Software Lab 2019/2020

Co-Simulating Autoware with Simulink – Software Installation Report

Autoware: A Brief Introduction and an Installation Guide

Supervised By:

Christoph Hahn

Steve Schaefer

Authors:

Alperen Kıral

Daniyal Ahmed

Phuris Khunphakdee

19.10.2019

Contents

List of Figures

List of Tables

Chapter 1: Introduction to Autoware

Autoware is a Robotic Operating System or ROS based software running on Ubuntu. It is widely used in autonomous vehicle development. The graphical interface of Autoware is developed by using an open-source widget toolkit called Qt. Moreover, CUDA, the parallel computing platform developed by Nvidia can also be used to improve the efficiency of data processing. ROS and Qt is required for Autoware, but CUDA is an optional.

The compatible versions of Autoware and Ubuntu the other supporting software can be seen in the table below [1].

Table 1: Compatible versions of Autoware and Ubuntu

Autoware Version	**Ubuntu 14.04**	**Ubuntu 16.04**	**Ubuntu 18.04**
v1.12.0		✓	✓
v1.11.1		✓	
v1.11.0		✓	
v1.10.0		✓	
v1.9.1	✓	✓	
v1.9.0	✓	✓	

Table 2: Compatible versions of Autoware and other supporting software

Product	**Ubuntu 14.04**	**Ubuntu 16.04**	**Ubuntu 18.04**
ROS	Indigo	Kinetic	Melodic
Qt	4.8.6 or higher	5.2.1 or higher	5.9.5 or higher
CUDA *(optional)*	8.0	9.0	10.0

In our case, we first use Autoware v1.10.0 along with Ubuntu 16.04 because the newer versions of Autoware require more RAM-consuming way of workspace compiling and our computer before having the computer support from MATLAB has limited resources. But after receiving the desired computer support, we have changed to Autoware V1.12.0. The recommended specification to run Autoware on the computer that we first used as well as the other software version can be summarized as in the table below.

Table 3: Recommended specifications for Autoware

CPU	Intel Core i7 (preferred), Core i5, Atom
GPU	NVIDIA GTX GeForce GPU (980M or higher performance)
RAM	16GB to 32GB
Storage	More than 30GB of SSD

Table 4: Computer specifications at disposal

Laptop Model	Lenovo Y5070
CPU	Intel Core i7-4720HQ
GPU	NVIDIA GeForce GTX 960M (2GB GDDR5)
RAM	12 GB DDR3L
Storage	HDD 1 TB 5400 RPM

Table 5: Configurations that were used

Product	**Version**
Ubuntu	16.04 LTS
ROS	Kinetic
Qt	5.5.1
Autoware	V1.10.0 and V1.12.0

Chapter 2: Installation of Autoware

2.1 Installation of Ubuntu 16.04 LTS

For Ubuntu installation only, please refer to the link provided below:

- https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop-1604#0

For Ubuntu installation alongside Windows 10 (Dualboot), please refer to the following link:

- https://vitux.com/how-to-install-ubuntu-18-04-along-with-windows-10/

Please, note the following points:

- The installation of Ubuntu 16.04 is similar to the installation of 18.04.
- During an installation, you will have to choose the location and the keyboard layout. Please, choose the native English speaking city (e.g. New York) and English (US) as the keyboard layout to prevent the transformation of .(dot) and ,(comma).Otherwise, this can lead to unsuccessful installation of Autoware

2.2 Installation of ROS Kinetic

For ROS Kinetic installation, please refer to the link provided below:

- http://wiki.ros.org/kinetic/Installation/Ubuntu?fbclid=IwAR2hUY66jx58g_zDdP8oD3JppSa29NJ53_FIPojZnO5WQC_o6CJdEqBA8hU

Below are listed the necessary commands to install ROS Kinetic successfully on Ubuntu 16.04:

```
➢ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
➢ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80'
  --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
➢ sudo apt-get update
➢ sudo apt-get install ros-kinetic-desktop-full
➢ sudo rosdep init
➢ rosdep update
➢ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
➢ source ~/.bashrc
➢ sudo apt install python-rosinstall python-rosinstall-
  generator python-wstool build-essential
```

Please note that, in the fourth command, it was recommended to install every module of ROS Kinetic to be on the safe side.

In order to check if ROS Kinetic is installed correctly or not, any command in the following link can be tested:

- http://wiki.ros.org/ROS/Tutorials

2.3 Installation of Qt

The installation guide for Qt is mentioned in detail in the link that follows:

- https://www.lucidar.me/en/dev-c-cpp/how-to-install-qt-creator-on-ubuntu-16-04/?fbclid=IwAR1ZPUmVp6YmQuZuuaqunz7anIR8nP5hzwdfSpX9Lmt0L_qg9lN2up4glo8

The commands below can be used to install Qt creator. There is no need for confirmation of installation in this part.

```
➢ sudo apt-get install build-essential
➢ sudo apt-get install qtcreator
➢ sudo apt-get install qt5-default
```

2.4 Installation of Autoware v1.12.0 with a Demo Run

In order to install Autoware, it was recommended by us to use *Source Build* which is much simpler compared to the second method which is called *Docker Build*. However, it should be noted that Autoware Foundation suggests users to install Autoware using *Docker Build* since NVIDIA tools are utilized during Docker method.

