基于ORB-SLAM3库搭建SLAM系统

作者：郭鸿源

原文地址：基于ORB-SLAM3库搭建SLAM系统

一、基础部分：基于ORB-SLAM3库搭建SLAM系统

1. 注意事项

📌 如果是新系统，需要先安装个git、vi编辑器、DBoW2、g2o（安装过程及可能遇到的问题、相应的对策解决方案放在附录中）

- 备注：DBoW2、g2o在安装ORB-SLAM3中会自动安装编译，不再在附录中另行安装
- DBoW2：主要用于回环检测
- g2o：用于图优化

2. 准备工作，下载安装要用的文件

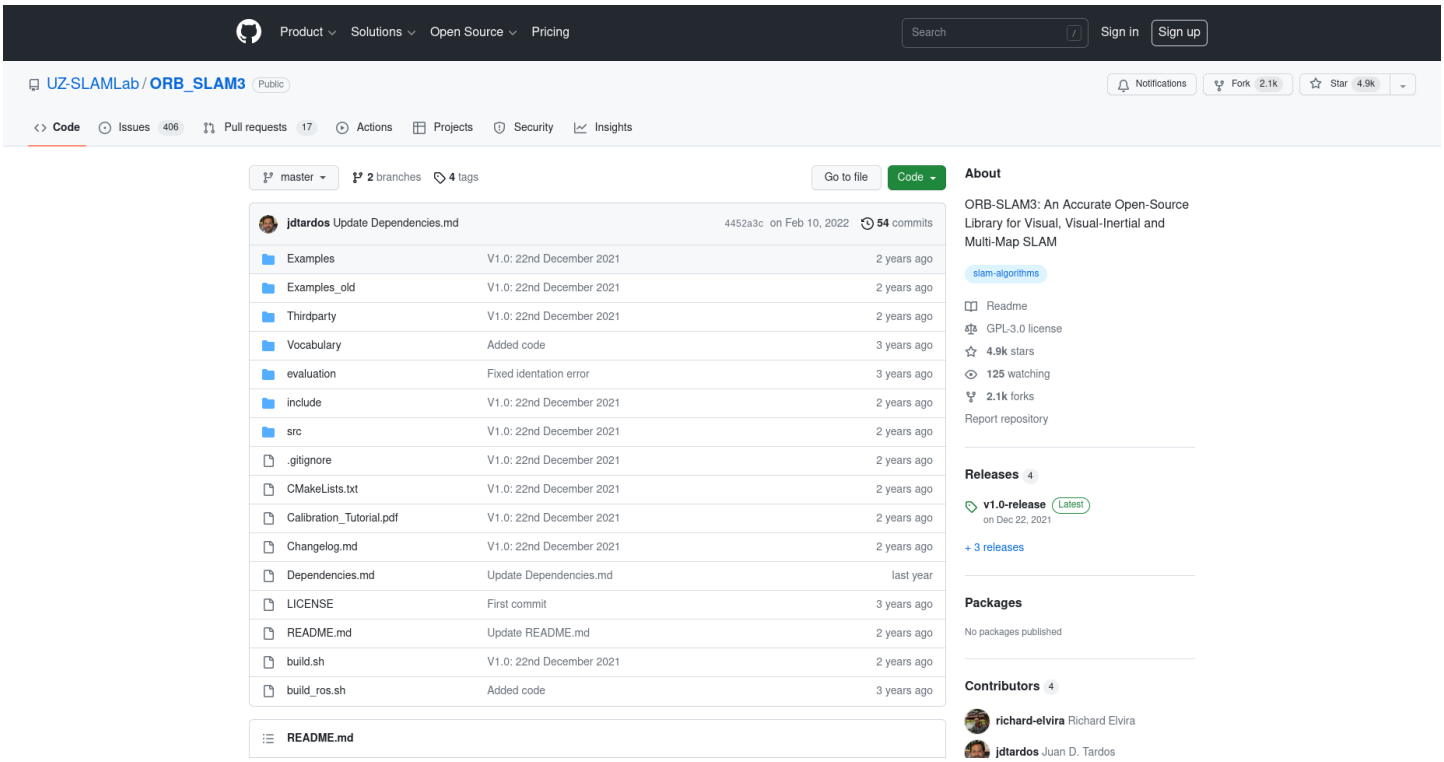
2.1 新建一个文件夹，用于存放下载的文件

```
这里我们将这个文件夹取名为 homework_exercise
```

2.2 在github上下载ORB-SLAM3库

在 homework_exercise 文件夹下打开终端，输入下列命令行或直接在github上下载提取

```
git clone https://github.com/UZ-SLAMLab/ORB_SLAM3
```



2.3 下载Pangolin

在 homework_exercise 文件夹下打开终端，输入下列命令行或直接在github上下载提取

```
git clone https://github.com/stevenlovegrove/Pangolin.git
```

2.4 下载Eigen3

在 homework_exercise 文件夹下打开终端，输入下列命令行或直接在github上下载提取

```
git clone https://github.com/eigenteam/eigen-git-mirror
```

3. 安装下载的及相关所需的库

3.1 安装 Eigen3库

在 homework_exercise 文件夹下打开终端，输入下列命令行

```
cd eigen-git-mirror
mkdir build
cd build
cmake ..
sudo make install

```

```
7 #安装后,头文件安装在/usr/local/include/eigen3/
```

3.2 安装 Pangolin

在 homework_exercise 文件夹下打开终端，输入下列命令行

3.2.1 安装Pangolin需要的依赖工具

在终端依次输入以下指令

```
1 sudo apt install libgl1-mesa-dev
2 sudo apt install libglew-dev
3 sudo apt install cmake
4 sudo apt install libpython2.7-dev
5 sudo apt install pkg-config
6 sudo apt install libegl1-mesa-dev libwayland-dev libxkbcommon-dev wayland-protoc
```

3.2.2 安装 Pangolin

在 homework_exercise 文件夹下打开终端，输入下列命令行

```
1 cd Pangolin
2 mkdir build
3 cd build
4 cmake ..
5 cmake --build .
```

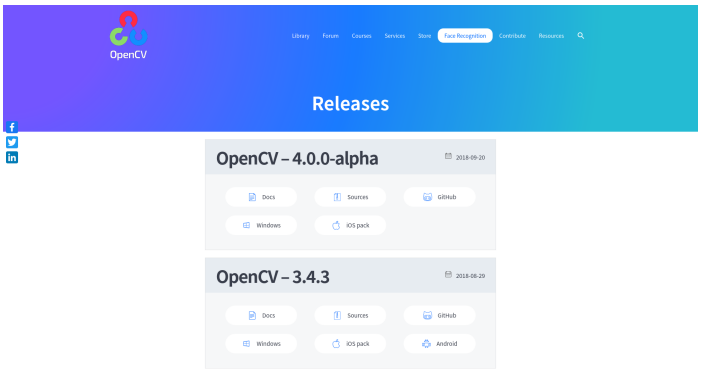
3.3 安装 OpenCV3.4.3

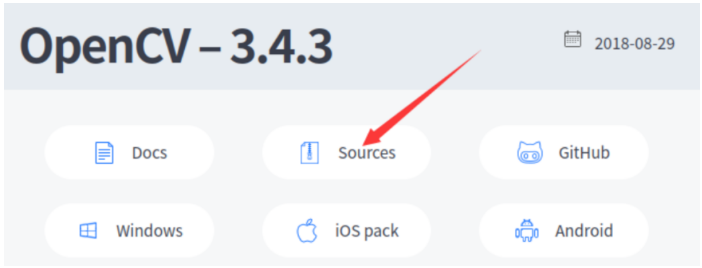
3.3.1 在官网下载OpenCV库

官网网址：https://opencv.org/releases/page/5/

📌 由于OpenCV - 4.0.0-alpha使用的博主不多，更常用的是3.4.3版本，这里笔者使用的是3.4.3

在“OpencV -3.4.3”板块 中找到“source”并点击





点击下载之后，将压缩文件放到 homework_exercise 文件夹，右键提取文件进行解压

更新一下，准备下一步，安装OpenCV依赖库。在终端输入以下指令

```
sudo apt-get update
```

3.3.2 安装OpenCV依赖

在终端中输入下列指令

```
sudo apt-get install build-essential libgtk2.0-dev libavcodec-dev libavformat-dev libjpeg-dev libtiff5-dev libswscale-dev libjasper-dev

# 原博客安装的是libtiff4-dev，有博主运行过程报错，改成libtiff5-dev
# 这里，笔者使用安装libtiff4-dev时也没有报错，笔者最后选择libtiff5-dev安装。大家可以根据实际情况选择。
```

