Mediapipe gesture tracking

Mediapipe gesture tracking
- 1. Program Functionality
- 2. Program Code Reference Path
- 3. Program Startup
  - 3.1. Startup Command
  - 3.2 Dynamic Parameter Adjustment
- 4. Core Code
  - 4.1. gestureTracker.py

1. Program Functionality

> After the program starts, it automatically detects hand gestures in the image. The servo gimbal will lock onto the detected hand gesture position (the base of the index finger joint), keeping it in the center of the image. Tracking stops when a "0" (fist) gesture is detected, while tracking continues for other gestures. To exit the program, press `Ctrl+c` in the terminal.

2. Program Code Reference Path

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_astra/gestureTracker.py
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_astra/common/media_common.py
```

- gestureTracker.py
  Mainly performs gesture detection and tracking. Based on the detected gesture keypoint coordinates, it calculates the desired servo rotation angle and publishes the servo control angle data to the car.
- media_common.py

  This is a gesture and posture recognition system based on MediaPipe.

  - HandDetector Class:
    - Real-time hand keypoint detection and tracking
    - Gesture recognition (supports multiple gestures, including 0-8)
    - Provides finger state detection and angle calculation

3. Program Startup

3.1. Startup Command

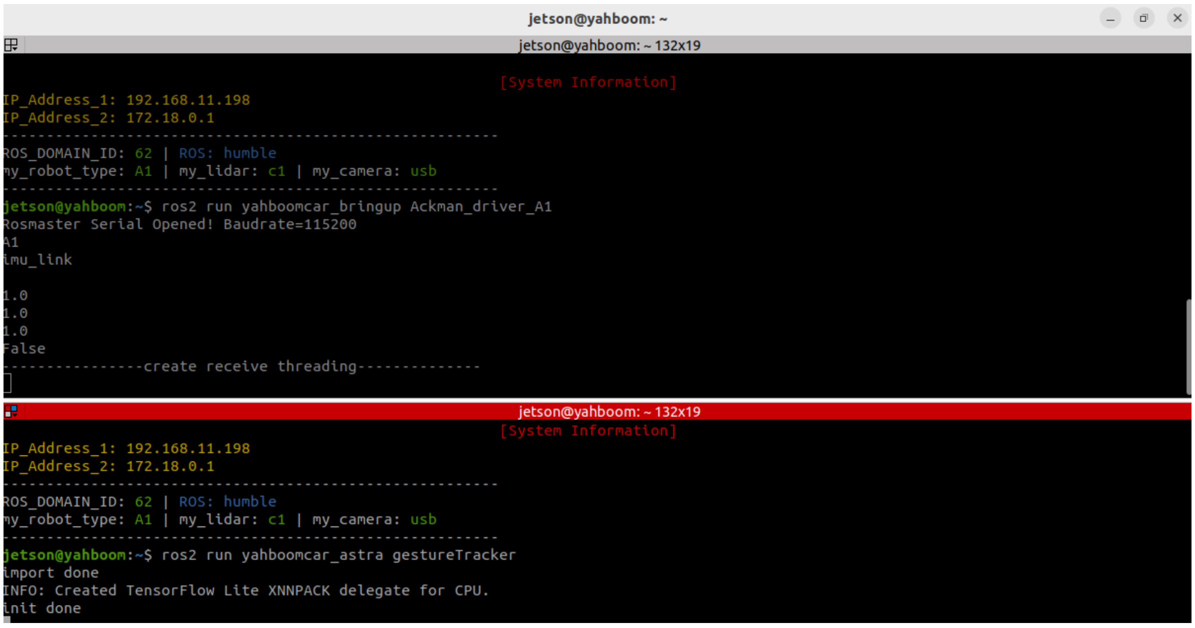
For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

Enter the Docker container (see [Docker course] --- [4. Docker Startup Script] for steps).

All the following commands must be executed within the same Docker container **(see [Docker course] --- [3. Docker Submission and Multi-Terminal Access] for steps).**

Enter the terminal.

