URDF Model Understanding

ROS packages for robot modeling:

ROS provides some good packages that can be used to build 3D robotic models. In this section, we will discuss some of the important ROS packages that are commonly used to build robotic models:

- robot_model: ROS has a meta package called robot_model, which contains important packages that help build the 3D robot models.
- urdf: One of the important packages inside the robot_model Meta package is **Unified Robot Description Format** (**URDF**). The URDF package contains a C++ parser for the URDF, which is an XML file to represent a robot model.

We can define a robot model, sensors, and a working environment using URDF and can parse it using URDF parsers. We can only describe a robot in URDF that has a tree-like structure in its links, that is, the robot will have rigid links and will be connected using joints. Flexible links can't be represented using URDF. The URDF is composed using special XML tags and we can parse these XML tags using parser programs for further processing. We can work on URDF modeling in the following sections.

- **joint_state_publisher**: This tool is very useful while designing robot models using URDF. This package contains a node called joint_state_publisher, which reads the robot model description, finds all joints, and publishes joint values to all non-fixed joints using GUI sliders. The user can interact with each robot joint using this tool and can visualize using RViz. While designing URDF, the user can verify the rotation and translation of each joint using this tool.
- **robot_state_publisher**: This package reads the current robot joint states and publishes the 3D poses of each robot link using the kinematics tree built from the URDF. The 3D pose of the robot is published as ROS tf (transform). ROS tf publishes the relationship between the coordinate frames of a robot.
- **xacro**: xacro stands for (XML Macros) and we can define how xacro is equal to URDF plus add-ons. It contains some add-ons to make URDF shorter and readable. It can be used for building complex robot descriptions. We can convert xacro to URDF at any time using some ROS tools.

Understanding robot modeling using URDF

In this section, we will look at the **URDF** XML tags, which help to model the robot. We have to create a file and write the relationship between each link and joint in the robot and save the file with the **.urdf extension**.

The URDF can represent the kinematic and dynamic description of the robot, a visual representation of the robot.

The following tags are the commonly used URDF tags to compose a URDF robot model:

- **link:** The link tag represents a single link of a robot. Using this tag, we can model a robot link and its properties. The modeling includes size, shape, color, and can even import a 3D mesh to represent the robot link. We can also provide dynamic properties of the link such as inertial matrix and collision properties.

The syntax is as follows:

```
<link name="<name of the link>">

<inertial>...........</inertial>

  <visual> ............</visual>

  <collision>..........</collision>

</link>
```

The following is a representation of a single link. The **Visual** section represents the real link of the robot, and the area surrounding the real link is the **Collision** section. The **Collision** section encapsulates the real link to detect a collision before hitting the real link.

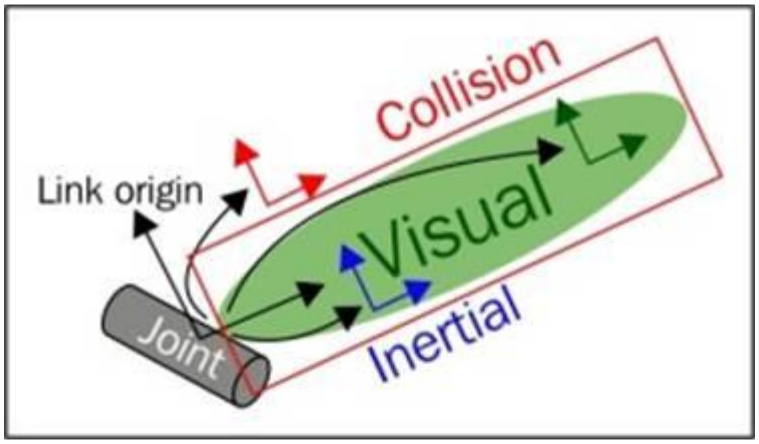


Figure 1.1: Visualization of a URDF link

- **joint:** The joint tag represents a robot joint. We can specify the kinematics and dynamics of the joint and also set the limits of the joint movement and its velocity. The joint tag supports the different types of joints such as **revolute**, **continuous**, **prismatic**, **fixed**, **floating**, and **planar**.

