Wireless Sound Control

# **Abstract**

Wireless sound control refers to the ability to control the volume of a wireless speaker using hand gestures or other physical movements. In this project, a machine learning (ML) model is trained to detect hand landmarks(which uses supervised machine learning)in video frames captured by a camera. The model is then used to select the hand landmarks corresponding to the index finger and thumb finger, and calculate the distance between them. Based on this distance, the volume of the wireless speaker is adjusted. The user can control the volume by moving their index finger and thumb closer together or further apart, and can exit the program by pressing the "q" key.

## **Introduction**

### Wireless sound control refers to the ability to control the volume of an audio device, such as a speaker, wirelessly using a device such as a smartphone or a computer. In this project, I have implemented a system for controlling the volume of a speaker wirelessly using hand gestures captured by a webcam.

### it involves the creation of a wireless sound control system that uses machine learning and computer vision techniques to allow a user to control the volume of a speaker using hand gestures captured by a webcam. By eliminating the need for physical buttons or controls, this system provides a simple and intuitive way to adjust the volume of an audio device wirelessly.

## **Hardware and Software Requirements:**

### To implement this system, the following hardware and software are required:

### A computer with a webcam

### A speaker connected to the computer or inbuilt speaker

### The Python programming language, with the following libraries installed: cv2, mediapipe, ctypes, comtypes, pycaw, and numpy

# **Overview of the Code:**

### The code for the wireless sound control system consists of the following steps:

### Capture video frames from the webcam using the cv2 library

### Process the video frames to detect and track hand landmarks using the mediapipe library

### Use the ctypes and comtypes libraries to control the volume of the speaker using the IAudioEndpointVolume interface

### Use the numpy library to interpolate the volume level based on the distance between the detected hand landmarks

### Visualize the detected hand landmarks and the volume control feedback using the cv2 library

### **Hand Landmark Detection using MediaPipe:**

### The mediapipe library is used to detect and track hand landmarks in the video frames captured by the webcam. This is done using a machine learning model trained to recognize the positions of key points on the hand, such as the fingertips, knuckles, and wrist. The model returns a set of hand landmarks for each detected hand, including the x and y coordinates of each landmark relative to the frame size.

# **Volume Control using ctypes and comtypes:**

### The ctypes and comtypes libraries are used to interface with the IAudioEndpointVolume interface provided by the operating system. This interface allows the code to retrieve and set the volume level of the speaker. The volume range is represented as a decimal value between -63.5 (minimum volume) and 0.0 (maximum volume).

# **Interpolating the Volume Level using numpy:**

### The numpy library is used to interpolate the volume level based on the distance between the detected hand landmarks. The distance is calculated using the Pythagorean theorem and is used as an input to the numpy.interp function, which maps the distance to the corresponding volume level within the specified range.

# **Visualizing the Results using cv2:**

### The cv2 library is used to visualize the detected hand landmarks and the volume control feedback in the video frames. This includes drawing circles and lines to indicate the positions of the hand landmarks, as well as changing the color of the circles to green when the distance between the landmarks falls below a certain threshold (indicating that the volume should be reduced). The processed frames are then displayed to the user using the cv2.imshow function.

# **Testing the System:**

### To test the system, the user can place their hand in front of the webcam and move their index finger and thumb closer or farther apart to adjust the volume of the speaker. The system should detect the hand landmarks and update the volume level accordingly, as indicated by the color of the circles and the volume level displayed in the console.

# **Limitations :**

### One potential limitation of the system is its reliance on the accuracy of the hand landmark detection model provided by

mediapipe. If the model is unable to accurately detect the hand landmarks, the system will not function as intended. Additionally, the system currently only supports the control of a single speaker, and does not have the ability to select or switch between multiple audio devices.

