Project Title

Autonomous driving sensors' simulation with ROS and Gazebo (with Ubuntu 16.04)

Developed by: MSc. Shiva Agrawal

Place: Germany

Date: October 2018

Content

List of Figures

Abbreviations

ROS	Robot Operating System
URDF	Unified Robot Description Format
XACRO	XML Macro
XML	eXtensible Markup Language
GPS	Global Position System
IMU	Inertial Measurement Unit

1. Introduction

1.1. Project description

ROS is very powerful and efficient way of implementing and designing the Robotics project including self-driving cars. ROS has capability to do both hardware interfacing of different sensors and also to do simulation of the sensors using Gazebo.

For the self-driving vehicle, the very first step is the perception and information extraction as shown in figure below [1].

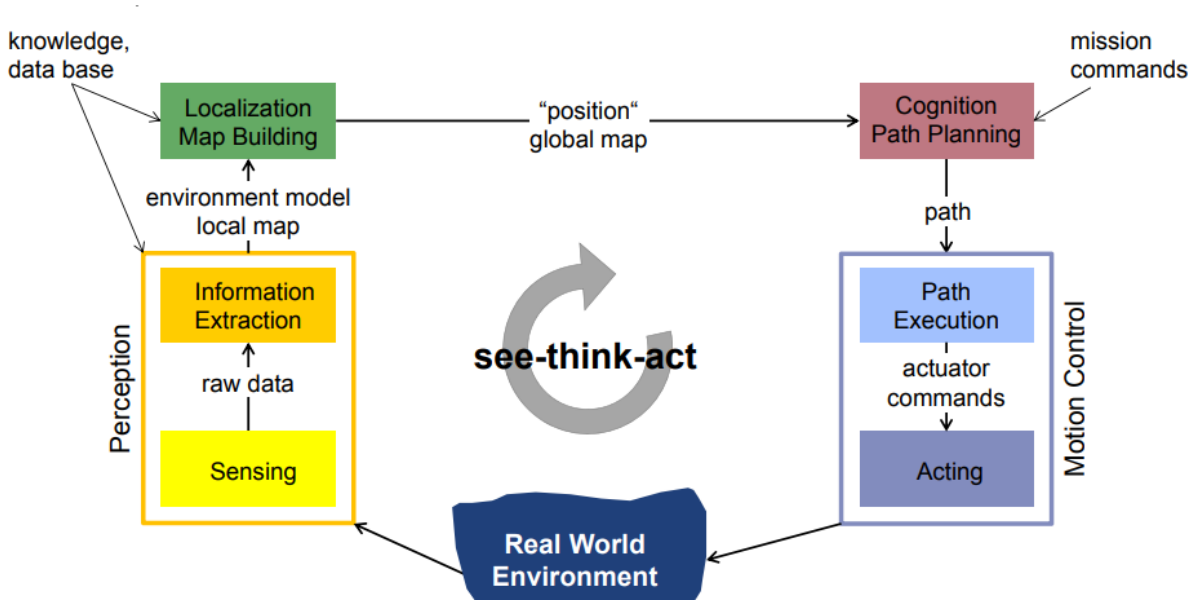

Figure 1 See Think Act cycle | Autonomous mobile robots

Environment perception in case of autonomous driving cars is achieved by following major sensors:

1. Velodyne (360-degree Laser scanner)
2. Radar *(not included in this project)*
3. Mono / stereo camera
4. Ultrasonic sensor

Apart from environment perception, some other sensors are also used to know the location of the vehicle itself (required for localization). This includes

1. GPS
2. IMU (Inertial Measurement Unit)
3. Odometry using wheel encoders *(not included in this project)*

Simulation of the sensors is helpful to learn and understand, how different sensors work and what kind of data is available by each of them. The main aim of this project is to simulate different sensors used for the development of autonomous driving vehicle in ROS and Gazebo and visualize the output data by creating some obstacles in simulated environment.

ROS Kinetic installed on Ubuntu 16.04 is used for the project work. The reference sources used during the development of the project includes ROS wiki [2], ROS Robotics projects book [3] , XACRO wiki [4], URDF wiki [5] and hector Gazebo plugin [6]

1.2. Outline

- Chapter 1: Project description and outline of the project report
- Chapter 2: ROS and related packages
- Chapter 3: Autonomous driving Sensors' simulation in Gazebo
- Chapter 4: Conclusion
- Chapter 5: References

2. ROS and related packages Python Packages

2.1. ROS (Robot Operating System)

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license. The official website of the ROS www.ros.org is full with everything one needs to start with ROS.

Different versions of the ROS are available for the user purpose, but I have used ROS Kinetic.