The syntax is as follows:

```
<joint name="<name of the joint>">
```

```
  <parent link="link1"/>

  <child link="link2"/>

  <calibration .... />

  <dynamics damping ..../>

  <limit effort .... />

</joint>
```

A URDF joint is formed between two links; the first is called the **Parent** link and the second is the **Child** link. The following is an illustration of a joint and its link:

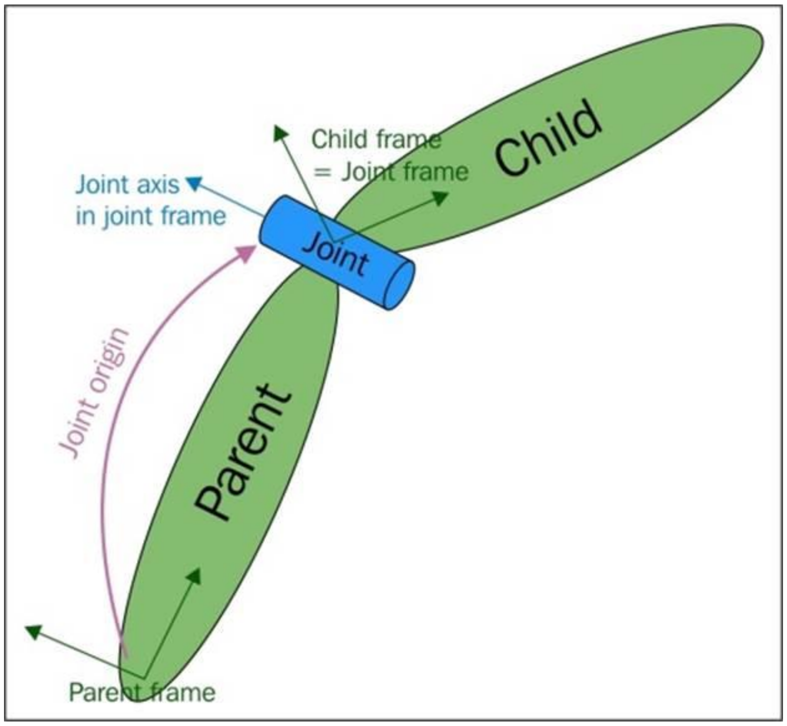


Figure 1.2: Visualization of a URDF joint

- **robot:** This tag encapsulates the entire robot model that can be represented using URDF. Inside the robot tag, we can define the name of the robot, the links, and the joints of the robot.

The syntax is as follows:

```
<robot name="<name of the robot>"

  <link>  ..... </link>
  <link> ...... </link>

  <joint> ....... </joint>

  <joint> ........</joint>

</robot>
```

A robot model consists of connected links and joints. Here is a visualization of the robot model:

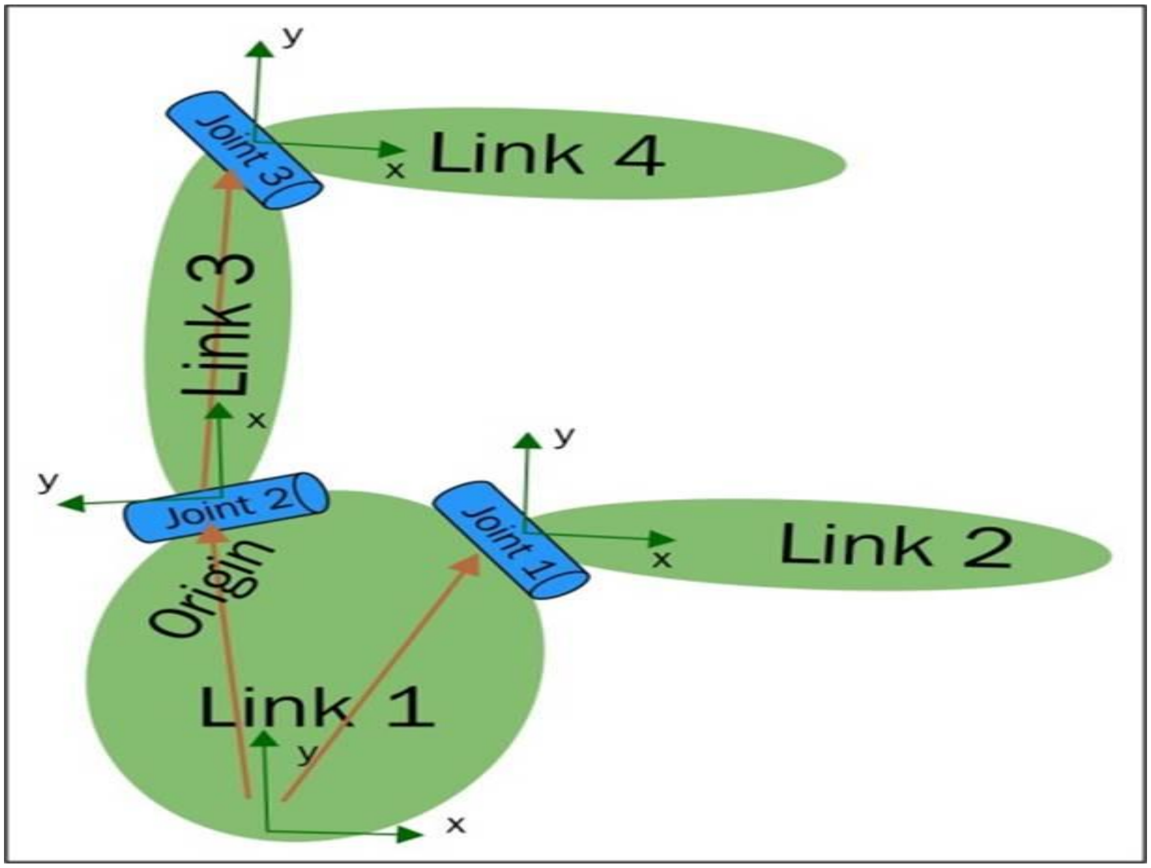


Figure 1.3: Visualization of a robot model having joints and links

- **gazebo:** This tag is used when we include the simulation parameters of the Gazebo simulator inside URDF. We can use this tag to include **gazebo plugins**, gazebo material properties, and so on. The following shows an example using gazebo tags:

```
<gazebo reference="link_1">

 <material>Gazebo/Black</material>

  </gazebo>
```

We can find more URDF tags at http://wiki.ros.org/urdf/XML.

Creating our first URDF model

After learning about URDF and its important tags, we can start some basic modeling using URDF. The first robot mechanism that we are going to design is a pan and tilt mechanism as shown in Figure 1.4.

Let's see the URDF code of this mechanism. Navigate to task_1/urdf and open pan_tilt.urdf.

There are three links and two joints in this mechanism. The base link is static, in which all other links are mounted. The first joint can pan on its axis and the second link is mounted on the first link and it can tilt on its axis. The two joints in this system are of a revolute type. The mechanism will be well understood at the end of this document when you take a look at the simulation.

