深蓝学院
shenlanxueyuan.com

机器人操作系统ROS理论与实践

—— 第三讲：机器人系统设计

主讲人 胡春旭

机器人博客“古月居”博主（www.guyuehome.com）
深圳星河智能科技有限公司 联合创始人
华中科技大学自动化学院 硕士
Email：huchunxu@hust.edu.cn

课程概览

1. 认识ROS

- 课程介绍
- ROS现状与起源
- ROS总体架构
- ROS系统实现
- 第一个ROS例程

2. ROS基础

- 创建工作空间
- ROS通信编程
- 实现分布式通讯
- ROS中的关键组件

3. 机器人系统设计

- 机器人的定义与组成
- 机器人系统构建
- URDF机器人建模

4.机器人仿真

- 机器人模型优化
- ArbotiX+rviz功能仿真
- gazebo物理仿真

5.机器人感知

- 机器视觉
(图像校准、图像识别等)
- 机器语音
(科大讯飞SDK)

6.机器人SLAM与自主导航

- 机器人必备条件
- SLAM功能包的应用
- ROS中的导航框架
- 导航框架的应用

7. MoveIt!机械臂控制

- MoveIt!系统架构
- 创建机械臂模型
- MoveIt!编程学习
- Gazebo机械臂仿真
- ROS-I框架介绍

8. ROS机器人综合应用

- ROS机器人实例介绍
(PR2、Turtlebot、HRMRP、Kungfu Arm)
- 构建综合机器人平台

9. ROS 2.0

- 为什要有ROS 2
- 什么是ROS 2
- 如何安装ROS 2
- 话题与服务编程
- ROS 2与ROS 1的集成
- 课程总结与展望

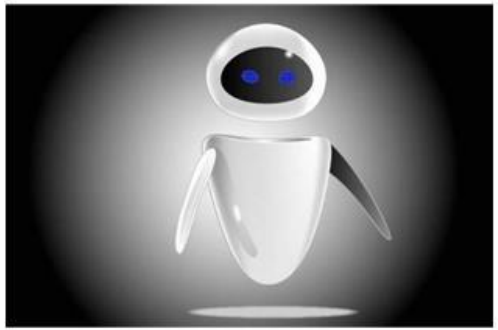
问题汇总及作业指导

目录

1. 机器人的定义与组成

机器人是什么

1. 机器人的定义与组成——机器人是什么



理想中的机器人

1. 机器人的定义与组成——机器人是什么

现实中的机器人

1. 机器人的定义与组成——机器人是什么

诞生	1920年捷克斯洛伐克作家卡雷尔·恰佩克在他的科幻小说《罗萨姆的机器人万能公司》中，根据Robota（捷克文，原意为“劳役、苦工”）和Robotnik（波兰文，原意为“工人”），创造出“机器人”这个词。
百度百科	机器人（Robot）是自动执行工作的机器装置。它既可以接受人类指挥，又可以运行预先编排的程序，也可以根据以人工智能技术制定的原则纲领行动。它的任务是协助或取代人类工作，例如生产业、建筑业，或是危险的工作
美国机器人协会（RIA）	机器人是用以搬运材料、零件、工具的可编程序的多功能操作器或是通过可改变程序动作来完成各种作业的特殊机械装置

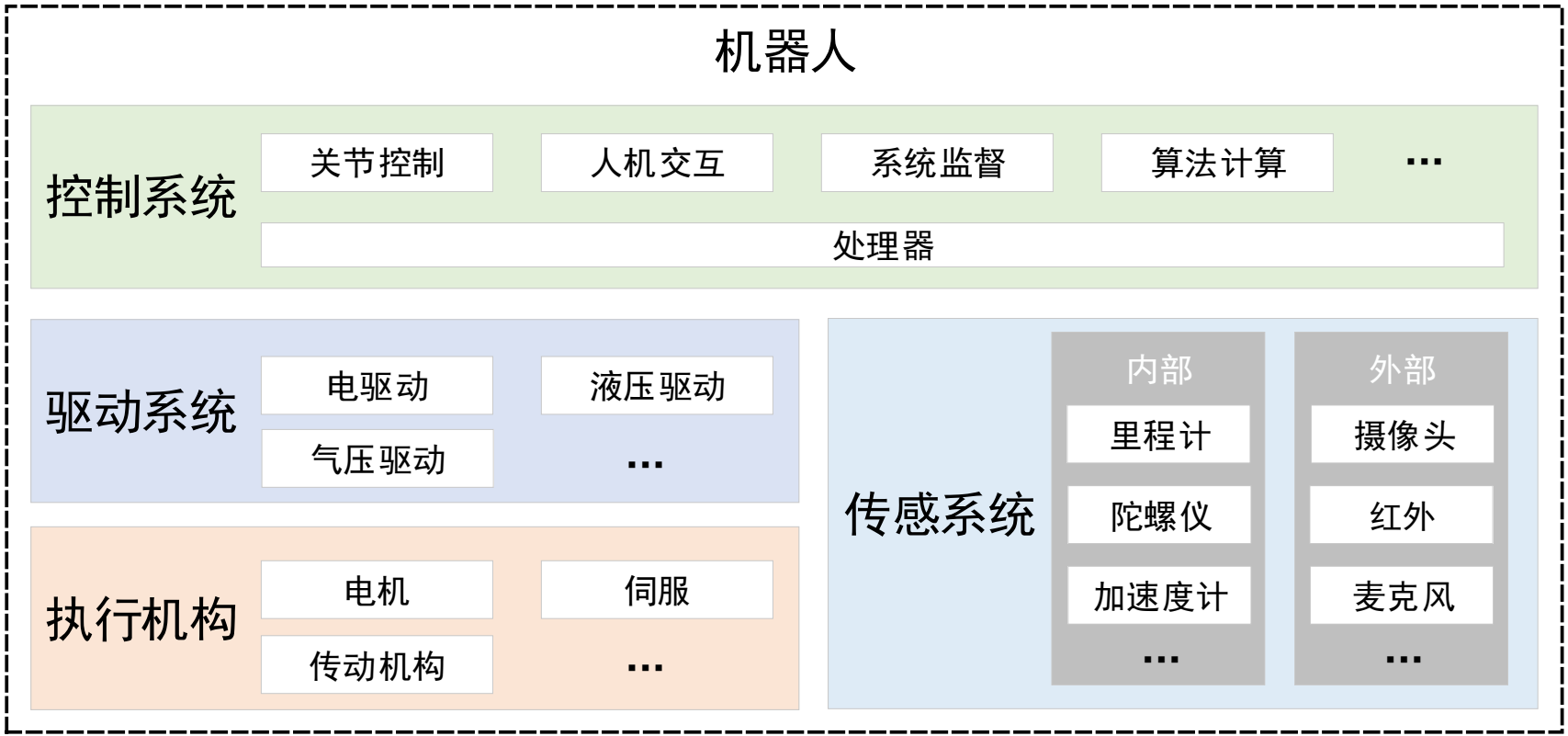
1. 机器人的定义与组成——机器人是什么

国际标准化组织(ISO)

1. 机器人的动作机构具有类似于人或其他生物体的某些器官（肢体、感受等）的功能；
2. 机器人具有通用性，工作种类多样，动作程序灵活易变；
3. 机器人具有不同程度的智能性，如记忆、感知、推理、决策、学习等；
4. 机器人具有独立性，完整的机器人系统在工作中可以不依赖于人的干预。

机器人是如何组成的

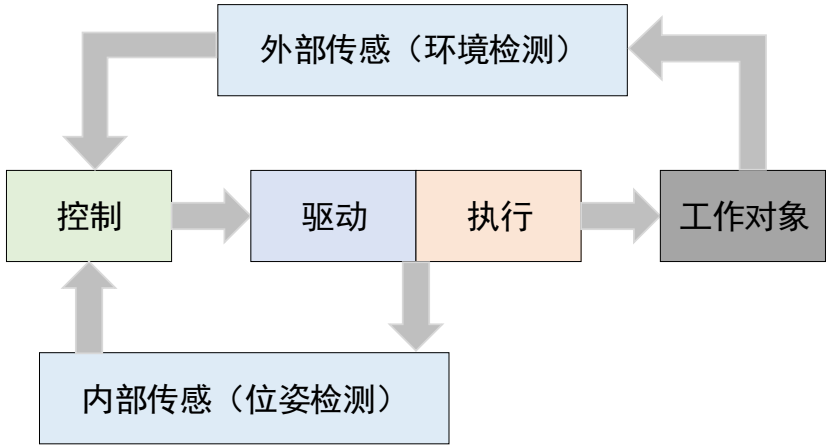
1. 机器人的定义与组成——机器人是如何组成的



机器人的组成（控制的角度）

1. 机器人的定义与组成——机器人是如何组成的

- **执行机构**：人体的手和脚，直接面向工作对象的机械装置。
- **驱动系统**：人体的肌肉和筋络，负责驱动执行机构，将控制系统下达的命令转换成执行机构需要的信号。
- **传感系统**：人体的感官和神经，主要完成信号的输入和反馈，包括内部传感系统和外部传感系统。
- **控制系统**：人体的大脑，实现任务及信息的处理，输出控制命令信号。

