1.雷达数据屏蔽需求

使用激光雷达建图和导航,实际应用中,不可能保证雷达在最上层(即车体不会遮挡雷达),一般应用为雷达放在下面,对于360°的雷达,车体就会遮挡从而导致车体被人作为障碍物,那如何解决这个问题

2.简单实现

我们只需要订阅激光雷达的数据,把相应角度的数据屏蔽即可,然后在重新发布新的数据

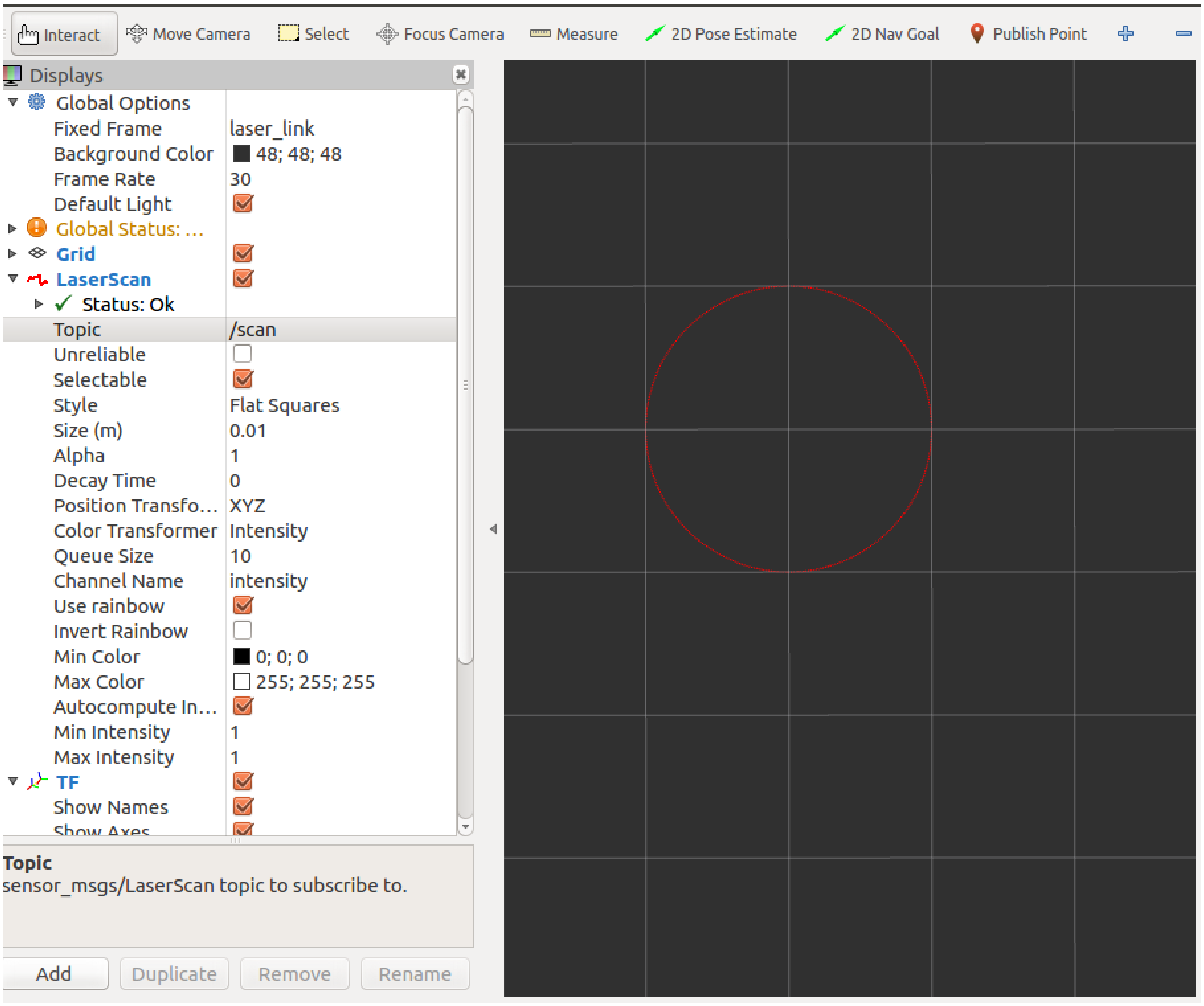
```
#!/usr/bin/env python

from __future__ import print_function
import sys
import rospy
from sensor_msgs.msg import LaserScan
from std_msgs.msg import String

class DoFilter:
    def __init__(self):

        self.sub = rospy.Subscriber("scan", LaserScan, self.callback)
        self.pub = rospy.Publisher("filteredscan", LaserScan, queue_size=10)

    def callback(self, data):
        newdata = data
        newdata.ranges = list(data.ranges)
        newdata.intensities = list(data.intensities)

        for x in range(120,130):
            newdata.ranges[x]=0
            newdata.intensities[x]=0

        self.pub.publish(newdata)
        rospy.loginfo(data)

if __name__ == '__main__':
    rospy.init_node('LidarFilter', anonymous=False)
    lidar = DoFilter()

    rospy.spin()
```

