Workshop 4 3D Robot Modelling in ROS

Aims and Objectives

Understanding how to create a virtual robot model to be used in ROS

Things to learn

- ROS packages for robot modelling
- Creating the ROS package for the robot description
- Understanding robot modelling using URDF
- Understanding robot modelling using xacro
- Converting xacro to URDF
- Creating a robot description for a seven DOF robot manipulator
- Working with the joint state publisher and robot state publisher

ROS has a standard meta package for designing and creating robot models called robot_model, which consists of a set of packages, some of which are called urdf, kdl_parser, robot_state_publisher, and collada_urdf. These packages help us create the 3D robot model description with the exact characteristics of the real hardware.

Exercise

1. **Understanding robot modelling using URDF**
   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.
     http://wiki.ros.org/urdf/XML/link

   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 collision before hitting the real link

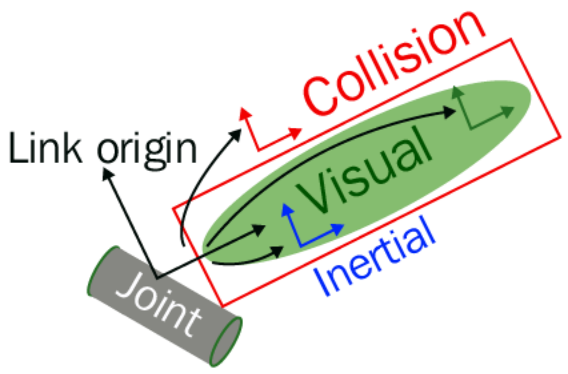


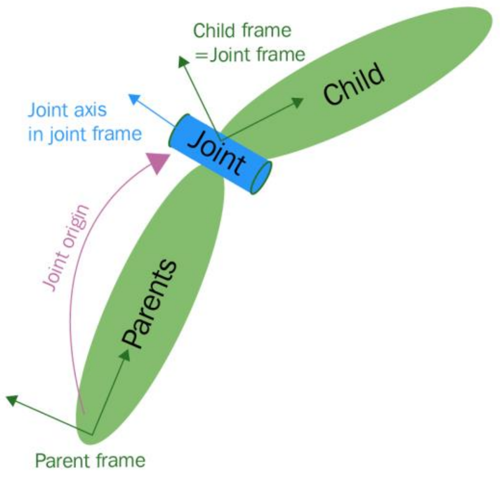
Figure 1: Visualization of a URDF link

   - joint: The joint tag represents a robot joint.
     http://wiki.ros.org/urdf/XML/joint

The syntax is as follows:

```
<joint name="<name of the joint>">
        <parent link="link1"/>
        <child link="link2"/>
        <calibration .... />
        <dynamics damping ..../>
        <limit effort .... />
</joint>
```

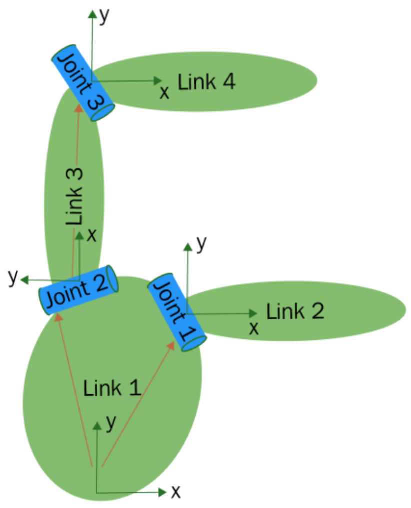
A URDF joint is formed between two links; the first is called the Parent link, and the second is called the Child link. The following is an illustration of a joint and its link:



- robot: This tag encapsulates the entire robot model that can be represented using URDF. http://wiki.ros.org/urdf/XML/robot

  The syntax is as follows:

