25.About ROS2URDF model

1. URDF introduction

The modeling method in ROS is called URDF, which stands for Unified Robot Description Format. It can not only clearly describe the robot's own model, but also describe the robot's external environment. URDF model files use XML format.

2. Composition of robot

In the process of modeling and describing the robot, we need to first familiarize ourselves with the composition and parameters of the robot. For example, a robot is generally composed of four major parts: hardware structure, drive system, sensor system, and control system. Some common robots on the market , whether it is a mobile robot or a robotic arm, we can decompose it according to these four major components.

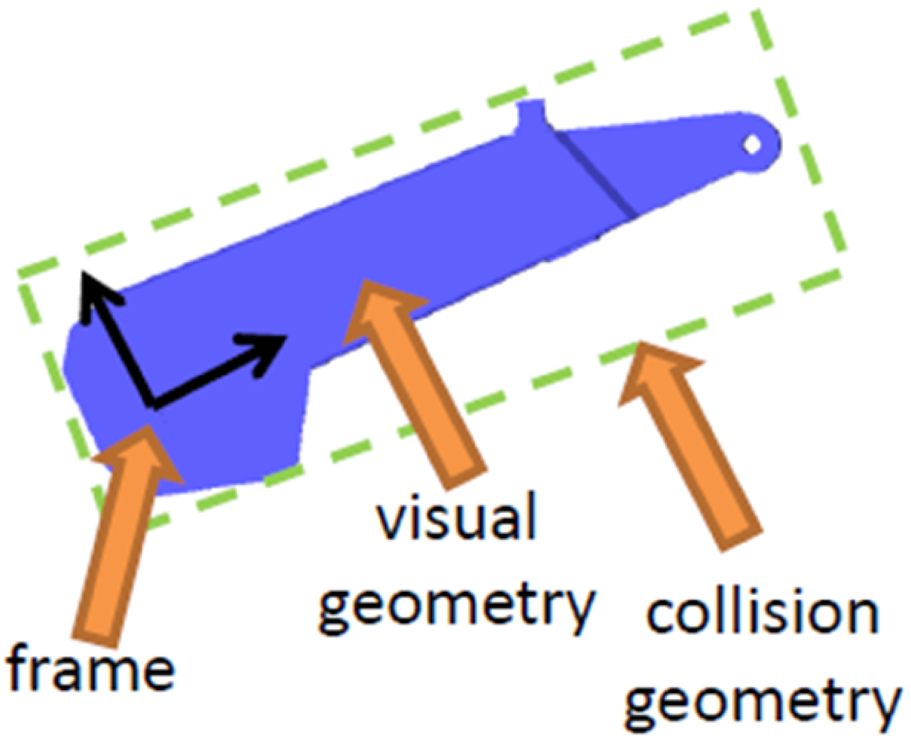
- The hardware structure is the chassis, casing, motor and other equipment that can be seen in real life;
- The drive system is the device that can drive these devices to use normally, such as motor drivers, power management systems, etc.;
- The sensing system includes the encoder on the motor, the onboard IMU, the installed camera, the radar, etc. to facilitate the robot to sense its own status and the external environment;
- The control system is the main carrier of our development process, usually a computing platform such as a Raspberry Pi or a computer, as well as the operating system and application software inside.

The process of robot modeling is actually a process of describing each part of the robot clearly through modeling language and then putting it together according to a similar idea.

3. URDF syntax

3.1. Description of connecting rod Link

The tag is used to describe the appearance and physical attributes of a rigid body part of the robot. The appearance includes size, color, and shape, and the physical attributes include mass, inertia matrix, collision parameters, etc.



Take this robot arm link as an example. Its link is described as follows:

```
<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>
```

The name in the link tag represents the name of the link. We can customize it. This name will be used when the joint connects to the link in the future.

