4、雷达跟随

功能包路径：~/rplidar_ws/src/transbot_laser

雷达跟随玩法介绍：

- 设定激光雷达检测角度和距离。
- 开启小车后，小车追随距离小车最近的目标，并保持一定距离。
- 当小车的后面有障碍物时，蜂鸣器一直响并停止后退，直到没有障碍物。
- 可调节小车线速度和角速度的PID，使得小车跟随效果最佳。

4.1、使用方法

注意：遥控手柄的【R2】具备所有玩法的【暂停/开启】的功能。

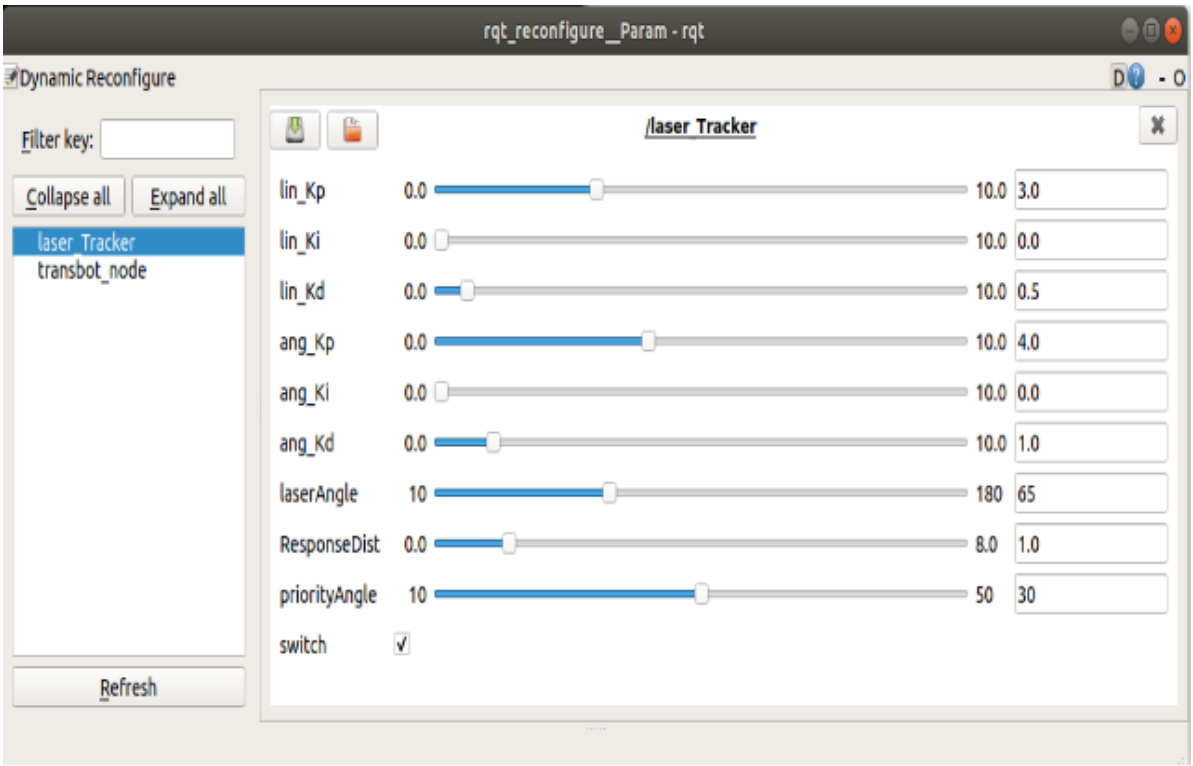
一键启动

```
roslaunch transbot_laser laser_Tracker.launch lidar_type:=a1
```

lidar_type参数：使用激光雷达的型号：[a1,a2,a3,s1,s2]。

动态调试参数

```
rosrun rqt_reconfigure rqt_reconfigure
```



参数解析：

参数	范围	解析
【laserAngle】	【10，180】	激光雷达检测角度（左右一侧角度）
【ResponseDist】	【0.0，8.0】	小车跟随距离
【priorityAngle】	【10，50】	小车优先考虑跟随范围（左右一侧角度）
【switch】	【False，True】	小车开始暂停

