[【古月居】古月·ROS入门21讲 | 一学就会的ROS机器人入门教程\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1zt411G7Vn?spm_id_from=333.337.search-card.all.click)

首先推荐大家先学习这门课程，对ROS系统有一个简要的认识，课程时间很短，有助快速了解。

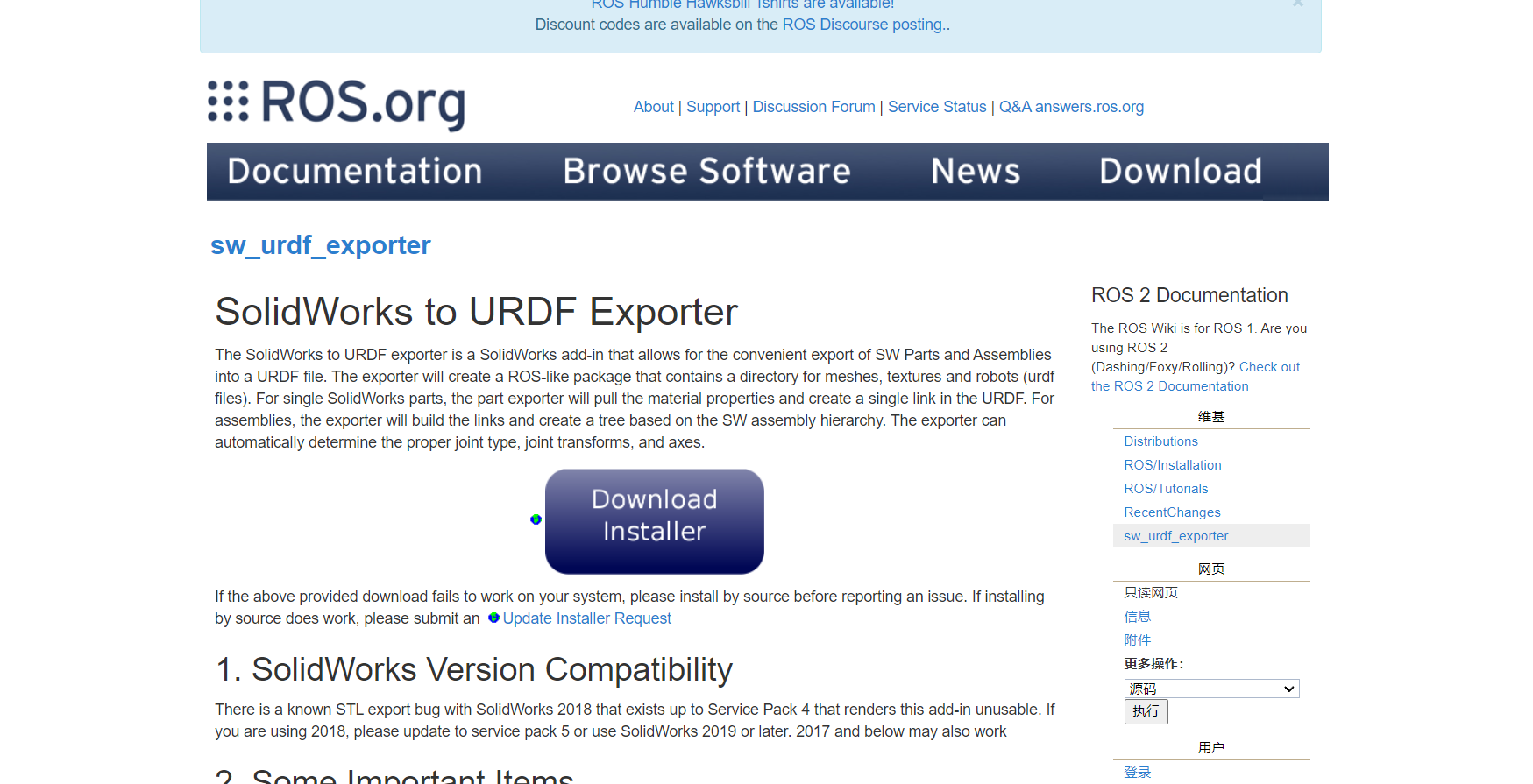
所需设备：Ubuntu18.04 以及配置好的ROS系统

# 利用SW建立机器人模型

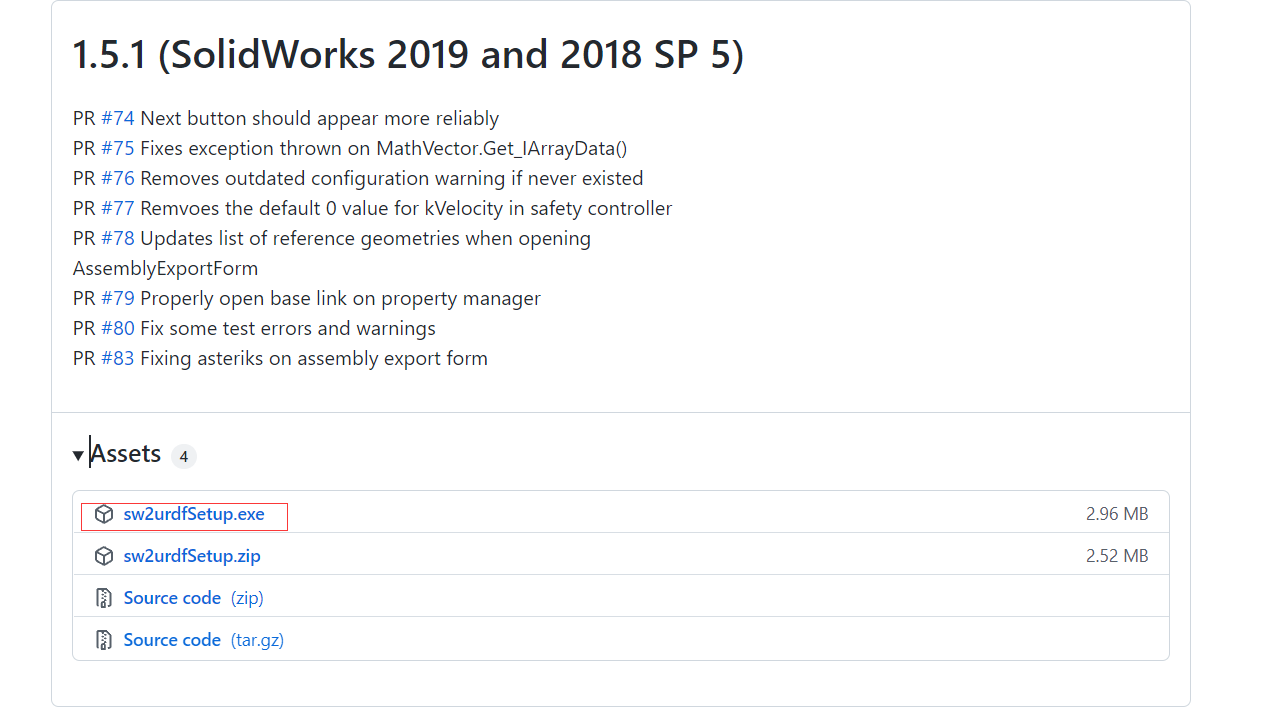
推荐使用SW2016版本，18版本会出现模型导出错位现象等BUG。

[wiki.ros.org/sw\_urdf\_exporter](http://wiki.ros.org/sw_urdf_exporter)

