Introduction to Mobile Robotics

Practical Work 5: tf2 & URDF

Zhi Yan

ENSTA, Institut Polytechnique de Paris

1 Objective

The goal of this session is to describe a robot's physical structure using the **Unified Robot Description Format (URDF)** and to manage its coordinate frames using **tf2**. By the end, you will be able to:

- Write a simple URDF file for a differential drive robot.
- Launch a `robot_state_publisher` to publish the robot's kinematic chain as tf2 transforms.
- Visualize the robot model and its coordinate frames in RViz2.
- Publish joint states to animate the robot model.

2 What are URDF and tf2?

- URDF: An XML file that describes the robot as a tree of links and joints.
  - A **link** is a rigid part of the robot (e.g., the chassis, a wheel). It has inertial and visual properties.
  - A **joint** connects two links and defines how one link can move relative to the other (e.g., revolute, continuous, fixed).
- **tf2:** A ROS 2 library that lets you keep track of multiple coordinate frames over time. The `robot_state_publisher` node reads the URDF and automatically publishes the transformations between the robot's links to tf2.

3 Creating a Robot Description Package

1. Open a new terminal and navigate to your workspace's source directory.

```
cd ~/ros2_ws/src
```

2. Create a new package to hold our robot's description.

```
ros2 pkg create --build-type ament_cmake robot_description_pkg
```

3. Inside this new package, create directories for launch files and URDF files.

```
cd robot_description_pkg
mkdir launch urdf
```

4 Writing the URDF File

1. Create a new file named `urdf/diff_drive_robot.urdf`.

```
touch urdf/diff_drive_robot.urdf
```

2. Open the file in a text editor and add the following content. This describes a simple robot with a chassis link and two wheel links.

```
<?xml version="1.0"?>
<robot name="diff_drive_robot">

  <link name="base_link">
    <visual>
      <geometry>
  <box size="0.4 0.2 0.1"/>
      </geometry>
      <origin xyz="0 0 0.05" rpy="0 0 0"/>
      <material name="blue">
  <color rgba="0.0 0.0 0.8 1.0"/>
      </material>
    </visual>
  </link>

  <link name="left_wheel_link">
    <visual>
      <geometry>
  <cylinder radius="0.05" length="0.04"/>
      </geometry>
      <origin xyz="0 0 0" rpy="1.5707 0 0"/>
      <material name="black">
  <color rgba="0.0 0.0 0.0 1.0"/>
      </material>
    </visual>
  </link>

  <link name="right_wheel_link">
    <visual>
      <geometry>
  <cylinder radius="0.05" length="0.04"/>
      </geometry>
      <origin xyz="0 0 0" rpy="1.5707 0 0"/>
      <material name="black"/>
    </visual>
  </link>

  <joint name="left_wheel_joint" type="continuous">
    <parent link="base_link"/>
    <child link="left_wheel_link"/>
    <origin xyz="0.1 0.13 0" rpy="0 0 0"/>
    <axis xyz="0 1 0"/>
  </joint>

  <joint name="right_wheel_joint" type="continuous">
    <parent link="base_link"/>
    <child link="right_wheel_link"/>
    <origin xyz="0.1 -0.13 0" rpy="0 0 0"/>
    <axis xyz="0 1 0"/>
```

```
  </joint>

</robot>
```

Study this file: Notice how each `<joint>` connects a `<child>` link to a `<parent>` link at a specific `<origin>` (xyz offset and rpy orientation). This defines the kinematic tree.

5 Visualizing the Robot with a Launch File

We will create a launch file that starts RViz2 and the necessary nodes to publish the robot's state.

1. Create a file named `launch/display.launch.py`.
2. Add the following Python code. This launch file looks a bit complex, but it's a standard template for visualizing robots.

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch_ros.actions import Node
import xacro

def generate_launch_description():

    # Get the path to the URDF file
    urdf_file_path = os.path.join(
        get_package_share_directory('robot_description_pkg'),
        'urdf',
        'diff_drive_robot.urdf')

    # Read the URDF file content
    with open(urdf_file_path, 'r') as file:
        robot_description_content = file.read()

    return LaunchDescription([
        # Node to publish joint states (e.g., wheel rotations)
        Node(
            package='joint_state_publisher_gui',
            executable='joint_state_publisher_gui',
            name='joint_state_publisher_gui'
        ),

        # Node to publish the robot's state (tf2 transforms) from the
   URDF
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            parameters=[{'robot_description':
   robot_description_content}]
        ),

        # RViz2 for visualization
        Node(
            package='rviz2',
```

```
            executable='rviz2',
            name='rviz2',
            output='screen'
        )
    ])
```

3. We should then edit `CMakeLists.txt` and `package.xml` to correctly install the URDF and launch files. Add these lines to `CMakeLists.txt` before the `ament_package()` call:

```
install(DIRECTORY
  urdf
  launch
  DESTINATION share/${PROJECT_NAME}
)
```

Also, add the dependencies to `package.xml`:

```
<exec_depend>joint_state_publisher_gui</exec_depend>
<exec_depend>robot_state_publisher</exec_depend>
<exec_depend>rviz2</exec_depend>
```

6 Build, Run, and Visualize

1. You may need to install some dependencies first:

```
sudo apt install ros-$ROS_DISTRO-joint-state-publisher-gui ros-
    $ROS_DISTRO-robot-state-publisher
```

2. Navigate to the root of your workspace and build:

```
cd ~/ros2_ws
colcon build --packages-select robot_description_pkg
```

3. Source the workspace and run the launch file:

```
source install/setup.bash
ros2 launch robot_description_pkg display.launch.py
```

4. Two windows should appear: RViz2 and a "Joint State Publisher" GUI with sliders.

5. Configure RViz2:

   - In the top-left "Displays" panel, set the "Fixed Frame" to `base_link`.
   - Click the "Add" button in the bottom-left.
   - In the "By display type" tab, add a **RobotModel** display.
   - In the properties displayed below **RobotModel**, ensure that "Description Topic" is set to `/robot_description`. Your blue robot should appear!
   - Click "Add" again and add a **TF** display. This will show all the coordinate frames.

6. Animate the robot:

   - Find the "Joint State Publisher" GUI window.
   - Move the sliders for the `left_wheel_joint` and `right_wheel_joint`.

- **Observe the wheels spinning in the RViz2 window!** The `joint_state_publisher_gui` is publishing joint positions, and the `robot_state_publisher` is using them to update the tf2 transforms for the wheels.

This concludes the practical session. You have successfully described a robot's kinematics, published its state to ROS 2, and visualized it.