RoArm-M1 Tutorial IV: URDF Model Tutorial

From Waveshare Wiki
Jump to: navigation, search

RoArm-M1 Tutorial Directory

- RoArm-M1 Tutorial I: How To Use (/wiki/RoArm-M1_Tutorial_I:_How_To_Use)
- RoArm-M1 Tutorial II: Secondary Development Tutorial (/wiki/RoArm-M1_Tutorial_II:_Secondary_Development_Tutorial)
- RoArm-M1 Tutorial III: VMware ROS2 Getting Started Tutorial (/wiki/RoArm-M1_Tutorial_III:_VMware_ROS2_Getting_Started_Tutorial)
- RoArm-M1 Tutorial IV: URDF Model Tutorial
- RoArm-M1 Tutorial V: ROS2 Serial Communication Node (/wiki/RoArm-M1_Tutorial_V:_ROS2_Serial_Communication_Node)
- RoArm-M1 Tutorial VI: How to assemble RoArm-M1 (/wiki/RoArm-M1_Tutorial_VI:_How_to_assemble_RoArm-M1)
- RoArm-M1 Tutorial VII: Assembly Graphics Tutorial (/wiki/RoArm-M1_Tutorial_VII:_Assembly_Graphics_Tutorial)
- RoArm-M1 Tutorial VIII: Use of rqt in ROS2 (/wiki/RoArm-M1_Tutorial_VIII:_Use_of_rqt_in_ROS2)
- RoArm-M1 Tutorial IV: URDF Model Tutorial
- RoArm-M1 Main Page (/wiki/RoArm-M1)

RoArm-M1 ROS2 URDF Model Tutorial

- This tutorial is for building the RoArm-m1 ROS2 URDF model.

Install urdf_tutorial Software Package=

If you have followed the previous tutorials slowly up to this point, then you can ignore the step of installing the joint_state_publisher package; if you are following this tutorial directly, then make sure you have installed the joint_state_publisher package first. Execute the following command in your Ubuntu terminal:

```
sudo apt-get install joint_state_publisher
```

Install urdf_tutorial software package:

```
sudo apt update && sudo apt install ros-humble-urdf-tutorial
```

Download urdf_tutorial official software package:

```
sudo git clone https://github.com/ros/urdf_tutorial.git (https://github.com/ros/urdf_tutorial.git)
```

All the robot models (and the source files) mentioned in this tutorial can be found in the urdf_tutorial package.

Create a Model

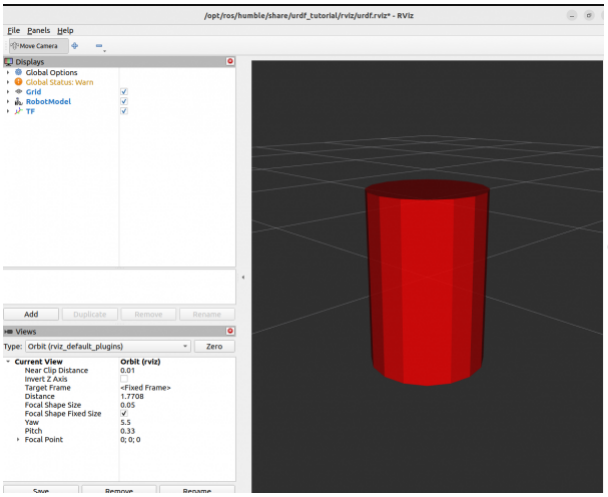
First, we can try to create a simple model. Here is a simple URDF file example:

```
<?xml version="1.0"?>
<robot name="myfirst">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>
</robot>
```

This is a robot named "myfirst", which only includes a model. Its visual model is a cylinder with a length of 0.6m and a radius of 0.2m.
To view the model, start the display.launch.py file by executing the following command:

```
cd urdf_tutorial
ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf
```

Note that the above startup command must have been executed by going into the urdf_tutorial directory first; if not, the relative path to 01-myfirst.urdf will be invalid.
After starting display.launch.py, you should see Rviz displayed as shown:



(/wiki/File:URDF_Model_Tutorial-01.png)

As you can see, the origin of this visual model (cylinder) is by default located at the center of its geometry. Thus, half of the cylinder is located below the mesh.

Create Multiple Models

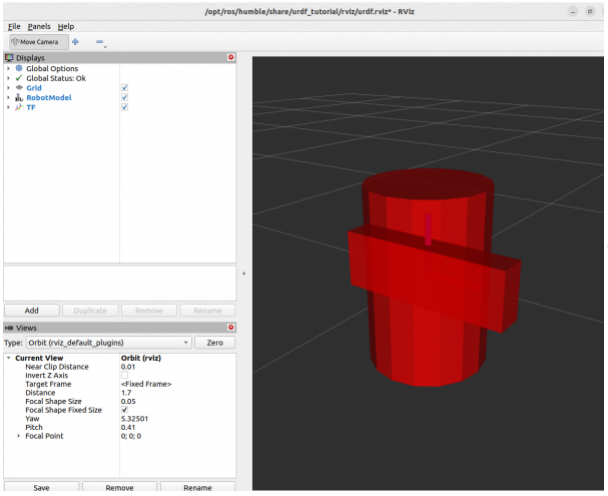
Now let's see how to add multiple models. If we just add more models to the URDF, the parser will not know how to arrange them. Therefore, we also need to add the link part of the connecting joints. Joints can refer to both bendable and non-bendable joints. We will start with the non-bendable joints.

```
<?xml version="1.0"?>
<robot name="multipleshapes">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
  </joint>
</robot>
```

