重庆大学
CHONGQING UNIVERSITY

机器人操作系统

Robot Operating System

——第四章：ROS传感器

江　涛、熊祖明、张博文

重庆大学　自动化学院

2023年5月23日

本章目标

1、熟悉机器人常用传感器，了解传感器使用场景，

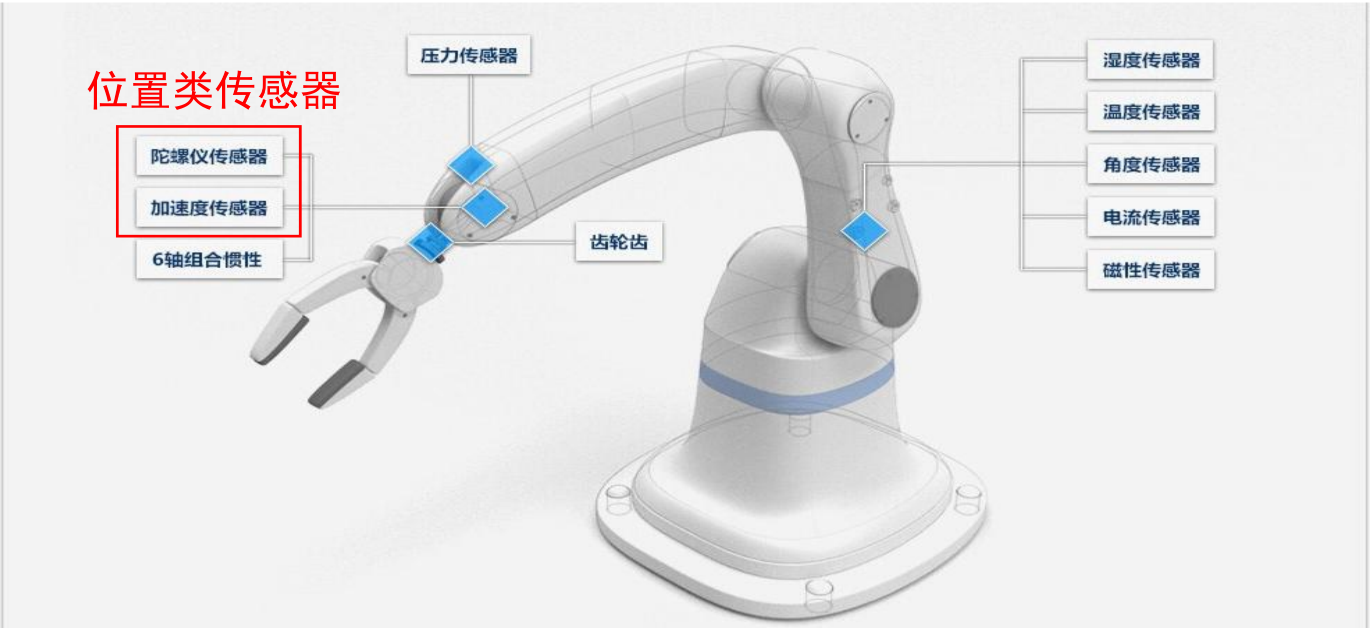
2、掌握传感器ROS中相关msg数据格式，如：激光、图像、点云等

3、掌握Gazebo中传感器模型添加、可视化方法

4、实战：传感器数据采集

一、传感器简介：概念

- 传感器是一种可以**感知**、**采集**被测物理量（温度、光、声等），并将其**转换**、**输出**为另一种物理量或信号（电压、电流等）的装置或系统。
- 在机器人和自动化控制系统中，传感器常常用于**感知环境**和**机器人本身的状态**，通过采集的数据提供给控制系统进行决策和控制。



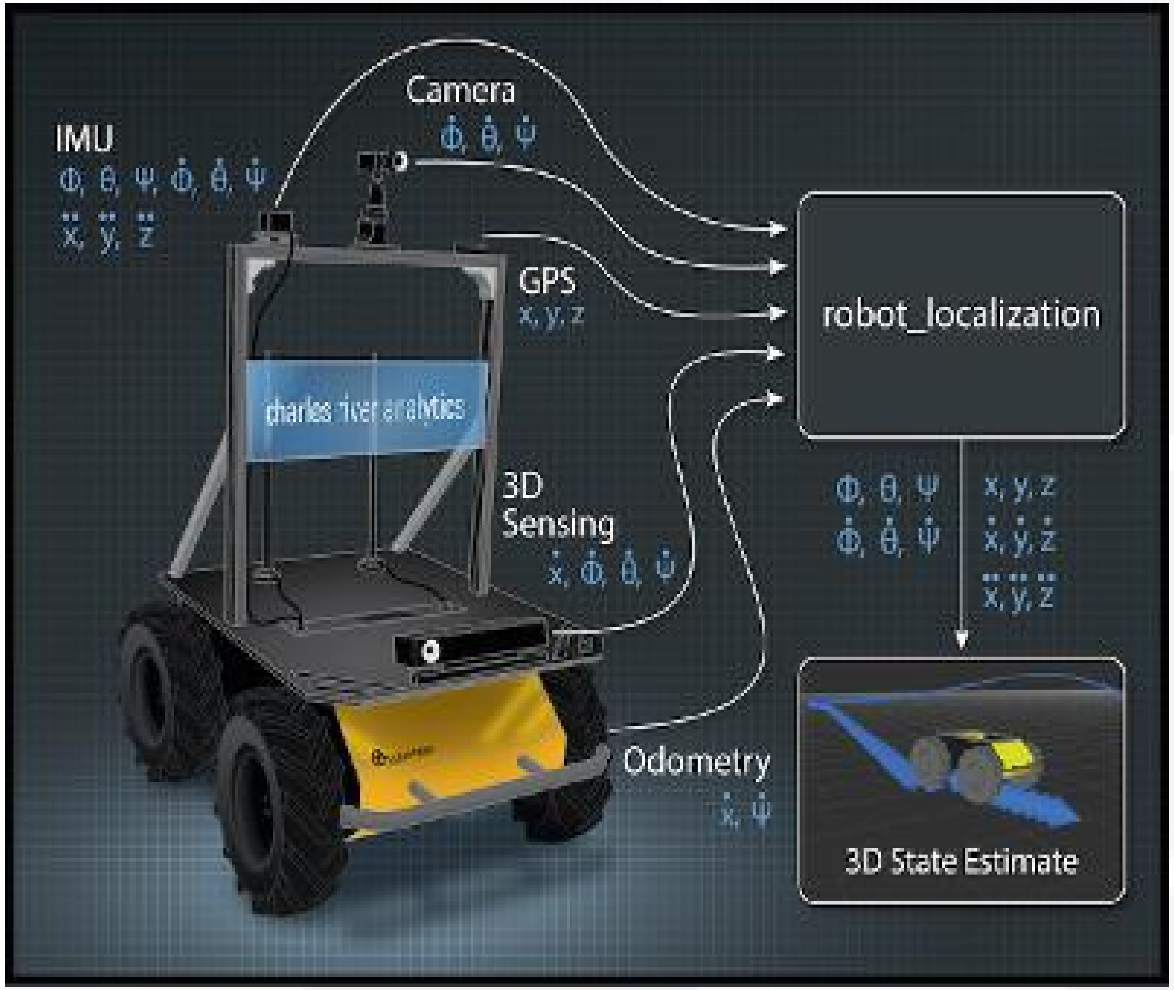
按照其用途可分为：压力传感器、**位置类传感器**、温度传感器、**速度传感器**、**加速度传感器**、**视觉传感器**等。

一、传感器简介:作用

ROS提供了丰富的传感器插件和仿真模型，包括LIDRA、Camera、IMU、GPS等。

传感器可以通过ROS消息接口发布感知数据，供其他节点使用。

- ◆LIDRA ：获取环境三维点云数据
- ◆Camera ：获取图像信息
- ◆IMU ：获取机器人姿态信息
- ◆GPS ：获取机器人位置信息



一、传感器简介:仿真

传感器在 ROS 中的仿真:

- ➢ 如何添加新的传感器
- ➢ 新的传感器如何链接到机器人模型上

基本流程

1. 创建传感器模型，URDF或者xacro文件（对应实体部分）
2. 配置传感器基本信息，并加载插件（<plugin> 标签），相当于加载设备驱动（对应功能模块）
3. 通过 reference=”XXX” 参数设置，将模型和插件对应
4. 通过读取、订阅传感器发布的话题消息，即可得到对应传感器信息

```
<!-- 摄像头相关的 xacro 文件 --> ①
<robot name="my_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <!-- 摄像头属性 -->
    <xacro:property name="camera_length" value="0.01" /> <!-- 摄像头长度(x) -->
    <xacro:property name="camera_width" value="0.025" /> <!-- 摄像头宽度(y) -->
    <xacro:property name="camera_height" value="0.025" /> <!-- 摄像头高度(z) -->
    <xacro:property name="camera_x" value="0.08" /> <!-- 摄像头安装的x坐标 -->
    <xacro:property name="camera_y" value="0.0" /> <!-- 摄像头安装的y坐标 -->
    <xacro:property name="camera_z" value="${base_link_length / 2 + camera_height / 2}" /> <!-- 摄像头安装的z坐标:底盘高
```

```
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">
  <!-- 被引用的link -->
  <gazebo reference="camera"> ③
    <!-- 类型设置为 camara -->
    <sensor type="camera" name="camera_node">
      <update_rate>30.0</update_rate> <!-- 更新频率 -->
      <!-- 摄像头基本信息设置 -->
      <camera name="head">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>1280</width>
          <height>720</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.02</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <!-- 核心插件 -->
      <plugin name="gazebo_camera" filename="libgazebo_ros_camera.so"> ②
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
        <cameraName>/camera</cameraName>
        <imageTopicName>image_raw</imageTopicName> ④
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>camera</frameName>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
    </sensor>
  </gazebo>
</robot>
```

