DATA SCIENCE WITH PYTHON PROJECT (HAND GESTURE VOLUME CONTROL)

T. SANJIV

ABSTRACT

The Hand Gesture Volume Controller is a Python-based project that aims to create a system that allows users to control the volume of an audio output device using hand gestures captured by a camera. The system utilizes computer vision techniques and machine learning algorithms to recognize specific hand gestures as volume commands, translating them into corresponding volume adjustments. This project offers a novel and intuitive way to interact with audio devices, promoting a touchless and more immersive user experience.

INTRODUCTION

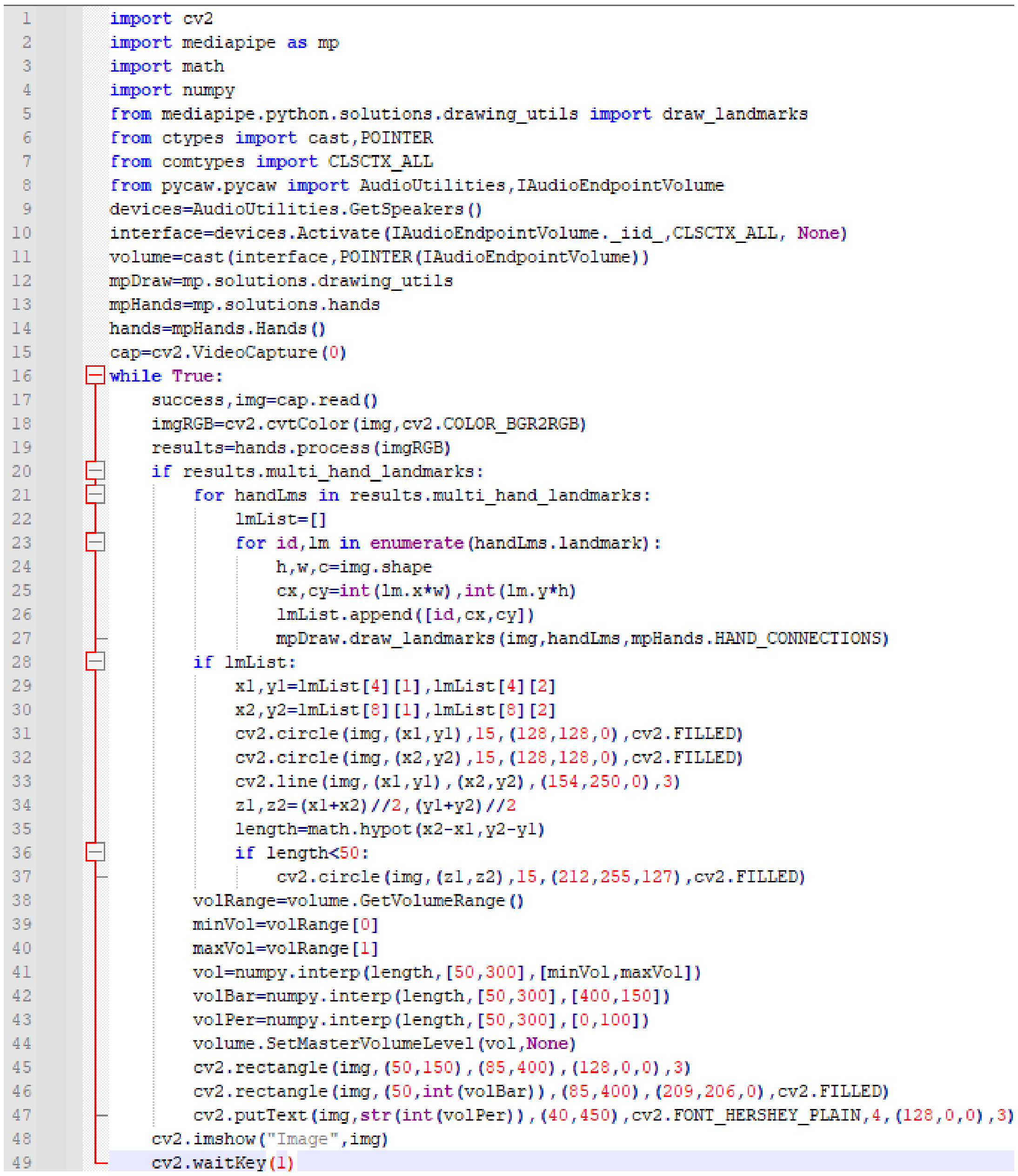
The traditional methods of adjusting volume through buttons or software controls often require users to divert their attention away from their primary tasks, which can be cumbersome and inconvenient in certain situations. By employing hand gestures as a control mechanism, this project seeks to offer users a more intuitive and natural way to manage audio volume. The system will leverage a camera to capture real-time video input, which will then be processed using computer vision algorithms to detect and track hand gestures. A machine learning model will be trained to recognize specific hand configurations associated with volume up, volume down, and mute commands. Once the gestures are recognized, the system will interface with the audio output device to effectuate the appropriate volume adjustments.