3.3.3 编译OpenCV

在 homework_exercise 文件夹下打开终端，输入下列命令行

```
cd opencv-3.4.3
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

再次执行cmake指令，完成编译任务，在终端中输入下列命令行

```
cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

用make完成后续编译任务，在终端中输入下列命令行

```
make -j4
```

最后，完成编译安装

```
sudo make install
```

3.3.4 配置环境

3.3.4.1 添加库路径

```
sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
```

3.3.4.2 更新系统库

```
sudo ldconfig
```

3.3.4.3 配置bash

```
sudo gedit /etc/bash.bashrc
```

打开后在文件末尾添加下面两行代码

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

保存文件，执行下面的指令使配置生效

```
source /etc/bash.bashrc
```

更新

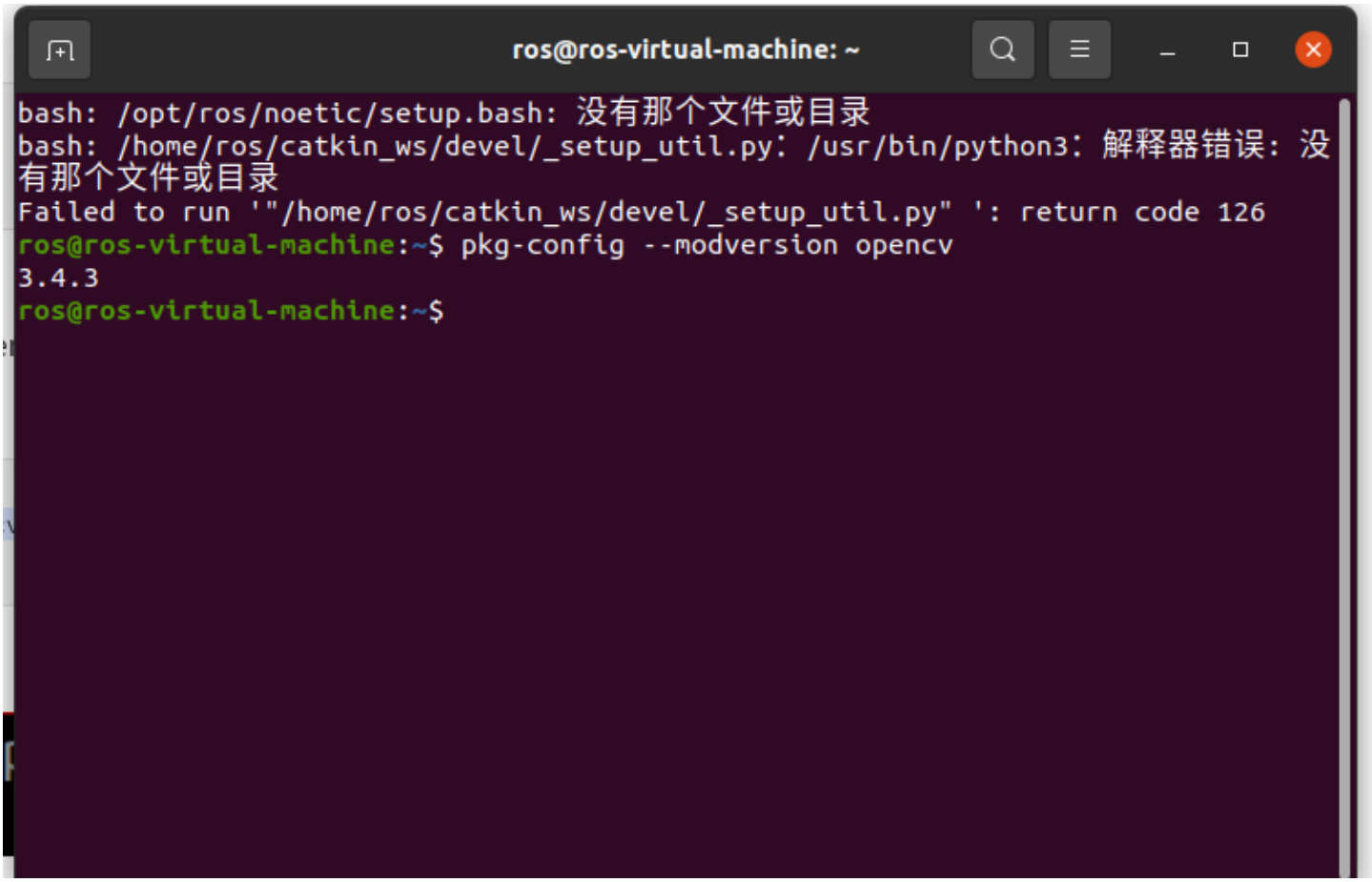
```
sudo updatedb
```

3.3.4.4 版本检测

安装好之后，我们肯定要对自己安装的OpenCV版本进行检测

输入下面的代码，查询OpenCV版本

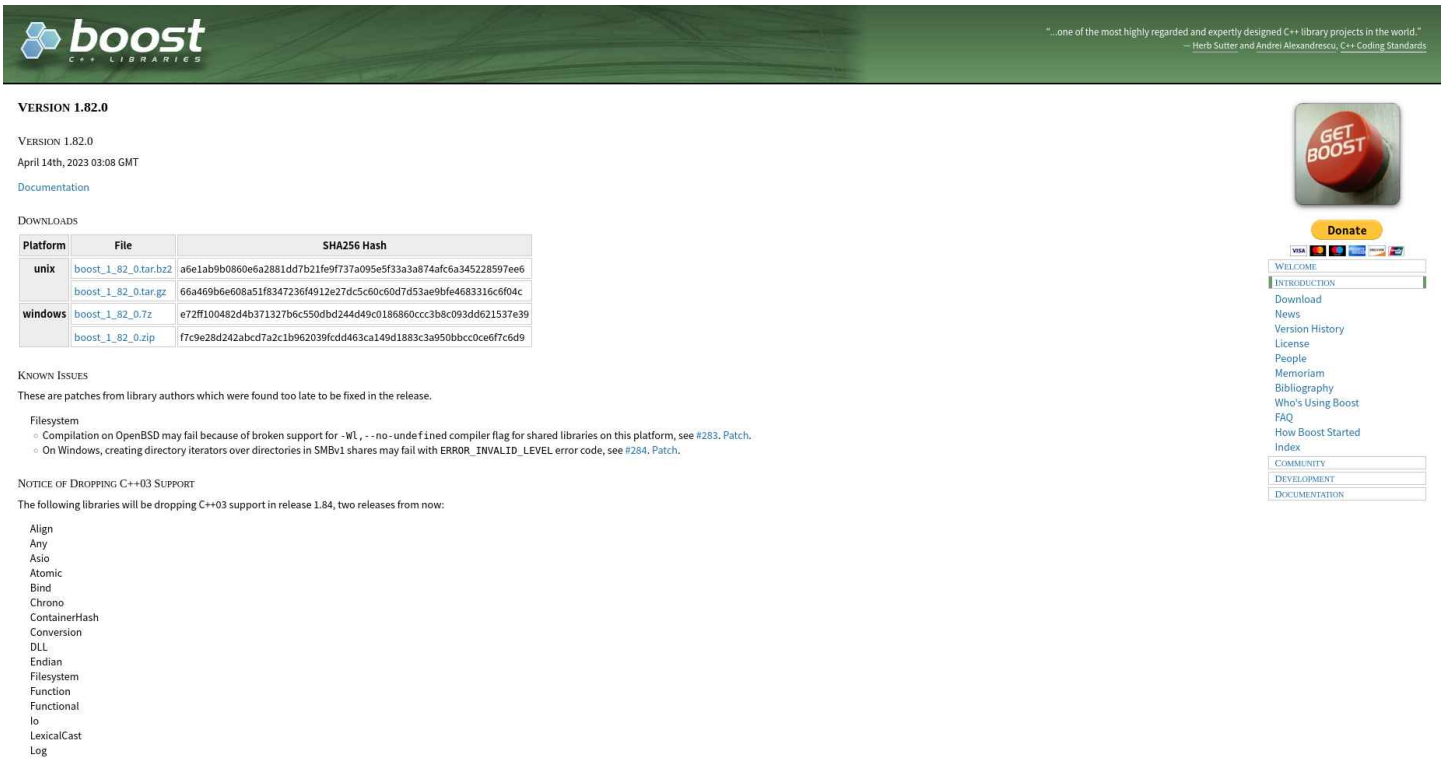
```
pkg-config --modversion opencv
```



3.4 安装 boost 库

前往boost库官网进行下载，这里，笔者下载的是1.82.0版本

boost官网地址：https://www.boost.org/



下载之后，我们需要将下载的文件放在 homework_exercise 文件夹下并提取解压

解压后我们进入解压出来的文件夹，执行下面的指令

```
1 sudo ./bootstrap.sh
```

文件夹此时多了一些文件，我们再执行下面这个脚本

```
1 sudo ./b2 install
```

3.5 安装 libssl-dev