进入网址下载SW TO URDF插件

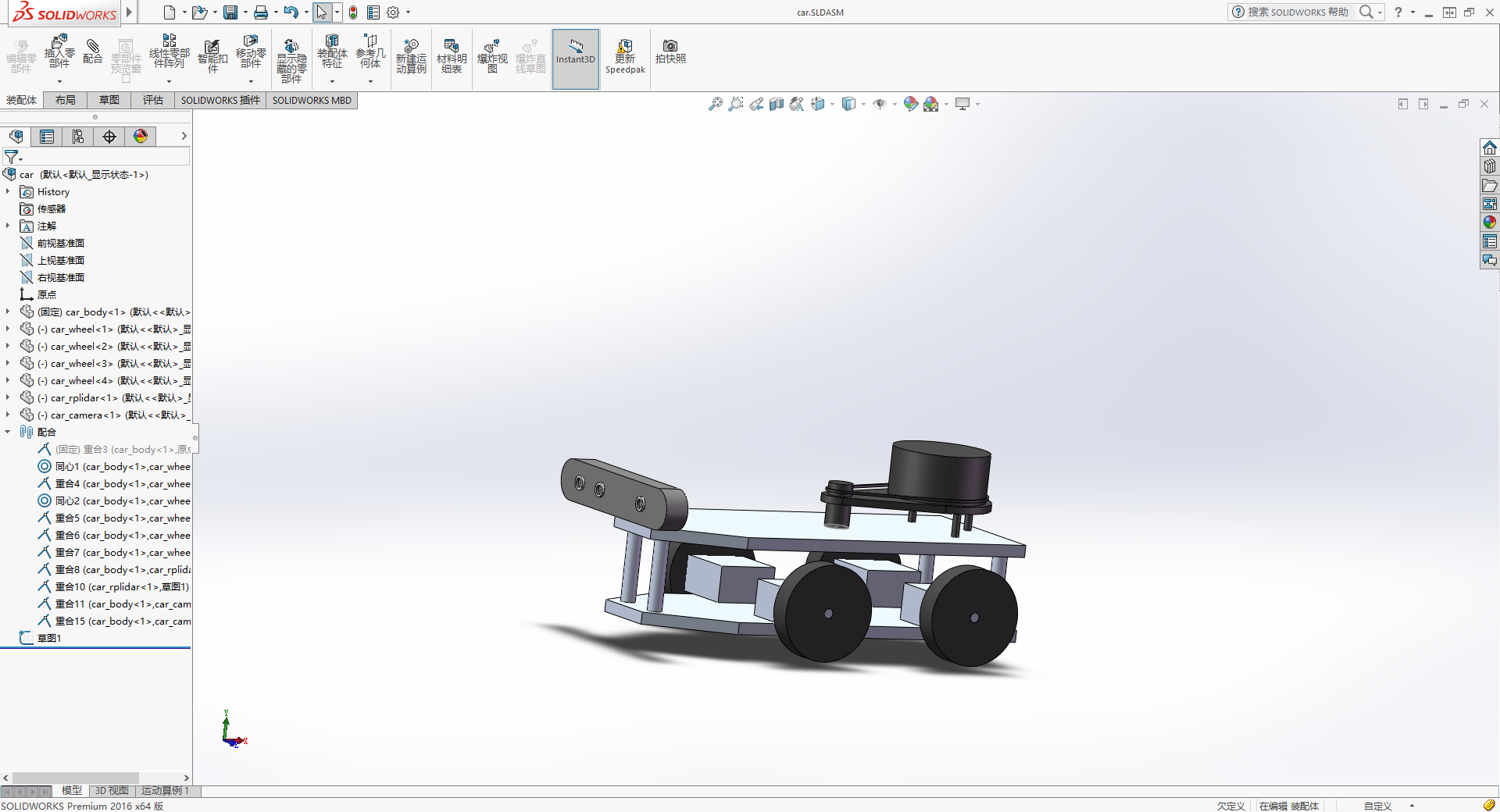


点击 download installer 下载并安装



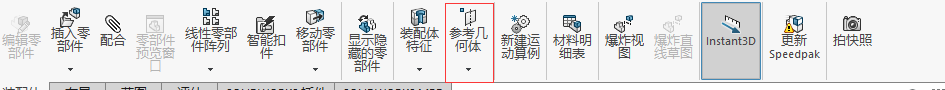
安装这个版本即可完美适配SW2016。

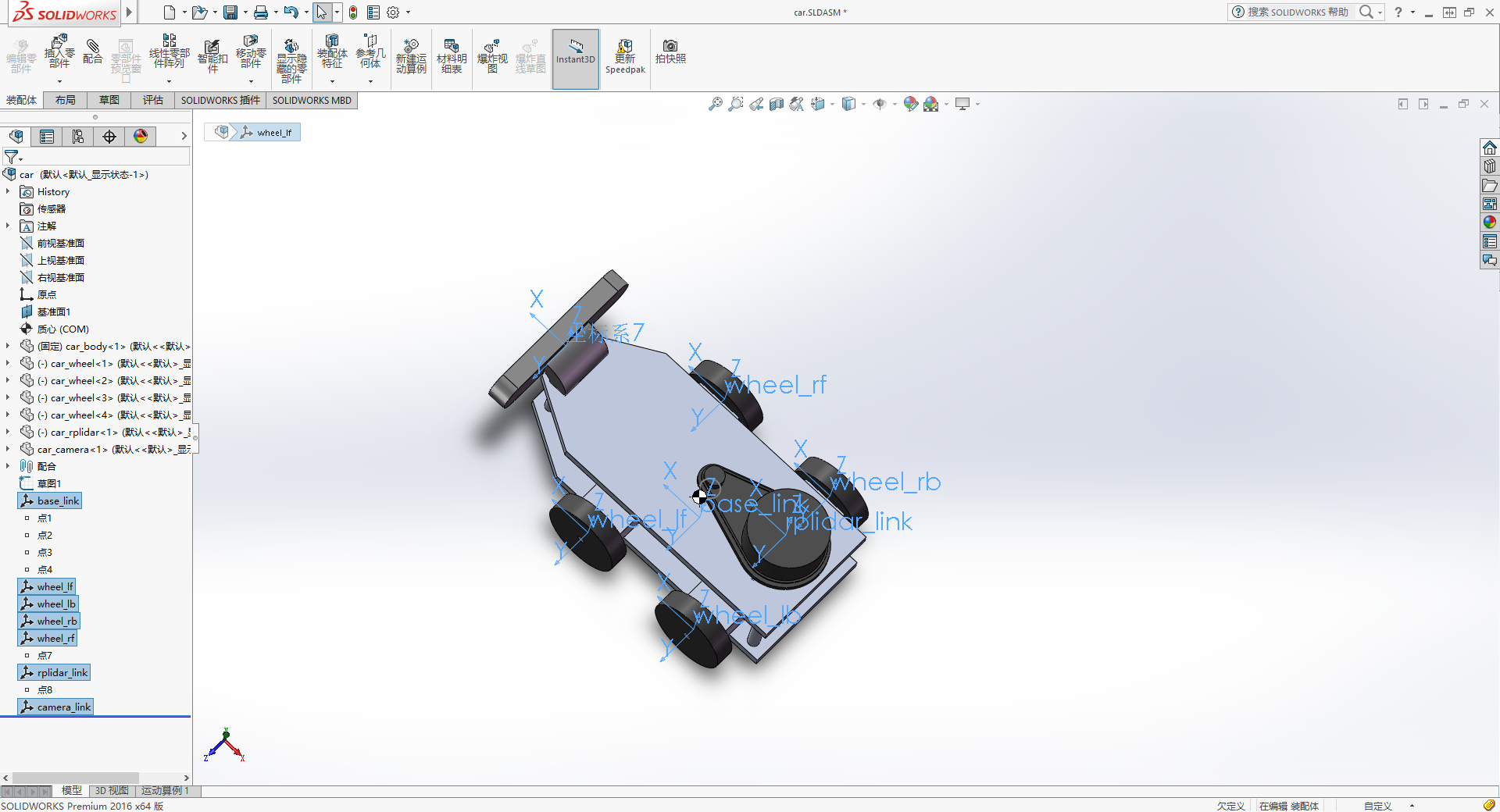
安装好后进入SW建立机器人模型



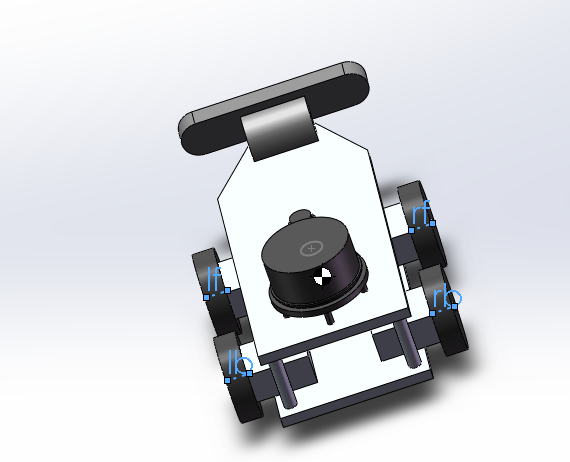
建模过程省略，建立一个四轮差速模型即可，无需一摸一样（激光雷达建立圆柱即可，无需像我一样）