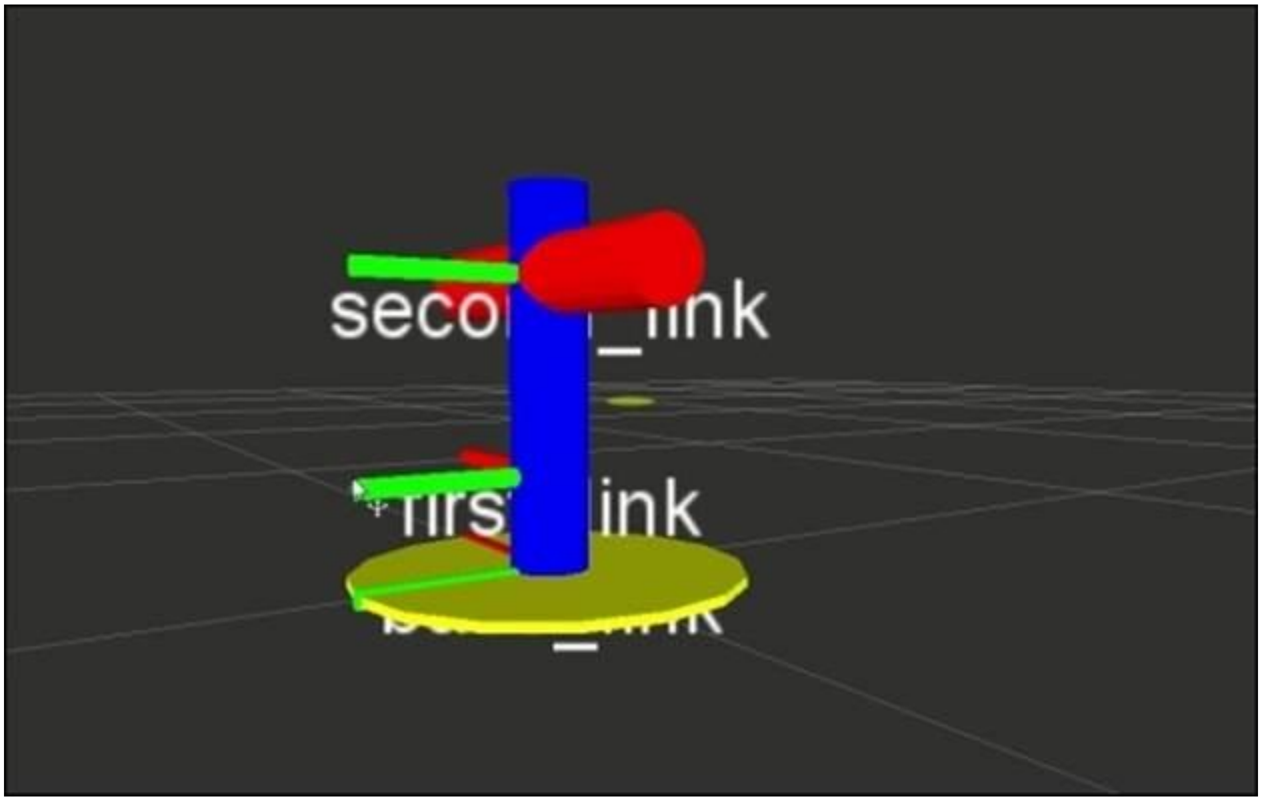


Figure 1.4: Visualization of a pan and tilt mechanism in RViz

The code indentation in URDF is not mandatory but maintaining indentation can improve code readability.

Explaining the URDF file

When we check the code, we can add a <robot> tag at the top of the description:

```
<?xml version="1.0"?>

<robot name="pan_tilt">
```

The <robot> tag defines the name of the robot that we are going to create. Here, we named the robot pan_tilt.

If we check the sections after the <robot> tag definition, we can see link and joint definitions of the pan and tilt mechanism:

```
<link name="base_link">

  <visual>

    <geometry>

    <cylinder length="0.01" radius="0.2"/>

    </geometry>

    <origin rpy="0 0 0" xyz="0 0 0"/>

    <material name="yellow">

      <color rgba="1 1 0 1"/>

    </material>

  </visual>

</link>
```

The preceding code snippet is the base_link definition of the pan and tilt mechanism. The <visual> tag can describe the visual appearance of the link, which is shown on the robot simulation. We can define the link geometry (cylinder, box, sphere, or mesh) and the material (color and texture) of the link using this tag:

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

  <parent link="base_link"/>

  <child link="pan_link"/>
  <origin xyz="0 0 0.1"/>

  <axis xyz="0 0 1" />

</joint>
```

In the preceding code snippet, we define a joint with a unique name and its joint type. The joint type we used here is revolute and the parent link and child link are base_link and the pan_link respectively. The joint origin is also specified inside this tag.

Save the preceding URDF code as pan_tilt.urdf and check whether the urdf contains errors using the following command:

check_urdf pan_tilt.urdf

The check_urdf command will parse urdf and show an error. If everything is OK, it will show an output as follows:

```
robot name is: pan_tilt

---------- Successfully Parsed XML ---------------

root Link: base_link has 1 child(ren)

    child(1):  pan_link

        child(1):  tilt_link
```

If we want to view the structure of the robot links and joints graphically, we can use a command tool called urdf_to_graphiz:

urdf_to_graphiz pan_tilt.urdf

This command will generate two files: pan_tilt.gv and pan_tilt.pdf. We can view the structure of this robot using the following command:

evince pan_tilt.pdf

You will get the following as shown in Figure 1.5.

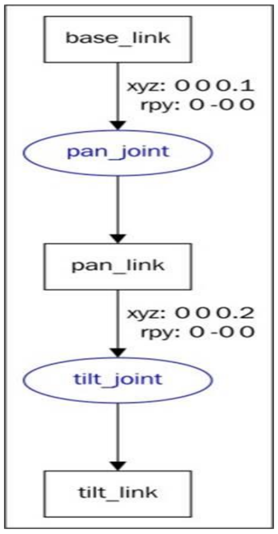


Figure 1.5: Graph of joint and links in pan and tilt mechanism

Visualizing the robot 3D model in RViz

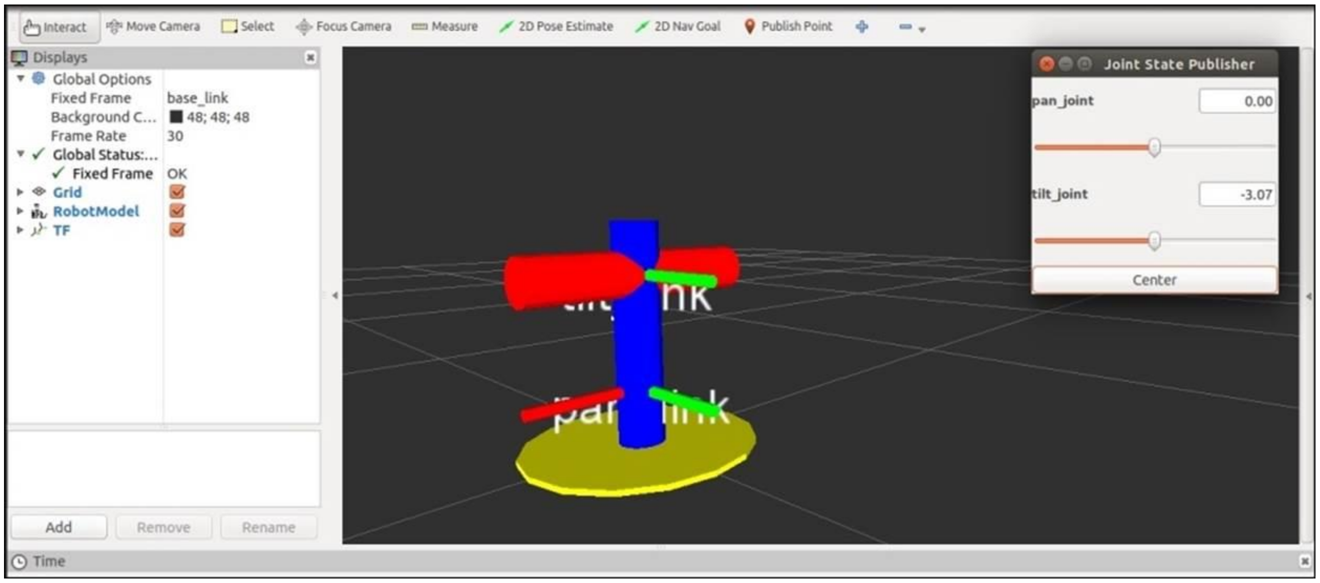
After designing URDF, we can view it on RViz. We can create a **view_demo.launch** file and put the following code into the launch folder.

```
<launch>

  <arg name="model" />

<param name="robot_description"textfile="$(find task_1)/urdf/pan_tilt.urdf" />

<param name="use_gui" value="true"/>

<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />

        <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />

 <node name="rviz" pkg="rviz" type="rviz" args="-d $(find task_1)/urdf/robot.rviz" required="true" />

</launch>
```

We can launch the model using the following command:

```
roslaunch task_1 view_demo.launch
```

If everything works fine, we will get a pan and tilt mechanism in RViz.