在 homework_exercise 文件夹下打开终端，输入下列命令行

```
1 sudo apt-get install libssl-dev
```

3.6 ORB-SLAM3 的编译和安装

3.6.1 ORB-SLAM3 编译

在 homework_exercise 文件夹下打开终端，输入下列命令行

```
```

```
cd ORB_SLAM3-master
chmod +x build.sh
```

打开ORB_SLAM3-master对应的CMakeLists.txt
找到 find_package(OpenCV 4.4)这行代码，将OpenCV版本号改为find_package(OpenCV 3.4)

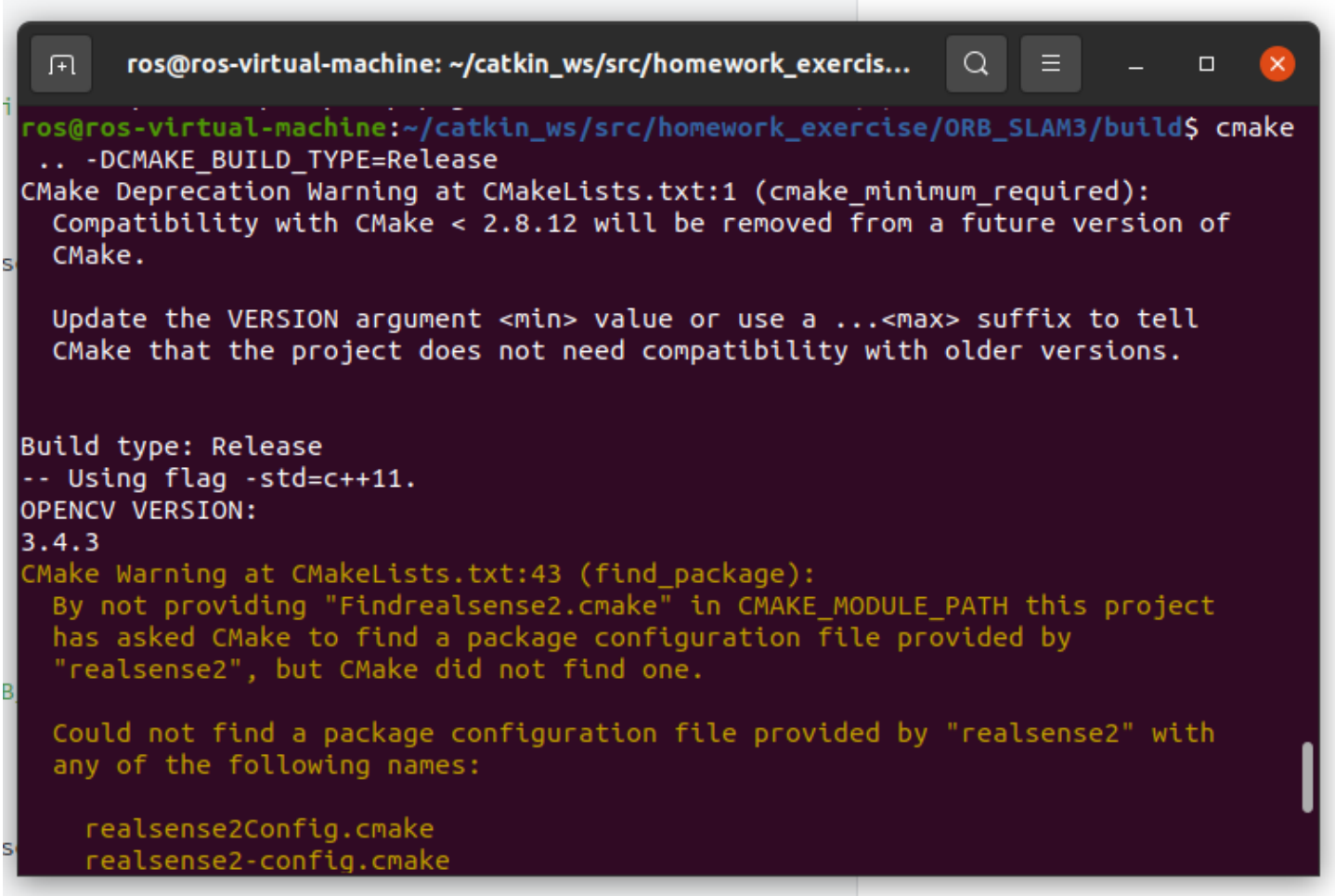
3.6.2 安装 ORB-SLAM3

逐行依次执行下面的指令

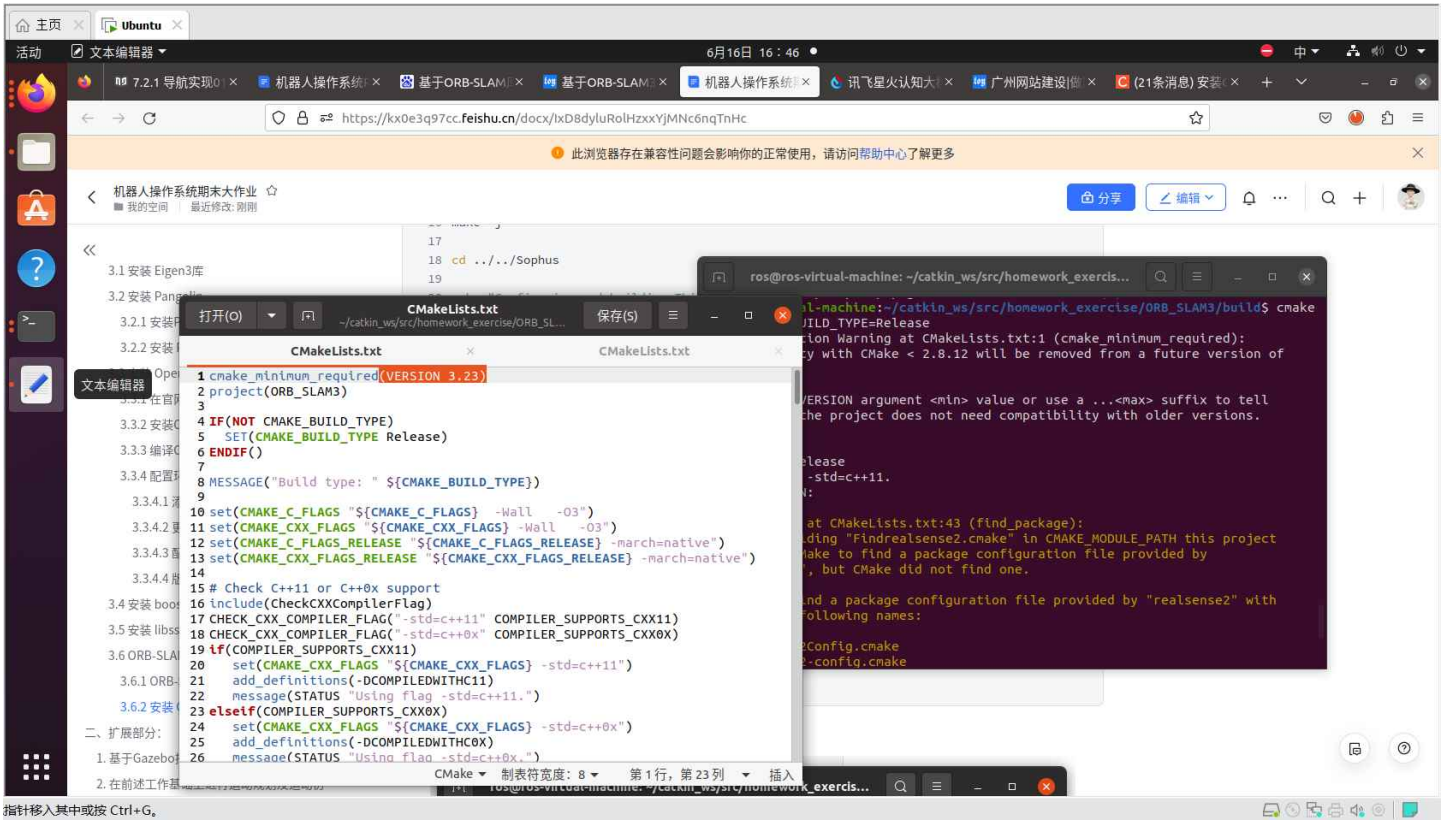
```
echo "Configuring and building Thirdparty/DBoW2 ..."

cd Thirdparty/DBoW2
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make -j

cd ../../g2o

echo "Configuring and building Thirdparty/g2o ..."

mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make -j

cd ../../Sophus

echo "Configuring and building Thirdparty/Sophus ..."

mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make -j

cd ../../../

echo "Uncompress vocabulary ..."

cd Vocabulary
tar -xf ORBvoc.txt.tar.gz
cd ..

echo "Configuring and building ORB_SLAM3 ..."
```

```
36
37 mkdir build
38 cd build
39 cmake .. -DCMAKE_BUILD_TYPE=Release
40 make -j5
```

