

# ROS理论与实践

## —— 第6讲：构建机器人仿真平台



主讲人 胡春旭



机器人博客“古月居”博主  
《ROS机器人开发实践》作者  
武汉精锋微控科技有限公司 联合创始人  
华中科技大学 人工智能与自动化学院 硕士



 1. 优化物理仿真模型

 2. 创建物理仿真环境

 3. 传感器仿真及应用



# 1. 优化物理仿真模型

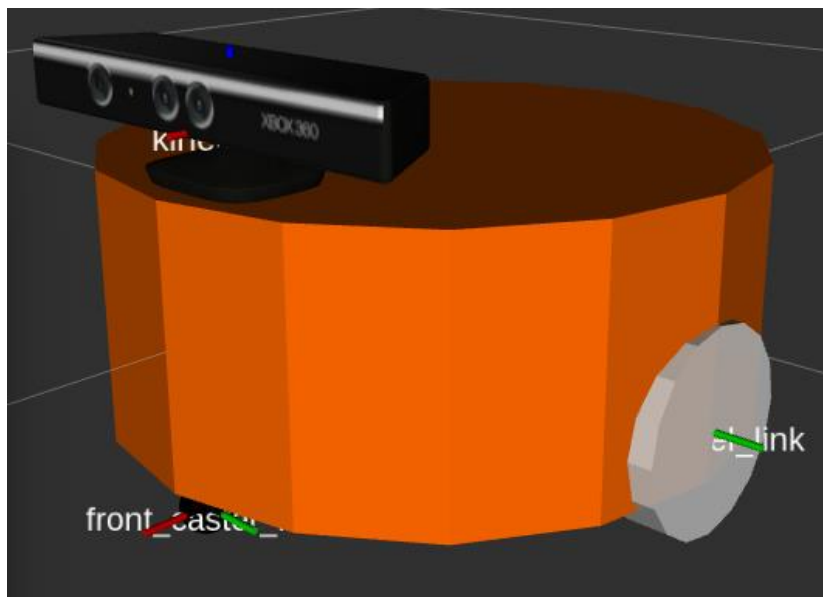


# 1. 优化物理仿真模型

- 使用xacro文件优化URDF模型
- 完善机器人模型的物理仿真属性
- 在机器人模型中添加控制器插件



# 1. 优化物理仿真模型



## URDF建模存在哪些问题？

- 模型冗长，重复内容过多；
- 参数修改麻烦，不便于二次开发；
- 没有参数计算的功能；
- ...

```
<?xml version="1.0" ?>
<robot name="mbot">

  <link name="base_link">
    <visual>
      <origin xyz=" 0 0 0" rpy="0 0 0" />
      <geometry>
        <cylinder length="0.16" radius="0.20"/>
      </geometry>
      <material name="yellow">
        <color rgba="1 0.4 0 1"/>
      </material>
    </visual>
  </link>

  <joint name="left_wheel_joint" type="continuous">
    <origin xyz="0 0.19 -0.05" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="left_wheel_link"/>
    <axis xyz="0 1 0"/>
  </joint>

  <link name="left_wheel_link">
    <visual>
      <origin xyz="0 0 0" rpy="1.5707 0 0" />
      <geometry>
        <cylinder radius="0.06" length = "0.025"/>
      </geometry>
      <material name="white">
        <color rgba="1 1 1 0.9"/>
      </material>
    </visual>
  </link>

</robot>
```

URDF模型



# 1. 优化物理仿真模型

## URDF模型的进化版本——xacro模型文件

### ➤ 精简模型代码

- 创建宏定义
- 文件包含

### ➤ 提供可编程接口

- 常量
- 变量
- 数学计算
- 条件语句

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

  <xacro:property name="M_PI" value="3.14159"/>
  <xacro:property name="wheel_radius" value="0.033"/>
  <xacro:property name="wheel_length" value="0.017"/>
  <xacro:property name="base_link_radius" value="0.13"/>
  <xacro:property name="base_link_length" value="0.005"/>
  <xacro:property name="motor_radius" value="0.02"/>
  <xacro:property name="motor_length" value="0.08"/>
  <xacro:property name="motor_x" value="-0.055"/>
  <xacro:property name="motor_y" value="0.075"/>
  <xacro:property name="plate_height" value="0.07"/>
  <xacro:property name="standoff_x" value="0.12"/>
  <xacro:property name="standoff_y" value="0.10"/>

  <!-- 定义MRobot本体的宏 -->
  <xacro:macro name="mrobot_standoff_2in" params="parent number x_loc y_loc z_loc">
    <joint name="standoff_2in_${number}_joint" type="fixed">
      <origin xyz="${x_loc} ${y_loc} ${z_loc}" rpy="0 0 0" />
      <parent link="${parent}"/>
      <child link="standoff_2in_${number}_link" />
    </joint>

    <link name="standoff_2in_${number}_link">
      <inertial>
        <mass value="0.001" />
        <origin xyz="0 0 0" />
        <inertia ixx="0.0001" ixy="0.0" ixz="0.0"
                iyy="0.0001" iyz="0.0"
                izz="0.0001" />
      </inertial>

      <visual>
        <origin xyz="0 0 0" rpy="0 0 0" />
        <geometry>
          <box size="0.01 0.01 0.07" />
        </geometry>
        <material name="black">
          <color rgba="0.16 0.17 0.15 0.9"/>
        </material>
      </visual>

      <collision>
        <origin xyz="0.0 0.0 0.0" rpy="0 0 0" />
        <geometry>
          <box size="0.01 0.01 0.07" />
        </geometry>
      </collision>
    </link>
  </xacro:macro>
```

xacro优化后的URDF模型



# 1. 优化物理仿真模型

## 常量定义

```
<xacro:property name="M_PI" value="3.14159"/>
```

## 常量使用

```
<origin xyz="0 0 0" rpy="{M_PI/2} 0 0" />
```

```
<!-- PROPERTY LIST -->
<xacro:property name="M_PI" value="3.1415926"/>
<xacro:property name="base_radius" value="0.20"/>
<xacro:property name="base_length" value="0.16"/>

<xacro:property name="wheel_radius" value="0.06"/>
<xacro:property name="wheel_length" value="0.025"/>
<xacro:property name="wheel_joint_y" value="0.19"/>
<xacro:property name="wheel_joint_z" value="0.05"/>

<xacro:property name="caster_radius" value="0.015"/>
<xacro:property name="caster_joint_x" value="0.18"/>
```

常量定义

```
<joint name="base_footprint_joint" type="fixed">
  <origin xyz="0 0 ${base_length/2 + caster_radius*2}" rpy="0 0 0" />
  <parent link="base_footprint"/>
  <child link="base_link" />
</joint>

<link name="base_link">
  <visual>
    <origin xyz=" 0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder length="{base_length}" radius="{base_radius}" />
    </geometry>
    <material name="yellow" />
  </visual>
</link>
```

常量使用



# 1. 优化物理仿真模型

## 数学计算

```
<origin xyz="0 ${(motor_length+wheel_length)/2} 0" rpy="0 0 0"/>
```

```
<joint name="base_footprint_joint" type="fixed">  
  <origin xyz="0 0 ${base_length/2 + caster_radius*2}" rpy="0 0 0" />  
  <parent link="base_footprint"/>  
  <child link="base_link" />  
</joint>
```

## 数学计算

注意：所有数学运算都会转换成浮点数进行，以保证运算精度





# 1. 优化物理仿真模型

## 宏定义

```
<xacro:macro name="name" params="A B C">  
.....  
</xacro:macro>
```

## 宏调用

```
<name A= "A_value" B= "B_value" C= "C_value" />
```

```
<!-- Macro for robot wheel -->  
<xacro:macro name="wheel" params="prefix reflect">  
  <joint name="${prefix}_wheel_joint" type="continuous">  
    <origin xyz="0 ${reflect*wheel_joint_y} ${-wheel_joint_z}" rpy="0 0 0"/>  
    <parent link="base_link"/>  
    <child link="${prefix}_wheel_link"/>  
    <axis xyz="0 1 0"/>  
  </joint>  
  
  <link name="${prefix}_wheel_link">  
    <visual>  
      <origin xyz="0 0 0" rpy="${M_PI/2} 0 0" />  
      <geometry>  
        <cylinder radius="${wheel_radius}" length = "${wheel_length}"/>  
      </geometry>  
      <material name="gray" />  
    </visual>  
  </link>  
</xacro:macro>
```

## 宏定义

```
<wheel prefix="left" reflect="-1"/>  
<wheel prefix="right" reflect="1"/>
```

## 宏调用



# 1. 优化物理仿真模型

## 文件包含

```
<xacro:include filename="$(find mbot_description)/urdf/mbot_base_gazebo.xacro" />
```