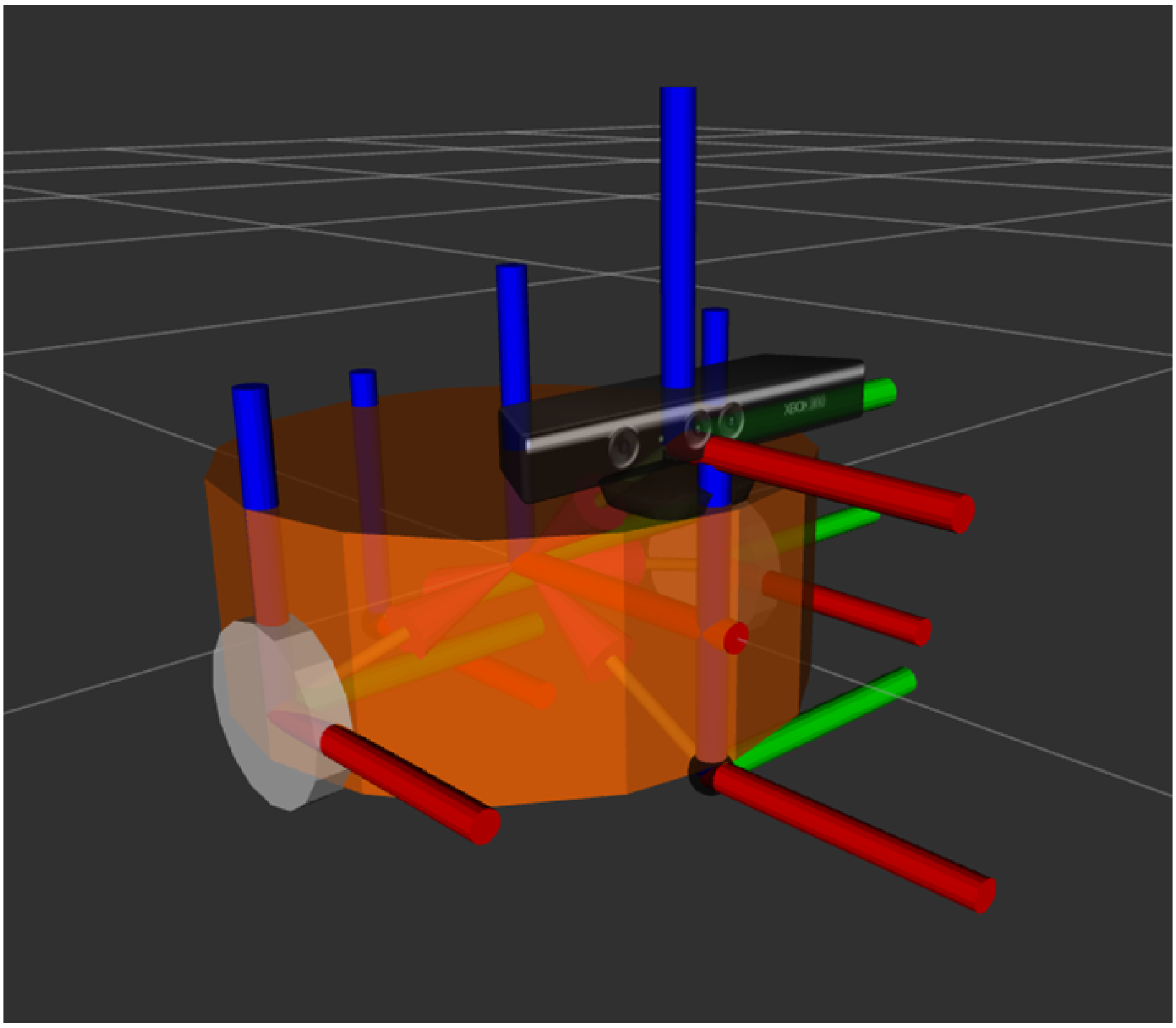
The part in the link is used to describe the appearance of the robot, such as:

- represents the geometric shape, and is used inside to call a blue appearance designed in advance in the 3D software. This is this stl file, which looks consistent with the real robot.
- represents the offset of the coordinate system relative to the initial position, which are translation in the x, y, and z directions, and roll, pitch, and raw rotation. If no offset is needed, all values will be 0.