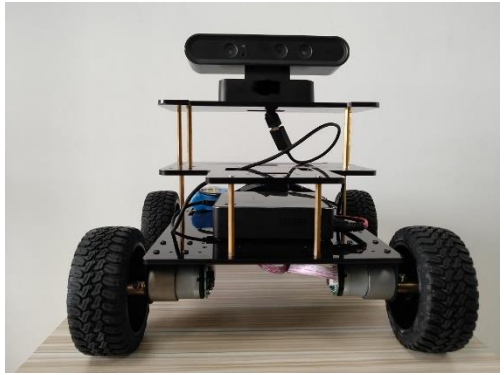
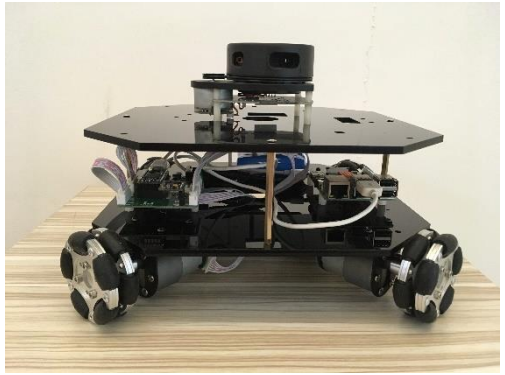


机器人的控制回路

2. 机器人系统构建

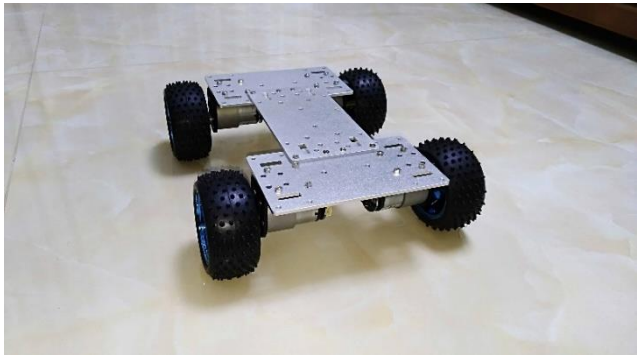
- 执行机构的实现
- 驱动系统的实现
- 内部传感系统的实现
- 控制系统的实现
- 外部传感系统的实现

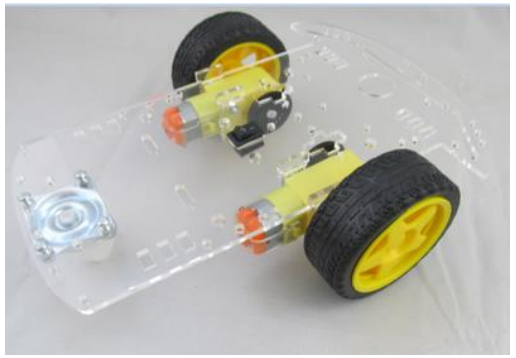
2. 机器人系统构建


机器人系统示例

2. 机器人系统构建——执行机构的实现

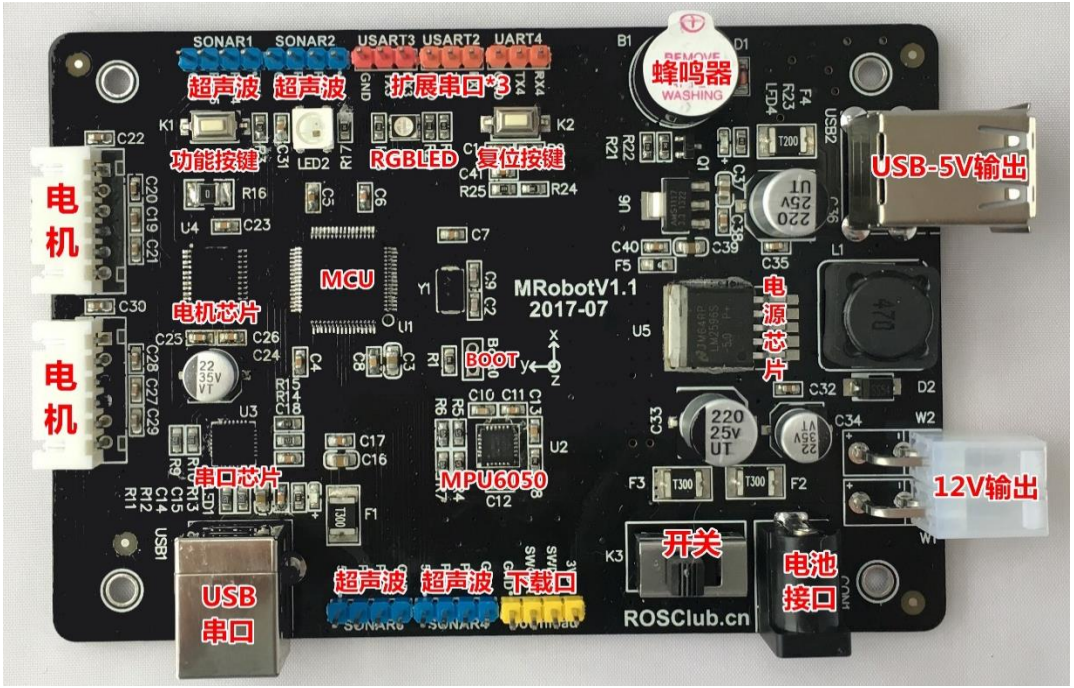




机器人底盘、电机、舵机

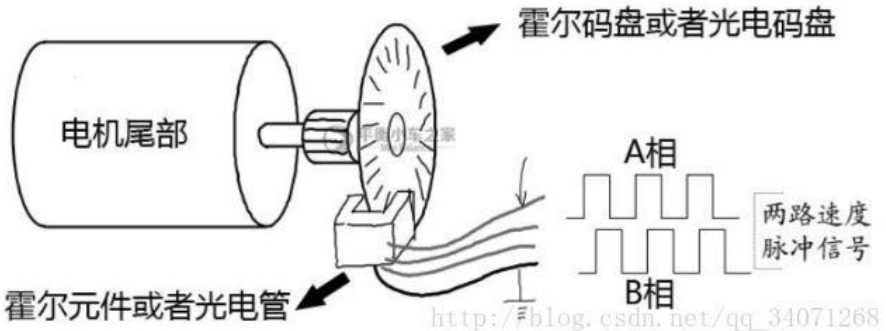
2. 机器人系统构建——驱动系统的实现

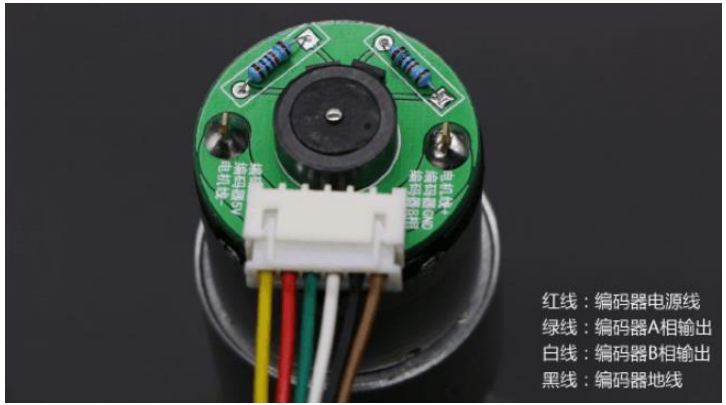
1. 电源子系统
   - 12V/5V/3V
   - 电源保护、滤波
2. 电机驱动子系统
   - 电机驱动：控制信号→电信号
   - 电机控制：闭环驱动
3. 传感器接口
   - 超声波
   - 里程计
   - …



