10. Mediapipe gesture control robotic arm action group

10.1. Introduction

MediaPipe is a data stream processing machine learning application development framework developed by Google and open source. It is a graph-based data processing pipeline for building data sources using 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 live and streaming media.

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 and run the [02_PoseCtrlArm.launch] file. The NX main control effect will be better. You can try it.

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

```
roslaunch arm_mediapipe mediaArm.launch # robot
```

<PI5 needs to open another terminal to enter the same docker container

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                 COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9   "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$
```

2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                 COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9   "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$ docker exec -it 5b698ea10535 /bin/bash
------------------------------------------------------
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
------------------------------------------------------
root@ubuntu:/#
```

After successfully entering the container, you can open countless terminals to enter the container.

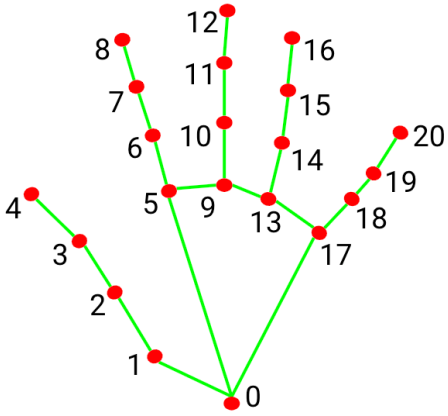
```
rosrun arm_mediapipe FingerCtrl.py # The robot can also be started in a virtual machine, but the virtual machine needs to be configured with a camera
```

After the program is running, press the R2 key of the handle to touch the control. The camera will capture the image and there are six gestures, as follows

- Gesture Yes: the robotic arm dances
- Gesture OK: The robotic arm shakes its head and the clamping claws tighten
- Gesture contempt (clench fist, extend thumb, thumb down): The robotic arm kneels down and the clamping claws are tightened
- Gesture number 1: Robotic arm nods
- Gesture rock (the index finger and little finger are straightened, and the other fingers are bent): the robotic arm is straightened and turned left and right
- Gesture number 5: robotic arm applauds

Here, after each gesture is completed, 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

10.4. Core files

10.4.1、mediaArm.launch

```
<launch>
     <include file="$(find yahboomcar_ctrl)/launch/yahboom_joy.launch"/>
     <include file="$(find yahboomcar_bringup)/launch/yahboomcar.launch"/>
     <node pkg="web_video_server" type="web_video_server" name="web_video_server"
output="screen"/>
     <node name="msgToimg" pkg="arm_mediapipe" type="msgToimg.py" output="screen"
required="true"/>
</launch>
```

10.4.2, FingerCtrl.py

```
frame, lmList, bbox = self.hand_detector.findHands(frame) #Detect palms
fingers = self.hand_detector.fingersUp(lmList) #Get finger coordinates
gesture = self.hand_detector.get_gesture(lmList) #Get gestures
For the specific implementation process of the above three functions, please
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 to the process function, which performs "detect palm" -> "get finger coordinates" -> "get gesture" in order, and then determine the needs based on the gesture results. The action performed.

10.5. Flowchart