The second part, , describes the collision parameters. The content inside seems to be the same as . There are also and , which seem to be the same, but in fact the difference is quite big.

- The part focuses on describing the appearance of the robot, that is, the visual effect;
- The part describes the state of the robot during its movement, such as how the robot contacts the outside world and counts as a collision.

In this robot model, the blue part is described by . In the actual control process, such a complex appearance requires high computing power when calculating collision detection.In order to simplify the calculation, we simplify the model used for collision detection to a cylinder with a green box, which is the shape described by in . The coordinate system offset is similar and can describe the offset of the center of mass of the rigid body.

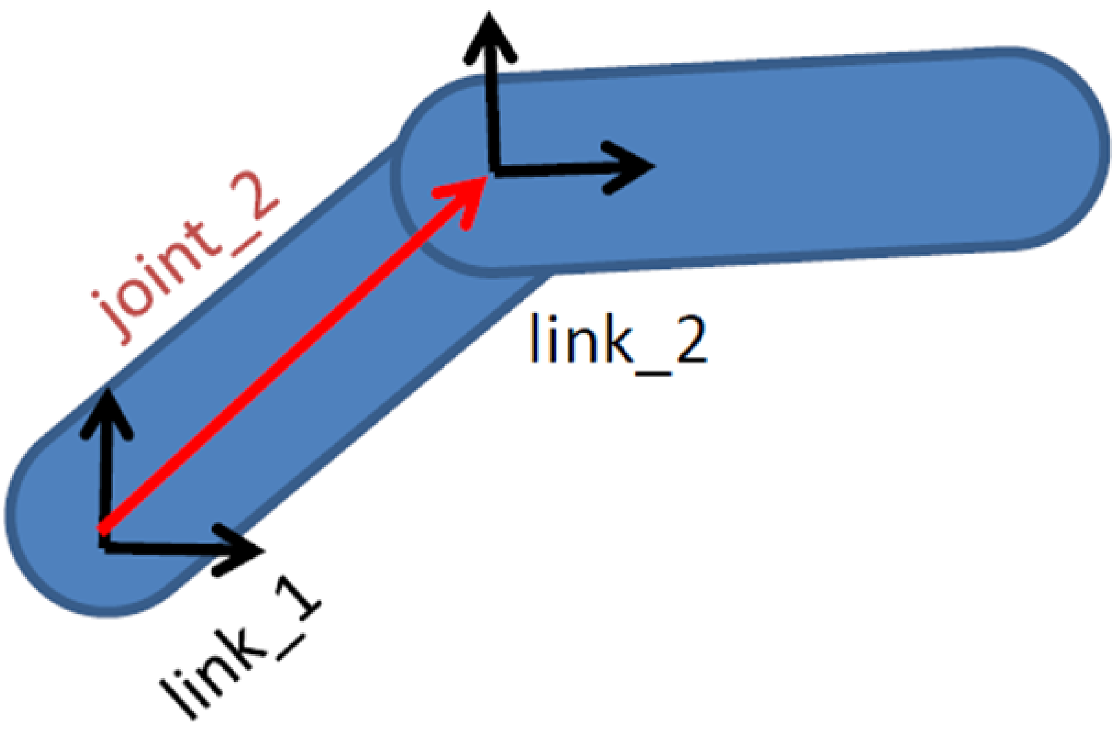


If it is a mobile robot, link can also be used to describe the body, wheels and other parts of the car.

3.2. Joint description

The rigid bodies in the robot model must finally be connected through joints before relative motion can occur.

1. Continuous, describes rotational motion, which can rotate infinitely around a certain axis. For example, the wheels of a car fall into this type.
2. Revolute is also a rotating joint. The difference from the continuous type is that it cannot rotate infinitely, but has angle restrictions. For example, the two links of a mechanical arm fall into this kind of motion.
3. Prismatic is a sliding joint that can translate along a certain axis and also has a position limit. Generally, linear motors use this method of movement.
4. Fixed joint is the only joint that does not allow movement, but it is still used frequently.For example, when the camera link is installed on the robot, its relative position will not change. The connection method used at this time is Fixed.
5. Floating is a floating joint, and the sixth type of planar is a plane joint. These two are relatively rarely used.



