机械与能源工程系
SUSTech Department of Mechanical and Energy Engineering

项 目 报 告

项目名称：_____

课程名称：_____

课程编号：_____

学　　号：_____

姓　　名：_____

专　　业：_____

指导教师：_____

年　　月　　日

目　录

1.
2. 导航

   导航共分为三部分，分别为为机器人添加并实现传感器插件；实时地图构建仿真和路径规划仿真。

   a) 为机器人添加传感器插件

      i. 添加 Kinect

         Kinect 是一种常见的 RGB-D 摄像头，为机器人添加 kinect，需要三维模型文件和模型描述文件。模型文件 kinect.dae 可以在 turtlebot 功能包中找到，模型描述文件参考纸质教材中的 kinect.xacro：

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="kinect_camera">

    <xacro:macro name="kinect_camera" params="prefix:=camera">
        <link name="${prefix}_link">
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <visual>
                <origin xyz="0 0 0" rpy="0 0 ${M_PI/2}"/>
                <geometry>
                    <mesh filename="package://mrobot_description/meshes/kinect.dae"
                </geometry>
            </visual>
            <collision>
                <geometry>
                    <box size="0.07 0.3 0.09"/>
                </geometry>
            </collision>
        </link>

        <joint name="${prefix}_optical_joint" type="fixed">
            <origin xyz="0 0 0" rpy="-1.5708 0 -1.5708"/>
            <parent link="${prefix}_link"/>
            <child link="${prefix}_frame_optical"/>
        </joint>

        <link name="${prefix}_frame_optical"/>
    </xacro:macro>

</robot>
```

在机器人模型的 xacro 中可以将机器人和 kinect 的 xacro 文件拼装在一起：

```
<xacro:include filename="$(find mrobot_description)/urdf/mrobot_body.urdf.xacro'
<xacro:include filename="$(find mrobot_description)/urdf/kinect.xacro" />

<xacro:property name="kinect_offset_x" value="-0.06" />
<xacro:property name="kinect_offset_y" value="0" />
<xacro:property name="kinect_offset_z" value="0.035" />

<!-- Kinect -->
<joint name="kinect_frame_joint" type="fixed">
    <origin xyz="${kinect_offset_x} ${kinect_offset_y} ${kinect_offset_z}" rpy='
    <parent link="plate_2_link"/>
    <child link="camera_link"/>
</joint>
<xacro:kinect_camera prefix="camera"/>
```

如上图所示，kinect_frame_joint 定义了 kinect 部分和机器人主体的关节信息，通过修改之前定义的三个 xacro 属性 x,y,和 z，可以修改 kinect 和主体的相对位置。

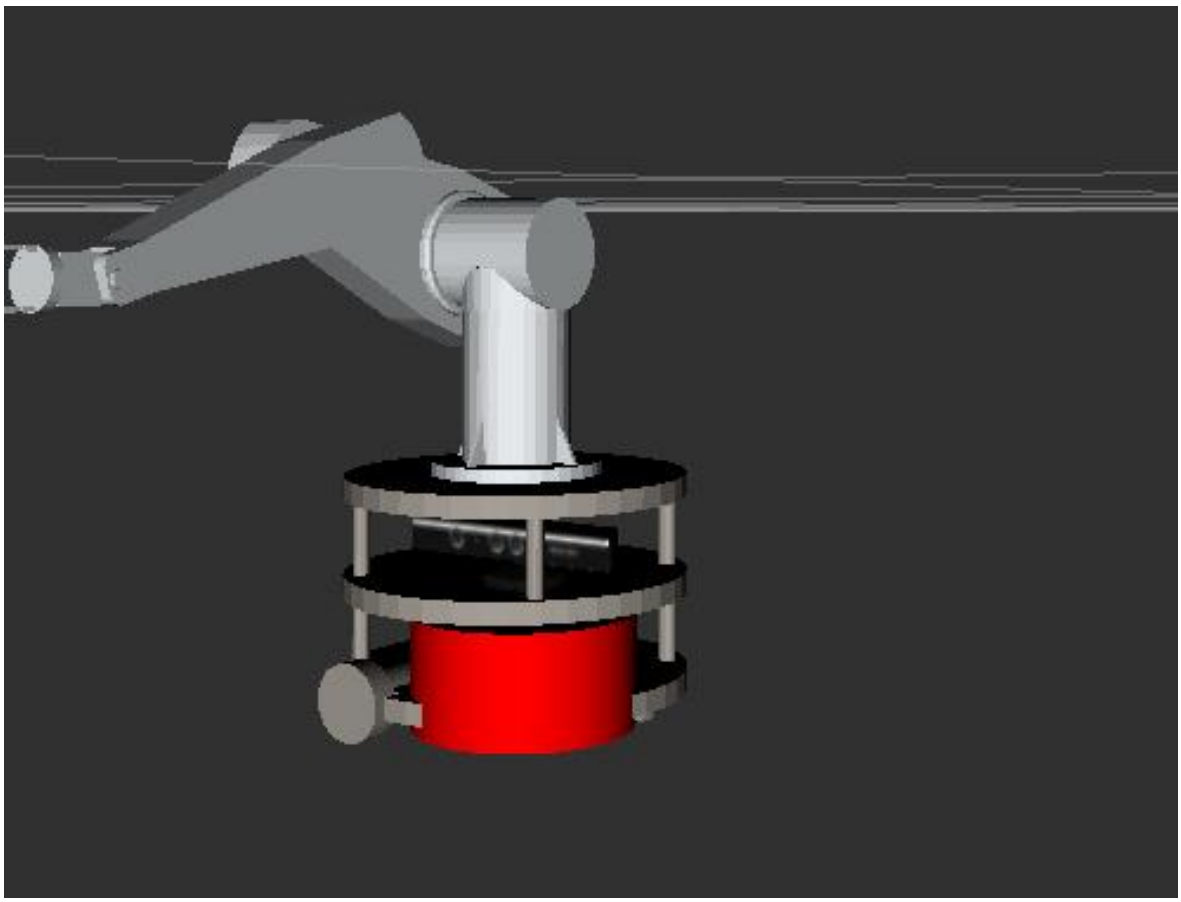
此时运行 xacro 文件，就可以在 rviz 中看到 kinect 的模型已经可视化

ii. 添加激光雷达

添加激光雷达的过程与 kinect 相似：

拼装雷达和机器人主体的 xacro 文件

