Lidar avoiding

This chapter needs to be used with a car chassis and other sensors to operate. This is just an explanation of the implementation method. In fact, it cannot be run. It needs to be used with the company's rosmaster-X3 car to achieve this function.

If you need to transplant it to your own motherboard, you need to install other dependency files.

1. Function Description

After the program is started, the car will move forward. When an obstacle appears within the detection range, it will adjust its attitude to avoid the obstacle, and then continue to move forward.

2. Code path

```
~/oradar_ws/src/yahboomcar_laser/yahboomcar_laser/laser_avoidance.py
```

3. Start up

Input following command:

```
ros2 launch yahboomcar_laser laser_avoidance_launch.py
```

4. Core code analysis

laser_avoidance_launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node

import os
from ament_index_python.packages import get_package_share_directory
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():
    lidar_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
         get_package_share_directory('rplidar_ros'), 'launch'),
         '/lidar.launch.py'])
      )

    laser_avoidance_node = Node(
        package='yahboomcar_laser',
        executable='laser_avoidance',
        output = 'screen'
    )

    bringup_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
        get_package_share_directory('yahboomcar_bringup'), 'launch'),
        '/yahboomcar_bringup_launch.py'])
    )

    launch_description = LaunchDescription([
        lidar_node,
        laser_avoidance_node,
        bringup_node
        ])
    return launch_description
```

It mainly starts the radar lidar_node, chassis bringup_node and obstacle avoidance function node laser_avoidance_node. Mainly look at the laser_avoidance.py file.

laser_avoidance.py

Mainly look at the radar callback function. Here is an explanation of how to obtain the obstacle distance information at each angle.

```
ranges = np.array(scan_data.ranges)
for i in range(len(ranges)):
    if ranges[i] < self.ResponseDist:
        angle = (scan_data.angle_min + scan_data.angle_increment * i) * 180 / pi
        if angle > 180: angle = angle - 360
        if -self.LaserAngle < angle < -20:
            self.right_warning += 1
        elif abs(angle) <= 20:
            self.front_warning += 1
        elif 20 < angle < self.LaserAngle:
            self.left_warning += 1
if self.Joy_active == True or self.Switch == False:
    if self.Moving == True:
        self.pub_vel.publish(Twist())
        self.Moving = not self.Moving
            return
self.Moving = True
```