重庆大学 CHONGQING UNIVERSITY

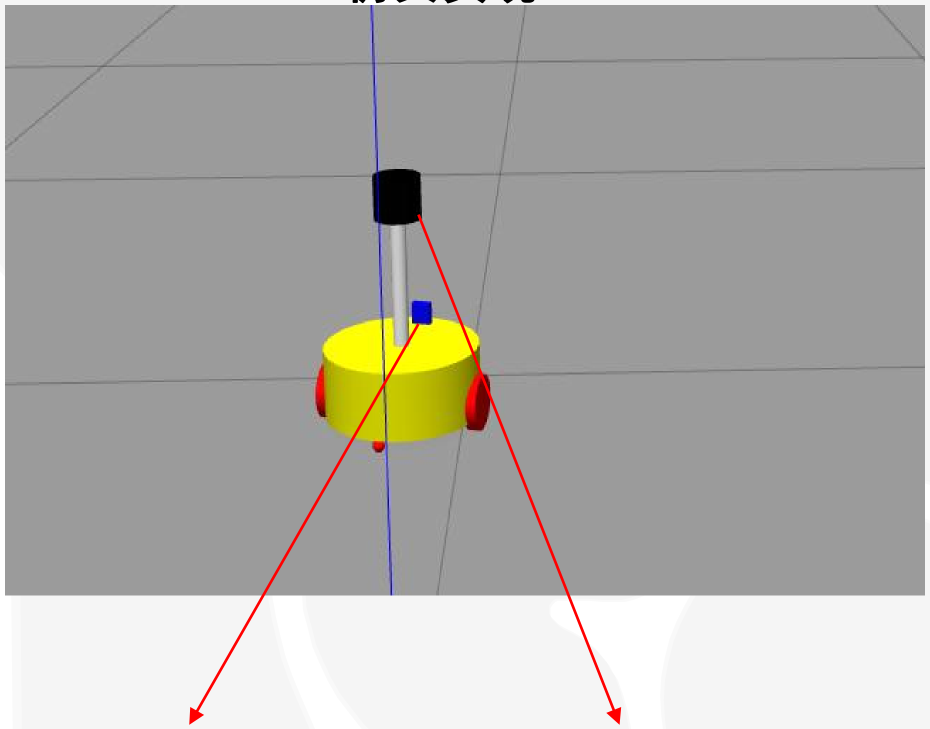
传感器在 ROS 中的仿真:

- ➤ 如何添加新的传感器
- ➤ 新的传感器如何链接到机器人模型上

基本流程

1. 创建传感器模型，URDF或者xacro文件（对应实体部分）
2. 配置传感器基本信息，并加载插件（<plugin> 标签），相当于加载设备驱动（对应功能模块）
3. 通过 reference=” XXX” 参数设置，将模型和插件对应
4. 通过读取、订阅传感器发布的话题消息，即可得到对应传感器信息

仿真实现:

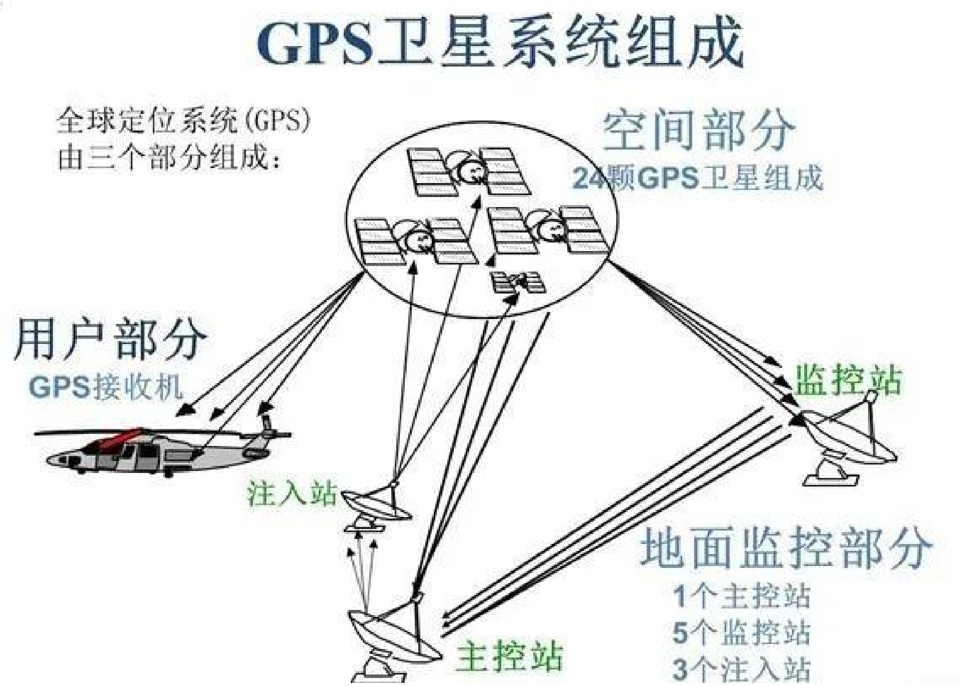


二、定位类传感器:GPS

重庆大学 CHONGQING UNIVERSITY

一：概念

- 美国GPS（Global Positioning System），源自美国军方，是现使用最广泛的全球定位系统
- 欧盟的“伽利略”系统
- 俄罗斯的“格洛纳斯”系统
- **中国北斗卫星导航系统**



GPS系统包含三个部分：

- 空间部分主要是卫星群，向用户部分发送位置、时间等信息
- 地面监控部分，监视控制空间部分
- 用户部分接受空间部分发送的信息，根据信息计算本身的三维位置、速度和时间等

二、定位类传感器:GPS

重慶大學 CHONGQING UNIVERSITY

二：数据格式

ROS中GPS数据主要包含：gps 裸数据ros封装，位置，时间以及速度。

GPS 裸数据

nmea_msgs/Sentence Message

File: nmea_msgs/Sentence.msg

Raw Message Definition

```
# A message representing a single NMEA0183 sentence.

# header.stamp is the ROS Time when the sentence was read.
# header.frame_id is the frame of reference reported by the satellite
#        receiver, usually the location of the antenna.  This is a
#        Euclidean frame relative to the vehicle, not a reference
#        ellipsoid.
Header header

# This should only contain ASCII characters in order to be a valid NMEA0183 sentence.
string sentence
```

Compact Message Definition

```
std_msgs/Header header
string sentence
```

Meassage格式为：nmea_msgs/Sentence

header：消息头；包含seq、stamp以及frame_id，分别表示序列号、时间戳和帧id

sentence：GPS 裸数据

使用rosmsg指令查看此消息，命令如下：

```
rosmsg show nmea_msgs/Sentence
```

```
bonen@bonen:~/sensor_ws$ rosmsg show nmea_msgs/Sentence
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string sentence
```

二、定位类传感器:GPS

二：数据格式

sensor_msgs/NavSatFix Message

File: sensor_msgs/NavSatFix.msg

位置消息

Raw Message Definition

```
# Navigation Satellite fix for any Global Navigation Satellite System
#
# Specified using the WGS 84 reference ellipsoid

# header.stamp specifies the ROS time for this measurement (the
#        corresponding satellite time may be reported using the
#        sensor_msgs/TimeReference message).
#
# header.frame_id is the frame of reference reported by the satellite
#        receiver, usually the location of the antenna.  This is a
#        Euclidean frame relative to the vehicle, not a reference
#        ellipsoid.
Header header

# satellite fix status information
NavSatStatus status

# Latitude [degrees]. Positive is north of equator; negative is south.
float64 latitude

# Longitude [degrees]. Positive is east of prime meridian; negative is west.
float64 longitude

# Altitude [m]. Positive is above the WGS 84 ellipsoid
# (quiet NaN if no altitude is available).
float64 altitude

# Position covariance [m^2] defined relative to a tangential plane
# through the reported position. The components are East, North, and
# Up (ENU), in row-major order.
#
# Beware: this coordinate system exhibits singularities at the poles.

float64[9] position_covariance

# If the covariance of the fix is known, fill it in completely. If the
# GPS receiver provides the variance of each measurement, put them
# along the diagonal. If only Dilution of Precision is available,
# estimate an approximate covariance from that.

uint8 COVARIANCE_TYPE_UNKNOWN = 0
uint8 COVARIANCE_TYPE_APPROXIMATED = 1
uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN = 2
uint8 COVARIANCE_TYPE_KNOWN = 3

uint8 position_covariance_type
```


