Name: Naman Gupta Homework – 3 UID: 116949330

Q1. Program a simple robot vehicle in a simulated environment (robot simulation tools and libraries may be used). Your simulated robot should exhibit at least one sensor input (e.g., forward-looking range sensor that returns the distance to the nearest obstacle) and two control outputs (e.g., left and right wheels, or speed and direction of vehicle motion). Show that you can drive your robot around through mouse or keyboard inputs.

Ans: First create a repository on the GitHub and clone it on your PC in $ (ros_workspace)/src. You will see a robot_vehicle folder in src directory. Create a subdirectory and rename it as urdf. Inside urdf folder, create a file and title it as my_robot.xacro.

Using the instructions of building a robot from gazebo website: link, I built my_robot step by step.

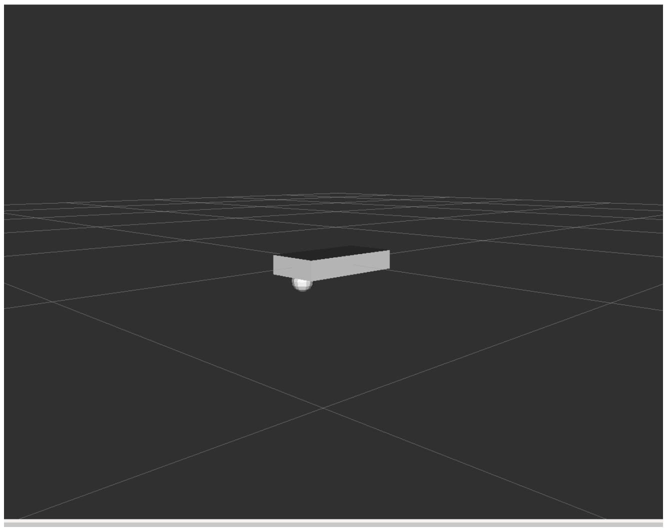


Fig: Chassis and Castor Wheel

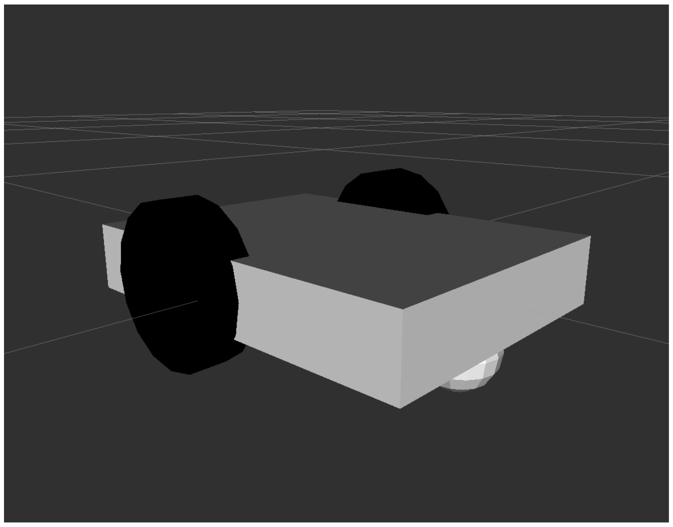


Fig: Wheels Added

First created a rectangle block, then added a caster wheel specifying its moment of inertia value, mass value, etc. Then, added left and right wheel of the robot by creating cylinder. Added rolling joints on the wheel with its moment of inertia value, mass, etc.