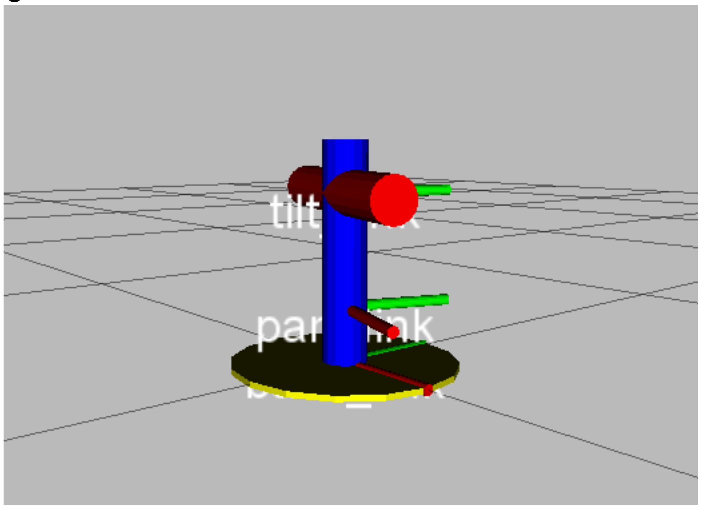
```
<robot name="<name of the robot>"
        <link> ..... </link>
        <link> ...... </link>
        <joint> ....... </joint>
        <joint> ........</joint>
</robot>
```



- gazebo: This tag is used when we include the simulation parameters of the Gazebo simulator inside the 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>
```

2. **Creating the ROS package for the robot**
   - cd ~/catkin_ws/src
   - catkin_create_pkg MystudentID_robot_description_pkg roscpp tf geometry_msgs urdf rviz xacro
   - Install the urdf and xacro packages by: (You don't need this step as I have installed)
     $sudo apt-get install ros-melodic-urdf
     $sudo apt-get install ros-melodic-xacro
3. **Creating our first URDF model**
   The first robot mechanism that we are going to design is a pan-and-tilt mechanism, as shown in the following figure.



There are three links and two joints in this mechanism. The base link is static, and all the other links are mounted on it. 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.
Open the file browser and copy the MystudentID_robot_desciption_pkg in folder in ~/Eng_7_rob/workshop_4/ to your ~/catkin_ws/src/ folder. In the urdf folder of this package, open the pan_tilt.urdf file. I will explain the meaning of each line in the lab time.

Open a terminal and do the following:

- *cd ~/catkin_ws*
- *catkin_make*
- *source devel/setup.bash*
- *cd src/MystudentID_robot_description_pkg/urdf*
- *check_urdf pan_tilt.urdf*
  copy the output of this operation to your logbook
- *urdf_t0_graphiz pan_tilt.urdf* (this command will generate two files: pan_tilt.gv and pan_tilt.pdf)
- *evince pan_tilt.pdf (*put the output to your logbook)

4. **Visualizing the 3D robot model in RViz**

We can create a *view_demo.launch* launch file and put the following code into the launch folder. Navigate to the MystudentID _ros_robot_description_pkg/launch directory for the code:

```
<launch>
        <arg name="model" />
        <param name="robot_description" textfile="$(find
                MyStudentID_robot_description_pkg)/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
                MyStudentID_robot_description_pkg)/urdf.rviz" required="true" />
</launch>
```

We can launch the model using the following command:

```
$ roslaunch MyStudentID_robot_description_pkg view_demo.launch
```

If everything works fine, we will get a pan-and-tilt mechanism in RViz. Please put the screen shot of the RViz output to your logbook. I should explain a bit of the RViz with you in the class.

5. **Adding physical and collision properties to a URDF model**

   Before simulating a robot in a robot simulator, such as Gazebo or V-REP, we need to define the robot link's physical properties, such as geometry, colour, mass, and inertia, as well as the collision properties of the link. We should be able to see this in the model provided.

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

6. **Understanding robot modelling using xacro**

   The flexibility of URDF reduces when we work with complex robot models. Some of the main features that URDF is missing are simplicity, reusability, modularity, and programmability.

   The URDF is a single file and we can't include other URDF files inside it. This reduces the modular nature of the code. All code should be in a single file, which reduces the code's simplicity.
   Also, if there is some programmability, such as adding variables, constants, mathematical expressions, and conditional statements, in the description language, it will be more userfriendly.
   The robot modeling using xacro meets all of these conditions. Some of the main features of xacro are as follows:

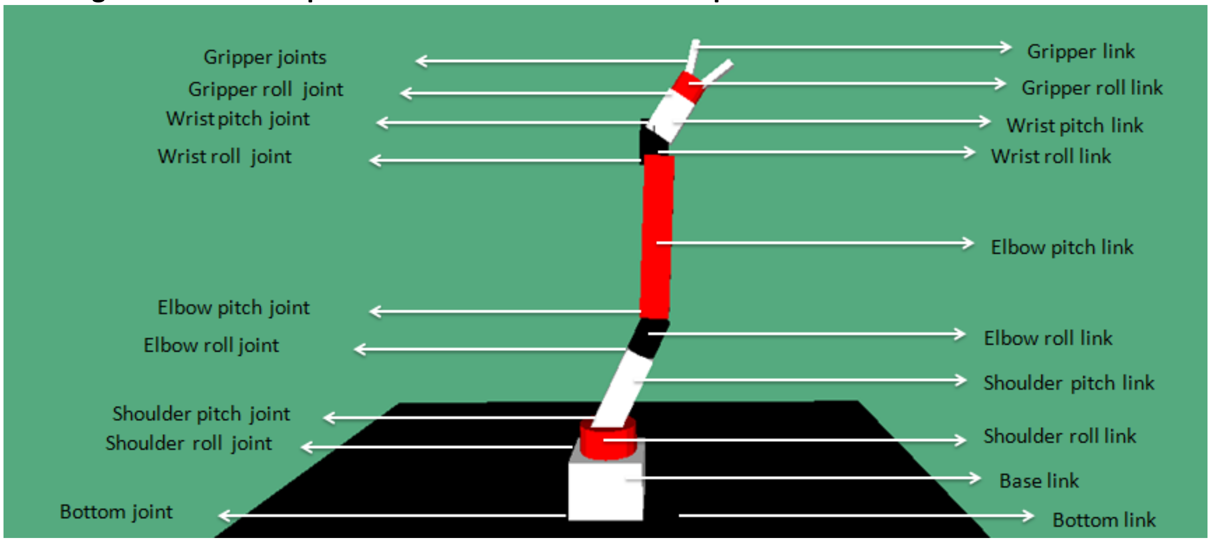
   Simplify URDF: The xacro is the cleaned-up version of URDF. It creates macros inside the robot description and reuses the macros. This can reduce the code length. Also, it can include macros from other files and make the code simpler, more readable, and more modular.
   Programmability: The xacro language supports a simple programming statement in its description. There are variables, constants, mathematical expressions, conditional statements, and so on that make the description more intelligent and

efficient.

We can say that xacro is an updated version of URDF, and we can convert the xacro definition to URDF whenever it is necessary, using some ROS tools.

I should explain in class that how you can use: properties, the math expression and macros.

7. Creating the robot description for a seven DOF robot manipulator



Arm specification

Here is the robot arm specification of this seven DOF arm:
Degrees of freedom: 7
Length of the arm: 50 cm
Reach of the arm: 35 cm
Number of links: 12
Number of joints: 11

Type of joints

Here is the list of joints containing the joint name and its type of robot:

Joint number	Joint name	Joint type	Angle limits (in degrees)
1	bottom_joint	Fixed	--
2	shoulder_pan_joint	Revolute	-150 to 114
3	shoulder_pitch_joint	Revolute	-67 to 109
4	elbow_roll_joint	Revolute	-150 to 41
5	elbow_pitch_joint	Revolute	-92 to 110
6	wrist_roll_joint	Revolute	-150 to 150
7	wrist_pitch_joint	Revolute	92 to 113
8	gripper_roll_joint	Revolute	-150 to 150
9	finger_joint1	Prismatic	0 to 3 cm
10	finger_joint2	Prismatic	0 to 3 cm

You can try to built seven_dof_arm.urdf yourself. However in the package folder there are two files; one is seven_dof_arm.urdf and one is seven_dof_arm.xacro. I shall briefly take a look at the xacro file with you. Try to understand these files. They are good examples of how to create 3D modelling package for a robot.

8. Viewing the seven DOF arm in RViz

Launch the view_arm.launch file to view 7 DOF arm inside the launch folder by :

roslaunch MyStudentID_robot_description_pkg view_arm.launch

Copy the screen shot to your logbook. Try to interact with the joint slider and move the joints of the robot. Comments on your observations.