Meassage格式为：

sensor_msgs/NavSatFix.msg

header：消息头，同上

status：全球定位系统类型及其状态

latitude：纬度

longitude：经度

altitude：高度

positioncovariance：位置协方差

使用rosmsg指令查看此消息，
命令如下：

```
rosmsg show sensor_msgs/NavSatFix
```

二：数据格式

sensor_msgs/TimeReference Message

File: sensor_msgs/TimeReference.msg　时间消息

Raw Message Definition

```
# Measurement from an external time source not actively synchronized with the system clock.

Header header        # stamp is system time for which measurement was valid
                     # frame_id is not used

time   time_ref      # corresponding time from this external source
string  source       # (optional) name of time source
```

geometry_msgs/TwistStamped Message

File: geometry_msgs/TwistStamped.msg　速度消息

Raw Message Definition

```
# A twist with reference coordinate frame and timestamp
Header header
Twist  twist
```

Meassage格式为：sensor_msgs/TimeReference

header：消息头，同上

time_ref：表示外部源时间

source：表示外部源名称

Meassage格式为：geometry_msgs/TwistStamped

header：消息头，同上

twist：代表速度，线速度和角速度

二、定位类传感器:GPS

重庆大学 CHONGQING UNIVERSITY

三：GPS插件使用

第一步：下载插件

http://wiki.ros.org/hector_gazebo下载对应版本的插件：

libhector_gazebo_ros_gps.so

第二步：编译

将下载下来的文件放入到 ros 的工作空间内，然后catkin_make

第三步：在仿真模型里面添加插件

```xml
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">
    <gazebo>
        <!-- 核心插件 -->
        <plugin name="gps" filename="libhector_gazebo_ros_gps.so">
        <!-- GPS 基本信息 -->
            <updateRate>10.0</updateRate>
            <topicName>sensor_msgs/NavSatFix</topicName>
            <gaussianNoise>0.0 0.0 0.0</gaussianNoise>
            <offset>0 0 0</offset>
            <velocityGaussianNoise>0 0 0</velocityGaussianNoise>
            <frameId>base_link</frameId>
        </plugin>
    </gazebo>
</robot>
```

GPS消息话题名

③

```xml
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <!-- 包含惯性矩阵文件 -->
    <xacro:include filename="inertia_matrix.xacro" />
    <!-- 组合小车底盘与摄像头与雷达 -->
    <xacro:include filename="car_base.urdf.xacro" />
    <xacro:include filename="car_camera.urdf.xacro" />
    <xacro:include filename="car_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
    <!-- 摄像头仿真的 xacro 文件 -->
    <xacro:include filename="car_camera_sensor.urdf.xacro" />
    <!--GPS仿真的 xacro 文件 -->
    <xacro:include filename="car_GPS_sensor.urdf.xacro" />
    <!-- imu仿真的 xacro 文件 -->
    <xacro:include filename="car_imu_sensor.urdf.xacro" />
</robot>
```

④

二、定位类传感器:GPS

三：GPS插件使用

结果展示：

启动launch文件后，使用 rostopic list 指令查看 GPS 发布消息话题，并使用 rostopic echo 指令打印消息

rostopic list	rostopic echo /sensor_msgs/NavSatFix

```
/sensor_msgs/NavSatFix
/sensor_msgs/NavSatFix/position/parameter_descriptions
/sensor_msgs/NavSatFix/position/parameter_updates
/sensor_msgs/NavSatFix/status/parameter_descriptions
/sensor_msgs/NavSatFix/status/parameter_updates
/sensor_msgs/NavSatFix/velocity/parameter_descriptions
/sensor_msgs/NavSatFix/velocity/parameter_updates
/tf
/tf_static
bonen@bonen:~/urdf_ws$
```

```
bonen@bonen:~/urdf_ws$ rostopic echo /sensor_msgs/NavSatFix
header:
  seq: 1539
  stamp:
    secs: 411
    nsecs: 500000000
  frame_id: "base_link"
status:
  status: 0
  service: 0
latitude: 49.900000005494476
longitude: 8.900000007899445
altitude: 9.630073555255692e-08
position_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
position_covariance_type: 2
---
```

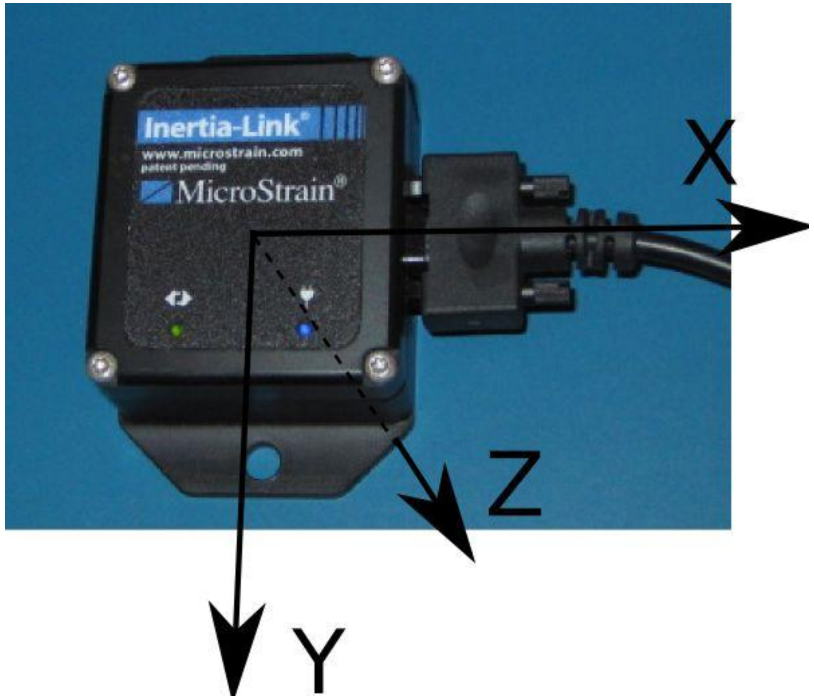
二、定位类传感器:IMU

重庆大学 CHONGQING UNIVERSITY

一：概念

Inertial Measurement Unit（惯性测量单元）

主要用于测量自身位姿（包含位置和姿态），主要包含两个器件，加速计和陀螺仪



- 加速计测量三轴上的加速度
- 陀螺仪测量绕三轴的角速度

通过融合两种传感器数据，可以计算出IMU自身的位姿变化，即当前时刻相对于上一时刻的姿态变化。

CHONGQING UNIVERSITY

二：数据格式

sensor_msgs/Imu Message

File: sensor_msgs/Imu.msg

Raw Message Definition

```
# This is a message to hold data from an IMU (Inertial Measurement Unit)
#
# Accelerations should be in m/s^2 (not in g's), and rotational velocity should be in rad/sec
#
# If the covariance of the measurement is known, it should be filled in (if all you know is the
# variance of each measurement, e.g. from the datasheet, just put those along the diagonal)
# A covariance matrix of all zeros will be interpreted as "covariance unknown", and to use the
# data a covariance will have to be assumed or gotten from some other source
#
# If you have no estimate for one of the data elements (e.g. your IMU doesn't produce an orientation
# estimate), please set element 0 of the associated covariance matrix to -1
# If you are interpreting this message, please check for a value of -1 in the first element of each
# covariance matrix, and disregard the associated estimate.

Header header

geometry_msgs/Quaternion orientation
float64[9] orientation_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 angular_velocity
float64[9] angular_velocity_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 linear_acceleration
float64[9] linear_acceleration_covariance # Row major x, y z
```

Meassage格式为：sensor_msgs/Imu

header：消息头，同前文GPS

orientation：姿态，使用四元数表示

orientation_covariance：姿态协方差

angular_velocity：角速度

angular_velocity_covariance：角速度协方差

linear_acceleration：线加速度

linear_acceleration_covariance：线加速度协方差

二、定位类传感器:IMU

三：IMU 插件使用

1. 配置 IMU 传感器信息

```xml
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">
  <gazebo reference="base_link">
    <gravity>true</gravity>
    <sensor name="imu_sensor" type="imu">
      <always_on>true</always_on>
      <update_rate>100</update_rate>
      <visualize>true</visualize>
      <topic>__default_topic__</topic>
      <plugin filename="libgazebo_ros_imu_sensor.so" name="imu_plugin">
        <topicName>imu</topicName>
        <bodyName>imu_link</bodyName>
        <updateRateHZ>10.0</updateRateHZ>
        <gaussianNoise>0.0</gaussianNoise>
        <xyzOffset>0 0 0</xyzOffset>
        <rpyOffset>0 0 0</rpyOffset>
        <frameName>imu_link</frameName>
        <initialOrientationAsReference>false</initialOrientationAsReference>
      </plugin>
      <pose>0 0 0 0 0 0</pose>
    </sensor>
  </gazebo>
</robot>
```

2. 集成到小车 xacro 文件

```xml
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
  <!-- 包含惯性矩阵文件 -->
  <xacro:include filename="inertia_matrix.xacro" />
  <!-- 组合小车底盘与摄像头与雷达 -->
  <xacro:include filename="car_base.urdf.xacro" />
  <xacro:include filename="car_camera.urdf.xacro" />
  <xacro:include filename="car_laser.urdf.xacro" />
  <xacro:include filename="move.urdf.xacro" />
  <!-- 摄像头仿真的 xacro 文件 -->
  <xacro:include filename="car_camera_sensor.urdf.xacro" />
  <!--GPS仿真的 xacro 文件 -->
  <xacro:include filename="car_GPS_sensor.urdf.xacro" />
  <!-- imu仿真的 xacro 文件 -->
  <xacro:include filename="car_imu_sensor.urdf.xacro" />
</robot>
```