```
Global Options
  Fixed Frame        base_link
  Background C...    48; 48; 48
  Frame Rate         30
Global Status:...
  Fixed Frame        OK
```

```
pan_joint    0.00
tilt_joint   -3.07
```

Figure 1.6: Joint level of pan and tilt mechanism

Interacting with pan and tilt joints

You can see an **extra GUI** as shown in Figure 1.6, which contains sliders to control pan joints and tilt joints. This GUI is called the **Joint State Publisher** node from the joint_state_publisher package:

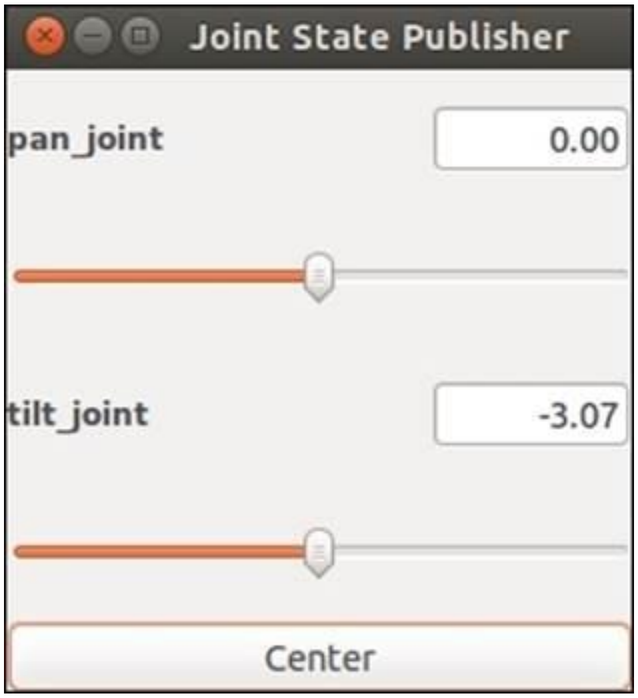
```
<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
```

We can include this node in the launch file using the above statement. The limits of pan and tilt should be mentioned inside the joint tag:

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

  <parent link="base_link"/>

  <child link="pan_link"/>

  <origin xyz="0 0 0.1"/>

  <axis xyz="0 0 1" />

  <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/>

  <dynamics damping="50" friction="1"/>
```

```
</joint>
```

The <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/> defines the limits of effort, velocity, and angle limits. The effort is the maximum force supported by this joint; lower and upper indicates the lower and upper limit of the joint in radians for the revolute type joint, and meters for prismatic joints. The velocity is the maximum joint velocity.



pan_joint 0.00

tilt_joint -3.07

Figure 1.7: Joint level of pan and tilt mechanism

Figure 1.7 shows the GUI of **Joint State Publisher** with sliders and current joint values shown in the box.

Adding physical and collision properties to a URDF model

Before simulating a robot in a robot simulator, such as Gazebo, and rviz, we need to define the robot link's physical properties such as geometry, color, mass, and inertia, and the collision properties of the link.

We will only get good simulation results if we define all these properties inside the robot model. URDF provides tags to include all these parameters and code snippets of base_link contained in these properties as given here:

```
<link>

......

<collision>
```

```
      <geometry>

      <cylinder length="0.03" radius="0.2"/>

      </geometry>

      <origin rpy="0 0 0" xyz="0 0 0"/>

    </collision>

    <inertial>

    <mass value="1"/>

    <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>

    </inertial>

...........

</link>
```

Here, we define the collision geometry as cylinder and the mass as 1 Kg, and we also set the inertial matrix of the link.

The collision and inertia parameters are required in each link; otherwise, Gazebo will not load the robot model properly.

You can study the **Firebird urdf model** for better understanding. You can find **Firebird urdf model** in **task1 package; urdf folder** provided in this task