6. Object tracking

6.1. Description of programme functions

After the programme starts, the camera image is displayed on the web side. Select the object to be tracked by mouse box, press [space] key, the cart enters the tracking mode. The cart will keep a distance of 1m from the tracked object and always ensure that the tracked object stays in the centre of the screen.

In addition, the [L1] key of the joystick can lock/open the motion control of the cart. When motion control is on, this function will be locked; when motion control is locked, this function can be turned on.

6.2, Code Reference Path

After SSH connecting the cart, the location of the source code of this function is located at.

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

Where the node or launch file that is launched is explained as follows, the

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

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

- astra.launch.xml: launch Astra camera
- yahboomcar_bringup_launch.py: launch chassis
- rosbridge_websocket_launch.xml: launch websocket, implement ROS and Web interaction
- rosboard_node: launch ROSboard

6.3 Starting the programme

6.3.1 Starting the programme and displaying the web page

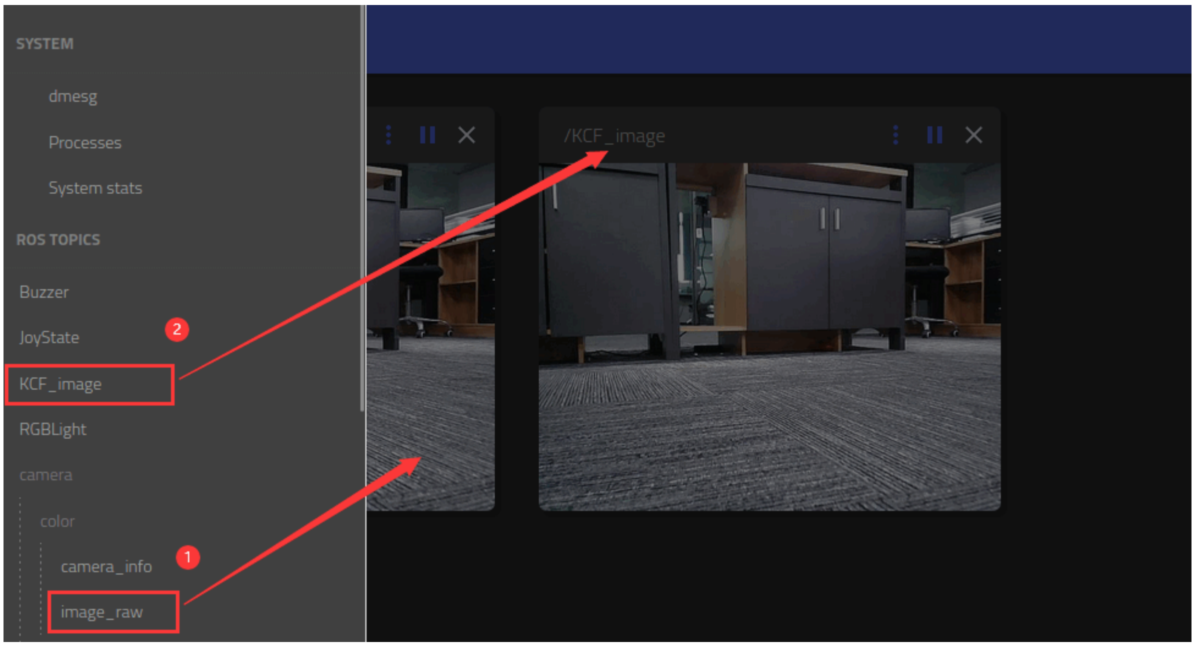
After SSH connection to the car, terminal input.

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

Open the browser on the PC side (note that the computer and the RDK-X3 network must be on the same LAN), enter the URL: cart IP:9999, for example, my cart IP is 192.168.2.67, and enter the URL in the browser on the VM side to open the ROSboard web side:

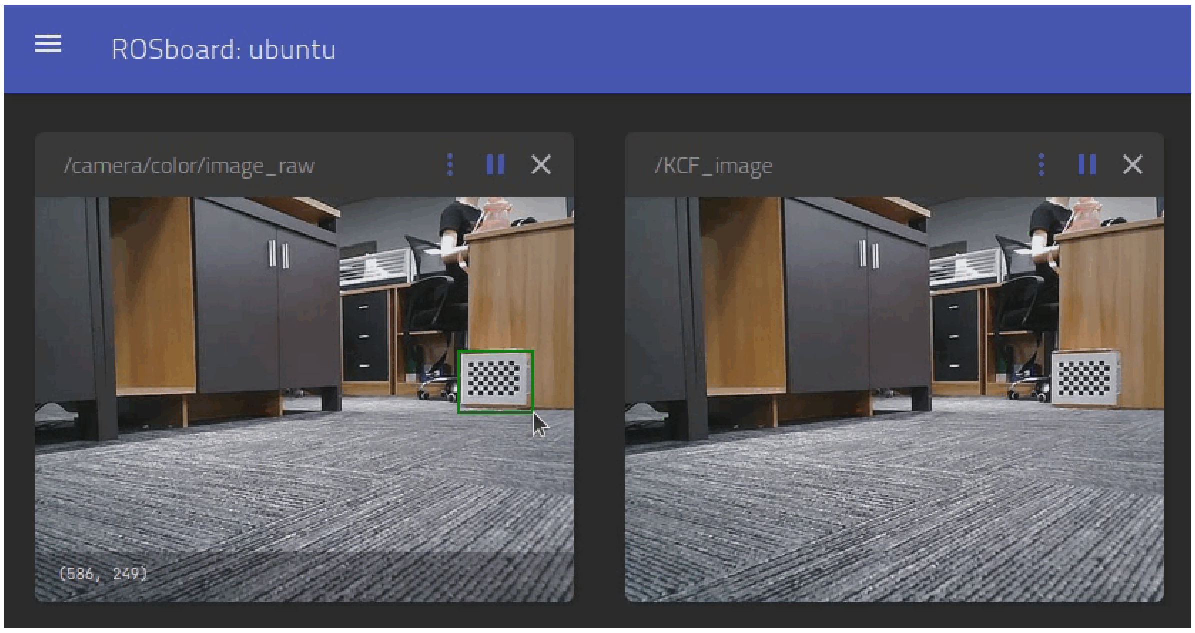
```
192.168.2.67:9999
```

Select the topics /camera/colour/image_raw and /KCF_image respectively.

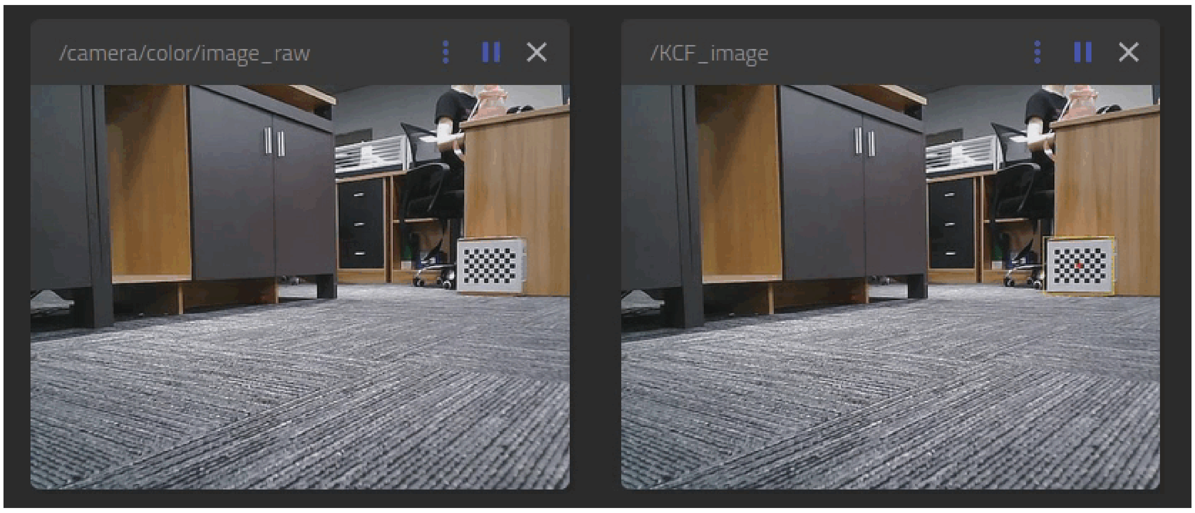


6.3.2 Object Tracking

In the image of /camera/colour/image_raw press the [i] key to enter the recognition mode, click the mouse twice to frame the object to be tracked, the



Release means that the target is selected and the position of the target centre point is displayed in /KCF_image.



