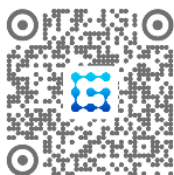


# ROS机械臂开发：从入门到实战

## —— 第3讲：如何从零创建一个机器人模型






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 自动化学院 硕士



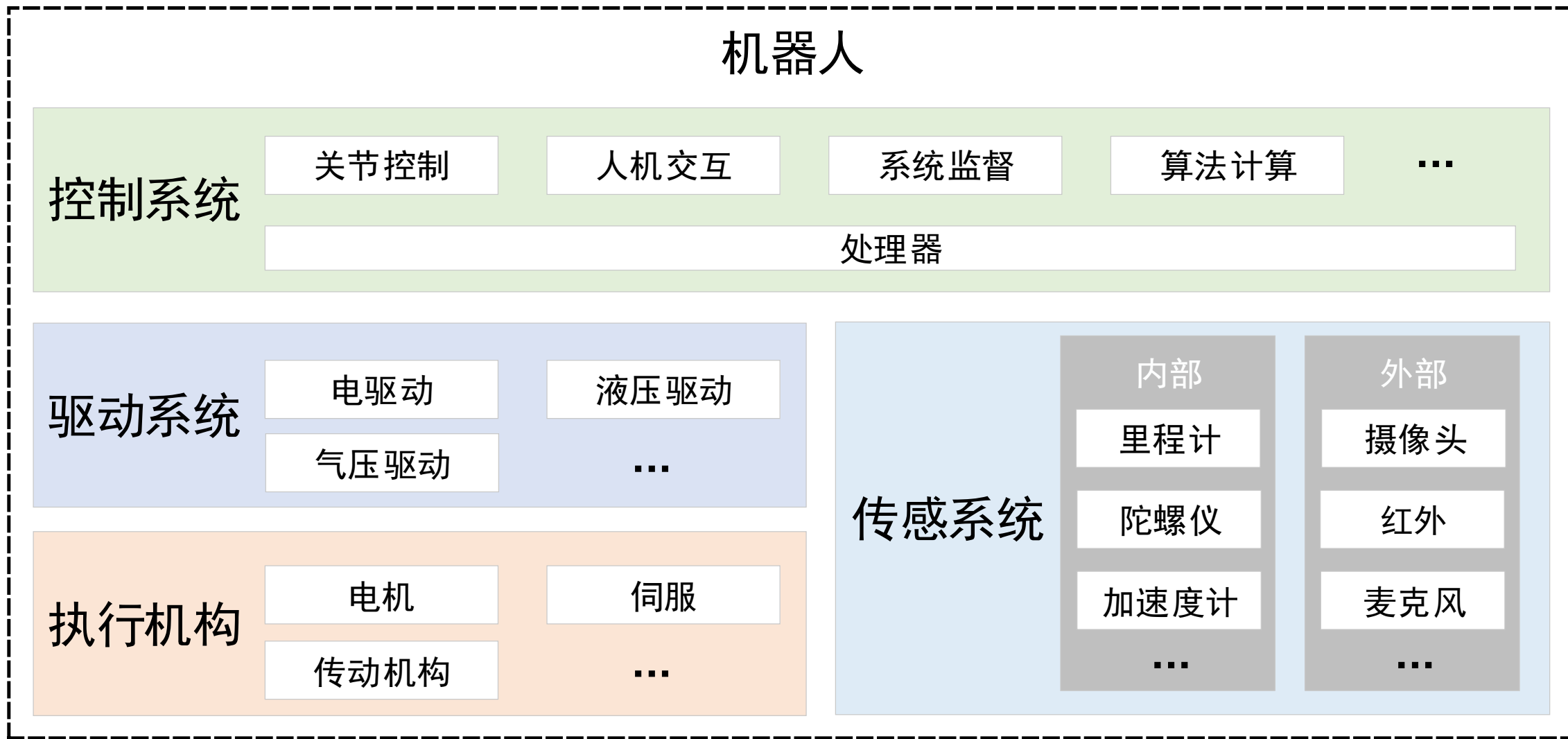
-  1. URDF建模原理
-  2. 机械臂URDF建模
-  3. 三维模型导出URDF



# 1. URDF建模原理



# 1. URDF建模原理

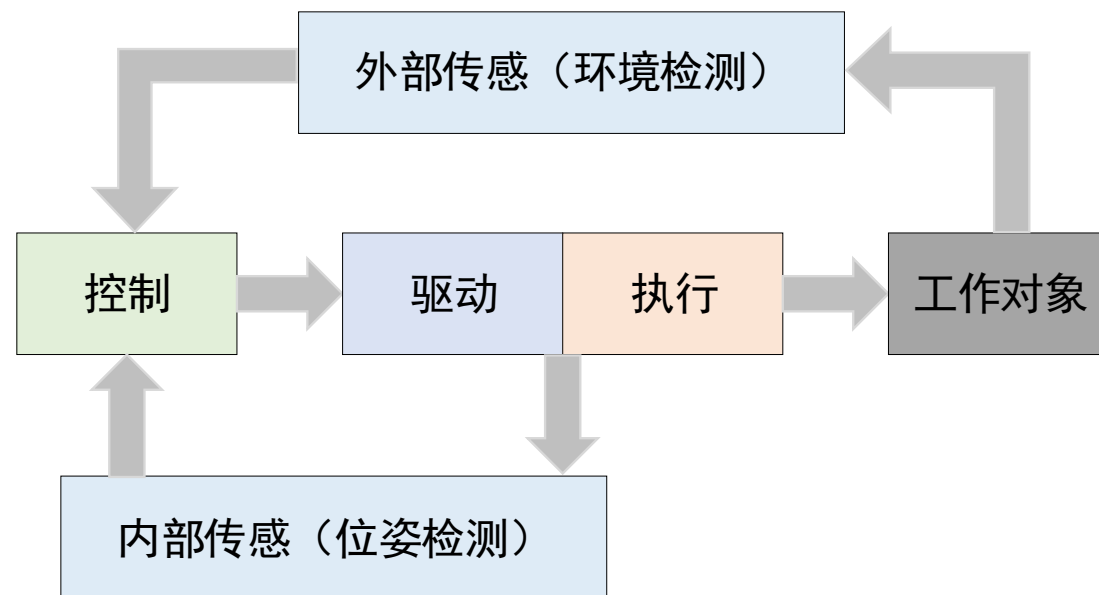


机器人的组成 (控制角度)