```
<xacro:rplidar prefix="laser"/>

<joint name="rplidar_joint" type="fixed">
                <origin xyz="${rplidar_offset_x} $
{rplidar_offset_y} ${rplidar_offset_z}" rpy="0 0 0" />
                <parent link="base_link"/>
                <child link="laser_link"/>
</joint>
```



iii. Kinect 仿真

为了让 kinect 可以运作，还需要为 kinect 的 xacro 文件添加 gazebo 标签：

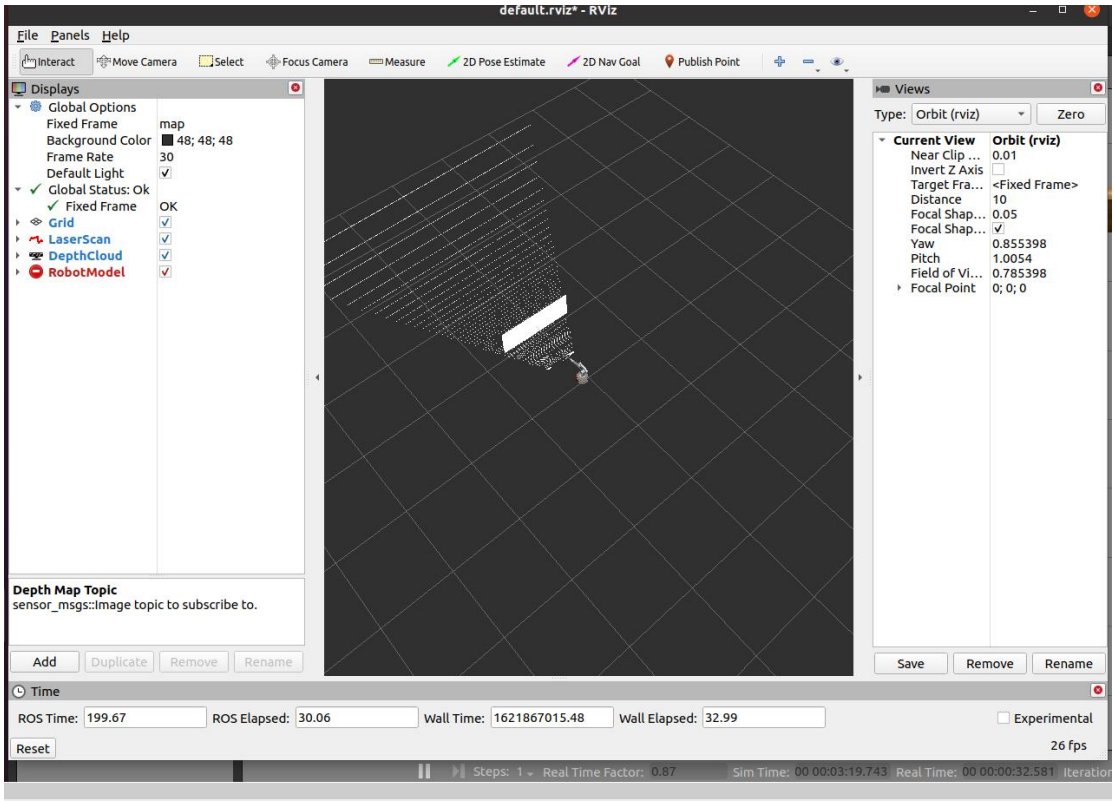
```
<gazebo reference="${prefix}_link">
    <sensor type="depth" name="${prefix}">
        <always_on>true</always_on>
        <update_rate>20.0</update_rate>
        <camera>
            <horizontal_fov>${60.0*M_PI/180.0}</horizontal_fov>
            <image>
                <format>R8G8B8</format>
                <width>640</width>
                <height>480</height>
            </image>
            <clip>
                <near>0.05</near>
                <far>8.0</far>
            </clip>
        </camera>
        <plugin name="kinect_${prefix}_controller" filename="libgazebo_ros_openni_ki
            <cameraName>${prefix}</cameraName>
            <alwaysOn>true</alwaysOn>
            <updateRate>10</updateRate>
            <imageTopicName>rgb/image_raw</imageTopicName>
            <depthImageTopicName>depth/image_raw</depthImageTopicName>
            <pointCloudTopicName>depth/points</pointCloudTopicName>
            <cameraInfoTopicName>rgb/camera_info</cameraInfoTopicName>
            <depthImageCameraInfoTopicName>depth/camera_info</depthImageCameraInfo-
TopicName>
            <frameName>${prefix}_frame_optical</frameName>
            <baseline>0.1</baseline>
            <distortion_k1>0.0</distortion_k1>