```
<xacro:include filename="$(find mbot_description)/urdf/mbot_base_gazebo.xacro" />  
<xacro:include filename="$(find mbot_description)/urdf/sensors/camera_gazebo.xacro" />
```

文件包含



# 1. 优化物理仿真模型

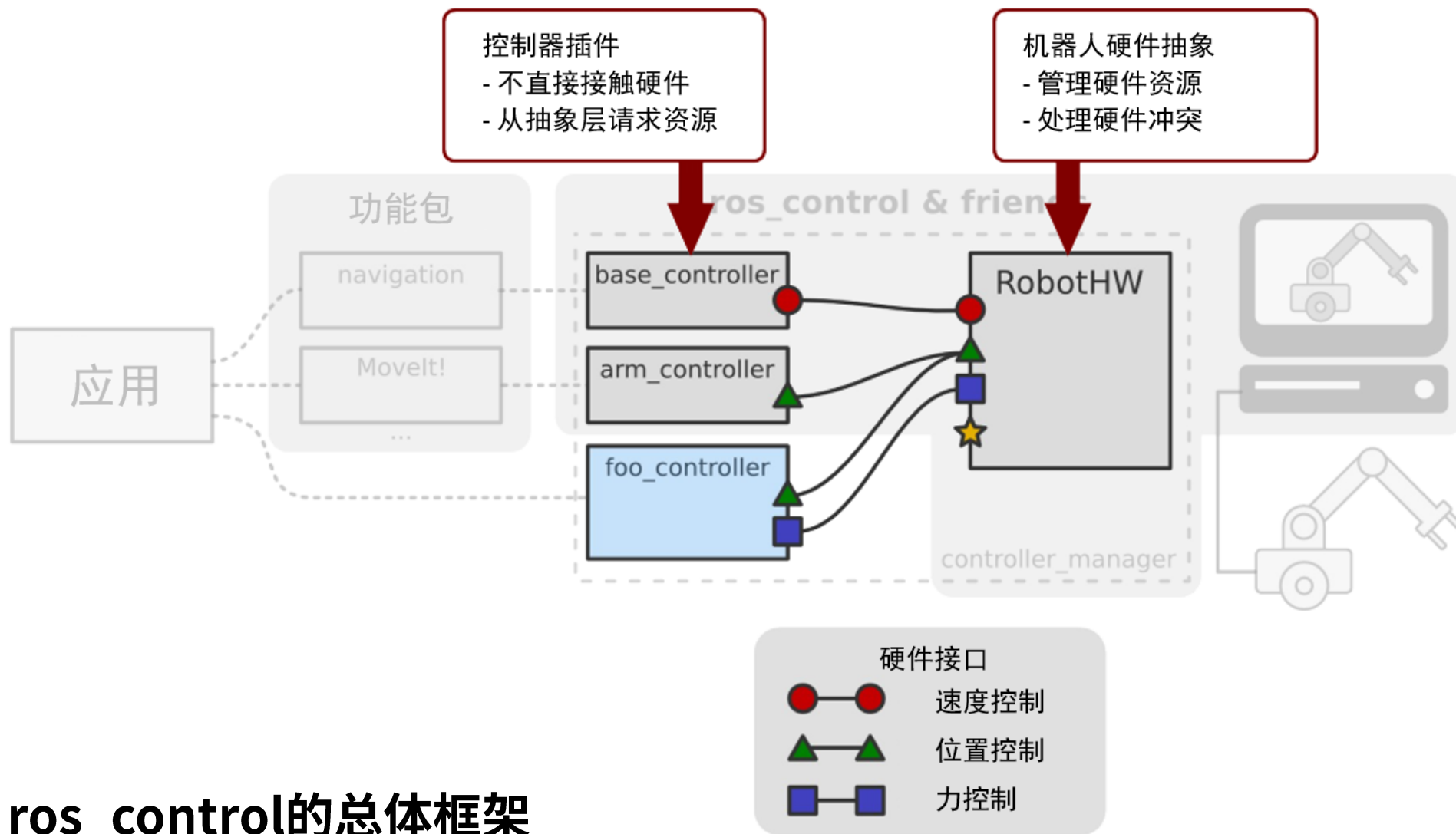


## ros\_control是什么?

- ROS为开发者提供的机器人控制中间件
- 包含一系列控制器接口、传动装置接口、硬件接口、控制器工具箱等等
- 可以帮助机器人应用功能包快速落地，提高开发效率



# 1. 优化物理仿真模型



ros\_control的总体框架



# 1. 优化物理仿真模型

## ➤ 控制器管理器

提供一种通用的接口来管理不同的控制器。

## ➤ 控制器

读取硬件状态，发布控制命令，完成每个 joint 的控制。

## ➤ 硬件资源

为上下两层提供硬件资源的接口。

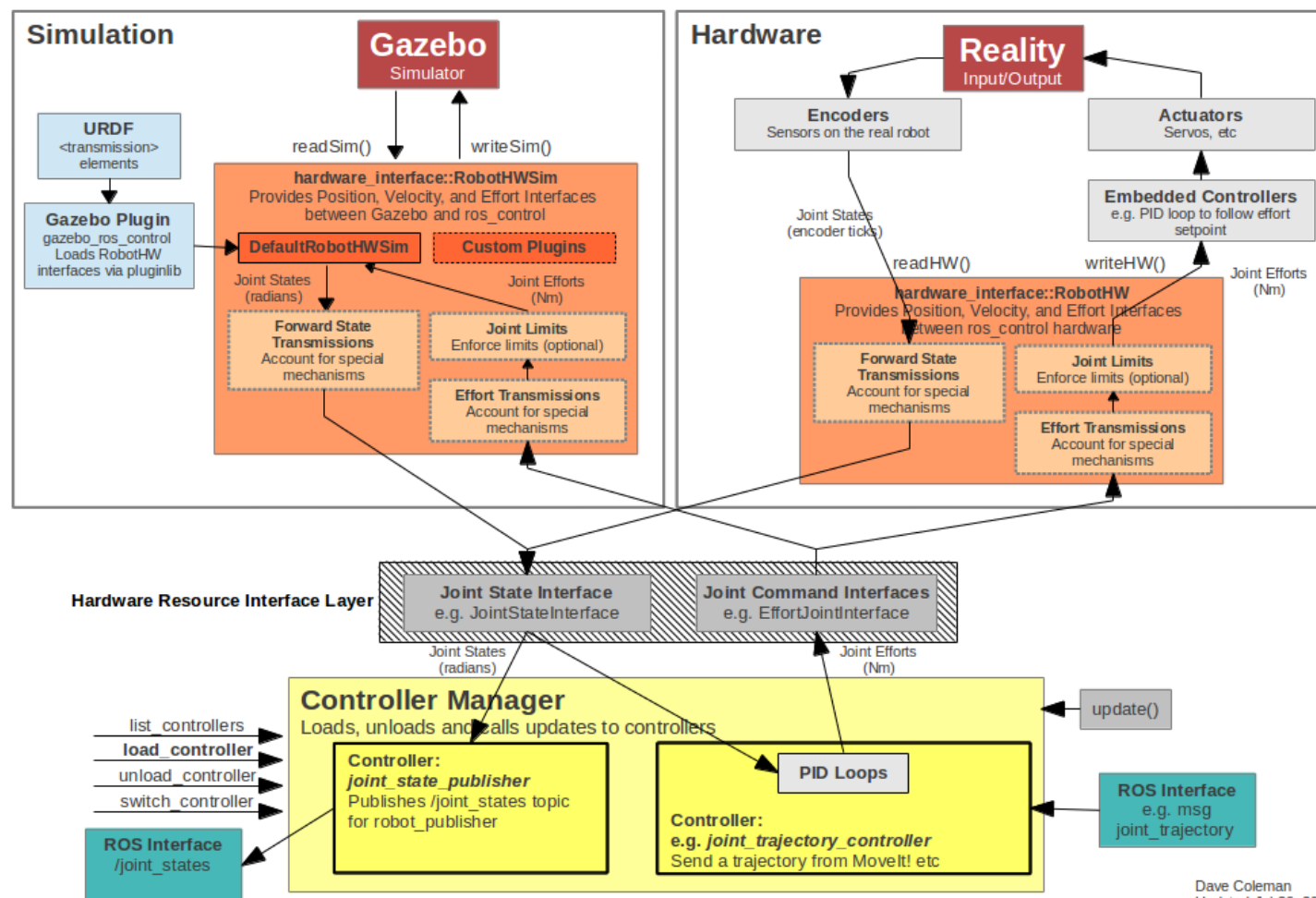
## ➤ 机器人硬件抽象

机器人硬件抽象和硬件资源直接打交道，通过 write 和 read 方法完成硬件操作。

## ➤ 真实机器人

执行接收到的命令。

GAZEBO + ROS + ros\_control

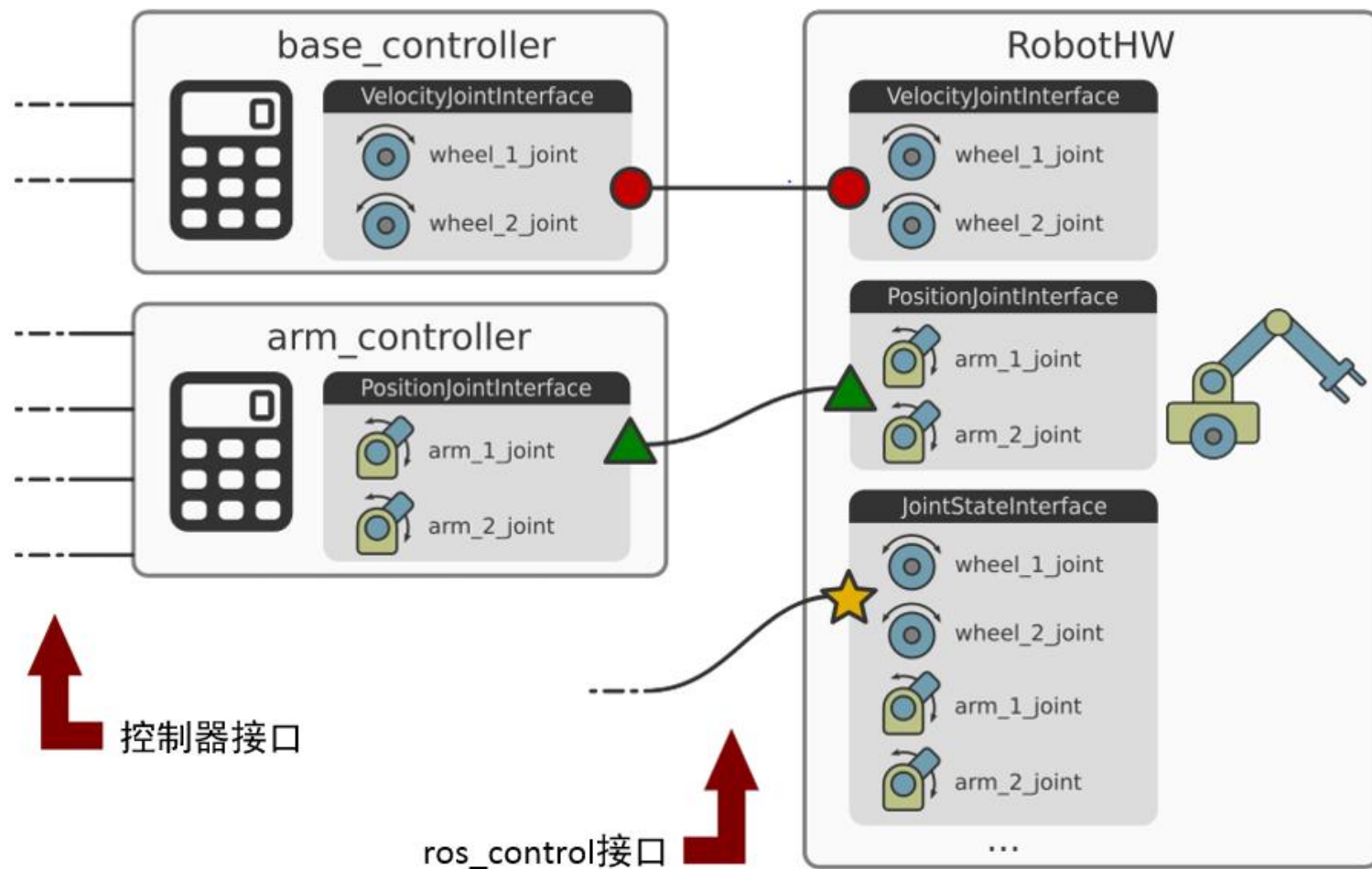




# 1. 优化物理仿真模型

## 控制器 (Controllers)

- joint\_state\_controller
- joint\_effort\_controller
- joint\_position\_controller
- joint\_velocity\_controller





# 1. 优化物理仿真模型

## 第一步：为link添加惯性参数和碰撞属性