建立好模型后为每一个link（零件）建立独立坐标系，X轴方向指向机器人前进方向,Z轴向上，右手定则确定Y轴（指向左转方向）

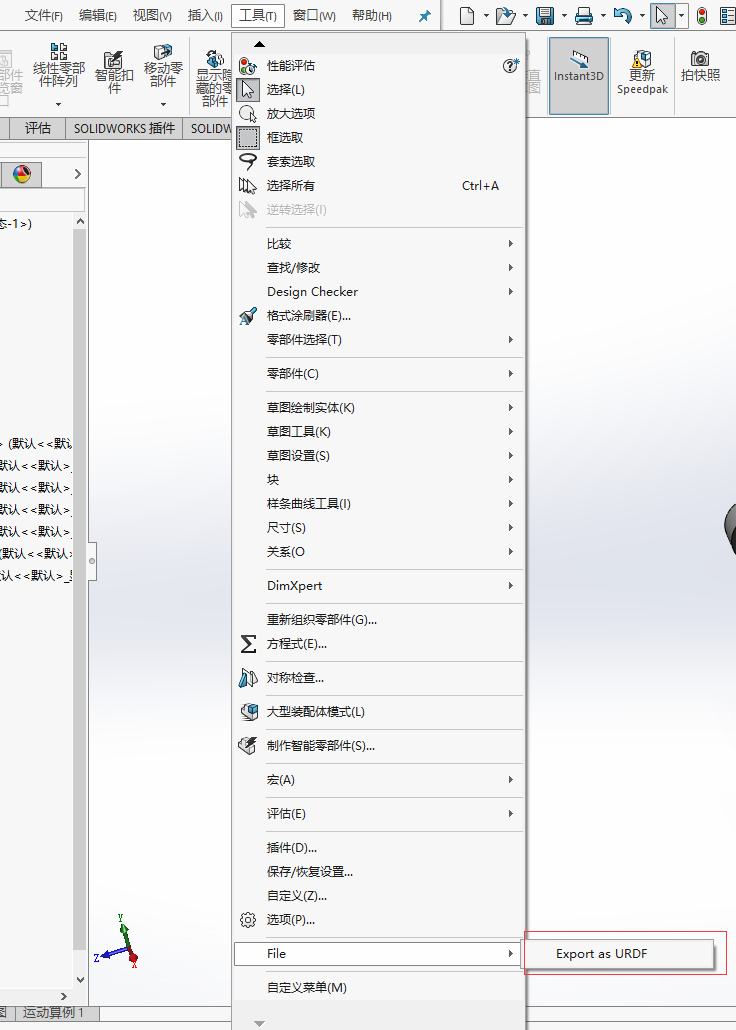


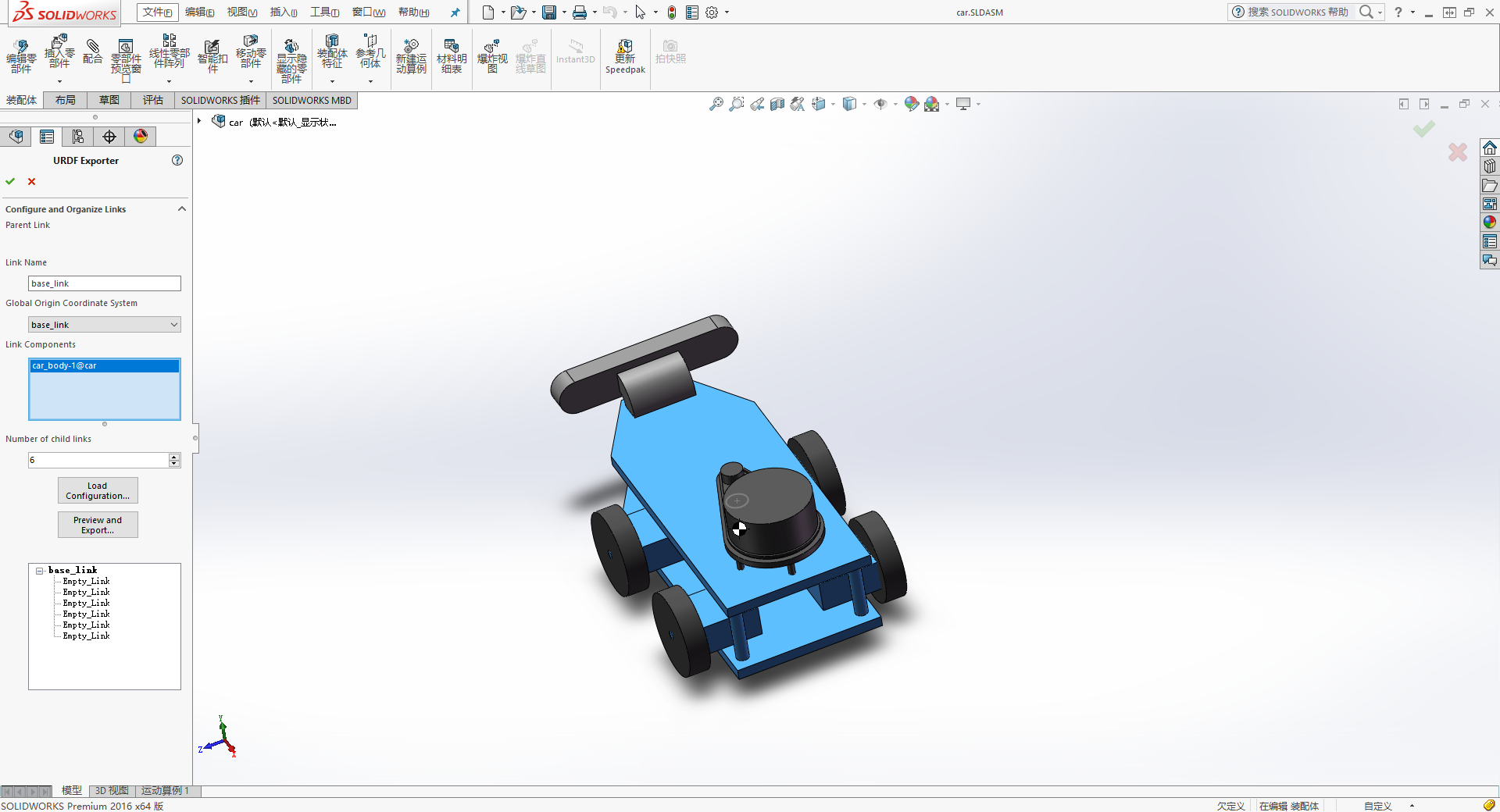


并为每个车轮设置旋转轴



点击如图选项



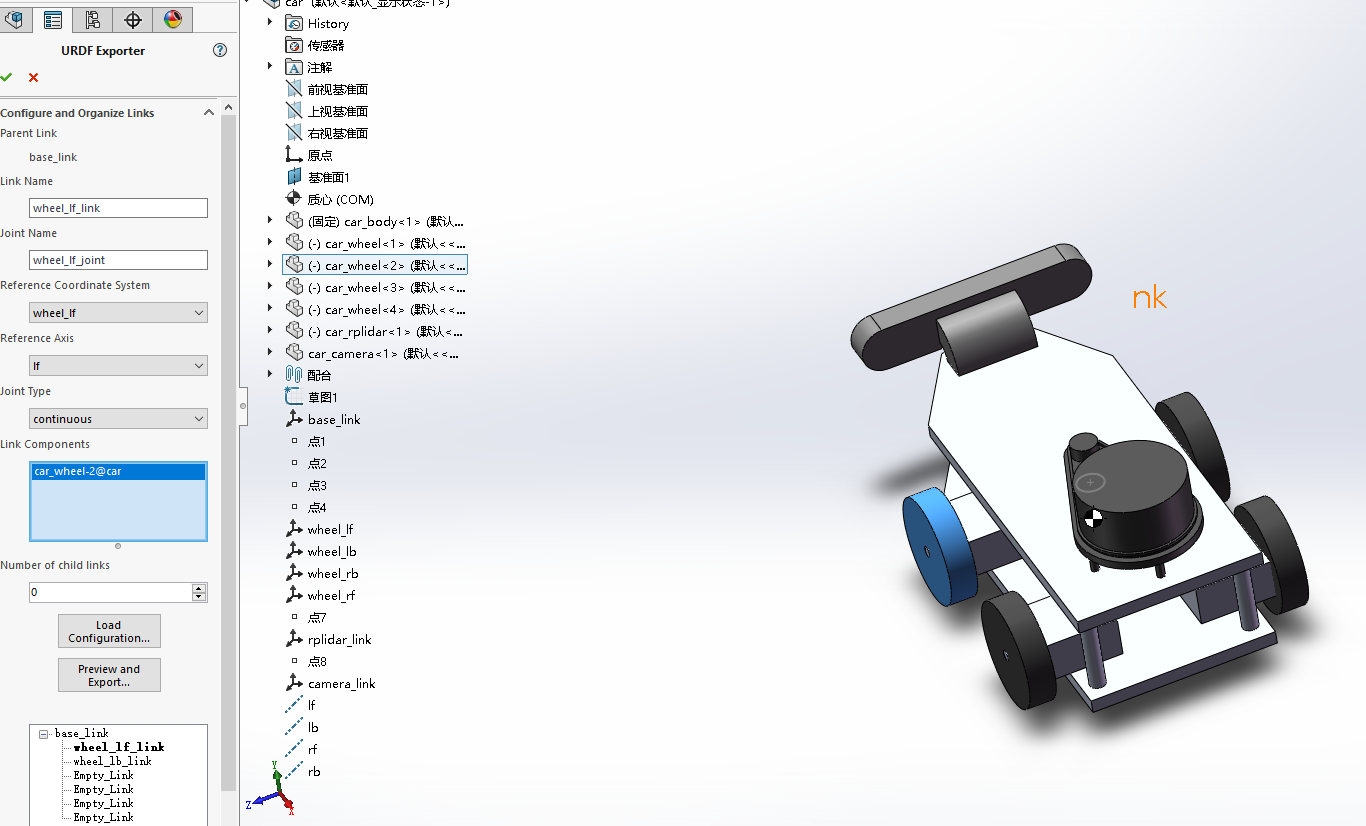


Link\_name：自己设置，为urdf模型文件代码中link名

Global Origin Coordinate System：选择相应坐标系

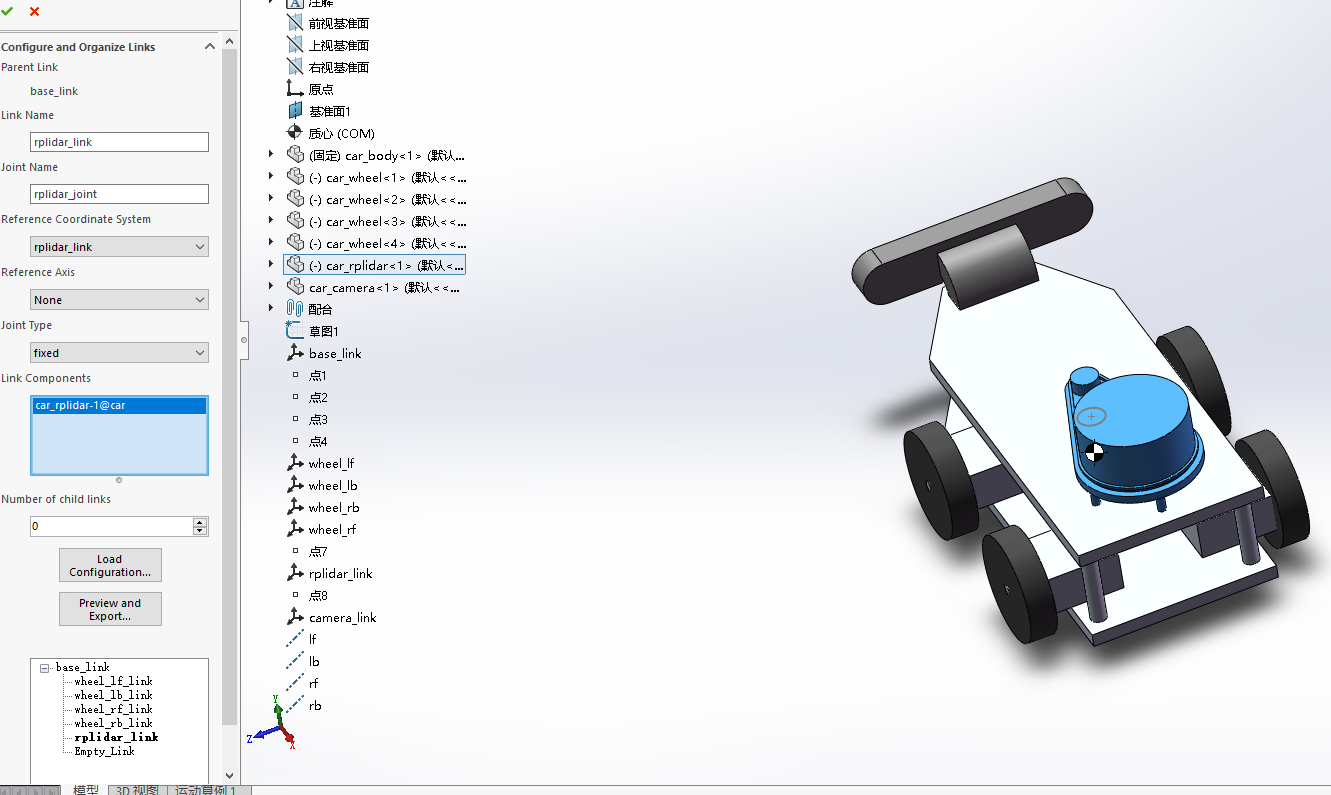
Link Components：选择对应零件

然后设置有6个子link：四个车轮，相机，激光雷达

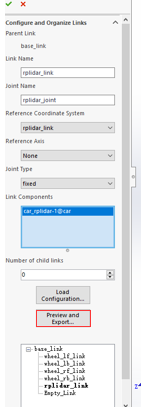


车轮如图设置

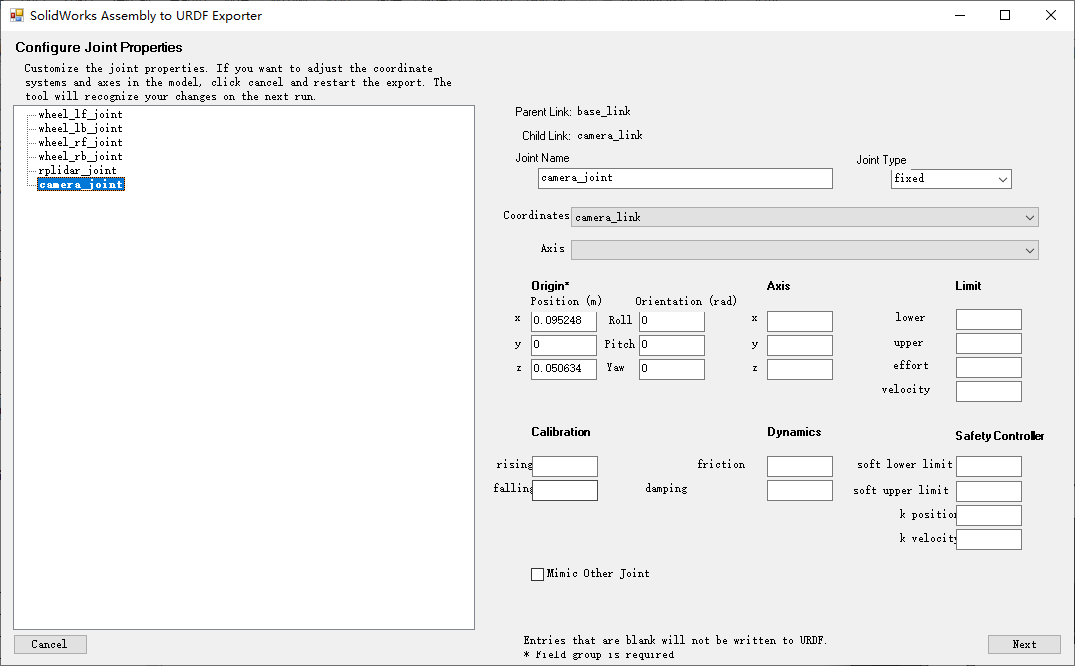
Joint为关节：自己设置名字，然后选择坐标系，旋转轴，关节类型设置为countinuous，无限位旋转，四个轮子依次设置。