```
# Start the car chassis
ros2 run yahboomcar_bringup Mcnamu_driver_M1
# Start the gesture tracking program
ros2 run yahboomcar_astra gestureTracker
```

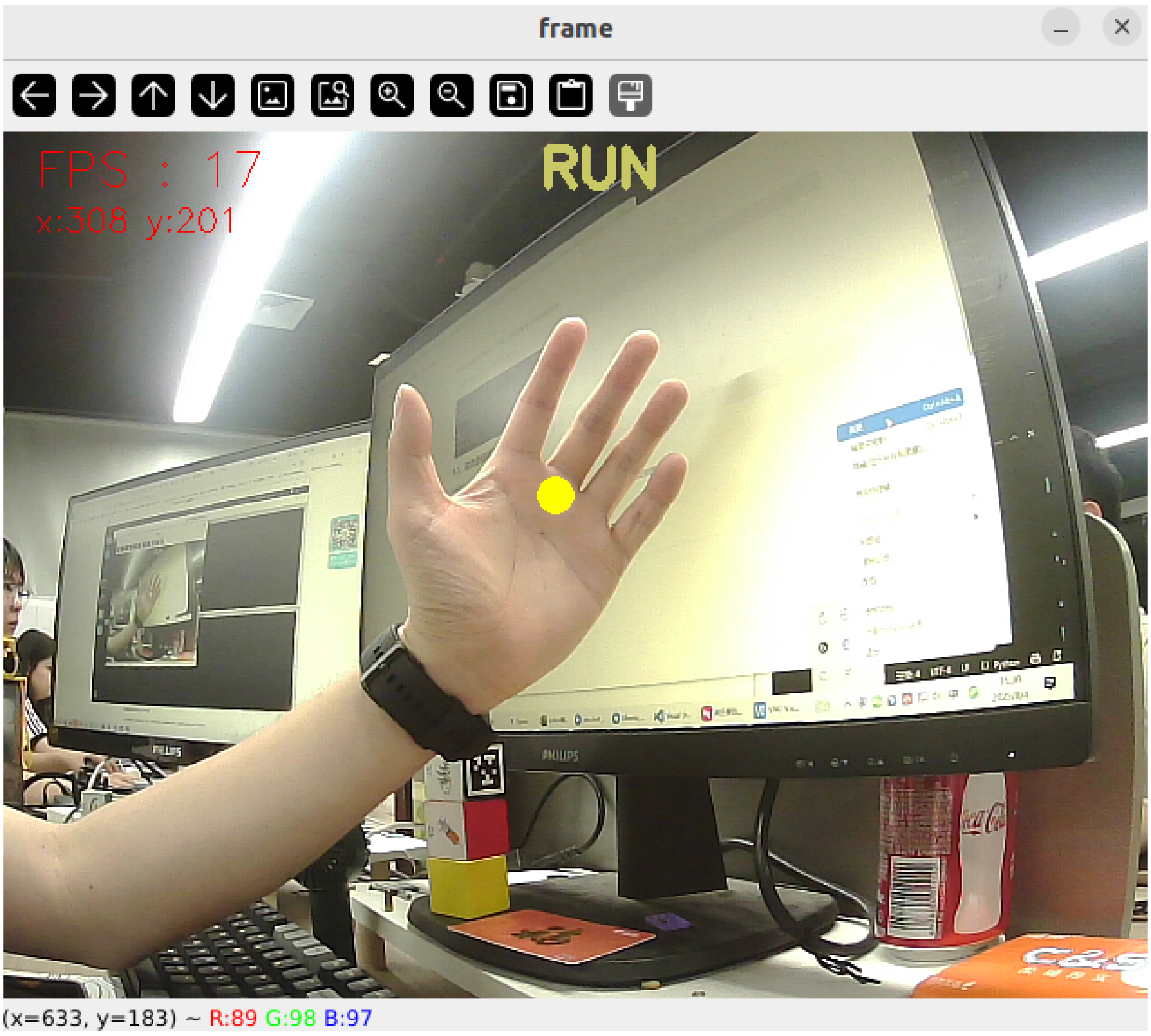


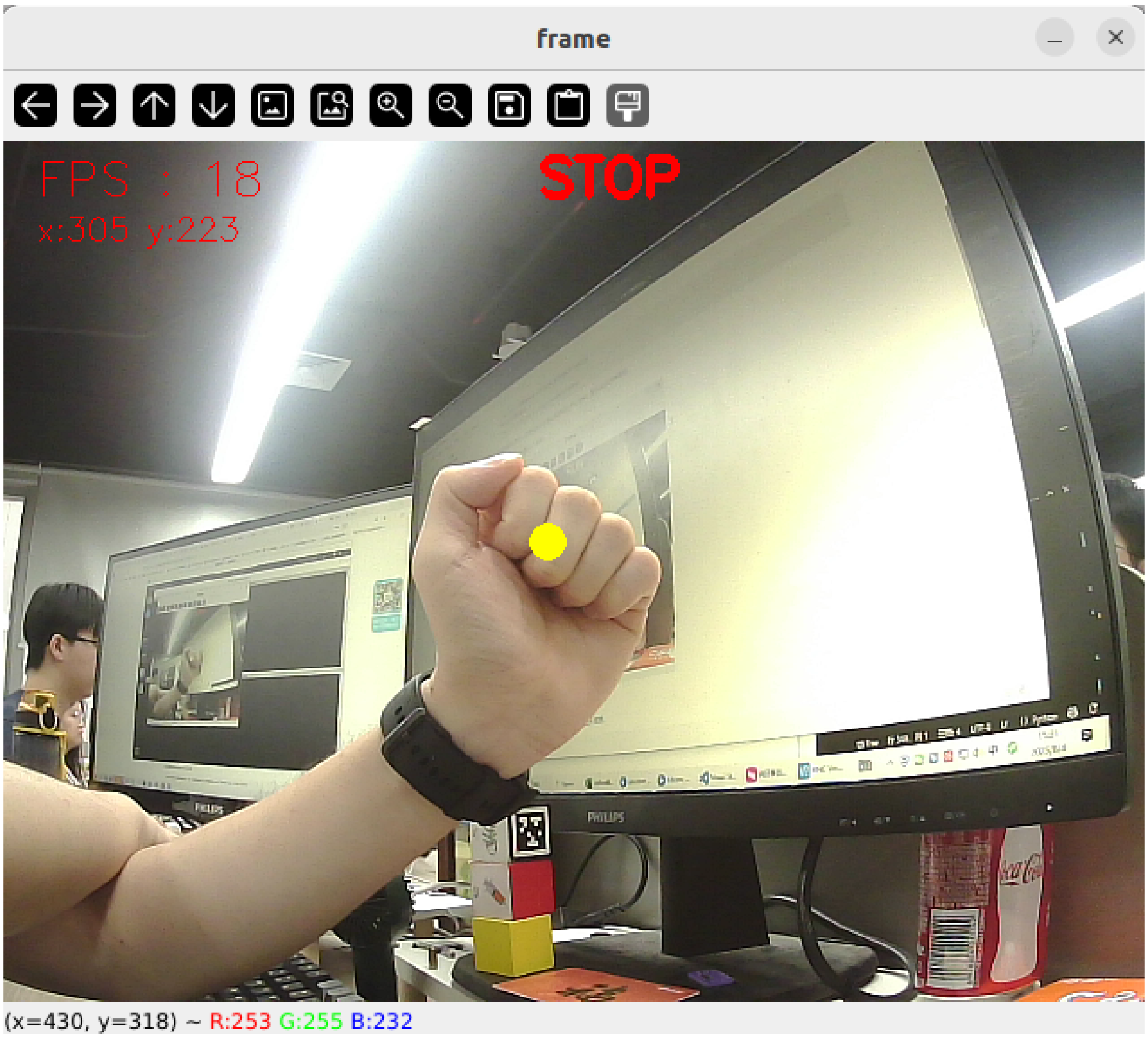
```
jetson@yahboom: ~
jetson@yahboom: ~ 132x19
                                                                  [System Information]
IP_Address_1: 192.168.11.198
IP_Address_2: 172.18.0.1
------------------------------------------------------
ROS_DOMAIN_ID: 62 | ROS: humble
my_robot_type: A1 | my_lidar: c1 | my_camera: usb
------------------------------------------------------
jetson@yahboom:~$ ros2 run yahboomcar_bringup Ackman_driver_A1
Rosmaster Serial Opened! Baudrate=115200
A1
imu_link