```
<xacro:macro name="cylinder_inertial_matrix" params="m r h">
  <inertial>
    <mass value="${m}" />
    <inertia ixx="${m*(3*r*r+h*h)/12}" ixy = "0" ixz = "0"
      iyy="${m*(3*r*r+h*h)/12}" iyz = "0"
      izz="${m*r*r/2}" />
  </inertial>
</xacro:macro>

<link name="base_link">
  <visual>
    <origin xyz=" 0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder length="${base_length}" radius="${base_radius}" />
    </geometry>
    <material name="yellow" />
  </visual>
  <collision>
    <origin xyz=" 0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder length="${base_length}" radius="${base_radius}" />
    </geometry>
  </collision>
  <cylinder_inertial_matrix m="${base_mass}" r="${base_radius}" h="${base_length}" />
</link>
```



# 1. 优化物理仿真模型

## 第二步：为link添加gazebo标签

```
<gazebo reference="base_link">  
|   <material>Gazebo/Blue</material>  
</gazebo>
```

```
<gazebo reference="${prefix}_wheel_link">  
|   <material>Gazebo/Gray</material>  
</gazebo>
```

```
<gazebo reference="base_footprint">  
|   <turnGravityOff>false</turnGravityOff>  
</gazebo>
```

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





## 第三步：为joint添加传动装置

```
<!-- Transmission is important to link the joints and the controller -->
<transmission name="${prefix}_wheel_joint_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="${prefix}_wheel_joint" >
    <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
  </joint>
  <actuator name="${prefix}_wheel_joint_motor">
    <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```



## 第四步：添加gazebo控制器插件

- **<robotNamespace>**：机器人的命名空间。
- **<leftJoint>**和**<rightJoint>**：左右轮转动的关节joint。
- **<wheelSeparation>**和**<wheelDiameter>**：机器人模型的相关尺寸，在计算差速参数时需要用到。
- **<commandTopic>**：控制器订阅的速度控制指令，生成全局命名时需要结合<robotNamespace>中设置的命名空间。
- **<odometryFrame>**：里程计数据的参考坐标系，ROS中一般都命名为odom。

```
<!-- controller -->
<gazebo>
  <plugin name="differential_drive_controller"
    filename="libgazebo_ros_diff_drive.so">
    <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_wheel_joint</leftJoint>
    <rightJoint>right_wheel_joint</rightJoint>
    <wheelSeparation>${wheel_joint_y*2}</wheelSeparation>
    <wheelDiameter>${2*wheel_radius}</wheelDiameter>
    <broadcastTF>1</broadcastTF>
    <wheelTorque>30</wheelTorque>
    <wheelAcceleration>1.8</wheelAcceleration>
    <commandTopic>cmd_vel</commandTopic>
    <odometryFrame>odom</odometryFrame>
    <odometryTopic>odom</odometryTopic>
    <robotBaseFrame>base_footprint</robotBaseFrame>
  </plugin>
</gazebo>
```



# 1. 优化物理仿真模型

<launch>

```
<!-- 设置launch文件的参数 -->
<arg name="paused" default="false"/>
<arg name="use_sim_time" default="true"/>
<arg name="gui" default="true"/>
<arg name="headless" default="false"/>
<arg name="debug" default="false"/>

<!-- 运行gazebo仿真环境 -->
<include file="$(find gazebo_ros)/launch/empty_world.launch">
  <arg name="debug" value="$(arg debug)" />
  <arg name="gui" value="$(arg gui)" />
  <arg name="paused" value="$(arg paused)" />
  <arg name="use_sim_time" value="$(arg use_sim_time)" />
  <arg name="headless" value="$(arg headless)" />
</include>
```

```
<!-- 加载机器人模型描述参数 -->
<param name="robot_description" command="$(find xacro)/xacro --inorder '$(find
mbot_description)/urdf/mbot_gazebo.xacro'" />
```

```
<!-- 运行joint_state_publisher节点，发布机器人的关节状态 -->
<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" ></node>

<!-- 运行robot_state_publisher节点，发布tf -->
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" output="screen" >
  <param name="publish_frequency" type="double" value="50.0" />
</node>
```

```
<!-- 在gazebo中加载机器人模型-->
<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
  args="-urdf -model mrobot -param robot_description"/>
```