2.2. ROS package

Software are organized in ROS using packages. Each package then further contains one or more nodes (a process that performs computation), dataset, configuration files, 3rd party software piece, robot model files, launch files, service and parameter files, etc. [7]

Common files and directories inside a package directory are:

- include/package_name: C++ include headers
- msg/: Folder containing Message (msg) types
- src/package_name/: Source files (C++ / Python)
- srv/: Folder containing Service (srv) types
- scripts/: executable scripts
- launch/: contain launch files to execute multiple nodes at a time
- urdf/: contains xacro and urdf files of robot model
- CMakeLists.txt: CMake build file
- package.xml: Package catkin/package.xml

NOTE: detail description of package development is out of scope of this document. Readers are requested to check ROS wiki.

2.3. Gazebo

Gazebo offers the ability to accurately and efficiently simulate variety of robots in complex indoor and outdoor environments. It provides a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant community [8].

2.4. URDF

It is abbreviation for Unified Robot Description Format. This is XML based format with predefined tags which can be used to create 3D model. It contains tags to create joints, links, colors, meshes, etc. for robot. We can also develop sensors and working environment using

URDF files. ROS metapackage 'robot_model' contains various packages to design and create the 3D models of robots and sensors. Urdf is one of the packages of this meta package.

2.5. XACRO

It is abbreviation for XML Macros. The URDF representation for 3D model development is good but it has limitations of not able to define variables, equations, functions and other macros in it. Hence when the robot model is complex, URDF limits the user flexibility. These limitations of URDF are overcome by XACRO files. These are also sort of XML type files with different tag representations.

When the model is created using XACRO then there are two ways to visualization and use it in Gazebo and other visualization and/or simulation tools.

1. Directly launch the xacro file
2. Converting xacro file to urdf file and then launch it

2.6. Hector_gazebo_plugin

[6] hector_gazebo_plugins provides gazebo plugins from Team Hector. It is developed by team at TU Darmstadt, Germany. It contains plugins to simulate GPS, IMU, sonar sensor. It is used to simulate these sensors in this project

3. Autonomous driving Sensors' simulation in Gazebo

Following sensors are simulated in this project

1. Velodyne (HDL-32E and VLP-16)
2. Mono camera
3. Stereo camera
4. GPS
5. IMU
6. Ultrasonic sensor

The simulation environment is Gazebo installed with ROS kinetic in Ubuntu 16.04

3.1. Velodyne (HDL-32E and VLP-16)

The velodyne simulator can be installed directly as

$ sudo apt-get install ros-kinetic-velodyne-simulator

Then the simulation can be launched as

$ roslaunch velodyne_description example.launch

The sensors with obstacles in Gazebo are shown below

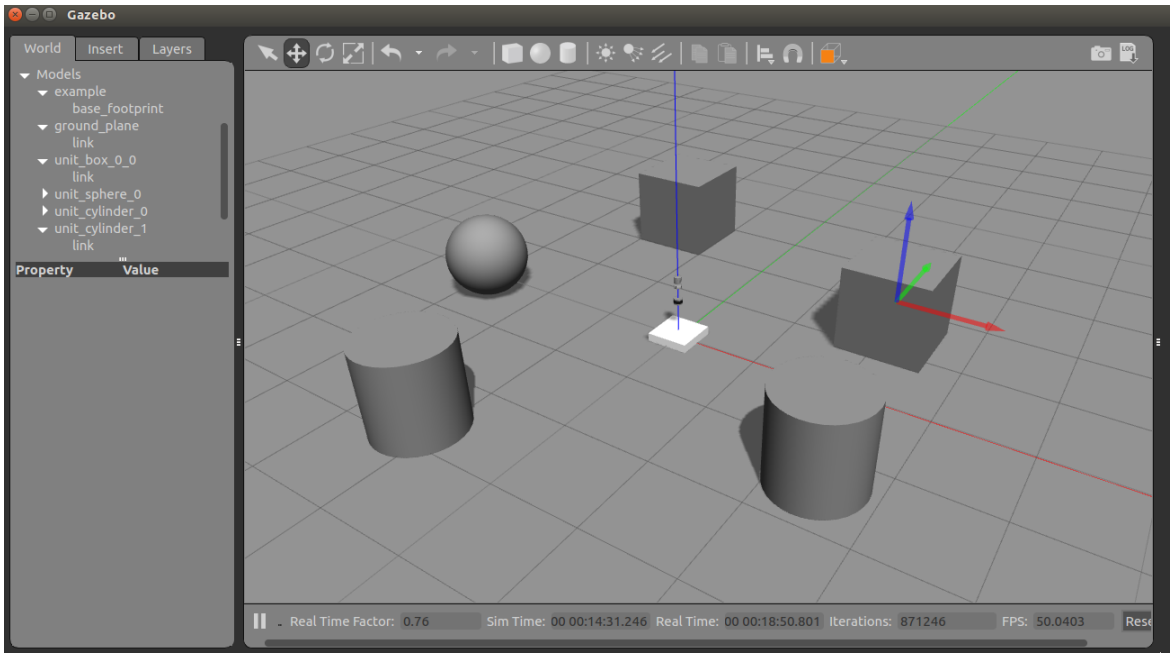

Figure 2 - Velodyne with obstacles in Gazebo

The sensor data of VLP-16 in Rviz is shown below in fig 3. The point cloud is seen in the image and the obstacles around the sensor are clearly visible in it.