【lin_Kp】、【lin_Ki】、【lin_Kd】：小车线速度PID调试。

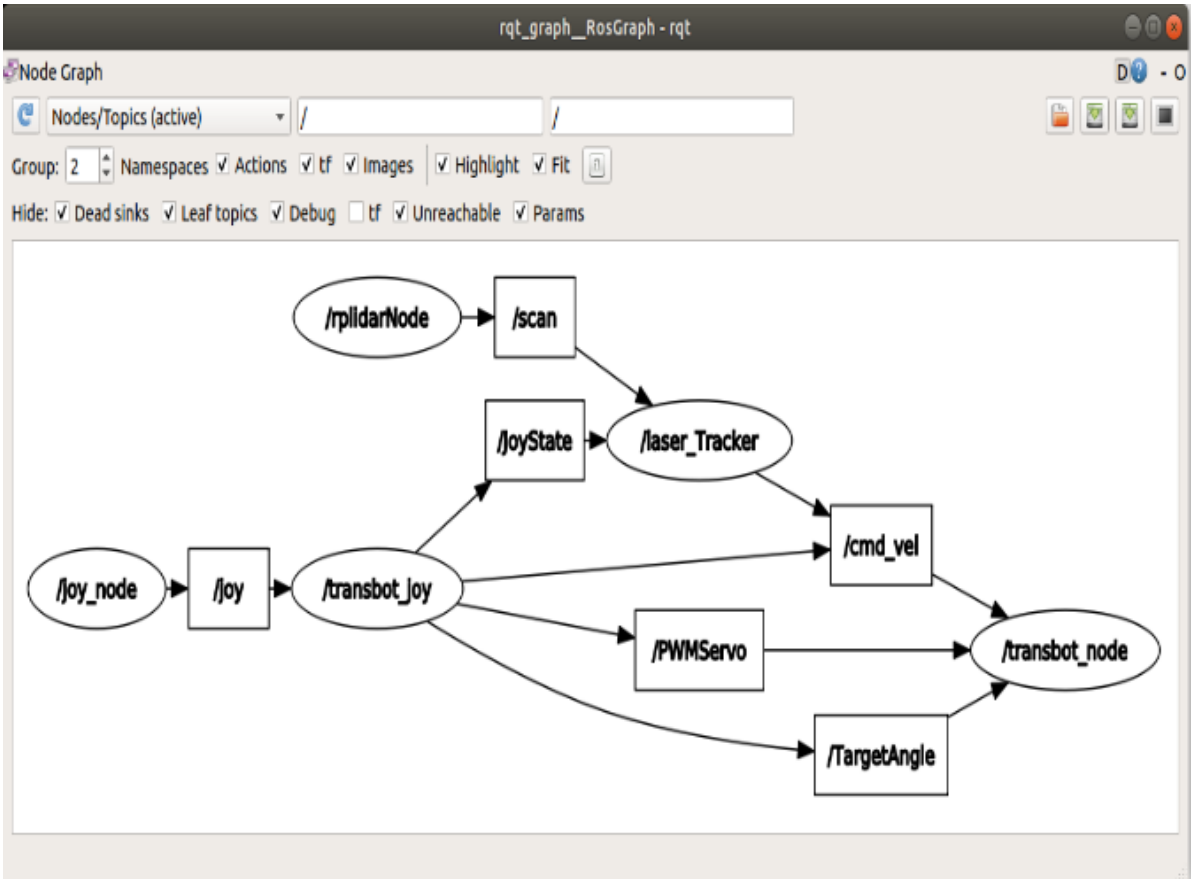
【ang_Kp】、【ang_Ki】、【ang_Kd】：小车角速度PID调试。

【switch】前面的方框，点击【switch】的值为True，小车停止。【switch】默认为False，小车运动。

参数【priorityAngle】不能比【laserAngle】大，否则没意义。

节点查看

```
rqt_graph
```



4.2、源码解析

launch文件

- base.launch

```
<launch>
    <!-- 启动激光雷达节点-->
    <arg name="lidar_type" default="a1" doc="lidar_type type [a1,a2,a3,s1,s2]"/>
    <include file="$(find rplidar_ros)/launch/rplidar.launch">
        <arg name="lidar_type" value="$(arg lidar_type)"/>
```

```
    </include>
    <!-- 动态调试工具节点-->
<!--    <node pkg="rqt_reconfigure" type="rqt_reconfigure" name="rqt_reconfigure"
output="screen"/>-->
    <!-- 启动小车底盘驱动节点-->
    <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>
    <!-- 手柄控制节点 -->
    <include file="$(find transbot_ctrl)/launch/transbot_joy.launch"/>
</launch>
```

- laser_Avoidance.launch

```
<launch>
    <!-- 启动base.launch文件 -->
    <arg name="lidar_type" default="a1" doc="lidar_type type [a1,a2,a3,s1,s2]"/>
    <include file="$(find transbot_laser)/launch/base.launch">
        <arg name="lidar_type" value="$(arg lidar_type)"/>
    </include>
    <!-- 启动激光雷达跟随节点 -->
    <node name='laser_Tracker' pkg="transbot_laser" type="laser_Tracker.py"
required="true" output="screen"/>
</launch>
```

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

```
front_state = False
back_state = True
offset = 0.5
... ...
# 如果激光雷达扫描一圈有720个ID
if len(np.array(scan_data.ranges)) == 720:
    for i in range(270, 450):
        # 后方是否有障碍物
        if ranges[i] < 0.5: back_state = False
    for i in range(0, self.priorityAngle * 2):
        # 左前方是否有目标
        if ranges[i] < (self.ResponseDist + offset): front_state = True
    for i in range(720 - self.priorityAngle * 2, 720):
        # 右前方是否有目标
        if ranges[i] < (self.ResponseDist + offset): front_state = True
    if front_state == True:
        # 前方有目标的情况
        angle_min = self.priorityAngle * 2
        angle_max = 720 - self.priorityAngle * 2
        # 获取目标ID和最小距离
        minDistID, self.minDist = self.Get_ID_minDist(angle_min, angle_max,
ranges)
    else:
        # 前方没有目标的情况
        angle_min = self.laserAngle * 2
        angle_max = 720 - self.laserAngle * 2
        # 获取目标ID和最小距离
```

```
        minDistID, self.minDist = self.Get_ID_minDist(angle_min, angle_max,
ranges)
```

源码参数解析：

参数	默认值	判断
front_state	默认为False	当为True时，说明正前方有目标。
back_state	默认为True	当为False时，说明后方不可通行
offset	默认为0.5	优先考虑正前方的距离是【ResponseDist】+【offset】