### **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

cap = cv2.VideoCapture(0)

mpHands = mp.solutions.hands

hands = mpHands.Hands()

mpDraw = mp.solutions.drawing\_utils

devices = AudioUtilities.GetSpeakers()

interface = devices.Activate(IAudioEndpointVolume.\_iid\_, CLSCTX\_ALL, None)

volume = cast(interface, POINTER(IAudioEndpointVolume))

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

while True:

    success,img = cap.read()

    imgRGB = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

    results = hands.process(imgRGB)

    lmList = []

    if results.multi\_hand\_landmarks:

        for handlandmark in results.multi\_hand\_landmarks:

            for id,lm in enumerate(handlandmark.landmark):

                h,w,\_ = img.shape

                cx,cy = int(lm.x\*w),int(lm.y\*h)

                lmList.append([id,cx,cy])

            mpDraw.draw\_landmarks(img,handlandmark,mpHands.HAND\_CONNECTIONS)

    if lmList != []:

        x1,y1 = lmList[4][1],lmList[4][2]

        x2,y2 = lmList[8][1],lmList[8][2]

        cx,cy = (x1+x2)//2,(y1+y2)//2

        cv2.circle(img,(x1,y1),15,(255,0,255 ),cv2.FILLED)

        cv2.circle(img,(x2,y2),15,(255,0,255),cv2.FILLED)

        cv2.line(img,(x1,y1),(x2,y2),(255,0,255),3)

        cv2.circle(img,(cx,cy),15,(255,0,255),cv2.FILLED)

        length = hypot(x2-x1,y2-y1)

        if length<50:

            cv2.circle(img,(cx,cy),15,(0,255,0),cv2.FILLED)

        vol = np.interp(length,[15,220],[volMin,volMax])

        print(vol,length)

        volume.SetMasterVolumeLevel(vol, None)

        # Hand range 15 - 220

        # Volume range -63.5 - 0.0

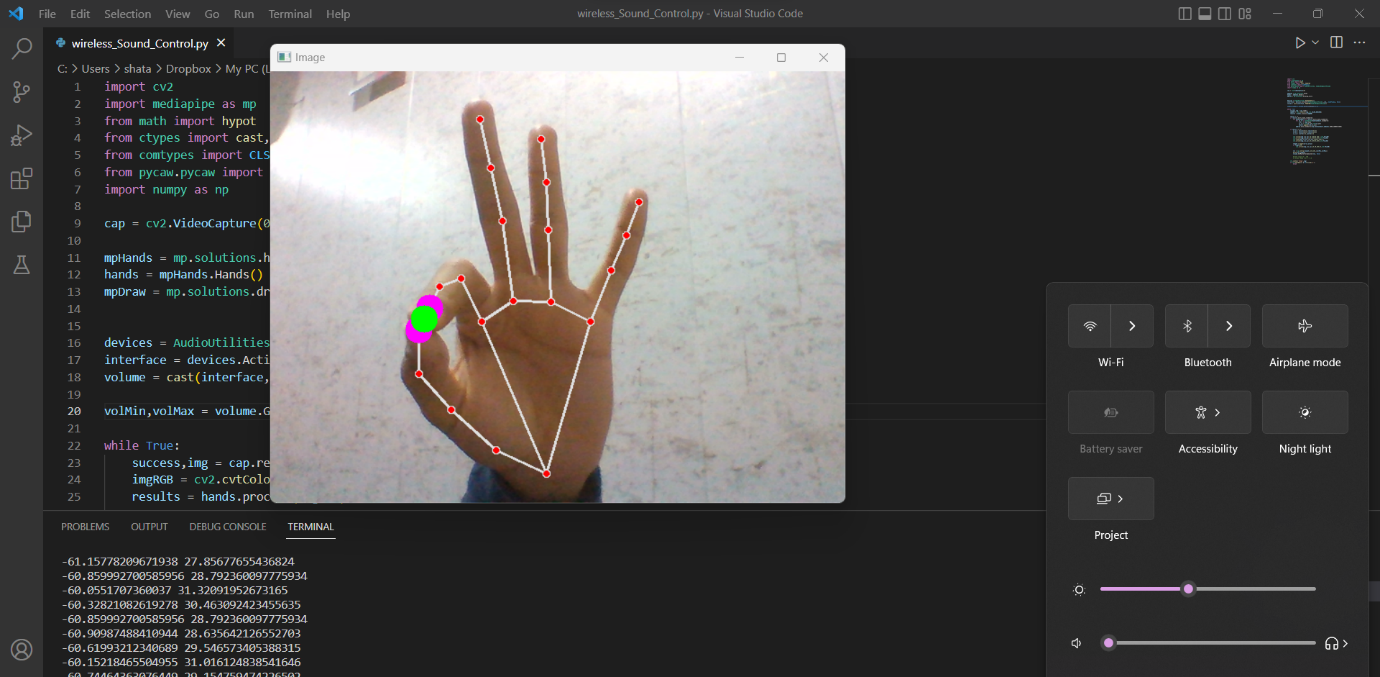
    cv2.imshow('Image',img)

    if cv2.waitKey(1) & 0xff==ord('q'):