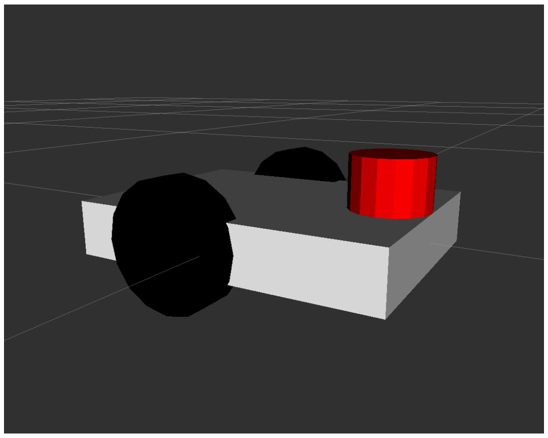
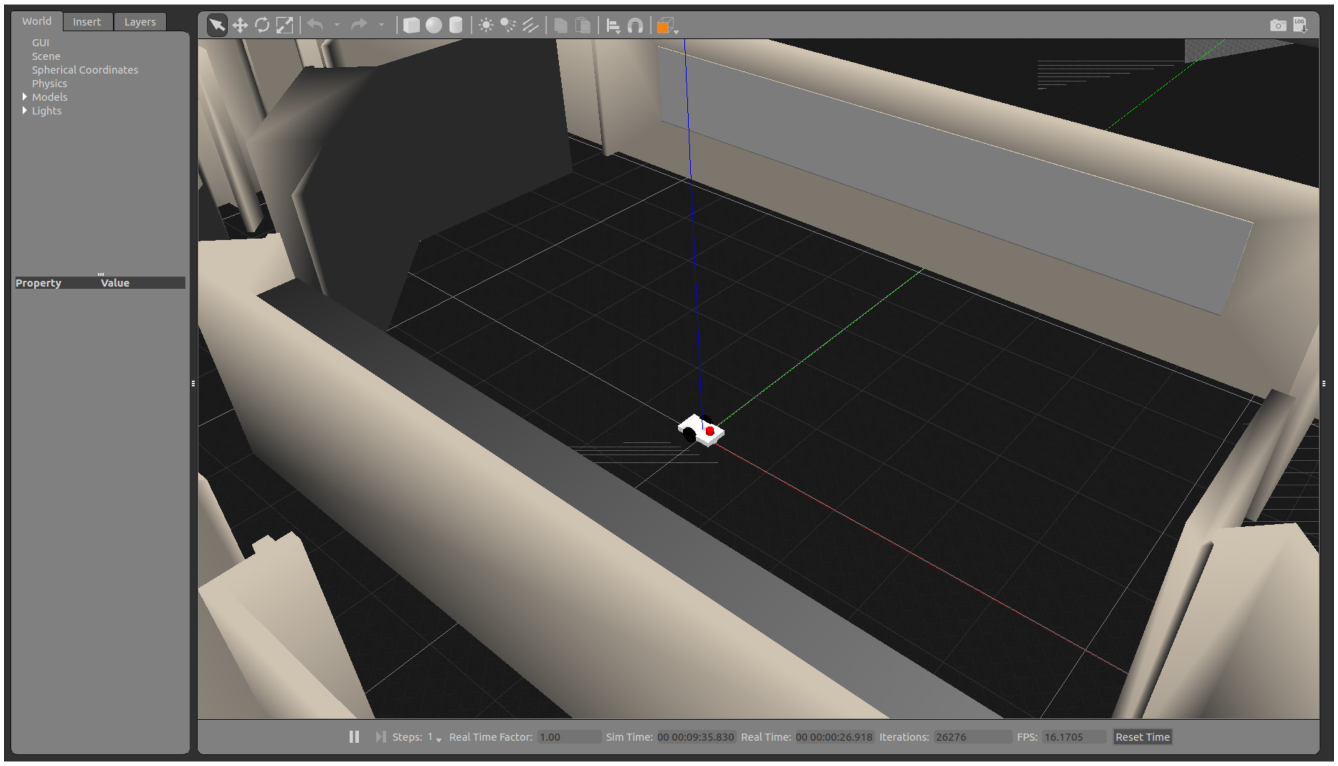


Fig: Final Robot in Rviz

Next, create a laser scanner by adding cylinder and sensor plug in of sensor type **head_hokuyo_sensor** as shown in the above picture.

Next, let’s create a world. Open Gazebo. Import Willowfield_Garage and save it as world.world in the urdf folder. Create a directory in robot_vehicle folder as scripts. In the scripts folder, create a launch file to which combines our robot model and world.world so that when we launch our launch file, the our robot is already there in gazebo in willowfield_garage.



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:09:35.830 Real Time: 00 00:00:26.918 Iterations: 26276 FPS: 16.1705

Fig: Robot in custom world

For controlling robot using keyboard, create a python file and use curses library. My logic is universal, press ‘i’ to move straight, ‘k’ to stop,, ‘j’ to turn around, ‘l’ to turn right, ‘q’ to increase velocity and ‘z’ to reduce velocity.

Create a laser scanner file in the scripts folder, bu subscribing to the topic /laser/scan, we can get the laser scanner data. A simple subscription and ROS info prints the laser scanner data on the console. Follow the instructions as mentioned in the link. Type in the terminal:

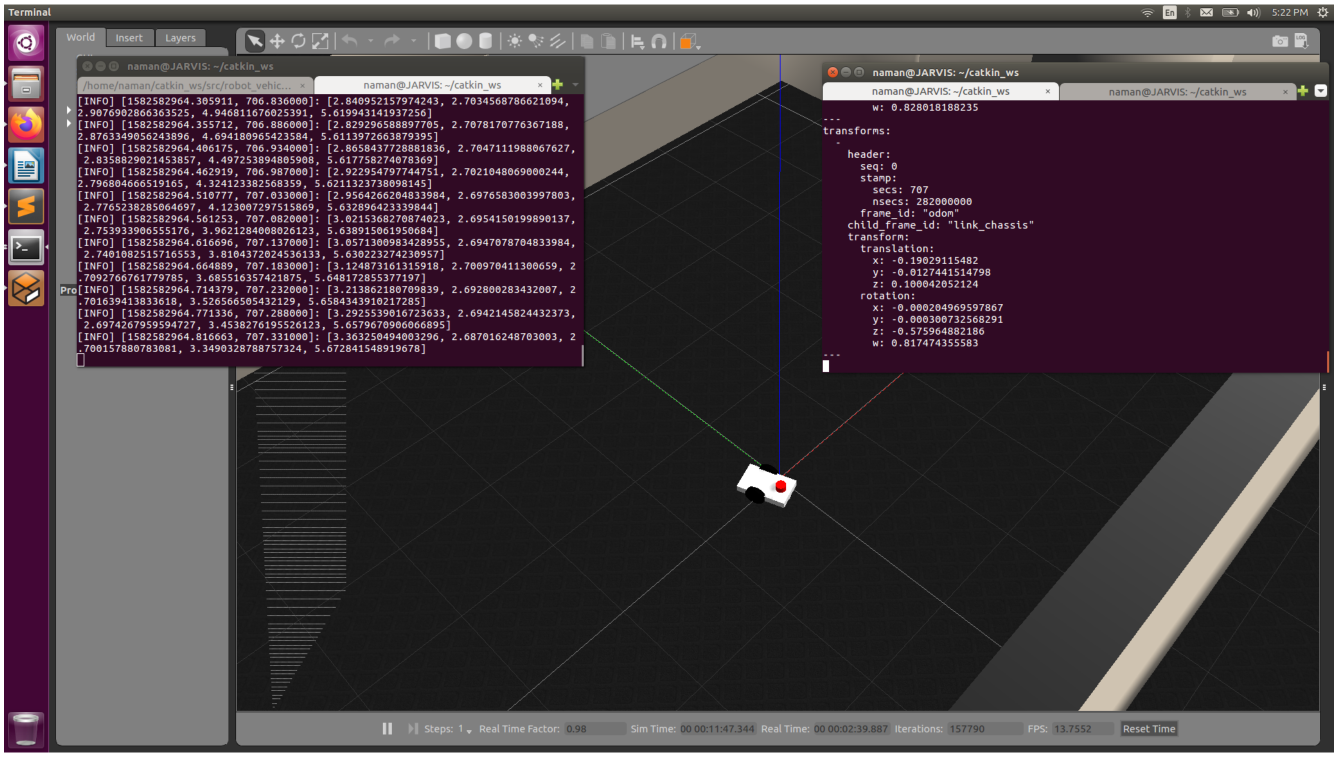
```
$ ~/catkin_ws: roslaunch robot_vehicle spawn.launch
```

In the separate terminal, type:

```
$ ~/catkin_ws: rosrun robot_vehicle read_laser.py
```

This will display the laser data on the terminal.

Name: Naman Gupta Homework – 3 UID: 116949330



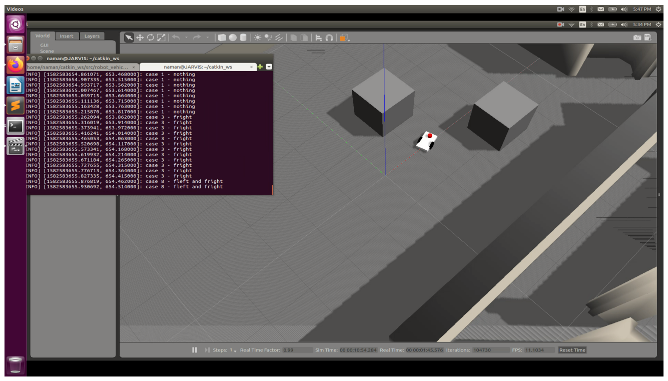
```
[INFO] [1582582964.305911, 706.836000]: [2.840952157974243, 2.7034568786621094,
2.9076902866363525, 4.946811676025391, 5.619943141937256]
[INFO] [1582582964.355712, 706.886000]: [2.829296588897705, 2.7078170776367188,
2.876334905624389 6, 4.694180965423584, 5.611397266387939 5]
[INFO] [1582582964.406175, 706.934000]: [2.8658437728881836, 2.7047111988067627,
 2.8358829021453857, 4.497253894805908, 5.617758274078369]
[INFO] [1582582964.462919, 706.987000]: [2.922954797744751, 2.7021048069000244,
2.796804666519165, 4.324123382568359, 5.621132373809814 5]
[INFO] [1582582964.510777, 707.033000]: [2.9564266204833984, 2.6976583003997803,
 2.7765238285064697, 4.123007297515869, 5.632896423339844]
[INFO] [1582582964.561253, 707.082000]: [3.0215368270874023, 2.6954150199890137,
 2.753933906555176, 3.9621284008026123, 5.638915061950684]
[INFO] [1582582964.616696, 707.137000]: [3.0571300983428955, 2.6947078704833984,
 2.7401082515716553, 3.8104372024536133, 5.630223274230957]
[INFO] [1582582964.664889, 707.183000]: [3.124873161315918, 2.700970411300659, 2
.7092766761779785, 3.685516357421875, 5.648172855377197]
[INFO] [1582582964.714379, 707.232000]: [3.213862180709839, 2.692800283432007, 2
.701639413833618, 3.526566505432129, 5.658434391021728 5]
[INFO] [1582582964.771336, 707.288000]: [3.292553901672363 3, 2.6942145824432373,
 2.6974267959594727, 3.4538276195526123, 5.657967090606689 5]
[INFO] [1582582964.816663, 707.331000]: [3.363250494003296, 2.687016248703003, 2
.700157880783081, 3.3490328788757324, 5.672841548919678]
```

```
        w: 0.828018188235
---
transforms:
  -
    header:
      seq: 0
      stamp:
        secs: 707
        nsecs: 282000000
      frame_id: "odom"
    child_frame_id: "link_chassis"
    transform:
      translation:
        x: -0.19029115482
        y: -0.0127441514798
        z: 0.100042052124
      rotation:
        x: -0.000204969597867
        y: -0.000300732568291
        z: -0.575964882186
        w: 0.817474355583
---
```

