Palm control of car movement

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)

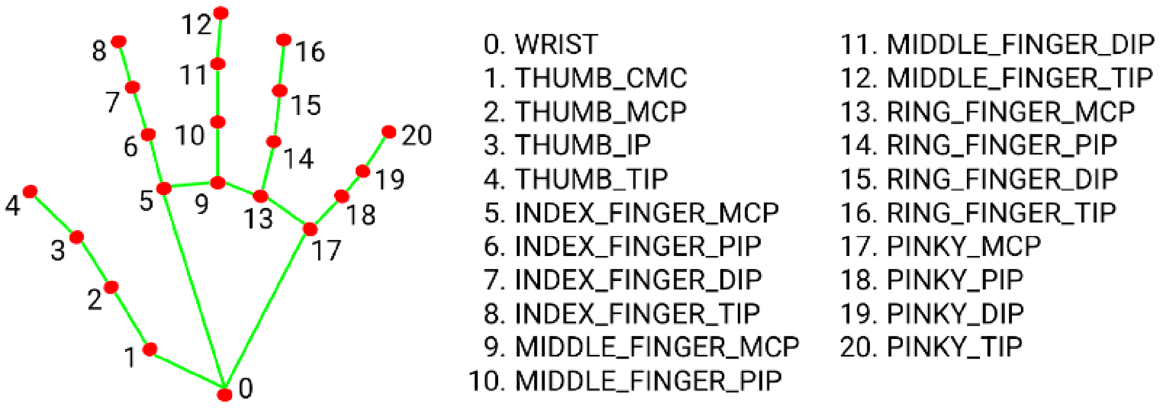
1. Introduction to mediapipe

MediaPipe is an open source data stream processing machine learning application development framework developed by Google. It is a graph-based data processing pipeline used to build data sources in various 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 media.

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It uses machine learning (ML) to infer 21 3D hand coordinates from a single frame.

After palm detection for the entire image, accurate keypoint localization of 21 3D hand joint coordinates within the detected hand region is performed by regression based on the hand landmark model, i.e. direct coordinate prediction. The model learns a consistent internal hand pose representation that is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, ~30K real-world images were manually annotated with the 21 3D coordinates as shown below (Z values were taken from the image depth map, if available for each corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the properties of the hand geometry, high-quality synthetic hand models were also drawn against various backgrounds and mapped to the corresponding 3D coordinates.



2. Program Description

This section may run very slowly on the robot main control. You can test the robot after recognizing the palm. This will have a better effect.

The robot will control the movement of the chassis according to the position of the palm in the picture.

Palm on the top of the screen->Car moves forward

Palm on the bottom of the screen->Car moves backward

Palm on the left of the screen->Car moves left

Palm on the bottom of the screen->Car moves right

2.1. Source code path

The source code of this function is located at,

```
/home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/RobotCtrl
.py
```

3. Program startup

3.1. Start command

Terminal input,

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

```
ros2 run yahboom_esp32ai_car RobotCtrl
```

If the camera image is inverted, you need to see **3. Camera Image Correction (Select)** document to correct it yourself. This experiment will not be described.

Turn on this function, then put your hand in front of the camera, the screen will recognize the palm, and after the program recognizes the position of the palm, it will send the speed to the chassis, thereby controlling the movement of the car.

4. Core code

4.1. RobotCtrl.py

- Code reference location

```
/home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/RobotCtrl
.py
```

- Code analysis

1. import the corresponding library file

```
from media_library import *
```

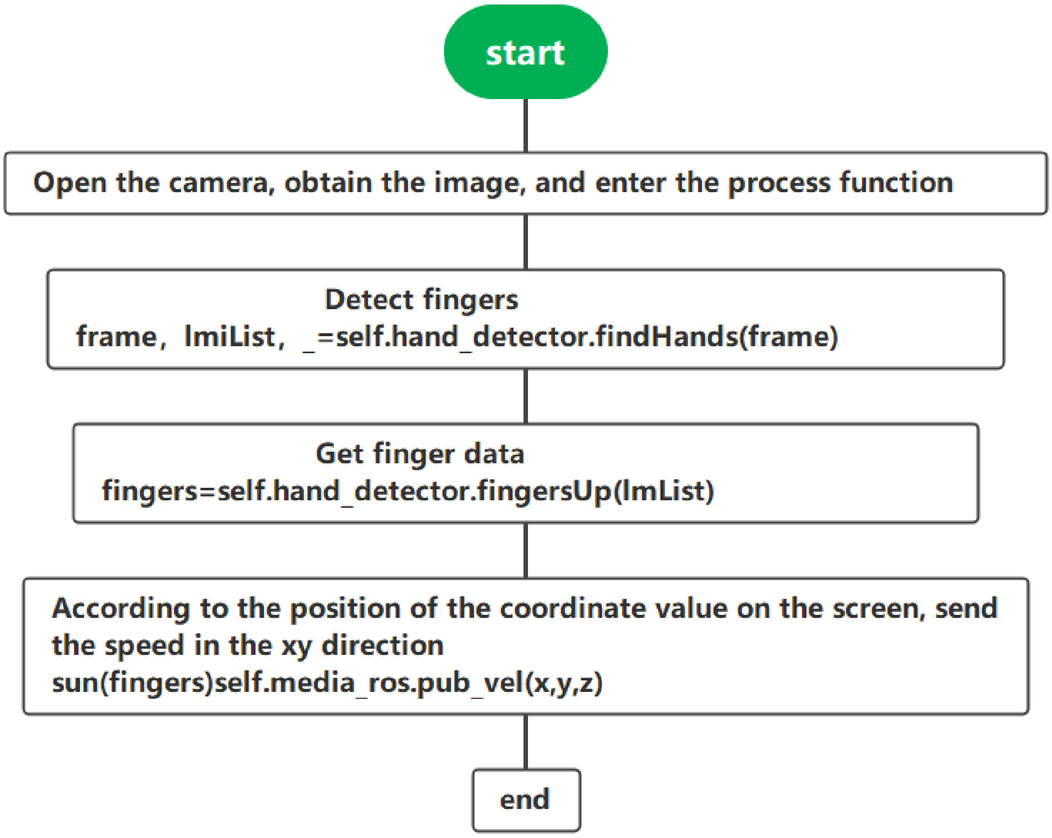
This library file mainly includes detecting palms, fingers and obtaining the coordinates of each finger joint.