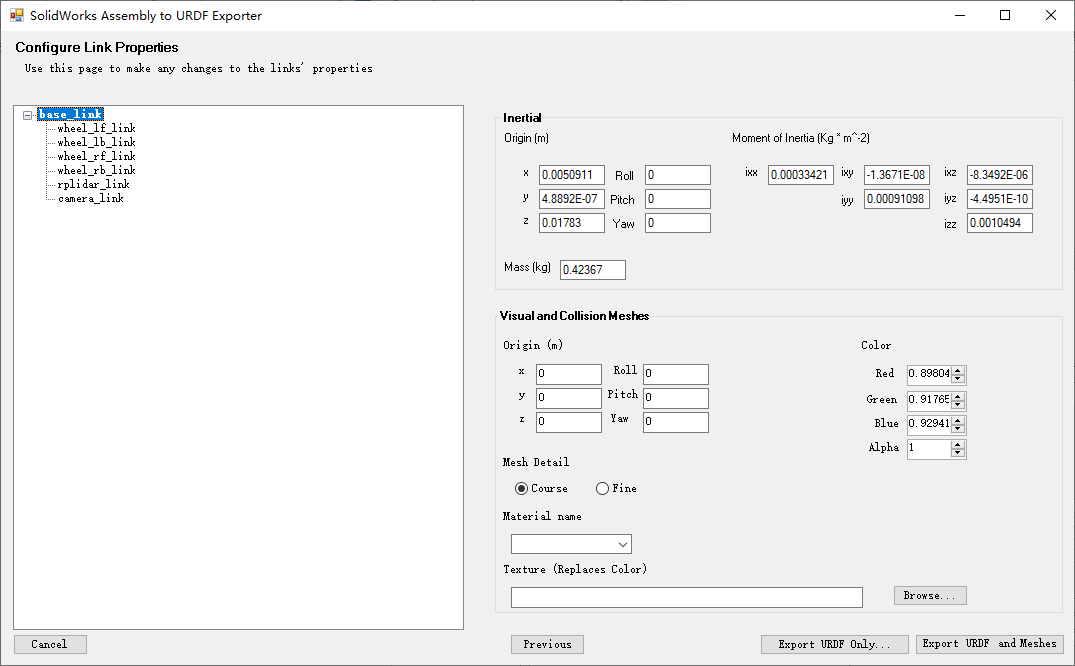
雷达和相机如图设置，选择fixed固定。



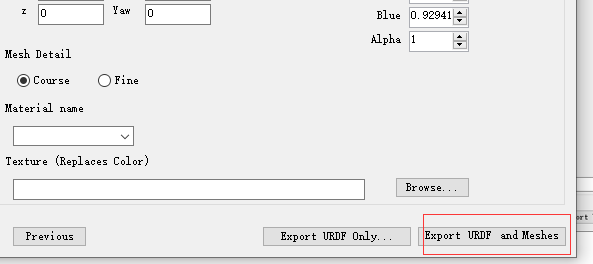
全部设置好后点击Preview and export导出功能包



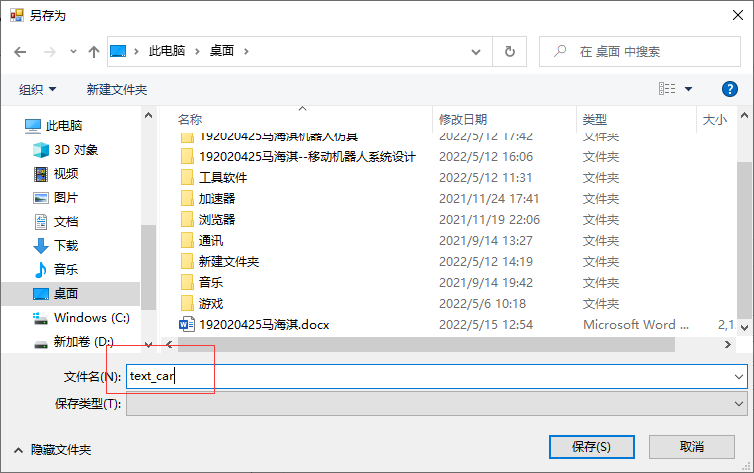
检查关节设置是否正确



检车link设置是否正确



点击导出功能包。



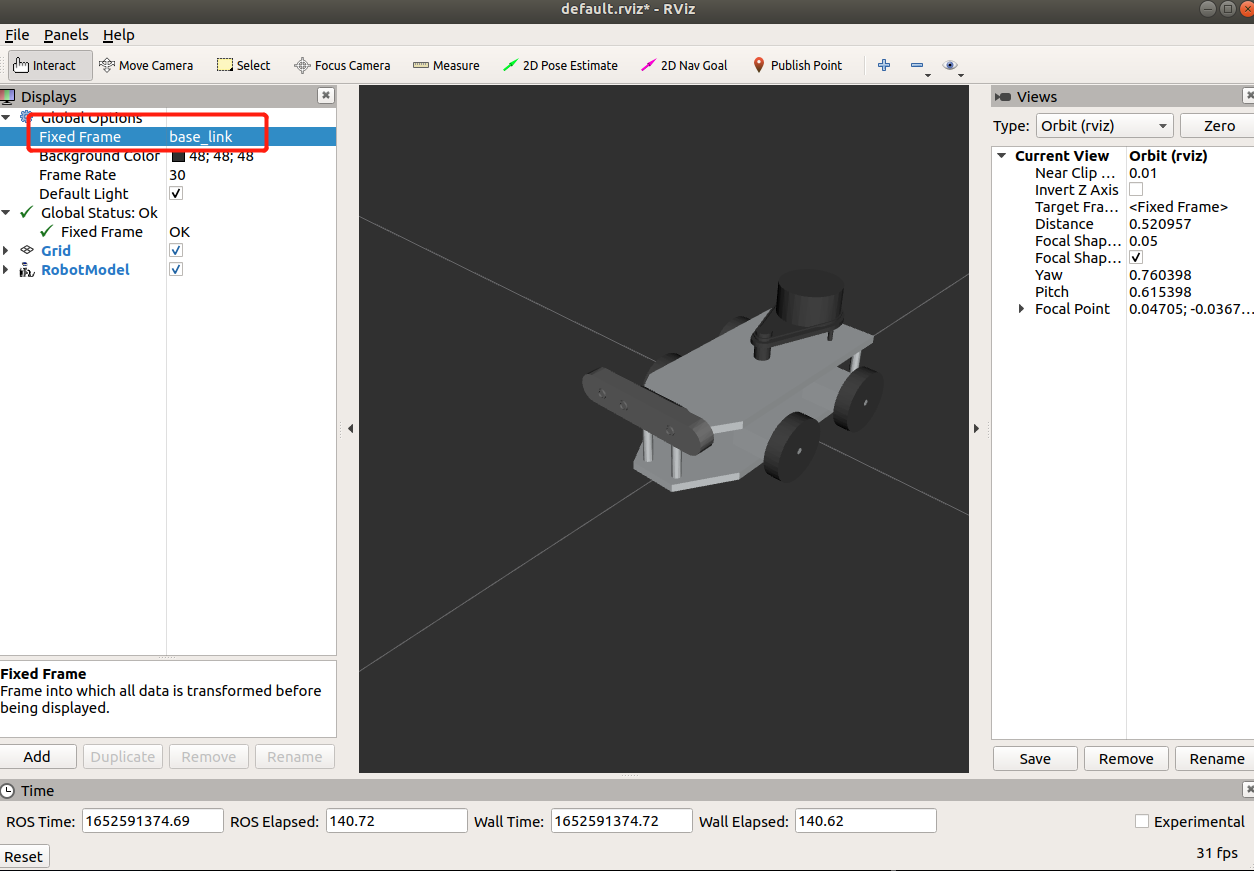
去掉后面的后缀。这样，我们第一步建立机器人基本模型就完成了，但是到这里我们只是建立了一个模型，没有任何系统。

# ROS系统中显示以及添加控制插件

将打出的功能包复制到Ubuntu系统中的工作空间src路径下。（先看ROS21讲）

然后catkin\_make 编译工作空间

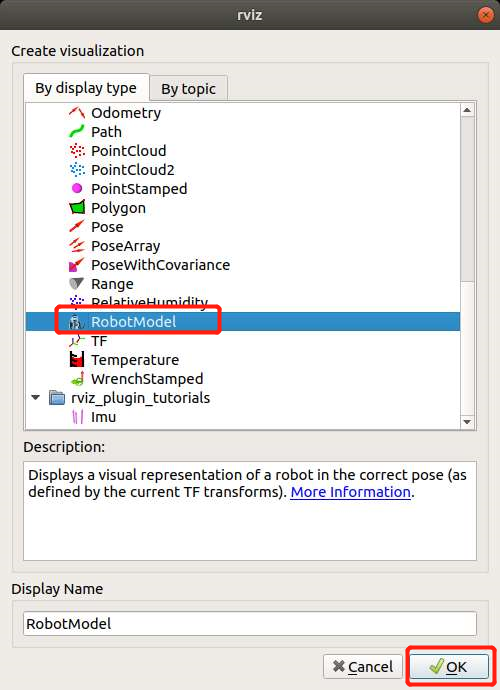
在终端输入 roslaunch text\_car(功能包名) display.launch



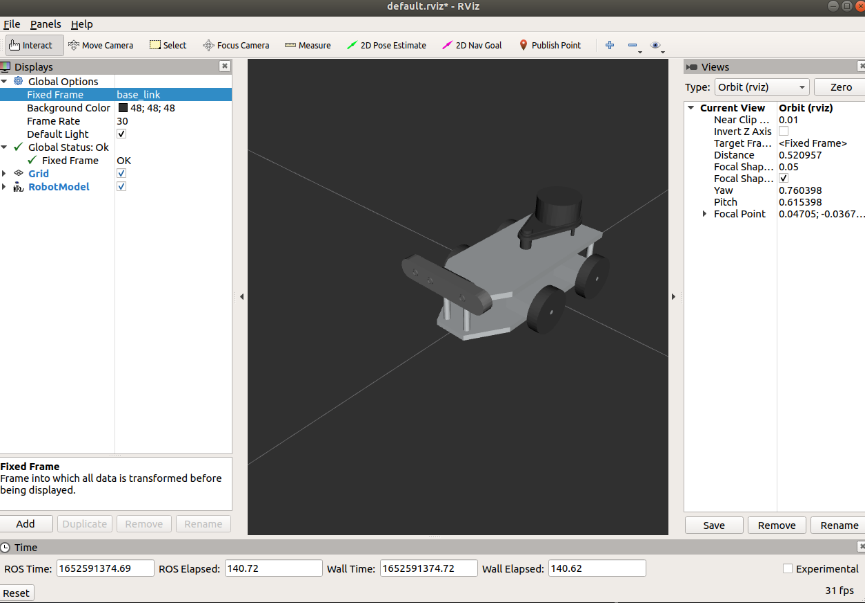
基座标系选择base\_link



选择add



选择加入机器人模型。之后就能看到我们的机器人已经再RVIZ中显示出来了。



接着输入 roslaunch text\_car gazebo.launch就可以看到机器人模型再gaebo仿真环境中显示出来了，但是是纯灰色，接下来我们就来添加gazebo颜色标签。

