Gesture-controlled movement

1. Purpose of the experiment

The robot dog will detect and recognize the current gesture and make corresponding movements.

2. Main source code path

First, end the big program, then open the browser and enter "ip (ip is the ip of the robot dog): 8888", enter the password "yahboom" and enter

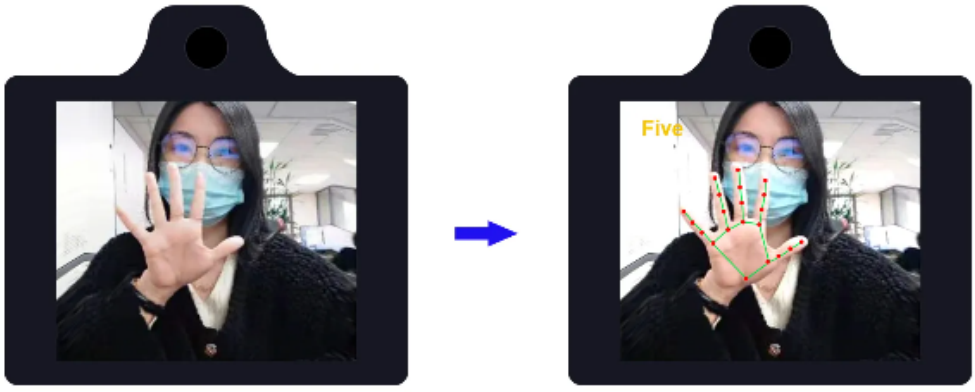
the path to ~/DOGZILLA_Lite_class/6.AI Visual Interaction Course/10.Gesture controlled. Open the **hands.ipynb** program and run it ，or enter it in the terminal

```
cd ~/DOGZILLA_Lite_class/6.AI Visual Interaction Course/10.Gesture controlled
python3 hands.py
```

3. Experimental Phenomenon

After running the source code, the robot dog will detect and recognize the current gesture and perform corresponding movements.

Recognize gestures such as 1, 2, 3, 4, 5, 6, good, fist, etc. Note that you need to wait until the robot dog completes the previous action before recognizing again.



Perform corresponding actions for the recognized numbers

- 1: Roll
- 2: Rotate the pitch

- 3: Rotate yaw
- 4: Look around
- 5: Get Down
- 6: Naughty
- good: dancing
- Fist: Chicken head

4. Main source code analysis

```
try:
  with mp_hands.Hands(
      model_complexity=0,
      min_detection_confidence=0.5,
      min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
      ret, frame = cap.read()
      if not ret:
          print("Can not receive frame (stream end?). Exiting...")
          break
      frame_RGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
      result = hands.process(frame_RGB)
      frame_height = frame.shape[0]
      frame_width  = frame.shape[1]
      gesture_result=[]
      if result.multi_hand_landmarks:
          for i, handLms in enumerate(result.multi_hand_landmarks):
              mpDraw.draw_landmarks(frame,
                                    handLms,
                                    mpHands.HAND_CONNECTIONS,
                                    landmark_drawing_spec=handLmsStyle,
                                    connection_drawing_spec=handConStyle)

              for j, lm in enumerate(handLms.landmark):
                  xPos = int(lm.x * frame_width)
                  yPos = int(lm.y * frame_height)
                  landmark_ = [xPos, yPos]
                  landmark[j,:] = landmark_

              for k in range (5):
                  if k == 0:
                      figure_ =
finger_stretch_detect(landmark[17],landmark[4*k+2],landmark[4*k+4])
                  else:
                      figure_ =
finger_stretch_detect(landmark[0],landmark[4*k+2],landmark[4*k+4])

                  figure[k] = figure_

              gesture_result = detect_hands_gesture(figure)

      b,g,r = cv2.split(frame)
      frame = cv2.merge((r,g,b))
      frame = cv2.flip(frame, 1)
      if result.multi_hand_landmarks:
        cv2.putText(frame, f"{gesture_result}", (10,30),
cv2.FONT_HERSHEY_COMPLEX, 1, (255 ,255, 0), 5)
```

```
      if time.time()>dogtime:
        if gesture_result=="good":
          dogtime=time.time()
          dog.action(23)
          dogtime+=3
        elif gesture_result=="one":
          dogtime=time.time()
          dog.action(7)
          dogtime+=3
        elif gesture_result=="two":
          dogtime=time.time()
          dog.action(8)
          dogtime+=3
        elif gesture_result=="three":
          dogtime=time.time()
          dog.action(9)
          dogtime+=3
        elif gesture_result=="four":
          dogtime=time.time()
          dog.action(22)
          dogtime+=3
        elif gesture_result=="five":
          dogtime=time.time()
          dog.action(1)
          dogtime+=3
        elif gesture_result=="six":
          dogtime=time.time()
          dog.action(24)
          dogtime+=3
        elif gesture_result=="OK":
          dogtime=time.time()
          dog.action(19)
          dogtime+=3
        elif gesture_result=="stone":
          dogtime=time.time()
          dog.action(20)
          dogtime+=3

      imgok = Image.fromarray(frame)
      display.ShowImage(imgok)

      r,g,b = cv2.split(frame)
      framecv2 = cv2.merge((b,g,r))
      cv2.imshow('framecv2',framecv2)

      if cv2.waitKey(5) & 0xFF == 27:
        break
      if button.press_b():
        dog.reset()
        break

except:
  dog.reset()
  cap.release()
```

The above source code will detect and recognize the current gesture and control the robot dog to perform corresponding actions through the action-API.