Here we have defined a rectangle with dimensions 0.6m long, 0.1m wide, and 0.2m high. The definition of joints involves parent and child links. This means that the position of the model named right_leg depends on the position of the base_link model.
In the terminal interface where 01-myfirst.urdf is executed, pressing Ctrl+C exits the Rviz interface. To view the models of 02-multipleshapes.urdf, start the display.launch.py file by executing the following command:

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/02-multipleshapes.urdf
```

After launching display.launch.py, you should see Rviz displayed as shown:



(/wiki/File:URDF_Model_Tutorial-02.png)

As you can see the two models partly share the same origin, so currently the two models overlap. If we don't want them to overlap, we have to define more origins.

Origin

The model of right_leg is attached to one side of the upper part of the cylinder model, so we specify it as the origin of the joint. In addition, we need to offset the origin for the right_leg model. We also rotate the model to make it upright.

```
<?xml version="1.0"?>
<robot name="origins">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0 -0.22 0.25"/>
  </joint>
</robot>
```

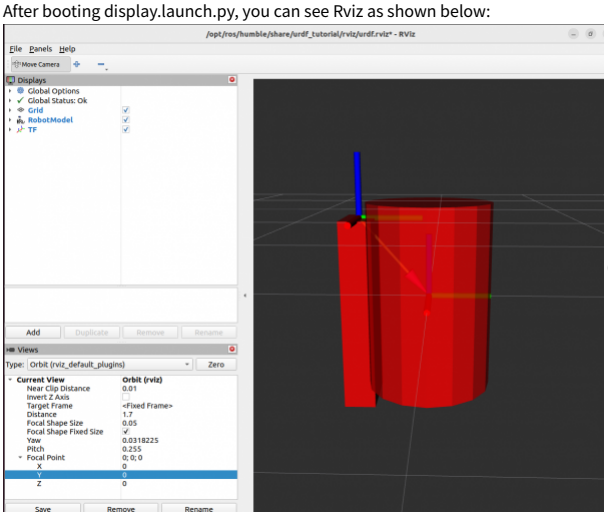
First, let's examine the origin of the joint. It's defined relative to the parent link's reference coordinate system. Earlier, we mentioned that the origin of this cylinder is at the center of this geometric shape, where x, y, and z are all 0. Since we want the 'right_leg' model to be attached to the side of the cylinder, we'll adjust it first by -0.22 meters along the y-axis (which is to the right concerning the coordinate axis but appears to the left from our perspective). Additionally, we want the 'right_leg' to attach to the upper part of the cylinder's side, so we'll adjust it 0.25 meters along the z-axis (upward).

Next, check the origin of the 'right_leg' model, which has offsets in xyz and rpy. Since we want the 'right_leg' model to attach to the upper part of the cylinder's side, we'll offset the z-axis by -0.3 meters, allowing the 'right_leg' origin to move downwards. Additionally, we want the 'right_leg' model to align its length with the z-axis, so we rotate it around the y-axis for visual inspection.

To view the model in '03-origins.urdf', execute the following command to start the 'display.launch.py' file:

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/03-origins.urdf
```

After booting display.launch.py, you can see Rviz as shown below:



(/wiki/File:URDF_Model_Tutorial-11.png)

The launch file will run some packages that create TF frames for each link in your URDF model. These packages generate TF frames based on your URDF. Rviz uses these TF frames to determine where each part of the model should be displayed.

<material> Tag Selected As Reference

This part was explained in detail in tutorial four. Currently, since we haven't set the <material> tag, all models are displayed in red.

```
<material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>

  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>

  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      <material name="white"/>
    </visual>
  </link>

  <link name="left_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      <material name="white"/>
    </visual>
  </link>
```

Now, you can see we've referenced <material> tags with names 'blue' for the cylinder, and 'white' for the right and left legs, respectively. Use the following command to view the model colors:

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/04-materials.urdf
```

Finish Model Building

Now let's add the feet, wheels, and head to complete the model. It's worth noting that we've added a sphere and some mesh files.
Below is the code for adding the sphere:

```
<link name="head">
  <visual>
    <geometry>
      <sphere radius="0.2"/>
    </geometry>
    <material name="white"/>
  </visual>
</link>
```

The mesh files used here are separate and require specifying their paths. You can use the format 'package://PACKAGE_NAME/path' to specify the path. The mesh files in this tutorial are located within a folder named 'meshes' in the 'urdf_tutorial' package. Therefore, the specific path is as follows:

```
<link name="left_gripper">
  <visual>
    <origin rpy="0.0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://urdf_tutorial/meshes/l_finger.dae"/>
    </geometry>
  </visual>
</link>
```

These mesh files can be imported using various formats. STL is a common format, but the engine also supports DAE format, which can have its color data, meaning you don't need to specify color/material.
This constitutes a complete URDF model. Now, you can proceed to learn the tutorial on constructing the RoArm-M1 robotic arm model.