Aim

To develop a hand gesture-based volume control system that adjusts the system volume using hand movements captured via a webcam.

Objective

- Implement real-time hand tracking using Mediapipe to detect hand landmarks.
- Map the distance between specific hand landmarks to control system volume.
- Use OpenCV for image processing and visualization.
- Integrate PyAutoGUI for simulating volume control key presses based on hand gesture analysis.

Summary

This project utilizes a webcam to capture real-time hand movements and processes the video feed to detect hand landmarks. By measuring the distance between the tip of the thumb and the index finger, the system determines whether to increase or decrease the system volume. The hand gestures are tracked using Mediapipe, and the volume control is managed through PyAutoGUI.

Tools and Libraries Used

- **OpenCV**: For video capture, image processing, and visualization.
- **Mediapipe**: For hand tracking and landmark detection.
- **PyAutoGUI**: For simulating keyboard presses to control system volume.

Procedure (Explain the Code)

1. **Initialize Libraries**: Import necessary libraries (cv2, pyautogui, mediapipe).
2. **Setup Camera and Screen Dimensions**:
    - Initialize webcam using cv2.VideoCapture(0).
    - Retrieve screen dimensions using pyautogui.size() for potential mapping of hand positions to screen coordinates.
3. **Start Video Capture Loop**:
    - Capture frames from the webcam.
    - Flip the image horizontally for a mirror effect.
    - Convert the image from BGR to RGB format as required by Mediapipe.
4. **Hand Tracking**:
    - Initialize Mediapipe's hand tracking solution.
    - Process the RGB image to detect hands and landmarks.

- Draw hand landmarks on the image for visualization.

5. **Gesture Detection**:
    - Iterate over detected hands and landmarks.
    - Extract coordinates of the tip of the index finger (id=8) and thumb (id=4).
    - Calculate the distance between these points.
6. **Volume Control**:
    - If the distance is greater than a threshold (50 pixels), simulate a "volume up" key press using PyAutoGUI.
    - Otherwise, simulate a "volume down" key press.
7. **Display and Cleanup**:
    - Display the processed video feed with hand landmarks.
    - Break the loop if the ESC key is pressed.
    - Release the camera and close all OpenCV windows.

CODE:

```
import cv2
import pyautogui
import mediapipe as mp


camera = cv2.VideoCapture(0)
screen_width, screen_height = pyautogui.size()


while True:
    _, image = camera.read()
    image_height, image_width, _ = image.shape
    image = cv2.flip(image, 1)
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands()
```

```
drawing_options = mp.solutions.drawing_utils

output_hands = hands.process(rgb_image)
all_hands = output_hands.multi_hand_landmarks

if all_hands:
    for hand in all_hands:
        drawing_options.draw_landmarks(image, hand)
        one_hand_landmarks = hand.landmark

        x1, y1, x2, y2 = 0, 0, 0, 0

        for id, lm in enumerate(one_hand_landmarks):
            x = int(lm.x * image_width)
            y = int(lm.y * image_height)
            print(x, y)

            if id == 8:
                #mouse_x = int(screen_width / image_width * x)
                #mouse_y = int(screen_height / image_height * y)
                cv2.circle(image, (x, y), 10, (0, 255, 255))
                #pyautogui.moveTo(mouse_x, mouse_y)
                x1, y1 = x, y

            if id == 4:
                x2, y2 = x, y
                cv2.circle(image, (x, y), 10, (0, 255, 255))

        dist = ((x2-x1)**2 + (y2-y1)**2)**(0.5)//4
```

```
            cv2.line(image,(x1,y1),(x2,y2),(0,255,0),5)
            print(dist)

            if dist > 50:
                pyautogui.press("volumeup")
            else:
                pyautogui.press("volumedown")

        cv2.imshow("Hand movement video capture", image)
        key = cv2.waitKey(100)
        if key == 27:
            break

camera.release()
cv2.destroyAllWindows()
```

Highlights

- **Hand Landmark Detection**: Utilizes Mediapipe's robust hand tracking to detect and track hand landmarks accurately.
- **Distance Calculation**: Computes the Euclidean distance between thumb and index finger tips to decide volume adjustment.
- **Real-Time Interaction**: Integrates webcam feed processing with immediate system volume control using PyAutoGUI.
- **User Feedback**: Visual feedback is provided by drawing landmarks and distance lines on the video feed.

Conclusion

This project successfully demonstrates the application of hand gesture recognition for controlling system volume. The integration of OpenCV, Mediapipe, and PyAutoGUI allows for a seamless real-time interaction system. Future improvements could include expanding gesture recognition for more complex controls and enhancing the user interface for better feedback.

Feel free to adjust any part of this outline to better fit your specific needs or preferences!