6. Object tracking

6.1 Program function description

After the program starts, the camera image is displayed on the web page. Select the object to be tracked through the mouse frame, press the [space] key, and the car will enter the tracking mode. The car will maintain a distance of 1 meter from the tracked object, and always ensure that the tracked object remains in the center of the picture.

In addition, the [L1] button on the handle locks/turns on the car's motion controls. When motion control is enabled, the function is locked; This function can be turned on when the motion control is locked.

6.2. Code Reference Paths

After SSH connection car, the location of the function source code is located at,

```
#launch文件#launch file
/userdata/yahboomcar_ws/src/yahboomcar_kcftracker/launch/kcftracker_launch.xml
```

The node or launch file is explained as follows:

- KCF_Tracker_Node: mainly completes image processing and calculates the center coordinates of the tracked object. The code path is,

```
/userdata/yahboomcar_ws/src/yahboomcar_kcftracker/src/KCF_Tracker.cpp
```

- astra.launch.xml: starts the Astra camera
- yahboomcar_bringup_launch. py: starts the chassis
- rosbridge_websocket_launch.xml: enables the websocket to interact with the Web
- rosboard_node: starts the ROSboard

6.3. Program Startup

6.3.1. App Launch and page Display

After SSH connects to the car, terminal input,

```
ros2 launch yahboomcar_kcftracker kcftracker_launch.xml
```

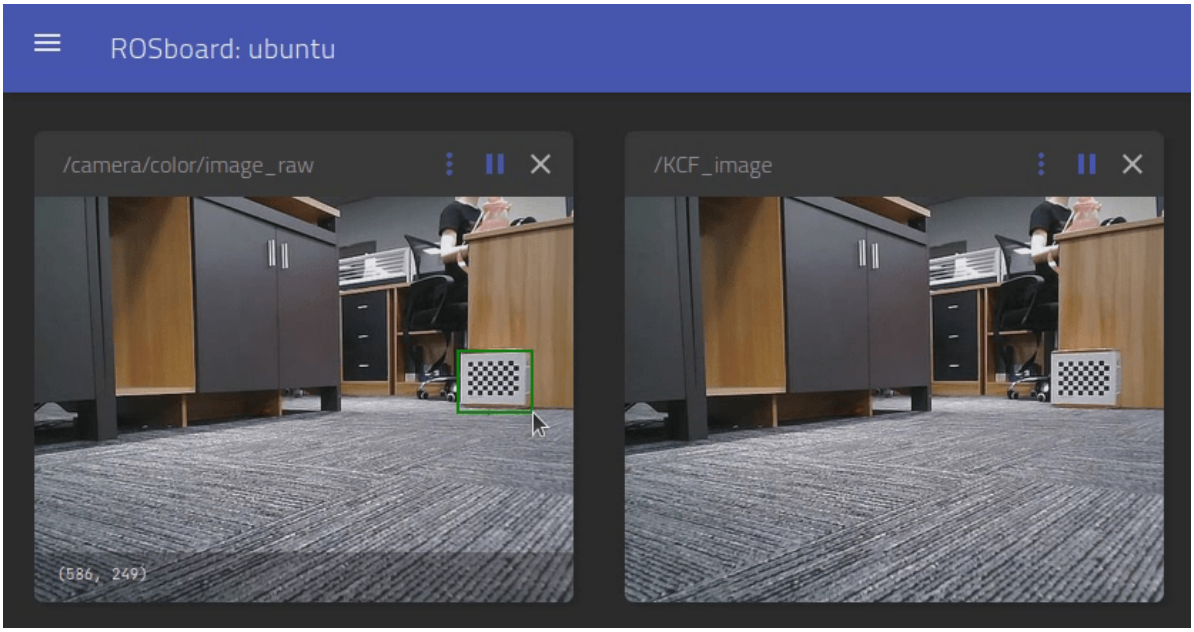
Open the browser on the PC side (note that the computer and the Rising Sun network must be in the same LAN), enter the URL: car IP:8888, for example, my car IP is 192.168.2.67, enter the URL in the browser on the virtual machine side to open the ROSboard webpage:

```
192.168.2.67:8888
```