            <distortion_k2>0.0</distortion_k2>
            <distortion_k3>0.0</distortion_k3>
            <distortion_t1>0.0</distortion_t1>
            <distortion_t2>0.0</distortion_t2>
            <pointCloudCutoff>0.4</pointCloudCutoff>
        </plugin>
    </sensor>
</gazebo>
```

这里用 plugin 标签添加了 kinect 插件，使 kinect 发布各种数据以及设置参考坐标系等参数，这时启动机器人模型，用 rostopic list ，命令查看话题，如下图则确保 kinect 插件已经启动：

```
/camera/depth/camera_info
/camera/depth/image_raw
/camera/depth/points
/camera/parameter_descriptions
/camera/parameter_updates
/camera/rgb/camera_info
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_raw/theora
/camera/rgb/image_raw/theora/parameter_descriptions
/camera/rgb/image_raw/theora/parameter_updates
```

为了确实在 rviz 中查看点云数据，需要修改 fixed frame 为

“camera_frame_optical”，并添加一个 PointCloud2 插件，修改插件订阅的话题，就可以在主界面看到点云信息了：

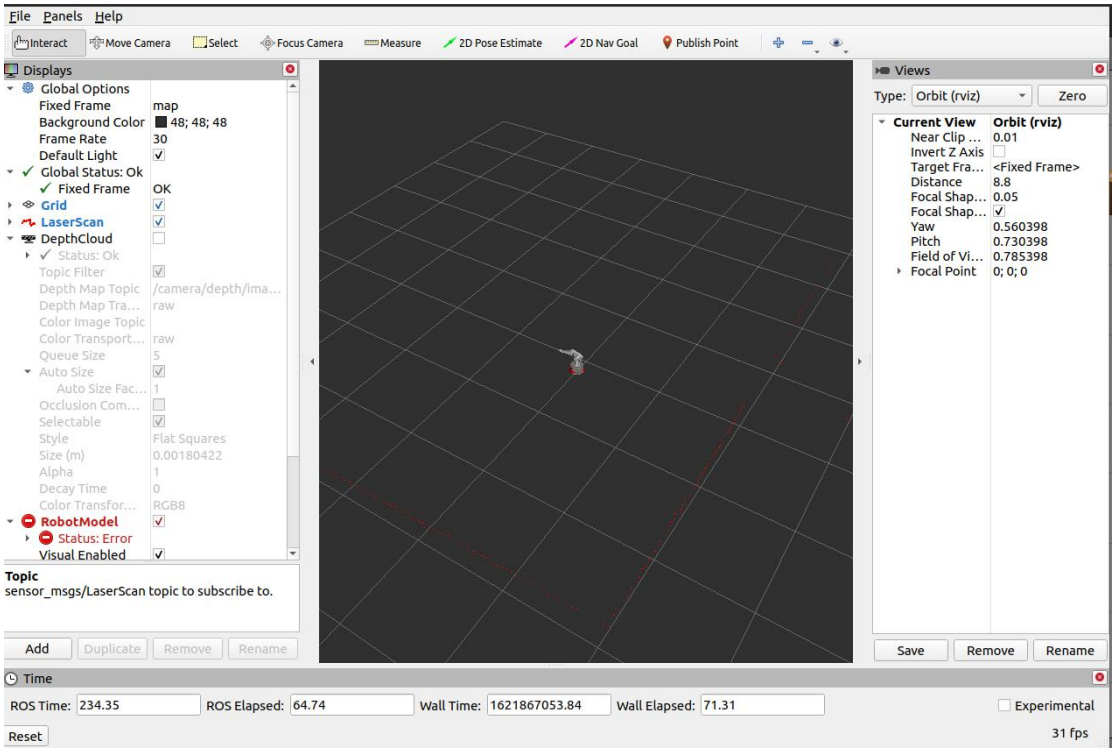


iv. 激光雷达仿真

同样，在激光雷达的 xacro 文件中需要添加 rplidar 插件：