3.`laser_filters`包

3.1 LaserScanBoxFilter

先贴地址 laser_filters - ROS Wiki,改包提供了强大的功能,这里我们使用一个简单的例子看下如何屏蔽

我们先写个模拟的激光雷达,假设1m范围内为障碍物



Global Options
Fixed Frame laser_link
Background Color 48; 48; 48
Frame Rate 30
Default Light ✓
Global Status: ...
Grid ✓
LaserScan ✓
Status: Ok
Topic /scan
Unreliable
Selectable ✓
Style Flat Squares
Size (m) 0.01
Alpha 1
Decay Time 0
Position Transfo... XYZ
Color Transformer Intensity
Queue Size 10
Channel Name intensity
Use rainbow ✓
Invert Rainbow
Min Color 0; 0; 0
Max Color 255; 255; 255
Autocompute In... ✓
Min Intensity 1
Max Intensity 1
TF ✓
Show Names ✓
Show Axes ✓

Topic
sensor_msgs/LaserScan topic to subscribe to.

代码贴上如下,不在赘述

```
#!/usr/bin/env python

from __future__ import print_function
import sys
import rospy
from sensor_msgs.msg import LaserScan
from std_msgs.msg import String

class ScanSimulator:
    def __init__(self):

        self.pub = rospy.Publisher("scan", LaserScan, queue_size=10)
        self.newdata = LaserScan()
        self.newdata.ranges = [0]*360
        self.newdata.intensities = [0]*360
        self.newdata.header.frame_id="laser_link"

        self.newdata.angle_min=-1.57
        self.newdata.angle_max=1.57
        self.newdata.angle_increment=3.14*2 / 360
        self.newdata.time_increment=(1 / 100) / (360);
        self.newdata.range_min = 0.0;
        self.newdata.range_max = 100.0;

    def pub(self):
        self.newdata.header.stamp=rospy.Time.now()
        for x in range(0,360):
            self.newdata.ranges[x]=1
            self.newdata.intensities[x]=1

        self.pub.publish(self.newdata)

if __name__ == "__main__":
    rospy.init_node('scan_simulator',anonymous=True)

    r = rospy.Rate(100)
    lidar = ScanSimulator()

    while not rospy.is_shutdown():
        lidar.pub()
        r.sleep()
```

