```
import cv2
import mediapipe as mp
import os
import time
import math

# ----------------------------
# MediaPipe Setup
# ----------------------------
BaseOptions = mp.tasks.BaseOptions
HandLandmarker = mp.tasks.vision.HandLandmarker
HandLandmarkerOptions = mp.tasks.vision.HandLandmarkerOptions
VisionRunningMode = mp.tasks.vision.RunningMode

options = HandLandmarkerOptions(
   base_options=BaseOptions(model_asset_path="hand_landmarker.task"),
   running_mode=VisionRunningMode.VIDEO,
   num_hands=1
)

landmarker = HandLandmarker.create_from_options(options)

# Hand skeleton connections
HAND_CONNECTIONS = [
   (0,1),(1,2),(2,3),(3,4),
   (0,5),(5,6),(6,7),(7,8),
   (5,9),(9,10),(10,11),(11,12),
   (9,13),(13,14),(14,15),(15,16),
   (13,17),(17,18),(18,19),(19,20),
   (0,17)
]
```

```
cap = cv2.VideoCapture(0)


last_action_time = 0

cooldown = 1.2


gesture_buffer = []

buffer_size = 5


# volume mode variables

volume_mode = False

volume_mode_start = 0


# ----------------------------

# Finger Counting

# ----------------------------

def count_fingers(hand_landmarks):

    fingers = []

    tips = [4, 8, 12, 16, 20]


    if hand_landmarks[tips[0]].x < hand_landmarks[tips[0]-1].x:

        fingers.append(1)

    else:

        fingers.append(0)


    for tip in tips[1:]:

        if hand_landmarks[tip].y < hand_landmarks[tip-2].y:

            fingers.append(1)

        else:

            fingers.append(0)
    return fingers.count(1)


# ----------------------------
# Distance for volume control
# ----------------------------
def finger_distance(hand_landmarks):
    x1, y1 = hand_landmarks[4].x, hand_landmarks[4].y
    x2, y2 = hand_landmarks[8].x, hand_landmarks[8].y
    return math.sqrt((x2 - x1)*2 + (y2 - y1)*2)


# ----------------------------
# Main Loop
# ----------------------------
while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.flip(frame, 1)
    frame = cv2.resize(frame, (480, 360))
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=rgb_frame)
    timestamp = int(time.time() * 1000)
    result = landmarker.detect_for_video(mp_image, timestamp)

    if result.hand_landmarks and result.handedness:

        confidence = result.handedness[0][0].score
        if confidence < 0.8:
            continue
```

```
hand_landmarks = result.hand_landmarks[0]

# filter small false detections
xs = [lm.x for lm in hand_landmarks]
if (max(xs) - min(xs)) < 0.15:
    continue

# convert to pixel coordinates
points = []
for lm in hand_landmarks:
    x = int(lm.x * frame.shape[1])
    y = int(lm.y * frame.shape[0])
    points.append((x, y))

# draw skeleton
for pt in points:
    cv2.circle(frame, pt, 6, (0,255,0), -1)

for connection in HAND_CONNECTIONS:
    cv2.line(frame, points[connection[0]], points[connection[1]], (255,0,0), 2)

finger_count = count_fingers(hand_landmarks)

# smoothing
gesture_buffer.append(finger_count)
if len(gesture_buffer) > buffer_size:
    gesture_buffer.pop(0)

stable_gesture = max(set(gesture_buffer), key=gesture_buffer.count)
```

```
cv2.putText(frame, f"Fingers: {stable_gesture}", (20, 50),
            cv2.FONT_HERSHEY_SIMPLEX, 1,
            (0,0,255), 3)

current_time = time.time()

# ---------------------------
# ENTER volume mode (✌ hold)
# ---------------------------
if stable_gesture == 2 and not volume_mode:
    if volume_mode_start == 0:
        volume_mode_start = time.time()
    elif time.time() - volume_mode_start > 1:
        volume_mode = True
        print("Volume Mode ON")
else:
    volume_mode_start = 0

# EXIT volume mode (fist)
if stable_gesture == 0 and volume_mode:
    volume_mode = False
    print("Volume Mode OFF")
    time.sleep(0.5)

# ---------------------------
# NORMAL MEDIA CONTROLS
# ---------------------------
if not volume_mode and current_time - last_action_time > cooldown:

    if stable_gesture == 5:
        os.system("playerctl -p vlc play")
        cv2.putText(frame, "PLAY", (20,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (0,255,0), 3)
        last_action_time = current_time

    elif stable_gesture == 0:
        os.system("playerctl -p vlc pause")
        cv2.putText(frame, "PAUSE", (20,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (0,255,0), 3)
        last_action_time = current_time

    elif stable_gesture == 1:
        os.system("playerctl -p vlc previous")
        cv2.putText(frame, "PREVIOUS", (20,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (0,255,0), 3)
        last_action_time = current_time

    elif stable_gesture == 3:
        os.system("playerctl -p vlc next")
        cv2.putText(frame, "NEXT", (20,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (0,255,0), 3)
        last_action_time = current_time

# ----------------------------
# VOLUME CONTROL MODE
# ----------------------------
if volume_mode:
    distance = finger_distance(hand_landmarks)
```

```
            if distance > 0.16:
                os.system("pamixer -i 3")
                cv2.putText(frame, "VOL UP", (20,150),
                            cv2.FONT_HERSHEY_SIMPLEX, 1,
                            (255,255,0), 3)
            else:
                os.system("pamixer -d 3")
                cv2.putText(frame, "VOL DOWN", (20,150),
                            cv2.FONT_HERSHEY_SIMPLEX, 1,
                            (255,255,0), 3)

            cv2.putText(frame, "VOLUME MODE", (20,200),
                        cv2.FONT_HERSHEY_SIMPLEX, 1,
                        (0,255,255), 3)

    cv2.imshow("Touchless Media Control", frame)

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

cap.release()
cv2.destroyAllWindows()
```