机器人驱动板

2. 机器人系统构建——内部传感系统的实现



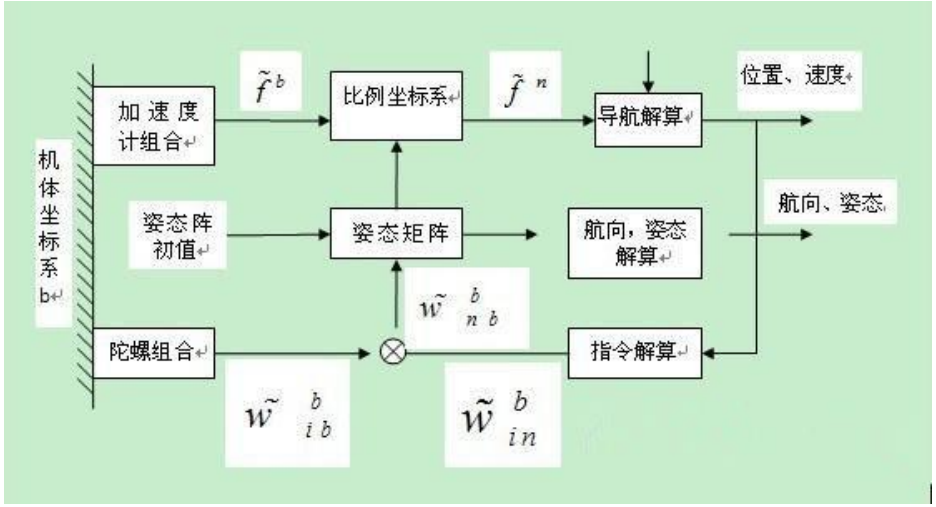


机器人里程计测距原理

1. 根据单位时间内产生的脉冲数计算电机/轮子的旋转圈数
2. 根据轮子的周长计算机器人的运动速度
3. 根据机器人的运动速度积分计算里程

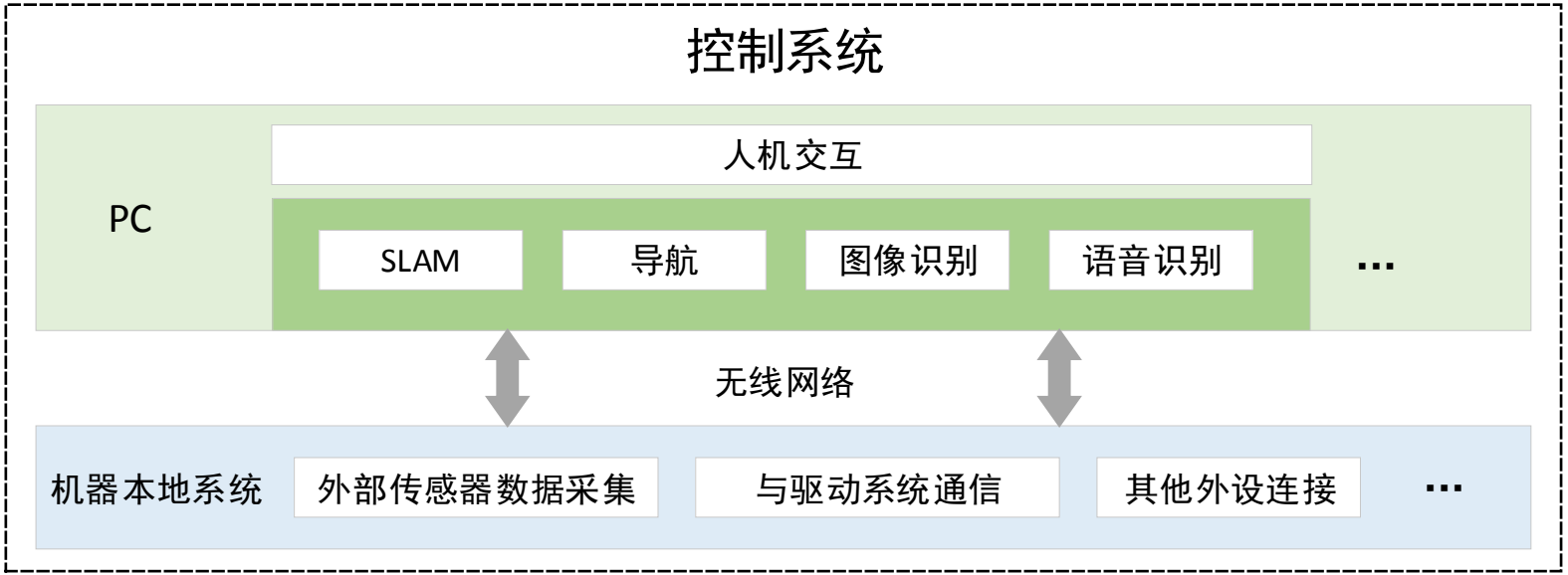
2. 机器人系统构建——内部传感系统的实现

惯性测量单元（IMU）：
测量物体速度姿态，主要包括三轴陀螺、三轴加速度计、磁力计等



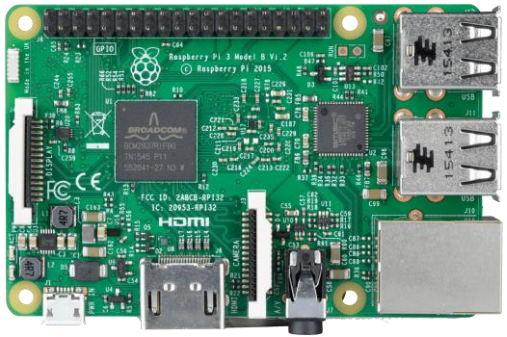
IMU实现原理

2. 机器人系统构建——控制系统的实现



常用控制系统实现框架

2. 机器人系统构建——控制系统的实现



Raspberry Pi

Odroid-XU4

Nvidia Jetson TK1

Mini PC

Intel NUC

ARM架构开发板可参考：http://wiki.ros.org/indigo/Installation/UbuntuARM

2. 机器人系统构建——外部传感系统的实现

常用传感器的ROS驱动可参考：http://wiki.ros.org/Sensors

2. 机器人系统构建——连接摄像头

```
$ sudo apt-get install ros-kinetic-usb-cam
$ roslaunch usb_cam usb_cam-test.launch
$ rqt_image_view
```

usb_cam功能包中的话题

	名称	类型	描述
Topic发布	~<camera_name>/image	sensor_msgs/Image	发布图像数据

usb_cam功能包中的参数

参数	类型	默认值	描述
~video_device	string	"/dev/video0"	摄像头设备号
~image_width	int	640	图像横向分辨率
~image_height	int	480	图像纵向分辨率
~pixel_format	string	"mjpeg"	像素编码，可选值：mjpeg，yuyv， uyvy
~io_method	string	"mmap"	IO通道，可选值：mmap，read，userptr
~camera_frame_id	string	"head_camera"	摄像头坐标系
~framerate	int	30	帧率
~brightness	int	32	亮度，0~255
~saturation	int	32	饱和度，0~255
~contrast	int	32	对比度，0~255
~sharpness	int	22	清晰度，0~255
~autofocus	bool	false	自动对焦
~focus	int	51	焦点（非自动对焦状态下有效）
~camera_info_url	string	-	摄像头校准文件路径
~camera_name	string	"head_camera"	摄像头名称

2. 机器人系统构建——连接摄像头

```
<launch>
    <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
        <param name="video_device" value="/dev/video0" />
        <param name="image_width" value="640" />
        <param name="image_height" value="480" />
        <param name="pixel_format" value="yuyv" />
        <param name="camera_frame_id" value="usb_cam" />
        <param name="io_method" value="mmap"/>
    </node>
    <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
        <remap from="image" to="/usb_cam/image_raw"/>
        <param name="autosize" value="true" />
    </node>
</launch>
```

usb_cam-test.launch

2. 机器人系统构建——连接Kinect

$ sudo apt-get install ros-kinetic-freenect-*

$ git clone https://github.com/avin2/SensorKinect.git

$ cd SensorKinect/Bin

$ tar xvf SensorKinect093-Bin-Linux-x86-v5.1.2.1.tar.bz2

$ sudo ./install.sh

freenect_camera功能包中的话题和服务

	名称	类型	描述
Topic 发布	rgb/camera_info	sensor_msgs/CameraInfo	RGB相机校准信息
	rgb/image_raw	sensor_msgs/Image	RGB相机图像数据
	depth/camera_info	sensor_msgs/CameraInfo	深度相机校准信息
	depth/image_raw	sensor_msgs/Image	深度相机数据
	depth_registered/camera_info	sensor_msgs/CameraInfo	配准后的深度相机校准信息
	depth_registered/image_raw	sensor_msgs/Image	配准后的深度相机数据
	ir/camera_info	sensor_msgs/CameraInfo	红外相机校准信息
	ir/image_raw	sensor_msgs/Image	红外相机数据
	projector/camera_info	sensor_msgs/CameraInfo	深度相机校准信息
	/diagnostics	diagnostic_msgs/DiagnosticArray	传感器诊断信息
Services	rgb/set_camera_info	sensor_msgs/SetCameraInfo	设置RGB相机的校准信息
	ir/set_camera_info	sensor_msgs/SetCameraInfo	设置红外相机的校准信息

2. 机器人系统构建——连接Kinect

```
<launch>

    <!-- 启动freenect驱动 -->
    <include file="$(find freenect_launch)/launch/freenect.launch">
        <arg name="publish_tf"                    value="false" />
        <arg name="depth_registration"            value="true" />
        <arg name="rgb_processing"                value="true" />
        <arg name="ir_processing"                 value="false" />
        <arg name="depth_processing"              value="false" />
        <arg name="depth_registered_processing"   value="true" />
        <arg name="disparity_processing"          value="false" />
        <arg name="disparity_registered_processing" value="false" />
        <arg name="sw_registered_processing"      value="false" />
        <arg name="hw_registered_processing"      value="true" />
    </include>

</launch>
```

freenect.launch

2. 机器人系统构建——连接激光雷达

$ sudo apt-get install ros-kinetic-rplidar-ros

$ rosrun rplidar_ros rplidarNode

rplidar功能包中的话题和服务

	名称	类型	描述
Topic发布	scan	sensor_msgs/LaserScan	发布激光雷达数据
Services	stop_motor	std_srvs/Empty	停止旋转电机
	start_motor	std_srvs/Empty	开始旋转电机

rplidar功能包中的参数

参数	类型	默认值	描述
serial_port	string	"/dev/ttyUSB0"	激光雷达串口名称
serial_baudrate	int	115200	串口波特率
frame_id	string	"laser"	激光雷达坐标系
inverted	bool	false	是否倒置安装
angle_compensate	bool	true	角度补偿