关联link标签

核心插件

话题名称

IMU基本信息设置

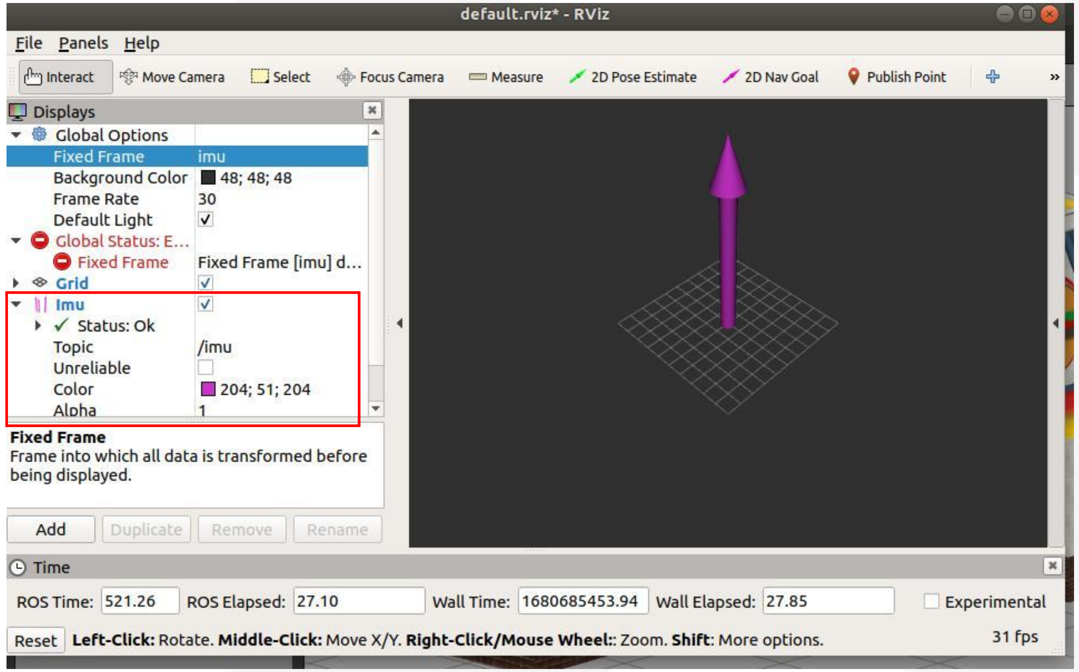
三：IMU 插件使用

结果展示：

启动launch文件后，使用 rostopic list 指令查看 GPS 发布消

```
rostopic list
```

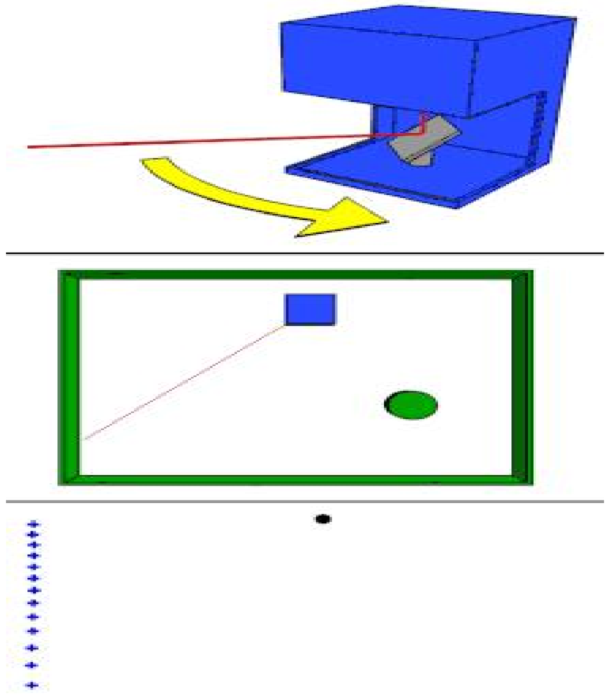
```
/clock
/cmd_vel
/fix_velocity
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/imu
/initialpose
/joint_states
/move_base_simple/goal
/odom
/rosout
/rosout_agg
```



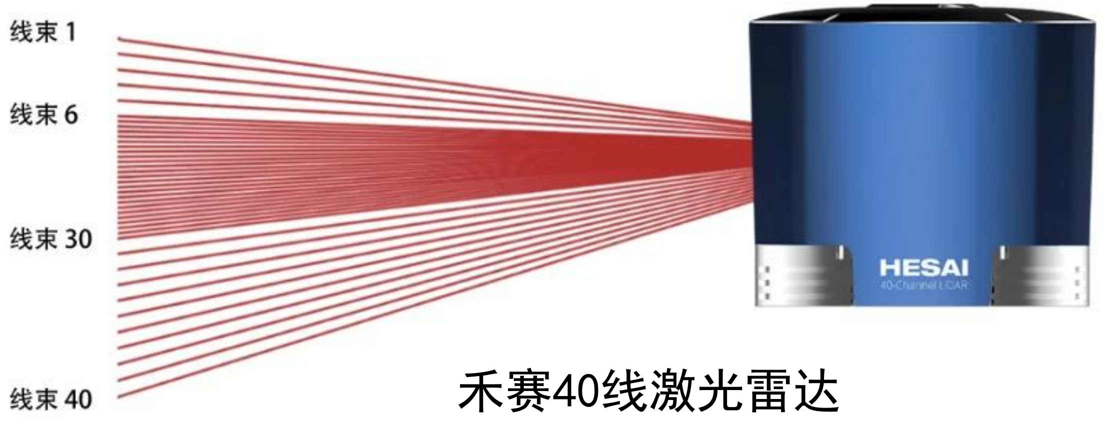
三、LIDAR：概念

- **定义**：使用激光束来测量距离、位置和其他环境信息的传感器。
- **作用**：实时感知周围环境，帮助车辆进行导航、路径规划和避障

按照线束可分为两类：一是单线激光雷达，二是多线激光雷达。



单线雷达扫描



禾赛40线激光雷达

- 单线雷达只有一个激光发射器和接收器，只能检测同一个高度的障碍物，不能测量整体轮廓；
- 多线雷达在垂直方向上具有多个发射器和接收器，可以计算物体的高度信息，并对周围环境进行3D建模。

三、LIDAR:数据格式

CHONGQING UNIVERSITY 重庆大学

sensor_msgs/LaserScan 描述一次扫描的数据，包括扫描角度、时间以及扫描距离等

sensor_msgs/LaserScan Message

File: sensor_msgs/LaserScan.msg

Raw Message Definition

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

header：消息头，同前文GPS
angle_min：扫描起始角度
angle_max：扫描结束角度
angle_increment：两次扫描间隔的角度
time_increment：两次扫描的时间间隔

scan_time：此帧数据所用的时间，一般取第一帧和最后一帧的平均时间

range_min：最小有效距离
range_max：最大有效距离
ranges：所有扫描点的距离
intensities：所有扫描点的反射强度

使用rosmsg指令查看此消息，命令如下：

```
rosmsg show sensor_msgs/LaserScan
```

三、LIDAR:使用

单线激光雷达

1. 配置 laser

```xml
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">

  <!-- 雷达 -->
  <gazebo reference="laser">
    <sensor type="ray" name="rplidar">
      <pose>0 0 0 0 0 0</pose>
      <visualize>true</visualize>
      <update_rate>5.5</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>360</samples>
            <resolution>1</resolution>
            <min_angle>-3</min_angle>
            <max_angle>3</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.10</min>
          <max>30.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin name="gazebo_rplidar" filename="libgazebo_ros_laser.so">
        <topicName>/scan</topicName>
        <frameName>laser</frameName>
      </plugin>
    </sensor>
  </gazebo>

</robot>
```

2. 集成到小车 xacro 文件

```xml
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <!-- 包含惯性矩阵文件 -->
    <xacro:include filename="inertia_matrix.xacro" />
    <!-- 组合小车底盘与摄像头与雷达 -->
    <xacro:include filename="car_base.urdf.xacro" />
    <xacro:include filename="car_camera.urdf.xacro" />
    <xacro:include filename="car_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
    <!-- 摄像头仿真的 xacro 文件 -->
    <xacro:include filename="car_camera_sensor.urdf.xacro" />
    <!--GPS仿真的 xacro 文件 -->
    <xacro:include filename="car_GPS_sensor.urdf.xacro" />
    <!-- imu仿真的 xacro 文件 -->
    <xacro:include filename="car_imu_sensor.urdf.xacro" />
    <!-- 激光雷达仿真的 xacro 文件 -->
    <xacro:include filename="car_laser64_sensor.urdf.xacro" />
</robot>
```