下面看下如何使用该包

`box_filter_example.launch`

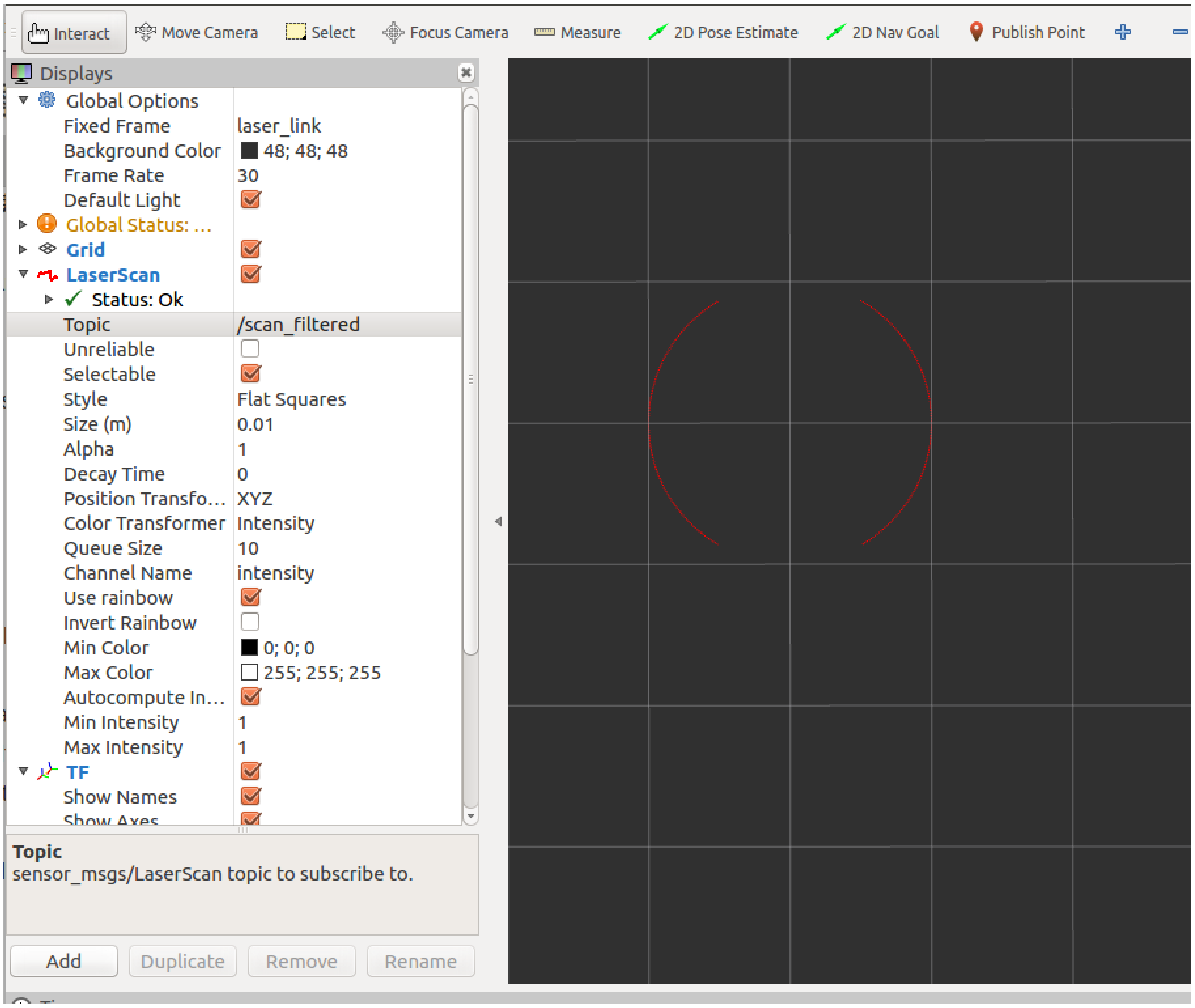
```
<launch>
<node pkg="laser_filters" type="scan_to_scan_filter_chain" output="screen"
name="laser_filter">
      <rosparam command="load" file="$(find pibot_bringup)/params/box_filter.yaml"
/>
</node>
</launch>
```

`box_filter.yaml`

```
scan_filter_chain:
- name: box_filter
  type: laser_filters/LaserScanBoxFilter
  params:
    box_frame: laser_link
    min_x: -1.0
    max_x: 1.0
    min_y: -0.5
    max_y: 0.5
```

```
      min_z: -0.1
      max_z: 0.1
```