# 1. URDF建模原理

- **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

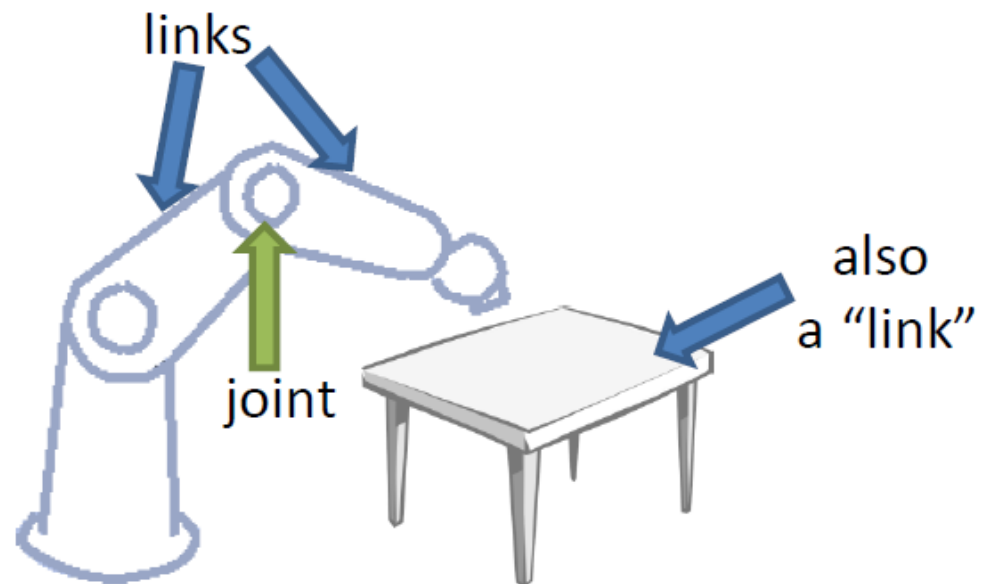


# 1. URDF建模原理

➤ URDF：一种使用XML格式描述的机器人模型文件

- Links：坐标系与几何关系
- Joints：Links之间的连接关系

➤ 类似于D-H参数



➤ 可以描述完成的工作空间，不局限于机器人

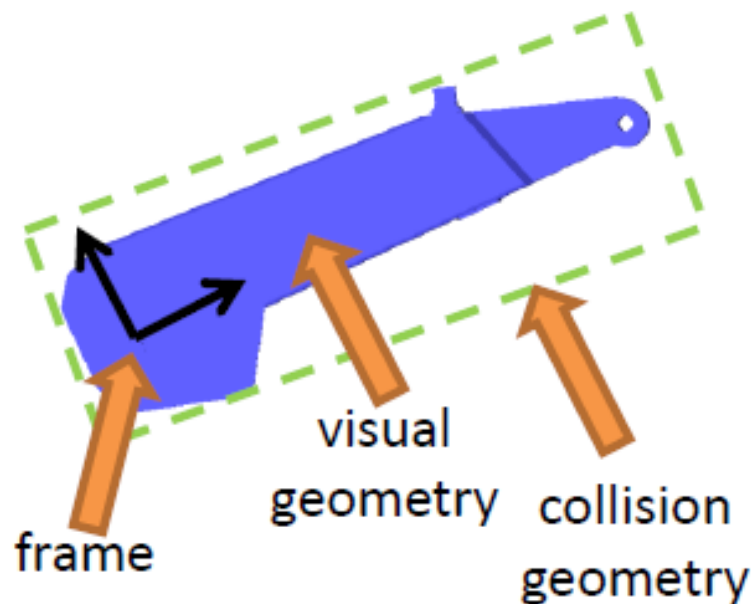


# 1. URDF建模原理

## <link>

- 描述机器人某个刚体部分的外观和物理属性;
- 描述连杆尺寸 (size)、颜色 (color)，形状 (shape)，惯性矩阵 (inertial matrix)，碰撞参数 (collision properties) 等。
- 每个Link会成为一个坐标系

```
<link name="link_4">
  <visual>
    <geometry>
      <mesh filename="link_4.stl"/>
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0" />
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.5" radius="0.1"/>
    </geometry>
    <origin xyz="0 0 -0.05" rpy="0 0 0" />
  </collision>
</link>
```



