Report on ENPM662 Project-1
CAD Modelling & Simulation using GAZEBO

❖ **Procedure**:

1. **Built Solidworks Model and Exported it into URDF file**:

- **Made part files:** Made a part file of each part (Chassis, Axel, Front wheel and Rear wheel) of the model separately using solidworks.
- **Made assembly file:**
  - Assembled all the parts files in solidworks. While making assembly, gave two constraints for each joint: concentric constraint and coincidence constraint.
  - Made the model origin coincide with the solidworks assembly origin by giving coincidence constraint in the assembly using mate feature. This was done to sure that model would be launched at the origin in gazebo.
- **URDF Export:**
  - Clicked on 'tool>Export as URDF'. Selected chassis as a base link. Selected all the remaining link in a proper parents-child structure. Gave a unique name to each link and each joint.
  - Clicked on the preview and export as URDF. In the next window, selected joint type in the joint property window as per model design. Revolute joint for steering and continuous joint for wheel joint.
  - Clicked on the 'Export URDF and Meshes' and gave name to the URDF file as "**trolley**".
  - Copy this URDF files and paste them in the catkin workspace. Built the ROS package in the catkin workspace. Name of the ROS packages is **"trolley".**

2. **Added Controller and Lidar to the Robot**:

- **Checked trolley.urdf file:** Checked URDF file using '**check_urdf**' command to verify the proper parents-child structure.
- **Modified trolley.urdf file:**
  - Added dummy link and dummy joint in the **trolley.urdf** file.
  - Added transmission block for steering and longitudinal motion in the **trolley.urdf** file: provided EffortJointInterface for steering and VelocityJointInterface for longitudinal motion.
- **Added Lidar files:** Added **ydlidar.urdf** and **ydlidar.dae** file in the urdf and meshes folder of the trolley ROS package respectively.
- **Modified ydlidar.urdf file**: Modified the ydlidar.dae file path in the ydlidar.urdf file.
- **Added .xacro file and modified it**:
  - Downloaded **my_robot_integration.urdf.xacro** file and placed it in the ROS package.
  - Added the **path of ydlidar.dae** and **trolley.urdf** in the .xacro file.

- Added **Lidar sensor** data. Set the origin of the Lidar to place Lidar in the correct position and orientation on the robot in the .xacro file.
- modified the given **'gazebo plugin for control'** code in the .xacro file to access the controller from the core ROS packages.
- Wrote a code to create **a joint between robot and Lidar** in the .xacro file.
- Added a **code for providing color** to the robot in the .xacro file.

➢ **Checked .xacro file:** Created a single urdf file (name: **my_trolley.urdf**) of Robot+Lidar and checked the parent-child structure in the Robot+Lidar urdf file is proper or not.

➢ **Added Controller:**
- Downloaded the **config_controllers.yaml** file and placed it in the ROS package.
- Added the appropriate controller (Effort controller for steering and velocity controller for longitudinal motion) and joints name (as per .urdf file) in the .yaml file.
- Tuned the PID gain by hit and trial method.

➢ **Created launch file to navigate in the world given in the problem statement:** Created a launch **template_launch.launch** file to integrate all the .urdf files, .yaml file and .xacro file to launch the robot in the world given in the problem statement to run the teleop.

➢ **Created launch file to navigate in the empty world:** Created a launch **empty_launch.launch** file to integrate all the .urdf files, .yaml file and .xacro file to launch the robot in the empty world to check publisher-subscriber code.

➢ **Robot Launching:** Launched the robot in the empty world and given world to make sure that it is launching properly.

3. **Running Teleop**:

➢ **Added .py file:**
- Added **teleop_template.py** file provided to us in the ROS package.
- Changed the name of the robot and topic name for each object for publishing to that exact topic in this file.
- Declare this file as a ROS executable.
- Launched the robot in the GAZEBO, run this .py file in the terminal and though this file we controlled the robot to guide it to move from GAZEBO centre to the desire destination.

4. Code up a publisher subscriber:

➢ **Coding robot_publisher.py file:**
- Initialized the file as a ROS publisher node.
- Defined the individual objects which publish data to the respective topics. These data(commands) are received at each controller of the wheel.
- Constant values for motion are provided to make the robot move.

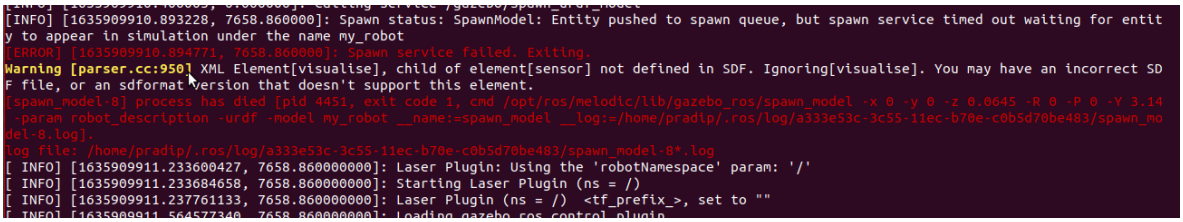
- Launched the robot in the empty world though terminal and run the **robot_publisher.py** file in another terminal and confirmed that the robot is moving circular pattern.

➢ **Coding robot_subscriber.py file:**

- Initialized the file as a ROS publisher node.
- Subscribed to the same topics the publishers(**robot_publisher.py**) publish to, to receive the data.
- Launched the robot in the empty world though terminal, run the **robot_publisher.py** file in another terminal and run the **robot_subscriber.py** in another terminal to check that the messages are being sent by the publisher.
- The data was printed on the terminal, showing the communication between the publisher and subscriber.

❖ Problems faced:

1. **Problem:** Model did not launch at gazebo origin.
   Result: In solidworks assembly file, provided a constraint to match the model origin and solidworks assembly origin.
2. **Problem:**



Result: We tried to solve the above problem for about 2-3 days thinking that it is related to code that we have done. After consulting with TA, we came to know that it is not related to the code rather it is related to inbuilt ROS packages and it would not create any issue in teleop operation.

3. **Problem:** While running the teleop operation, we encountered a problem of different python versions.
   Result: Declare the teleop file teleop_template.py publisher files as an executable which publishes the tele-operating commands. And use **python teleop_template.py** to run the file instead of **rosrun**. Same is done for the publisher and subscriber python files.

❖ My contribution in this Project:

➢ I did step 1 (Built Solidworks Model and Exported it into URDF file:) and step 2 (Added Controller and Lidar to the Robot) mentioned in the procedure above. Step 3 and Step 4 were taken care by my project partner.

❖ Video Links:

- Teleop+Rviz : https://youtu.be/-QdYIL1R_R4
- Publisher+sub : https://youtu.be/4NFgasUSKfg