1.0
1.0
1.0
False
---------------create receive threading--------------
```

```
jetson@yahboom: ~ 132x19
                                                                  [System Information]
IP_Address_1: 192.168.11.198
IP_Address_2: 172.18.0.1
------------------------------------------------------
ROS_DOMAIN_ID: 62 | ROS: humble
my_robot_type: A1 | my_lidar: c1 | my_camera: usb
------------------------------------------------------
jetson@yahboom:~$ ros2 run yahboomcar_astra gestureTracker
import done
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
init done
```

When a gesture is detected, a yellow dot is drawn in the center of the palm and the center coordinates are displayed. The servo gimbal begins tracking.

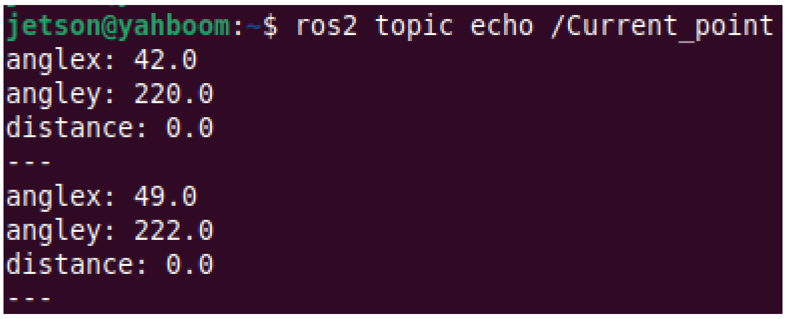
After starting the program, the following screen will appear.


frame
FPS : 17
x:308 y:201
RUN
(x=633, y=183) ~ R:89 G:98 B:97



In addition, we can also enter the following command to print information about the target center coordinates.