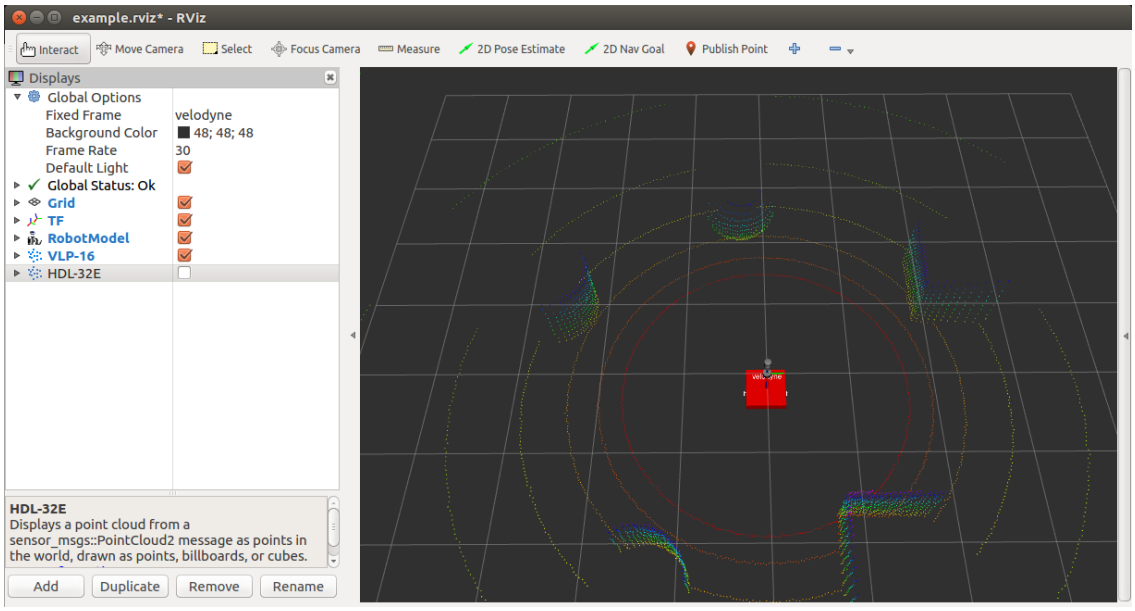

Figure 3 - Velodyne VLP-16 point cloud data in Rviz

Similarly, the data from velodyne HDL-32E in Rviz is shown in fig 4 for the same obstacles.

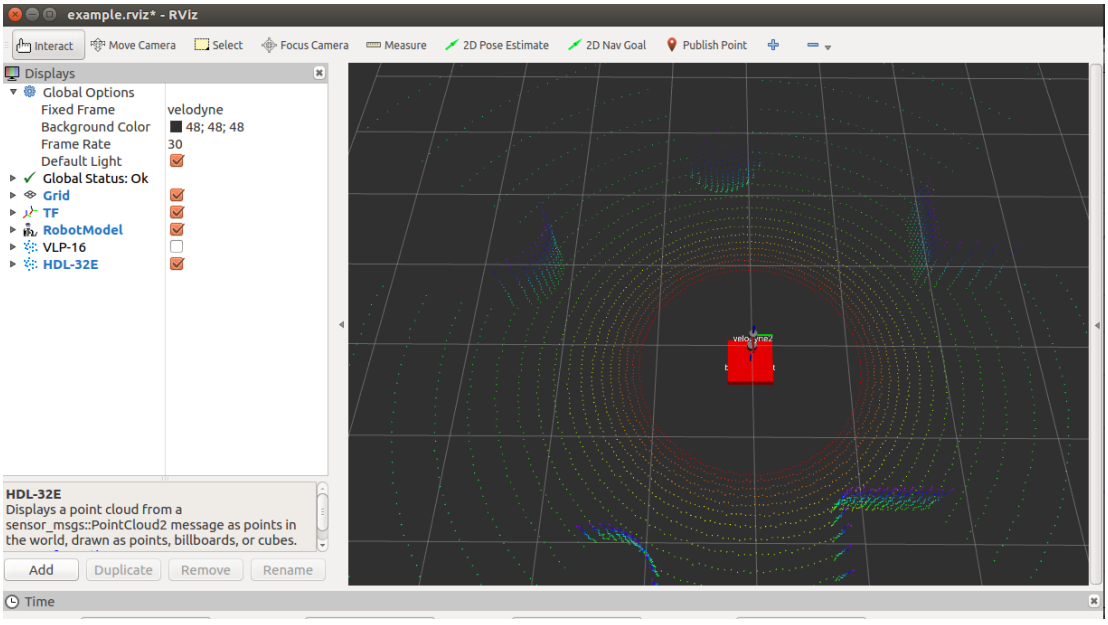

Figure 4 - Velodyne HDL-32E point cloud data in Rviz

For remaining other sensors, the ROS package is created which contains the corresponding xacro files of the sensor models and their launch files to do simulation with Gazebo.

