10. Mediapipe gesture control robotic arm action group

10.1. Introduction

MediaPipe is a data stream processing machine learning application development framework developed and open sourced by Google. It is a graph-based data processing pipeline for building data sources that use many forms, such as video, audio, sensor data, and any time series data. MediaPipe is cross-platform and can run on embedded platforms (Raspberry Pi, etc.), mobile devices (iOS and Android), workstations and servers, and supports mobile GPU acceleration. MediaPipe provides cross-platform, customizable ML solutions for real-time and streaming.

10.2. Use

Note: [R2] on the remote control handle has the [pause/start] function for this gameplay.

The case in this section may run very slowly on the robot main control. It is recommended to connect the camera to the virtual machine. The NX main control effect will be better. You can try it.

jetson motherboard/Raspberry Pi 4B

```
roslaunch arm_mediapipe mediaArm.launch # robot
rosrun arm_mediapipe FingerCtrl.py # The robot can also be started in a virtual machine, but the virtual machine needs to be configured with a camera
```

Raspberry Pi 5

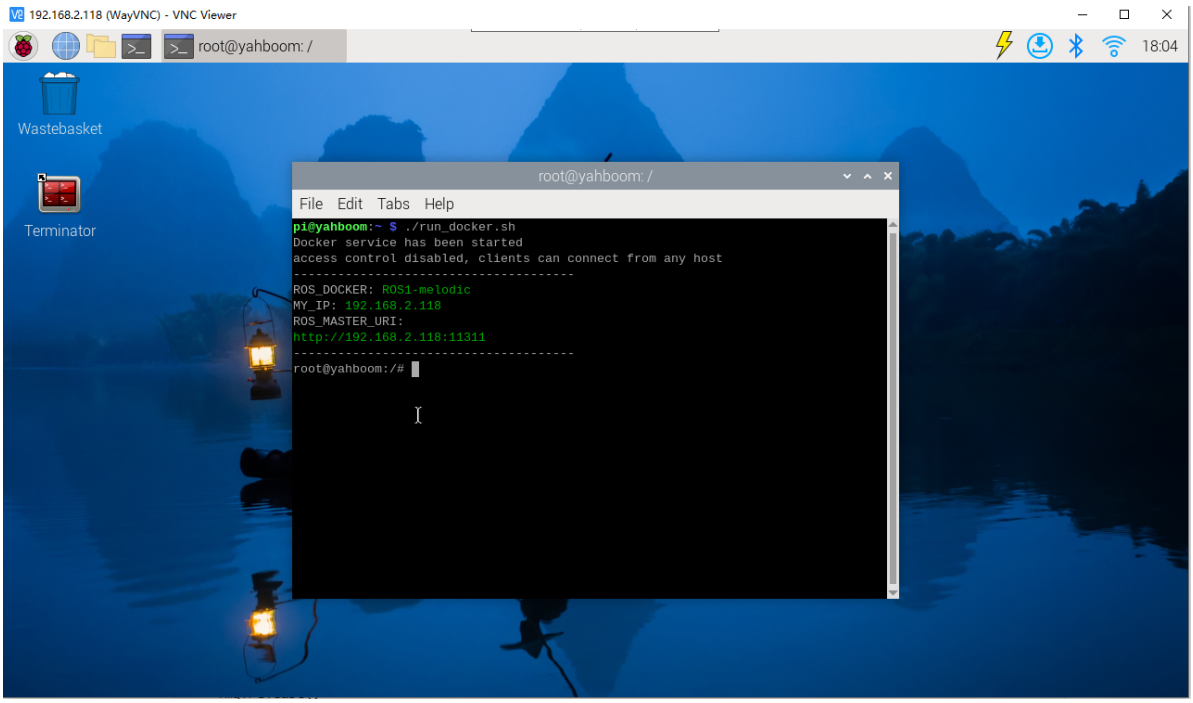
After the program is running, press the R2 key of the handle to touch the control. The camera will capture images and there are five gestures, as follows **Before running, please confirm that the large program has been permanently closed**

Enter docker

Note: If there is a terminal that automatically starts docker, or there is a docker terminal that has been opened, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```



