**Gesture Volume Control using Python**

**Objective:**

Hand Volume Control is a Python program that utilizes computer vision techniques to track hand movements and control the system's volume accordingly.

**Software Requirements:**

1. Python 3.11.4

2. OpenCV library

3. Mediapipe library

4. PyAutoGUI library

5. Webcam or video source

**Usage:**

1. Install the required libraries using pip:

- OpenCV: pip install opencv-python

- Mediapipe: pip install mediapipe

- PyAutoGUI: pip install pyautogui

2. Connect a webcam or provide a video source.

3. Run the program:

- Open a terminal or command prompt.

- Navigate to the directory where the script is saved.

- Execute the command: python hand\_volume\_control.py

**Description:**

The Hand Volume Control program performs using the following steps:

1. Import necessary libraries:

- cv2: For image processing and displaying images.

- mediapipe: For hand tracking using pre-trained models.

- pyautogui: For controlling the system's volume.

2. Initialize variables and objects:

- x1, y1, x2, y2: Variables to store the coordinates of two fingertips.

- webcam: Represents the webcam or video source.

- my\_hands: Object for hand tracking using the Mediapipe library.

- drawing\_utils: Utility object from Mediapipe for drawing landmarks on the image.

3. Start a while loop to continuously process frames from the webcam:

- Read a frame from the webcam using the "webcam.read()" function.

- Flip the image horizontally using "cv2.flip()" for a mirror-like effect.

- Get the dimensions of the frame using "image.shape".

- Convert the image to RGB format using "cv2.cvtColor()".

- Process the RGB image with Mediapipe's hand tracking model and obtain the hand landmarks.

4. Perform hand tracking and control volume:

- If hand landmarks are detected (hands are not empty):

- Iterate over each hand.

- Draw landmarks on the image using "drawing\_utils. draw\_landmarks()".

- Retrieve the pixel coordinates of the index fingertip and thumb fingertip.

- Draw circles at the fingertip positions using "cv2.circle()".

- Calculate the distance between the two fingertip points using the Euclidean distance formula.

- Draw a line connecting the two fingertips using "cv2.line()".

- If the distance is greater than 30, increase the system's volume using

"Pyautogui.press("volumeup")".

Otherwise, decrease the volume using "pyautogui.press("volumedown")".

5. Display the image in a window:

- Show the image using "cv2.imshow()".

- Wait for a key press and exit the loop if the "Esc" key is pressed.

6. Clean up and exit:

* Release the webcam using "webcam.release()".
* Close all windows using "cv2.destroyAllWindows()".

**Outcome:**

The outcome of gesture volume control has been highly favorable, enhancing user experience, improving accessibility, and simplifying design aesthetics. As this technology continues to evolve, we can anticipate future advancements and widespread adoption, leading to more immersive interactions with audio devices in various contexts.

**Conclusion:**

You now have all the skills required to create a hand gesture volume controller. If you are working while listening to your favourite music, by just a gesture of your hand, you will be able to control the volume level of your music.

**Source Code:**

import cv2 #display an image in a window.  
import mediapipe as mp #we use mediapipe for handtracking  
import pyautogui #controling the system's volume  
#taking variables and objects  
x1 = y1 = x2 = y2 = 0  
webcam = cv2.VideoCapture(0)  
my\_hands = mp.solutions.hands.Hands()  
drawing\_utils = mp.solutions.drawing\_utils  
# Running a while loop to continuously process frames from the webcam  
while True:  
 \_ , image = webcam.read()  
 image = cv2.flip(image,1)  
 frame\_height, frame\_width, \_ = image.shape  
 rgb\_image = cv2.cvtColor(image,cv2.COLOR\_BGR2RGB)  
 output = my\_hands.process(rgb\_image) # Process the image with hand tracking  
 hands = output.multi\_hand\_landmarks  
 if hands:  
 for hand in hands:  
 drawing\_utils.draw\_landmarks(image, hand) # Draw landmarks on the image  
 landmarks = hand.landmark  
 for id,landmark in enumerate(landmarks):  
 # Get the pixel coordinates of the landmark  
 x = int(landmark.x \* frame\_width)  
 y = int(landmark.y \* frame\_height)  
 if id == 8:  
 # Draw a circle at the fingertip position  
 cv2.circle(img=image,center=(x,y),radius=8,color=(0,255,255),thickness=3)  
 x1 = x  
 y1 = y  
 if id == 4:  
 # Draw a circle at the fingertip position  
 cv2.circle(img=image,center=(x,y),radius=8,color=(0,0,255),thickness=3)  
 x2 = x  
 y2 = y  
 # Calculate the distance between the two fingertip points  
 dist = ((x2-x1)\*\*2 + (y2-y1)\*\*2)\*\*(0.5)//4  
 cv2.line(image,(x1,y1),(x2,y2),(0,255,0),5) # Draw a lines between two fingertips  
 if dist > 40:  
 pyautogui.press("volumeup")  
 else:  
 pyautogui.press("volumedown")  
  
 # Display the image in a window titled "Hand volume control using python  
 cv2.imshow("Hand volume control using python", image)  
 key = cv2.waitKey(30)  
 if key == 27:  
 break  
webcam.release()  
cv2.destroyAllWindows()

# 