11. Mediapipe gesture recognition to control car movement

11.1. Use

After the function is turned on, the camera captures images and recognizes gestures to control the movement of the car. **Multi-machine communication needs to be configured**

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

jetson motherboard/Raspberry Pi 4B

Start command (robot side)

```
roslaunch arm_mediapipe mediaArm.launch
```

Start command (virtual machine)

```
rosrun arm_mediapipe HandCtrl.py
```

Raspberry Pi 5

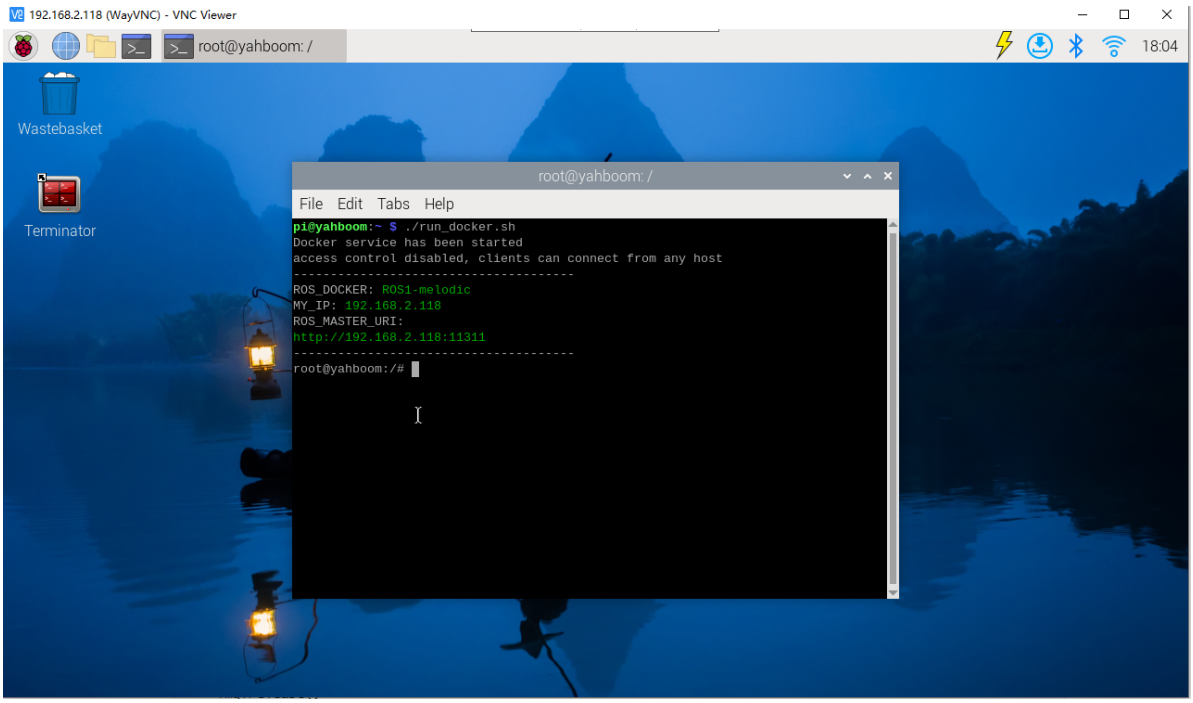
Before running, please confirm that the large program has been permanently closed

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
```



Start command (robot side)

```
roslaunch arm_mediapipe mediaArm.launch
```

Start command (virtual machine)

```
rosrun arm_mediapipe HandCtrl.py
```

After the program is started, press the R2 button on the handle to turn on the function, and then put your hand in front of the camera. The shape of the finger will be drawn on the screen. After the program recognizes the gesture, it will send the speed to the chassis to control the movement of the car.

Gesture number 1: the car moves left

Gesture number 2: the car moves right

Gesture number 3: the camera rotates left

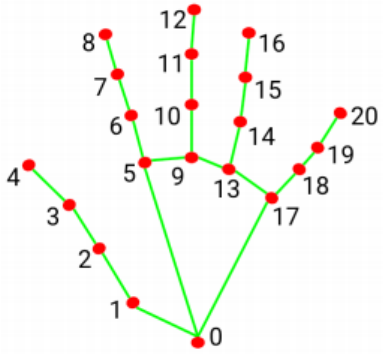
Gesture number 4: the camera rotates right

Gesture number 5: the car moves forward

Gesture fist: the car moves back

Gesture rock (the index finger and the little finger are straight, the others are bent): the buzzer sounds

MediaPipe Hands infers 3D coordinates of 21 hand-valued joints from one 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

11.2. Core code analysis HandCtrl.py

Code reference path：~/transbot_ws/src/arm_mediapipe/scripts

- Import critical libraries