2. Detect the palm and obtain the coordinates of the fingers

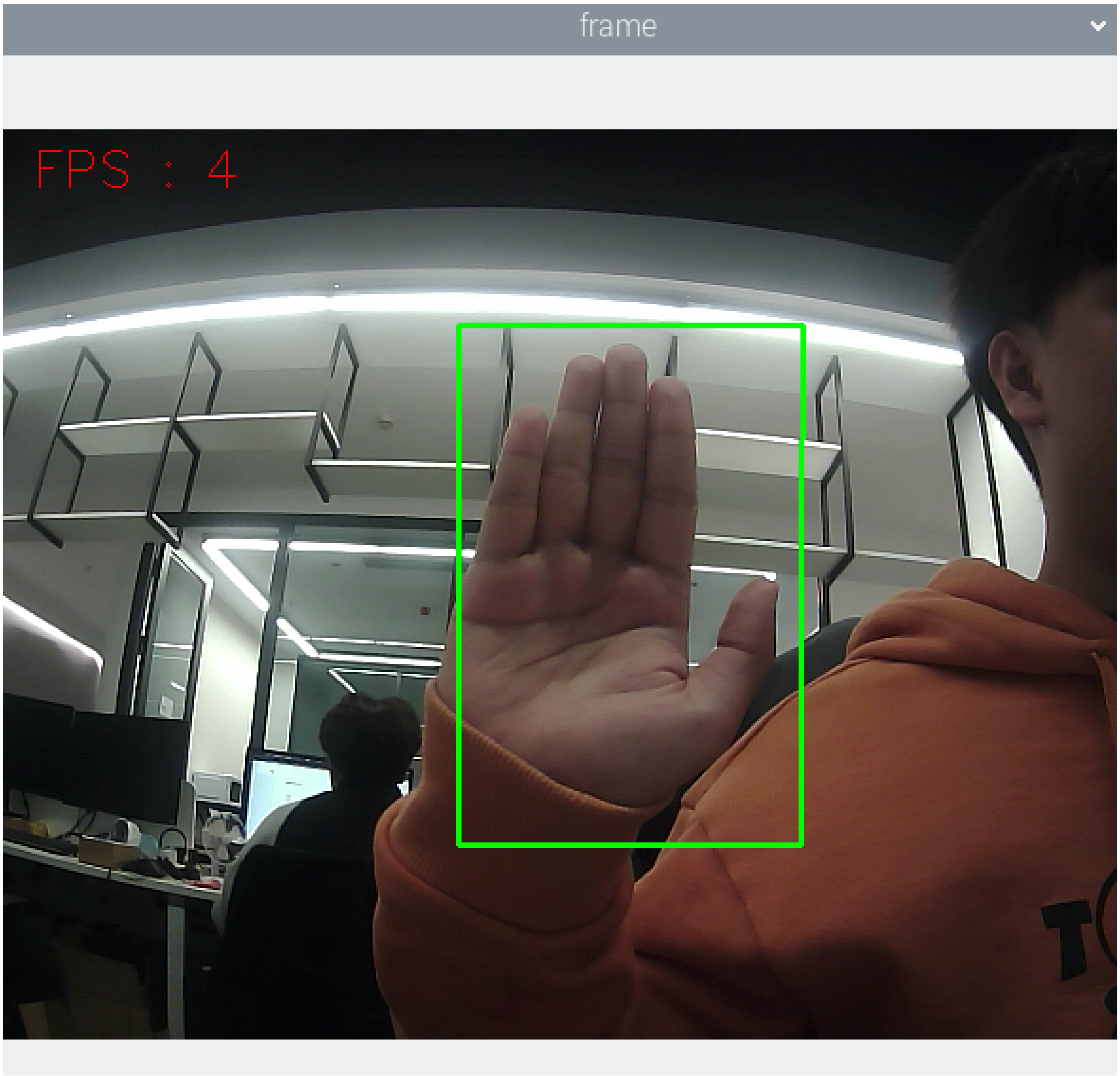
```
fingers = self.hand_detector.fingersUp(lmList)
point_x = lmList[9][1] #x value
point_y = lmList[9][2] #Y value
```