2. 机器人系统构建——连接激光雷达

```
robot@raspi2:~$ rosrun rplidar_ros rplidarNode
RPLIDAR running on ROS package rplidar_ros
SDK Version: 1.5.7
RPLIDAR S/N: 96DFFBF2C8E4DCCFC6E49FF1680D130D
Firmware Ver: 1.20
Hardware Rev: 0
RPLidar health status : 0
```

```
➜  ~ rostopic list
/rosout
/rosout_agg
/scan
```

启动rplidar激光雷达

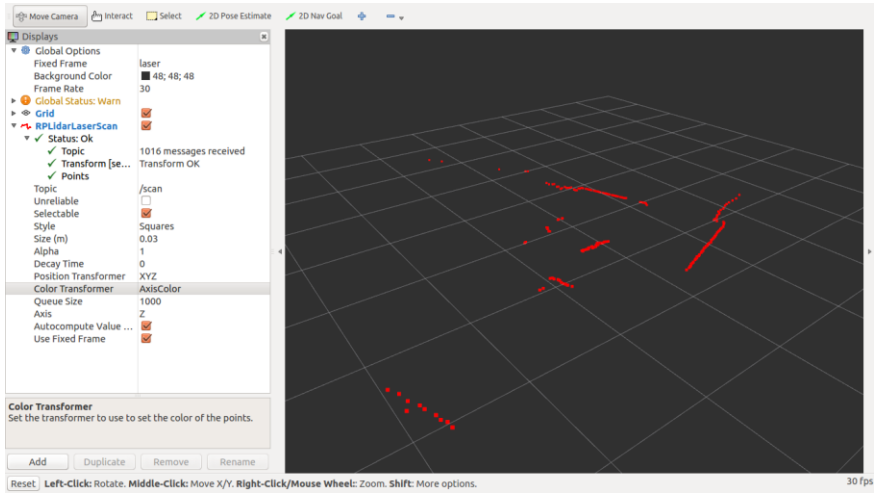
串口权限问题

$ sudo gpasswd --add USER_NAME dialout

```
NODES
  /
    rplidarNode (rplidar_ros/rplidarNode)
    rviz (rviz/rviz)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[rplidarNode-1]: started with pid [22497]
process[rviz-2]: started with pid [22498]
Error, cannot bind to the specified serial port /dev/ttyUSB0.
```

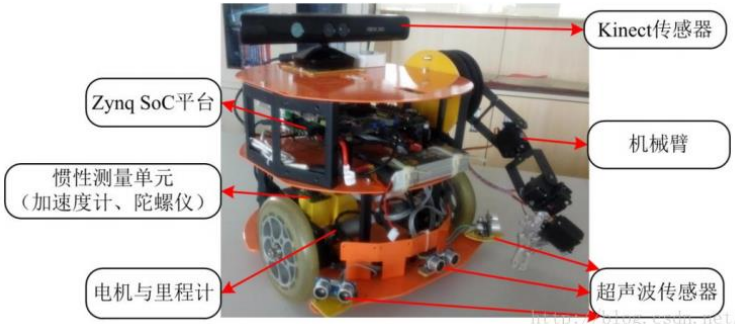
2. 机器人系统构建——连接激光雷达

```
→  ~ rostopic echo /scan
header:
  seq: 285
  stamp:
    secs: 1508991290
    nsecs: 876669183
  frame_id: laser_frame
angle_min: -3.12413907051
angle_max: 3.14159274101
angle_increment: 0.0174532923847
time_increment: 4.27053976182e-07
scan_time: 0.000153312372277
range_min: 0.15000000596
range_max: 8.0
ranges: [inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf,
 0.6140000224113464, 0.6449999809265137, 0.6790000200271606, 0.7170000076293945, inf, 0.
75, 0.9150000214576721, inf, inf, inf, inf, inf, inf, 0.5419999957084656, 0.538999974727
6306, inf, inf, 1.4670000076293945, 1.5980000495910645, 1.7549999952316284, 1.9620000123
977661, 2.193000078201294, inf, 2.5250000953674316, inf, inf, 2.86899995803833, 2.868999
95803833, 2.867000102996826, 2.865000009536743, 2.868000030517578, 2.868000030517578, in
f, 2.865999937057495, 2.869999885559082, 2.874000072479248, 2.872999906539917, 2.8870000
8392334, 2.8910000324249268, inf, 2.2880001068115234, 2.260999917984009, 2.2320001125335
693, inf, 2.188999891281128, 2.755000114440918, inf, 2.7769999504089355, 2.7960000038146
973, 2.806999921798706, inf, 2.740999937057495, inf, inf, 2.7269999980926514, 2.69899988
```