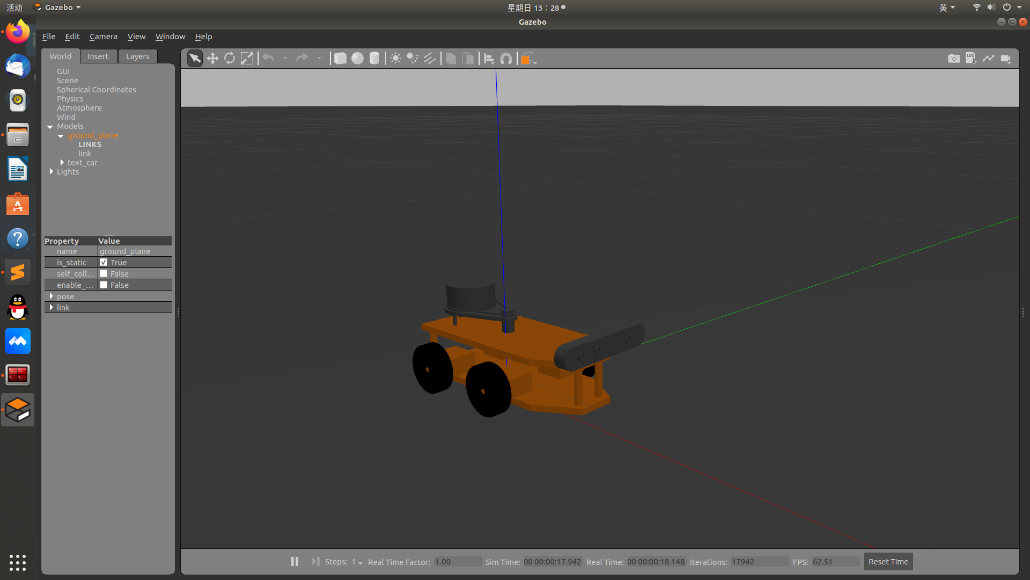
打开功能包urdf文件夹下的urdf文件，在其中插入（注意插入位置，）



要在robot标签以内位置插入如下代码

|  |
| --- |
| 1. **<gazebo** reference="base\_link"**>** 2. **<material>**Gazebo/Orange**</material>** 3. **</gazebo>** 4. **<gazebo** reference="rplidar\_link"**>** 5. **<material>**Gazebo/Grey**</material>** 6. **</gazebo>** 7. **<gazebo** reference="wheel\_lf\_link"**>** 8. **<material>**Gazebo/Black**</material>** 9. **</gazebo>** 10. **<gazebo** reference="wheel\_lb\_link"**>** 11. **<material>**Gazebo/Black**</material>** 12. **</gazebo>** 13. **<gazebo** reference="wheel\_rf\_link"**>** 14. **<material>**Gazebo/Black**</material>** 15. **</gazebo>** 16. **<gazebo** reference="wheel\_rb\_link"**>** 17. **<material>**Gazebo/Black**</material>** 18. **</gazebo>** 19. **<gazebo** reference="camera\_link"**>** 20. **<material>**Gazebo/Grey**</material>** 21. **</gazebo>** |

再次roslaunch text\_car gazebo.launch就可以看待机器人已经显示出我们所设置 的颜色了。



输入rostopic list我们可以看到，其中没有/cmd\_vel运动控制指令，也就是说，小车并不能进行运动控制，接下来我们就为小车添加运动控制插件。

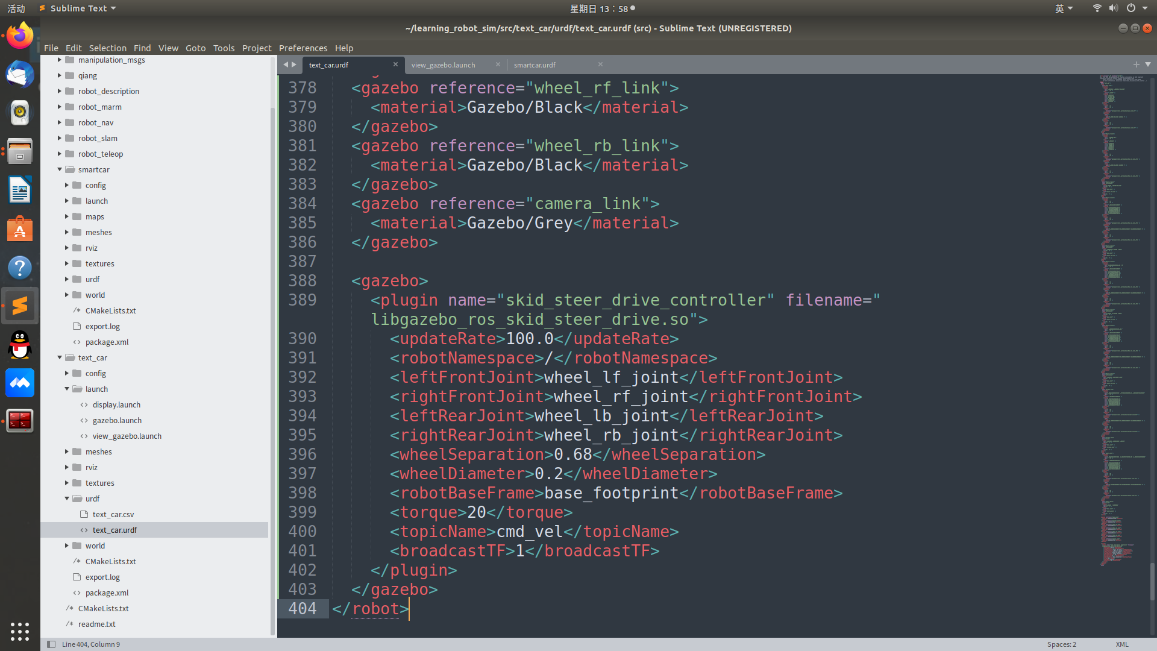
Gazebo插件有很多

differential\_drive\_controller：这是一个差速控制插件，使用与两个驱动轮的机器人（很明显不适用于我们的机器人）

skid\_steer\_drive\_controller：滑动转向插件，适用于四轮驱动，选用这款插件。

还有激光雷达插件，深度相机插件，具体到后面代码实现，我们不在这里赘述。

继续在urdf模型文件中加入skid\_steer\_drive\_controller插件：



|  |
| --- |
| 1. **<gazebo>** 2. **<plugin** name="skid\_steer\_drive\_controller" filename="libgazebo\_ros\_skid\_steer\_drive.so"**>** 3. **<updateRate>**100.0**</updateRate>** 4. **<robotNamespace>**/**</robotNamespace>** 5. **<leftFrontJoint>**wheel\_lf\_joint**</leftFrontJoint>** 6. **<rightFrontJoint>**wheel\_rf\_joint**</rightFrontJoint>** 7. **<leftRearJoint>**wheel\_lb\_joint**</leftRearJoint>** 8. **<rightRearJoint>**wheel\_rb\_joint**</rightRearJoint>** 9. **<wheelSeparation>**0.68**</wheelSeparation>** 10. **<wheelDiameter>**0.2**</wheelDiameter>** 11. **<robotBaseFrame>**base\_footprint**</robotBaseFrame>** 12. **<torque>**20**</torque>** 13. **<topicName>**cmd\_vel**</topicName>** 14. **<broadcastTF>**1**</broadcastTF>** 15. **</plugin>** 16. **</gazebo>** |

将如图所示代码加入urdf文件，此时还不能直接运行gazebo.luanch文件，因为其卵巢文件缺少很多东西，所以我们重写一份，在功能包路径下luanch文件夹内新建view\_gazebo.launch

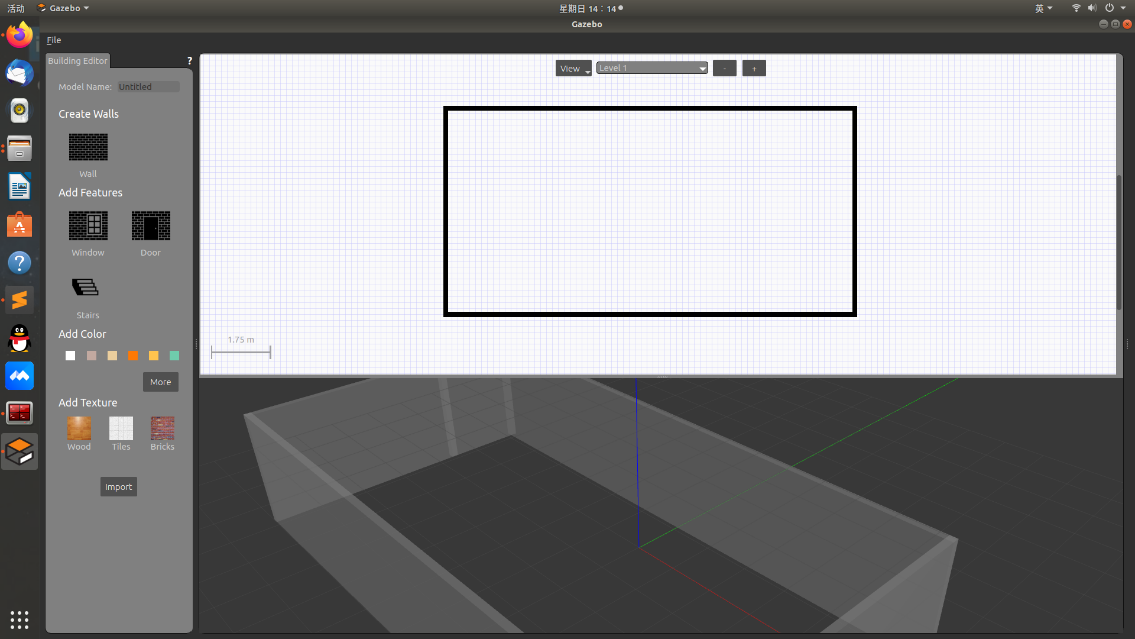
|  |
| --- |
| 1. **<launch>** 3. <!-- 设置launch文件的参数 --> 4. **<arg** name="world\_name" value="$(find text\_car)/world/smartcar\_text.world"**/>** 5. **<arg** name="paused" default="false"**/>** 6. **<arg** name="use\_sim\_time" default="true"**/>** 7. **<arg** name="gui" default="true"**/>** 8. **<arg** name="headless" default="false"**/>** 9. **<arg** name="debug" default="false"**/>** 11. <!-- 运行gazebo仿真环境 --> 12. **<include** file="$(find gazebo\_ros)/launch/empty\_world.launch"**>** 13. **<arg** name="world\_name" value="$(arg world\_name)" **/>** 14. **<arg** name="debug" value="$(arg debug)" **/>** 15. **<arg** name="gui" value="$(arg gui)" **/>** 16. **<arg** name="paused" value="$(arg paused)"**/>** 17. **<arg** name="use\_sim\_time" value="$(arg use\_sim\_time)"**/>** 18. **<arg** name="headless" value="$(arg headless)"**/>** 19. **</include>** 21. <!-- 加载机器人模型描述参数 --> 22. **<param** 23. name="robot\_description" 24. textfile="$(find text\_car)/urdf/text\_car.urdf" **/>** 26. <!-- 运行joint\_state\_publisher节点，发布机器人的关节状态  --> 27. **<node** name="joint\_state\_publisher" pkg="joint\_state\_publisher" type="joint\_state\_publisher" **></node>** 29. <!-- 运行robot\_state\_publisher节点，发布tf  --> 30. **<node** name="robot\_state\_publisher" pkg="robot\_state\_publisher" type="robot\_state\_publisher"  output="screen" **>** 31. **<param** name="publish\_frequency" type="double" value="50.0" **/>** 32. **</node>** 34. <!-- 在gazebo中加载机器人模型--> 35. **<node** name="urdf\_spawner" pkg="gazebo\_ros" type="spawn\_model" respawn="false" output="screen" 36. args="-file $(find text\_car)/urdf/text\_car.urdf -urdf -model text\_car"**/>** 37. **<node** name="rviz" pkg="rviz" type="rviz" args="-d $(find text\_car)/rviz/smart.rviz" required="true" **/>** 39. **</launch>** |