在执行到最后一个模块时，笔者报错



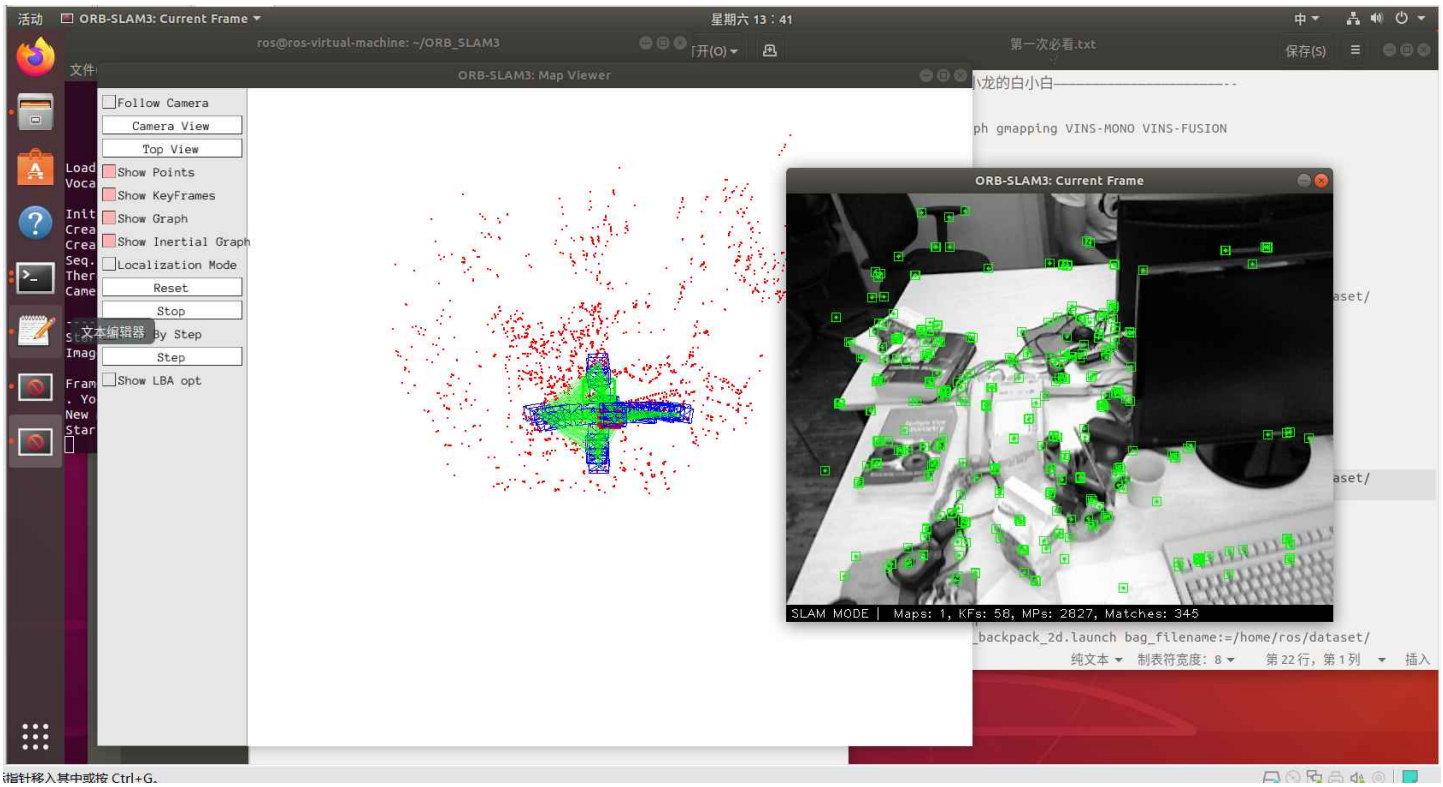
报错信息显示，我们需要将realsense2添加到路径中



4. 例程效果图

打开 ORB_SLAM3 文件夹，并输入以下指令，跑例程数据集

```
./Examples/RGB-D/rgbd_tum Vocabulary/ORBvoc.txt Examples/RGB-D/TUM1.yaml
~/dataset/rgbd_dataset_freiburg1_xyz/
~/dataset/rgbd_dataset_freiburg1_xyz/associations.txt
```



二、扩展部分：

1. 基于Gazebo搭建移动机器人，并结合SLAM系统进行定位和建图仿真；

1.1 创建功能包，导入依赖项

创建新功能包，导入依赖包：urdf、xacro、gazebo_ros、gazebo_ros_control、gazebo_plugins

1.2 编写URDF或Xacro文件

1.2.1 编写封装惯性矩阵算法的xacro文件

```
<robot name="mycar" xmlns:xacro="http://wiki.ros.org/xacro">
    <!--包含惯性矩阵文件-->
    <xacro:include filename="head.xacro" />
    <!--包含底盘、摄像头与雷达的xacro文件-->
    <xacro:include filename="demo05_car_base.urdf.xacro" />
    <xacro:include filename="demo06_car_camera.urdf.xacro" />
    <xacro:include filename="demo07_car_laser.urdf.xacro" />
    <!--运动控制-->
    <xacro:include filename="gazebo/move.xacro" />
    <!--laser lindar-->
    <xacro:include filename="gazebo/laser.xacro" />
    <!--camera-->
    <xacro:include filename="gazebo/camera.xacro" />
    <!--kinect-->
    <xacro:include filename="gazebo/kinect.xacro" />

</robot>


```

搭建机器人模型

```
<!--
    创建一个机器人模型(盒状即可)，显示在 Gazebo 中
-->

<robot name="mycar">
    <link name="base_link">
        <!--可视化部分-->
        <visual>
            <geometry>
```

```
                <box size="0.5 0.2 0.1" />
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
            <material name="yellow">
                <color rgba="0.5 0.3 0.0 1" />
            </material>
        </visual>

        <!--设置碰撞参数-->
        <collision>
            <geometry>
                <box size="0.5 0.2 0.1" />
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
        </collision>

        <!--设置惯性矩阵-->
        <inertial>
            <origin xyz="0 0 0" />
            <mass value="6" />
            <inertia ixx="1" ixy="0" ixz="0" iyy="1" iyz="0" izz="1" />
        </inertial>


    </link>

    <!--gazebo自己的颜色标签-->
    <gazebo reference="base_link">
        <material>Gazebo/Black</material>
    </gazebo>

</robot>
```