The following link provides a detailed guide on installation of Autoware:

- https://gitlab.com/autowarefoundation/autoware.ai/autoware/wikis/Source-Build

Installation of Autoware v1.12.0 using *Source Build* method is explained below. Firstly, some dependencies that are prerequisite for the installation of Autoware needs be build:

```
➢ sudo apt-get update
➢ sudo apt-get install -y python-catkin-pkg python-rosdep ros-
  $ROS_DISTRO-catkin gksu
➢ sudo apt-get install -y python3-pip python3-colcon-common-
  extensions python3-setuptools python3-vcstool
➢ pip3 install -U setuptools
```

After finishing the above steps, Autoware can be downloaded and built using following commands:

```
➢ mkdir -p autoware.ai/src
➢ cd autoware.ai
➢ wget                -O               autoware.ai.repos
  https://gitlab.com/autowarefoundation/autoware.ai/autoware/
  raw/1.12.0/autoware.ai.repos?inline=false
➢ vcs import src < autoware.ai.repos
➢ rosdep update
➢ rosdep install -y --from-paths src --ignore-src --rosdistro
  $ROS_DISTRO
➢ Colcon build --cmake-args -DCMAKE_BUILD_TYPE=Release
```

After the installation is complete, we run the demo to check if Autoware is installed correctly or not.

The ROSBAG files necessary for demo running as well as the video tutorial can be downloaded and installed from the links that follow respectively:

- https://gitlab.com/autowarefoundation/autoware.ai/autoware/wikis/ROSBAG-Demo
- https://www.youtube.com/watch?v=OWwtr_71cqI&t=252s

Chapter 3: MATLAB and Simulink

MATLAB is a programming platform which has its own matrix-based language or MATLAB language. This software is very popular in the engineering field. Simulink is a graphical programming environment based on MATLAB. Simulink's user draws a block diagram which represents a certain system. The system then can be simulated, and the users can modify any parameters to observe the result or to optimize or tune the system. This is very popular in engineering control. MATLAB and Simulink are available on a variety of operating systems.

With Simulink and Model-Based Design, a user can model and simulate the system under test. A user can avoid expensive costs involved in prototyping by testing the system under real-life conditions which are otherwise too risky or time-consuming to consider; hence, a user can test early and often. Furthermore, a user can automatically generate code without having to write C, C++, or HDL code. For further information on how to design and simulate your system in Simulink before moving to hardware, please refer to the link that follows:

- https://www.mathworks.com/products/simulink.html

Because MATLAB can create ROS nodes and topics which can be connected to an existing ROS Network (e.g. Network of Autoware nodes and topics). Therefore, it is not necessary to install MATLAB on the same computer as Autoware. However, it is more convenient to work if both Autoware and MATLAB are installed on the same computer in case the number of computers is limited, and the network is not reliable.

Robotic System Toolbox
In order to create ROS nodes and topics, an additional toolbox called *Robotic System Toolbox* provided by MATLAB must be installed together with MATLAB and Simulink.

Autoware Toolbox
There are sample MATLAB/Simulink codes which have been developed to use with Autoware. These codes are provided in *Autoware Toolbox* [4]. We also used these codes to see how the connection between MATLAB and Autoware can be established. This toolbox also requires additional MATLAB's toolboxes, and it defines the release of MATLAB to be installed.

The version of each product can be checked by entering the following command in the command window:

```
>> ver
```

The table that follows shows the release of MATLAB and toolboxes that are installed on our computer.

Table 6: MathWorks Products and their Releases

Product	**Release**
MATLAB	R2018b
Simulink	R2018b
Robotic System Toolbox	R2018b
Computer Vision System Toolbox	R2018b
Image Processing Toolbox	R2018b

For installation of MathWorks product on Windows, Ubuntu, Mac OS, please refer to the following link:

- https://www.mathworks.com/help/install/ug/install-mathworks-software.html

Chapter 4: Autoware Toolbox

This section covers some details about the sample MATLAB/Simulink codes that are developed to be used in conjunction with Autoware [4]. These codes are available on GitHub under the name 'Autoware Toolbox'. Please refer to the following link to access Autoware Toolbox:

- https://github.com/CPFL/Autoware_Toolbox.

For this project, we have made use of this Autoware Toolbox to see how the connection between MATLAB and Autoware is established. Autoware, a very popular open-source software, provides a complete set of self-driving modules. There are three principle functionalities of Autoware: sensing, computing, and actuating. Autonomous vehicles as cyber-physical systems can be abstracted into sensing, computing, and actuation.

The main requirements to use Autoware Toolbox are the pre-installation of Autoware, and additional toolboxes in MATLAB/Simulink such as Robotic System Toolbox, Computer Vision System Toolbox *(optional)*, Image Processing Toolbox *(optional)*. The release of MATLAB to be used is R2018b.

To prepare and install Autoware Toolbox for use with MATLAB, please refer to the following link for detailed installation guide:

- https://github.com/CPFL/Autoware_Toolbox/blob/master/docs/en/install_awtb_en.md

In addition to this, one needs to have read and write permissions for MATLAB folder. On Linux, these can be granted by using the following command:

```
$ sudo chmod -R a+rwx /path/to/matlab/folder
```

To prepare and install *Robotics System Toolbox*, please refer to the following link for detailed instructions:

- https://www.mathworks.com/help/releases/R2018a/robotics/ug/install-robotics-system-toolbox-support-packages.html

There are various modules in Autoware Toolbox such as detection module (consisting of ACF Detector node, LiDAR Euclidean Track node, and Vision Dummy Track node), Localization module (consisting of Vel pose connect node), Mission Planning module (consisting of Lane stop node and Lane Rule node), and Motion planning module (which includes Path Select node, Pure pursuit node, Twist filter node and WF simulator). In addition to these modules, there are various filters included, such as Voxel Grid Filter, Random Filter, Non-Uniform Voxel Grid Filter, and Fog Rectification filter.