```
<gazebo reference="${prefix}_link">
    <material>Gazebo/Black</material>
</gazebo>

<gazebo reference="${prefix}_link">
    <sensor type="ray" name="rplidar">
        <pose>0 0 0 0 0 0</pose>
        <visualize>false</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>6.0</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_link</frameName>
        </plugin>
    </sensor>
</gazebo>
```

激光雷达发布的话题是/scan，此时查看话题，可以看到 scan 话题已经激活，则 laser 插件启动成功，再次启动 rviz，添加一个 LaserScan 插件，就可以看到界面中的激光数据了：



b) 实时地图构建仿真

i. Gmapping

Gmapping 功能包是 SLAM 中比较成熟的一种，使用即使定位和地图建模方法，接收深度信息、IMU 信息即里程计信息并建立栅格地图，gmapping 功能包可以直接安装：

```
$ sudo apt-get install ros-kinetic-gmapping
```

使用这个功能包的第一步就是创一个基础的配置节点参数的 launch 文件：

```
<launch>
    <arg name="scan_topic" default="scan" />

    <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output= "screen"
        <param name="odom_frame" value="odom"/>
        <param name="map_update_interval" value="5.0"/>
        <!-- Set maxUrange < actual maximum range of the Laser -->
        <param name="maxRange" value="5.0"/>
        <param name="maxUrange" value="4.5"/>
        <param name="sigma" value="0.05"/>
        <param name="kernelSize" value="1"/>
        <param name="lstep" value="0.05"/>
        <param name="astep" value="0.05"/>
        <param name="iterations" value="5"/>
        <param name="lsigma" value="0.075"/>
        <param name="ogain" value="3.0"/>
        <param name="lskip" value="0"/>
        <param name="srr" value="0.01"/>
        <param name="srt" value="0.02"/>
        <param name="str" value="0.01"/>
        <param name="stt" value="0.02"/>
        <param name="linearUpdate" value="0.5"/>
        <param name="angularUpdate" value="0.436"/>
        <param name="temporalUpdate" value="-1.0"/>
        <param name="resampleThreshold" value="0.5"/>
        <param name="particles" value="80"/>