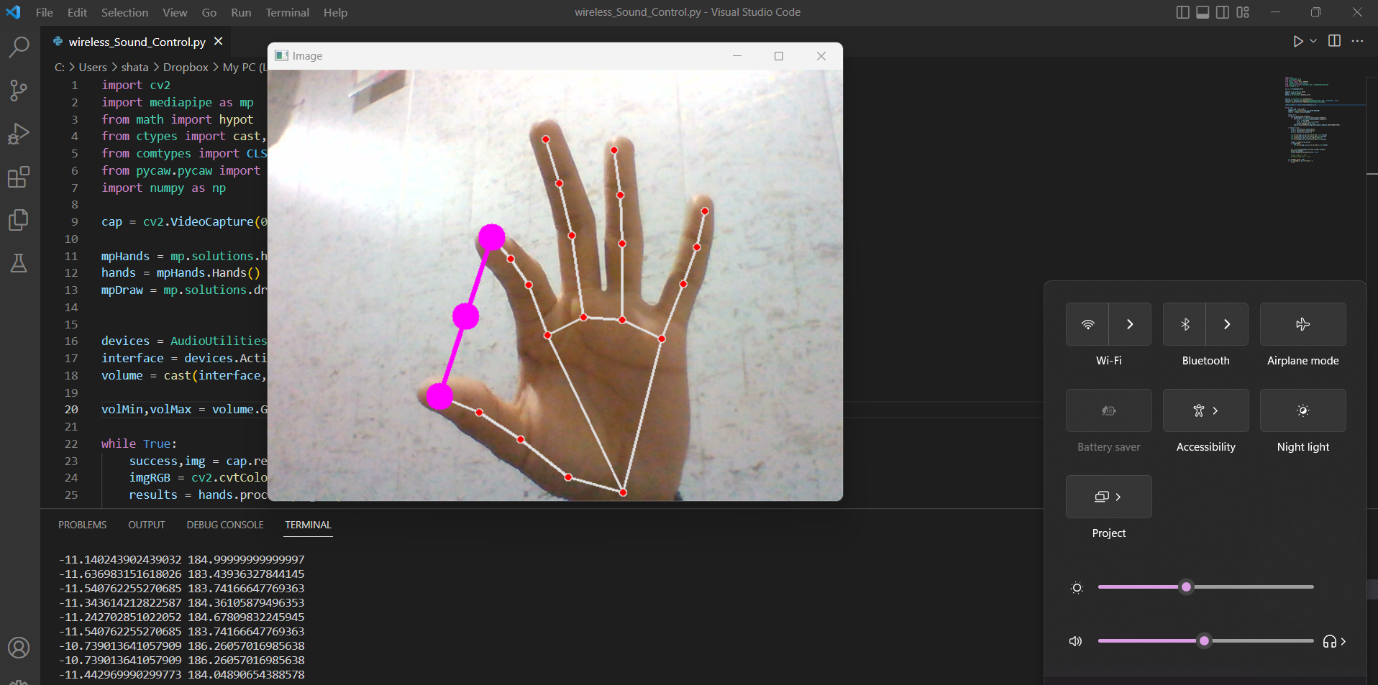
        break

# 100% volume **Screenshot Of The Code**

100% Volume when tip of thumb and index finger are at max. distance.



0% Volume when tip of thumb and index finger are at min. distance.

50% Volume when tip of thumb and index finger are at half the max. distance between them.

# **Conclusion**

In conclusion, the wireless sound control system presented in this project demonstrates the potential for using machine learning and computer vision techniques to create intuitive and convenient control interfaces for audio devices. By eliminating the need for physical buttons or controls, the system offers a simple and seamless way to adjust the volume of a speaker using hand gestures captured by a webcam.

One possible direction for future work is to improve the robustness and accuracy of the hand landmark detection model, perhaps by training the model on a larger and more diverse dataset. Another possibility is to expand the system to support the control of multiple audio devices, either by adding additional hardware or by using software-based audio routing.