4.1 A Short Description of the Functionalities of Autoware

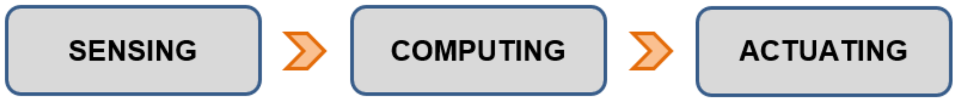


Figure 1: Three Principle Functionalities of Autoware

4.1.1 Sensing Module

Autoware mainly recognizes road environment with the help of sensing devices such as LiDARs, scanners, and cameras. This obtained data is then preprocessed by the sensing module. This data is in its raw form still which is then down-sampled with the help of filters such as point cloud filter and image filter. Point cloud filter nodes down-sample raw point cloud data as obtained from LiDARs, whereas Image filter filters the data obtained from cameras.

Point Cloud Library is mainly used to manage LiDAR scans and 3D mapping data, in addition to performing data filtering and visualization to localize, map, and detect objects in the point cloud (Euclidean distance between points). Therefore, Autoware takes advantage of this result to get an accurate position between 3D maps and the LiDAR scanner mounted on the vehicle.

4.1.2. Computing Module

Computing is an important functionality of Autoware, and it is an essential module for autonomous vehicles. Autoware can compute the final trajectory and communicate with the actuation module by using the sensor data obtained from sensing module and 3D Maps.

The computing module consists of localization (which exploits scan matching between 3D maps and LiDAR scanners to create 3D maps), detection (which includes deep learning and machine learning techniques to detect objects in the surrounding such as other vehicles, pedestrians, cyclists, and traffic signals), and mission and motion planning. In Autoware, Kalman filters and particle filters are employed to estimate the trajectory of other moving objects, and to determine the direction in which the vehicle should move.

4.1.3 Actuation Module

This module consists of vehicle control (which contains filtered actuation commands to control the vehicle) and path following (which includes pure pursuit algorithm to generate actuation commands for the vehicle to avoid dangerous acceleration, declaration or sudden change in the angle of steering wheel).

4.2 Voxel Grid Filter

During the second phase of this project, we have focused on the Voxel Grid Filter because it is one of the most important features in autonomous driving cars.

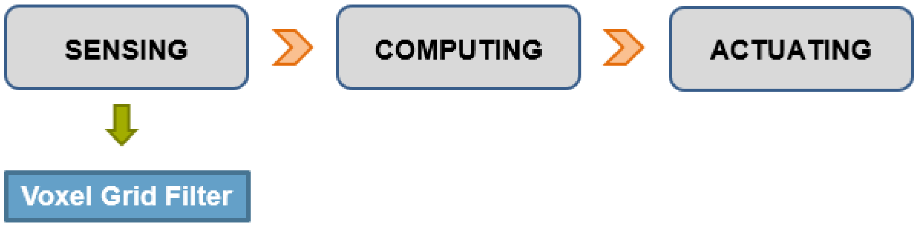


Figure 2: The Position of Voxel Grid Filter Among Different Functionalities of Autoware

Voxel Grid Filter falls under the sensing module of Autoware. Why do we need Voxel Grid Filter? We need Voxel Grid Filter because it separates irrelevant data from the useful data in the point cloud (3D data), and we have limited computation power to process all this huge output in terms of data rate from the sensing devices.

Data from LiDARs and stereo cameras are stored in a Point cloud in raw form. To achieve real-time processing, Autoware filters and preprocesses this raw point cloud data, which is then down-sampled using Voxel Grid Filter. This filter replaces a group of points contained in a cubic lattice to its centroid, and using Normal Distribution Transform relative position between two-point clouds can be obtained. Hence, an accurate position between 3D maps and sensors can be achieved.

