ENPM662 - Introduction to Robot Modelling

Project 1 : CAD Modelling & Simulation using Gazebo

Due Date : 24th October 2020

Project Goals

- Build a specific robot model on Solidworks and export it as URDF.
- Add LIDAR Sensor to your robot and show your visualisation on Rviz.
- Perform TeleOP and move around in the map.

Here is small presentation giving the overview of the project for clarity and here is project 1 related resources required.

Submission details

- Solidworks a) part files b) Assembly file.
- URDF export with meshes folder.
- Robot ROS package folder containing all necessary sub folders needed to reproduce on a different system.
- A video showing teleop from center of the map to bottom left or top right of your map.
- A pdf giving an explaining the process of CAD $->$ Simulation in your own words. 1-2 paragraphs should be enough.

1 Build a model on Solidworks

Solidworks is a powerful CAD tool used for modelling, performing design studies, performing simulations etc. Your task is to build a specific robot 3 using solidworks(2019-2020 version preferred).

- **Download and install SolidWorks** from here. It's free for UMD students and please make sure to download the previous version(2019-2020 than 2020-2021). You will need Windows OS for the same
- **Download the URDF exported tool** from here. A brief description of what this is as mentioned on their webpage : The SolidWorks to URDF exporter is a SolidWorks add-in that allows for the convenient export of SW Parts and Assemblies into a URDF file. The exporter will create a ROS-like package that contains a directory for meshes, textures and robots (urdf files). For single SolidWorks parts, the part exporter will pull the material properties and create a single link in the URDF. For assemblies, the exporter will build the links and create a tree based on the SW assembly hierarchy. The exporter can automatically determine the proper joint type, joint transforms, and axes.
- **Build your CAD Model :** Use the specs mentioned in 3 to build a 3D model of the Robot. Build individual CAD models of each part(chassis, wheel etc.) and then assemble them in a single file. You could refer these tutorials made by us. A few more links to videos are here, here and here.
- **Export as URDF :** Use the URDF exporter tool to define parent, child links. The tool automatically determines the axis between joints and sets origins for different links that are defined. Though sometimes the automatic axis generation might not be ideal, we can rotate the coordinate frames while exporting the URDF as well. Export the URDF along with meshes. To help with this process, you could refer this video.

2 Add Controllers and Laser to your Robot

- **Getting Started :** As soon as your done with previous step, make sure to add export the folder into catkin_ws/src in your Ubuntu OS. Explore the directories in the package to get an idea of the structure of the package. Also make sure to checkout the gazebo.launch file under the launch/ dir and try to understand the structure. This is the default launch file that is created by URDF exporter. It calls in several nodes required to launch the robot into your world. Navigate to /catkin_ws and just say $catkin_make. This will build all your packages inside the src directory. After a successful build, make sure to run $ source /catkin_ws/devel/setup.bash. You should be able to **launch the gazebo.launch file from your package**. You will see that your robot appears in an empty world in a gazebo simulation!.
- **Understanding the Lidar file:** URDF files are written in the XML format and have specification such as $< robot >$, $< link >$, $< joint >$, $< transmission >$ etc. To give an idea, a $< robot >$ would consist of $< link >$ $< joint >$ $< transmission >$ etc and each $< link >$ would contain few other properties such as $< visual >$ $< geometry >$ $< collision >$ and so on. You can refer to entire set of specs here. sw2urdf tool that you used, would do most of the work in filling this up for you, but for our project we need to modify the urdf slightly to get it to work. Navigate to URDF folder on your terminal and **run $check_urdf**

my_robot.urdf. This will create a tree like structure showing the parent child relationship that exists in your robot.

- **Modify URDF for extra links and joints:** When the URDF is created, the root link is usually the chassis of the robot. To run our model on gazebo, we need to **create a dummy link** with no properties specified at all except the name of the $< link >$. We also need to **create a dummy joint** with its $< parent >$ as the dummy_link and $< child >$ as the base of robot frame. You can place these two block any where in file but between the robot spec tags and in the same level as other links and joints.

- **Modify URDF for adding transmission:** Refer this link for a sample transmission block. This is present to define how the joint is to be interfaced, either through velocity command or position command. For steering it is preferred to use Effort Controllers and for longitudinal command, it is preferred to use Velocity based interface. **Add as many transmissions blocks as joints**.

- **Adding lidar files:** Download the Lidar's URDF and DAE file from this link and **place them in URDF and Meshes folder** respectively. For this project, we are using the ydlidar which is a 2D lidar and has a FOV of 180^0. All the characteristics of the lidar can be found in this file.