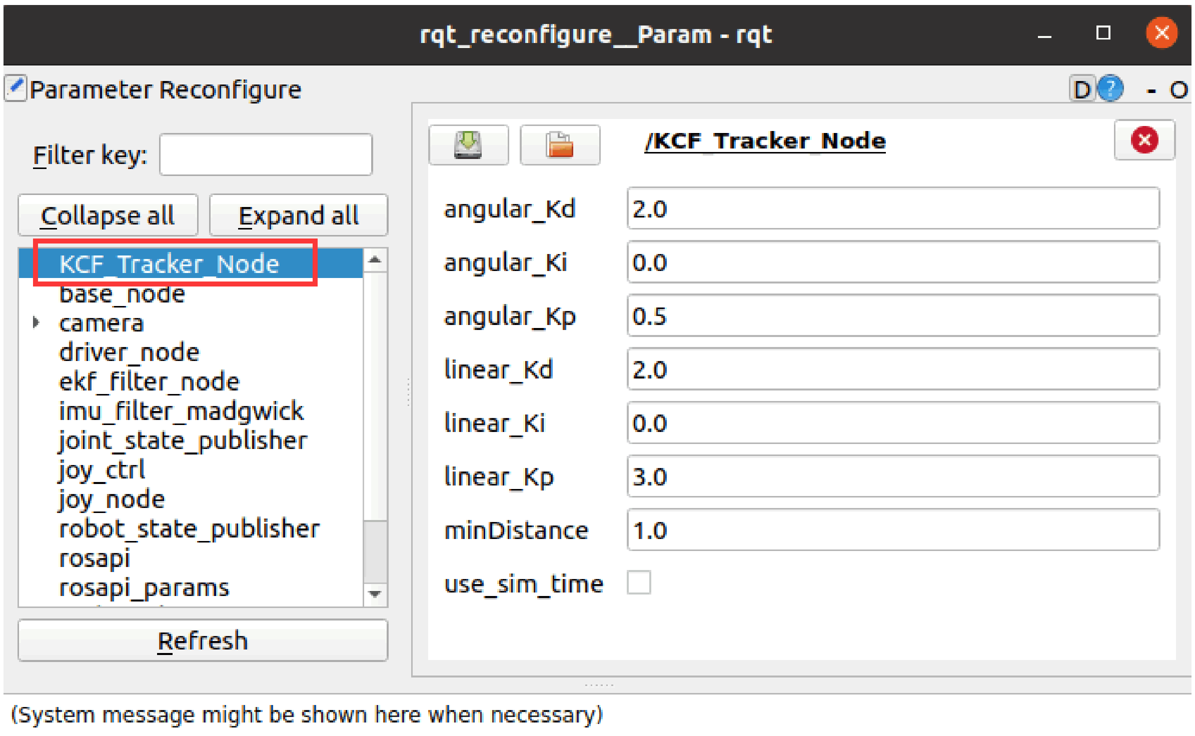
Then press [Space] to start tracking, the cart will move to a position 1 metre away from the target and stop; press [q] to stop tracking; press [r] to clear the selected target, and then press [i] to reselect the tracked object.

Move the object slowly, the trolley will follow and keep a distance of 1 metre from the object.

6.3.3 Dynamic Parameter Adjustment

VM terminal input.

```
ros2 run rqt_reconfigure rqt_reconfigure
```



KCF_Tracker_Node
base_node
camera
driver_node
ekf_filter_node
imu_filter_madgwick
joint_state_publisher
joy_ctrl
joy_node
robot_state_publisher
rosapi
rosapi_params

/KCF_Tracker_Node

angular_Kd	2.0
angular_Ki	0.0
angular_Kp	0.5
linear_Kd	2.0
linear_Ki	0.0
linear_Kp	3.0
minDistance	1.0
use_sim_time	☐

(System message might be shown here when necessary)

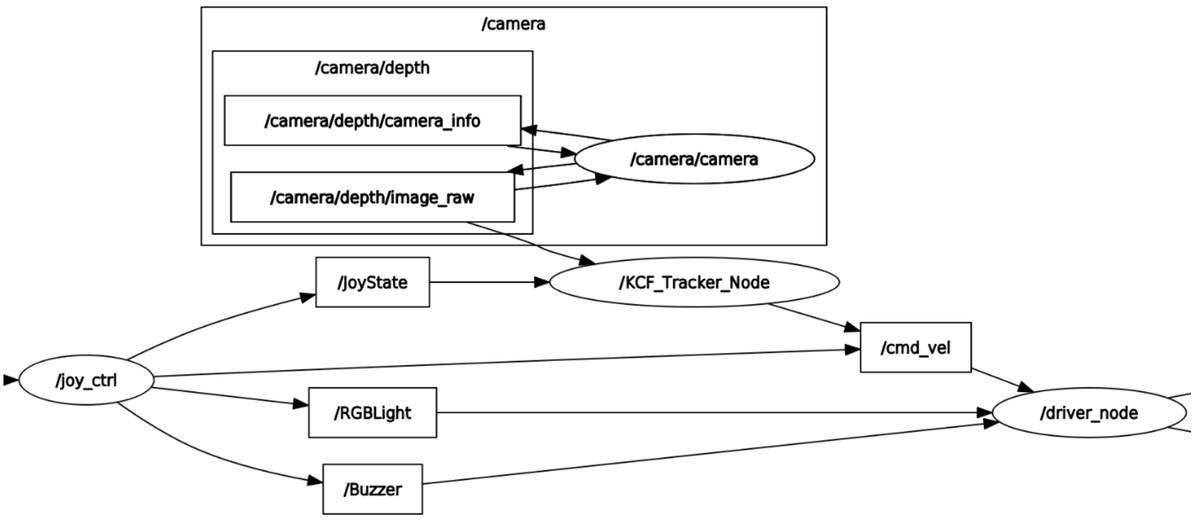
The parameters that can be adjusted are the PID of the trolley line speed, angular speed and the distance to be tracked.

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

6.3.4 Viewing Node Topic Communication Map

Open the VM terminal and view the topic communication between nodes by the following command.

```
ros2 run rqt_graph rqt_graph
```



6.4, Core Source Code Analysis

The principle of function implementation is almost the same as colour tracking, it is based on the target's centre coordinate and the depth information fed by the depth camera to calculate the linear and angular velocities, and then release 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 centre coordinates are derived and 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 centre_x and distance for calculating velocity
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;
```