其中代码主要修改位置为：

第4行：所创建世界路径为本功能包下world文件夹，没有就创建一个，然后将世界文件放入其中。终端上输入 gazebo ，

点击

在点击building editor，点击wall绘制一个地图（切记把原点包含进去）

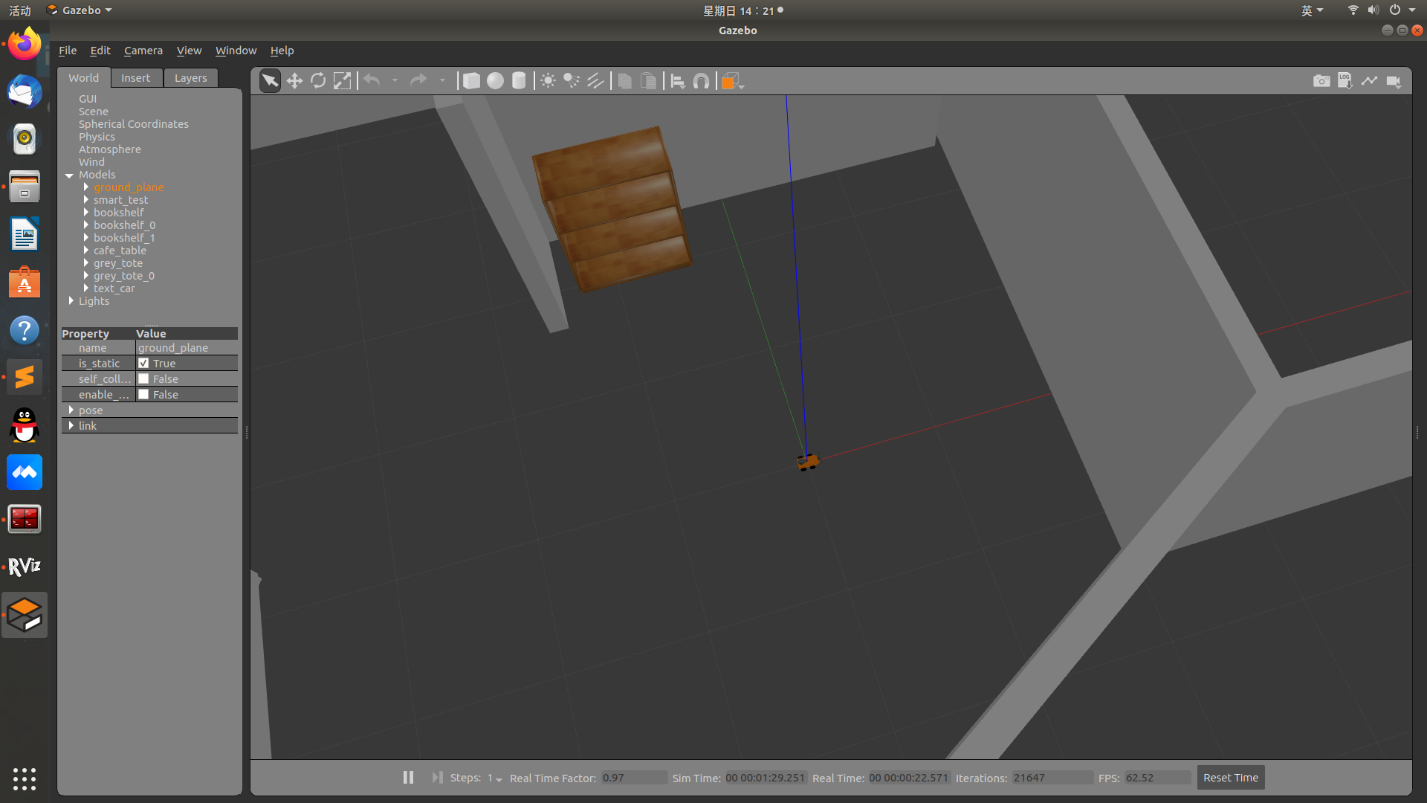


然后Ctrl + S保存一下，点击左上角file，exit building editor ，就可以看到地图已经建立成功，保存地图为launch文件中smartcar\_text.world，也可自定义，但记得修改launch文件，最后将该地图文件复制到功能包world文件夹下。

24 36行：记得修改路径。

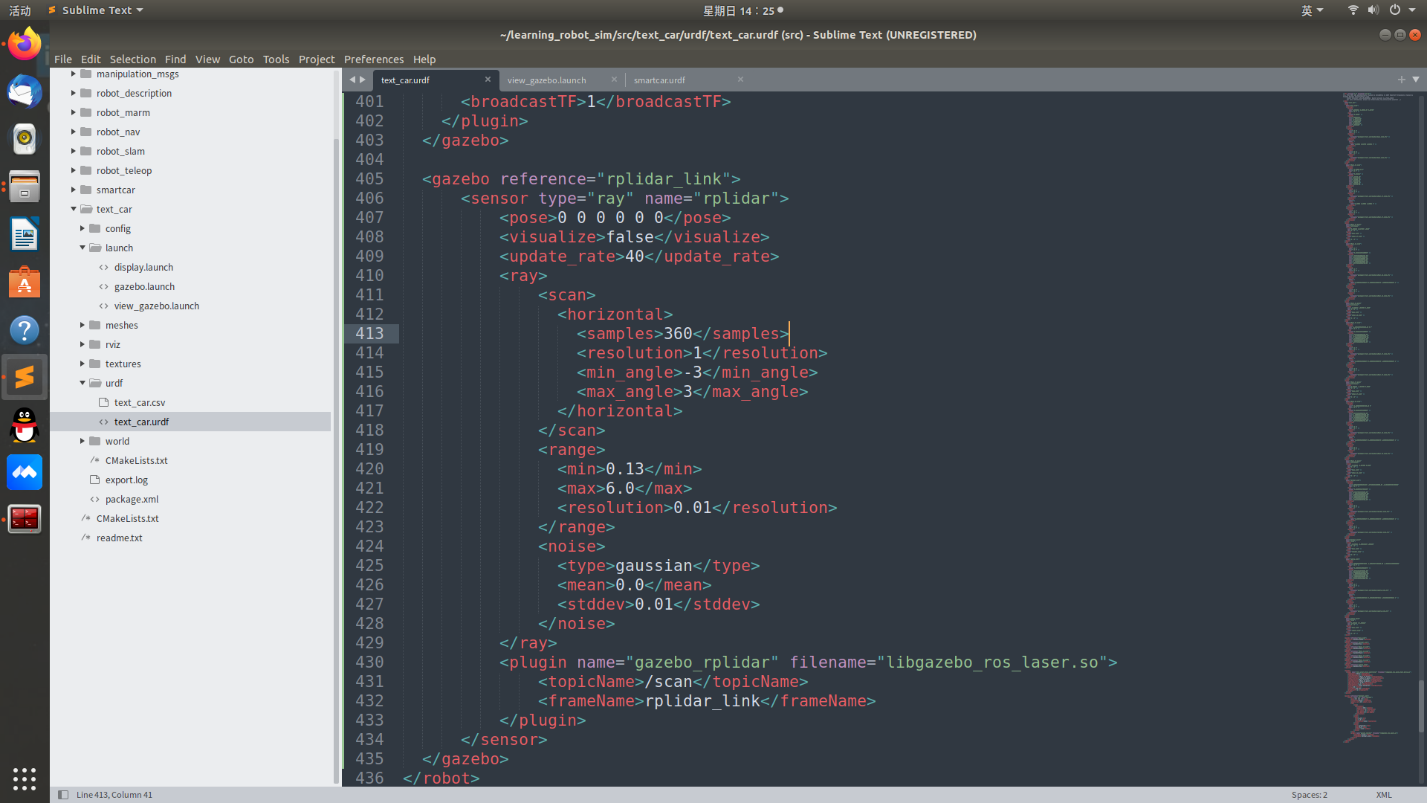
接下来 roslaunch text\_car view\_gazebo.launch

可以看到小车加载到了我们的世界中，



输入rostopic list可以看到已经有了/cmd\_vel这个topic，说明小车已经可以运动了，通过给小车发送运动控制指令就可以控制小车运动了，这些东西自行学习我们先不在这里赘述，直接进入下一部分内容。