        <param name="xmin" value="-1.0"/>
        <param name="ymin" value="-1.0"/>
        <param name="xmax" value="1.0"/>
        <param name="ymax" value="1.0"/>
        <param name="delta" value="0.05"/>
        <param name="llsamplerange" value="0.01"/>
        <param name="llsamplestep" value="0.01"/>
        <param name="lasamplerange" value="0.005"/>
        <param name="lasamplestep" value="0.005"/>
        <remap from="scan" to="$(arg scan_topic)"/>
    </node>
</launch>
```

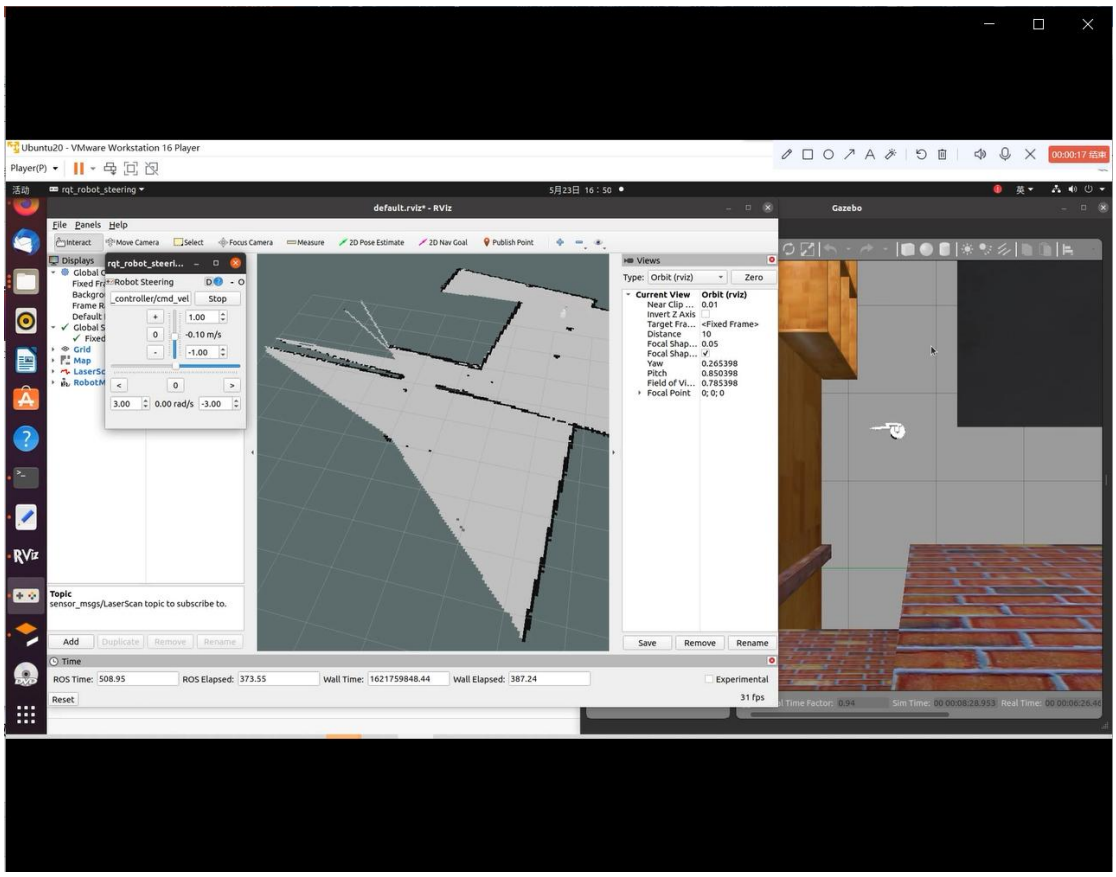
在启动机器人的 launch 文件中直接用 include file 添加即可。

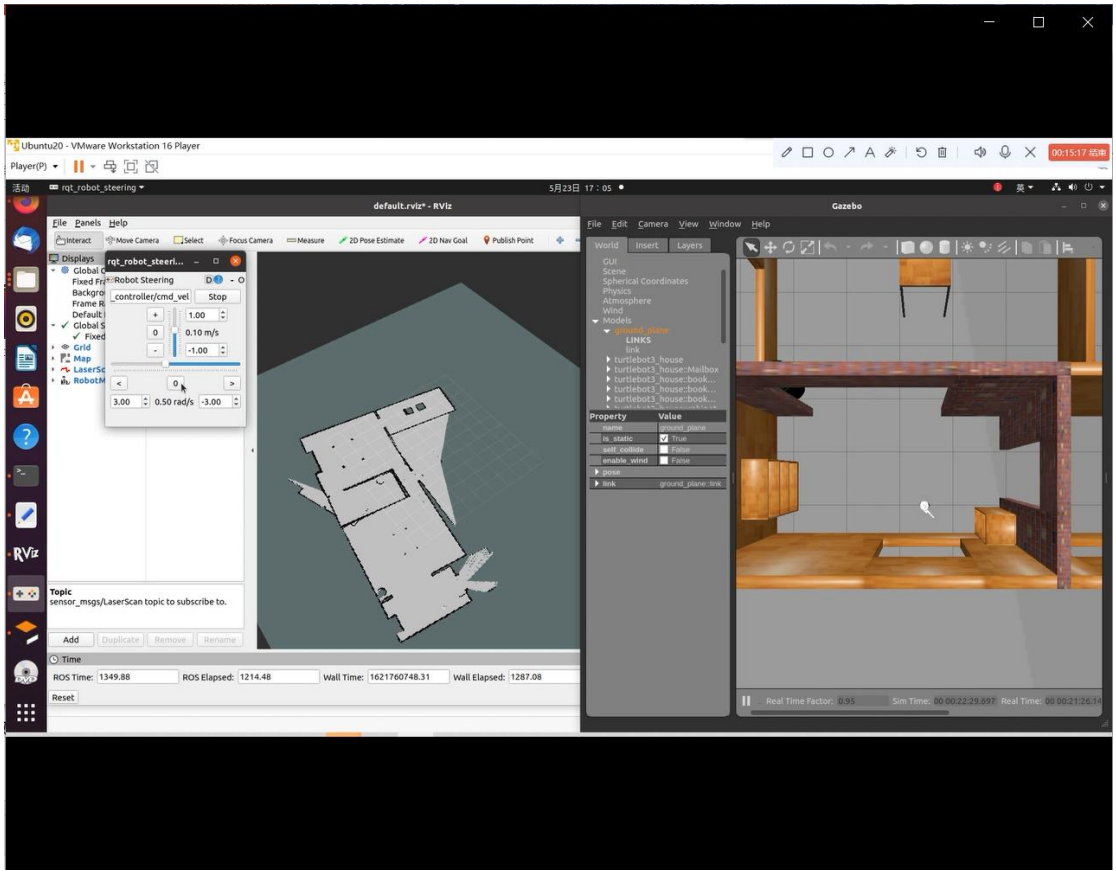
ii. 在 gazebo 中仿真

现在让机器人在 gazebo 和 rviz 中同步移动，就可以看到激光雷达传感器实时监测二维环境深度信息并实时更新，根据深度建立部分环境的地图：在机器人移动时，rviz 中的地图会不断更新，gmapping 也会自动校正之前建立的地图以及机器人在运动中会出现的偏差。

现在比较 kinect 和激光雷达的仿真效果：

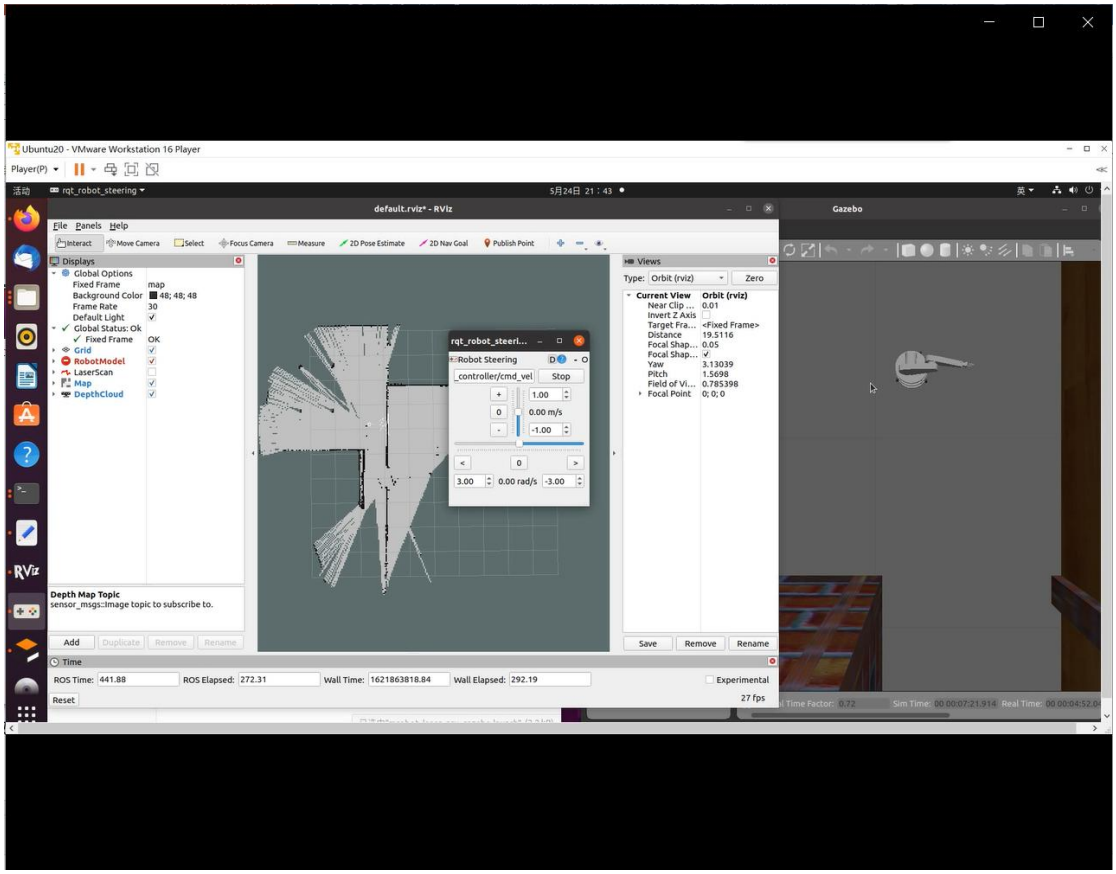
为保证准确性，首先使用公认较精确的激光雷达：

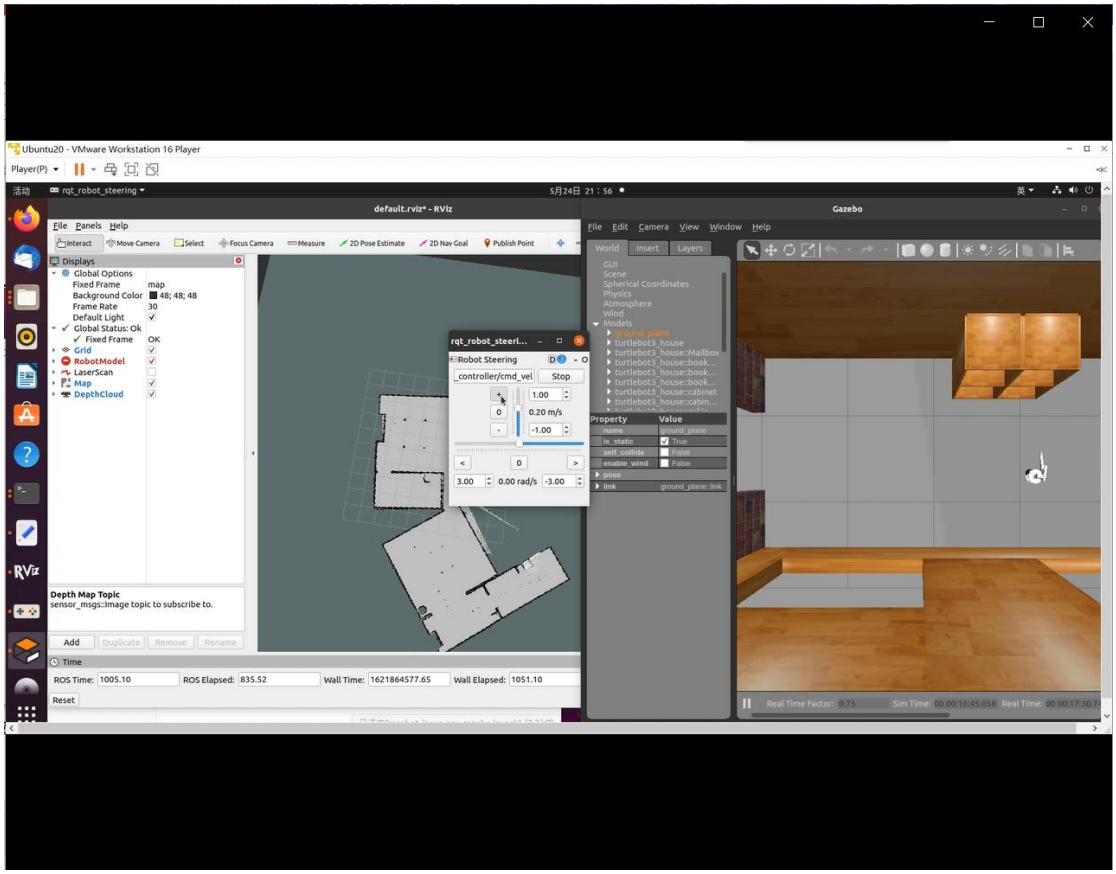