1.2.2 编写模型相关xacro文件，并设置collision inertial以及color等相关参数

1.2.2.1 底盘Xacro文件

```
<!--
    使用 xacro 优化 URDF 版的小车底盘实现:

    实现思路:
    1.将一些常量、变量封装为 xacro:property
      比如:PI 值、小车底盘半径、离地间距、车轮半径、宽度 ....
    2.使用 宏 封装驱动轮以及支撑轮实现，调用相关宏生成驱动轮与支撑轮

```

```
-->
<!-- 根标签，必须声明 xmlns:xacro -->
<robot name="my_base" xmlns:xacro="http://www.ros.org/wiki/xacro">
    <!-- 封装变量、常量 -->
    <!-- PI 值设置精度需要高一些，否则后续车轮翻转量计算时，可能会出现肉眼不能察觉的车轮倾
    <xacro:property name="PI" value="3.1415926"/>
    <!-- 宏:黑色设置 -->
    <material name="black">
        <color rgba="0.0 0.0 0.0 1.0" />
    </material>
    <!-- 底盘属性 -->
    <xacro:property name="base_footprint_radius" value="0.001" /> <!-- base_foot
    <xacro:property name="base_link_radius" value="0.1" /> <!-- base_link 半径 --
    <xacro:property name="base_link_length" value="0.08" /> <!-- base_link 长 --
    <xacro:property name="earth_space" value="0.015" /> <!-- 离地间距 -->
    <xacro:property name="base_link_m" value="0.5" /> <!-- 质量  -->

    <!-- 底盘 -->
    <link name="base_footprint">
      <visual>
        <geometry>
          <sphere radius="${base_footprint_radius}" />
        </geometry>
      </visual>
    </link>

    <link name="base_link">
      <visual>
        <geometry>
          <cylinder radius="${base_link_radius}" length="${base_link_length}" />
        </geometry>
        <origin xyz="0 0 0" rpy="0 0 0" />
        <material name="yellow">
          <color rgba="0.5 0.3 0.0 0.5" />
        </material>
      </visual>
      <collision>
        <geometry>
          <cylinder radius="${base_link_radius}" length="${base_link_length}" />
        </geometry>
        <origin xyz="0 0 0" rpy="0 0 0" />
      </collision>
      <xacro:cylinder_inertial_matrix m="${base_link_m}" r="${base_link_radius}"

    </link>


```

```
    <joint name="base_link2base_footprint" type="fixed">
      <parent link="base_footprint" />
      <child link="base_link" />
      <origin xyz="0 0 ${earth_space + base_link_length / 2 }" />
    </joint>
    <gazebo reference="base_link">
        <material>Gazebo/Yellow</material>
    </gazebo>

    <!-- 驱动轮 -->
    <!-- 驱动轮属性 -->
    <xacro:property name="wheel_radius" value="0.0325" /><!-- 半径 -->
    <xacro:property name="wheel_length" value="0.015" /><!-- 宽度 -->
    <xacro:property name="wheel_m" value="0.05" /> <!-- 质量  -->

    <!-- 驱动轮宏实现 -->
    <xacro:macro name="add_wheels" params="name flag">
      <link name="${name}_wheel">
        <visual>
          <geometry>
            <cylinder radius="${wheel_radius}" length="${wheel_length}" />
          </geometry>
          <origin xyz="0.0 0.0 0.0" rpy="${PI / 2} 0.0 0.0" />
          <material name="black" />
        </visual>
        <collision>
          <geometry>
            <cylinder radius="${wheel_radius}" length="${wheel_length}" />
          </geometry>
          <origin xyz="0.0 0.0 0.0" rpy="${PI / 2} 0.0 0.0" />
        </collision>
        <xacro:cylinder_inertial_matrix m="${wheel_m}" r="${wheel_radius}" h="${

      </link>

      <joint name="${name}_wheel2base_link" type="continuous">
        <parent link="base_link" />
        <child link="${name}_wheel" />
        <origin xyz="0 ${flag * base_link_radius} ${-(earth_space + base_link_le
        <axis xyz="0 1 0" />
      </joint>

      <gazebo reference="${name}_wheel">
        <material>Gazebo/Red</material>
      </gazebo>

    </xacro:macro>
```

```
103     <xacro:add_wheels name="left" flag="1" />
104     <xacro:add_wheels name="right" flag="-1" />
105     <!-- 支撑轮 -->
106     <!-- 支撑轮属性 -->
107     <xacro:property name="support_wheel_radius" value="0.0075" /> <!-- 支撑轮半径
108     <xacro:property name="support_wheel_m" value="0.03" /> <!-- 质量  -->
109 
110     <!-- 支撑轮宏 -->
111     <xacro:macro name="add_support_wheel" params="name flag" >
112       <link name="${name}_wheel">
113         <visual>
114             <geometry>
115                 <sphere radius="${support_wheel_radius}" />
116             </geometry>
117             <origin xyz="0 0 0" rpy="0 0 0" />
118             <material name="black" />
119         </visual>
120         <collision>
121             <geometry>
122                 <sphere radius="${support_wheel_radius}" />
123             </geometry>
124             <origin xyz="0 0 0" rpy="0 0 0" />
125         </collision>
126         <xacro:sphere_inertial_matrix m="${support_wheel_m}" r="${support_wheel_
127       </link>
128 
129       <joint name="${name}_wheel2base_link" type="continuous">
130           <parent link="base_link" />
131           <child link="${name}_wheel" />
132           <origin xyz="${flag * (base_link_radius - support_wheel_radius)} 0 ${-
133           <axis xyz="1 1 1" />
134       </joint>
135       <gazebo reference="${name}_wheel">
136         <material>Gazebo/Red</material>
137       </gazebo>
138     </xacro:macro>
139 
140     <xacro:add_support_wheel name="front" flag="1" />
141     <xacro:add_support_wheel name="back" flag="-1" />
142 
143 
144 </robot>
145 
```

1.2.2.2 摄像头Xacro文件

```
<!-- 摄像头相关的 xacro 文件 -->
<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_heigh

    <xacro:property name="camera_m" value="0.01" /> <!-- 摄像头质量 -->

    <!-- 摄像头关节以及link -->
    <link name="camera">
        <visual>
            <geometry>
                <box size="${camera_length} ${camera_width} ${camera_height}" />
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
            <material name="black" />
        </visual>
        <collision>
            <geometry>
                <box size="${camera_length} ${camera_width} ${camera_height}" />
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
        </collision>
        <xacro:Box_inertial_matrix m="${camera_m}" l="${camera_length}" w="${cam
    </link>

    <joint name="camera2base_link" type="fixed">
        <parent link="base_link" />
        <child link="camera" />
        <origin xyz="${camera_x} ${camera_y} ${camera_z}" />
    </joint>
    <gazebo reference="camera">
        <material>Gazebo/Blue</material>
    </gazebo>
</robot>

```

1.2.2.3 雷达Xacro文件

```
<!--
    小车底盘添加雷达
```

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

    <!-- 雷达支架 -->
    <xacro:property name="support_length" value="0.15" /> <!-- 支架长度 -->
    <xacro:property name="support_radius" value="0.01" /> <!-- 支架半径 -->
    <xacro:property name="support_x" value="0.0" /> <!-- 支架安装的x坐标 -->
    <xacro:property name="support_y" value="0.0" /> <!-- 支架安装的y坐标 -->
    <xacro:property name="support_z" value="${base_link_length / 2 + support_len

    <xacro:property name="support_m" value="0.02" /> <!-- 支架质量 -->

    <link name="support">
        <visual>
            <geometry>
                <cylinder radius="${support_radius}" length="${support_length}"
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
            <material name="red">
                <color rgba="0.8 0.2 0.0 0.8" />
            </material>
        </visual>