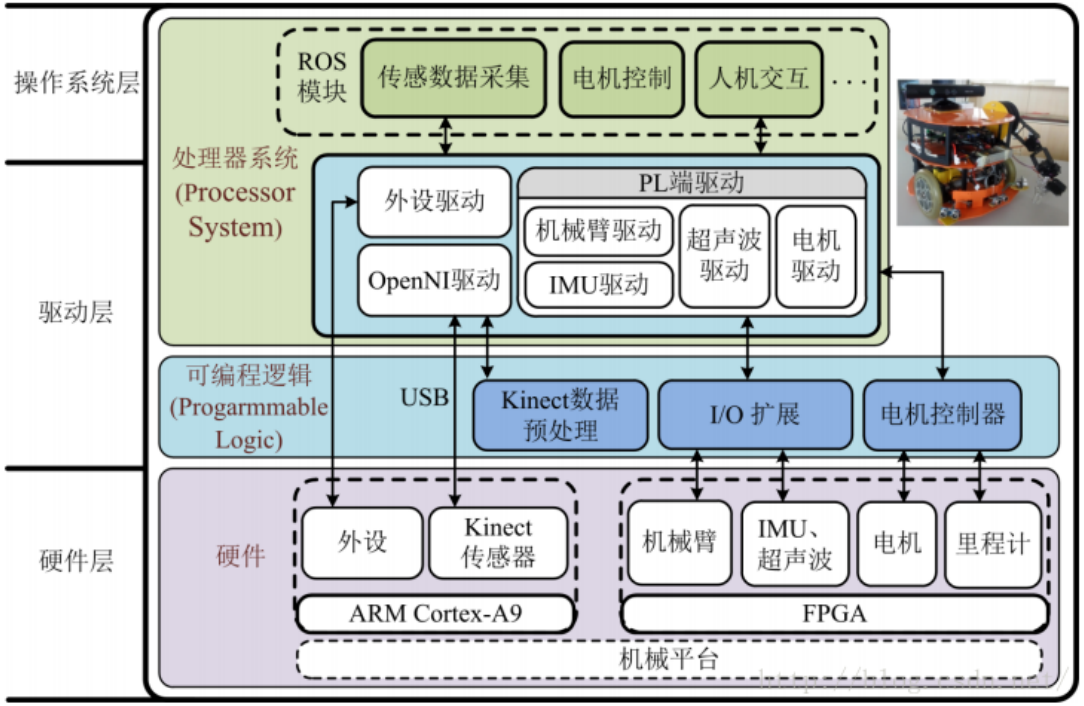


激光雷达的数据显示

2. 机器人系统构建——完整示例

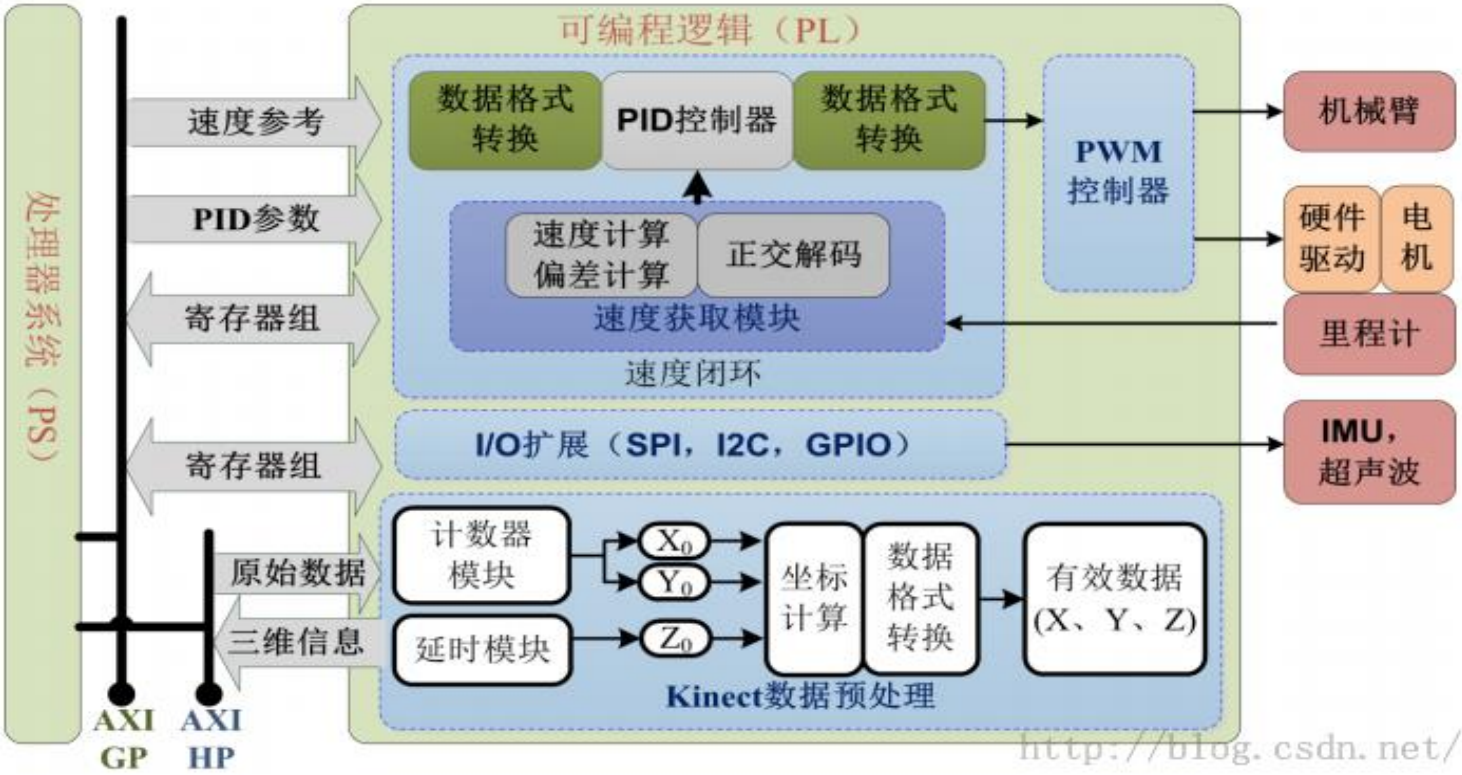


混合实时移动机器人平台
（HRMRP）



HRMRP的总体架构

2. 机器人系统构建——完整示例



HRMRP驱动框架

3. URDF机器人建模

3. URDF机器人建模

什么是URDF?

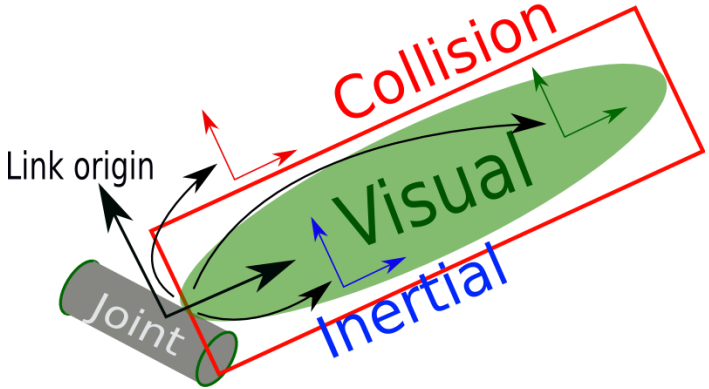
- Unified Robot Description Format，统一机器人描述格式；
- ROS中一个非常重要的机器人模型描述格式；
- 可以解析URDF文件中使用XML格式描述的机器人模型；
- ROS同时也提供URDF文件的C++解析器。

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

    <link name="base_link">
        <visual>
            <origin xyz=" 0 0 0" rpy="0 0 0" />
            <geometry>
                <cylinder length="0.005" radius="0.13"/>
            </geometry>
            <material name="yellow">
                <color rgba="1 0.4 0 1"/>
            </material>
        </visual>
    </link>

    <joint name="base_left_motor_joint" type="fixed">
        <origin xyz="-0.055 0.075 0" rpy="0 0 0" />
        <parent link="base_link"/>
        <child link="left_motor" />
    </joint>

    <link name="left_motor">
        <visual>
            <origin xyz="0 0 0" rpy="1.5707 0 0" />
            <geometry>
                <cylinder radius="0.02" length = "0.08"/>
            </geometry>
            <material name="gray">
                <color rgba="0.75 0.75 0.75 1"/>
            </material>
        </visual>
    </link>

    <joint name="left_wheel_joint" type="continuous">
        <origin xyz="0 0.0485 0" rpy="0 0 0"/>
        <parent link="left_motor"/>
        <child link="left_wheel_link"/>
        <axis xyz="0 1 0"/>
    </joint>
```

3. URDF机器人建模

<link>

- 描述机器人某个刚体部分的外观和物理属性；
- 尺寸（size）、颜色（color），形状（shape），惯性矩阵（inertial matrix），碰撞参数（collision properties）等。

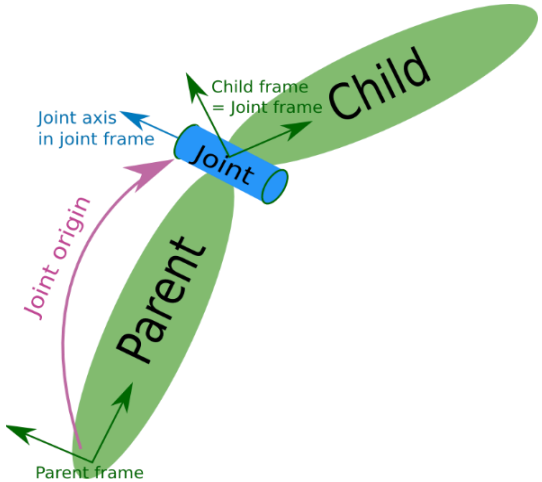


- **<visual>**：描述机器人link部分的外观参数；
- **<inertial>**：描述link的惯性参数；
- **<collision>**：描述link的碰撞属性。

```
<link name="<link name>">
    <inertial> . . . . . </inertial>
    <visual> . . . . . </visual>
    <collision> . . . . . </collision>
</link>
```

3. URDF机器人建模

```
<joint name="<name of the joint>" type = "<joint type>">
    <parent link="parent_link"/>
    <child link="child_link"/>
    <calibration .... />
    <dynamics damping ..../>
    <limit effort .... />
    ....
</joint>
```



<joint>

- 描述机器人关节的运动学和动力学属性；
- 包括关节运动的位置和速度限制；
- 根据关节运动形式，可以将其分为六种类型。

关节类型	描述
continuous	旋转关节，可以围绕单轴无限旋转
revolute	旋转关节，类似于continuous，但是有旋转的角度极限
prismatic	滑动关节，沿某一轴线移动的关节，带有位置极限
planar	平面关节，允许在平面正交方向上平移或者旋转
floating	浮动关节，允许进行平移、旋转运动
fixed	固定关节，不允许运动的特殊关节

3. URDF机器人建模

```
<joint name="<name of the joint>" type = "<joint type>">
    <parent link="parent_link"/>
    <child link="child_link"/>
    <calibration .... />
    <dynamics damping ..../>
    <limit effort .... />
    ....
</joint>
```

- **<calibration>**：关节的参考位置，用来校准关节的绝对位置；
- **<dynamics>**：描述关节的物理属性，例如阻尼值、物理静摩擦力等，经常在动力学仿真中用到；
- **<limit>**：描述运动的一些极限值，包括关节运动的上下限位置、速度限制、力矩限制等；
- **<mimic>**：描述该关节与已有关节的关系；
- **<safety_controller>**：描述安全控制器参数。

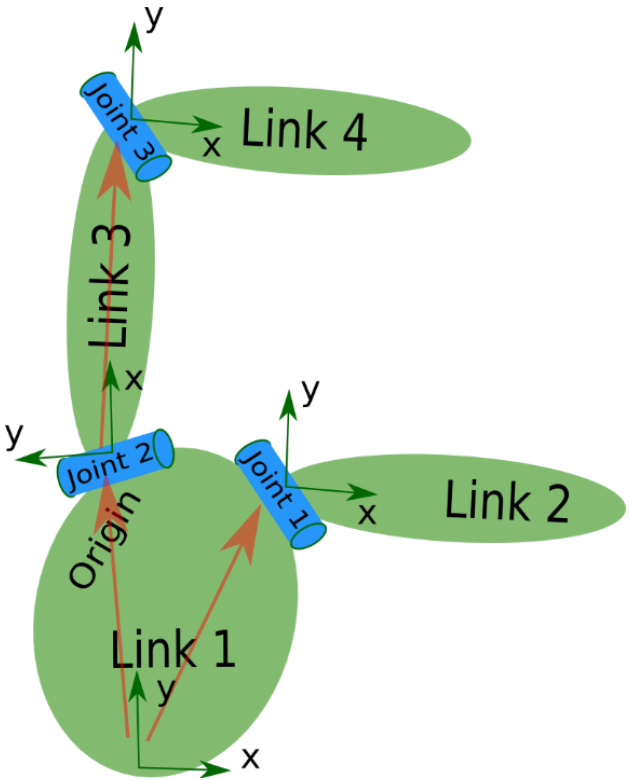
3. URDF机器人建模

<robot>

- 完整机器人模型的最顶层标签
- <link>和<joint>标签都必须包含在<robot>标签内

```
<robot name="<name of the robot>">
    <link> ..... </link>
    <link> ..... </link>