URDF Transforms

X/Y/Z Roll/Pitch/Yaw  
Meters Radians



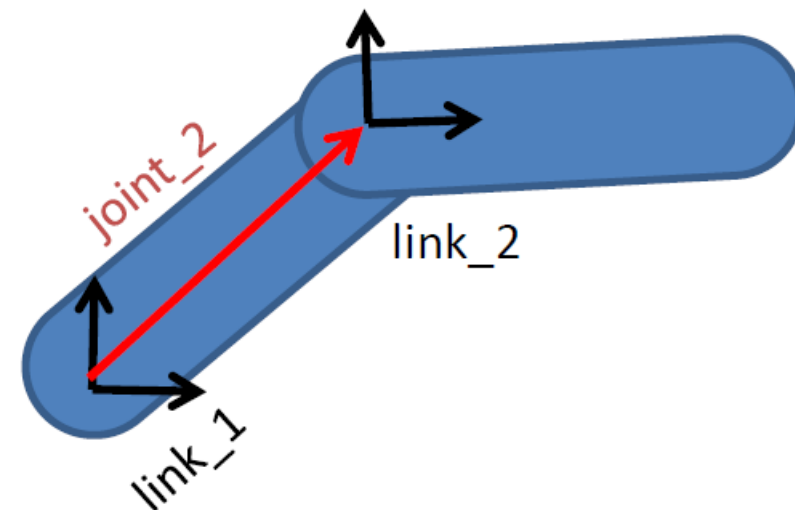
# 1. URDF建模原理

## <joint>

- 描述两个link之间的关系，分为六种类型；
- 包括关节运动的位置和速度限制；
- 描述机器人关节的运动学和动力学属性。

```
<joint name="joint_2" type="revolute">  
  <parent link="link_1"/>  
  <child link="link_2"/>  
  <origin xyz="0.2 0.2 0" rpy="0 0 0"/>  
  <axis xyz="0 0 1"/>  
  <limit lower="-3.14" upper="3.14" velocity="1.0"/>  
</joint>
```

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

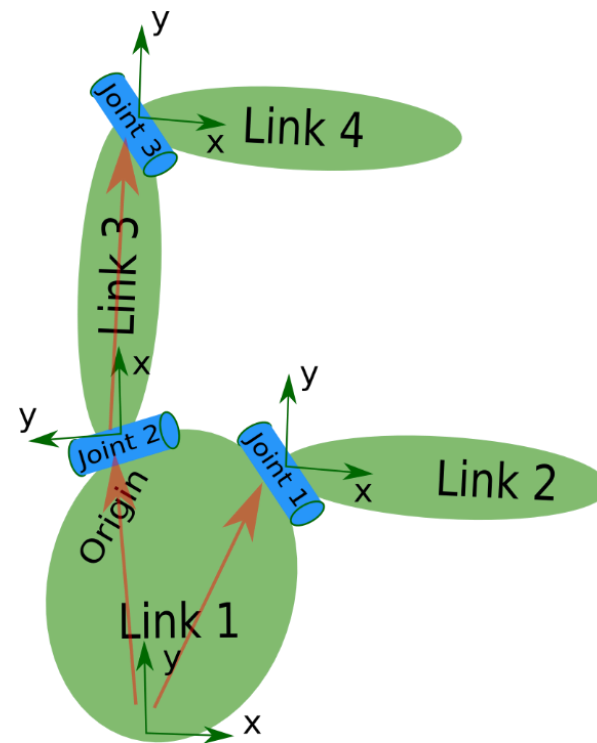




# 1. URDF建模原理

## <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
- 一个完整的机器人模型，由一系列<link>和<joint>组成



### 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；
- ...

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



## 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

```
<xacro:include filename="myRobot.xacro"/>
```

```
<xacro:myRobot prefix="left_"/>
```

```
<xacro:myRobot prefix="right_"/>
```

### ➤ 提供可编程接口

- 常量
- 变量
- 数学计算
- 条件语句

```
<property name="offset" value="1.3"/>
```

```
<joint name="world_to_left" type="fixed">
```

```
  <parent link="world"/>
```

```
  <child link="left_base_link"/>
```

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

```
</joint>
```



# 1. URDF建模原理

## 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



## 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. URDF建模原理

## 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. URDF建模原理

## 文件包含

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

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

## 文件包含



## 2.机械臂URDF建模



## 2. 机械臂URDF建模

```
<!-- Defining the colors used in this robot -->
```

```
<material name="Black">
  <color rgba="0 0 0 1"/>
</material>
<material name="White">
  <color rgba="1 1 1 1"/>
</material>
<material name="Blue">
  <color rgba="0 0 1 1"/>
</material>
<material name="Red">
  <color rgba="1 0 0 1"/>
</material>
```

颜色定义

```
<!-- Constants -->
```

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

```
<!-- link0 properties -->
```

```
<xacro:property name="link0_radius" value="0.05" />
<xacro:property name="link0_length" value="0.04" />
<xacro:property name="link0_mass" value="1" />
```

```
<!-- link1 properties -->
```

```
<xacro:property name="link1_radius" value="0.03" />
<xacro:property name="link1_length" value="0.10" />
<xacro:property name="link1_mass" value="1" />
```

```
<!-- link2 properties -->
```

```
<xacro:property name="link2_radius" value="0.03" />
<xacro:property name="link2_length" value="0.14" />
<xacro:property name="link2_mass" value="0.8" />
```

```
<!-- link3 properties -->
```

```
<xacro:property name="link3_radius" value="0.03" />
<xacro:property name="link3_length" value="0.15" />
<xacro:property name="link3_mass" value="0.8" />
```

参数定义

```
<!-- ////////////////////////////////// LINK0 ////////////////////////////////// -->
```

```
<link name="link0">
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder radius="${link0_radius}" length="${link0_length}" />
    </geometry>
    <material name="White" />
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <cylinder radius="${link0_radius}" length="${link0_length}" />
    </geometry>
  </collision>
  <cylinder_inertial_matrix m="${link0_mass}" r="${link0_radius}" h="${link0_length}" />
</link>
```

link定义

```
<joint name="joint1" type="revolute">
```

```
  <parent link="link0"/>
  <child link="link1"/>
  <origin xyz="0 0 ${link0_length/2}" rpy="0 ${M_PI/2} 0" />
  <axis xyz="-1 0 0" />
  <limit effort="300" velocity="1" lower="${-M_PI}" upper="${M_PI}" />
</joint>
```

joint定义

```
<!-- ////////////////////////////////// LINK1 ////////////////////////////////// -->
```

```
<link name="link1" >
  <visual>
    <origin xyz="-${link1_length/2} 0 0" rpy="0 ${M_PI/2} 0" />
    <geometry>
      <cylinder radius="${link1_radius}" length="${link1_length}" />
    </geometry>
    <material name="Blue" />
  </visual>
```