In the URDF model, each link is described using such an XML content, such as the name of the joint and the type of motion.

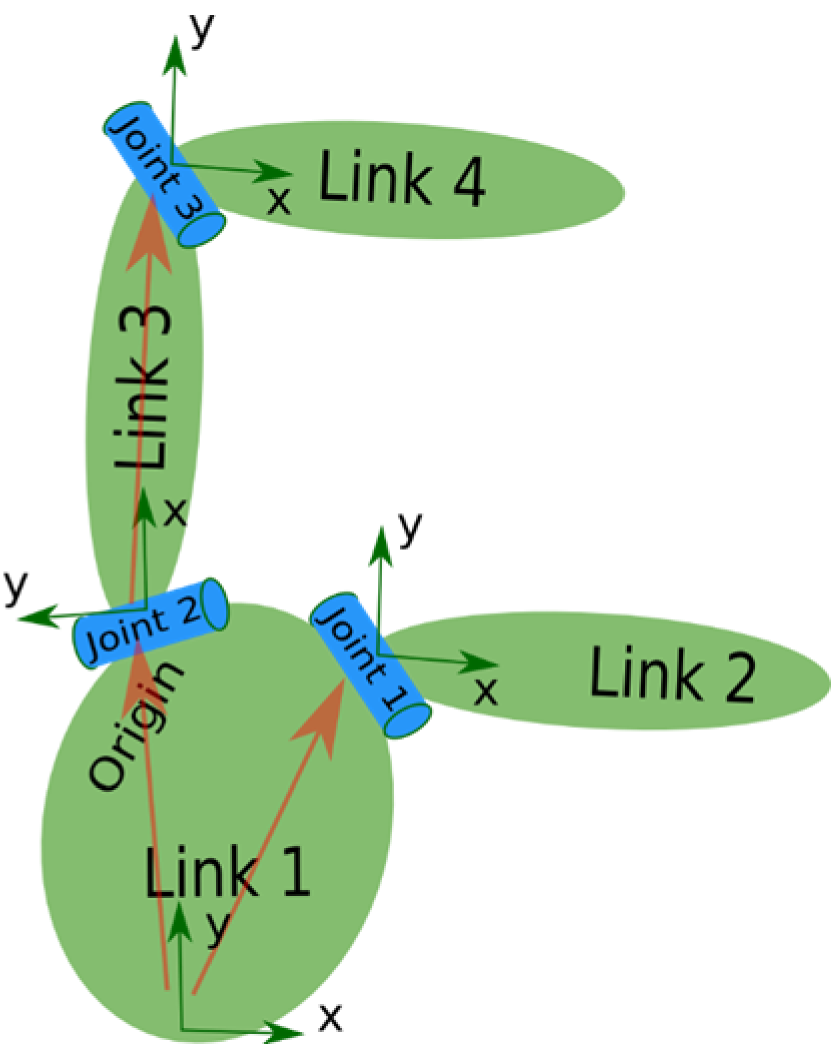
```
<joint name="joint_2" type="revolute">
    <parent link="link_1"/>
    <child link="link_2"/>
    <origin xyz="0.2 0.2 0" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-3.14" upper="3.14" velocity="1.0"/>
</joint>
```

- parent tag: describes the parent connecting rod;
- child tag: describes the child link, which will move relative to the parent link;
- origin: represents the relationship between the two connecting rod coordinate systems, which is the red vector in the picture, which can be understood as how the two connecting