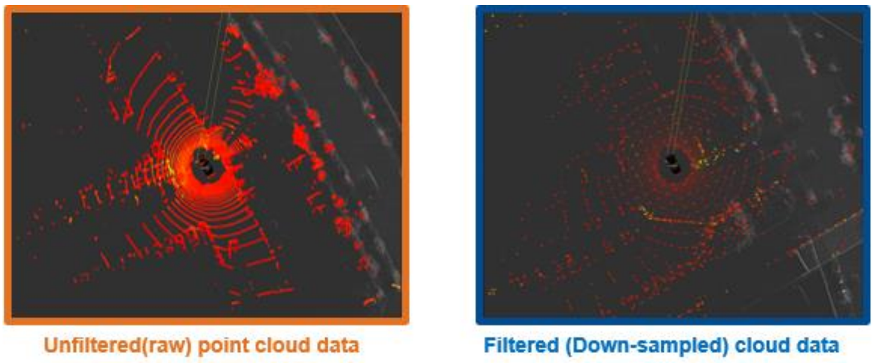

Unfiltered(raw) point cloud data
Filtered (Down-sampled) cloud data

Figure 3: A Comparison Between Raw and Down-Sampled Cloud Data

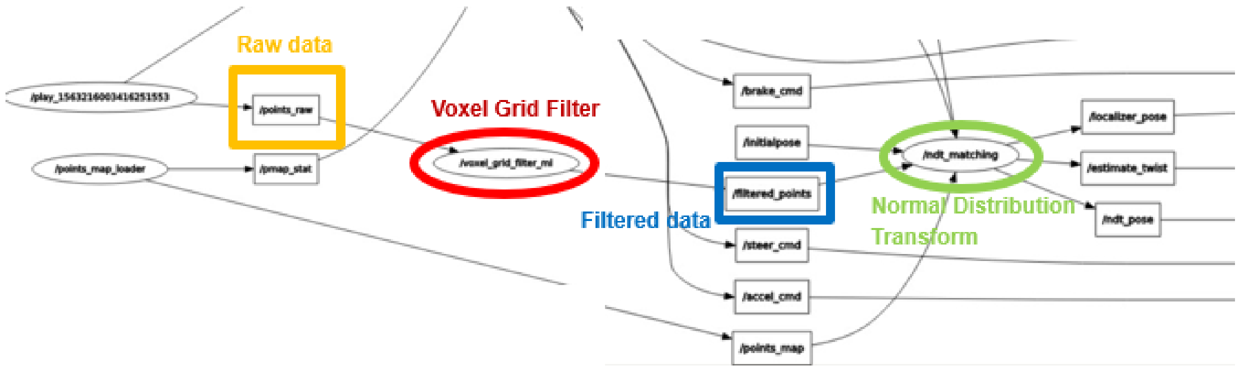


Figure 4: ROS Network Graph Displaying Connection Between Autoware and MATLAB

The diagram above shows a portion of ROS network graph displaying connection between Autoware and MATLAB. We can see that the raw data flows into Voxel Grid filter, and then the filtered data flows into Normal Distribution Transform to get an accurate position of the vehicle.

Chapter 5: Demonstration of Establishing a Connection Between MATLAB with Autoware

Before going to the demonstration of connecting MATLAB with Autoware, we motivate you to check the following tutorials from MATLAB:

- https://www.mathworks.com/help/ros/ug/get-started-with-ros.html
- https://www.mathworks.com/help/ros/ug/work-with-basic-ros-messages.html
- https://www.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html

The MATLAB code (on the next page) shows the function or ROS node that we have written to get the car's real-time position data from Autoware and plot it on MATLAB. This is called a *call-back* function. The difference between normal functions and call-back functions is that whenever a subscriber receives a new information, a call-back function will be triggered.

The subscriber `carpose` is created in MATLAB ROS node. This subscriber subscribes a message from the topic `/gnss_pose` from Autoware as well as calls the call-back function `CarPosCallback`. The `/gnss_pose` topic mediates the car position data and car orientation data. The global variables are declared to store the car's position and the car's orientation.

The call-back function `CarPosCallback` takes no specific arguments. Inside the function, the global variables are also defined to store the car's position and the car's orientation. Then the stored data are plotted to show the instantaneous position of the car.

```
carpos = rossubscriber('/gnss_pose',@CarPosCallback);

global posX;
global posY;
global posZ;
global orient;
plot(posX,posY);


function CarPosCallback(~, message)
    global posX;
    global posY;
    global posZ;
    global orient;
    posX = message.Pose.Position.X;
    posY = message.Pose.Position.Y;
    posZ = message.Pose.Position.Z;
    hold on;
    xlabel('X POSITION (m)');
    ylabel('Y POSITION (m)');
    title('INSTANTANEOUS CAR POSITION');
    plot(posX,posY,'-.ro');
    orient = [message.Pose.Orientation.X
message.Pose.Orientation.Y message.Pose.Orientation.Z];
end
```

To run this code, you can follow the procedure below. Please, note that before going through the following steps, make sure that you have run the Autoware Demo, and you have all the necessary files downloaded on your computer.

Step 1: To run Autoware, please type in following commands in terminal:

```
$ cd .autoware
$ cp ~/shared_dir/sample_moriyama_* .
$ tar zxfv sample_moriyama_150324.tar.gz
$ tar zxfv sample_moriyama_data.tar.gz
```

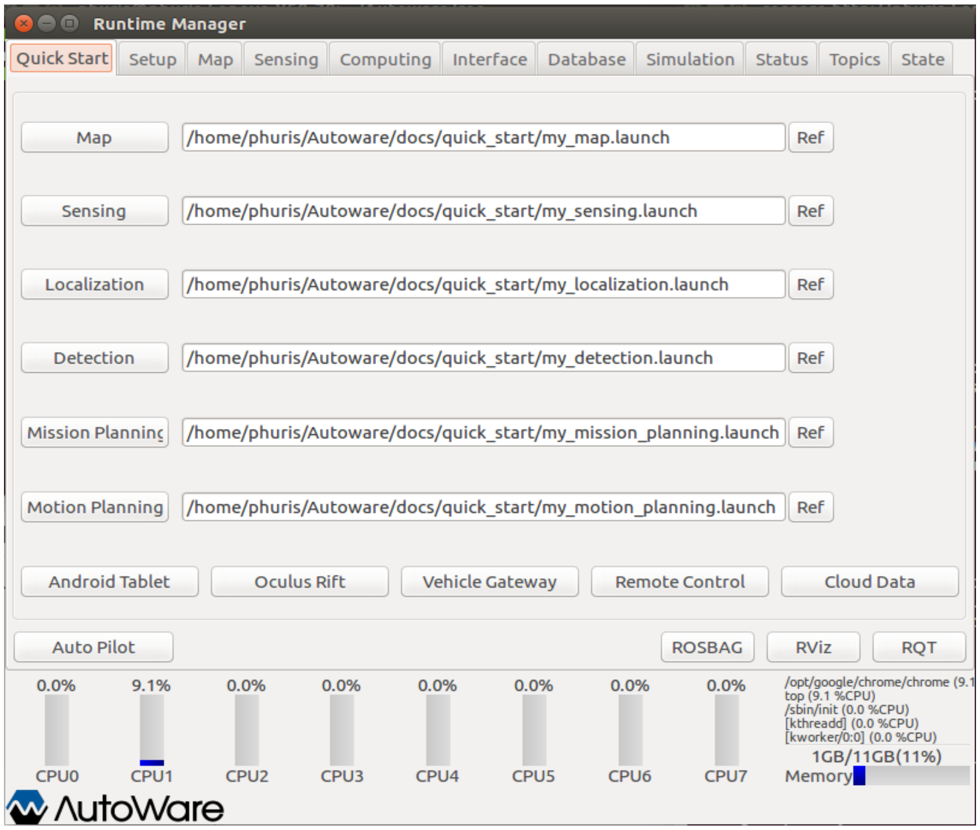
For Autoware version 1.12.0 and Newer

```
$ cd autoware.ai
$ source install/setup.bash
$ roslaunch runtime_manager runtime_manager.launch
```