</launch>

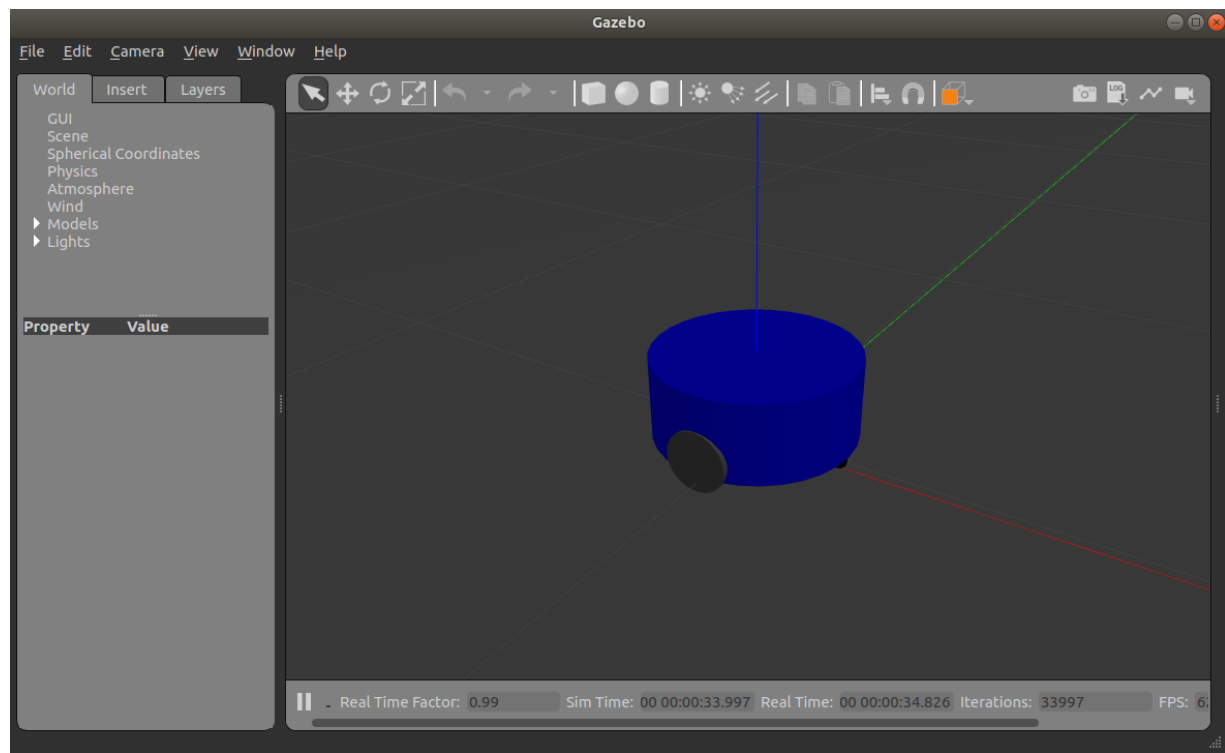
## 在gazebo中加载机器人模型

view\_mbot\_gazebo\_empty\_world.launch



# 1. 优化物理仿真模型

## 空环境中的机器人



```
$ roslaunch mbot_gazebo view_mbot_gazebo_empty_world.launch
```

**建议：**为保证模型顺利加载，请将离线模型文件库下载并放置到~/.gazebo/models下  
[https://bitbucket.org/osrf/gazebo\\_models/downloads/](https://bitbucket.org/osrf/gazebo_models/downloads/)