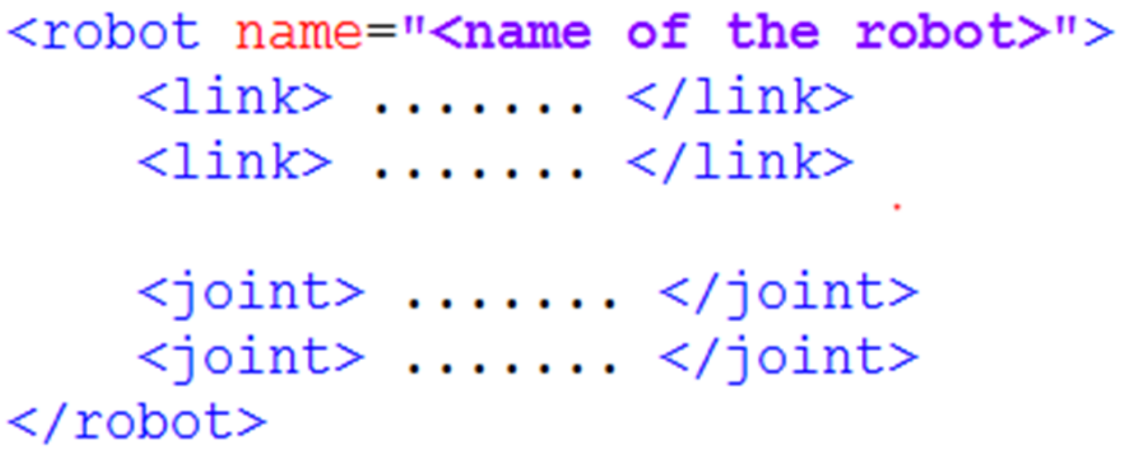
rods should be installed together;

- axis represents the unit vector of the joint motion axis. For example, z is equal to 1, which means that the rotation movement is performed around the positive direction of the z-axis;
- Limit represents some limitations of movement, such as minimum position, maximum position, and maximum speed.

4. Complete robot model



Finally, all link and joint tags complete the description and combination of each part of the robot, and are all placed in one robot tag to form a complete robot model.



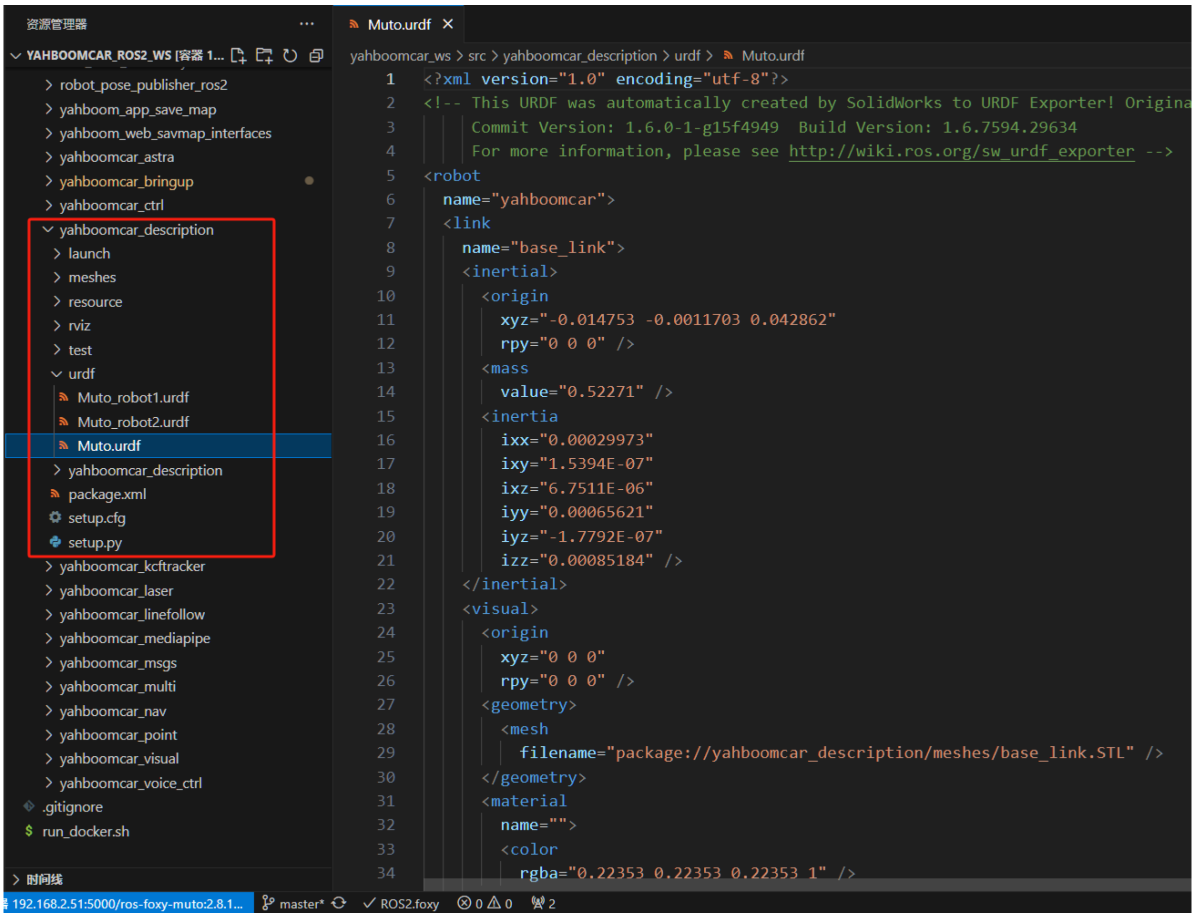
```
<robot name="<name of the robot>">
    <link> ....... </link>
    <link> ....... </link>

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

So when you look at a certain URDF model, don't rush to look at the details of each piece of code. First look for the links and joints to see what parts the robot is composed of. After understanding the whole situation, then look at the details.

4.1. Create robot model

Taking the muto model as an example, the function package structure is as follows:



```
资源管理器                                   ...
v YAHBOOMCAR_ROS2_WS [容器 1...
  > robot_pose_publisher_ros2
  > yahboom_app_save_map
  > yahboom_web_savmap_interfaces
  > yahboomcar_astra
  > yahboomcar_bringup
  > yahboomcar_ctrl
  v yahboomcar_description
    > launch
    > meshes
    > resource
    > rviz
    > test
    v urdf
      Muto_robot1.urdf
      Muto_robot2.urdf
      Muto.urdf
    > yahboomcar_description
    package.xml
    setup.cfg
    setup.py
  > yahboomcar_kcftracker
  > yahboomcar_laser
  > yahboomcar_linefollow
  > yahboomcar_mediapipe
  > yahboomcar_msgs
  > yahboomcar_multi
  > yahboomcar_nav
  > yahboomcar_point
  > yahboomcar_visual
  > yahboomcar_voice_ctrl
  .gitignore
  run_docker.sh
> 时间线
```

Muto.urdf

yahboomcar_ws > src > yahboomcar_description > urdf > Muto.urdf

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- This URDF was automatically created by SolidWorks to URDF Exporter! Origina
     Commit Version: 1.6.0-1-g15f4949  Build Version: 1.6.7594.29634
     For more information, please see http://wiki.ros.org/sw_urdf_exporter -->
<robot
  name="yahboomcar">
  <link
    name="base_link">
    <inertial>
      <origin
        xyz="-0.014753 -0.0011703 0.042862"
        rpy="0 0 0" />
      <mass
        value="0.52271" />
      <inertia
        ixx="0.00029973"
        ixy="1.5394E-07"
        ixz="6.7511E-06"
        iyy="0.00065621"
        iyz="-1.7792E-07"
        izz="0.00085184" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://yahboomcar_description/meshes/base_link.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.22353 0.22353 0.22353 1" />
```