关联link标签

laser基本信息设置

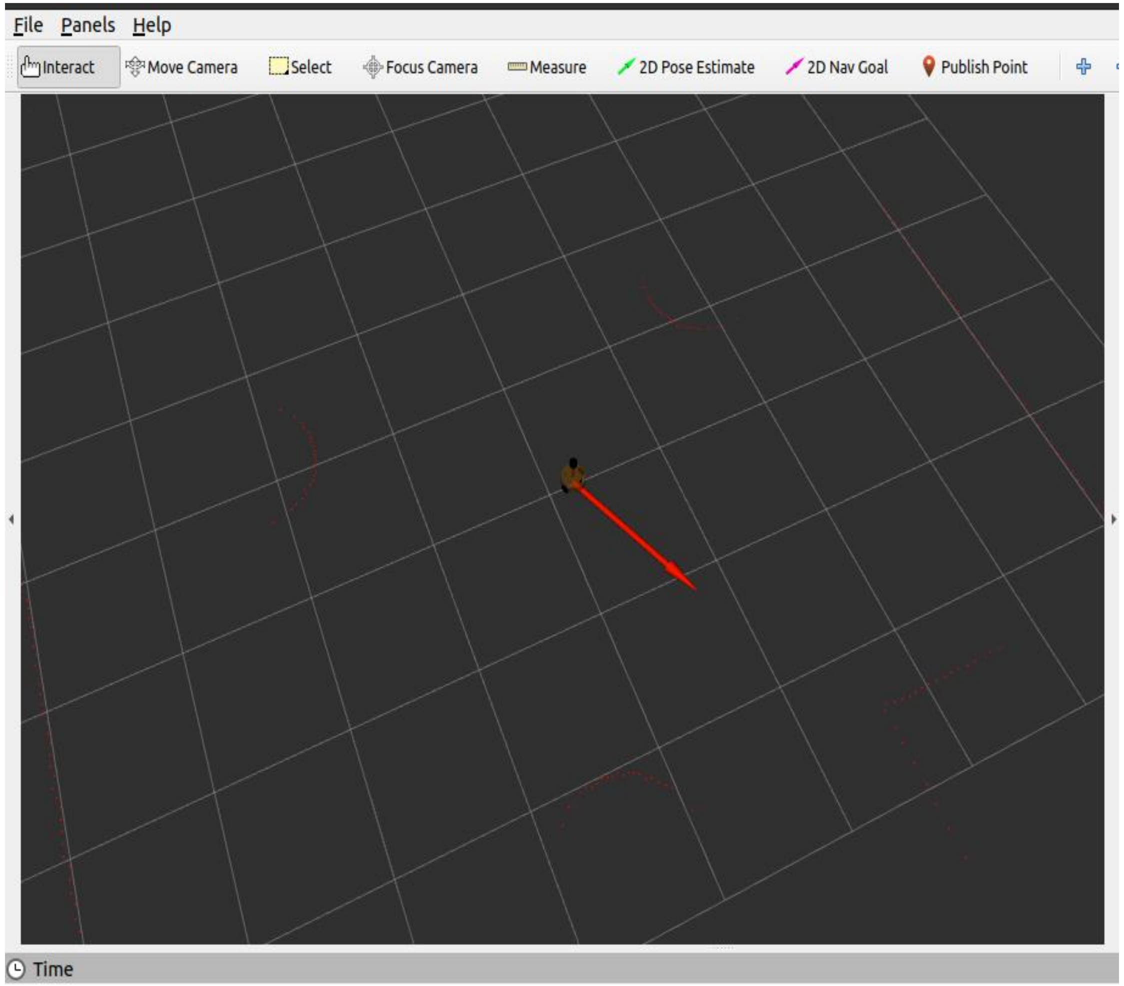
核心插件

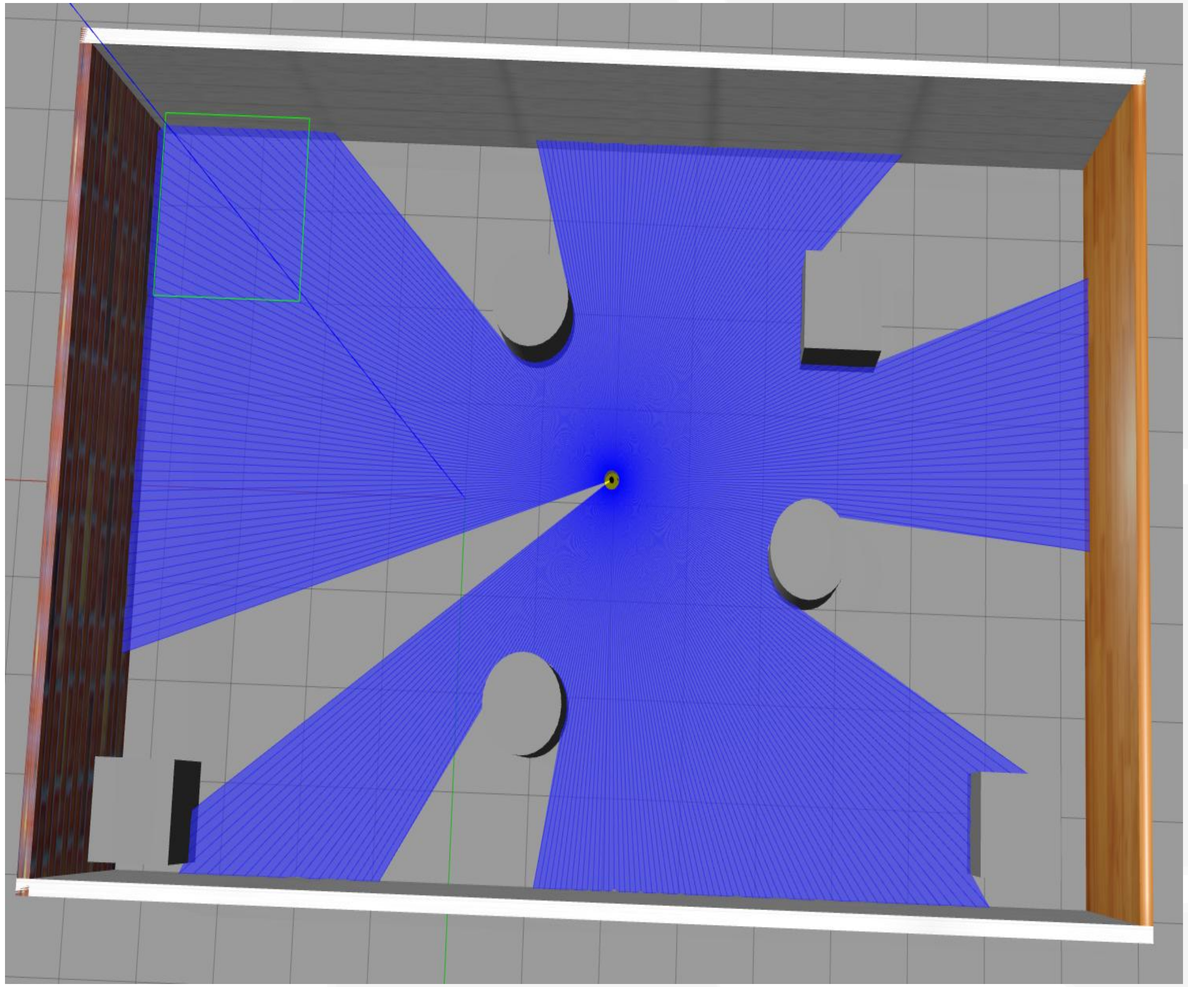
话题名称

重庆大学
CHONGQING UNIVERSITY

单线激光雷达

结果展示：





三、LIDAR:使用

重庆大学 CHONGQING UNIVERSITY

多线激光雷达

1. 配置laser

```
<!-- Gazebo requires the velodyne_gazebo_plugins package -->
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">
<gazebo reference="laser">
    <sensor type="gpu_ray" name="velodyne-VLP16">
      <pose>0 0 0 0 0 0</pose>
      <visualize>false</visualize>
      <update_rate>10</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>1875</samples>
            <resolution>1</resolution>
            <min_angle>0.9</min_angle>
            <max_angle>130.0</max_angle>
          </horizontal>
          <vertical>
            <samples>16</samples>
            <resolution>1</resolution>
            <min_angle>-15.0*3.1415926535897931/180.0</min_angle>
            <max_angle> 15.0*3.1415926535897931/180.0</max_angle>
          </vertical>
        </scan>
        <range>
          <min>0.3</min>
          <max>131</max>
          <resolution>0.001</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.0</stddev>
        </noise>
      </ray>
      <plugin name="gazebo_ros_laser_controller" filename="libgazebo_ros_velodyne_gpu_laser.so">
        <topicName>/velodyne_points</topicName>
        <frameName>/laser</frameName>
        <organize_cloud>false</organize_cloud>
        <min_range>-3</min_range>
        <max_range>3</max_range>
        <gaussianNoise>0.008</gaussianNoise>
      </plugin>
    </sensor>
  </gazebo>
</robot>
```

2. 集成到小车 xacro 文件

```
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <!-- 包含惯性矩阵文件 -->
    <xacro:include filename="inertia_matrix.xacro" />
    <!-- 组合小车底盘与摄像头与雷达 -->
    <xacro:include filename="car_base.urdf.xacro" />
    <xacro:include filename="car_camera.urdf.xacro" />
    <xacro:include filename="car_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
    <!-- 摄像头仿真的 xacro 文件 -->
    <!-- <xacro:include filename="car_camera_sensor.urdf.xacro" /> -->
    <!-- 深度相机仿真的 xacro 文件 -->
    <!-- <xacro:include filename="car_kinect_sensor.urdf.xacro" /> -->
    <!--GPS仿真的 xacro 文件 -->
    <!-- <xacro:include filename="car_GPS_sensor.urdf.xacro" /> -->
    <!-- imu仿真的 xacro 文件 -->
    <!-- <xacro:include filename="car_imu_sensor.urdf.xacro" /> -->
    <!-- 单线激光雷达仿真的 xacro 文件 -->
    <!-- <xacro:include filename="car_laser_sensor.urdf.xacro" /> -->
    <!-- 16线激光雷达仿真的 xacro 文件 -->
    <xacro:include filename="car_laser16_sensor.urdf.xacro" />
</robot>
```