    <joint> ..... </joint>
    <joint> ..... </joint>
</robot>
```

一个完整的机器人模型，由一系列<link>和<joint>组成

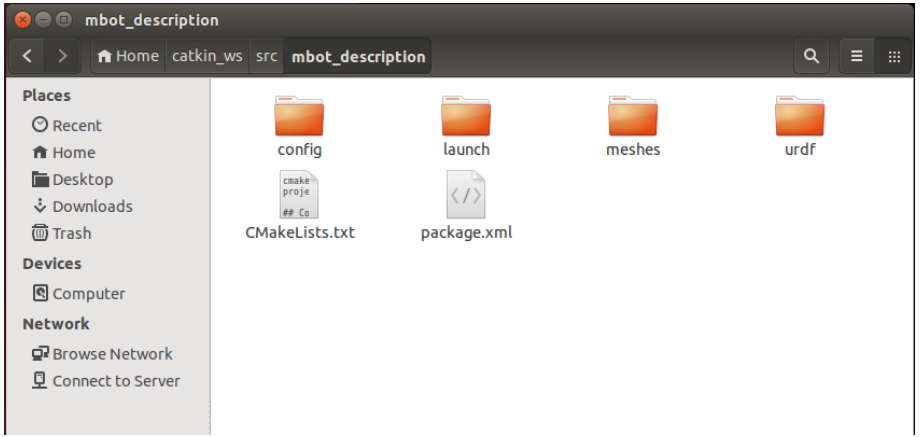


3. URDF机器人建模

创建一个机器人建模的功能包

```
$ catkin_create_pkg  mbot_description  urdf  xacro
```

- **urdf**：存放机器人模型的URDF或xacro文件
- **meshes**：放置URDF中引用的模型渲染文件
- **launch**：保存相关启动文件
- **config**：保存rviz的配置文件



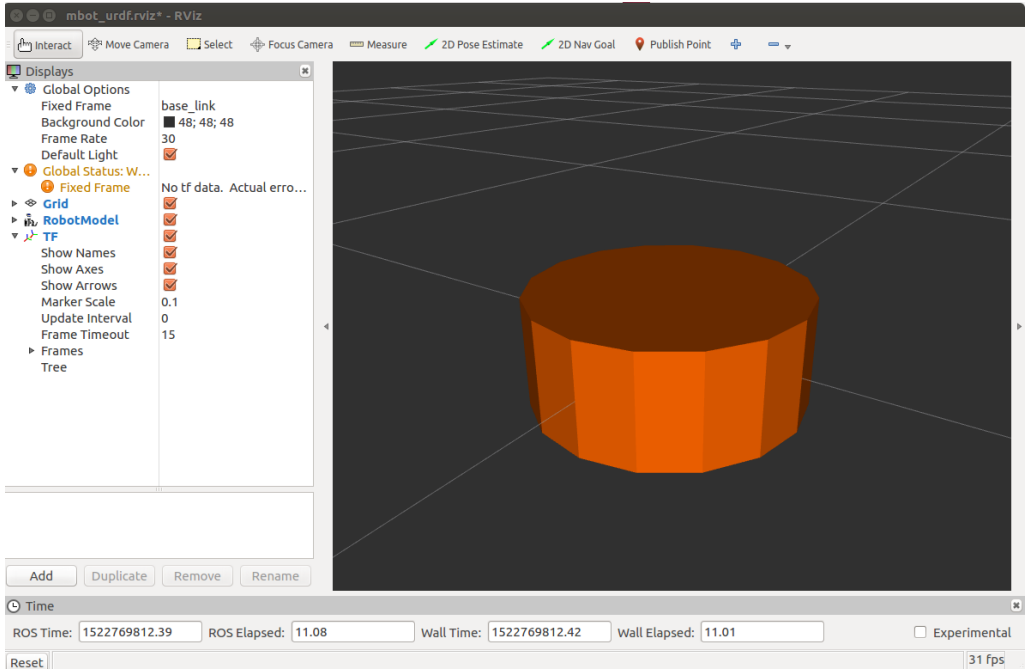
3. URDF机器人建模

```
<launch>
    <param name="robot_description" textfile="$(find mbot_description)/urdf/mbot_base.urdf" />

    <!-- 设置GUI参数，显示关节控制插件 -->
    <param name="use_gui" value="true"/>

    <!-- 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" />

    <!-- 运行robot_state_publisher节点，发布tf  -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />

    <!-- 运行rviz可视化界面 -->
    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find mbot_description)/config/mbot_urdf.rviz" required="true" />
</launch>
```

display_mbot_base_urdf.launch

- **joint_state_publisher**：发布每个joint（除fixed类型）的状态，而且可以通过UI界面对joint进行控制
- **robot_state_publisher**：将机器人各个links、joints之间的关系，通过TF的形式，整理成三维姿态信息发布

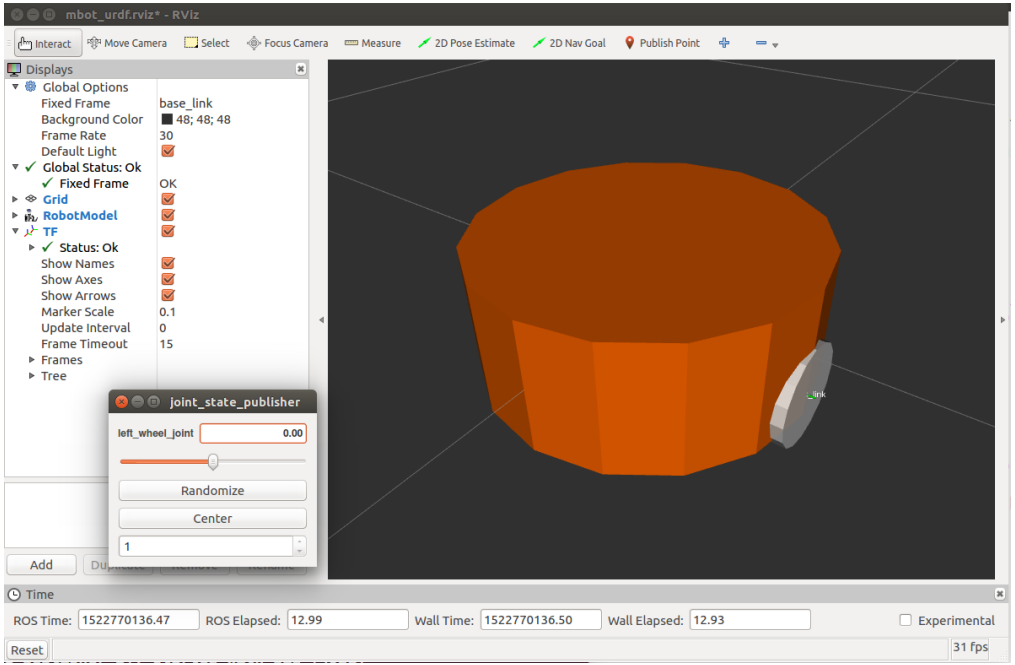
3. URDF机器人建模




```xml
<?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>

</robot>
```

第一步：使用圆柱体创建一个车体模型

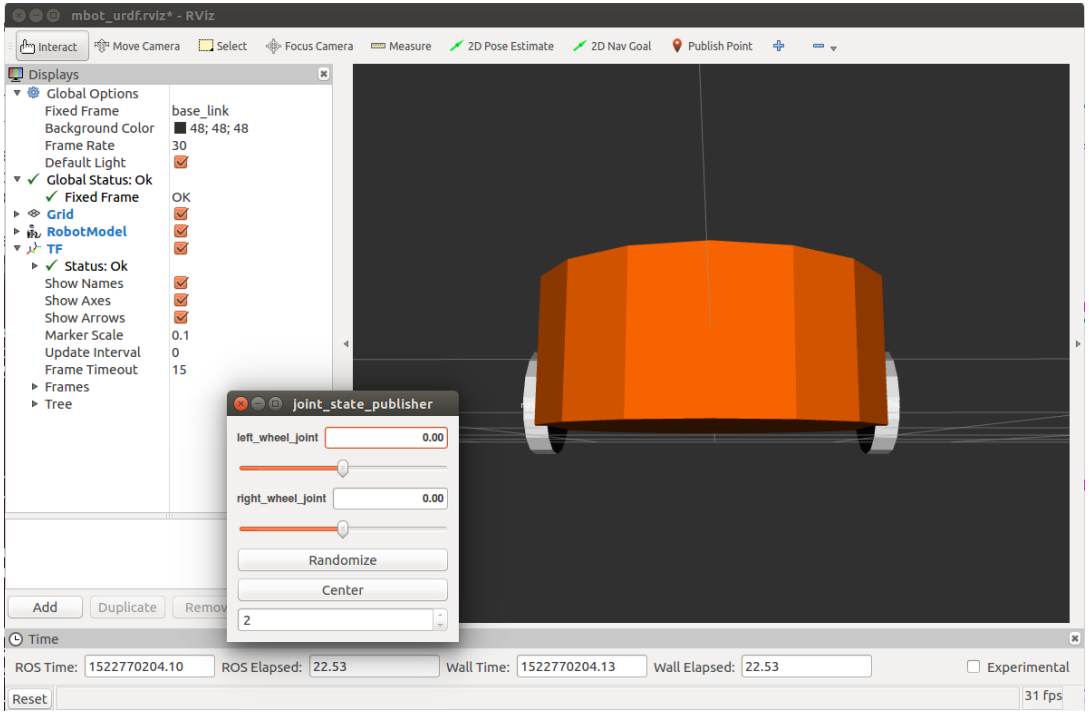
3. URDF机器人建模



第二步：使用圆柱体创建左侧车轮