METHODOLOGY

The steps involved are:

- ➔ Importing the necessary libraries
- ➔ Detecting and marking positions of the hands
- ➔ Accessing the speaker
- ➔ Finding the range of volume
- ➔ Capturing the image from camera
- ➔ Checking for multiple hands
- ➔ Specifying the points of the fingers
- ➔ Making a circle at the required points
- ➔ Connecting the circles through a line
- ➔ Finding the distance of the line
- ➔ Converting the hand range to volume range
- ➔ Display the video used to interact with user

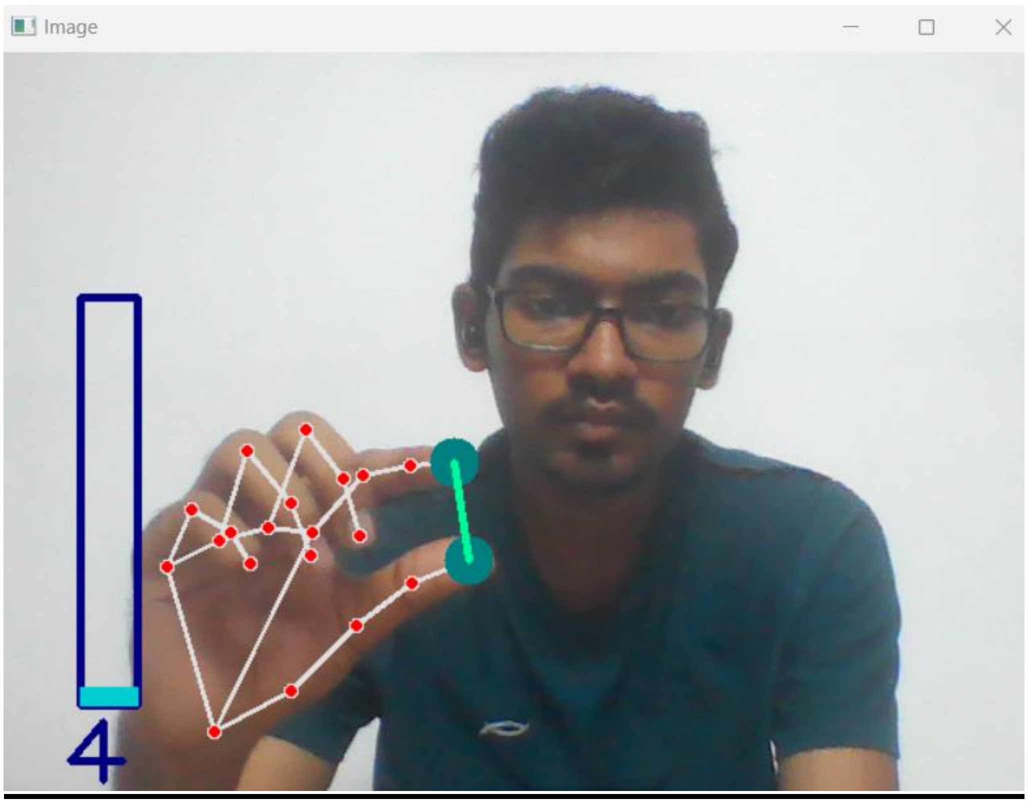
CODE

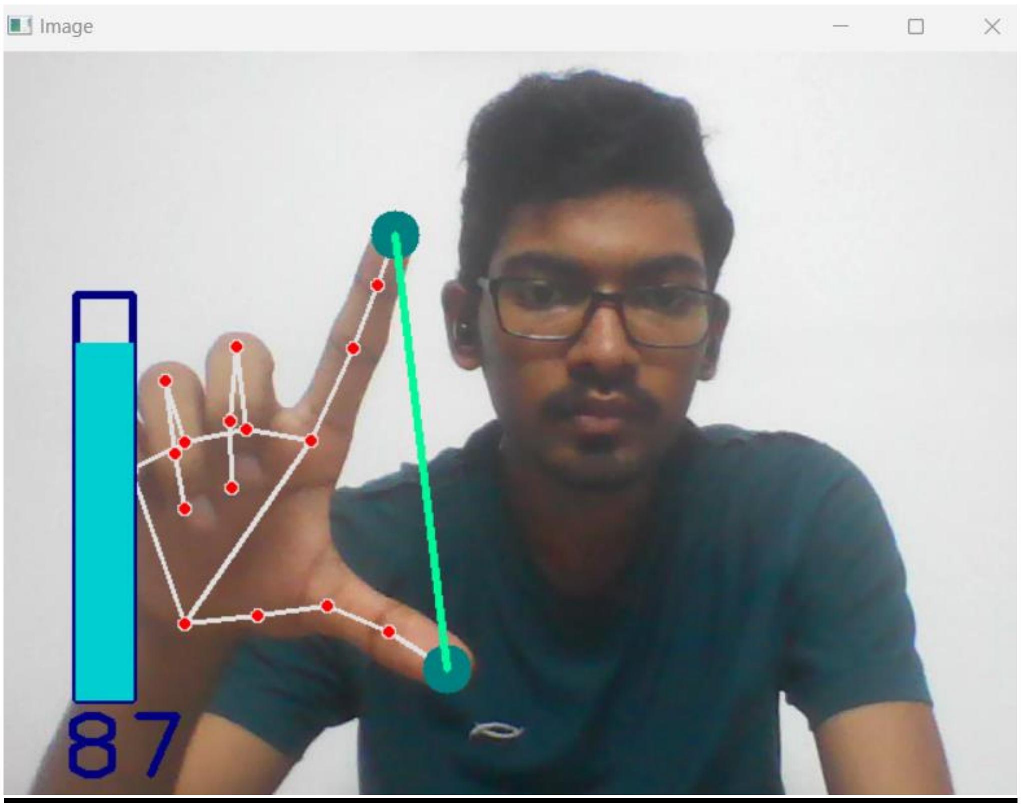


```python
import cv2
import mediapipe as mp
import math
import numpy
from mediapipe.python.solutions.drawing_utils import draw_landmarks
from ctypes import cast,POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities,IAudioEndpointVolume
devices=AudioUtilities.GetSpeakers()
interface=devices.Activate(IAudioEndpointVolume._iid_,CLSCTX_ALL, None)
volume=cast(interface,POINTER(IAudioEndpointVolume))
mpDraw=mp.solutions.drawing_utils
mpHands=mp.solutions.hands
hands=mpHands.Hands()
cap=cv2.VideoCapture(0)
while True:
    success,img=cap.read()
    imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    results=hands.process(imgRGB)
    if results.multi_hand_landmarks:
        for handLms in results.multi_hand_landmarks:
            lmList=[]
            for id,lm in enumerate(handLms.landmark):
                h,w,c=img.shape
                cx,cy=int(lm.x*w),int(lm.y*h)
                lmList.append([id,cx,cy])
                mpDraw.draw_landmarks(img,handLms,mpHands.HAND_CONNECTIONS)
        if lmList:
            x1,y1=lmList[4][1],lmList[4][2]
            x2,y2=lmList[8][1],lmList[8][2]
            cv2.circle(img,(x1,y1),15,(128,128,0),cv2.FILLED)
            cv2.circle(img,(x2,y2),15,(128,128,0),cv2.FILLED)