使用激光雷达时可以发现激光雷达探测的信息范围大，更新及时，只需行走一圈地图就比较精确。

现在使用 kinect 仿真：





相比较激光雷达，点云信息更新较慢，而且可以看出在地图构建过

程中，地图由于一次速度较快的转弯出现了极大的偏差，而且不能靠自动校正修复，因此相较于激光雷达，使用 kinect 时对机器人运动的速度限制也比较大。

在地图构建完毕时，使用指令 map server save 就可以 把构建的地图保存为 map 文件，在导航时可以使用。

c) 路径规划仿真

i. 构建配置文件

导航功能包也可以直接下载，本次路径规划使用的是 move_base。导航功能包使用两种代价地图存储障碍信息，因此一共需要三种配置文件，分别是通用、全局规划及本地规划配置文件，在这些配置文件中，需要注意修改的是机器人本体能用于参考的坐标系的名字以及使用的本地地图的名字。

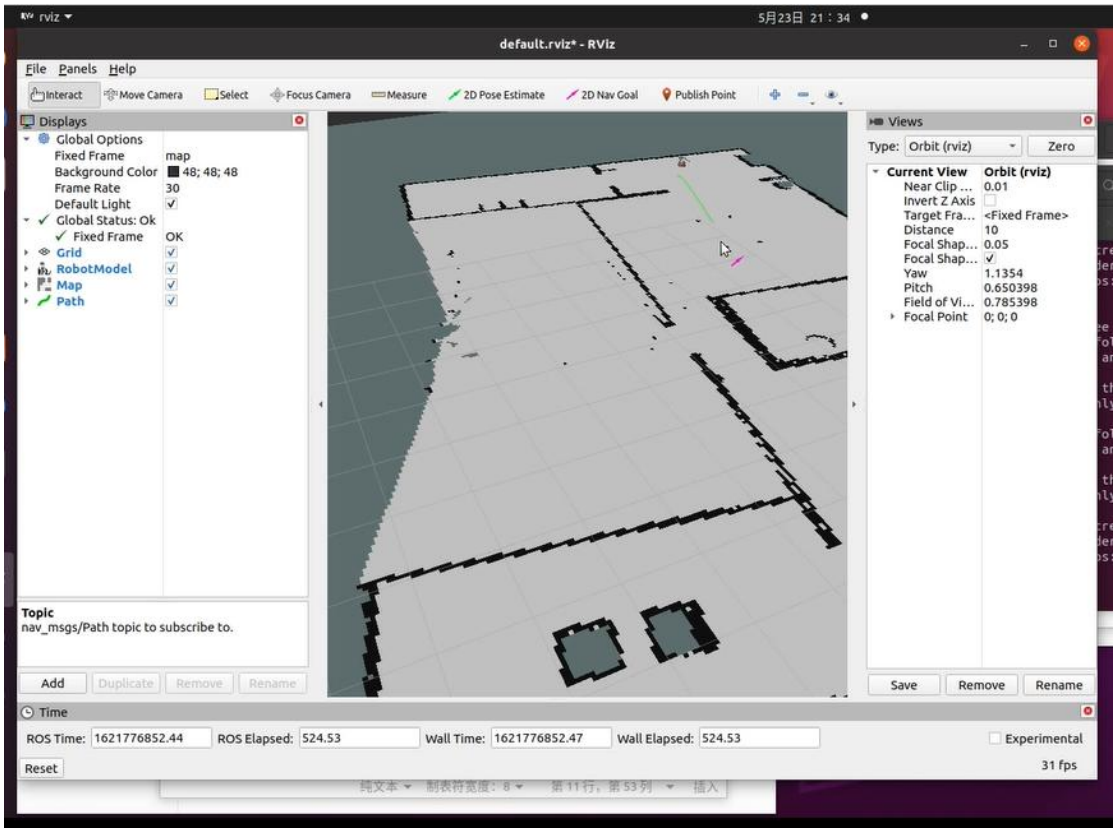
写好配置文件后就可以创建一个节点启动所有的 launch 文件：