使用该配置,我们屏蔽min_至max_ 形成的box里面的数据, 我们通过启动该`launch`后查看数据的filter_scan见下图



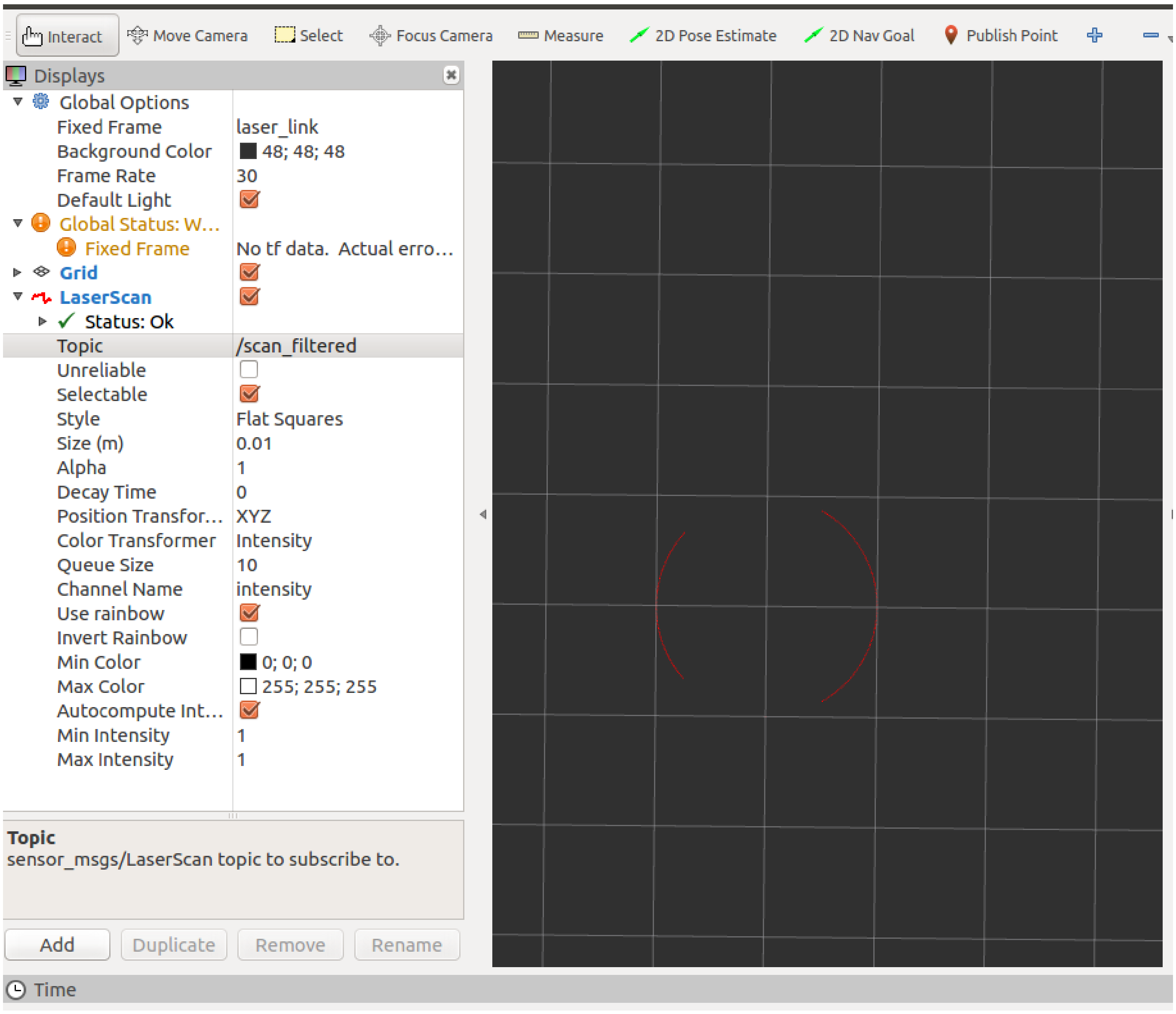
Global Options	
Fixed Frame	laser_link
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: ...	
Grid	✓
LaserScan	✓
✓ Status: Ok	
Topic	/scan_filtered
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0
Max Color	☐ 255; 255; 255
Autocompute In...	✓
Min Intensity	1
Max Intensity	1
TF	✓
Show Names	✓
Show Axes	✓

Topic
sensor_msgs/LaserScan topic to subscribe to.

`box_filter.yaml`

```
scan_filter_chain:
- name: box_filter
  type: laser_filters/LaserScanBoxFilter
  params:
    box_frame: laser_link
    min_x: -1.0
    max_x: 1.0
    min_y: -0.75
    max_y: 0.5
    min_z: -0.1
    max_z: 0.1
```