            cv2.line(img,(x1,y1),(x2,y2),(154,250,0),3)
            z1,z2=(x1+x2)//2,(y1+y2)//2
            length=math.hypot(x2-x1,y2-y1)
            if length<50:
                cv2.circle(img,(z1,z2),15,(212,255,127),cv2.FILLED)
        volRange=volume.GetVolumeRange()
        minVol=volRange[0]
        maxVol=volRange[1]
        vol=numpy.interp(length,[50,300],[minVol,maxVol])
        volBar=numpy.interp(length,[50,300],[400,150])
        volPer=numpy.interp(length,[50,300],[0,100])
        volume.SetMasterVolumeLevel(vol,None)
        cv2.rectangle(img,(50,150),(85,400),(128,0,0),3)
        cv2.rectangle(img,(50,int(volBar)),(85,400),(209,206,0),cv2.FILLED)
        cv2.putText(img,str(int(volPer)),(40,450),cv2.FONT_HERSHEY_PLAIN,4,(128,0,0),3)
    cv2.imshow("Image",img)
    cv2.waitKey(1)
```

```
//Code for implementing
import cv2
import mediapipe as mp
import math
import numpy
from mediapipe.python.solutions.drawing_utils import draw_landmarks
from ctypes import cast,POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities,IAudioEndpointVolume
devices=AudioUtilities.GetSpeakers()
interface=devices.Activate(IAudioEndpointVolume._iid_,CLSCTX_ALL, None)
volume=cast(interface,POINTER(IAudioEndpointVolume))
mpDraw=mp.solutions.drawing_utils
mpHands=mp.solutions.hands
hands=mpHands.Hands()
cap=cv2.VideoCapture(0)
while True:
    success,img=cap.read()
    imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    results=hands.process(imgRGB)
    if results.multi_hand_landmarks:
        for handLms in results.multi_hand_landmarks:
            lmList=[]
            for id,lm in enumerate(handLms.landmark):
                h,w,c=img.shape
                cx,cy=int(lm.x*w),int(lm.y*h)
                lmList.append([id,cx,cy])