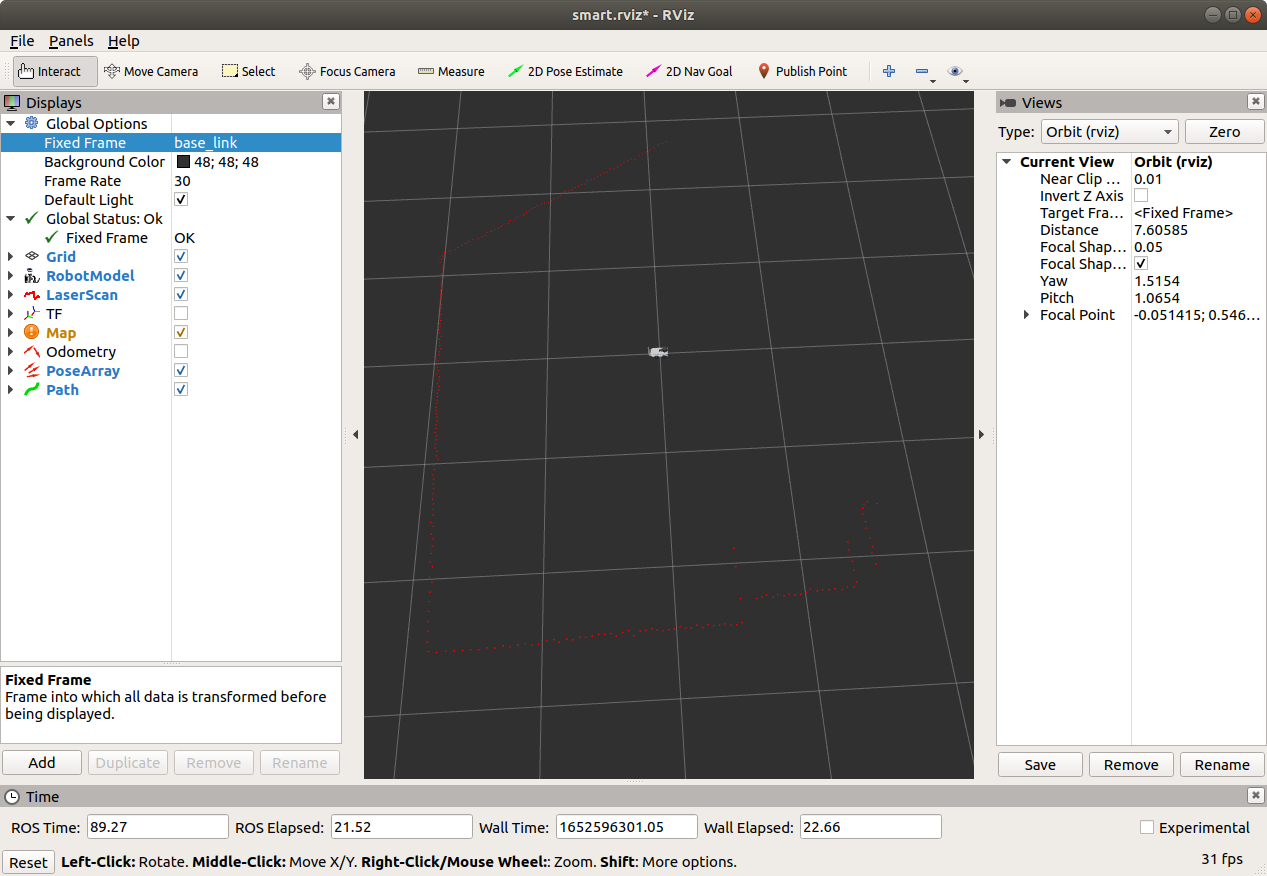
# 添加激光雷达插件



如图添加如下代码

|  |
| --- |
| 1. **<gazebo** reference="rplidar\_link"**>** 2. **<sensor** type="ray" name="rplidar"**>** 3. **<pose>**0 0 0 0 0 0**</pose>** 4. **<visualize>**false**</visualize>** 5. **<update\_rate>**40**</update\_rate>** 6. **<ray>** 7. **<scan>** 8. **<horizontal>** 9. **<samples>**360**</samples>** 10. **<resolution>**1**</resolution>** 11. **<min\_angle>**-3**</min\_angle>** 12. **<max\_angle>**3**</max\_angle>** 13. **</horizontal>** 14. **</scan>** 15. **<range>** 16. **<min>**0.13**</min>** 17. **<max>**6.0**</max>** 18. **<resolution>**0.01**</resolution>** 19. **</range>** 20. **<noise>** 21. **<type>**gaussian**</type>** 22. **<mean>**0.0**</mean>** 23. **<stddev>**0.01**</stddev>** 24. **</noise>** 25. **</ray>** 26. **<plugin** name="gazebo\_rplidar" filename="libgazebo\_ros\_laser.so"**>** 27. **<topicName>**/scan**</topicName>** 28. **<frameName>**rplidar\_link**</frameName>** 29. **</plugin>** 30. **</sensor>** 31. **</gazebo>** |

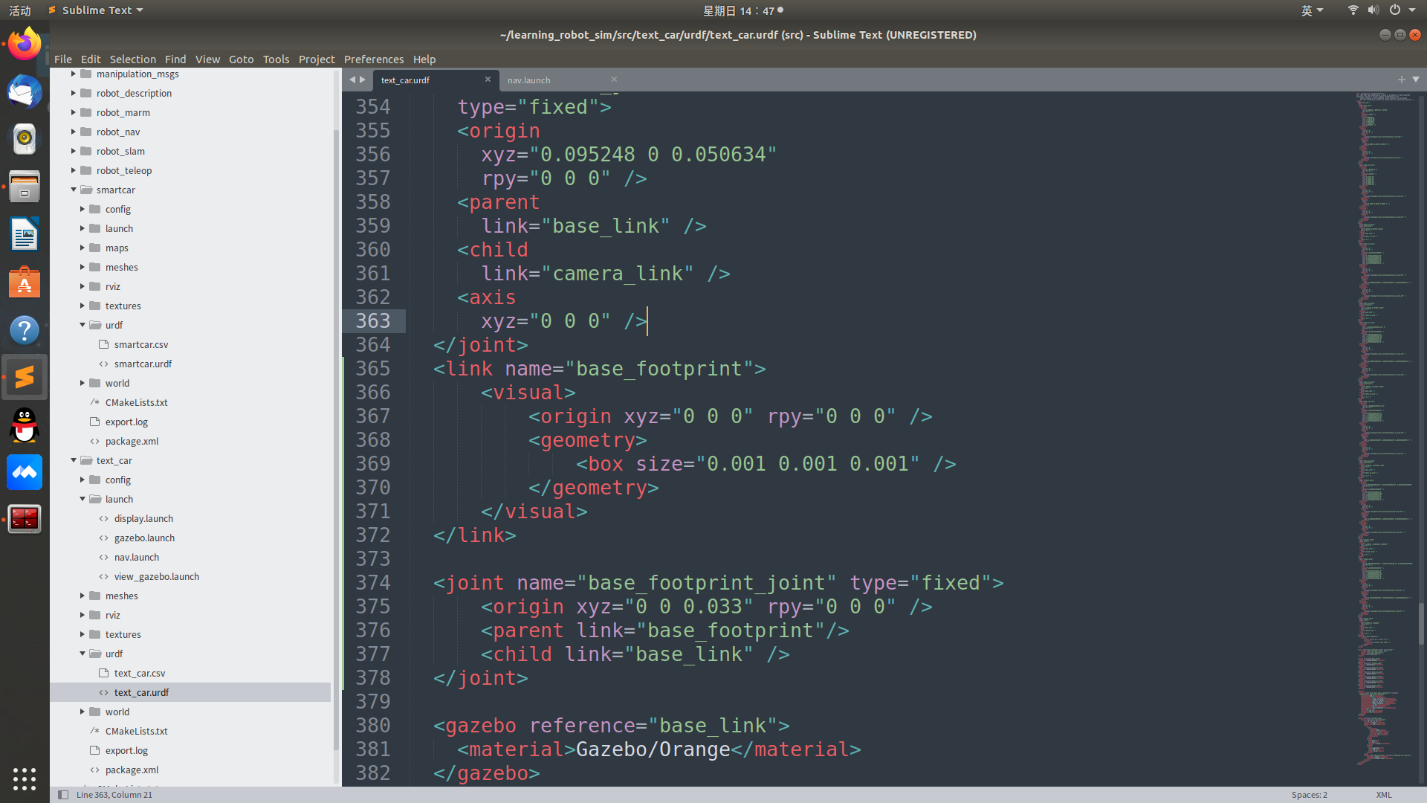
再次运动view文件就能在rviz看到激光雷达数据



# SLAM建图以及自主导航

SLAM建图以及自主导航，这两者可以分开运用，但为了节约时间，也为了让我早点写完，这里我们一步到位，然后机器人边导航边建图。

第一步：现在urdf文件中加入一个link以及其joint叫做base\_footprint(具体作用自行csdn学习，我不在赘述)其代码如下图：

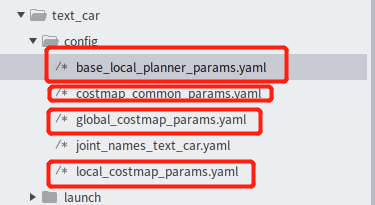


|  |
| --- |
| 1. **<link** name="base\_footprint"**>** 2. **<visual>** 3. **<origin** xyz="0 0 0" rpy="0 0 0" **/>** 4. **<geometry>** 5. **<box** size="0.001 0.001 0.001" **/>** 6. **</geometry>** 7. **</visual>** 8. **</link>** 10. **<joint** name="base\_footprint\_joint" type="fixed"**>** 11. **<origin** xyz="0 0 0.033" rpy="0 0 0" **/>** 12. **<parent** link="base\_footprint"**/>** 13. **<child** link="base\_link" **/>** 14. **</joint>** |

第二步就要来写导航方面的配置文件了，这方面的知识不是我们这个教程能讲完的量。

[(1条消息) 机器人基于ROS的move\_base导航功能的部署（代价地图）\_wallEVA96的博客-CSDN博客\_movebase不能实时避障](https://blog.csdn.net/walleva96/article/details/118027629?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522165259745916782388068572%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=165259745916782388068572&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-3-118027629-null-null.142%5ev9%5econtrol,157%5ev4%5econtrol&utm_term=ROS%E4%BB%A3%E4%BB%B7%E5%9C%B0%E5%9B%BE%E9%85%8D%E7%BD%AE&spm=1018.2226.3001.4187)

给大家推荐一篇博客，但我还是推荐看书，由于现在快递不通，我这里有书，有兴趣同学来找我拿。



本地路径规划器

|  |
| --- |
| 1. local\_costmap: 2. global\_frame: map 3. robot\_base\_frame: base\_footprint 4. update\_frequency: 5.0 5. publish\_frequency: 2.0 6. static\_map: false 7. rolling\_window: true 8. width: 4.0 9. height: 4.0 10. resolution: 0.05 11. origin\_x: 0 12. origin\_y: 0 13. transform\_tolerance: 0.5 14. plugins: 15. - {name: voxel\_layer,      type: "costmap\_2d::VoxelLayer"} 16. - {name: inflation\_layer,     type: "costmap\_2d::InflationLayer"} |

通用代价地图配置文件

|  |
| --- |
| 1. obstacle\_range: 2.5 2. raytrace\_range: 3.0 3. footprint: [[0.06,0.04], [-0.06,0.04], [-0.06,-0.04], [0.06,-0.04]] 4. inflation\_radius: 0.15 5. max\_obstacle\_height: 0.6 6. min\_obstacle\_height: 0.0 7. observation\_sources: scan 8. scan: {data\_type: LaserScan, topic: /scan, marking: true, clearing: true, expected\_update\_rate: 0} |