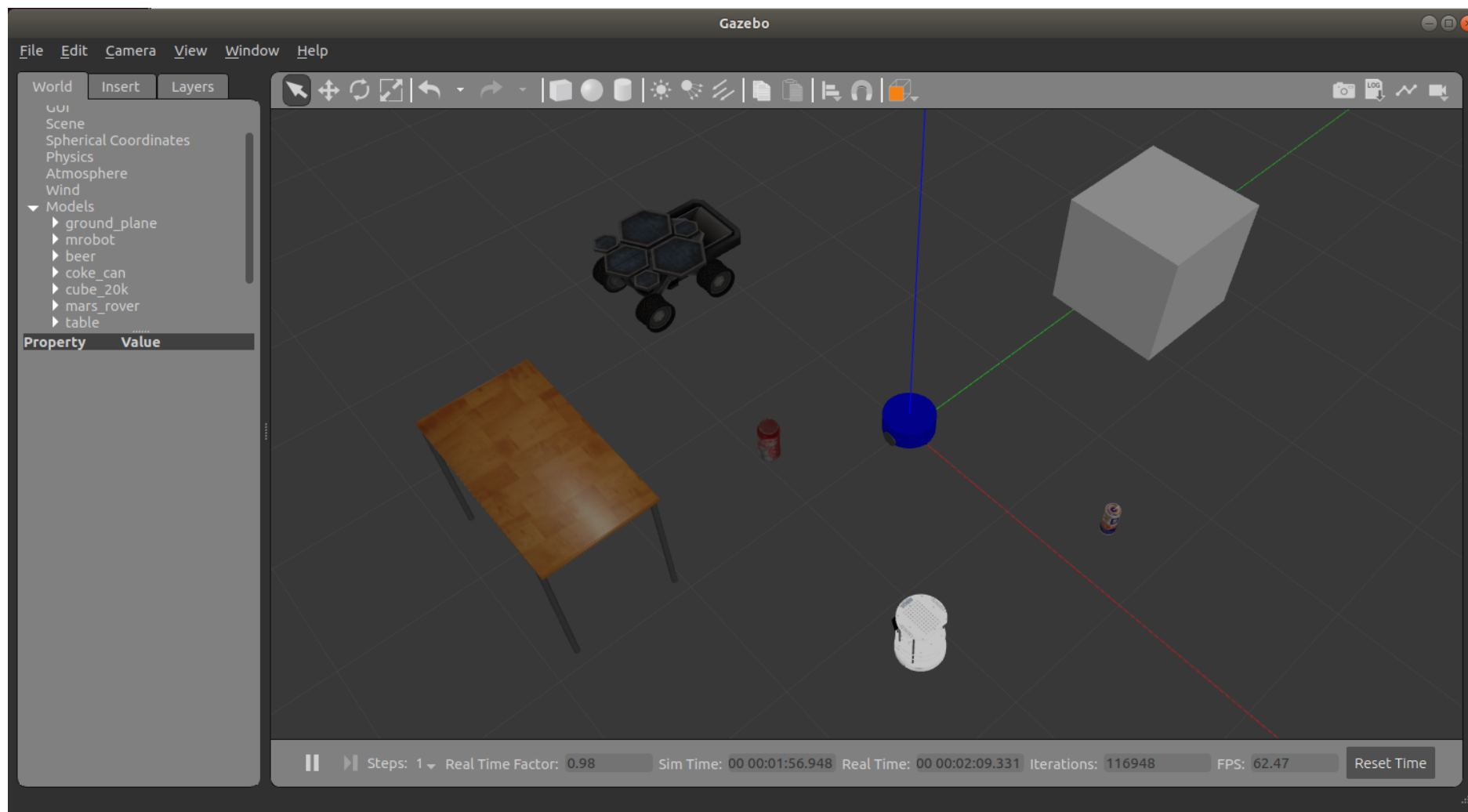


## 2. 创建物理仿真环境



## 2. 创建物理仿真环境

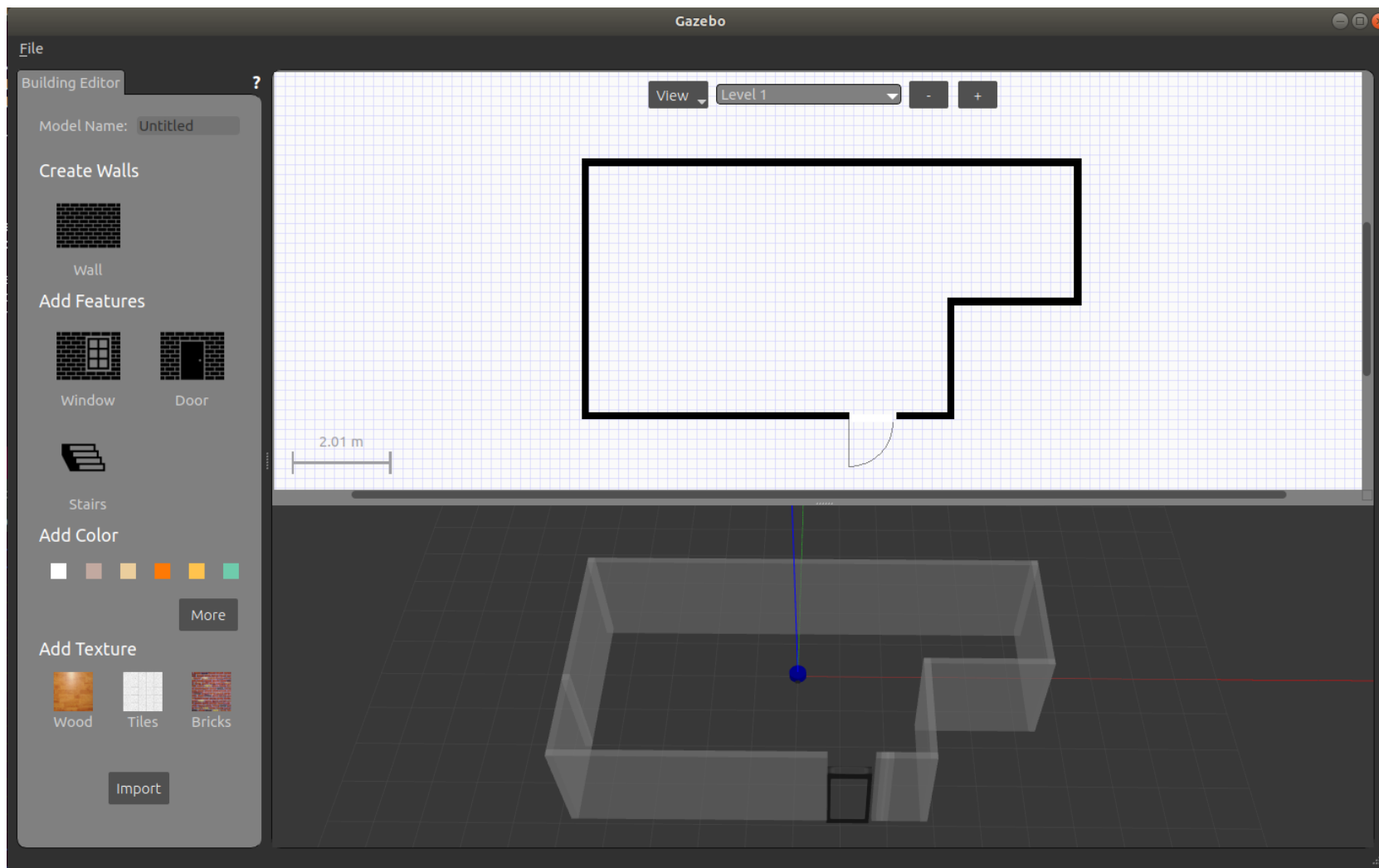
### 第一种方法：直接添加环境模型





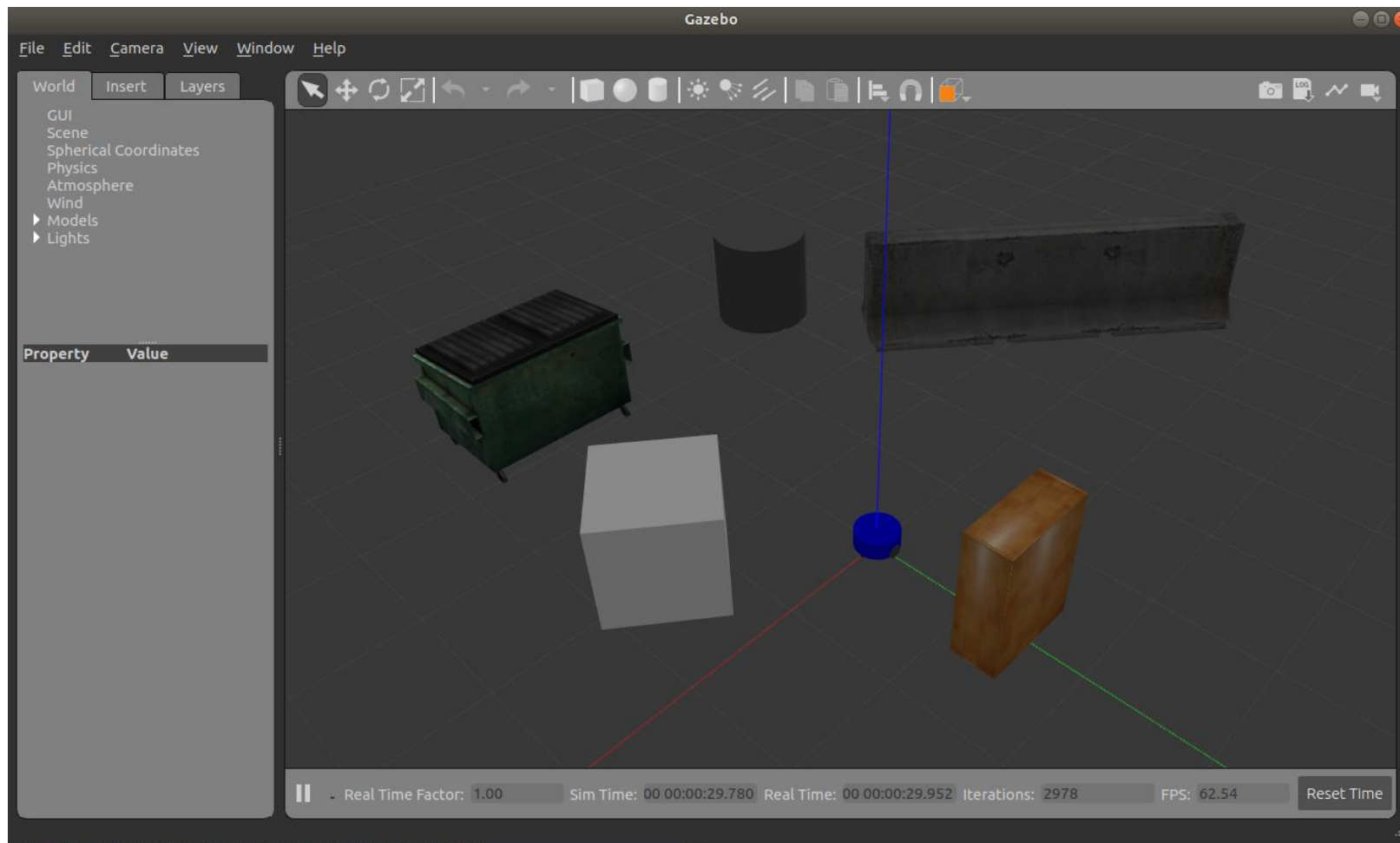
## 2. 创建物理仿真环境

### 第二种方法：使用Building Editor





## 2. 创建物理仿真环境



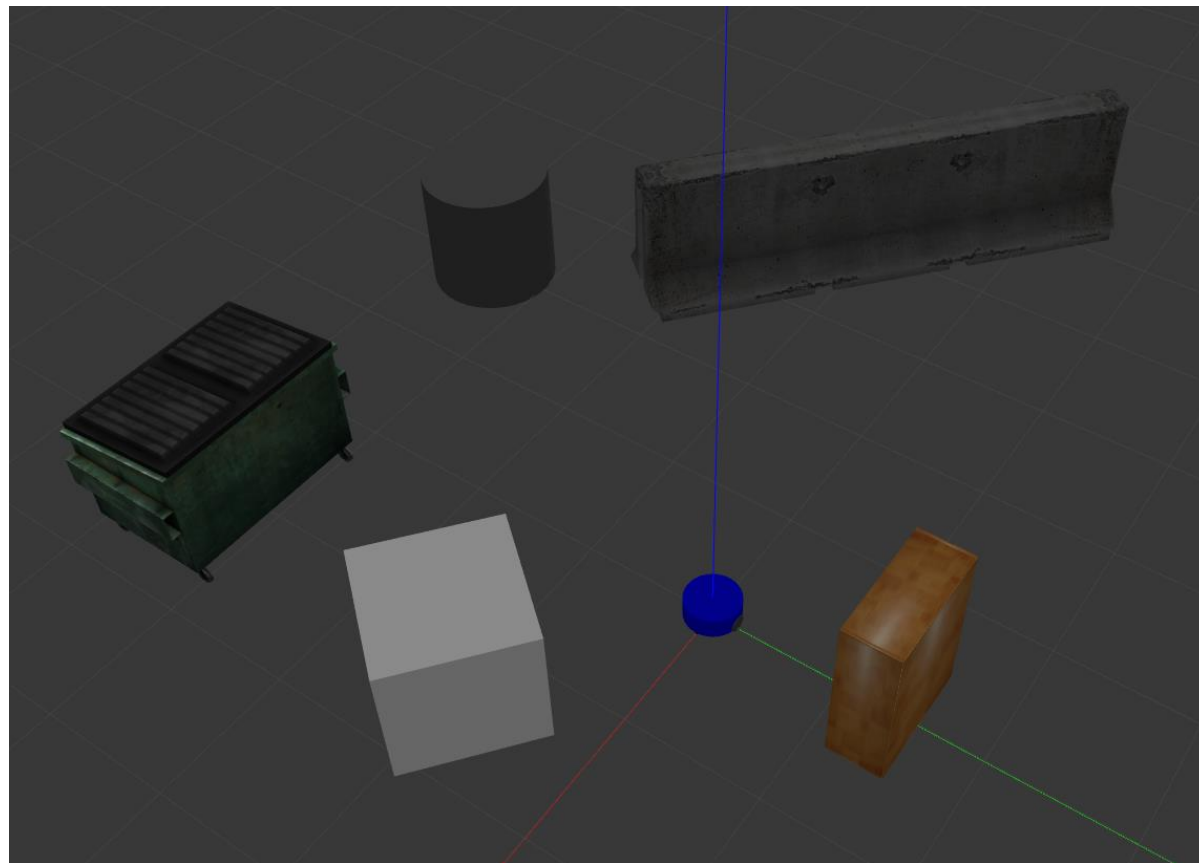
**启动仿真环境**    `roslaunch mbot_gazebo view_mbot_gazebo_play_ground.launch`





## 2. 创建物理仿真环境

```
hcx@hcx-pc:~$ rostopic list
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/joint_states
/odom
/rosout
/rosout_agg
/tf
/tf_static
```



