4、Lidar follow

Function package: ~/rplidar_ws/src/transbot_laser

Introduction of lidar follow:

- Set the detection angle and distance of the lidar.
- After turning on the car, the car will follow the target closest to the car and keep a certain distance.
- When there are obstacles behind the trolley, the buzzer keeps beeping and stops moving backwards until there are no obstacles.
- The PID of the linear speed and angular velocity of the trolley can be adjusted to make the car follow the best effect.

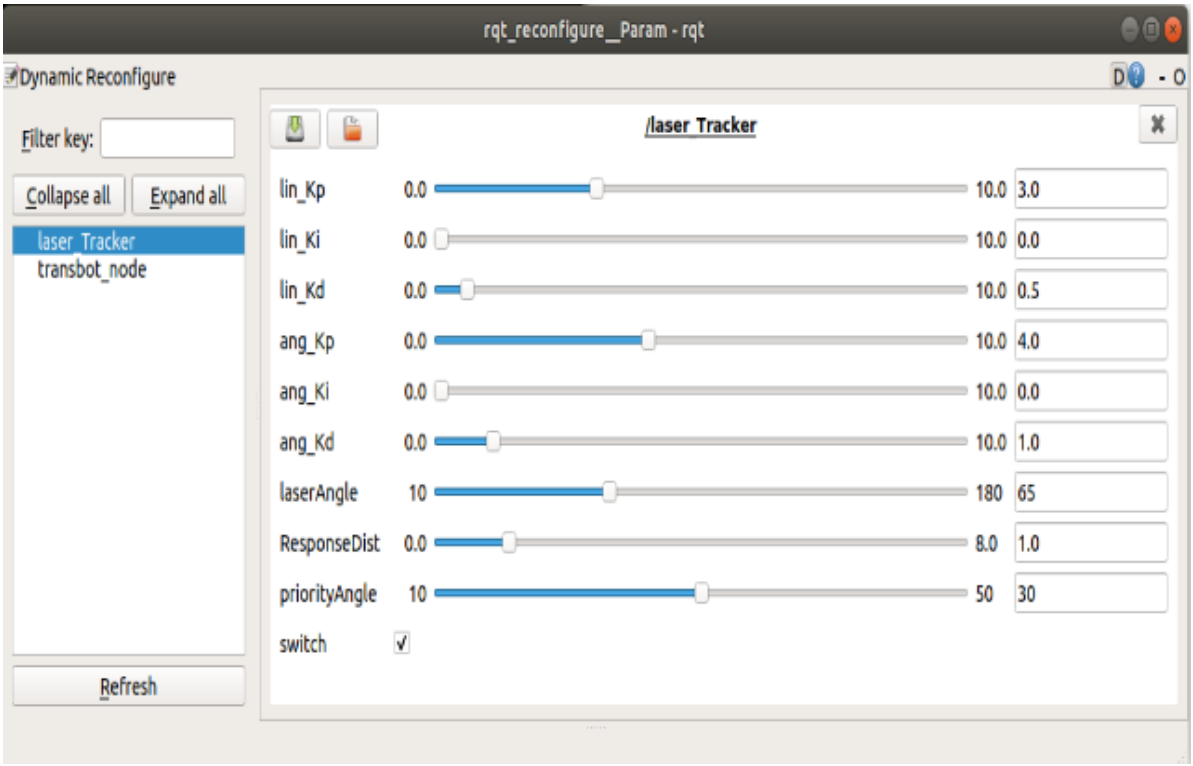
4.1、Instructions

Start up

```
roslaunch transbot_laser laser_Tracker.launch
```

Dynamic debugging parameters

```
rosrun rqt_reconfigure rqt_reconfigure
```



Parameter analysis:

Parameter	Range	Analysis
【LaserAngle】	【10，180】	Lidar detection angle (angle of left and right side)
【ResponseDist】	【0.0，8.0】	Robot follow distance
【priorityAngle】	【10，50】	The car prioritizes the following range (angle of left and right side)
【switch】	【False，True】	Robot movement【start/pause】

【lin_Kp】、【lin_Ki】、【lin_Kd】：PID debugging of car linear speed.

【ang_Kp】、【ang_Ki】、【ang_Kd】：PID debugging of car angular speed.

【switch】Click the box in front of [switch], the value of [switch] is True, and the car will stop. [Switch] The default is False, and the car moves.

【priorityAngle】cannot be smaller than【laserAngle】.

View node

```
rqt_graph
```

4.2、Source code analysis

launch file

- base.launch

```
<launch>
    <!-- Start the lidar node -->
    <include file="$(find ydlidar_ros_driver)/launch/X2.launch"/>
    <!-- Dynamic debugging tool node -->
<!--    <node pkg="rqt_reconfigure" type="rqt_reconfigure" name="rqt_reconfigure"
output="screen"/>-->
    <!-- Start the car chassis drive node-->
    <node pkg="transbot_bringup" type="transbot_driver.py" name="transbot_node"
required="true" output="screen">
        <param name="imu" value="/transbot/imu"/>
        <param name="vel" value="/transbot/get_vel"/>
    </node>
    <!-- Handle control node -->
    <include file="$(find transbot_ctrl)/launch/transbot_joy.launch"/>
</launch>
```

- laser_Avoidance.launch

```
<launch>
    <!-- Start the base.launch文件 -->
    <include file="$(find transbot_laser)/launch/base.launch">
    <!-- Start the laser_Tracker node -->
    <node name='laser_Tracker' pkg="transbot_laser" type="laser_Tracker.py"
required="true" output="screen"/>
</launch>
```

py code：~/rplidar_ws/src/transbot_laser/scripts/laser_Tracker.py

```
front_state = False
back_state = True
offset = 0.5
... ...
# If the lidar scans a circle, there are 720 IDs
if len(np.array(scan_data.ranges)) == 720:
    for i in range(270, 450):
        # Check whether there are target behind
        if ranges[i] < 0.5: back_state = False
    for i in range(0, self.priorityAngle * 2):
        # Check whether there are target front left
        if ranges[i] < (self.ResponseDist + offset): front_state = True
    for i in range(720 - self.priorityAngle * 2, 720):
        # Check whether there are target front right
        if ranges[i] < (self.ResponseDist + offset): front_state = True
    if front_state == True:
        # When there are target ahead
        angle_min = self.priorityAngle * 2
        angle_max = 720 - self.priorityAngle * 2
        # Get target ID and minimum distance
        minDistID, self.minDist = self.Get_ID_minDist(angle_min, angle_max,
ranges)
    else:
        # When there are no target ahead
        angle_min = self.laserAngle * 2
        angle_max = 720 - self.laserAngle * 2
        # Get obstacles ID and minimum distance
        minDistID, self.minDist = self.Get_ID_minDist(angle_min, angle_max,
ranges)
```

Source code parameter analysis:

Parameter	Defaults value	Judgment
front_state	False	When it is True, it means there is a target ahead
back_state	True	When it is False, it means that the rear is not passable
offset	0.5	The priority is [ResponseDist] + [offset]