8、Gesture controlled robot

The Raspberry Pi motherboard series does not currently support this tutorial.

8.1、Introduction to gameplay

The gesture control robot gameplay is a function realized by combining the MediaPipe machine learning framework and the preset action group of the Muto robot. The joint point position information of the fingers is detected through the MediaPipe framework, and how many fingers are first divided according to the characteristics of gestures 1-8. It is in the open state, and then the gestures 1-8 are calculated, which correspond to the eight preset actions for controlling the robot.

8.2、Core content analysis

Initialize the Muto robot and set the initial angle of the camera gimbal. The default is S1=90 and S2=90. The angle of the camera gimbal can be modified according to the actual viewing angle.

```
self.__robot = Muto()
self.__robot.Gimbal_1_2(90, 90)
```

Import the mediapipe module and other related modules

```
import math
import cv2 as cv
import numpy as np
import mediapipe as mp
```

Initialize the USB camera. The default is USB camera device /dev/video0. The USB camera device number can be modified according to the device number found in the system.

```
capture = cv.VideoCapture(0)
capture.set(6, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
```

Initialize the gesture detection object. detectorCon represents the detection value, which can be modified according to the actual effect. The recommended range is 0.5-1.0.

```
hand_detector = handDetector(detectorCon=0.75)
```

Read the camera image and then transfer it to the findHands function to calculate and output the detection result.

```
ret, frame = capture.read()
frame, img = hand_detector.findHands(frame, draw=False)
```

Determine whether the length of hand_detector.lmList is 0. If it is non-0, it means that the gesture feature is detected. Call the get_gesture function to extract the name of the gesture feature, then display the name on the image, and finally perform related actions.

```
if len(hand_detector.lmList) != 0:
    totalFingers = hand_detector.get_gesture()
    cv.rectangle(frame, (0, 430), (230, 480), (0, 255, 0), cv.FILLED)
    cv.putText(frame, str(totalFingers), (10, 470), cv.FONT_HERSHEY_PLAIN, 2, (255,
0, 0), 2)
    # print("Finger", totalFingers)
    hand_detector.parse_action(totalFingers)
```

The main function of the get_gesture function is to calculate how many of the five fingers are open based on the gesture characteristics. The fingersUp function calculates how many of the five fingers are open, then distinguishes the gesture characteristics based on the angle differences of different gestures, and finally converts them into English characters of the numbers 0-8.

```
def get_gesture(self):
        gesture = ""
        fingers = self.fingersUp()
        if fingers.count(1) == 3: gesture = "Three"
        elif fingers.count(1) == 4: gesture = "Four"
        elif fingers.count(1) == 0: gesture = "Zero"
        elif fingers.count(1) == 1: gesture = "One"
        elif fingers.count(1) == 2:
            if fingers[0] == 1 and fingers[4] == 1: gesture = "Six"
            elif fingers[0] == 1 and self.calc_angle(4, 5, 8) > 90: gesture =
"Eight"
            else: gesture = "Two"
        elif fingers.count(1)==5:
            if self.get_dist(self.lmList[4][1:], self.lmList[8][1:]) < 60 and \
                    self.get_dist(self.lmList[4][1:], self.lmList[12][1:]) < 60 and
\
                    self.get_dist(self.lmList[4][1:], self.lmList[16][1:]) < 60 and
\
                    self.get_dist(self.lmList[4][1:], self.lmList[20][1:]) < 60 :
gesture = "Seven"
            else:
                gesture = "Five"
        return gesture
```

Next, the Muto robot is controlled to perform actions according to the name of the gesture feature. If gesture 1 is recognized, action 1 will be performed. If gesture 2 is recognized, action 2 will be performed. And so on. If gesture 8 is recognized, action 8 will be executed.

```
def control_robot(self, action):
    if self.count > 0:
        return
    self.count = 20
    if action == "One":
        self.__robot.action(1)
    elif action == "Two":
        self.__robot.action(2)
    elif action == "Three":
        self.__robot.action(3)
    elif action == "Four":
        self.__robot.action(4)
    elif action == "Five":
        self.__robot.action(5)
    elif action == "Six":
        self.__robot.action(6)
    elif action == "Seven":
        self.__robot.action(7)
    elif action == "Eight":
        self.__robot.action(8)
    else:
        self.count = 0
    if self.count > 0:
        print("action:", action)
```

Finally, the original image and the detected gesture feature joint point image are merged and displayed on the desktop.

```
dist = hand_detector.frame_combine(frame, img)
cv.imshow('dist', dist)
```

8.3、Program execution and operation

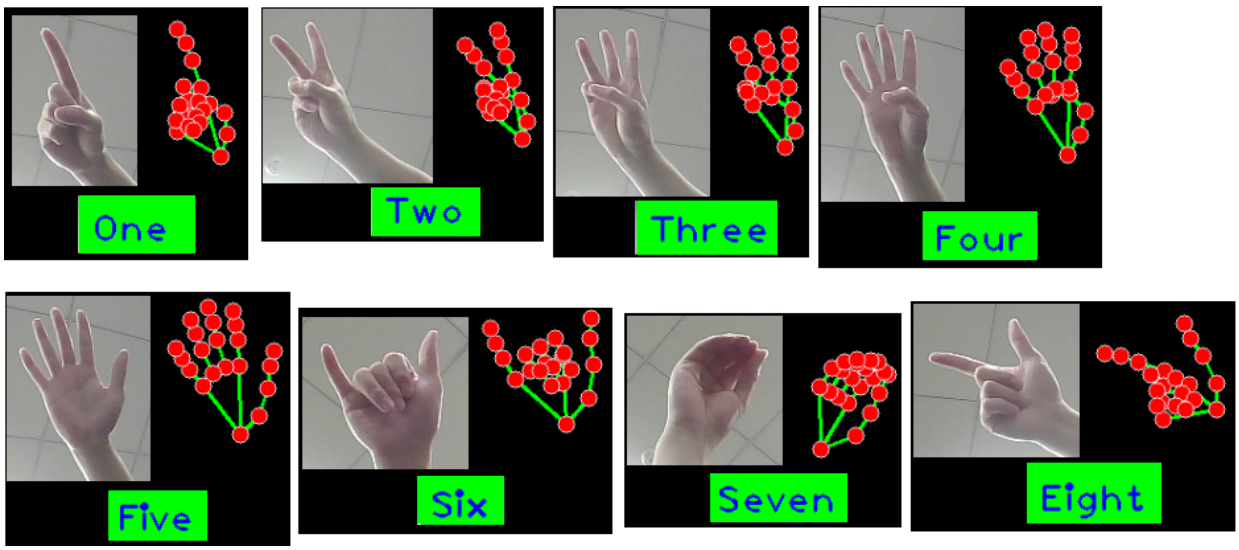
Note that the following commands need to be run in the system desktop terminal. You can connect the display or use VNC to remotely log in to the desktop to operate.

Open the terminal on the system desktop and run the following command

```
python3 ~/muto/Samples/Deep_Learning/gesture_action.py
```

At this time, the program starts running, and the camera screen pops up on the desktop. Please put your hand into the camera detection range and make gestures 1-8. After the robot recognizes the gesture characteristics, it will perform the corresponding action.

The corresponding actions of gesture features are shown in the figure below:



If you need to exit the program, press Ctrl+C in the terminal interface to exit the program.