        <collision>
            <geometry>
                <cylinder radius="${support_radius}" length="${support_length}"
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
        </collision>

        <xacro:cylinder_inertial_matrix m="${support_m}" r="${support_radius}" h

    </link>

    <joint name="support2base_link" type="fixed">
        <parent link="base_link" />
        <child link="support" />
        <origin xyz="${support_x} ${support_y} ${support_z}" />
    </joint>

    <gazebo reference="support">
        <material>Gazebo/White</material>
    </gazebo>

    <!-- 雷达属性 -->
    <xacro:property name="laser_length" value="0.05" /> <!-- 雷达长度 -->
    <xacro:property name="laser_radius" value="0.03" /> <!-- 雷达半径 -->
```

```
    <xacro:property name="laser_x" value="0.0" /> <!-- 雷达安装的x坐标 -->
    <xacro:property name="laser_y" value="0.0" /> <!-- 雷达安装的y坐标 -->
    <xacro:property name="laser_z" value="${support_length / 2 + laser_length /

    <xacro:property name="laser_m" value="0.1" /> <!-- 雷达质量 -->

    <!-- 雷达关节以及link -->
    <link name="laser">
        <visual>
            <geometry>
                <cylinder radius="${laser_radius}" length="${laser_length}" />
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
            <material name="black" />
        </visual>
        <collision>
            <geometry>
                <cylinder radius="${laser_radius}" length="${laser_length}" />
            </geometry>
            <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
        </collision>
        <xacro:cylinder_inertial_matrix m="${laser_m}" r="${laser_radius}" h="${
    </link>

    <joint name="laser2support" type="fixed">
        <parent link="support" />
        <child link="laser" />
        <origin xyz="${laser_x} ${laser_y} ${laser_z}" />
    </joint>
    <gazebo reference="laser">
        <material>Gazebo/Black</material>
    </gazebo>
</robot>

```

1.2.2.4 组合底盘、摄像头与雷达的Xacro文件

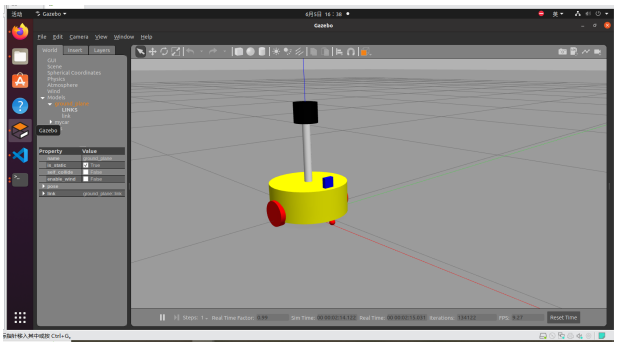
```
<!-- 组合小车底盘与摄像头 -->
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <xacro:include filename="my_head.urdf.xacro" />
    <xacro:include filename="my_base.urdf.xacro" />
    <xacro:include filename="my_camera.urdf.xacro" />
    <xacro:include filename="my_laser.urdf.xacro" />
```

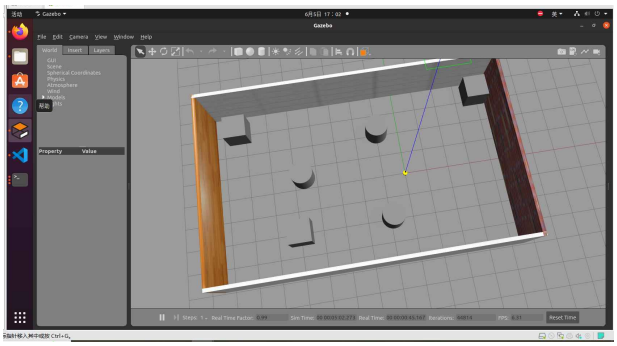
```
</robot>

```

1.3 启动Gazebo并显示机器人模型

编写launch文件实现

```
<launch>
    <!-- 将 Urdf 文件的内容加载到参数服务器 -->
    <param name="robot_description" command="$(find xacro)/xacro $(find demo02_u
    <!-- 启动 gazebo -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch" />

    <!-- 在 gazebo 中显示机器人模型 -->
    <node pkg="gazebo_ros" type="spawn_model" name="model" args="-urdf -model my
</launch>

```





2. 在前述工作基础上进行运动规划及运动仿真。

2.1 运动控制及里程计信息显示

2.1.1 添加机器人传动装置

两轮差速配置

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

    <!-- 传动实现:用于连接控制器与关节 -->
    <xacro:macro name="joint_trans" params="joint_name">
        <!-- Transmission is important to link the joints and the controller -->
```

```
        <transmission name="${joint_name}_trans">
            <type>transmission_interface/SimpleTransmission</type>
            <joint name="${joint_name}">
                <hardwareInterface>hardware_interface/VelocityJointInterface</ha
            </joint>
            <actuator name="${joint_name}_motor">
                <hardwareInterface>hardware_interface/VelocityJointInterface</ha
                <mechanicalReduction>1</mechanicalReduction>
            </actuator>
        </transmission>
    </xacro:macro>

    <!-- 每一个驱动轮都需要配置传动装置 -->
    <xacro:joint_trans joint_name="left_wheel2base_link" />
    <xacro:joint_trans joint_name="right_wheel2base_link" />

    <!-- 控制器 -->
    <gazebo>
        <plugin name="differential_drive_controller" filename="libgazebo_ros_dif
            <rosDebugLevel>Debug</rosDebugLevel>
            <publishWheelTF>true</publishWheelTF>
            <robotNamespace>/</robotNamespace>
            <publishTf>1</publishTf>
            <publishWheelJointState>true</publishWheelJointState>
            <alwaysOn>true</alwaysOn>
            <updateRate>100.0</updateRate>
            <legacyMode>true</legacyMode>
            <leftJoint>left_wheel2base_link</leftJoint> <!-- 左轮 -->
            <rightJoint>right_wheel2base_link</rightJoint> <!-- 右轮 -->
            <wheelSeparation>${base_link_radius * 2}</wheelSeparation> <!-- 车轮i
            <wheelDiameter>${wheel_radius * 2}</wheelDiameter> <!-- 车轮直径 -->
            <broadcastTF>1</broadcastTF>
            <wheelTorque>30</wheelTorque>
            <wheelAcceleration>1.8</wheelAcceleration>
            <commandTopic>cmd_vel</commandTopic> <!-- 运动控制话题 -->
            <odometryFrame>odom</odometryFrame>
            <odometryTopic>odom</odometryTopic> <!-- 里程计话题 -->
            <robotBaseFrame>base_footprint</robotBaseFrame> <!-- 根坐标系 -->
        </plugin>
    </gazebo>

</robot>

```

2.1.2 xacro文件集成

```
<!-- 组合小车底盘与摄像头 -->
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <xacro:include filename="my_head.urdf.xacro" />
    <xacro:include filename="my_base.urdf.xacro" />
    <xacro:include filename="my_camera.urdf.xacro" />
    <xacro:include filename="my_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
</robot>

```

注意，但前核心包含控制器以及传动配置的xacro文件

```
<xacro:include filename="move.urdf.xacro" />
```

2.1.3 启动gazebo控制机器人运动

```
<launch>

    <!-- 将 Urdf 文件的内容加载到参数服务器 -->
    <param name="robot_description" command="$(find xacro)/xacro $(find demo02_u
    <!-- 启动 gazebo -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="world_name" value="$(find demo02_urdf_gazebo)/worlds/hello.wo
    </include>

    <!-- 在 gazebo 中显示机器人模型 -->
    <node pkg="gazebo_ros" type="spawn_model" name="model" args="-urdf -model my
</launch>