192.168.2.51:5000/ros-foxy-muto:2.8.1...   master*   ✓ ROS2.foxy   ⊗ 0 ⚠ 0   2

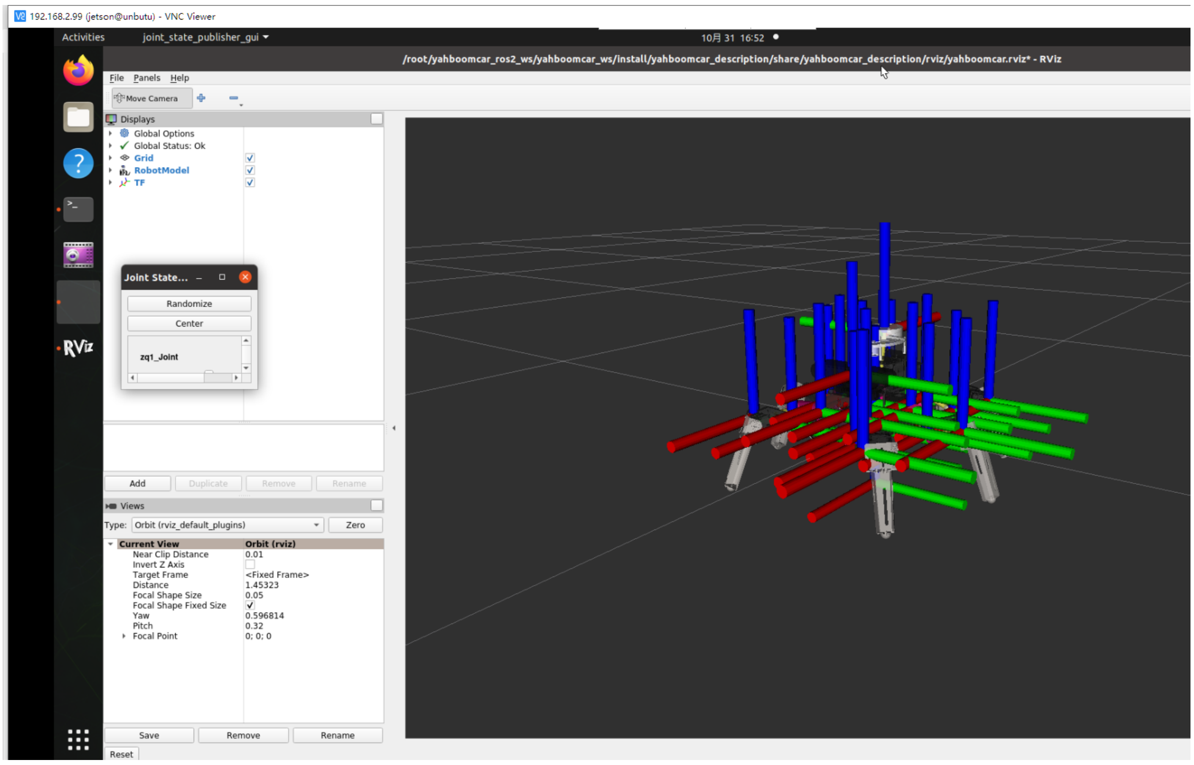
- urdf: URDF or xacro file that stores the robot model
- meshes: Place the model rendering files referenced in the URDF
- launch: Save related startup files
- rviz: Save rviz configuration file

4.2. Model visualization effect

In docker, ensure that GUI display is turned on and the terminal is started:

```
ros2 launch yahboomcar_description display.launch.py
```

In the host's vnc, you can see that Rviz is turned on and the robot model is displayed:



/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_description/share/yahboomcar_description/rviz/yahboomcar.rviz* - RViz

Global Status: Ok

Type: Orbit (rviz_default_plugins)

```
Current View           Orbit (rviz)
  Near Clip Distance   0.01
  Invert Z Axis
  Target Frame         <Fixed Frame>
  Distance             1.45323
  Focal Shape Size     0.05
  Focal Shape Fixed Size ✓
  Yaw                  0.596814
  Pitch                0.32
  Focal Point          0; 0; 0
```

4.3. Model file analysis

For the specific URDF model, please check the urdf file research:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_description/urdf/Muto.urdf
```