NOTE: Please not that the complete package with all the code is available in the src folder of the repository but this code is not completely written by myself. As the motto of the project is to learn the simulation, the complete sensor model development process is kept out of the scope of this project. But the codes are written using references and books and are well tested.

3.2. Mono camera

The mono camera visualization is shown in below in fig 5 and the small window is showing the camera view. This window can be run using image_view node from image_view package.

$ rosrun image_view image_view image:=/sensor/camera1/image_raw

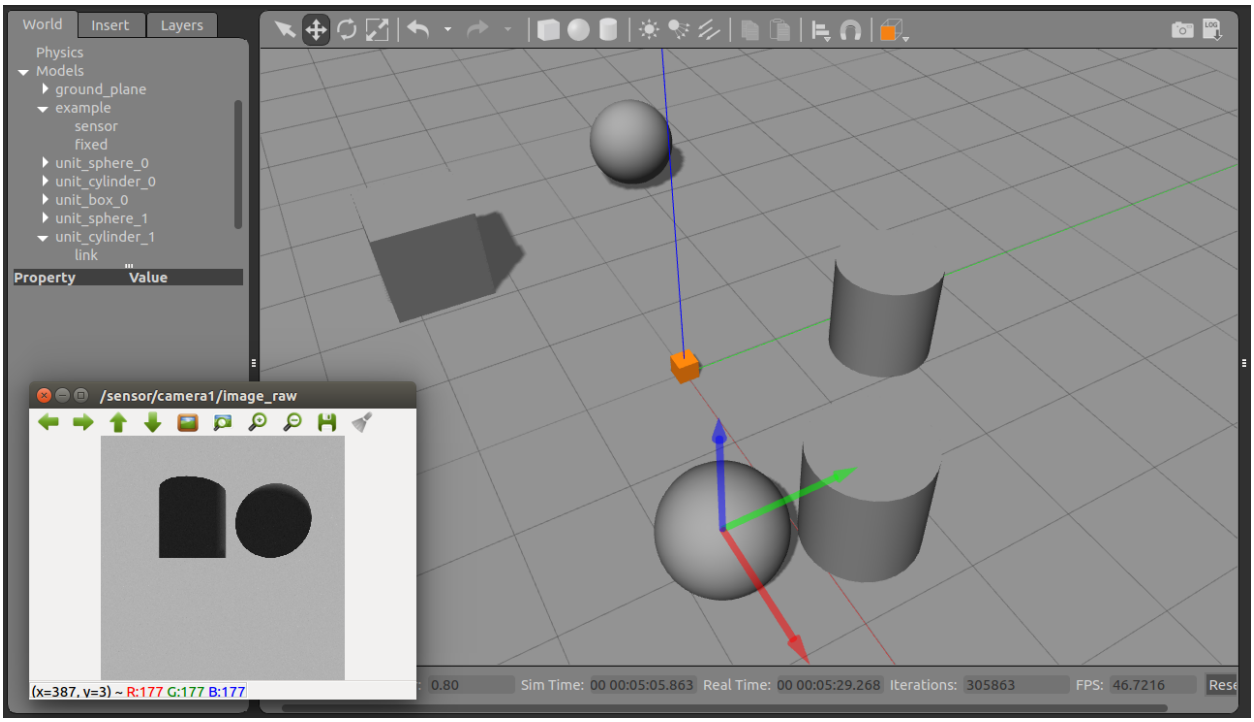

Figure 5 - Mono camera in Gazebo

3.3. Stereo camera

As stereo camera is pair of two cameras where the same image is viewed with slightly different angles, it is very useful to create the 3D information from them. This is possible by using the disparity between the two images.

Fig 6 below shows the stereo camera view in Gazebo and using image_view node. The camera view can be run as:

$ rosrun image_view image_view image:=/stereo/camera/right/image_raw

$ rosrun image_view image_view image:=/stereo/camera/left/image_raw

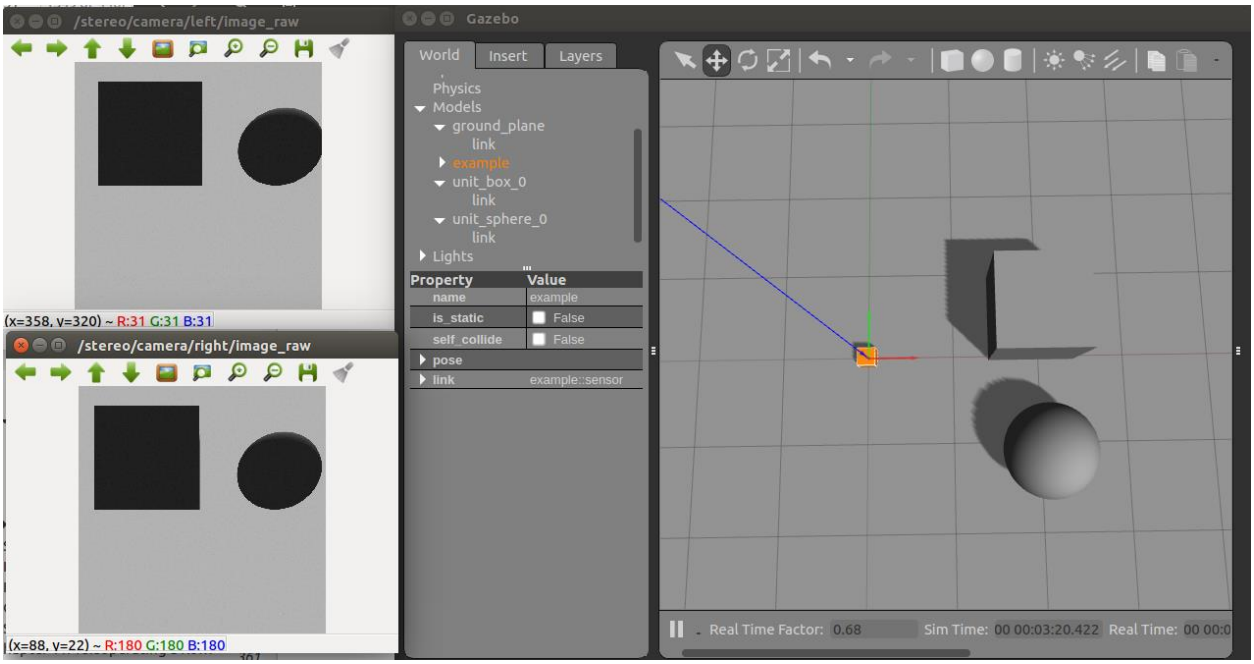

Figure 6 - Stereo camera in Gazebo

As shown above, the two images from left and right cameras have slightly different view. The square and circle (2D projections of cuboid and sphere) are slightly shifted to right as compared to below image. This difference is disparity which is used to then extract 3D information by using both the images of cameras taken at same time.

3.4. GPS

This is used to find the location of the vehicle itself. But since the accuracy of the GPS is not very high, it is generally used together with IMU (Inertial Measurement Unit).

Fig 7 shows the GPS sensor(orange) in Gazebo and fig 8 shows the list of all the topics providing GPS measurement data. The complete information is available in the topic **/gps/fix.** This information is further shown in fig 9 for one-time instance.