关联link标签

设置激光线束

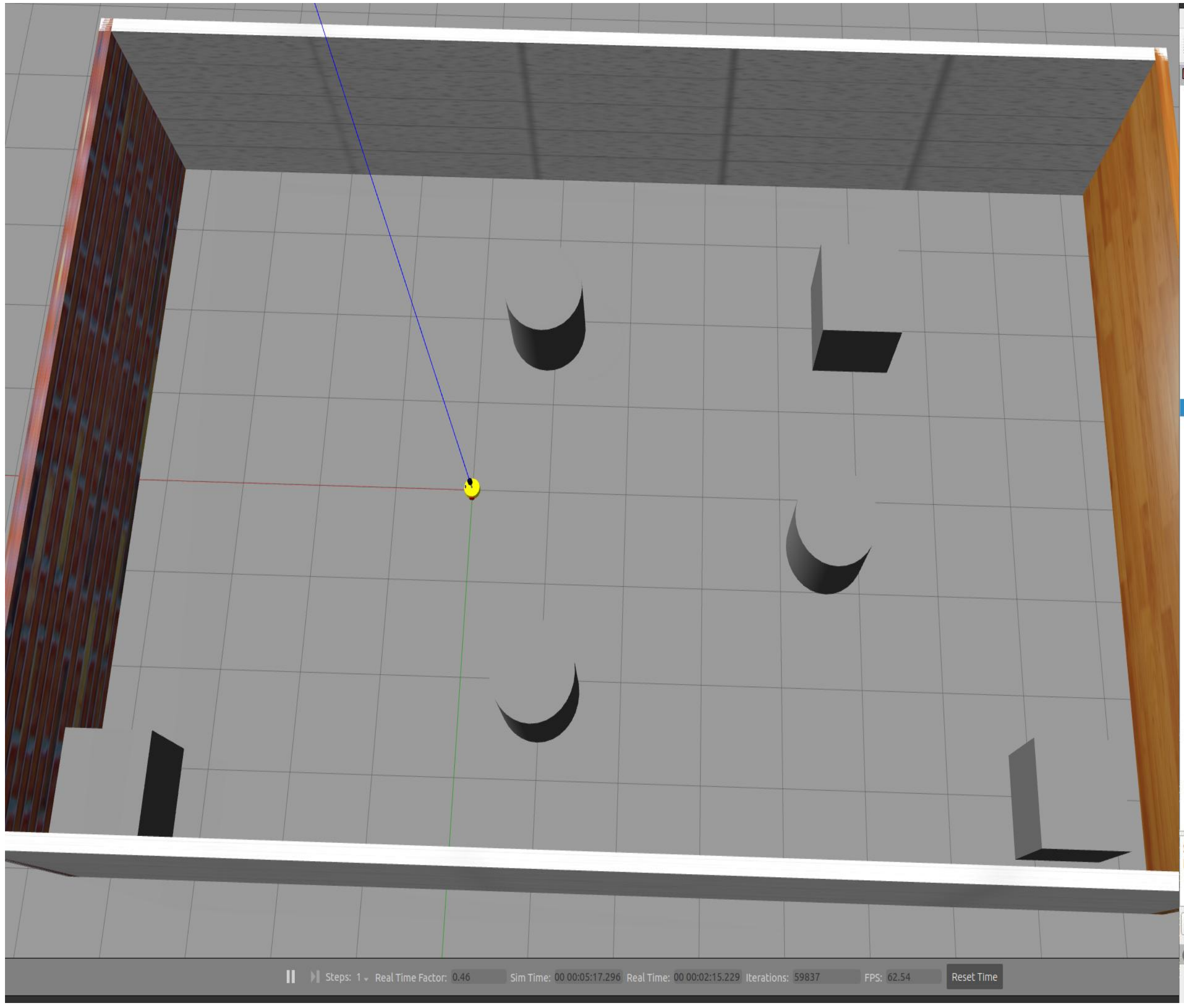
核心插件

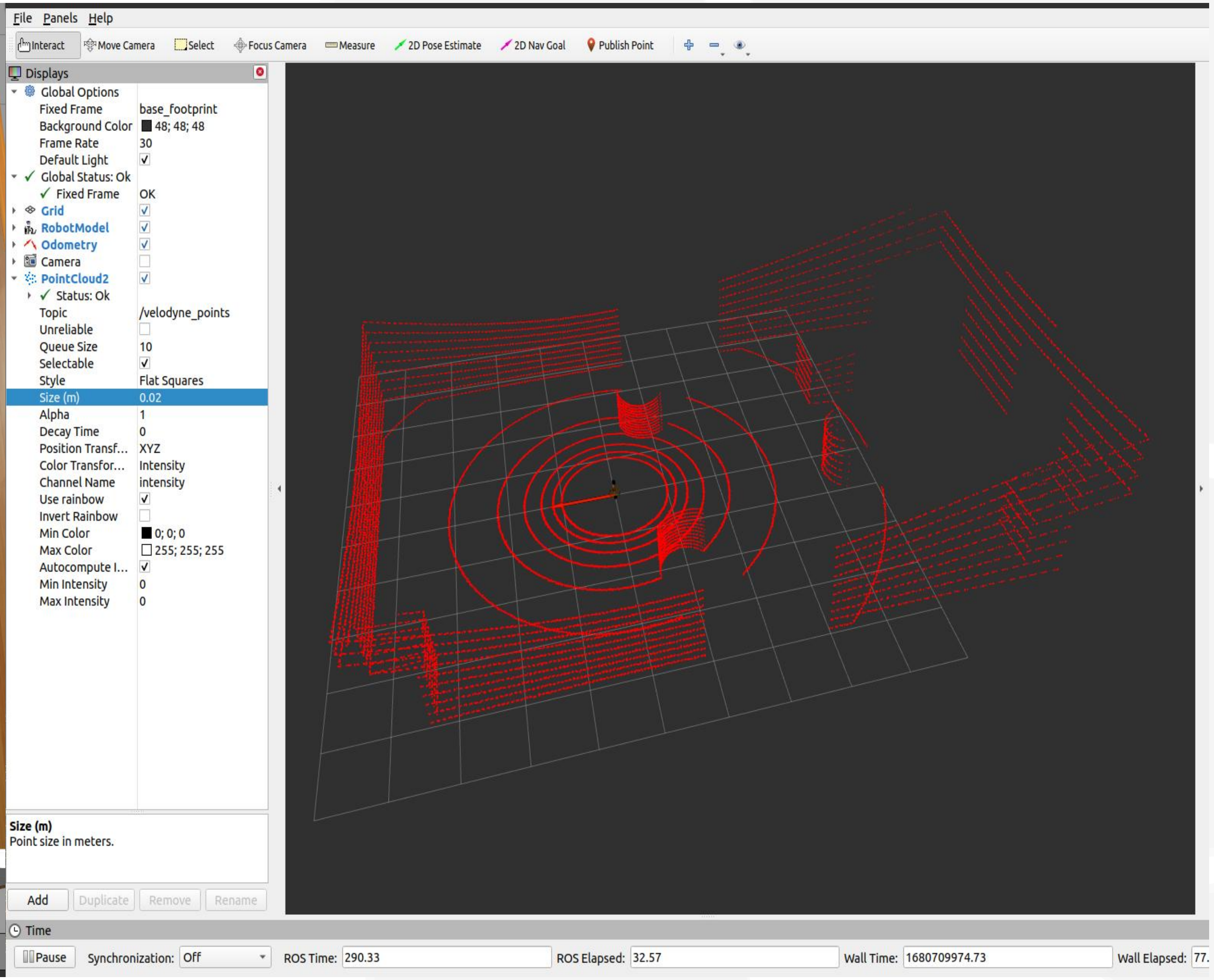
设置话题名称

三、LIDAR:使用

多线激光雷达

结果展示:





重庆大学 CHONGQING UNIVERSITY

Camera即相机，主要分为三类，分别是单目相机，双目相机和深度相机。

- **单目相机：**只有一个摄像头，用于获取周围图像信息
- **双目相机：**有两个摄像头，经过标定的双目相机除了获取图像信息外，还能通过计算获取深度信息
- **深度相机：**可以直接获取图像和深度信息，主要用于环境感知与物体识别