## 2. 机械臂URDF建模

```
<launch>
  <arg name="model" />
  <!-- 加载机器人模型参数 -->
  <param name="robot_description" command="$(find xacro)/xacro --inorder $(find marm_description)/urdf/marm.xacro" />

  <!-- 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 marm_description)/urdf.rviz" required="true" />
</launch>
```

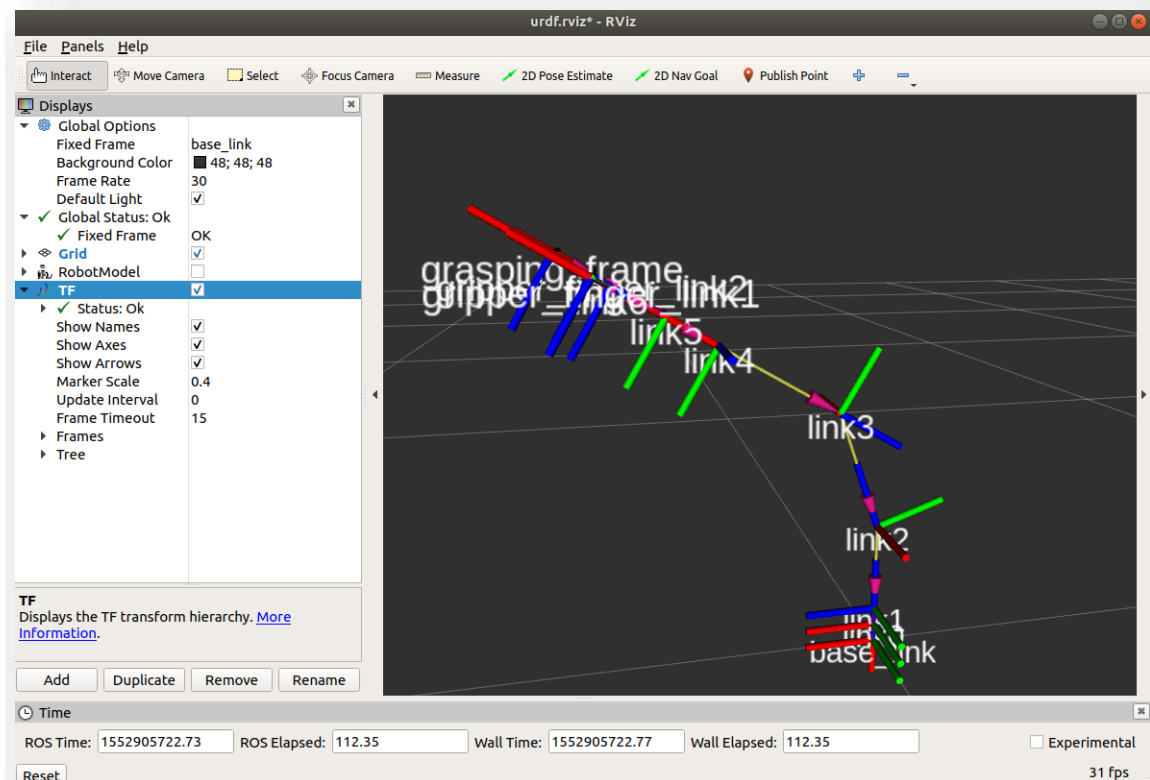
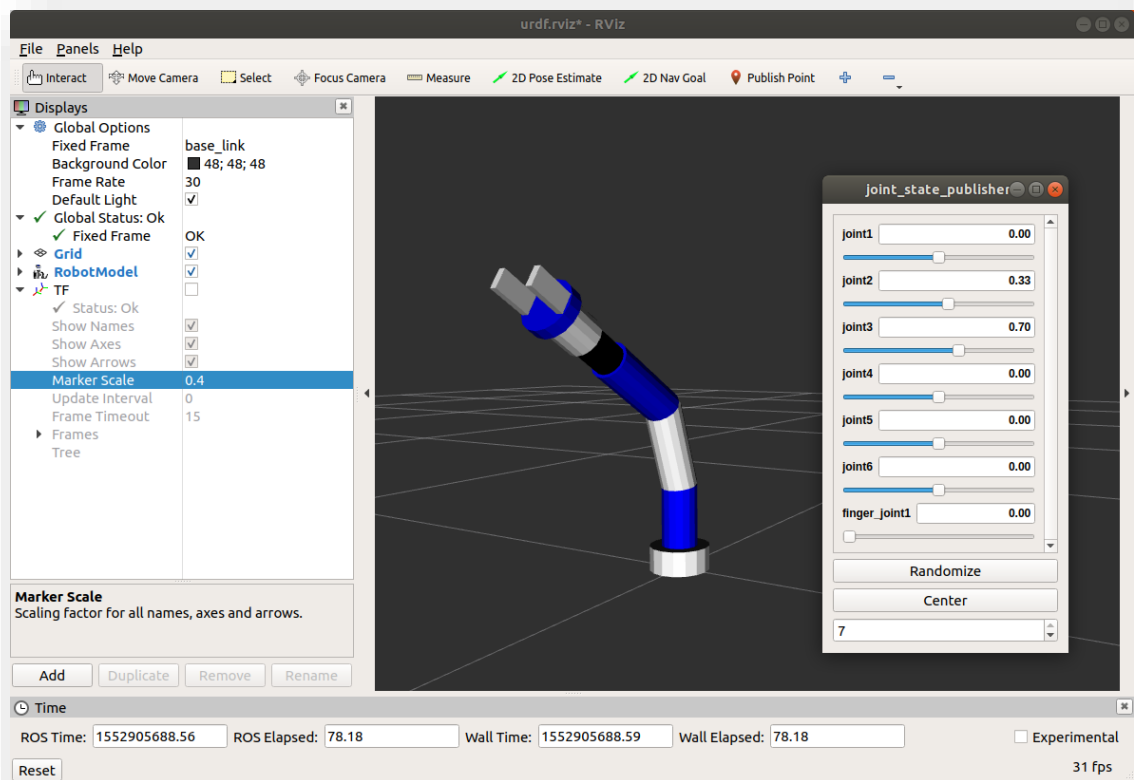
joint\_state