**启动键盘控制**

`roslaunch mbot_teleop mbot_teleop.launch`



## 3. 传感器仿真及应用



## 3. 传感器仿真及应用

### 摄像头仿真

#### ➤ <sensor>标签：描述传感器

- type: 传感器类型, camera
- name: 摄像头命名, 自由设置

#### ➤ <camera>标签：描述摄像头参数

- 分辨率, 编码格式, 图像范围, 噪音参数等

#### ➤ <plugin>标签：加载摄像头仿真插件

#### libgazebo\_ros\_camera.so

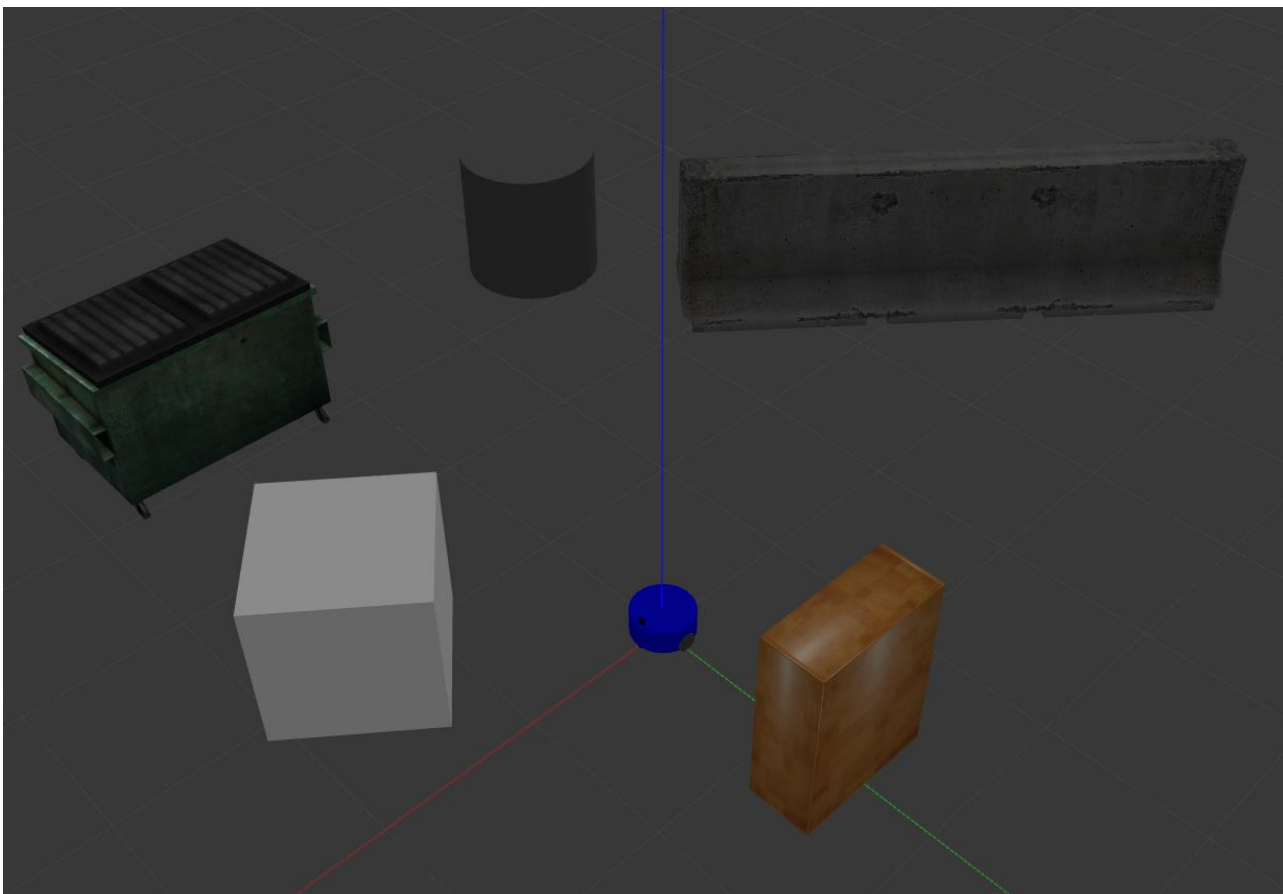
- 设置插件的命名空间、发布图像的话题、参考坐标系等

```
<gazebo reference="${prefix}_link">
  <sensor type="camera" name="camera_node">
    <update_rate>30.0</update_rate>
    <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>
    <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_link</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>
```

camera\_gazebo.xacro



### 3. 传感器仿真及应用



```
hcx@hcx-pc:~$ rostopic list
/camera/camera_info
/camera/image_raw
/camera/image_raw/compressed
/camera/image_raw/compressed/parameter_descriptions
/camera/image_raw/compressed/parameter_updates
/camera/image_raw/compressedDepth
/camera/image_raw/compressedDepth/parameter_descriptions
/camera/image_raw/compressedDepth/parameter_updates
/camera/image_raw/theora
/camera/image_raw/theora/parameter_descriptions
/camera/image_raw/theora/parameter_updates
/camera/parameter_descriptions
/camera/parameter_updates
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/joint_states
/odom
/rosout
/rosout_agg
/tf
/tf_static
```

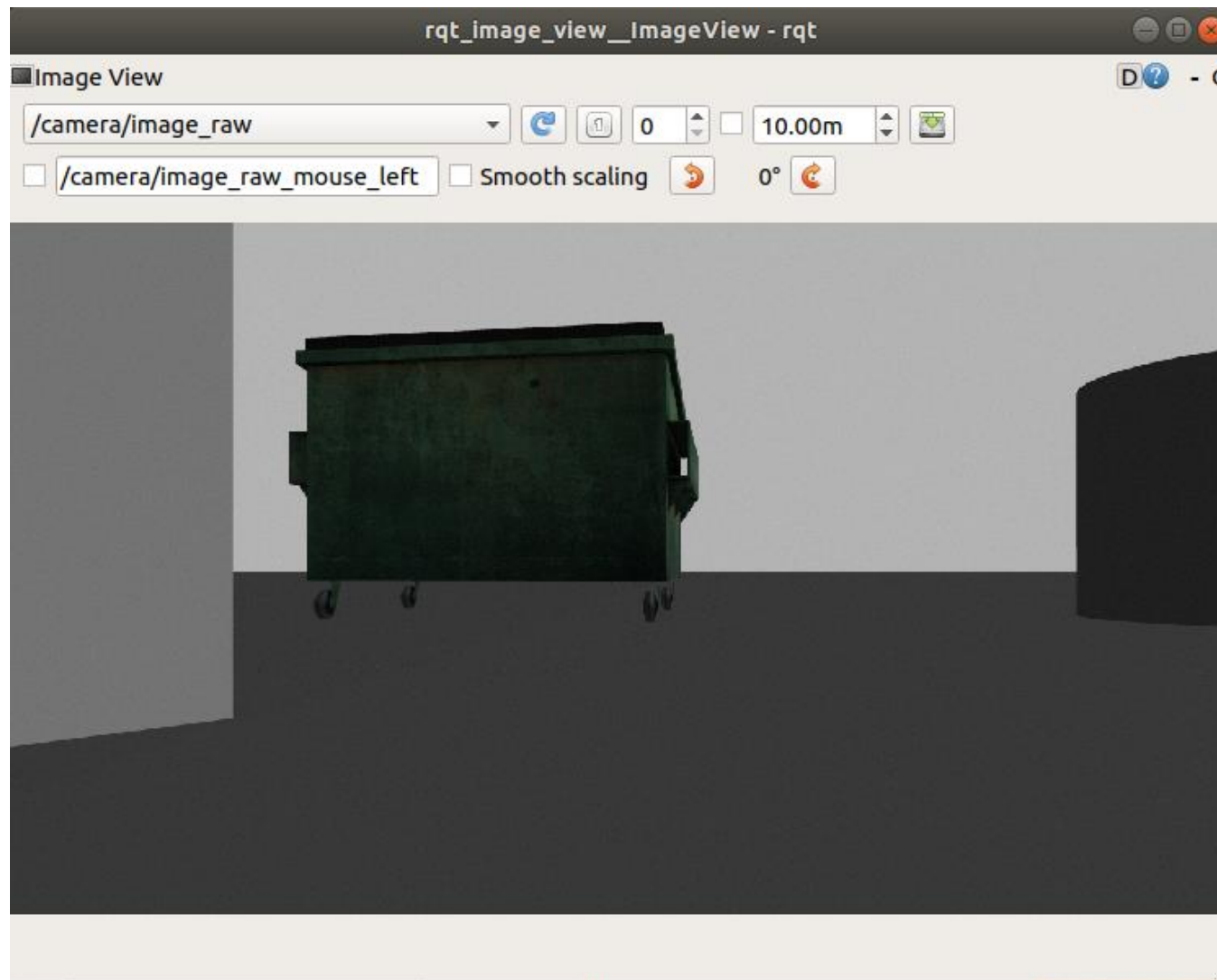
**启动仿真环境**    `roslaunch mbot_gazebo view_mbot_with_camera_gazebo.launch`