                mpDraw.draw_landmarks(img,handLms,mpHands.HAND_CONNECTIONS)
        if lmList:
            x1,y1=lmList[4][1],lmList[4][2]
            x2,y2=lmList[8][1],lmList[8][2]
            cv2.circle(img,(x1,y1),15,(128,128,0),cv2.FILLED)
            cv2.circle(img,(x2,y2),15,(128,128,0),cv2.FILLED)
            cv2.line(img,(x1,y1),(x2,y2),(154,250,0),3)
            z1,z2=(x1+x2)//2,(y1+y2)//2
            length=math.hypot(x2-x1,y2-y1)
            if length<50:
                cv2.circle(img,(z1,z2),15,(212,255,127),cv2.FILLED)
        volRange=volume.GetVolumeRange()
        minVol=volRange[0]
        maxVol=volRange[1]
        vol=numpy.interp(length,[50,300],[minVol,maxVol])
        volBar=numpy.interp(length,[50,300],[400,150])
        volPer=numpy.interp(length,[50,300],[0,100])
        volume.SetMasterVolumeLevel(vol,None)
        cv2.rectangle(img,(50,150),(85,400),(128,0,0),3)
        cv2.rectangle(img,(50,int(volBar)),(85,400),(209,206,0),cv2.FILLED)
        cv2.putText(img,str(int(volPer)),(40,450),cv2.FONT_HERSHEY_PLAIN,4,(128,0,0),3)
    cv2.imshow("Image",img)
    cv2.waitKey(1)
```

RESULT





CONCLUSION

The Hand Gesture Volume Controller project successfully achieved its objective of creating a touchless and intuitive audio control system using Python, computer vision, and machine learning. By leveraging hand gestures as a means of interaction, the project offers users a more natural and immersive experience when managing audio volume. The implementation of the Hand Gesture Volume Controller showcased the potential of gesture-based control systems in enhancing accessibility and user experience. Users can now effortlessly adjust audio volume without needing physical buttons or touch-based inputs, making it particularly valuable in scenarios where touch interaction is inconvenient or not feasible.