Human body posture recognition and tracking

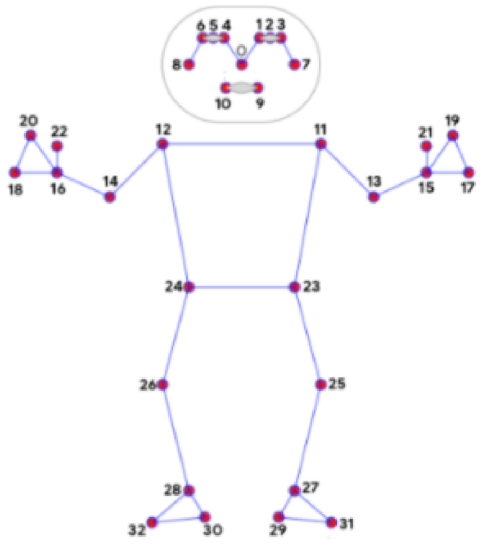
Note: The virtual machine, ROS-wifi image transmission module and ESP32 communication board ROS_DOMAIN_ID need to be consistent, and both must be set to 20. You can view [ESP32 communication board parameter configuration] to set the ESP32 communication board ROS_DOMAIN_ID, and view the tutorial [Connecting MicroROS Agent] to determine whether the ID is consistent.

Before running the experiment, please make sure that the microros balance car and ROS-wifi image transmission module have correctly enabled the agent on the virtual machine (linux with humbleROS2 system)

1. Introduction

MediaPipe Pose is an ML solution for high-fidelity body pose tracking. It uses BlazePose research to infer 33 3D coordinates and full-body background segmentation masks from RGB video frames. This research also provides momentum for the ML Kit pose detection API.

The landmark model in MediaPipe pose predicts the location of 33 pose coordinates (see the figure below).



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

2. Program Description

After the program is started, the camera captures the image, and the car will follow the left and right movement of the human body in the picture. The movement speed of the human body should not be too fast, and it is recommended to use image tracking, otherwise the image processing cannot keep up, which will cause lag.

2.1, Program code reference path

The location of the function source code is,

```
/home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/pose_fllo
w.py
```

3, Program startup

3.1, Start command

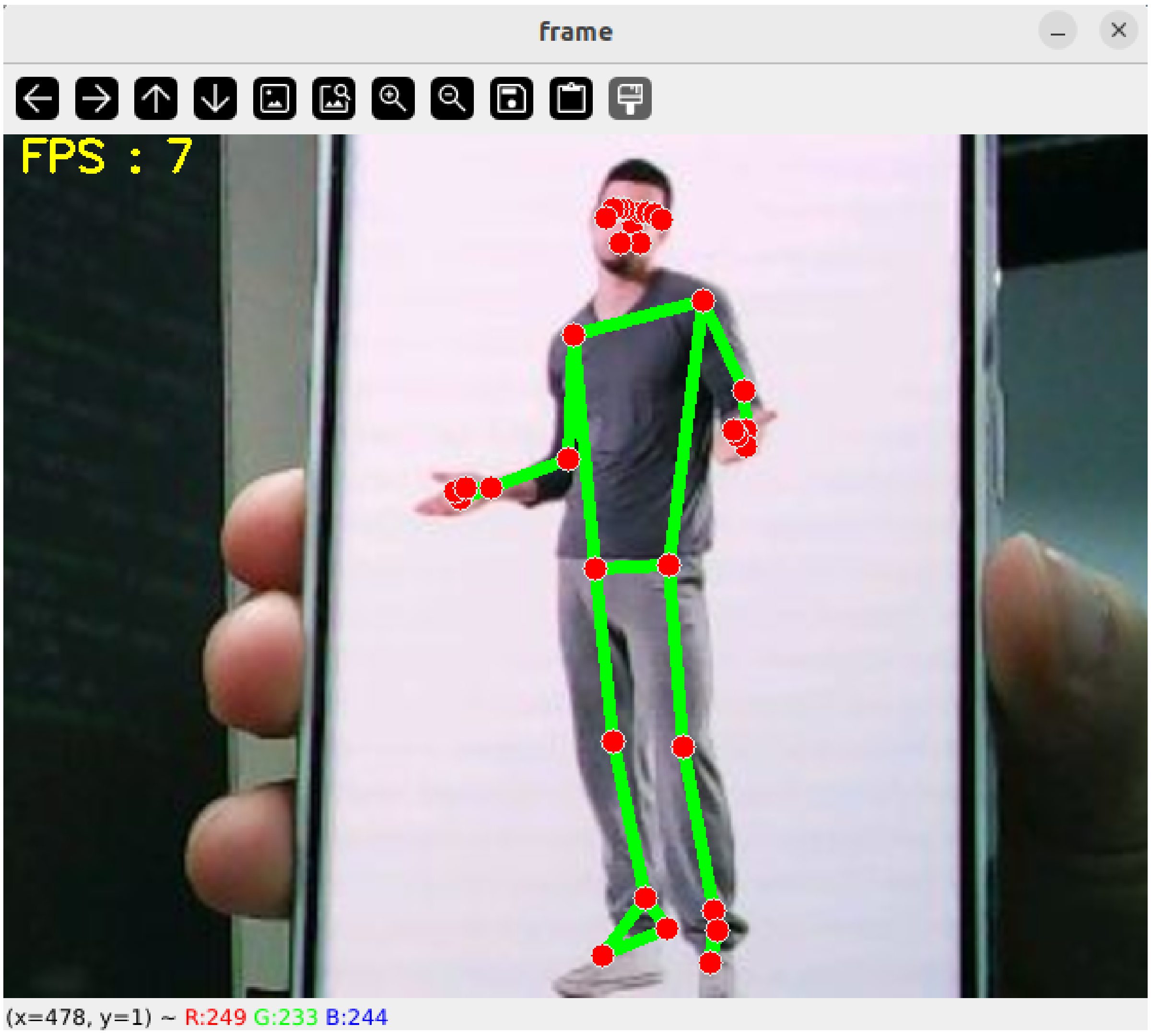
Input in the terminal,

```
# Start chassis driver
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