robot side

```
roslaunch arm_mediapipe mediaArm.launch
```

Enter the same docker from multiple terminals

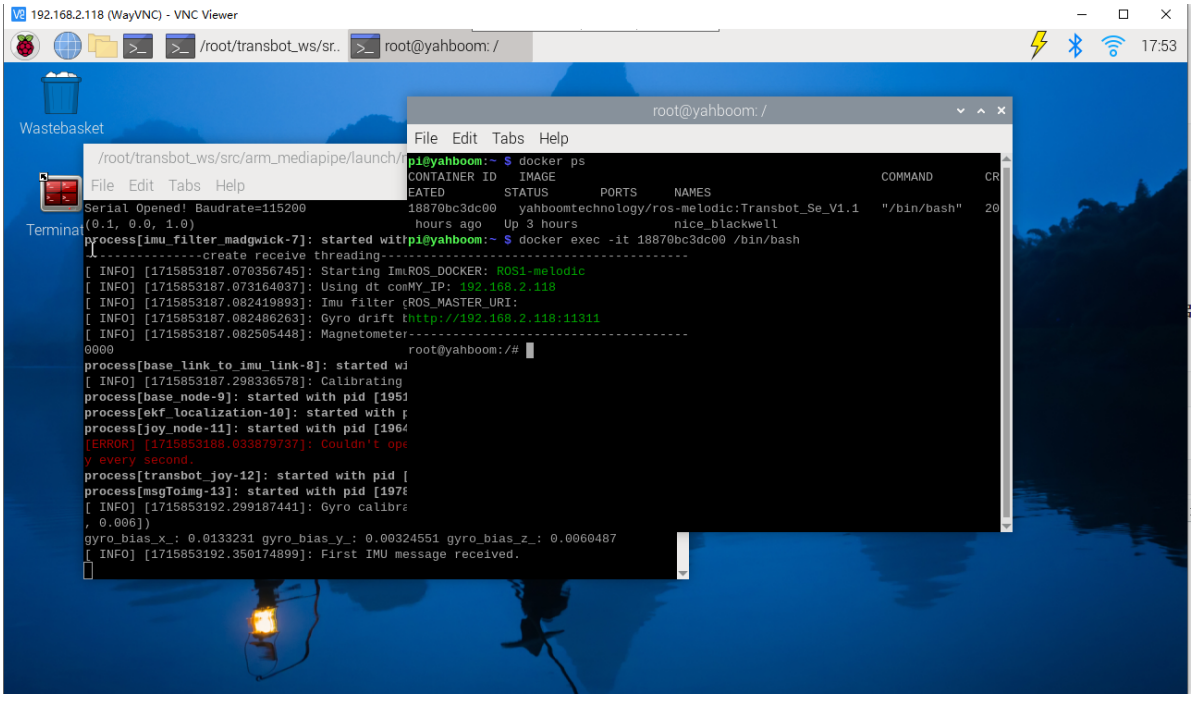
Keep the program of the previous docker terminal running and open a new terminal

Enter the following command

```
docker ps
```

Enter the same docker and use the following 18870bc3dc00 to modify the ID displayed on the actual terminal.

```
docker exec -it 18870bc3dc00 /bin/bash
```



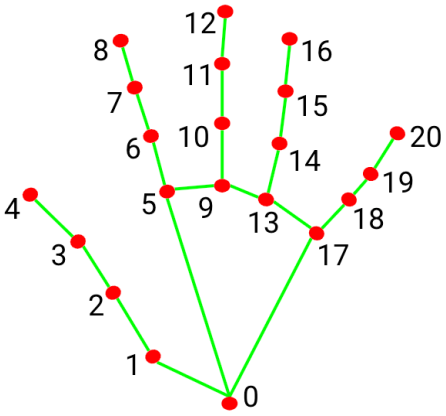
It is recommended that a virtual machine (equipped with a camera) can also be run on a car

```
rosrun arm_mediapipe RobotCtrl.py
```

- Gesture Yes: Robotic arm dancing
- Gesture OK: the robotic arm straightens, the gripper opens and closes alternately.
- Gestures contempt (clenched fist, thumbs out, thumbs down): the robotic arm raises its body,the gripper opens
- Gesture number 1: robotic arm nods
- Gesture number 5: robotic arm applauding

Here, when each gesture is finished, it will return to the initial position and beep, waiting for the next gesture recognition.

MediaPipe Hands infers the 3D coordinates of 21 hand-valued joints from a frame.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

10.3. Core files

10.3.1、mediaArm.launch

```
<launch>
    <include file="$(find transbot_bringup)/launch/bringup.launch"/>
    <node name="msgToimg" pkg="arm_mediapipe" type="msgToimg.py" output="screen"
required="true"/>
</launch>
```

10.3.2、 FingerCtrl.py

```
The implementation process here is also very simple. The main function opens the
camera to obtain data and then passes it into the process function. Inside it,
"detect palm" -> "obtain finger coordinates" -> "obtain gestures" in sequence,
and then decide what needs to be done according to the gesture results action
performed frame, lmList, bbox=
self.hand_detector.findHands(frame) #detect palm
fingers = self.hand_detector.fingersUp(lmList)  #get finger coordinates
gesture = self.hand_detector.get_gesture(lmList)    #get gesture
For the specific implementation process of the above three functions, you can
refer to the content in media_library.py
```

The implementation process here is also very simple. The main function opens the camera to obtain data and then passes it into the process function. Inside it, "detect palm" -> "obtain finger coordinates" -> "obtain gestures" in sequence, and then determine the needs according to the gesture results action performed.

10.4. Flowchart