### 3. 传感器仿真及应用

查看摄像头仿真图像

\$ rqt\_image\_view





## 3. 传感器仿真及应用

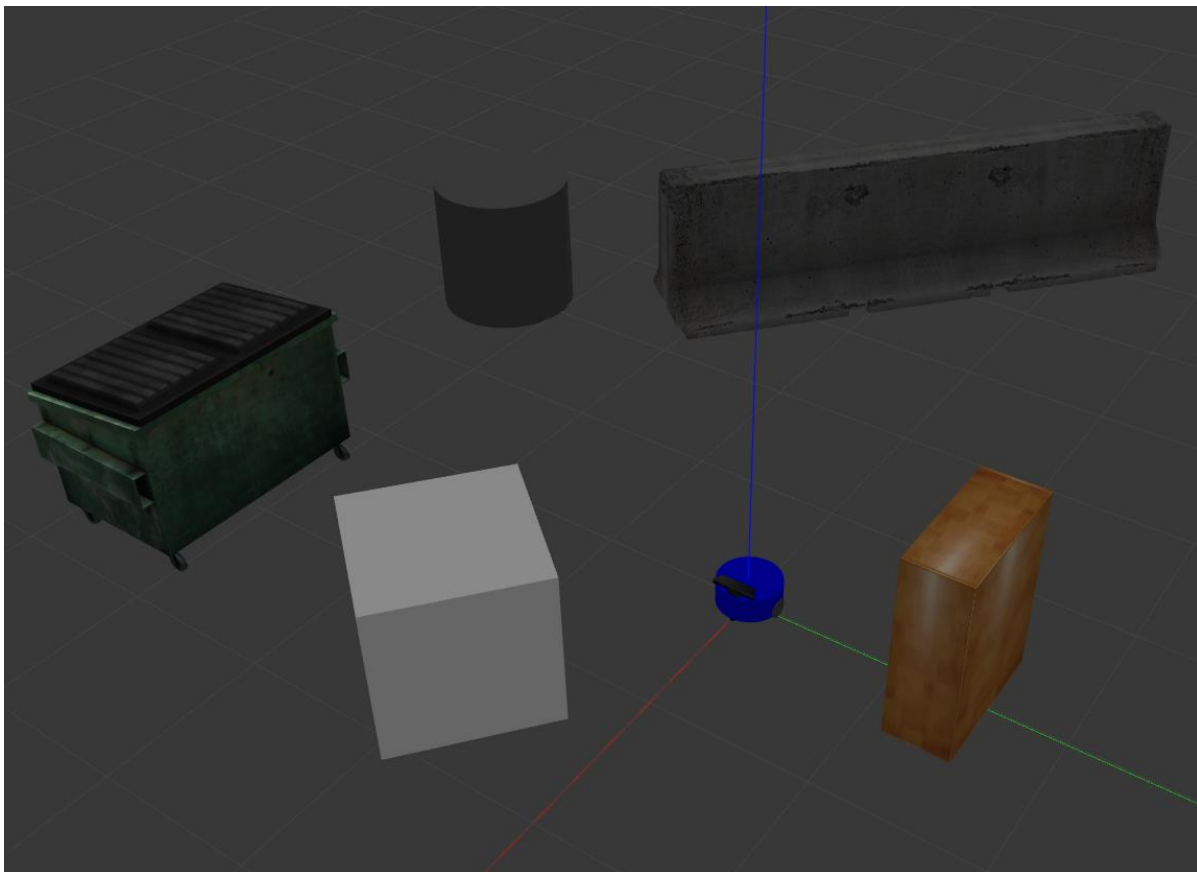
### RGB-D摄像头仿真 (kinect)

```
<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_kinect.so">
      <cameraName>${prefix}</cameraName>
      <alwaysOn>true</alwaysOn>
      <updateRate>10</updateRate>
      <imageTopicName>rgb/image_raw</imageTopicName>
      <depthImageTopicName>depth/image_raw</depthImageTopicName>
      <pointCloudTopicName>depth/points</pointCloudTopicName>
      <cameraInfoTopicName>rgb/camera_info</cameraInfoTopicName>
      <depthImageCameraInfoTopicName>depth/camera_info</depthImageCameraInfoTopicName>
      <frameName>${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>
```

kinect\_gazebo.xacro



### 3. 传感器仿真及应用



```
hcx@hcx-pc:~$ rostopic list
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/joint_states
/kinect/depth/camera_info
/kinect/depth/image_raw
/kinect/depth/points
/kinect/parameter_descriptions
/kinect/parameter_updates
/kinect/rgb/camera_info
/kinect/rgb/image_raw
/kinect/rgb/image_raw/compressed
/kinect/rgb/image_raw/compressed/parameter_descriptions
/kinect/rgb/image_raw/compressed/parameter_updates
/kinect/rgb/image_raw/compressedDepth
/kinect/rgb/image_raw/compressedDepth/parameter_descriptions
/kinect/rgb/image_raw/compressedDepth/parameter_updates
/kinect/rgb/image_raw/theora
/kinect/rgb/image_raw/theora/parameter_descriptions
/kinect/rgb/image_raw/theora/parameter_updates
/odom
/rosout
/rosout_agg
/tf
/tf_static
```

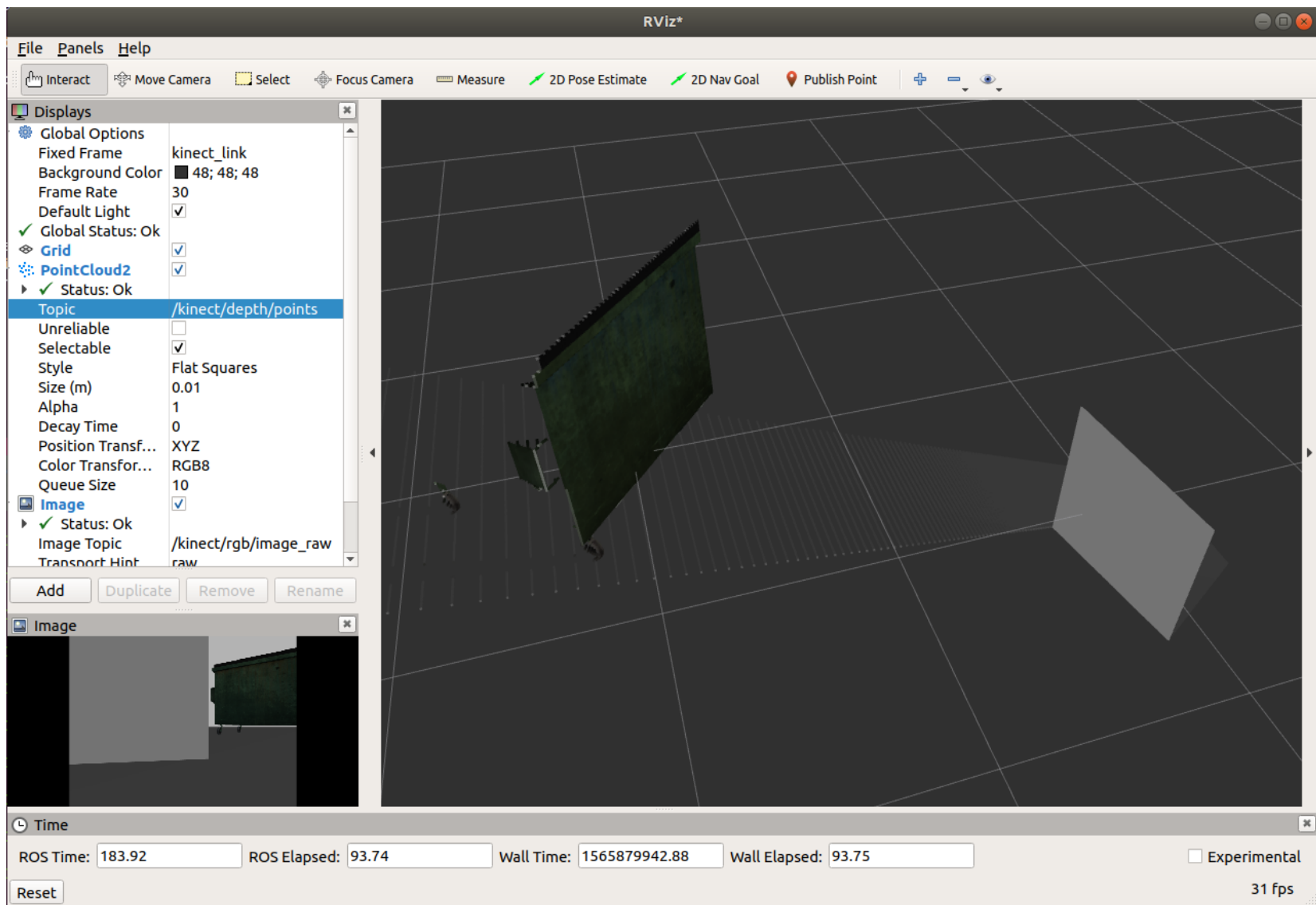
启动仿真环境 `roslaunch mbot_gazebo view_mbot_with_kinect_gazebo.launch`