```
# Start the human posture recognition and tracking program
ros2 run yahboom_esp32ai_car pose_fllow
```

If the camera image is inverted, you need to see **3. Camera image correction (select)** document to correct it yourself, and this experiment will not be described.

After the function is turned on, the car will track the left and right movement of the human body.



3.2, parameter adjustment

```
#angular PID
self.declare_parameter("angular_Kp",45.0)
self.angular_Kp =
self.get_parameter('angular_Kp').get_parameter_value().double_value
self.declare_parameter("angular_Ki",0.0)
self.angular_Ki =
self.get_parameter('angular_Ki').get_parameter_value().double_value
self.declare_parameter("angular_Kd",5.0)
self.angular_Kd =
self.get_parameter('angular_Kd').get_parameter_value().double_value
self.angular_PID = (self.angular_Kp, self.angular_Ki, self.angular_Kd)
```

Mainly adjust the three PID parameters to make the car tracking more sensitive, modify the source code, and then recompile and update.

4. Code analysis

4.1. pose_fllow.py

- Code reference location

  ```
  /home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/pose_
  fllow.py
  ```

- Code analysis

1. Import important library files

```
from media_library import *
```

2. Detect human body posture information

```
frame, pose_points = self.pose_detector.pubPosePoint(frame)

x_coords = [point[1] for point in pose_points if point[1] is not None]

if len(x_coords) > 0:
    center_x = sum(x_coords) / len(x_coords)
    threading.Thread(target=self.hand_threading, args=(center_x,)).start()
else:
    self.media_ros.pub_vel(0.0, 0.0, 0.0)
#frame is the image after posture detection, pose_points is the list of human
posture key points detected in the image, center_x is the average x coordinate of
all key points x_coords
```

3. Some core codes are as follows:

```python
#point_x value based on calculated angular velocity
[z_Pid, _] = self.PID_controller.update([(point_x - 320)/12, 0])

if point_x > 320 and z_Pid > 0:
    self.angularz = -z_Pid
else:
    self.angularz = +z_Pid
if abs(point_x-320) < 60:
    self.angularz = 0.0
self.linearx = 0.0
if point_x == 0:self.angularz = 0.0
print("z_Pid:{:.2f}, point_x:{:.2f}".format(z_Pid,point_x))
self.media_ros.pub_vel(self.linearx, 0.0,self.angularz)
```