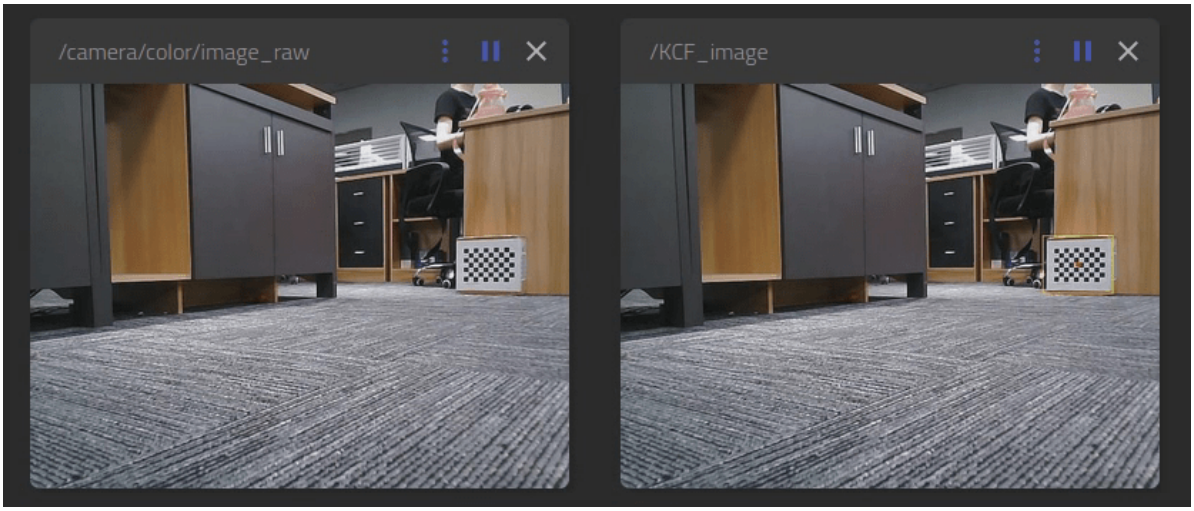
Select /camera/color/image_raw and /KCF_image respectively.

6.3.2 Object Tracking

Press the [i] key in the /camera/color/image_raw image to enter the recognition mode, click the mouse box twice to select the object to be tracked.



When released, the target is selected, and the center of the target is displayed in /KCF_image.



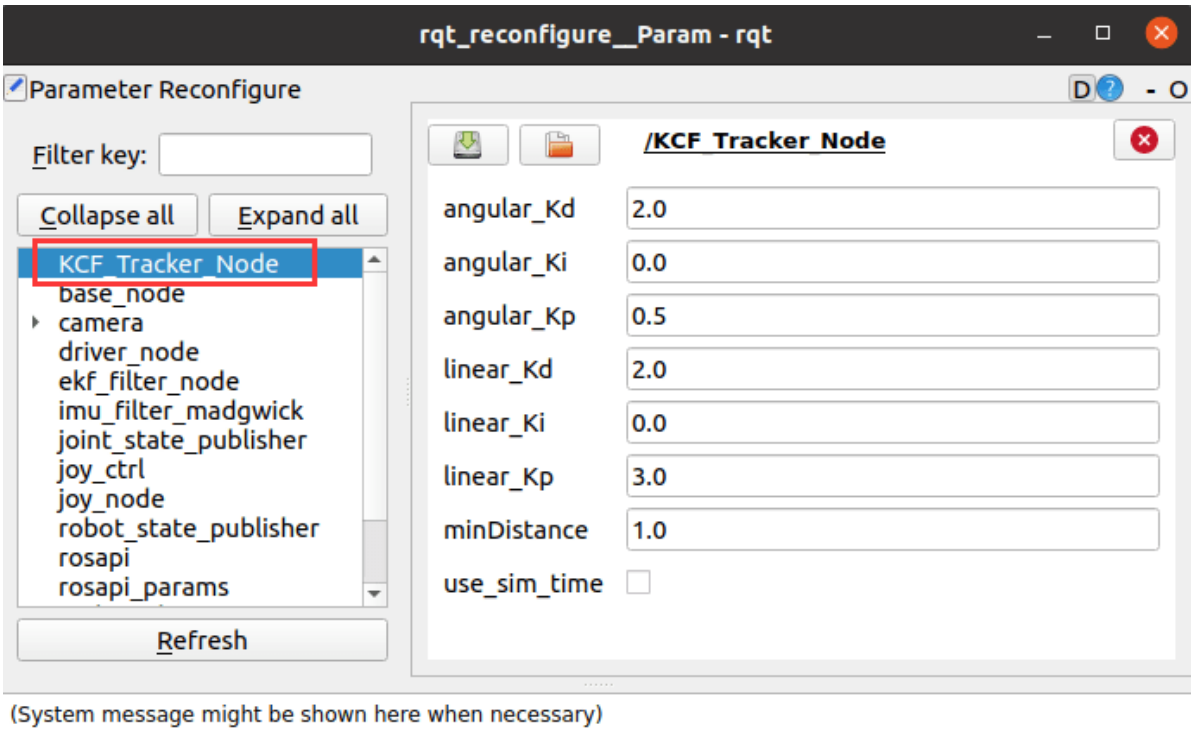
Then press the [space] key to start tracking, and the car will move to the position 1 meter away from the target to stop; Pressing [q] will stop tracking; Press the [r] keyboard to clear the selected target, then press the [i] key to re-select the object to track.