For Autoware version 1.11.0 and 1.11.1

```
$ cd autoware/ros
$ source install/setup.bash
$ ./run
```

Then the Runtime Manager window appears:

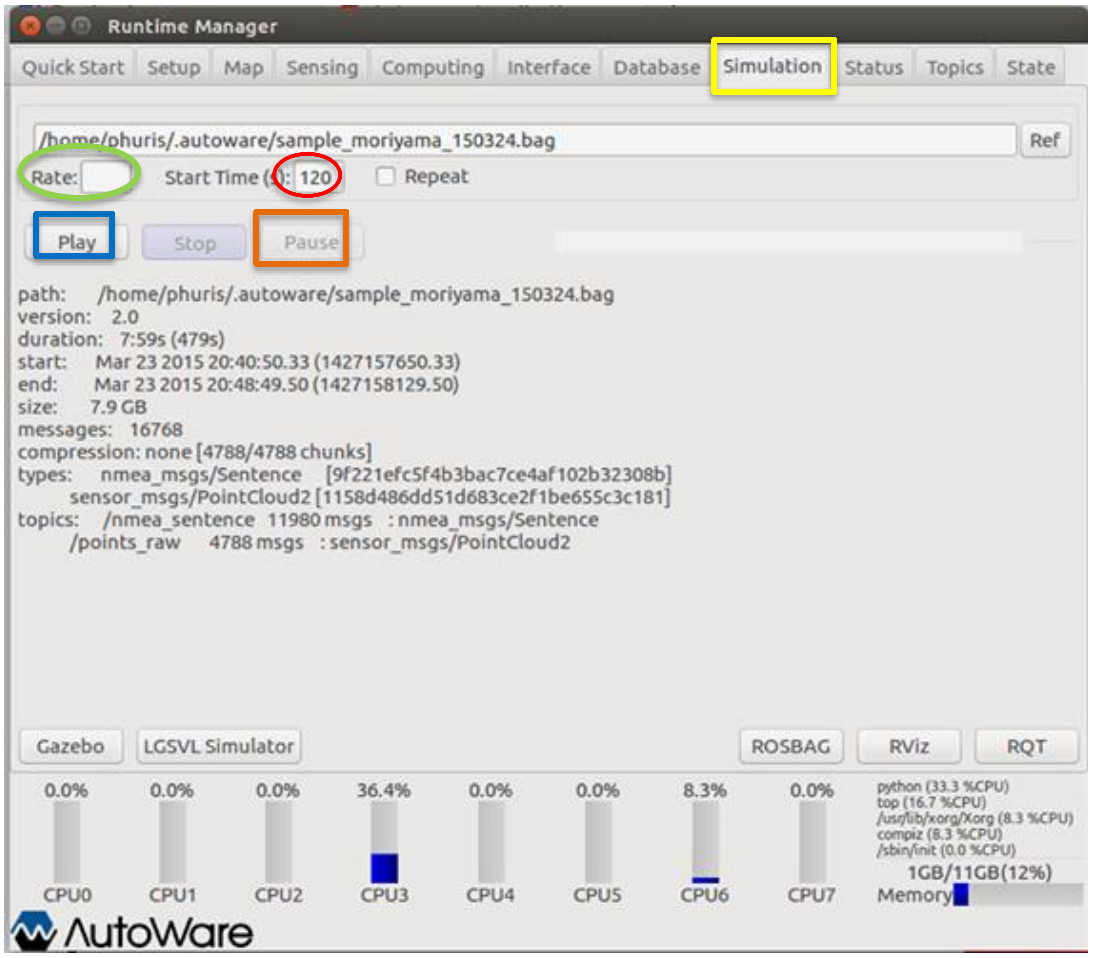


/home/phuris/Autoware/docs/quick_start/my_map.launch
/home/phuris/Autoware/docs/quick_start/my_sensing.launch
/home/phuris/Autoware/docs/quick_start/my_localization.launch
/home/phuris/Autoware/docs/quick_start/my_detection.launch
/home/phuris/Autoware/docs/quick_start/my_mission_planning.launch
/home/phuris/Autoware/docs/quick_start/my_motion_planning.launch

```
/opt/google/chrome/chrome (9.1
top (9.1 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
[kworker/0:0] (0.0 %CPU)
```

Figure 5: A screenshot of Runtime Manager Window

Step 2: Go to simulation tab, choosing the ROSBAG file as the one that you have downloaded when running the demo. Then set the start time to 120 seconds (this is done just to skip the time when the car drives out from the parking lot in the beginning of the playback). And leave the rate blank or put 1 (this is to adjust the speed of the playback)