tf

**marm\_description/launch/view\_marm.launch**



## 2. 机械臂URDF建模



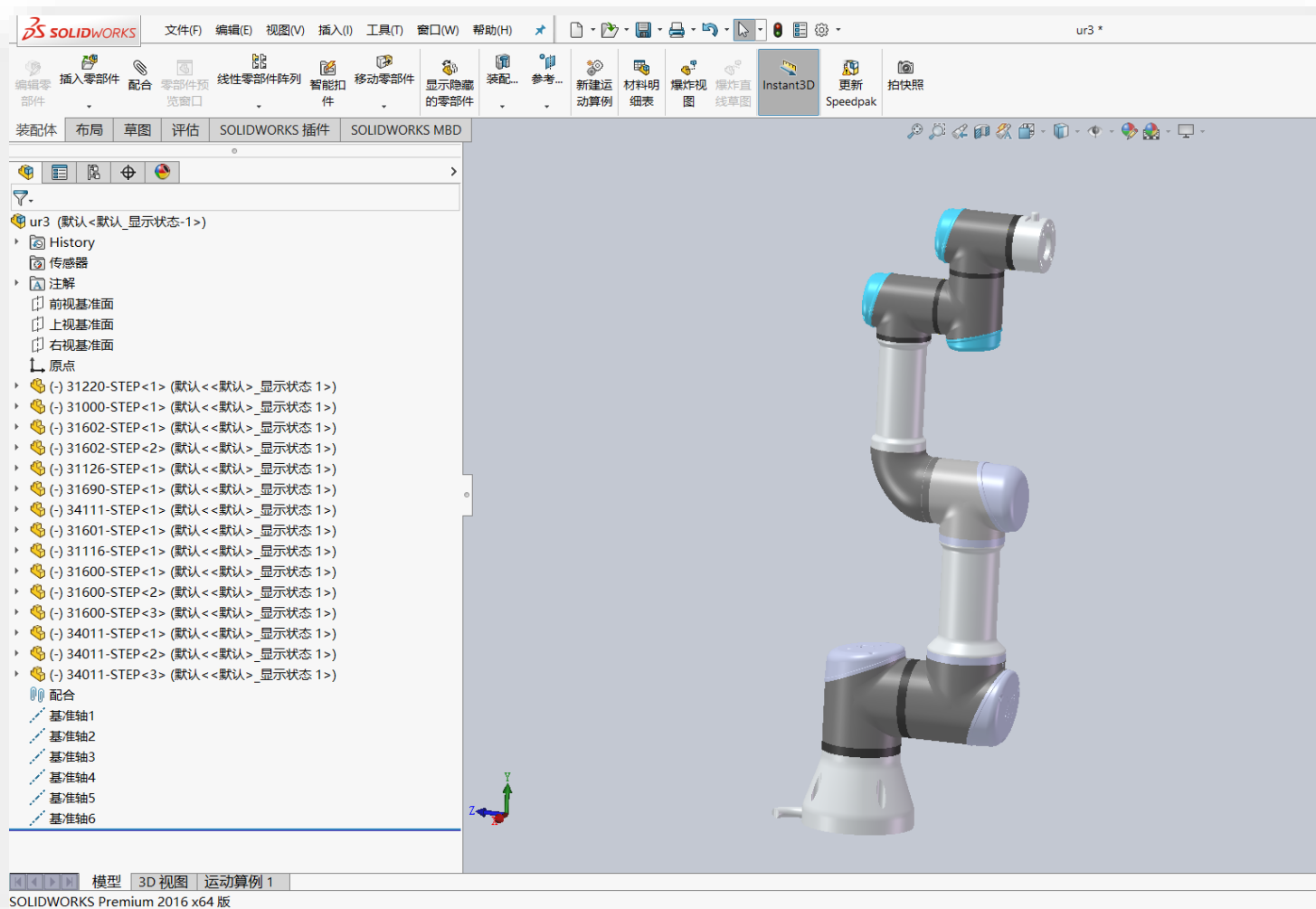
模型可视化 `$ roslaunch marm_description view_marm.launch`