- **Integrating the sensor** Download the .xacro from here. Xacro files are XML macros which helps in easier management of URDF files by making use of Macro like structure in programming. With the use of parameters and generic xacro blocks, one could avoid 100's of lines of repeated URDF code and replace it with just few lines of macro code. Here is a link all about xacros and here to understand xacros better. In our project, we however don't extensively use xacro's features. We just use it to integrate the sensor with the lidar.
  After you've downloaded the .xacro file, **fill in the code blocks after the comments**. To check for correctness of your .xacro file, we create single urdf file of the robot+lidar from the .xacro file. Run $rosrun xacro xacro -o $< output.urdf > < path\_to\_xacro\_file >$. This would create a .urdf file in the same folder as you are. Run $check_urdf your_output_file.urdf to check if parsing is successful. If successful, you should see a tree form of structure of the urdf being printed in the terminal screen. You should now see the our base link of laser attached to the body of your base link of robot. The above check is essential for error free running of our robot.

- **Updating config file:** Our config file is a .yaml file which has its own format of coding which can be found here. We define our list of controllers and type of controllers all within the .yaml file in the config file. Go ahead and download the config_controllers.yaml file. Change my_robot_name with your own robot package's name. After that, go ahead and change controller_1_name to the name of your controller, add the type of your controller, Add the name of joint as mentioned in your URDF file and finally add a PID gains set.

- **Launch file :** A launch file integrates all the sub tasks you've completed till now. Download the template launch file. Also download the world file into a newly created world folder which is where we will be navigating our robot. Here we will include the default gazebo empty world launch file with a an argument to change the world file. We would then import the robot's description we defined earlier using the .xacro file. Most importantly add in the controller names that you defined in the config folder's .yaml file. Here is more info on the controller manager node controller manager. The template launch file has empty code blocks to fill in. Specific

comments have been made to help you fill out the template launch file. After the launch file is filled, navigate to /catkin_ws folder. Run $ catkin_make clean && catkin_make. This will build your package. Don't forget to $source /catkin_ws/devel/setup.bash. Everytime after you build your package.

- Visualise Lidar points in rviz: Run your modified template launch file to see the robot in a world(the downloaded world file) with the lidar on top and visualization of the lidar seen. On rviz, add the laser scan topic and you should be able to visualise the 3D lidar point in rviz. **Capture a video of your lidar points visualisation in rviz.**

3 Run Teleop

- Download the template teleop file into a new src folder of your package and edit the publisher definition to publish onto specific topics. The topics would just be /name_of_robot/controller_name/command The teleop template file has 3 publishers for left, right steering and forward. Add in more publishers and modify the teleop as per your need and depending on the controller setup of your robot.

- Rebuild your package using catkin_make clean && catkin_make. In a new terminal, run $rosrun my_robot_pkg_name teleop_template.py. Use the navigation keys and watch your robot move around. As for task completion and check if controllers have been implemented properly, navigate from center of the map to the bottom left or top right of the map.

Appendix

Notes

- Please do not copy paste the commands. Where there are key words such as my_robot replace them with your robot's name

- For any dependency packages that are missing please try to use google the error first. The solution should most probably appear in the first few links. Try to use $sudo apt-get install $< packagename >$ if applicable.

- If any link to a file is broken, please make sure to check the link in the main page that has the project 1 resources. It's the main folder inside which all the relevant files are present.

- If your robot doesn't move in the direction expected, the sw2urdf tool must have assigned the joint axis in the wrong direction. Please make sure to open the assembly file in solidworks(You should see that the origin, axis of the links and joints in the assembly tree on the left hand side) and update the coordinate system by giving the correct directions to z-axis etc. Export the URDF and only replace the URDF file in your original package.

- When adding the fixed joint between your robot and laser, you might have to play around with xyz and rpy of the joint's origin inorder for the lidar to sit correctly on top of your robot. For some value's it might diorient the robot itself during visualization.

- Few useful Links:

  – URDF vs XACRO

- Running a URDF on gazebo
- Understand xacro file structures
- Example of complicated Xacro file
- Understanding gazebo plugins

Robot Specifications

- Front two wheel need to be steerable. Either through a common link or steered independently.
- Full length of chassis (Longitudinal) - 38 inches.
- Breadth of chassis (Lateral) - 20 inches.
- Wheelbase - 20 inches.
- Wheel thickness - 3 inches.
- Wheel diameter - 8 inches.