/home/phuris/.autoware/sample_moriyama_150324.bag
Rate: Start Time (s): 120 Repeat

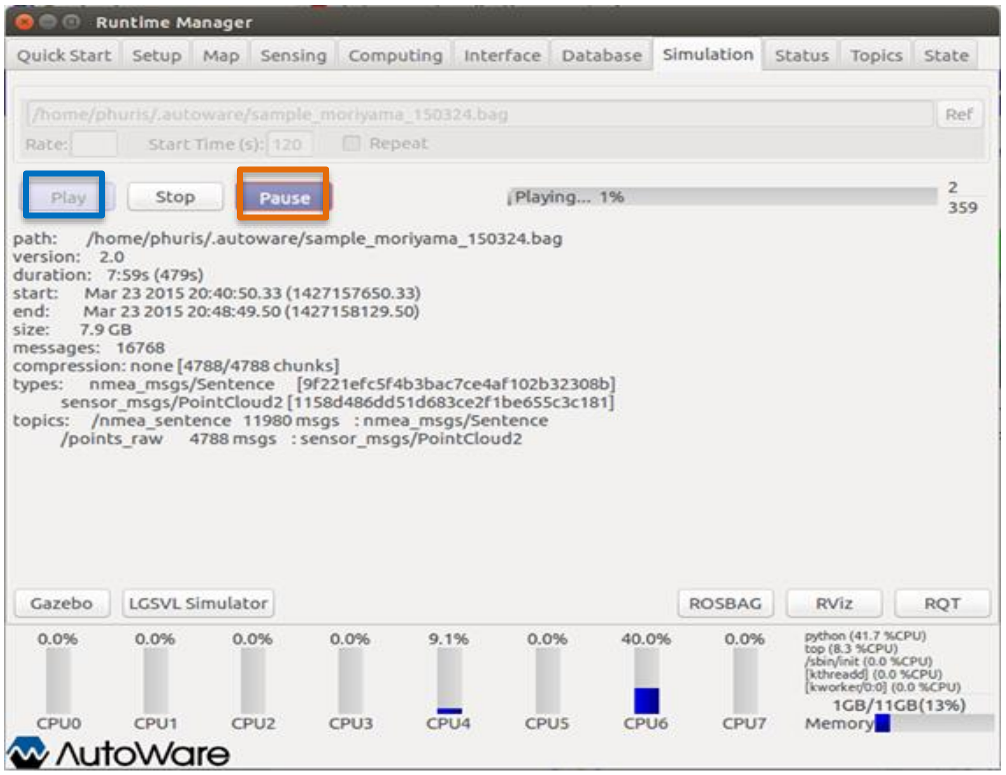
```
path:     /home/phuris/.autoware/sample_moriyama_150324.bag
version:   2.0
duration:  7:59s (479s)
start:     Mar 23 2015 20:40:50.33 (1427157650.33)
end:       Mar 23 2015 20:48:49.50 (1427158129.50)
size:      7.9 GB
messages:  16768
compression: none [4788/4788 chunks]
types:     nmea_msgs/Sentence     [9f221efc5f4b3bac7ce4af102b32308b]
           sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:    /nmea_sentence  11980 msgs  : nmea_msgs/Sentence
           /points_raw      4788 msgs  : sensor_msgs/PointCloud2
```

0.0%	0.0%	0.0%	36.4%	0.0%	0.0%	8.3%	0.0%
CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7

python (33.3 %CPU)
top (16.7 %CPU)
/usr/lib/xorg/Xorg (8.3 %CPU)
compiz (8.3 %CPU)
/sbin/init (0.0 %CPU)
1GB/11GB(12%)
Memory