```xml
<?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>
```

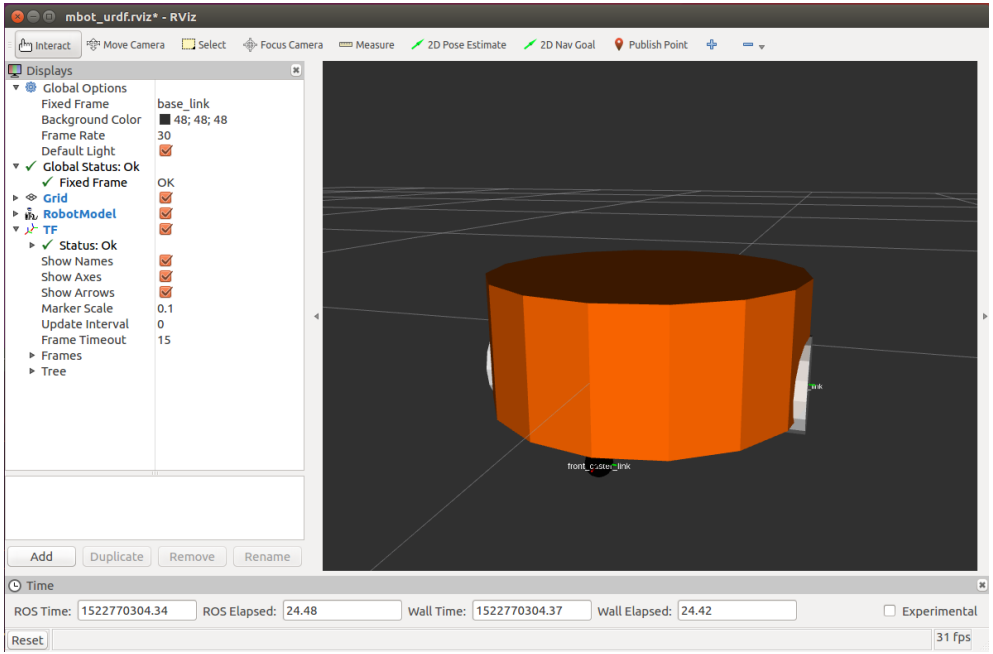
3. URDF机器人建模



第三步：使用圆柱体创建右侧车轮

```xml
<?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>

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

    <link name="right_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>
```

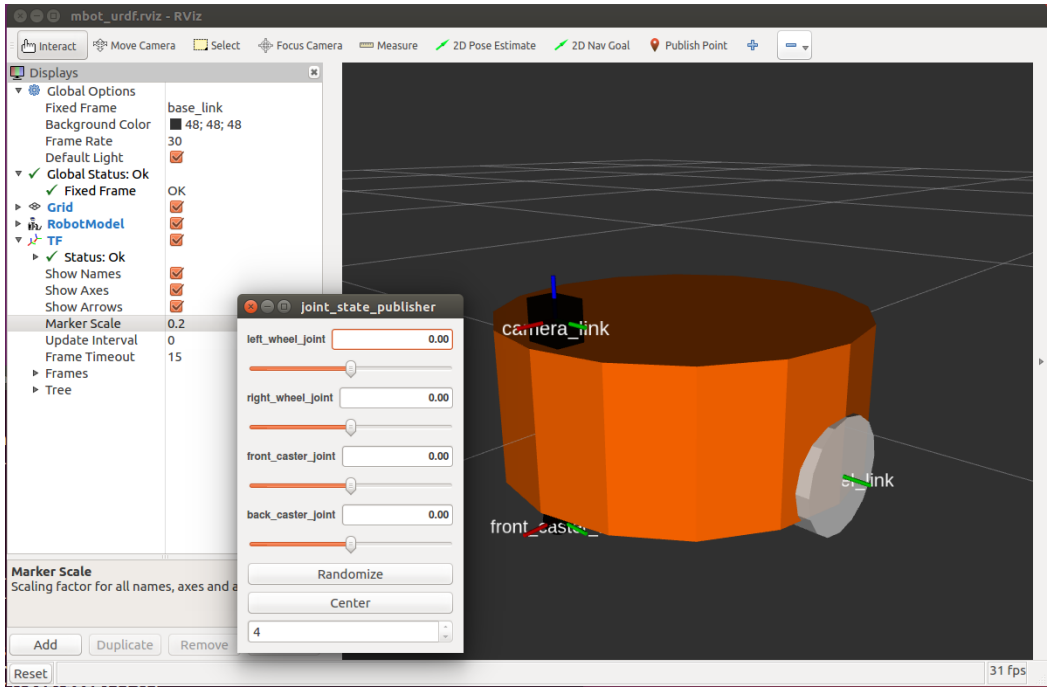
3. URDF机器人建模



第四步：使用球体创建前后支撑轮

```
<joint name="front_caster_joint" type="continuous">
    <origin xyz="0.18 0 -0.095" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="front_caster_link"/>
    <axis xyz="0 1 0"/>
</joint>

<link name="front_caster_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <sphere radius="0.015" />
        </geometry>
        <material name="black">
            <color rgba="0 0 0 0.95"/>
        </material>
    </visual>
</link>

<joint name="back_caster_joint" type="continuous">
    <origin xyz="-0.18 0 -0.095" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="back_caster_link"/>
    <axis xyz="0 1 0"/>
</joint>

<link name="back_caster_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <sphere radius="0.015" />
        </geometry>
        <material name="black">
            <color rgba="0 0 0 0.95"/>
        </material>
    </visual>
</link>
```

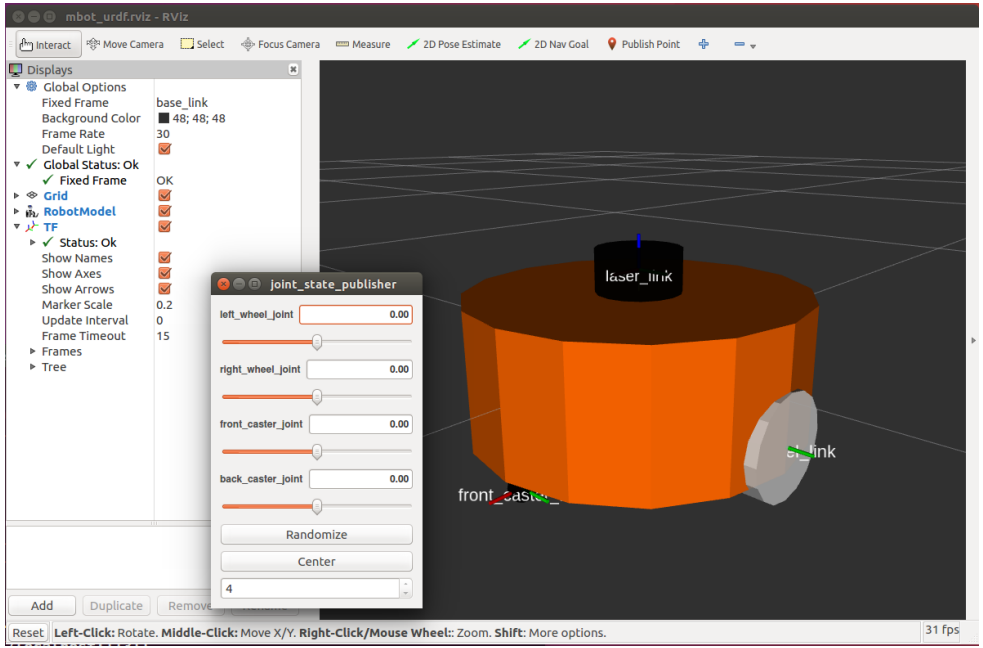
3. URDF机器人建模




```
<link name="camera_link">
    <visual>
        <origin xyz=" 0 0 0 " rpy="0 0 0" />
        <geometry>
            <box size="0.03 0.04 0.04" />
        </geometry>
        <material name="black">
            <color rgba="0 0 0 0.95"/>
        </material>
    </visual>
</link>

<joint name="camera_joint" type="fixed">
    <origin xyz="0.17 0 0.10" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="camera_link"/>
</joint>
```

第五步：创建传感器——摄像头

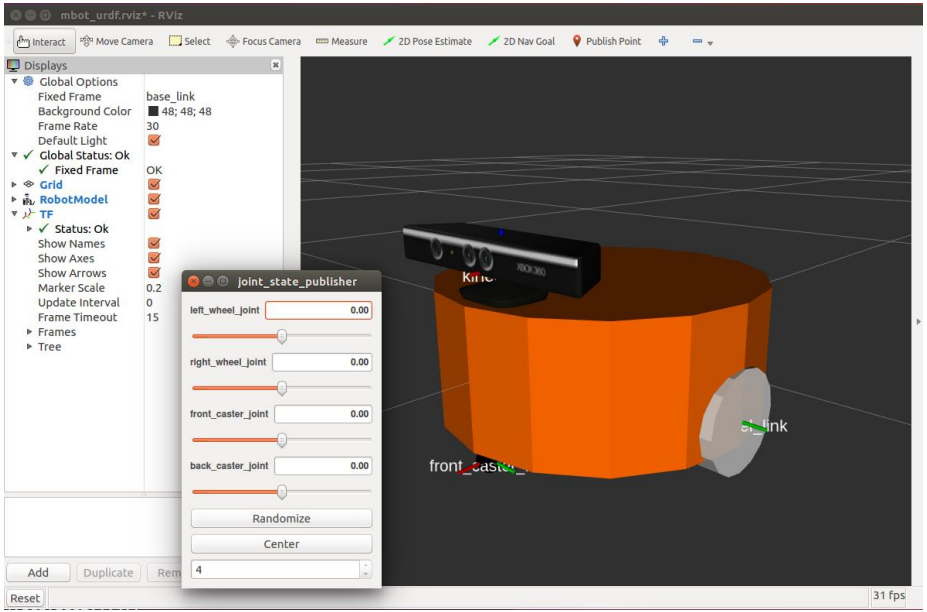
3. URDF机器人建模




