Project Report
Hemanth Joseph Raj

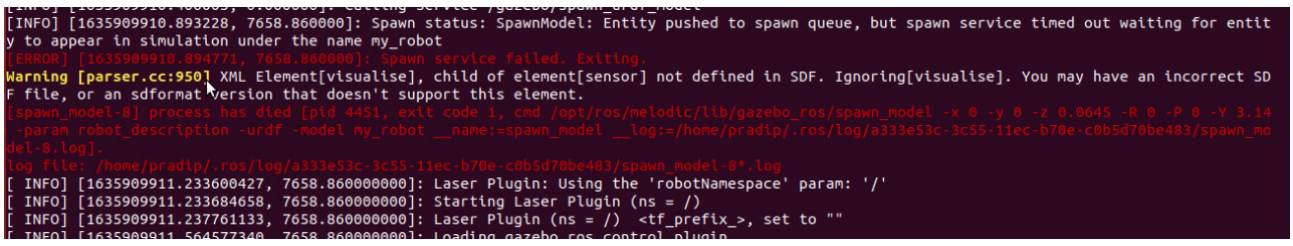
Our project was to model a simple toy car using Solidworks, adding appropiate controllers and used Gazebo to simulate our robot. The entire project was built on Robot Operating System to facilitate easy development of the project

Building a model on Solidworks

- First we made the individual part files for each aspect of our model, i.e.the chassis, axles, the front and rear wheels
- Then all the part files were appropriately assembled in Solidworks and a single assembly file was made. Here we made the origin of the car's model to be the same as that of the Solidworks assembly origin to ensure that the model would launch properly at the origin in Gazebo when launched so as to avoid the problem we faced where the model wasn't launching at the exact origin
- The next step was to export the assembly as a URDF. Ensure that proper parent to child link is followed when making the URDF file, here we should name the each link and joint separetely. And then select appropriate joints for the wheels (revolute for steering and continuous for the joint)
- This URDF file is then placed in the catkin workspace and built using **catkin_make** in the terminal

Adding the Controllers and Laser (LIDAR)

- First we check if the parent-child structure is correct.
- Then we modify the URDF file. Firstly add a dummy link and joint in the URDF file and then add the transmission block for the steering and motion control, we use the EffortJointInterface and VelocityJointInterface respectivelyy
- Then we add the LIDAR URDF and LIDAR DAE file in the URDF and Meshes folders respectively in the package. Then we modify the LIDAR URDF file to to include the LIDAR DAE path
- Then we add a XACRO file and modify it. First we download the MY ROBOT INTEGRATION URDF.XACRO file into our package. And then add the path for the LIDAR DAE file and the Trolley URDF in our XACRO file
- Then we add the LIDAR's sensor data and mention proper co-ordinates to place and orient it on our robot model
- Modify the GAZEBO PLUGIN FOR CONTROL section in the XACRO file and then make link between the robot model and LIDAR in the XACRO file
- Once again we run check on the XACRO file to ensure proper Parent-child structure is maintained between the robot and LIDAR
- Next we add the controllers. Download the controllers's configuration YAML file and place it in our package. Then we add the needed controllers, Effort Controller for steering and Velocity controller for motion and tune the PID gain appropriately
- Then we create a single launch file so as launch all the necessary files at once and spawn the model in the provided world environment to perform the teleoperation
- Here we noticed a single persistent error that the robot failed to spawn in the Gazebo when we do roslaunch. We did consult with the TA to solve this problem and he advised us that it in an inherent problem in ROS and to not worry about it.



Running Teleop

- Once the Robot Model was successfully launched in the environment, we had to code up a publisher script using Python to teleoperate the robot in the environment using certain keys assigned to movement from the keyboard.
- A sample template script was provided to us, we only had to tweak around the file to make it work according to our robot's description.
- In the script, we firstly changed the name of out robot and then changed the topic name for each object for publishing to that exact topic (move, left, right).
- Here, we encountered an issue with different python versions, so instead of directly using rosrun, we used '**sudo chmod +x teleop_template.py**' to declare the publisher file as an executable and in a different terminal, from the catkin_ws, we used **catkin_make** to build the environment and then sourced it with **source devel/setup.bash** and then in a new terminal, navigated to the src directory inside the package and executed the file by running **python3 teleop_template.py**. Thus we could control and teleoperate the robot from the keyboard.

Writing a Publisher and Subscriber Script

- Once we could control the robot using the keyboard, we then give the control back to ROS to make the robot move without providing any input (in any empty world for un-interrupted movement).
- In the Publisher script, we write the code to make the robot move around in a circular path.
- Firstly, we initialize this program as a ROS publisher node with an appropriate name. We define, for each wheel controller, different objects of Publisher class, which publishes to the respective topics associated with it.
- The message it publishes is an arbitriarily chosen constant values to facilitate rotation.
- We put it inside a while loop so it keeps constantly publishing this value. Behind the picture, the robot (controllers at the wheel) in Gazebo subscribes to this topic and follows the command it receives from that message (i.e), rotate in a circular path.
- For the Subscriber script, we subscribe to the same topics the publisher publishes.
- Firstly, we initialize this program as a ROS subscriber node with an appropriate name. We use a callback function which uses the message from the topics and prints on the terminal the message it receives.
- We use rospy.spin() to make sure that the python script doesn't terminate the node and that it keeps printing what ever value it receives in the message onto the screen.

Thus the task for the publisher and subscriber is fulfillled.

My Contribution towards the project: My project partner and I split up the tasks between the two of us in the following way, he would take responsibility of the design aspects and I would take care of the programming aspects. My contribution towards the project dealt with the development of the teleop publisher python script and coding up python scripts for the publisher and subscriber nodes.

Link to the videos:

1. Video of robot driven by teleop + rviz visualization of LIDAR:
https://youtu.be/-QdYIL1R_R4

2. Video of robot driven by the commands from the publisher :
https://youtu.be/4NFgasUSKfg

Instructions to run this project are mentioned in the Readme.txt file included in the ZIP file