```
from media_library import * #This library contains functions to detect
hands, get gestures, etc.
```

- Get finger data

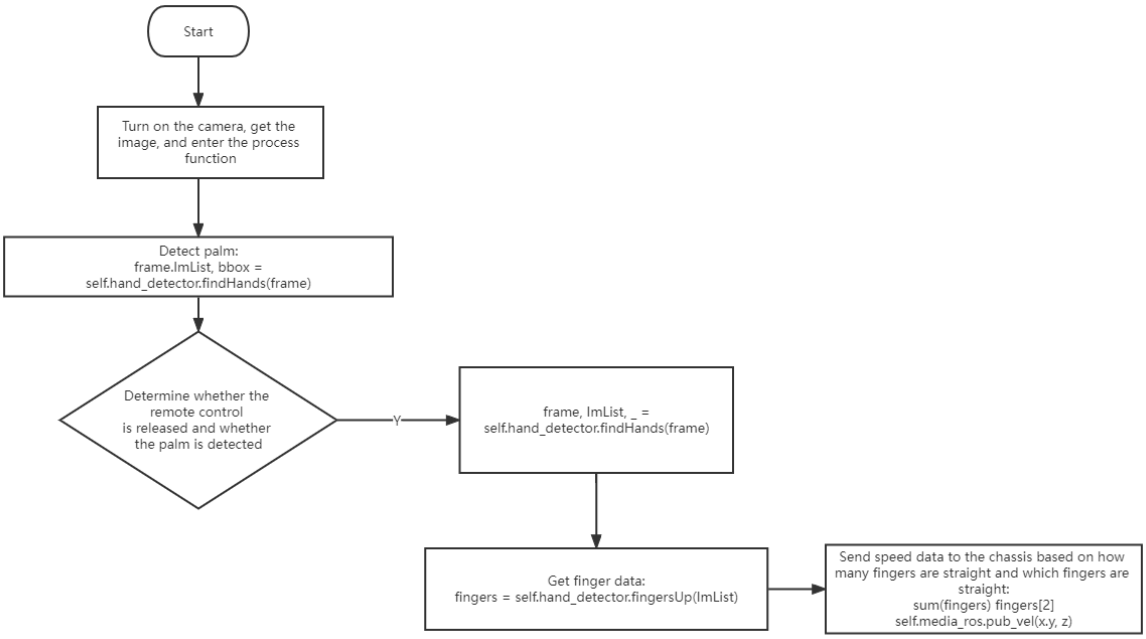
```
frame, lmList, _ = self.hand_detector.findHands(frame)
fingers = self.hand_detector.fingersUp(lmList)
sum(fingers)
fingers[]
```

It can be seen that the hand is detected first, the value of lmList is obtained, and then the fingersUp function is passed in. The fingersUp function is used to detect which fingers are straight. The value of the straight finger is 1. The specific code here can also be seen in the **media_library, py** function, which has a detailed explanation, which is actually to judge the finger joints xy value to judge when it is straightened. The sum(fingers) function is used to calculate the number of straight fingers, fingers[] can be used to enumerate fingers, such as the index finger, we use fingers[1] to represent.

- Post speed to chassis

```
self.media_ros.pub_vel(x,y,z)   #This function is also in media_library,py
```

11.3.Flowchart


Start
Turn on the camera, get the image, and enter the process function
Detect palm:
frame.lmList, bbox =
self.hand_detector.findHands(frame)
Determine whether the remote control is released and whether the palm is detected
Y
frame, lmList, _ =
self.hand_detector.findHands(frame)
Get finger data:
fingers = self.hand_detector.fingersUp(lmList)
Send speed data to the chassis based on how many fingers are straight and which fingers are straight:
sum(fingers) fingers[2]
self.media_ros.pub_vel(x.y, z)