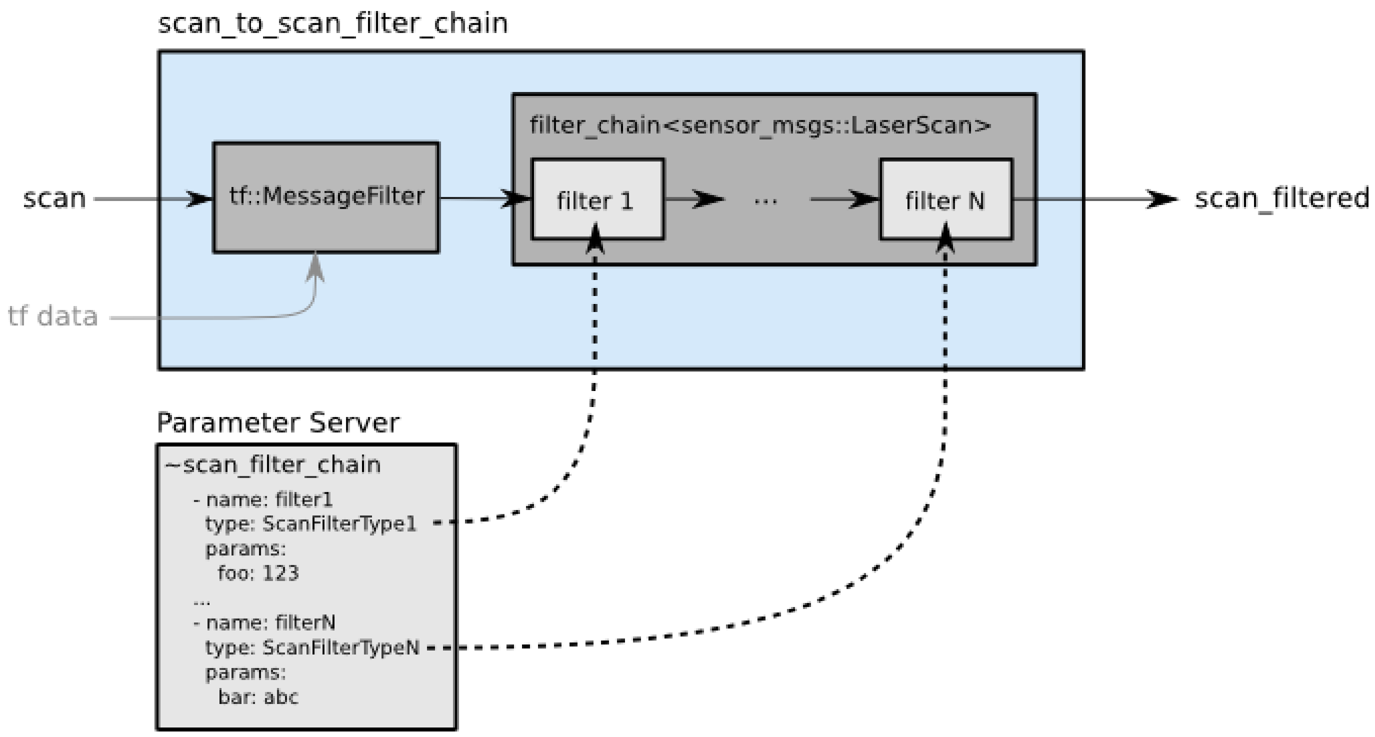
使用该配置效果如下图



Global Options	
Fixed Frame	laser_link
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: W…	
Fixed Frame	No tf data. Actual erro…
Grid	✓
LaserScan	✓
Status: Ok	
Topic	/scan_filtered
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfor…	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0
Max Color	□ 255; 255; 255
Autocompute Int…	✓
Min Intensity	1
Max Intensity	1

Topic
sensor_msgs/LaserScan topic to subscribe to.

3.2 scan_to_scan_filter_chain



可以看到该node为一个链,通过插件可以配置多层的过滤,可以配置下列滤波器

- LaserScanBoxFilter 无视一个区块内的数据（常用于无视机器人本体对激光雷达数据的干扰）
- ScanShadowsFilter 针对物体边沿的扫描和识别
- InterpolationFilter 插值滤波
- LaserScanIntensityFilter 设定强度阈值,超出则设置为nan
- LaserScanRangeFilter 设定距离阈值,超出则设置为nan
- LaserScanAngularBoundsFilter 将设定的角度外的扫描数据删除
- LaserScanAngularBoundsFilterInPlace 不会删除目标角度扇区外的数据，但会把对应扫描的距离值设为最大距离阈值+1
- LaserArrayFilter 使用中值过滤器等对距离和强度进行过滤