```
<launch>

    <node pkg="move_base" type="move_base" respawn="false" name="move_base" output=
        <rosparam file="$(find mrobot_navigation)/config/fake/costmap_common_params.
        <rosparam file="$(find mrobot_navigation)/config/fake/costmap_common_params.
        <rosparam file="$(find mrobot_navigation)/config/fake/local_costmap_params.y
        <rosparam file="$(find mrobot_navigation)/config/fake/global_costmap_params.
        <rosparam file="$(find mrobot_navigation)/config/fake/base_local_planner_par
    </node>

</launch>
```

同时在 launch 文件中添加其他需要的部分如里程计到地图坐标系的变换等。

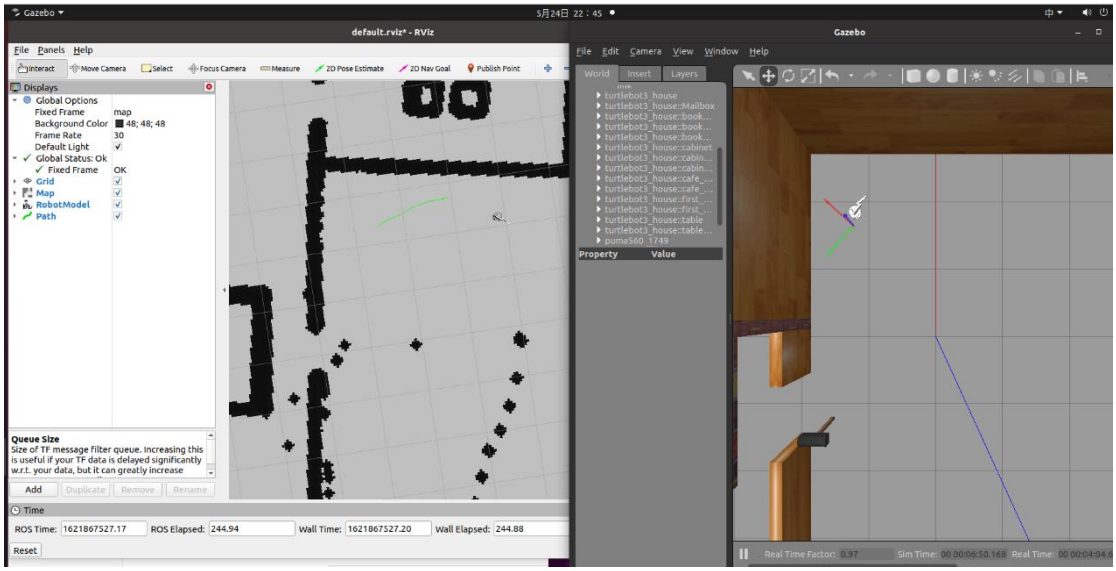
ii. Rviz 仿真

此时运行 launch，添加 map 和 path 插件，在 rviz 中添加 2D nav goal，就可以看到机器人沿一条规划出的路径移动：



iii. Gazebo 仿真

在导航中加入 gazebo 仿真，出现误差：在添加目标点后，发现 gazebo 与 rviz 到达的位置不一致，其中 gazebo 的位置却，继续观察发现机器人在 rviz 与 gazebo 中的运动轨迹包括线速度和角速度完全一致，发现在 rviz 加载建立完成的地图时，地图的初始位置相对于 gazebo 中的真实地图旋转九十度



五、实验结果分析

六、参考文献

胡春旭《ROS 机器人开发实践》

附件：程序核心代码