```
ros2 topic echo /Current_point
```

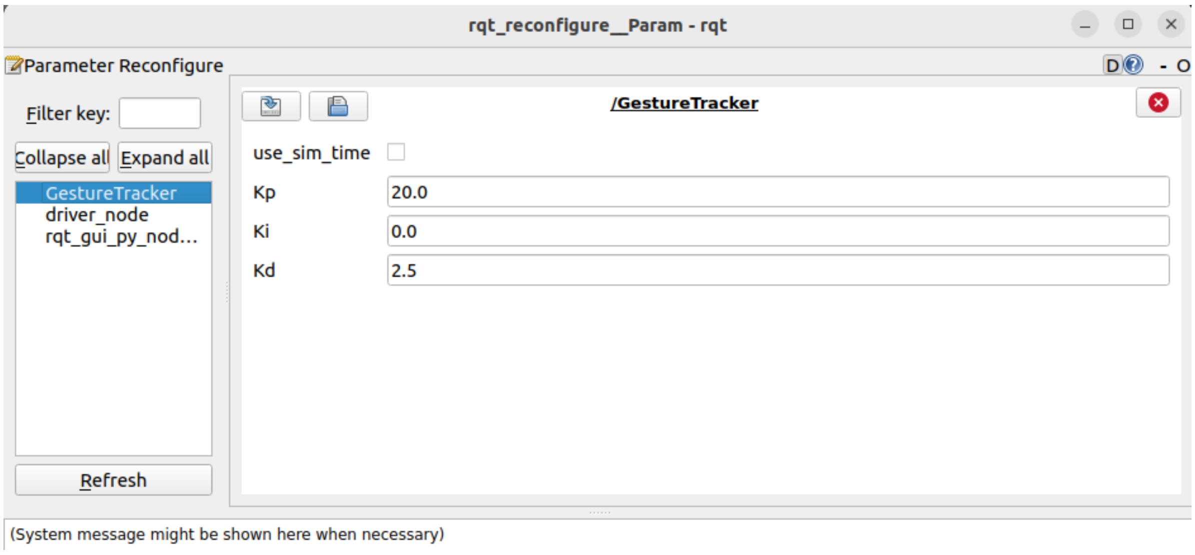


```
jetson@yahboom:~$ ros2 topic echo /Current_point
anglex: 42.0
angley: 220.0
distance: 0.0
---
anglex: 49.0
angley: 222.0
distance: 0.0
---
```

3.2 Dynamic Parameter Adjustment

In the terminal, enter:

```
ros2 run rqt_reconfigure rqt_reconfigure
```



use_sim_time ☐
Kp 20.0
Ki 0.0
Kd 2.5

(System message might be shown here when necessary)

☑ After modifying the parameters, click a blank space in the GUI to write the parameter value. Note that this will only take effect for the current boot. To permanently take effect, you need to modify the parameters in the source code.

As shown in the above image,

- gestureTracker is primarily responsible for servo gimbal movement. It can adjust PID-related parameters to achieve optimal gimbal movement.

🛠 Parameter Analysis:

[Kp], [Ki], [Kd]: Speed PID control for the servo gimbal during following.

4. Core Code

4.1. gestureTracker.py

This program has the following main functions:

- Initialize the gesture detector
- Open the camera and capture an image
- Detect gesture key points in the image
- Calculate the gesture center coordinates and publish them
- Use the PID algorithm to calculate the servo angle and issue control commands

Some of the core code is as follows:

```
# Create a publisher to publish the center coordinates of the tracked object
self.pub_position = self.create_publisher(Position, "/Current_point", 10)
# Define the servo data publisher
self.pub_Servo = self.create_publisher(ServoControl, 'Servo', 10)
...
# Initialize the gesture detector
self.hand_detector = handDetector(detectorCon=0.8)
...
# Detect gesture key points
self.hand_detector.findHands(frame, draw=False)
if len(self.hand_detector.lmList) != 0:
    x,y = self.hand_detector.findPoint(9)  # Get the index finger base joint
coordinates
    if self.hand_detector.get_gesture() != "Zero":  # Only gestures other than
"0" are tracked
```

```
        threading.Thread(target=self.execute, args=(x, y)).start()
...
# According to the x value and y value, use the PID algorithm to calculate the
servo angle
def execute(self, point_x, point_y):
    position = Position()
    position.anglex = point_x * 1.0
    position.angley = point_y * 1.0
    # Publish center coordinates message
    self.pub_position.publish(position)
    ...
    # Limit PID polarity and maximum value
    [x_Pid, y_Pid] = self.PID_controller.update([point_x - 320, point_y - 240])
    x_Pid = x_Pid * (abs(x_Pid) <= self.Kp/2)
    y_Pid = y_Pid * (abs(y_Pid) <= self.Kp/2)
    ...
    # Publish the calculated servo angle
    self.pub_Servo.publish(self.servo_angle)
```