4.Lidar guard

The following workspace contains the entire rplidar_ws function package.

If you need to transplant it to your own main development board, you need to copy all the function packages to the src of the workspace for compilation, and install the corresponding environment.

Note: This course uses Rosmaster-X3 as an example. Users need to modify it according to their own motion model. The course only explains the implementation method.

Code path: ~/rplidar_ws/src/yahboomcar_laser

Lidar guard fucntion:

- Set lidar detection angle and response distance
- After turning on the car, the car faces the target closest to the car
- When the distance between the target and the car is less than the response distance, the buzzer will sound until there is no target within the response distance.
- Adjustable car angular speed PID to achieve the best car rotation effect

4.1 Start

Input following command:

```
roslaunch yahboomcar_laser laser_Warning.launch
```

Input following command to debug dynamic parameters

```
rosrun rqt_reconfigure rqt_reconfigure
```

View the node graph

```
rqt_graph
```

4.2 Source code analysis

4.2.1 launch file

laser_Warning.launch

```
<launch>
    <!-- 启动base.launch文件 -->
    <!-- Launch the base.launch file -->
    <include file="$(find yahboomcar_laser)/launch/base.launch"/>
    <!-- 启动激光雷达警卫节点 -->
    <!-- Activate the Lidar guard node -->
    <node name='laser_Warning' pkg="yahboomcar_laser" type="laser_Warning.py"
required="true" output="screen"/>
</launch>
```

The base.launch file is to start the car chassis and lidar, mainly look at laser_Avoidance.py file.

Code path：~/oradar_ws/src/yahboomcar_laser/scripts

The core code part is as follows:

```
    def registerScan(self, scan_data):
        if not isinstance(scan_data, LaserScan): return
        # 记录激光扫描并发布最近物体的位置（或指向某点）
        # Record the laser scan and publish the position of the nearest object
(or point to a point)
        ranges = np.array(scan_data.ranges)
        # 创建距离列表，将检测范围内的有效距离放入列表中
        # create distance list, put the effective distance within the detection
range into the list
        minDistList = []
        # 创建序列号，将有效距离对应的ID放入列表中
        # Create a serial number and place the ID corresponding to the valid
distance in the list
        minDistIDList = []
        # 按距离排序以检查从较近的点到较远的点是否是真实的东西
        # if we already have a last scan to compare to:
        for i in range(len(ranges)):
            angle = (scan_data.angle_min + scan_data.angle_increment * i) *
RAD2DEG
            # if angle > 90: print "i: {},angle: {},dist: {}".format(i, angle,
scan_data.ranges[i])
            # 通过清除不需要的扇区的数据来保留有效的数据
            if 270 - self.LaserAngle < angle < 270 + self.LaserAngle:
                minDistList.append(ranges[i])
                minDistIDList.append(angle)
        if len(minDistList) == 0: return
        # 找到最小距离
        # Find the minimum distance
        minDist = min(minDistList)
        # 找到最小距离对应的ID
        # Find the ID corresponding to the minimum distance
        minDistID = minDistIDList[minDistList.index(minDist)]
        if self.ros_ctrl.Joy_active or self.switch == True:
            if self.Moving == True:
                self.ros_ctrl.pub_vel.publish(Twist())
                self.Moving = not self.Moving
            return
        self.Moving = True
        if minDist <= self.ResponseDist:
```

```
            if self.Buzzer_state == False:
                b = Bool()
                b.data = True
                self.pub_Buzzer.publish(b)
                self.Buzzer_state = True
        else:
            if self.Buzzer_state == True:
                self.pub_Buzzer.publish(Bool())
                self.Buzzer_state = False
        velocity = Twist()
        # 使用PID算法使得小车稳步移动到对应位置
        # The PID algorithm is used to make the car move to the corresponding
position steadily
        ang_pid_compute = self.ang_pid.pid_compute((180 - abs(minDistID)) / 36,
0)
        if minDistID > 0: velocity.angular.z = -ang_pid_compute
        else: velocity.angular.z = ang_pid_compute
        if ang_pid_compute < 0.02: velocity.angular.z = 0
        self.ros_ctrl.pub_vel.publish(velocity)
```

This part mainly involves entering the callback function after receiving the radar information. According to the value range of the radar data we set, find the smallest data, then find the ID corresponding to the minimum distance, and determine the distance of the ID with the minimum distance.

If it is greater than the set response speed value, the data will be released and the buzzer will sound.

Otherwise, calculate the pid based on the ID of the minimum distance, and publish the speed to the bottom layer so that the car moves with the identified object.