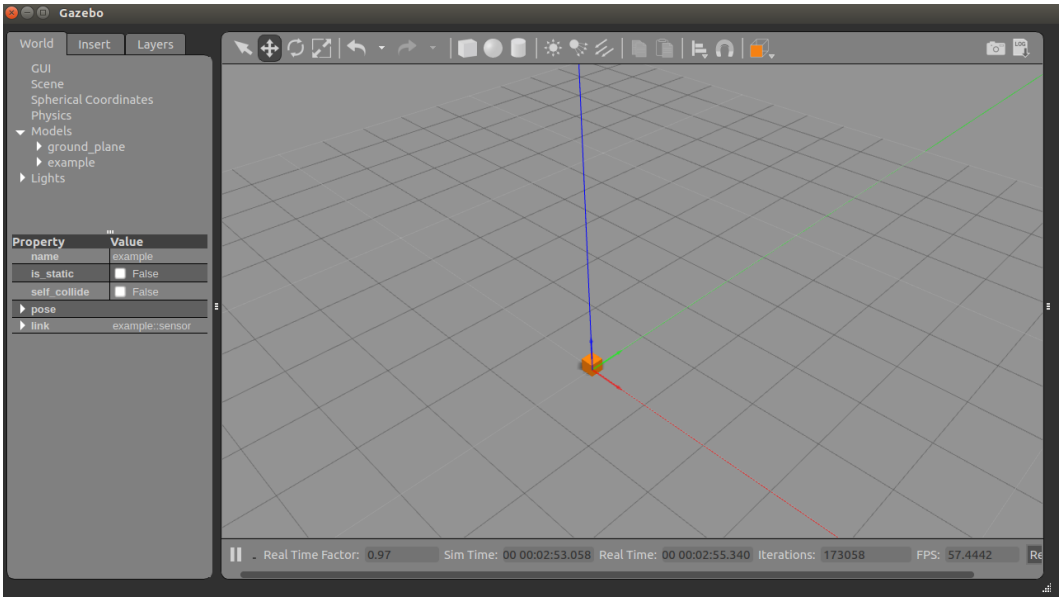

Figure 7 - GPS sensor in Gazebo

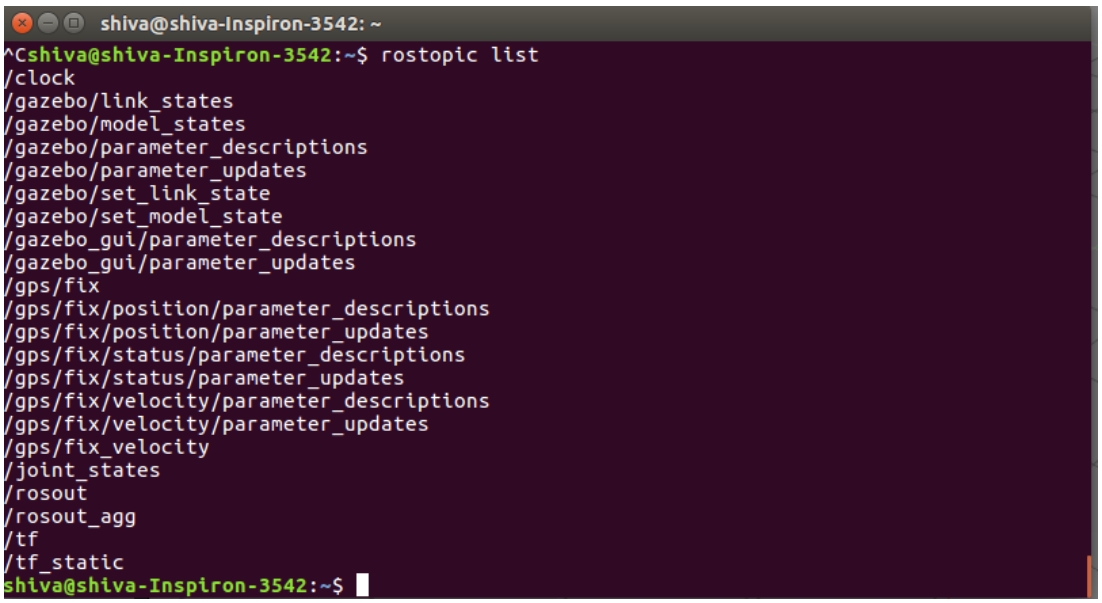

Figure 8 - GPS topics list

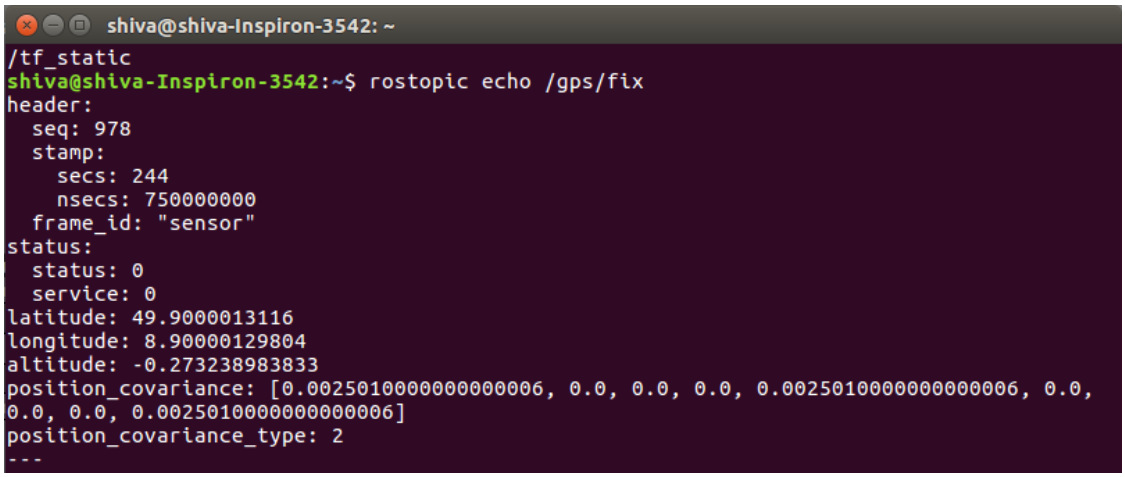

Figure 9 - GPS data of one measurement

3.5. IMU

Once the IMU model is loaded in Gazebo, the list of topics running can be seen as shown in fig 10. Here The topic **/imu** contains the complete measurement data from IMU sensors.