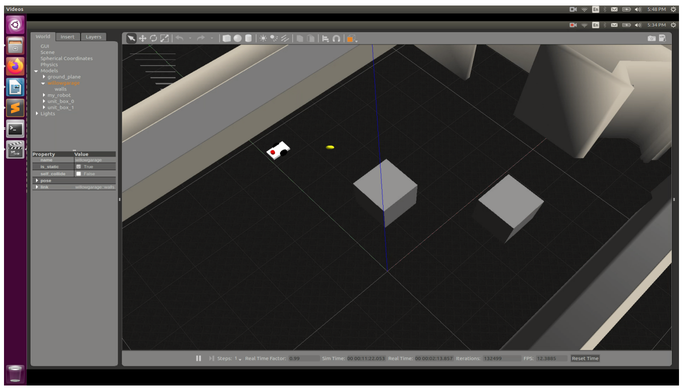
Fig: Laser Scan, Speed and Direction Data

Q2. Add a programmed behavior to your robot, such as following (or avoiding) a light, or wandering, while avoiding collisions with obstacles.

Ans: Let's add obstacles in the world. Add 2-3 blocks in the same world. I have used the /laser/scan to get the scanner data. If the obstacle is closer to 1 meters then the robot will act accordingly. This is how obstacle avoidance is implemented and for wall following the the robot keeps on taking left and move straight. This is the concept I have used for the last question. The video for the same is in the link.



```
[INFO] [1582583654.861071, 653.468000]: case 1 - nothing
[INFO] [1582583654.907335, 653.515000]: case 1 - nothing
[INFO] [1582583654.953717, 653.562000]: case 1 - nothing
[INFO] [1582583655.007467, 653.614000]: case 1 - nothing
[INFO] [1582583655.059715, 653.664000]: case 1 - nothing
[INFO] [1582583655.111136, 653.715000]: case 1 - nothing
[INFO] [1582583655.163428, 653.763000]: case 1 - nothing
[INFO] [1582583655.215870, 653.817000]: case 1 - nothing
[INFO] [1582583655.262094, 653.862000]: case 3 - fright
[INFO] [1582583655.316019, 653.914000]: case 3 - fright
[INFO] [1582583655.373941, 653.972000]: case 3 - fright
[INFO] [1582583655.416241, 654.014000]: case 3 - fright
[INFO] [1582583655.465053, 654.063000]: case 3 - fright
[INFO] [1582583655.520698, 654.117000]: case 3 - fright
[INFO] [1582583655.573341, 654.168000]: case 3 - fright
[INFO] [1582583655.619932, 654.214000]: case 3 - fright
[INFO] [1582583655.671184, 654.265000]: case 3 - fright
[INFO] [1582583655.727655, 654.315000]: case 3 - fright
[INFO] [1582583655.776713, 654.364000]: case 3 - fright
[INFO] [1582583655.827335, 654.415000]: case 3 - fright
[INFO] [1582583655.876819, 654.462000]: case 8 - fleft and fright
[INFO] [1582583655.930692, 654.514000]: case 8 - fleft and fright
```



GitHub Link for the Project: https://github.com/namangupta98/robot_vehicle