Move the object slowly, the car will follow and keep a distance of 1 meter from the object.

6.3.3 Dynamic parameter tuning

Virtual machine terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



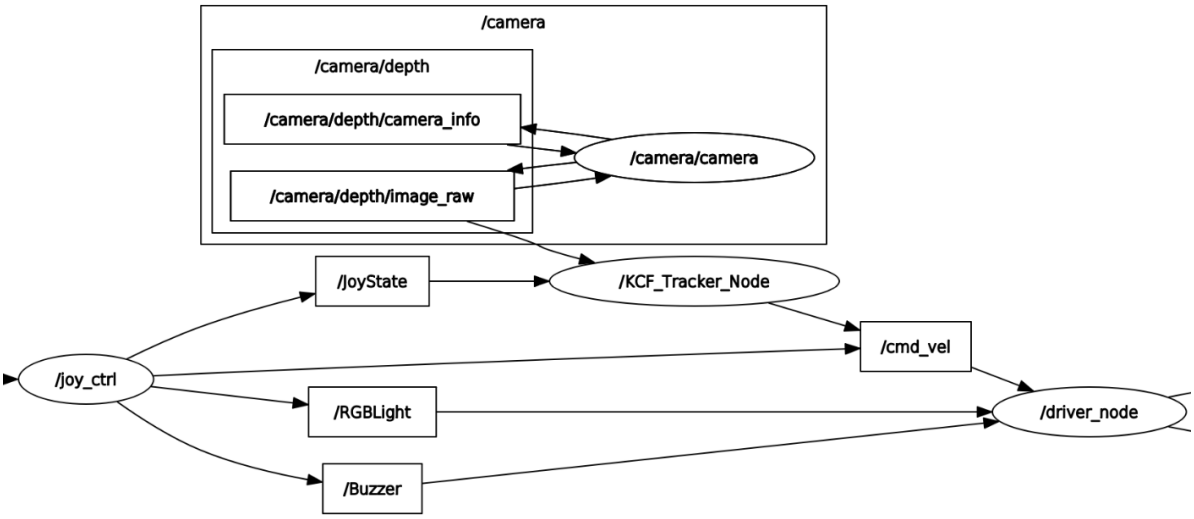
The adjustable parameters are the PID of the linear speed of the car, the angular speed, and the tracking distance.

After modifying the parameters, press Enter or click the blank area of the GUI to write the parameter values.

6.3.4. View node topic communication graph

Open the VM terminal and run the following command to view topic communication between nodes:

```
ros2 run rqt_graph rqt_graph
```



6.4. Core Source Code Parsing

The principle of functional implementation is similar to color tracking, which calculates linear and angular speed according to the central coordinates of the target and the depth information fed by the depth camera, and then releases it to the chassis. Part of the code in KCF_Tracker.cpp is as follows:

```
//选择物体后，得出中心坐标，用于计算角速度
// After selecting the object, the center coordinates are obtained, which are
used to calculate the angular velocity
if (bBeginKCF) {
    result = tracker.update(rgbimage);
    rectangle(rgbimage, result, Scalar(0, 255, 255), 1, 8);
    circle(rgbimage, Point(result.x + result.width / 2, result.y + result.height
/ 2), 3, Scalar(0, 0, 255),-1);
}
else rectangle(rgbimage, selectRect, Scalar(255, 0, 0), 2, 8, 0);

//计算出center_x和distance的值，用于计算速度
// Calculate the values of center_x and distance for speed calculation
int center_x = (int)(result.x + result.width / 2);
int num_depth_points = 5;
for (int i = 0; i < 5; i++) {
    if (dist_val[i] > 40 && dist_val[i] < 8000) distance += dist_val[i];
    else num_depth_points--;
}
if (num_depth_points == 0) distance = this->minDist;
else distance /= num_depth_points;

//计算线速度和角速度
// Calculate linear and angular velocities
linear_speed = -linear_PID->compute(this->minDist, distance);
rotation_speed = angular_PID->compute(320, center_x) / 1000.0;
```