单目相机

双目相机

深度相机

四、Camera:数据格式

Meassage格式为：*sensor_msgs/Image*

Raw Message Definition

```
# This message contains an uncompressed image
# (0, 0) is at top-left corner of image
#

Header header                # Header timestamp should be acquisition time of image
                             # Header frame_id should be optical frame of camera
                             # origin of frame should be optical center of cameara
                             # +x should point to the right in the image
                             # +y should point down in the image
                             # +z should point into to plane of the image
                             # If the frame_id here and the frame_id of the CameraInfo
                             # message associated with the image conflict
                             # the behavior is undefined

uint32 height                # image height, that is, number of rows
uint32 width                 # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding              # Encoding of pixels -- channel meaning, ordering, size
                             # taken from the list of strings in include/sensor_msgs/image_encodings.h

uint8 is_bigendian           # is this data bigendian?
uint32 step                  # Full row length in bytes
uint8[] data                 # actual matrix data, size is (step * rows)
```

header：消息头，同前文GPS

height：代表图像高度，单位是像素

width：表示图像宽度，单位是像素

encoding：代表图像像素的编码格式is_bigendian：表示数据存储是否按照小端存储

step：代表一行的数据量，单位byte

data：数组存储具体图像值

使用rosmsg指令查看此消息，命令如下：

```
rosmsg show sensor_msgs/Image
```

四、Camera:使用

重庆大學 CHONGQING UNIVERSITY

单目相机

插件配置

```xml
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro"><!-- 单目-->
  <!-- 被引用的link -->
  <gazebo reference="camera">
    <!-- 类型设置为 camara -->
    <sensor type="camera" name="camera_node">
      <update_rate>30.0</update_rate> <!-- 更新频率 -->
      <!-- 摄像头基本信息设置 -->
      <camera name="head">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>1280</width>
          <height>720</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.02</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
```

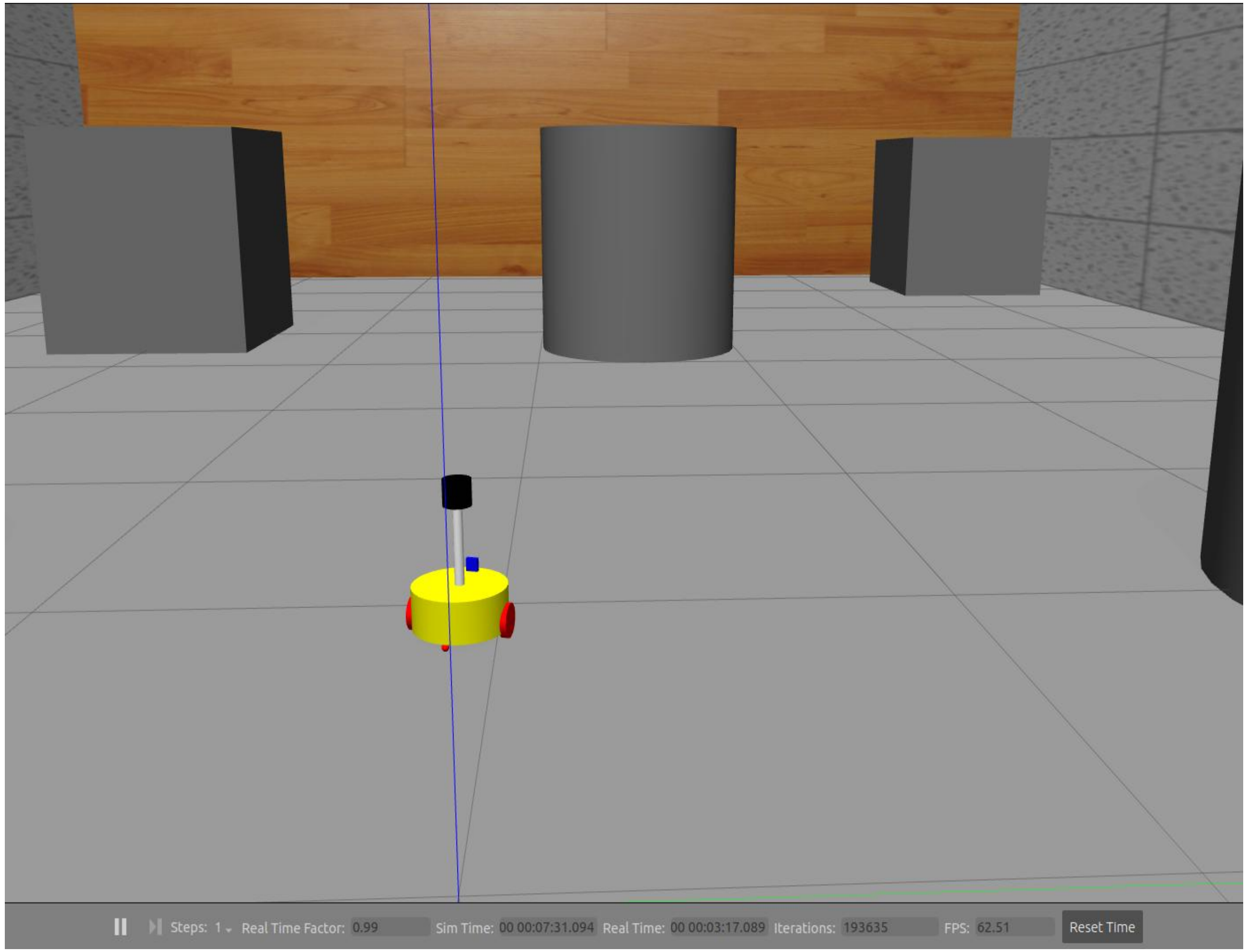
```xml
      <!-- 核心插件 -->
      <plugin name="gazebo_camera" filename="libgazebo_ros_camera.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
        <cameraName>/camera</cameraName>
        <imageTopicName>image_raw</imageTopicName>
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>camera</frameName>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
    </sensor>
  </gazebo>
</robot>
```

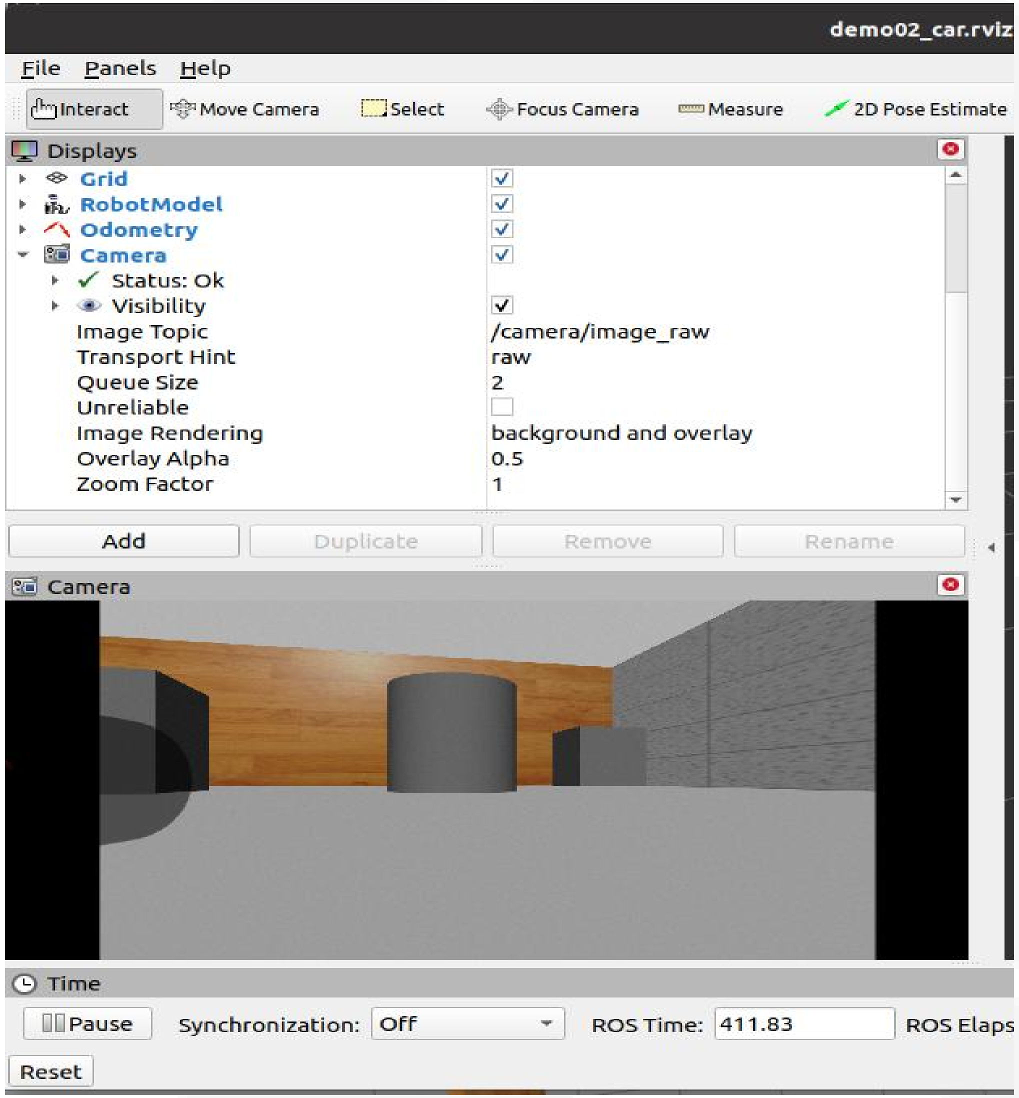
- **reference：** 将插件绑定在名为“camera”的link上
- **update_rate：** 发布频率
- **horizontal_fov：** 设置视场大小
- **clip：** 摄像头可视的最短距离和最远距离

四、Camera:使用

单目相机

结果展示:





四、Camera:使用

CHONGQING UNIVERSITY

深度相机

插件配置

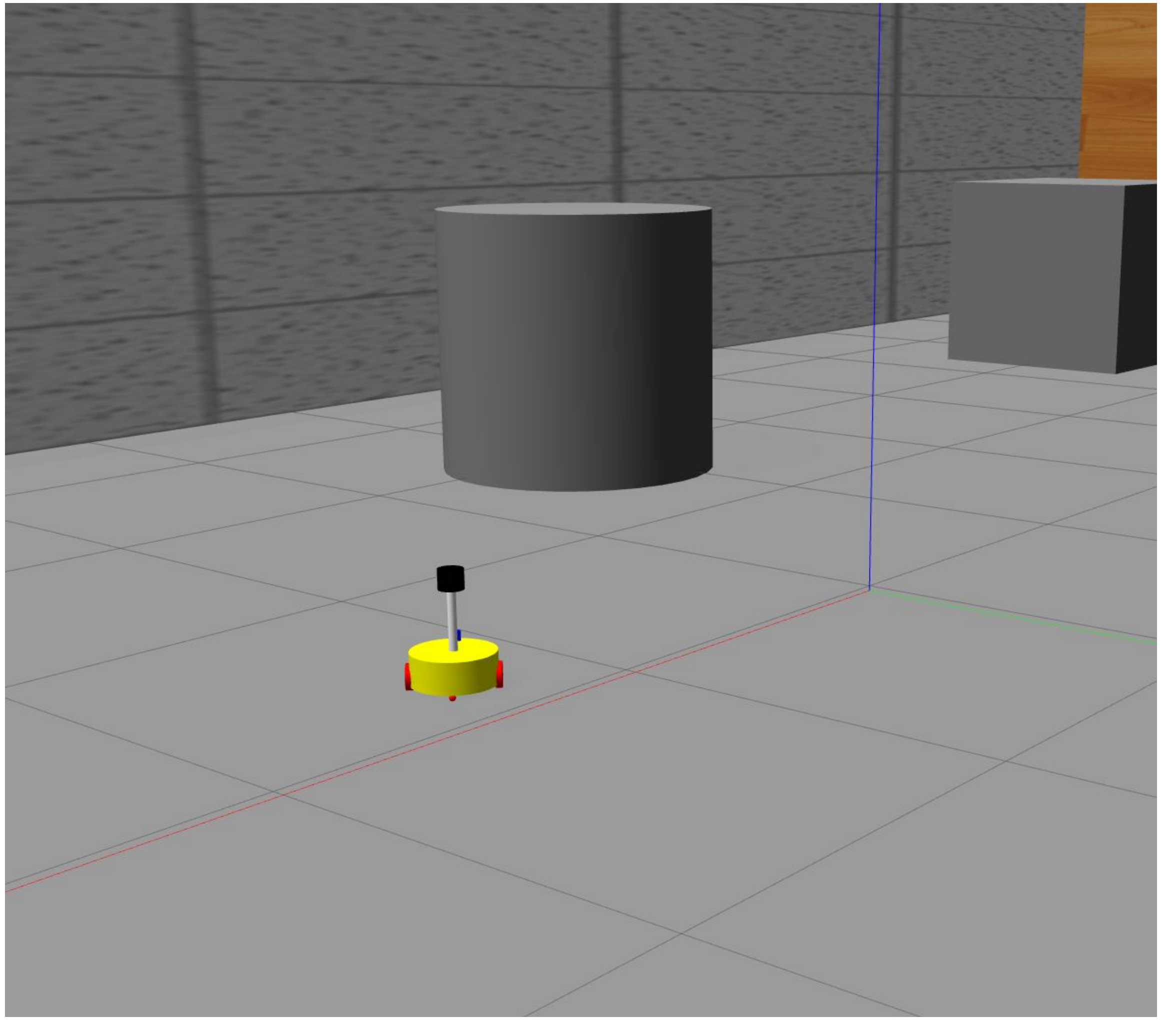
```xml
<robot name="my_sensors"
    xmlns:xacro="http://wiki.ros.org/xacro"> <!-- 深度相机-->
    <gazebo reference="camera">
        <sensor name="deep_camera_node" type="depth">
            <update_rate>20</update_rate>
            <camera>
                <horizontal_fov>1.047198</horizontal_fov>
                <image>
                    <width>640</width>
                    <height>480</height>
                    <format>R8G8B8</format>
                </image>
                <clip>
                    <near>0.05</near>
                    <far>3</far>
                </clip>
            </camera>
```

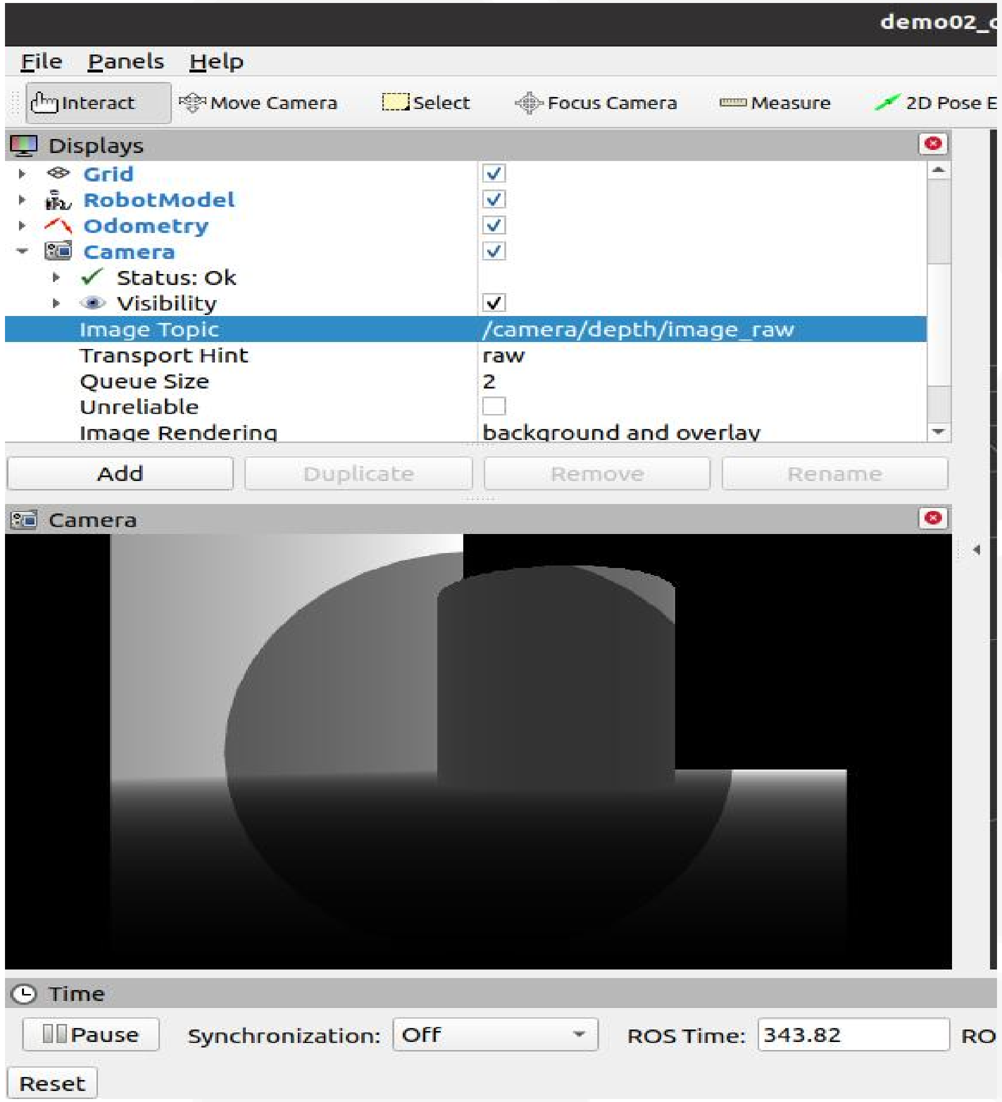
```xml
            <!-- Kinect as an example-->
            <plugin name="deep_camera_controller" filename="libgazebo_ros_openni_kinect.so">
                <baseline>0.2</baseline>
                <alwaysOn>true</alwaysOn>
                <updateRate>1.0</updateRate>
                <cameraName>deep_camera_ir</cameraName>
                <imageTopicName>/deep_camera/color/image_raw</imageTopicName>
                <cameraInfoTopicName>/deep_camera/color/camera_info</cameraInfoTopicName>
                <depthImageTopicName>/deep_camera/depth/image_raw</depthImageTopicName>
                <depthImageInfoTopicName>/deep_camera/depth/camera_info</depthImageInfoTopicName>
                <pointCloudTopicName>/deep_camera/depth/points</pointCloudTopicName>

                <frameName>deep_camera</frameName>

                <pointCloudCutoff>0.5</pointCloudCutoff>
                <pointCloudCutoffMax>3.0</pointCloudCutoffMax>
                <distortionK1>0.00000001</distortionK1>
                <distortionK2>0.00000001</distortionK2>
                <distortionK3>0.00000001</distortionK3>
                <distortionT1>0.00000001</distortionT1>
                <distortionT2>0.00000001</distortionT2>
                <CxPrime>0</CxPrime>
                <Cx>0</Cx>
                <Cy>0</Cy>
                <focalLength>0</focalLength>
                <hackBaseline>0</hackBaseline>
            </plugin>
        </sensor>
    </gazebo>
</robot>
```

四、Camera:使用

深度相机

结果展示:





感谢聆听！