image
    imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #Convert to rgb

    #Collection of gesture information
    results = hands.process(imgRGB) #completes the image processing.

    lmList = [] #empty list
    if results.multi_hand_landmarks: #list of all hands detected.
        #By accessing the list, we can get the information of each hand's corresponding flag bit
        for handlandmark in results.multi_hand_landmarks:
            for id,lm in enumerate(handlandmark.landmark): #adding counter and returning it
                # Get finger joint points
                h,w,_ = img.shape
                cx,cy = int(lm.x*w),int(lm.y*h)
                lmList.append([id,cx,cy]) #adding to the empty list 'lmList'
            mpDraw.draw_landmarks(img,handlandmark,mpHands.HAND_CONNECTIONS)

    if lmList != []:
        #getting the value at a point
                    #x      #y
        x1,y1 = lmList[4][1],lmList[4][2]  #thumb
        x2,y2 = lmList[8][1],lmList[8][2]  #index finger
        #creating circle at the tips of thumb and index finger
        cv2.circle(img,(x1,y1),13,(255,0,0),cv2.FILLED) #image #fingers #radius #rgb
        cv2.circle(img,(x2,y2),13,(255,0,0),cv2.FILLED) #image #fingers #radius #rgb
        cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)  #create a line b/w tips of index finger and thumb

        length = hypot(x2-x1,y2-y1) #distance b/w tips using hypotenuse
 # from numpy we find our length,by converting hand range in terms of volume range ie b/w
-63.5 to 0
```

```
        vol = np.interp(length,[30,350],[volMin,volMax])
        volbar=np.interp(length,[30,350],[400,150])
        volper=np.interp(length,[30,350],[0,100])


        print(vol,int(length))
        volume.SetMasterVolumeLevel(vol, None)

        # Hand range 30 - 350
        # Volume range -63.5 - 0.0
        #creating volume bar for volume level
        cv2.rectangle(img,(50,150),(85,400),(0,0,255),4) # vid ,initial position ,ending position ,rgb
,thickness
        cv2.rectangle(img,(50,int(volbar)),(85,400),(0,0,255),cv2.FILLED)
        cv2.putText(img,f'{int(volper)}%',(10,40),cv2.FONT_ITALIC,1,(0, 255, 98),3)
        #tell the volume percentage ,location,font of text,length,rgb color,thickness
    cv2.imshow('Image',img) #Show the video
    if cv2.waitKey(1) & 0xff==ord(' '): #By using spacebar delay will stop
        break

cap.release()     #stop cam
cv2.destroyAllWindows() #close window
```

RESULTs


FPS: 31
100 %


FPS : 27
Volume Set : 20
40%