```

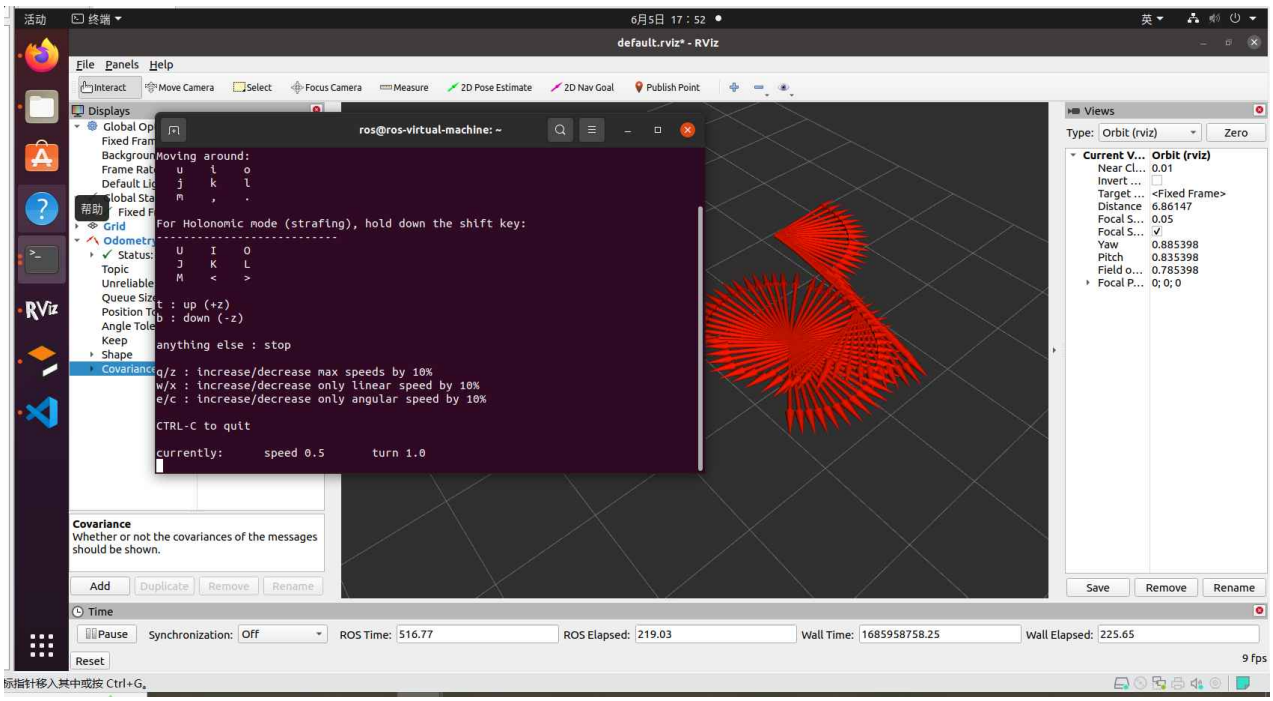
输入下面的命令行，可以使用命令控制

```
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.2, y: 0, z: 0},
```

2.1.4 Rviz查看里程计信息

```
<launch>
    <!-- 启动 rviz -->
```

```
    <node pkg="rviz" type="rviz" name="rviz" />

    <!-- 关节以及机器人状态发布节点 -->
    <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_s
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_s

</launch>

```



2.2 雷达信息仿真以及显示

2.2.1 新建xacro文件，配置雷达传感器信息

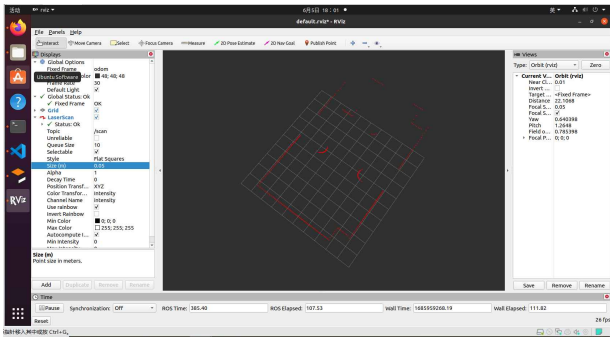
```
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">

  <!-- 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>
```

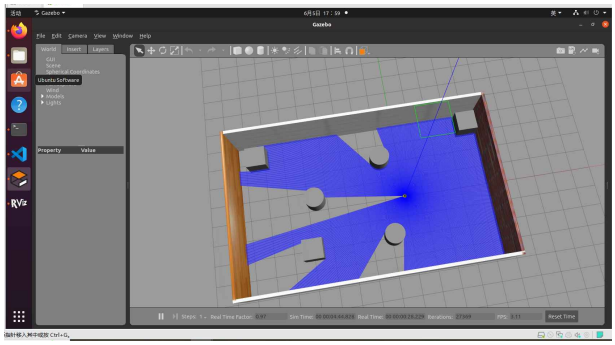
```
            <resolution>1</resolution>
            <min_angle>-3</min_angle>
            <max_angle>3</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.10</min>
          <max>30.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <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>

```

2.2.2 xacro文件集成

```
<!-- 组合小车底盘与传感器 -->
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <xacro:include filename="my_head.urdf.xacro" />
    <xacro:include filename="my_base.urdf.xacro" />
    <xacro:include filename="my_camera.urdf.xacro" />
    <xacro:include filename="my_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
    <!-- 雷达仿真的 xacro 文件 -->
    <xacro:include filename="my_sensors_laser.urdf.xacro" />
</robot>

```

2.2.3 编写launch文件，启动gazebo仿真环境





2.3 摄像头信息仿真及显示

2.3.1 新建xacro文件，配置摄像头传感器信息

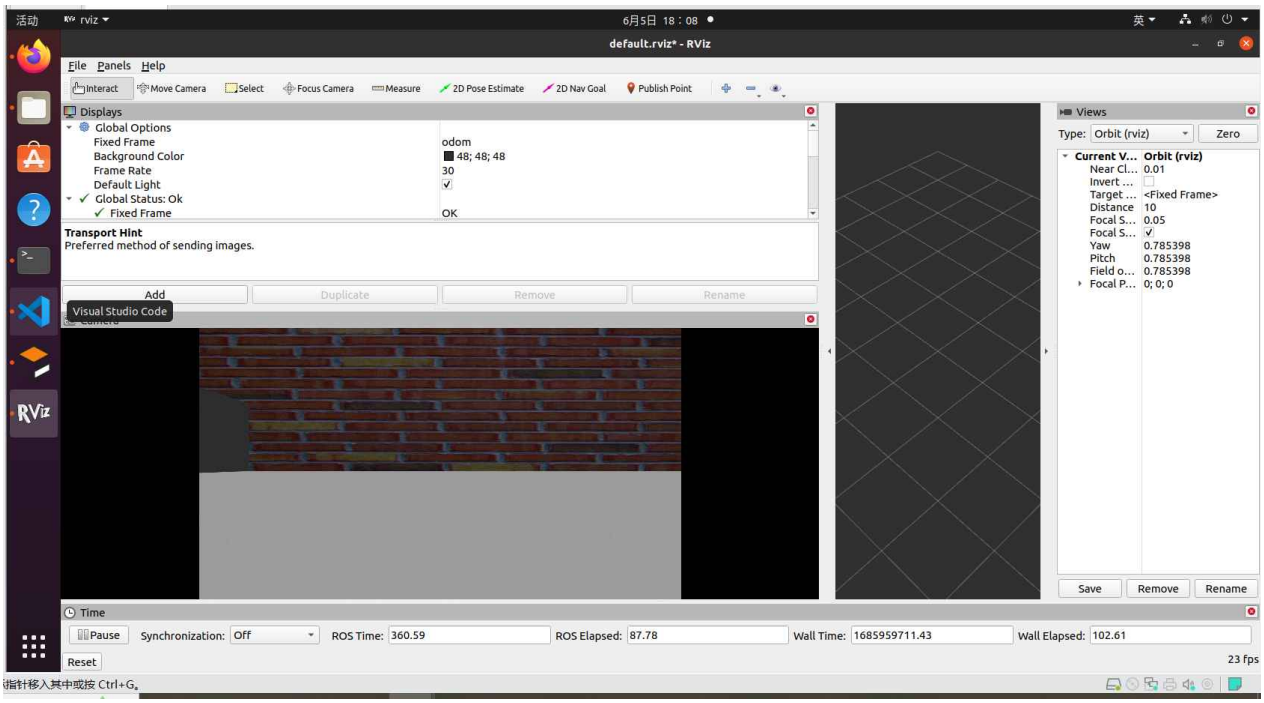
```
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">
  <!-- 被引用的link -->
  <gazebo reference="camera">
    <!-- 类型设置为 camara -->
    <sensor type="camera" name="camera_node">
      <update_rate>30.0</update_rate> <!-- 更新频率 -->
      <!-- 摄像头基本信息设置 -->
      <camera name="head">
        <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>
      <!-- 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>