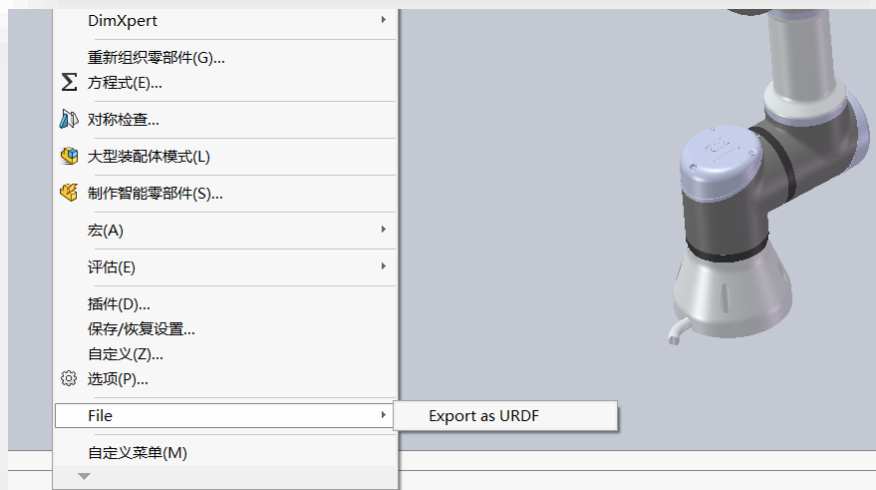
## 3. 三维模型导出URDF



# 3. 三维模型导出URDF



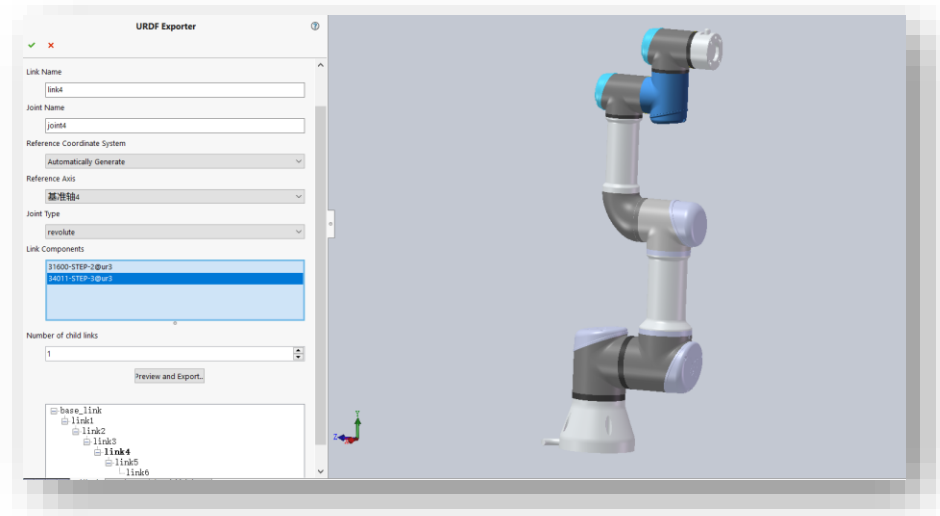
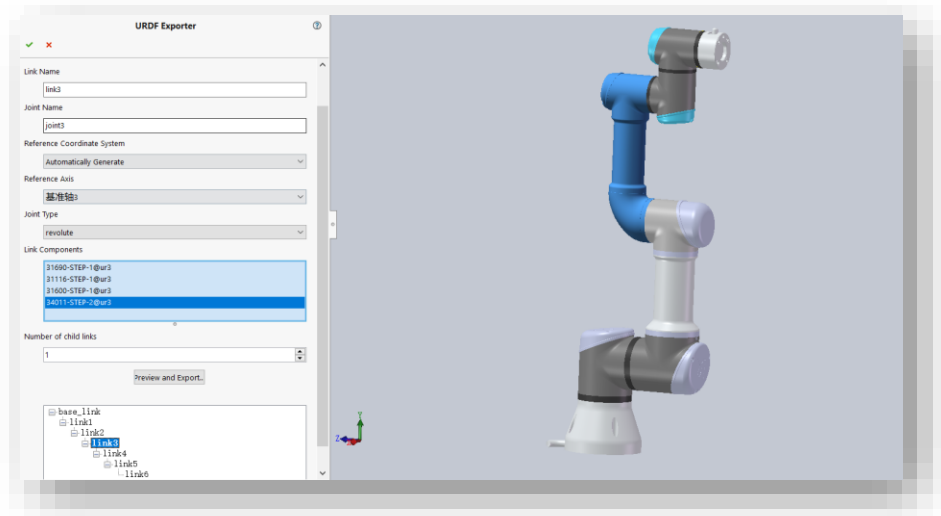
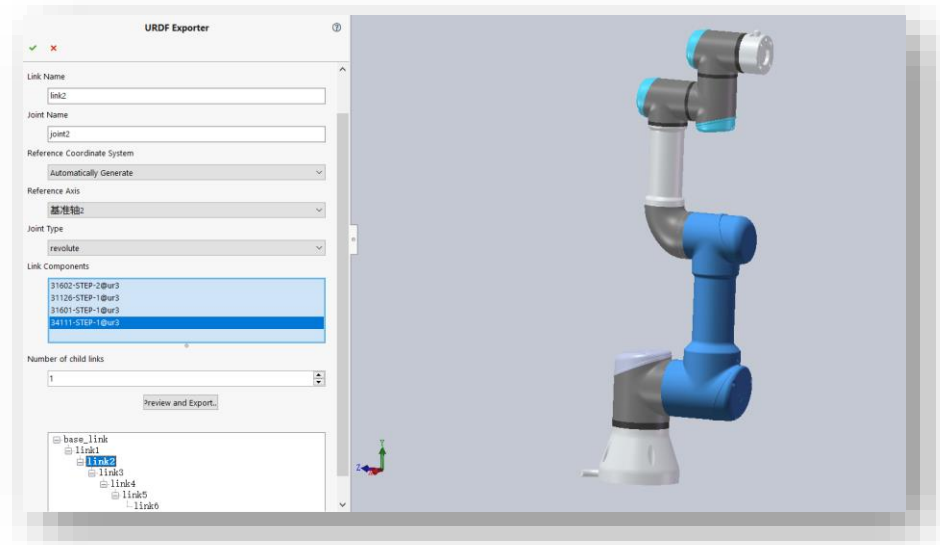
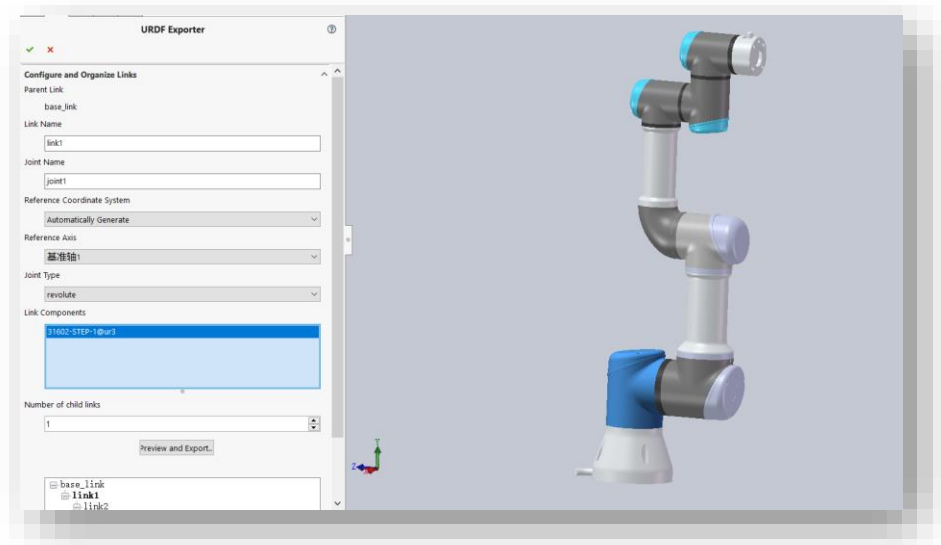
完成模型设计，安装sw2urdf插件