```
<link name="laser_link">
    <visual>
        <origin xyz=" 0 0 0 " rpy="0 0 0" />
        <geometry>
            <cylinder length="0.05" radius="0.05"/>
        </geometry>
        <material name="black"/>
    </visual>
</link>

<joint name="laser_joint" type="fixed">
    <origin xyz="0 0 0.105" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="laser_link"/>
</joint>
```

第五步：创建传感器——激光雷达

3. URDF机器人建模

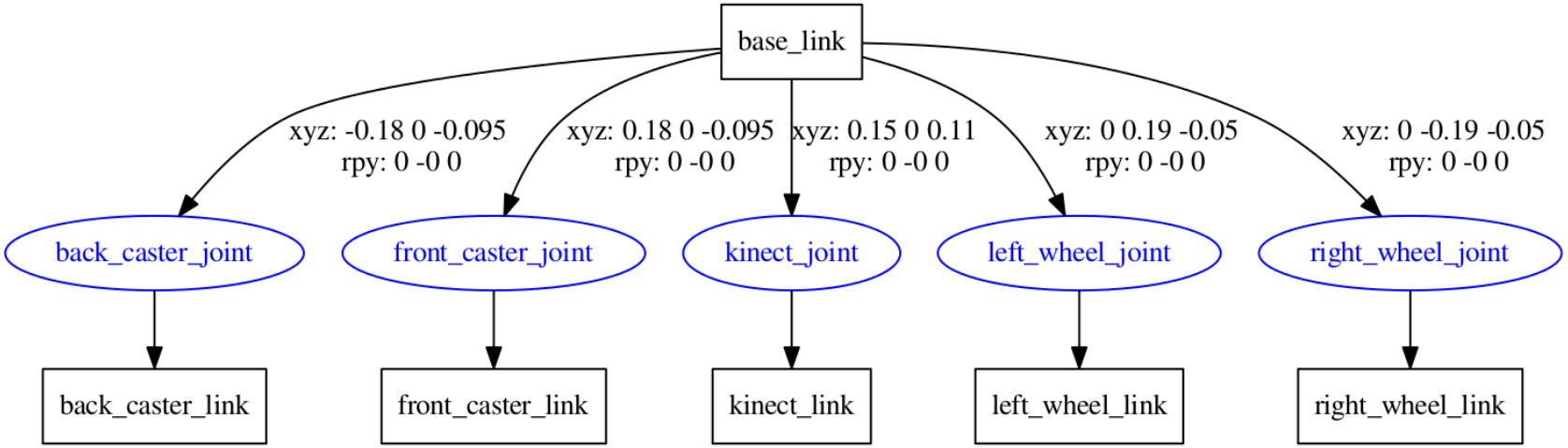



```
<link name="kinect_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 1.5708"/>
        <geometry>
            <mesh filename="package://mbot_description/meshes/kinect.dae" />
        </geometry>
    </visual>
</link>

<joint name="laser_joint" type="fixed">
    <origin xyz="0.15 0 0.11" rpy="0 0 0"/>
    <parent link="base_link"/>
    <child link="kinect_link"/>
</joint>
```

第五步：创建传感器——Kinect

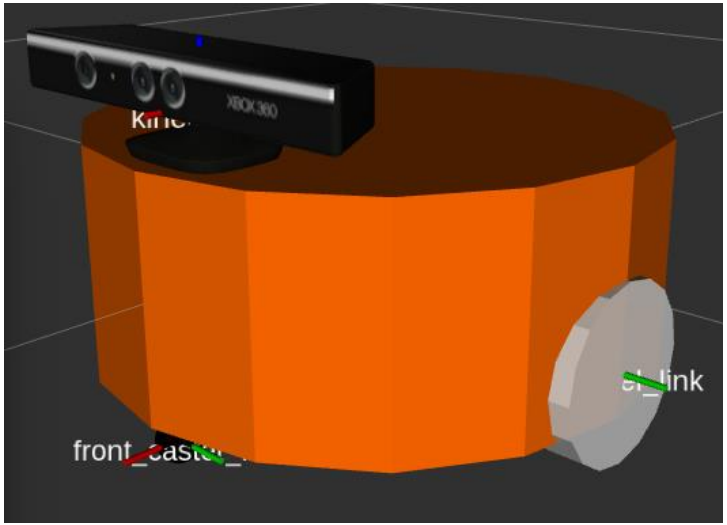
3. URDF机器人建模

第六步：检查URDF模型整体结构



$ urdf_to_graphiz mbot_base.urdf

3. URDF机器人建模



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；
- …

如何优化URDF模型

请待下回分解

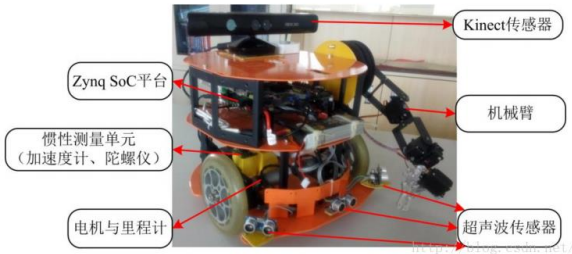
小结

机器人的定义与组成

- 定义：各国家、各组织的定义不尽相同，定义也会随着发展而变化
- 组成：**执行机构**：“手和脚”，电机、舵机等机械装置；
  驱动系统：“肌肉和经络”，电源、电机、传感器等驱动；
  传感系统：“感官和神经”，摄像头、雷达、声纳等传感器；
  控制系统：“大脑”，基于PC、ARM的控制、应用、交互等功能。

机器人系统构建

- 组装执行机构
- 搭建驱动系统
- 连接传感系统
- 实现控制系统



URDF机器人建模

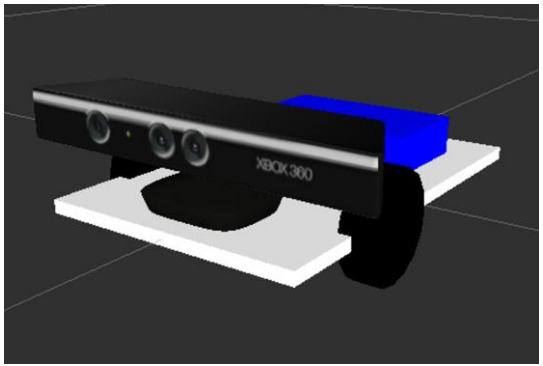
- URDF文件中的标签：<link>、<joint>、<robot>
- 创建机器人URDF模型：设计外观（link），拼装集成（joint）
- URDF模型的问题：代码冗长、不易修改、不支持计算等

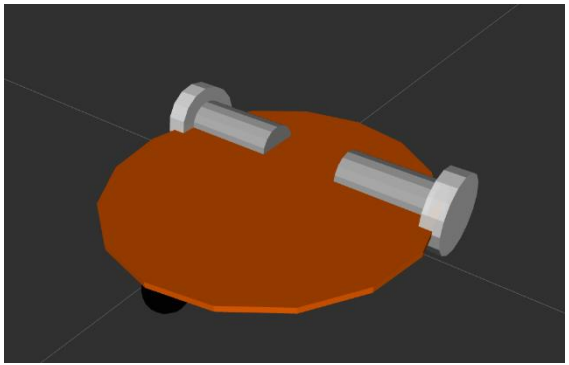
作业

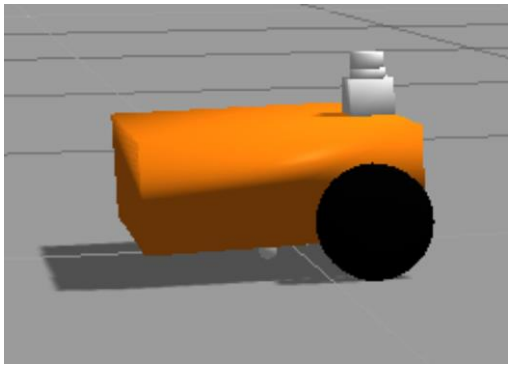
参考本讲机器人URDF模型

创建一个自己的机器人模型

（差速轮式移动机器人）







扩展阅读

- ROS探索总结（十六）——HRMRP机器人的设计
http://www.guyuehome.com/275

- ROS探索总结（五）——创建简单的机器人模型smartcar
http://www.guyuehome.com/243

- ROS URDF Tutorials
http://wiki.ros.org/urdf/Tutorials

- ARM端安装ROS
http://wiki.ros.org/indigo/Installation/UbuntuARM

感谢各位聆听！

Thanks for Listening