### 3. 传感器仿真及应用

在rviz中查看Kinect信息







## 3. 传感器仿真及应用

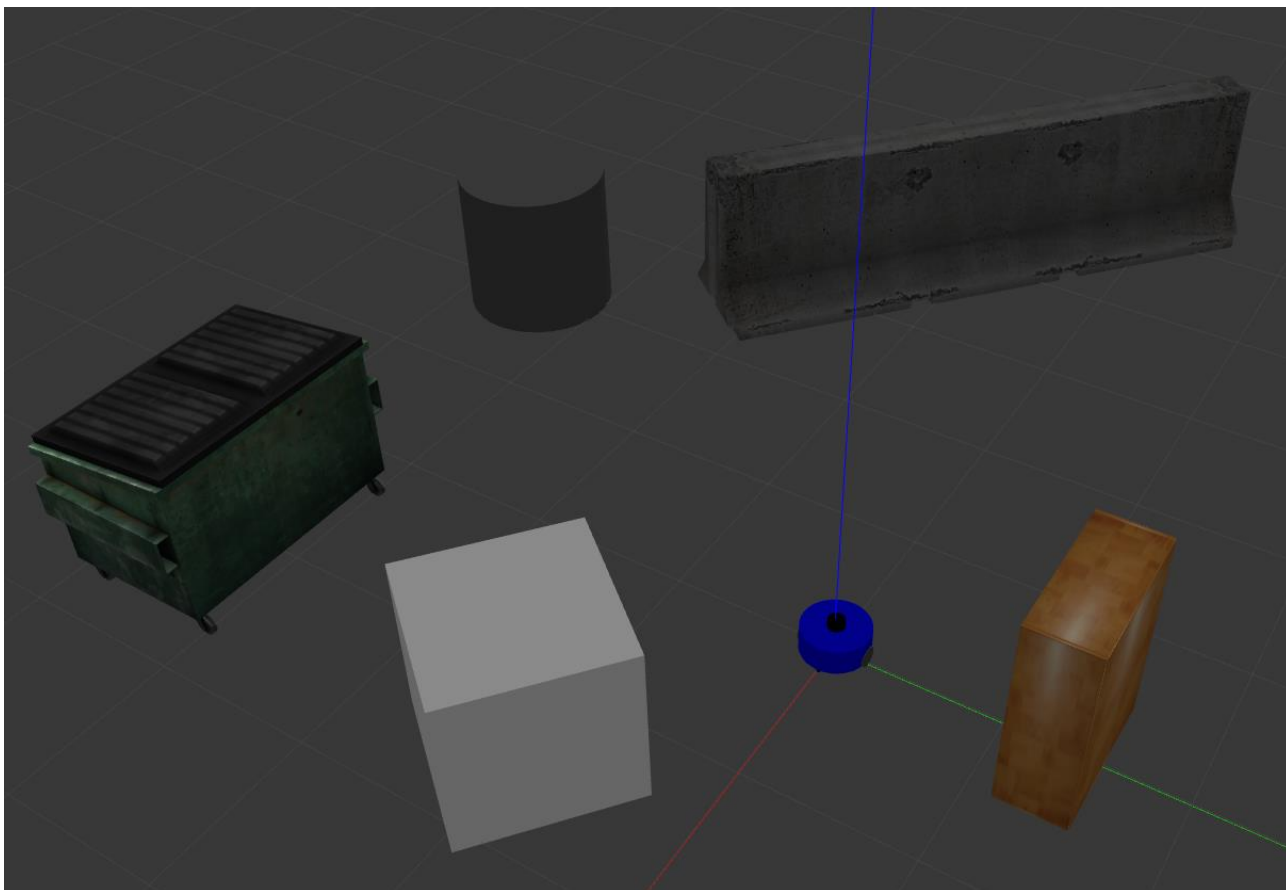
### 激光雷达仿真

```
<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>
```

lidar\_gazebo.xacro



### 3. 传感器仿真及应用



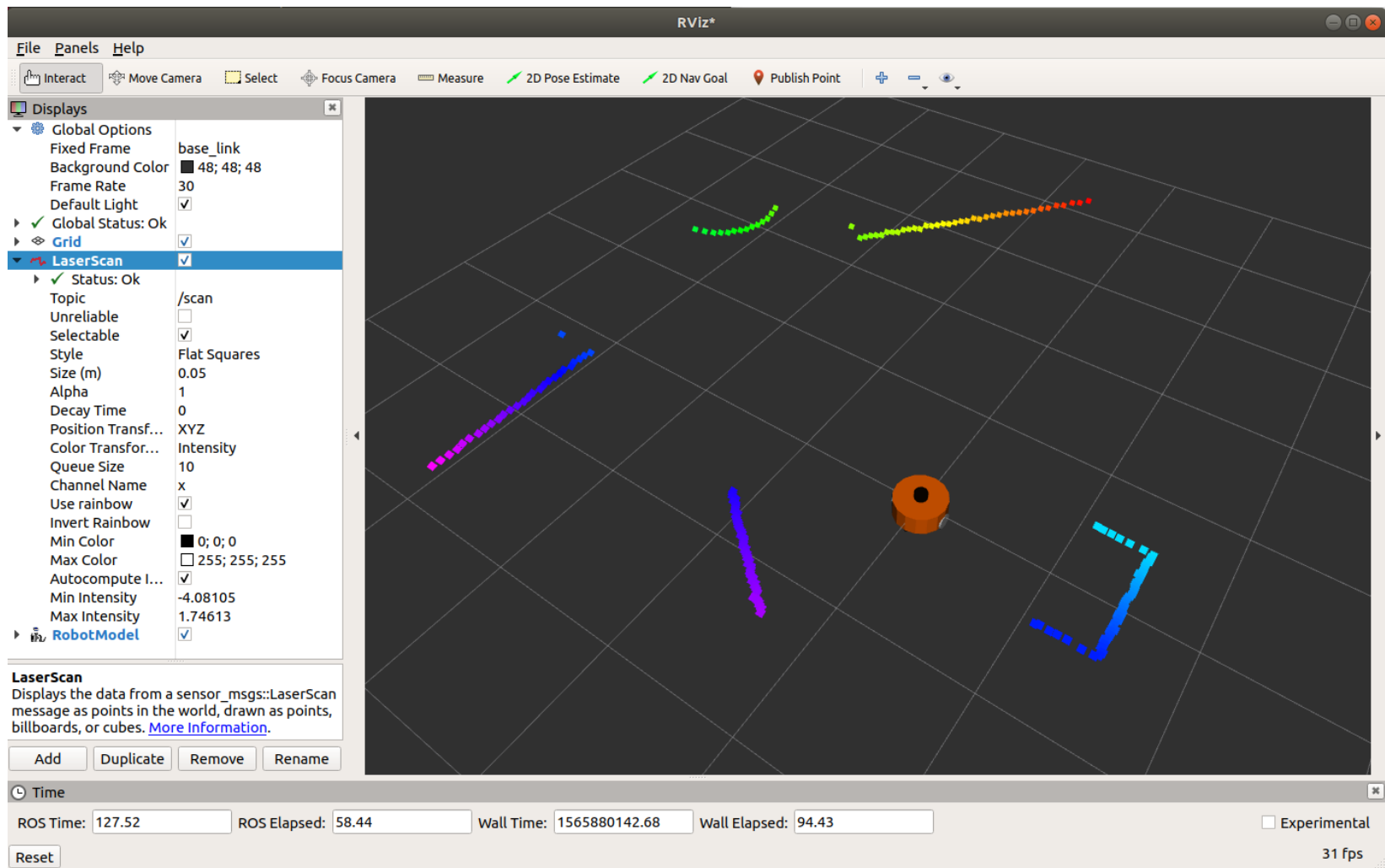
```
hcx@hcx-pc:~$ rostopic list
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/joint_states
/odom
/rosout
/rosout_agg
/scan
/tf
/tf_static
```

**启动仿真环境**     `roslaunch mbot_gazebo view_mbot_with_laser_gazebo.launch`



## 3. 传感器仿真及应用

在rviz中查看激光雷达信息





## 优化物理仿真模型

- 为link添加惯性参数和碰撞属性
- 为link添加gazebo标签
- 为joint添加传动装置
- 添加gazebo控制器插件

## 创建物理仿真环境

- 直接添加环境模型
- 使用Building Editor

## 传感器仿真及应用

摄像头、RGBD、激光雷达





1. 将上讲作业创建的机器人URDF模型，改写成xacro文件，创建仿真环境，将机器人模型加载到Gazebo仿真环境中，完成运动控制的仿真；
2. 在机器人模型上添加摄像头和激光雷达，将机器人模型加载到Gazebo中，完成传感器的仿真，并在rviz中显示传感器数据。



- ROS xacro使用方法  
<http://wiki.ros.org/xacro>
- rviz与Gazebo的区别  
<https://www.zhihu.com/question/268658280/answer/340190866>
- Gazebo仿真教程  
<http://gazebo-sim.org/tutorials>
- ROS探索总结（三十一）—— ros\_control  
<http://www.guyuehome.com/890>
- ROS探索总结（二十四）—— 使用gazebo中的插件  
<http://www.guyuehome.com/388>

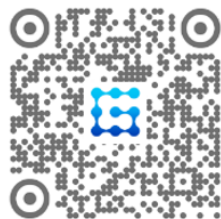




# Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注



 古月居



 古月春旭