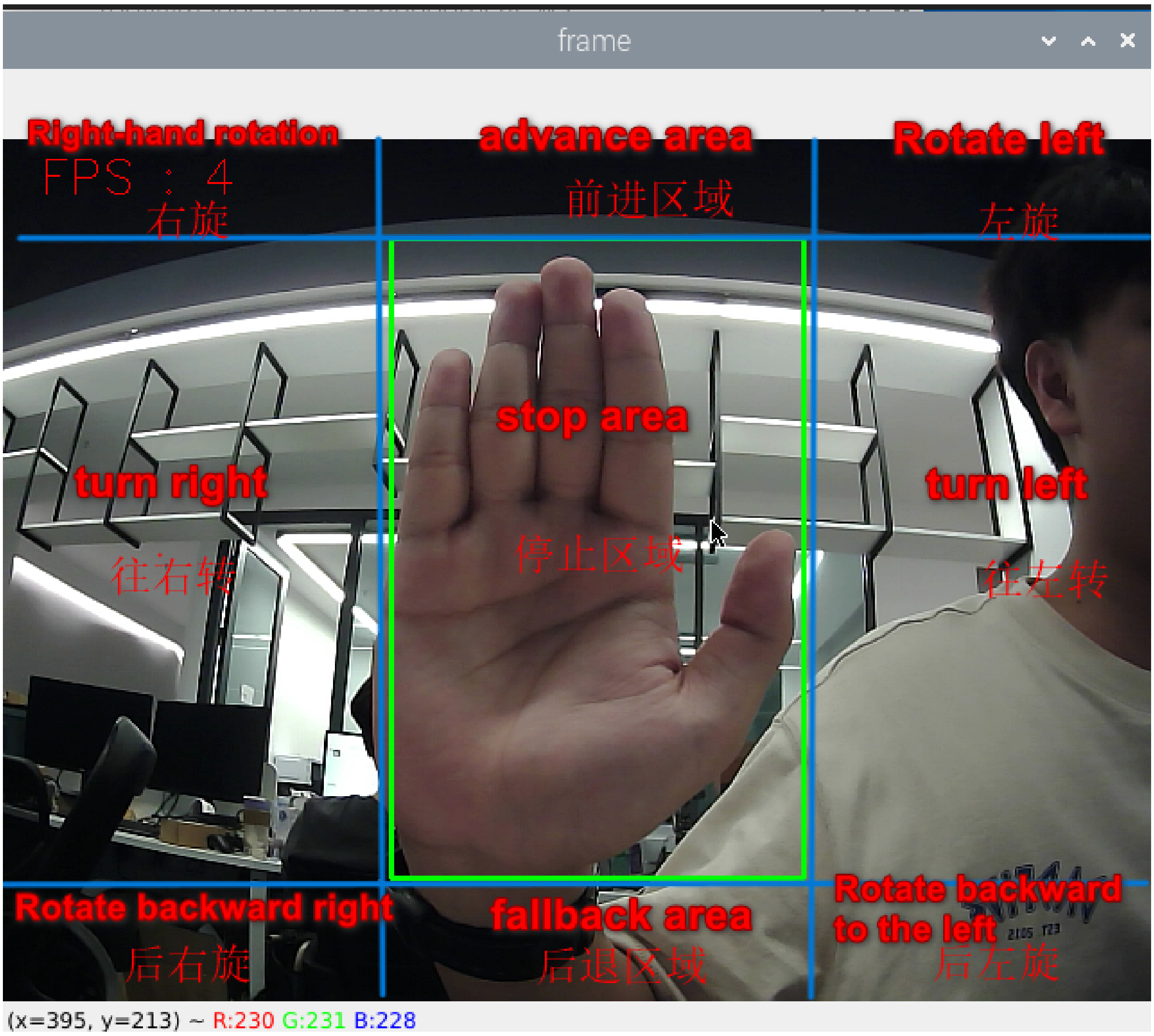
Combining the picture in 1. Introduction, we can know that what we get is actually the coordinates of the first joint of the middle finger of our palm. By judging the position of this coordinate in the picture and sending it to the speed in the xy direction of the chassis, control can be achieved.

4.2. Flowchart


start
Open the camera, obtain the image, and enter the process function
Detect fingers
frame，lmiList，_=self.hand_detector.findHands(frame)
Get finger data
fingers=self.hand_detector.fingersUp(lmList)
According to the position of the coordinate value on the screen, send the speed in the xy direction
sun(fingers)self.media_ros.pub_vel(x,y,z)
end

4.3. Result chart





```
(x=395, y=213) ~ R:230 G:231 B:228
```