Figure 6: A screenshot of Simulation Tab

Step 3: Play and pause the simulation

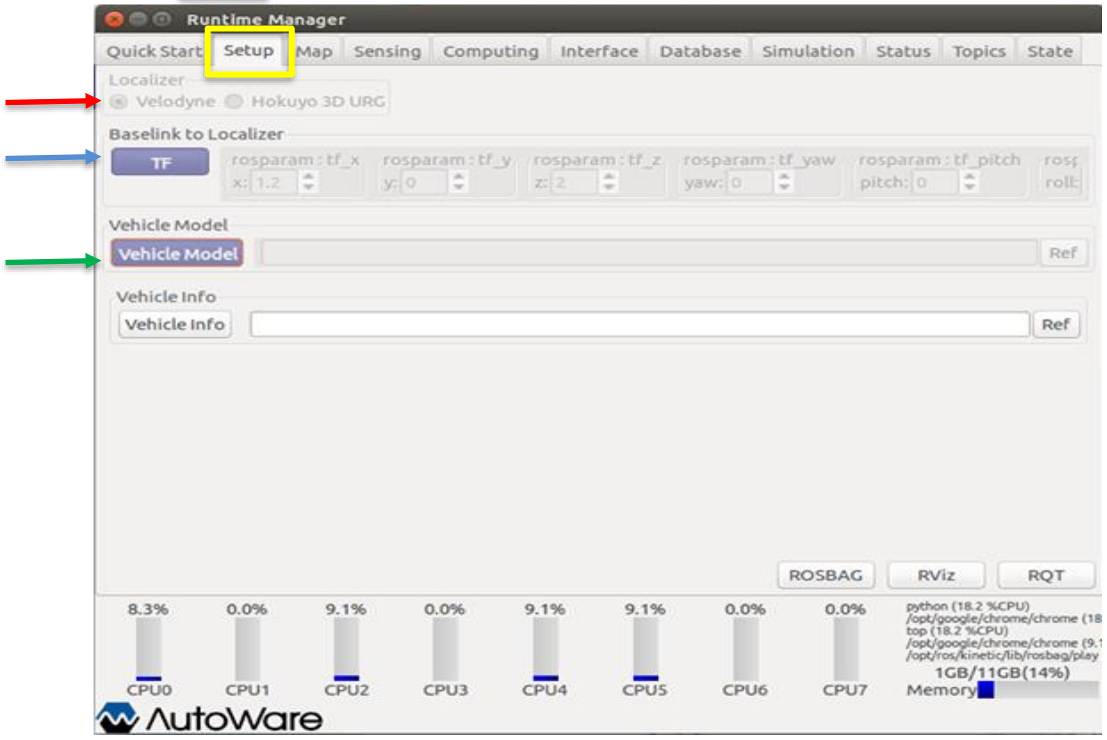


```
/home/phuris/.autoware/sample_moriyama_150324.bag
Rate:        Start Time (s): 120    Repeat
2
359
path:      /home/phuris/.autoware/sample_moriyama_150324.bag
version:   2.0
duration:  7:59s (479s)
start:     Mar 23 2015 20:40:50.33 (1427157650.33)
end:       Mar 23 2015 20:48:49.50 (1427158129.50)
size:      7.9 GB
messages:  16768
compression: none [4788/4788 chunks]
types:     nmea_msgs/Sentence     [9f221efc5f4b3bac7ce4af102b32308b]
           sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:    /nmea_sentence  11980 msgs   : nmea_msgs/Sentence
           /points_raw      4788 msgs   : sensor_msgs/PointCloud2
```

```
0.0%   0.0%   0.0%   0.0%   9.1%   0.0%   40.0%   0.0%
CPU0   CPU1   CPU2   CPU3   CPU4   CPU5   CPU6    CPU7
python (41.7 %CPU)
top (8.3 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
[kworker/0:0] (0.0 %CPU)
1GB/11GB(13%)
Memory
```

Figure 7: Play and Pause

Step 4: Go to Setup tab, set the localizer to Velodyne and turn the TF and Vehicle Model on.



```
Localizer
(●) Velodyne  ( ) Hokuyo 3D URG
Baselink to Localizer
rosparam : tf_x   rosparam : tf_y   rosparam : tf_z   rosparam : tf_yaw   rosparam : tf_pitch   rosp
x: 1.2            y: 0              z: 2              yaw: 0              pitch: 0              roll:
Vehicle Model
Vehicle Info
```

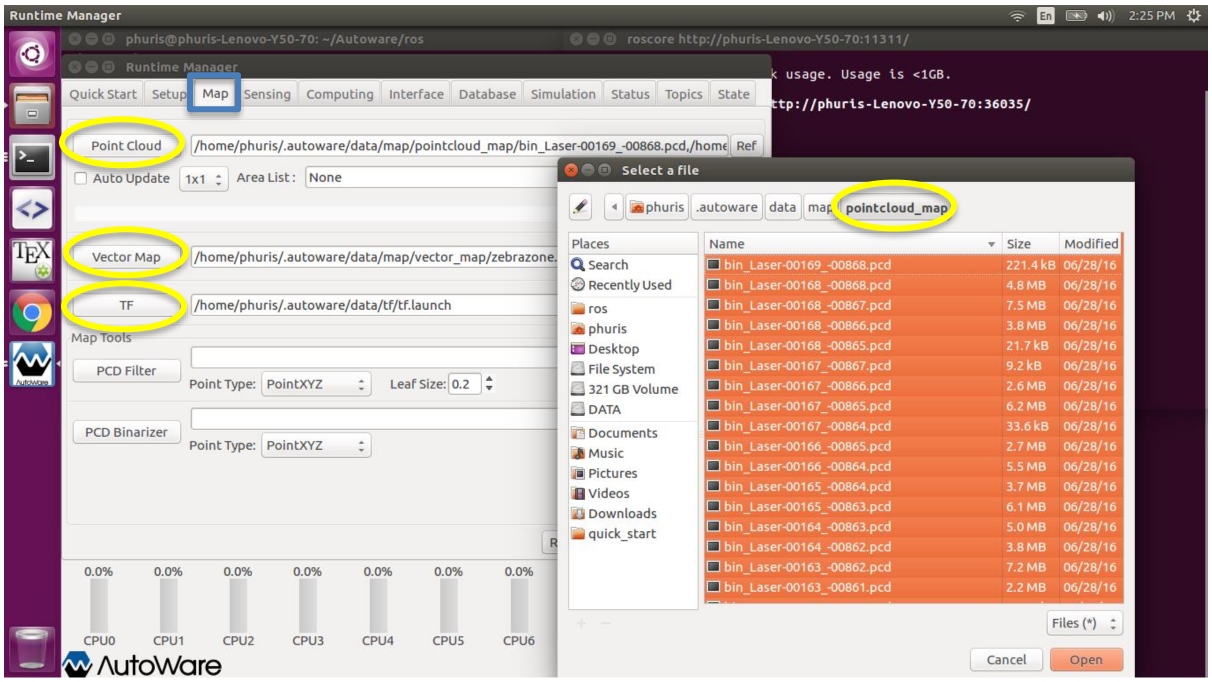
```
8.3%   0.0%   9.1%   0.0%   9.1%   9.1%   0.0%   0.0%
CPU0   CPU1   CPU2   CPU3   CPU4   CPU5   CPU6   CPU7
python (18.2 %CPU)
/opt/google/chrome/chrome (18
top (18.2 %CPU)
/opt/google/chrome/chrome (9.1
/opt/ros/kinetic/lib/rosbag/play
1GB/11GB(14%)
Memory
```

Figure 8: A screenshot of Setup Tab Window

Step 5:

- Go to the Map tab.