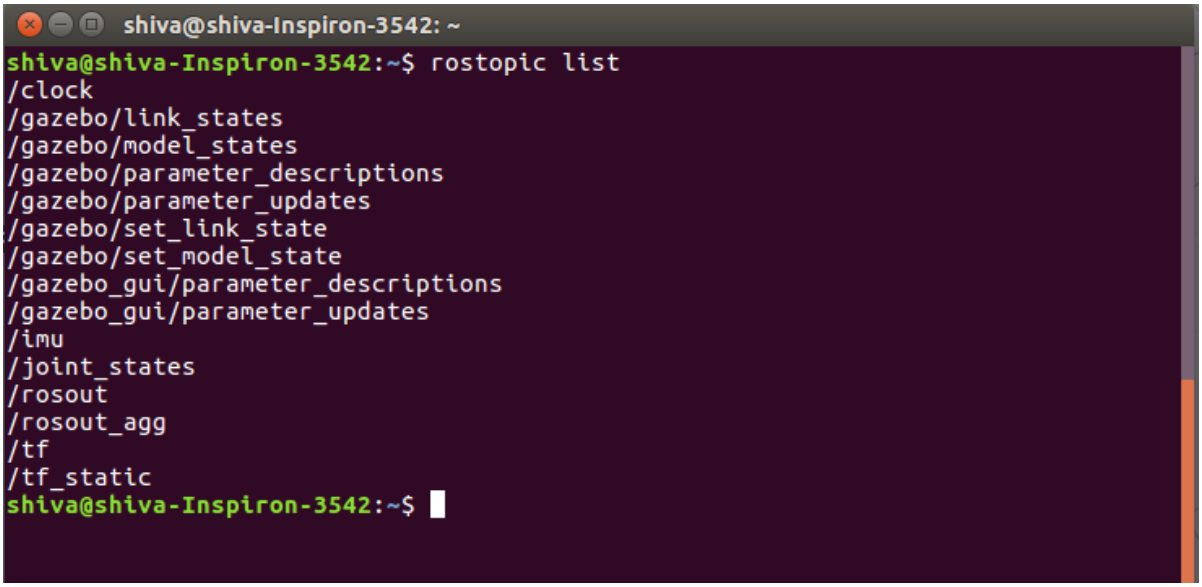

Figure 10 - imu topic

This **/imu** topic is then echoed to see the measurement data as shown in fig 11. For one measurement cycle. It contains the data as orientation along x, y, z axes, angular velocities along 3 axes and linear acceleration along 3 axes.

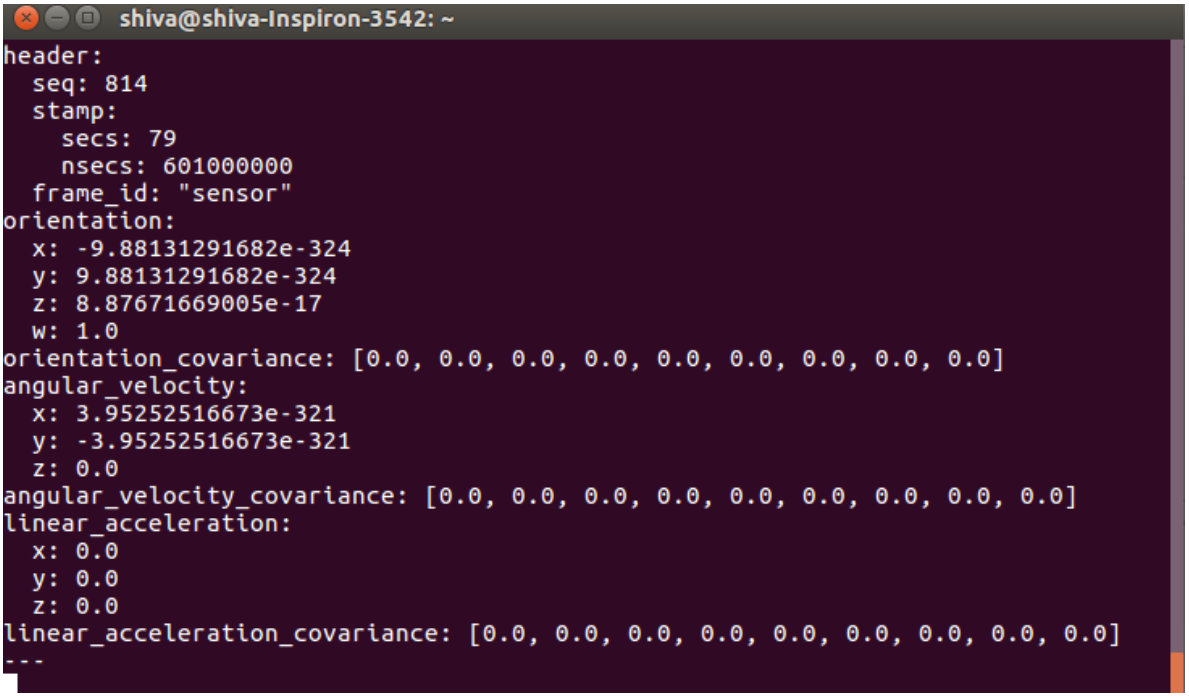

Figure 11 - IMU measurement data

3.6. Ultrasonic sensor

This is range measurement sensor use for nearby obstacle detections in self-driving vehicles. Fig 12 and Fig 13 shows the two measurements taken for different distances by keeping a square block in front of sensor in gazebo environment. Please note that the ultrasonic sensor (right side) in black color is very small and hence it is highlighted using red eclipse in the figures. The corresponding measurement is shown in terminal on left side (see range value at bottom left)