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



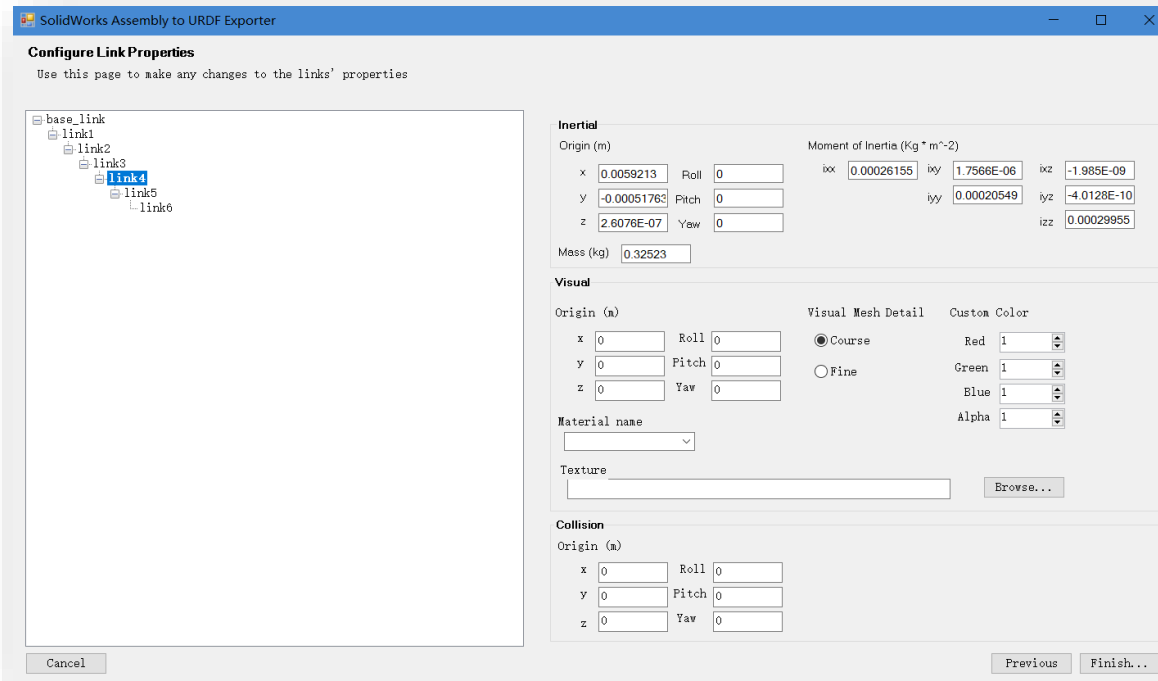
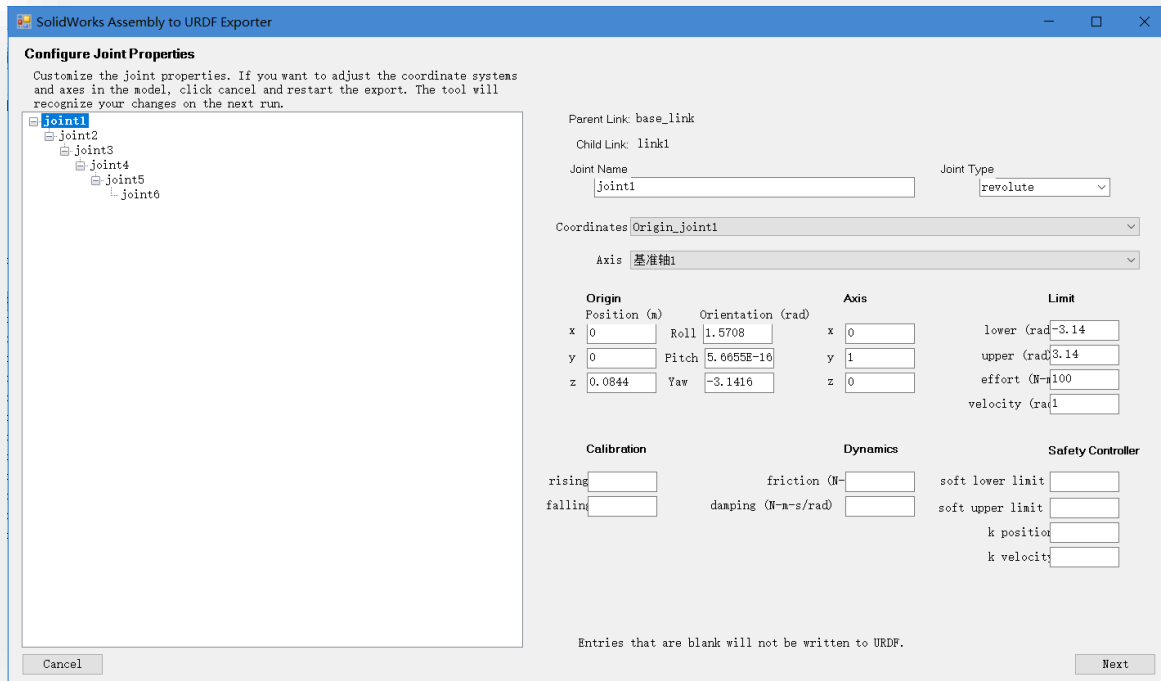
### 3. 三维模型导出URDF



