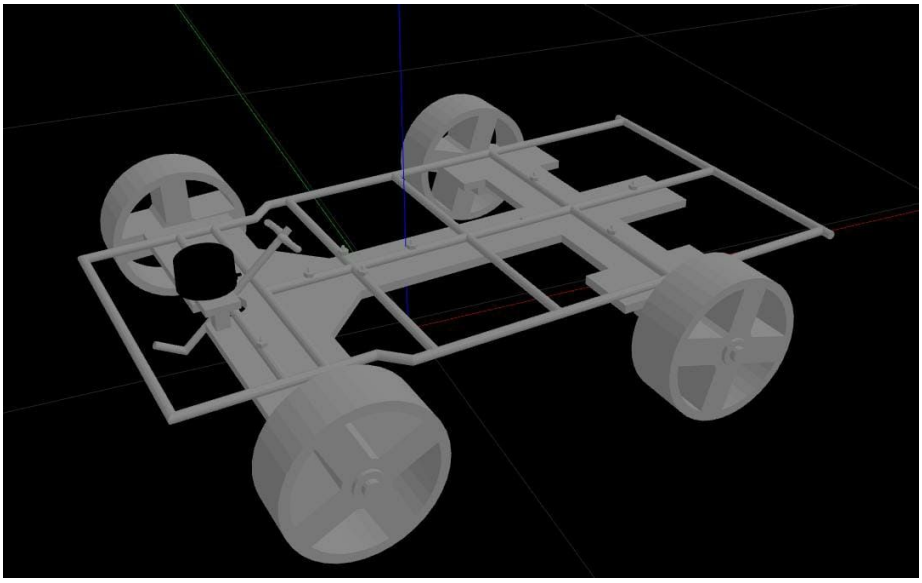
ENPM662 - Project 1 Report

Markose Jacob
UID : 117000269
markj11@umd.edu
Teammate : Pooja Kabra

The first step in this project for us was to come up with a robot model and after discussion we decided to make a robot which would use its back wheels to move and the front two wheels to steer. We used solidworks to design the model. I particularly enjoyed this part the most because I have good experience with Soldworks. We spent quite a bit of time with the assembly because our model contained many nuts and bolts (we tried to make the robot as realistic as possible). We even gave materials to the different parts, for example, Rubber to tyres, Stainless steel nuts and bolts, aluminium frame and wooden chassis. We initially came up with a design where the two front wheels would be connected and hence would need to use just one controller to steer the robot. We later had to change this design as solidworks to URDF converters would not allow us to use closed loops. The solidworks to URDF controller was a bit tricky as the type of joints selected would change at random and we had to manually select the correct type also, since we had a lot of nuts and bolts we had too many joints and links to name. We used revolute joints for the rear wheel, front left joint and front right joint. Continuous joints for the front left and right axle. Once we generated the URDF file and meshes we copied it into our catkin workspace in Ubuntu 18.04.



The remaining steps we completed on Ubuntu 18.04 by using ROS melodic and Gazebo 9.14. We copied the package created by Solidworks to URDF converter to our catkin workspace and we checked our URDF file using check_urdf from the terminal to make sure there were no errors in the URDF file. We then did catkin_make to build our workspace. We then spawned our robot in an empty world using the launch file created by the solidworks to URDF tool. The next step was to add a dummy link with its child as the root like of the robot (base_link). After this we added transmissions to the joints we intend to control. We used a velocity interface for the rear wheels and joint interface for the front two joints which we used to steer. Next, we downloaded and copied the lidar sensor files into our package and changed the paths to these files accordingly. We used the xacro file provided to us to integrate the lidar and model URDF files together. We took a while to figure out how to use the xacro file as this was our first time. After we successfully added the sensor details like its characteristics, frame name, topic, orientation, position and child and parent links we created another combined URDF to make sure the two files were integrated correctly, we did this using urdf_check. Our next step was to define our controllers in the yaml file. We used velocity controllers for the rear wheels and joint position controllers for the front two wheels. The next step was to create a launch file to spawn our model in gazebo in the map provided to us along with the controllers and lidar sensor. Figuring out the launch file took some time and after multiple attempts we were able to successfully launch our model. Once we were done with this we started up Rviz to visualise the output of the lidar sensor.

The final step was to use teleop to move the robot from the center of the map to the bottom left of the map. To do this we used the template teleop file and made the file publish to the right topics. We then fine tuned the PID values and joint limits to our satisfaction. Once we were done with this we were able to move the robot using teleop and complete the project.

Some of the issues we faced -

1. The check_urdf was giving us errors. We had accidentally given the same name to certain joints.
2. Multiple parts were spawned in Gazebo. If you give two different parents to a single part, then that part appears twice in Gazebo. You cannot have closed loops.
3. First time experience with xacro file. We took some time to get the xacro file to work correctly.
4. Could not load controller error on the terminal when we use the launch file. This was because there was a name conflict in the config file and launch file for the controller names.

5. Spawn error. This was because we did not change the name of the robot correctly everywhere.
6. Not getting correct lidar sensor output. The lidar sensor was deceiving. The sensor has a small protution and we thought this was the front side but in reality it was the opposite.
7. Not able to move the robot using teleop. This was the biggest issue we faced. The publishers and subscribers were working correctly but our robot wouldn't move. The problem was that revolute joints have joint limits and by default these values are zero and this prevented the joints from rotating. We overcame this issue by changing the joint limits in the urdf file.

Dimensions -