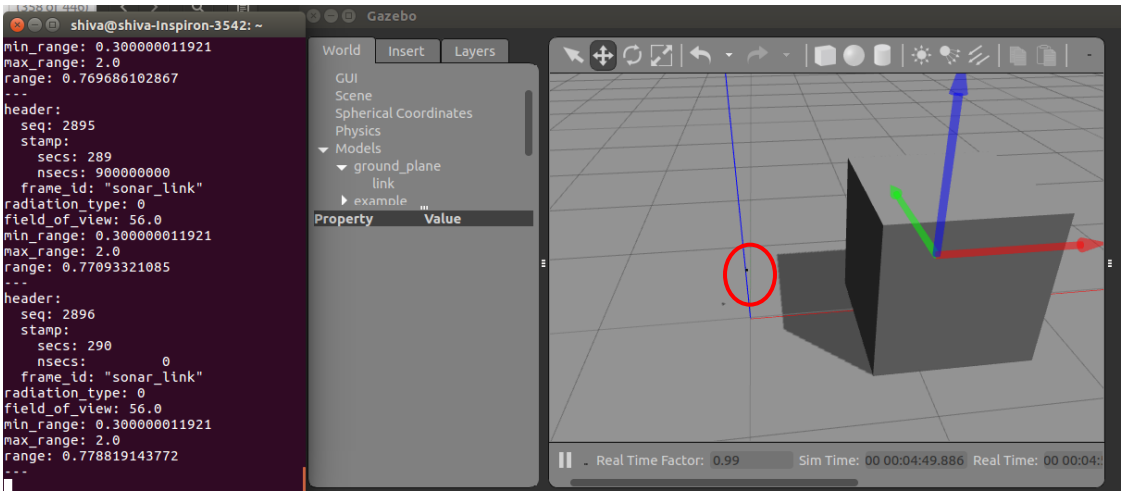

Figure 12 - Ultrasonic sensor distance measurement 01

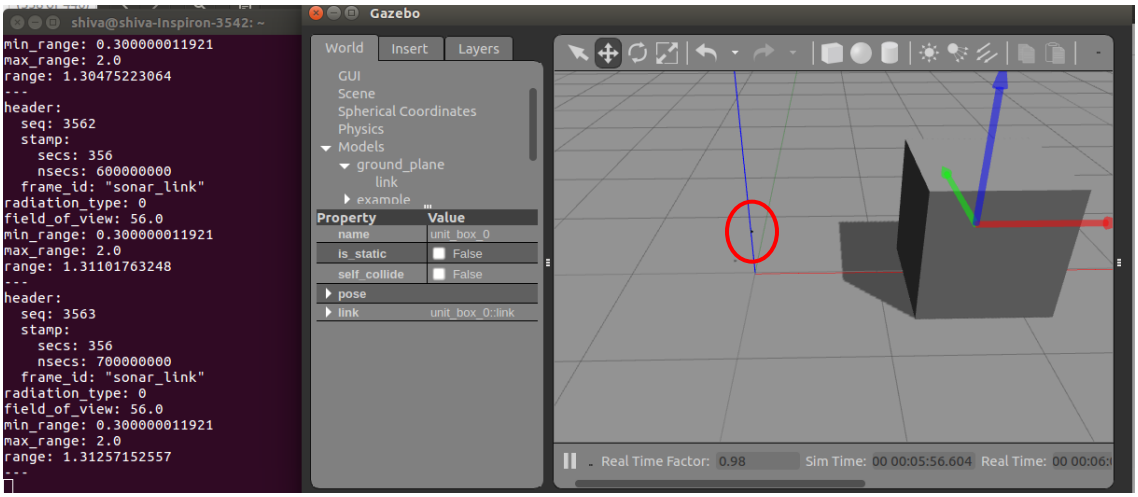

Figure 13 - Ultrasonic sensor distance measurement 02

All the images used in this section are also available in **test** folder of the project repository. Moreover, this project report is available in **doc** folder of the project repository for reference.

4. Conclusion

As the simulation of different sensors used in self-driving cars is very efficient approach to learn, understand and get insight of data, the aim of the project is hence achieved.

5. References

[1] R. Siegwart, Introduction to Autonomous Mobile Robots 2nd edition.

[2] "ROS," ROS, [Online]. Available: http://wiki.ros.org/.

[3] L. Joseph, ROS Robotics projects, Packt.

[4] "xacro," ROS open source, [Online]. Available: http://wiki.ros.org/xacro.

[5] "URDF," ROS open source, [Online]. Available: http://wiki.ros.org/urdf.

[6] "hector gazebo plugin," open source robotics foundation, [Online]. Available: http://wiki.ros.org/hector_gazebo_plugins.

[7] "packages," ROS open source, [Online]. Available: http://wiki.ros.org/Packages.

[8] "gazebosim," Open Source Robotics Foundation, [Online]. Available: http://gazebosim.org/.