根据说明，完成link和joint的配置



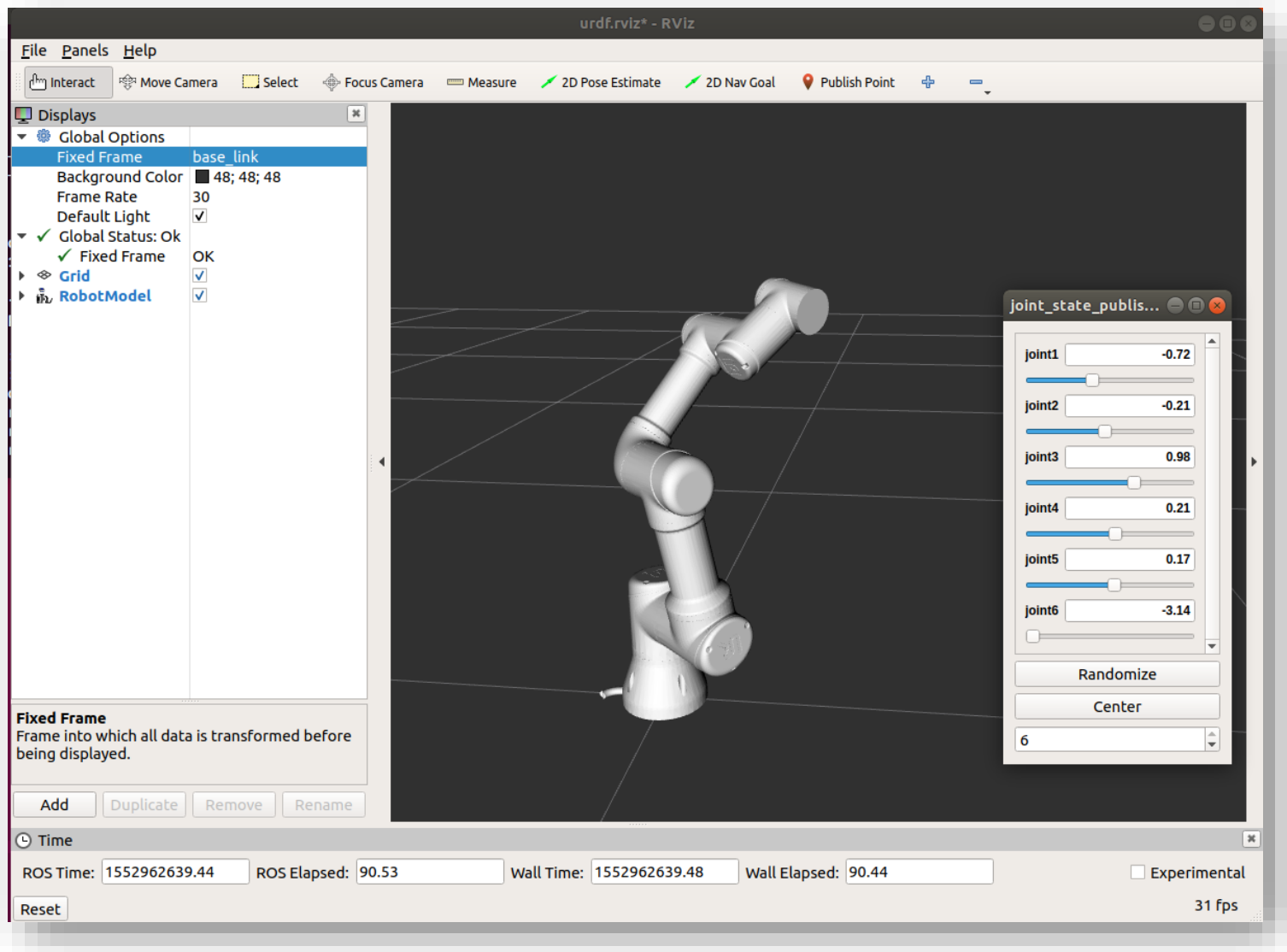
# 3. 三维模型导出URDF



检查并确认配置，生成模型功能包



# 3. 三维模型导出URDF



修改文件bugs,  
测试模型



## URDF 建模

- 机器人组成：执行机构，驱动系统，传感系统，控制系统
- 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）

## 机械臂 建模

- 六轴机器人：7 links + 6 joints
- 可视化显示：模型路径 + joint\_state + tf + rviz

## 三维模型 导出

- 完成模型设计，安装sw2urdf插件
- 根据说明，完成link和joint的配置
- 检查并确认配置，生成模型功能包
- 修改文件bugs，测试模型



1. 在marm模型的基础上，修改模型参数或结构，通过可视化效果的变化，熟悉各参数的含义；
2. 使用任意机器人模型，在Solidworks当中尝试转换成URDF模型包，并在ROS中测试模型的显示效果。

\* 扩展题：自己构想一个多自由度机械臂模型，并使用URDF语法创建出来



- ROS探索总结（二十三）——解读URDF

<http://www.guyuehome.com/372>

- URDF Tutorials:

<http://wiki.ros.org/urdf/Tutorials>

- URDF 语法规则:

<http://wiki.ros.org/urdf/XML>

- sw\_urdf\_exporter Tutorials

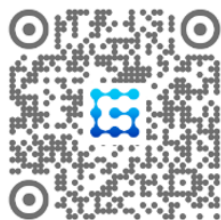
[http://wiki.ros.org/sw\\_urdf\\_exporter/Tutorials](http://wiki.ros.org/sw_urdf_exporter/Tutorials)





# Thank You

更多精彩，欢迎关注



 古月居



 古月春旭