Project 1: Modelling a four-wheeled bot

Gokul Hari

October 30, 2021

For this project, we need to model a Robot using Solidworks, launch and operate it in gazebo. This report explains the steps undertaken in achieving this objective.

1 Solid Works Model

First, we model the robot in SolidWorks. We created a rectangular base link, with 4 square protrusions. We then designed 4 L-joints where one end connects to each of the square protrusion in the base link while the other end connects to the axis of the wheel. The L-joints for the rear wheels are connected rigidly with the base link/chassis, forming the "parent link".
The L-joint connected in the front of the chassis forms a child link -level 1, each for left and right side. This front wheel setup enables a vertical revolute joint actuation that makes the front wheels steerable.
The front wheels in either side are in child link level 2. The rear wheels are directly connected to the rear L-joints that are already a part of the chassis (parent link) These rear wheels are continuous joint along horizontal axis. The rear wheels are at the child-level 1. The hierarchy of the parent and child links is shown in 1

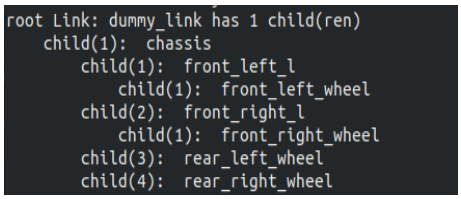

Figure 1: Link heirarchy in URDF

The model of the robot in Top and Front view is show in 2. The L joints are connected to the protrusions.The front wheels are revolute joints that rotate along the vertical axis. The rear wheels are continuous joints that rotate along horizontal axis. The model was exported to urdf using the URDF exporter tool.

2 URDF modifications

Once the urdf folder with meshes was generated, we need to identify the joints where the actuation takes place and ensure if their axes are assigned correctly. In our model, we noticed that the front L-joint axis needed correction. Moreover, for revolute joints, we need to explicitly set the limits, effort, and velocity values. A dummy link and a joint connecting to the chassis link was also added to the urdf.

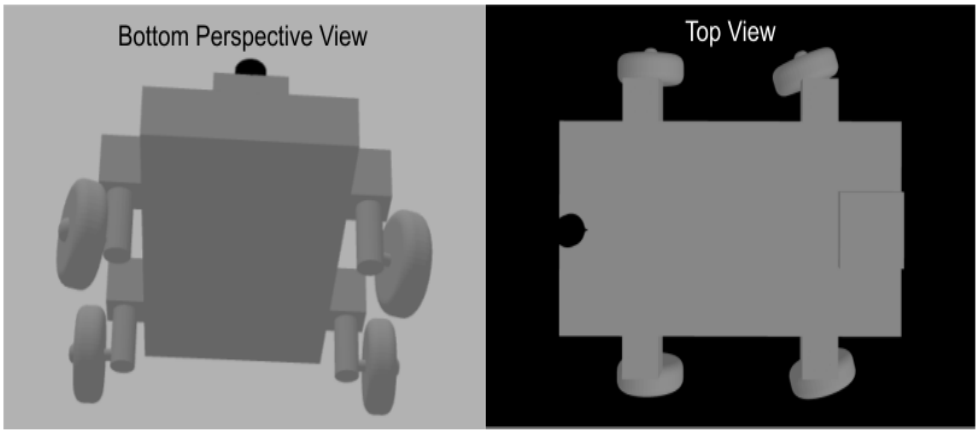

Figure 2: Top and Bottom View of the Robot

3 Controller Integration

For each movable joint, we added a transmission code block in the urdf and specified the corresponding joint names under the <joint> tag. Effort Controllers were chosen for all four transmissions. The control plugin code block was also included. In the controller configuration yaml file in the URDF, we added the names of the controller, specified the joint names and tuned the PID values. The type of controller is an important parameter here. Joint Position Controllers were chosen for the front steering wheel joints and Joint Effort Controllers were chosen for the rear wheels.

4 LIDAR Integration

Next the LIDAR needs to be integrated. The LIDAR's urdf and mesh files were moved to the corresponding folders and a xacro file was created with the LIDAR plugin code in it. We only edited the LIDAR urdf file to with the correct package name. The LIDAR "base_laser" frame is connected to the "chassis" frame which is our parent link of the robot. This xacro file is used to merge the LIDAR plugin code with our URDF file modified up till now. The resulting urdf file is used to spawn the robot in the launch file.

5 Launch File

In the launch file, we need to set the urdf file and controller configuration yaml file in parameter server. The URDF is later used to spawn the robot and yaml file is used for the controller spawner. We ensure the correct name of the controllers are used in the controller spawner. Next, in the tf node for static transform publisher, we need to set the map to base variable to dummy link of our robot to obtain the frame position in space. We also added a code snippet to launch rviz along when the gazebo simulation is started in the launch file.

6 Publisher

Once the model can spawn in the gazebo world, we need to actuate the joints of the robot. we can check the topic topics in "<controller name>/command" in rostopic. We wrote a

simple publisher that publishes a throttle value to both the rear wheel controller topics and a turn value to both the front steering wheel controller topics. When throttle is non zero, the robot moves forward/backward, The position controller in front steering wheel acts as a servo motor to rotate to the position to turn. When both turn and throttle values are non negative, the robot moves in a circular motion.
Similarly the throttle and turn value are specified in the teleop template file, to the rear and front wheels respectively, to perform teleoperation.
The video results of the project can be found here

7 References

- http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber
- http://wiki.ros.org/urdf/XML/Transmission