全局代价地图配置文件

|  |
| --- |
| 1. global\_costmap: 2. global\_frame: map 3. robot\_base\_frame: base\_footprint 4. update\_frequency: 5.0 5. publish\_frequency: 0.5 6. static\_map: true 7. rolling\_window: false 8. transform\_tolerance: 0.5 9. plugins: 10. - {name: static\_layer,            type: "costmap\_2d::StaticLayer"} 11. - {name: voxel\_layer,             type: "costmap\_2d::VoxelLayer"} 12. - {name: inflation\_layer,         type: "costmap\_2d::InflationLayer"} |

局部代价地图配置文件

|  |
| --- |
| 1. local\_costmap: 2. global\_frame: map 3. robot\_base\_frame: base\_footprint 4. update\_frequency: 5.0 5. publish\_frequency: 2.0 6. static\_map: false 7. rolling\_window: true 8. width: 4.0 9. height: 4.0 10. resolution: 0.05 11. origin\_x: 0 12. origin\_y: 0 13. transform\_tolerance: 0.5 14. plugins: 15. - {name: voxel\_layer,      type: "costmap\_2d::VoxelLayer"} 16. - {name: inflation\_layer,     type: "costmap\_2d::InflationLayer"}} |

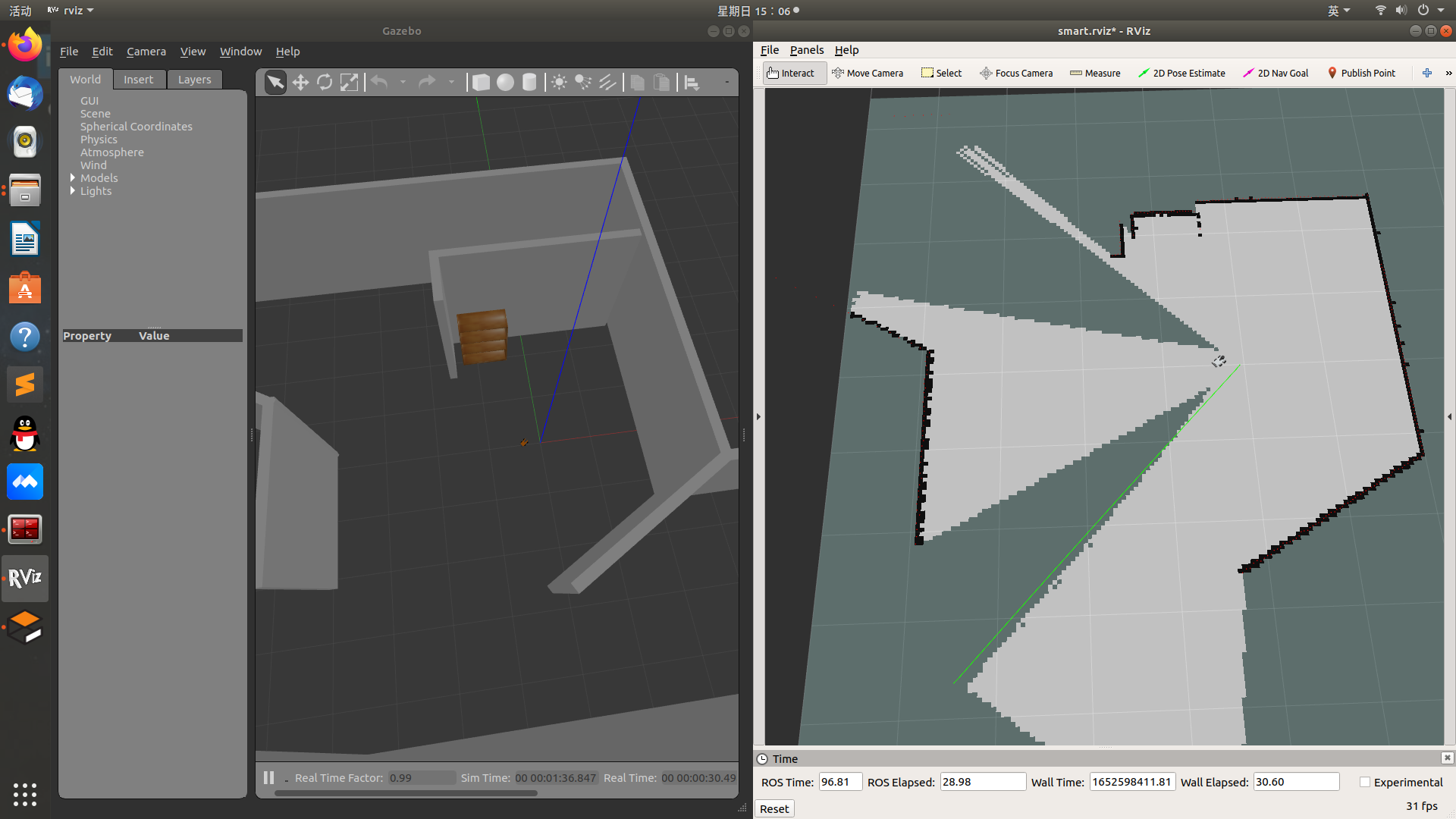
第三步编写建图导航launch文件

|  |
| --- |
| 1. **<launch>** 2. **<include** file="$(find text\_car)/launch/view\_gazebo.launch"**>** 3. **</include>** 5. **<node** pkg="gmapping" type="slam\_gmapping" name="simple\_gmapping" output="screen"**>** 6. **<param** name="map\_update\_interval" value="1.0"**/>** 7. **<param** name="maxUrange" value="5.0"**/>** 8. **<param** name="maxRange" value="6.0"**/>** 9. **<param** name="sigma" value="0.05"**/>** 10. **<param** name="kernelSize" value="1"**/>** 11. **<param** name="lstep" value="0.05"**/>** 12. **<param** name="astep" value="0.05"**/>** 13. **<param** name="iterations" value="5"**/>** 14. **<param** name="lsigma" value="0.075"**/>** 15. **<param** name="ogain" value="3.0"**/>** 16. **<param** name="lskip" value="0"**/>** 17. **<param** name="minimumScore" value="50"**/>** 18. **<param** name="srr" value="0.1"**/>** 19. **<param** name="srt" value="0.2"**/>** 20. **<param** name="str" value="0.1"**/>** 21. **<param** name="stt" value="0.2"**/>** 22. **<param** name="linearUpdate" value="1.0"**/>** 23. **<param** name="angularUpdate" value="0.5"**/>** 24. **<param** name="temporalUpdate" value="3.0"**/>** 25. **<param** name="resampleThreshold" value="0.5"**/>** 26. **<param** name="particles" value="50"**/>** 27. **<param** name="xmin" value="-5.0"**/>** 28. **<param** name="ymin" value="-5.0"**/>** 29. **<param** name="xmax" value="5.0"**/>** 30. **<param** name="ymax" value="5.0"**/>** 31. **<param** name="delta" value="0.05"**/>** 32. **<param** name="llsamplerange" value="0.01"**/>** 33. **<param** name="llsamplestep" value="0.01"**/>** 34. **<param** name="lasamplerange" value="0.005"**/>** 35. **<param** name="lasamplestep" value="0.005"**/>** 36. **<param** name="base\_frame" value="/base\_link"**/>** 37. **</node>** 39. <!-- 启动 move\_base 节点 --> 40. **<node** pkg="move\_base" type="move\_base" respawn="false" name="move\_base" output="screen" clear\_params="true"**>** 41. **<rosparam** file="$(find text\_car)/config/costmap\_common\_params.yaml" command="load" ns="global\_costmap" **/>** 42. **<rosparam** file="$(find text\_car)/config/costmap\_common\_params.yaml" command="load" ns="local\_costmap" **/>** 43. **<rosparam** file="$(find text\_car)/config/local\_costmap\_params.yaml" command="load" **/>** 44. **<rosparam** file="$(find text\_car)/config/global\_costmap\_params.yaml" command="load" **/>** 45. **<rosparam** file="$(find text\_car)/config/base\_local\_planner\_params.yaml" command="load" **/>** 46. **</node>** 48. <!-- 启动 AMCL 节点 --> 49. **<node** pkg="amcl" type="amcl" name="amcl" clear\_params="true"**>** 50. **<param** name="use\_map\_topic" value="false"**/>** 51. <!-- Publish scans from best pose at a max of 10 Hz --> 52. **<param** name="odom\_model\_type" value="diff"**/>** 53. **<param** name="odom\_alpha5" value="0.1"**/>** 54. **<param** name="gui\_publish\_rate" value="10.0"**/>** 55. **<param** name="laser\_max\_beams" value="60"**/>** 56. **<param** name="laser\_max\_range" value="12.0"**/>** 57. **<param** name="min\_particles" value="500"**/>** 58. **<param** name="max\_particles" value="2000"**/>** 59. **<param** name="kld\_err" value="0.05"**/>** 60. **<param** name="kld\_z" value="0.99"**/>** 61. **<param** name="odom\_alpha1" value="0.2"**/>** 62. **<param** name="odom\_alpha2" value="0.2"**/>** 63. **<param** name="odom\_alpha3" value="0.2"**/>** 64. **<param** name="odom\_alpha4" value="0.2"**/>** 65. **<param** name="laser\_z\_hit" value="0.5"**/>** 66. **<param** name="laser\_z\_short" value="0.05"**/>** 67. **<param** name="laser\_z\_max" value="0.05"**/>** 68. **<param** name="laser\_z\_rand" value="0.5"**/>** 69. **<param** name="laser\_sigma\_hit" value="0.2"**/>** 70. **<param** name="laser\_lambda\_short" value="0.1"**/>** 71. **<param** name="laser\_model\_type" value="likelihood\_field"**/>** 72. **<param** name="laser\_likelihood\_max\_dist" value="2.0"**/>** 73. **<param** name="update\_min\_d" value="0.25"**/>** 74. **<param** name="update\_min\_a" value="0.2"**/>** 76. **<param** name="odom\_frame\_id" value="odom"**/>** 78. **<param** name="resample\_interval" value="1"**/>** 79. <!-- Increase tolerance because the computer can get quite busy --> 80. **<param** name="transform\_tolerance" value="1.0"**/>** 81. **<param** name="recovery\_alpha\_slow" value="0.0"**/>** 82. **<param** name="recovery\_alpha\_fast" value="0.0"**/>** 83. <!-- scan topic --> 84. **<remap** from="scan" to="scan"**/>** 85. **</node>** 87. **</launch>** |

这一章节涉及到的算法非常多，例如我们运用到的gmapping建图算法，Dijkstra全局路径规划算法，DWA局部路径规划算法，以及AMCL定位算法。我不可能跟大家一一讲解，还是需要大家自己去理解，这些算法作用，如何运用。

到此为止，我们的机器人就可以进行建图导航了，编译功能包，然后

roslaunch text\_car nav.launch



如图，我们的小车已经开始按照我们用2D Nav Goal设定的目标进行运动了，并且运动过程成还实时建立地图。如此，我们就完成了这次教程的全部内容，希望大家好好学习。

还有什么问题直接在群里提问，就不要私聊我了，直接在群里问，这样大家都能有个参考。