```

2.3.2 xacro文件集成

```
<!-- 组合小车底盘与传感器 -->
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <xacro:include filename="my_head.urdf.xacro" />
    <xacro:include filename="my_base.urdf.xacro" />
    <xacro:include filename="my_camera.urdf.xacro" />
    <xacro:include filename="my_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
    <!-- 摄像头仿真的 xacro 文件 -->
    <xacro:include filename="my_sensors_camara.urdf.xacro" />
</robot>

```

2.3.3 编写launch文件，启动gazebo仿真环境



2.4 kinect信息仿真及显示

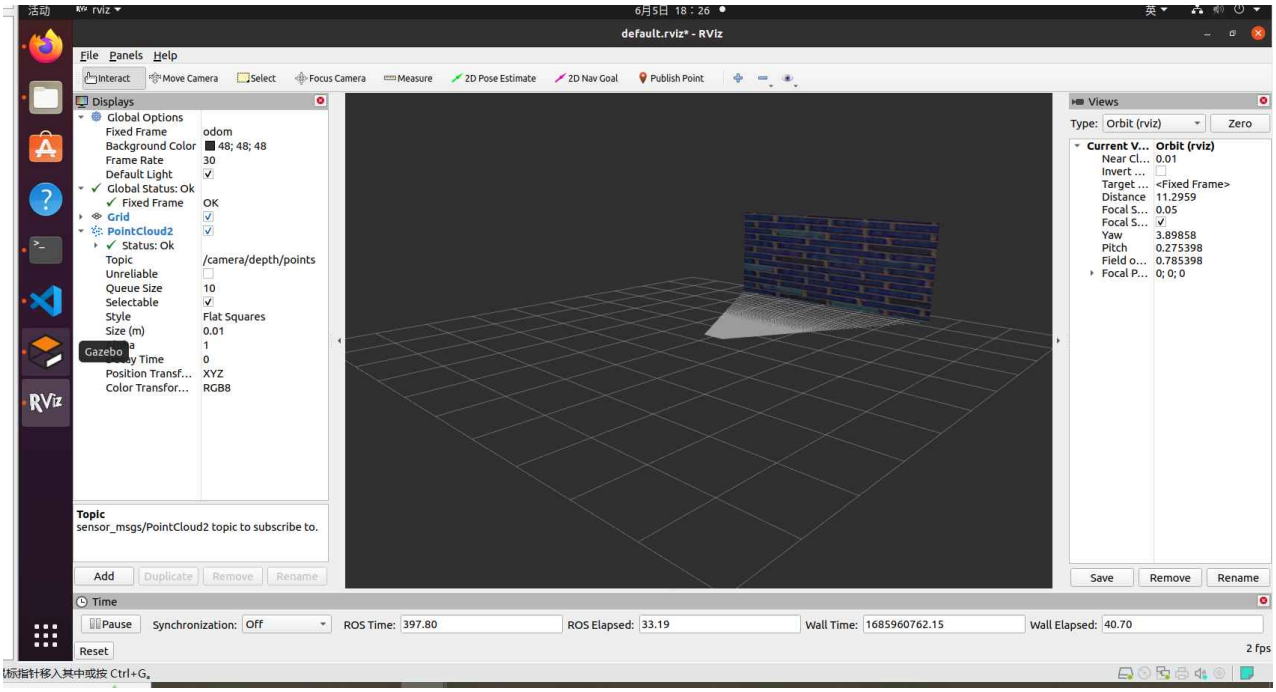
2.4.1 新建xacro文件，配置kinetic传感器信息

```
<robot name="my_sensors" xmlns:xacro="http://wiki.ros.org/xacro">
    <gazebo reference="kinect link名称">
      <sensor type="depth" name="camera">
        <always_on>true</always_on>
        <update_rate>20.0</update_rate>
        <camera>
          <horizontal_fov>${60.0*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_camera_controller" filename="libgazebo_ros_openni_k
          <cameraName>camera</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
          <frameName>kinect link名称</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>

</robot>

```

2.4.2 xacro文件集成

```
<!-- 组合小车底盘与传感器 -->
<robot name="my_car_camera" xmlns:xacro="http://wiki.ros.org/xacro">
    <xacro:include filename="my_head.urdf.xacro" />
    <xacro:include filename="my_base.urdf.xacro" />
    <xacro:include filename="my_camera.urdf.xacro" />
    <xacro:include filename="my_laser.urdf.xacro" />
    <xacro:include filename="move.urdf.xacro" />
    <!-- kinect仿真的 xacro 文件 -->
    <xacro:include filename="my_sensors_kinect.urdf.xacro" />
</robot>

```

2.4.3 编写launch文件，启动gazebo仿真环境



三、附录

1. git安装

```
sudo apt-get install git
```

2. vi编辑器安装

```
# 有些新系统的vi编辑器不完整，需要重新安装
sudo apt-get remove vim-common
sudo apt-get install vim
```

四、Q&A

这个部分是笔者自己做得时候遇到的一些通用问题，其余过程中遇到的问题文档相应位置，其他资料可参考本文第五部分“参考资料”。

📌
- 遇到问题好心态：不要怕，总能解决的，早晚的事情
- 解决问题的思路：查看报错信息 -> 在网站上搜索相关解决办法 -> 并不是所有问题都能够找到，但是能够找到相似的解决方法 -> 根据网上的提示解决问题

1. git clone

Q1：unable to access 'https://github.com/

这个问题可能是git配置的问题，也有可能是网络的问题，最暴力的解决办法是登陆具体的网址，直接下载代码文件，所以，

- 解决方案一：最直接的办法，直接登陆具体的网址，下载代码文件
- 解决方案二：查看git配置

输入下面的指令行，如果没有任何与https代理相关的内容，则没有问题

```
git config --global -l
```

如果有相关内容，请输入下列指令，并删除对应的内容，然后重试

```
~/ .gitconfig
```

- 解决方案三：手动配置git的代理

git客户端输入如下两个命令就可以了。

```
git config --global http.proxy http://127.0.0.1:1080
git config --global https.proxy http://127.0.0.1:1080
```

2. command

Q1:Command ‘make‘ not found, but can be installed...

这个指令是说没有找到对应指令集，当然也有可能是路径的问题，我们可以通过重新安装解决这个问题

比如我们可以依次执行下列指令

```
sudo apt-cdrom add
sudo apt-get update
sudo apt-get install build-essential
```

Q2：catkin_make:Command ‘catkin_make‘ not found, but can be installed with:sudo apt install catkin

同理上面，我们只需要依次执行下列指令行即可

```
source /opt/ros/melodic/setup.bash
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Q3：Linux环境下出现 cmake: command not found

这里，我们针对环境变量中缺少路径的情况提出解决办法

暂时添加环境变量关闭窗口后就失效了，不推荐，这里我们推荐永久添加环境变量（如下）

```
$ cd
$ touch .bash_profile
$ vim .bash_profile
export CMAKE_ROOT=/Applications/CMake.app/Contents/bin/
export PATH=$CMAKE_ROOT:$PATH
$ source .bash_profile
```

添加后，我们可以开一个新窗口，查看cmake版本号

```
$ cmake --version
```

五、参考资料

1. 基于ORB-SLAM3库搭建SLAM系统
2. git clone出现 fatal: unable to access 'https://github.com/...'的解决办法(亲测有效)
3. catkin_make:Command ‘catkin_make‘ not found, but can be installed with:sudo apt install catkin_Mr_D0
4. 报错Command ‘make‘ not found, but can be installed..._command make_Ham235XueBi的博客-CSDN博客
5. Mac/Linux环境下出现 cmake: command not found_songarpore的博客-CSDN博客
6. BUG List_Hangro的博客-CSDN博客（这篇文档记录了超级全的bug指南，推荐）