- Load the Point Cloud, Vector Map, and TF from the ROSBAG data which was downloaded when performing demo running after Autoware installation.
- After loading all the data, turn the Point Cloud, Vector Map, and TF on by simply clicking on them.



```
k usage. Usage is <1GB.
ttp://phuris-Lenovo-Y50-70:36035/
```

Point Cloud: /home/phuris/.autoware/data/map/pointcloud_map/bin_Laser-00169_-00868.pcd,/home

Auto Update 1x1 Area List : None

Vector Map: /home/phuris/.autoware/data/map/vector_map/zebrazone.

TF: /home/phuris/.autoware/data/tf/tf.launch

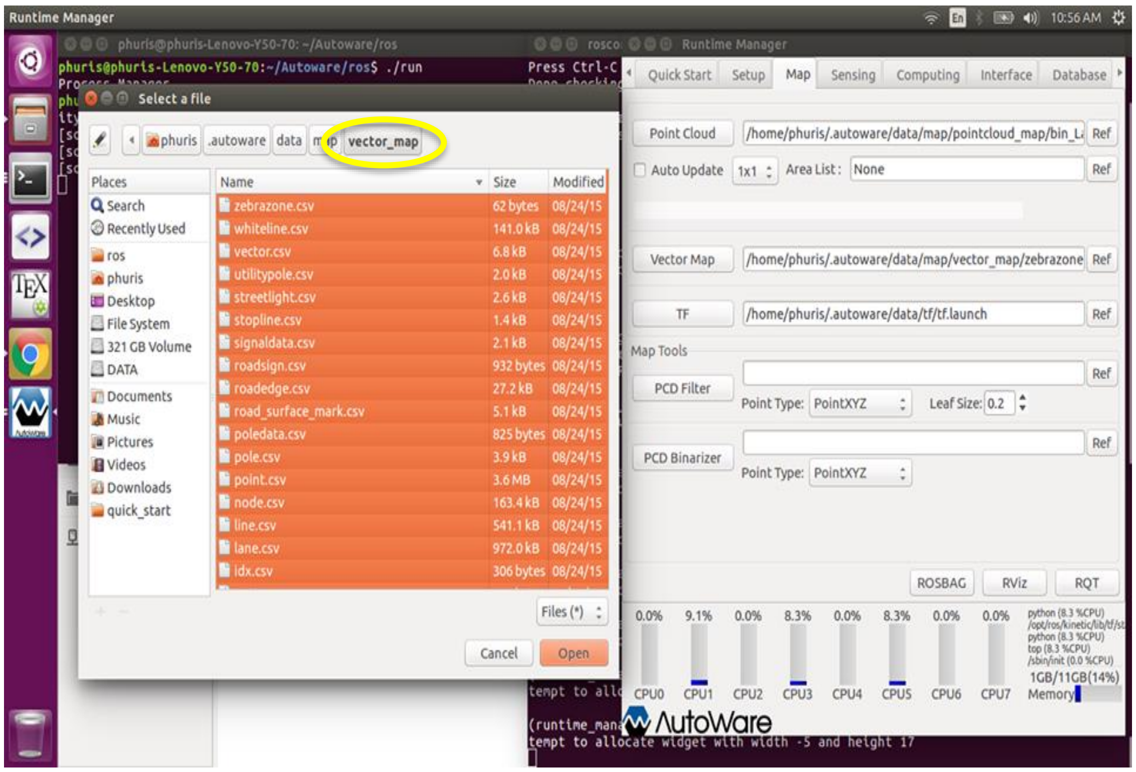
PCD Filter — Point Type: PointXYZ Leaf Size: 0.2

PCD Binarizer — Point Type: PointXYZ

CPU0 0.0% CPU1 0.0% CPU2 0.0% CPU3 0.0% CPU4 0.0% CPU5 0.0% CPU6 0.0%

Name	Size	Modified
bin_Laser-00169_-00868.pcd	221.4 kB	06/28/16
bin_Laser-00168_-00868.pcd	4.8 MB	06/28/16
bin_Laser-00168_-00867.pcd	7.5 MB	06/28/16
bin_Laser-00168_-00866.pcd	3.8 MB	06/28/16
bin_Laser-00168_-00865.pcd	21.7 kB	06/28/16
bin_Laser-00167_-00867.pcd	9.2 kB	06/28/16
bin_Laser-00167_-00866.pcd	2.6 MB	06/28/16
bin_Laser-00167_-00865.pcd	6.2 MB	06/28/16
bin_Laser-00167_-00864.pcd	33.6 kB	06/28/16
bin_Laser-00166_-00865.pcd	2.7 MB	06/28/16
bin_Laser-00166_-00864.pcd	5.5 MB	06/28/16
bin_Laser-00165_-00864.pcd	3.7 MB	06/28/16
bin_Laser-00165_-00863.pcd	6.1 MB	06/28/16
bin_Laser-00164_-00863.pcd	5.0 MB	06/28/16
bin_Laser-00164_-00862.pcd	3.8 MB	06/28/16
bin_Laser-00163_-00862.pcd	7.2 MB	06/28/16
bin_Laser-00163_-00861.pcd	2.2 MB	06/28/16

Figure 9: A screenshot of Map Tab and selection of Data for Point Cloud



```
phuris@phuris-Lenovo-Y50-70:~/Autoware/ros$ ./run
Process Manager
```

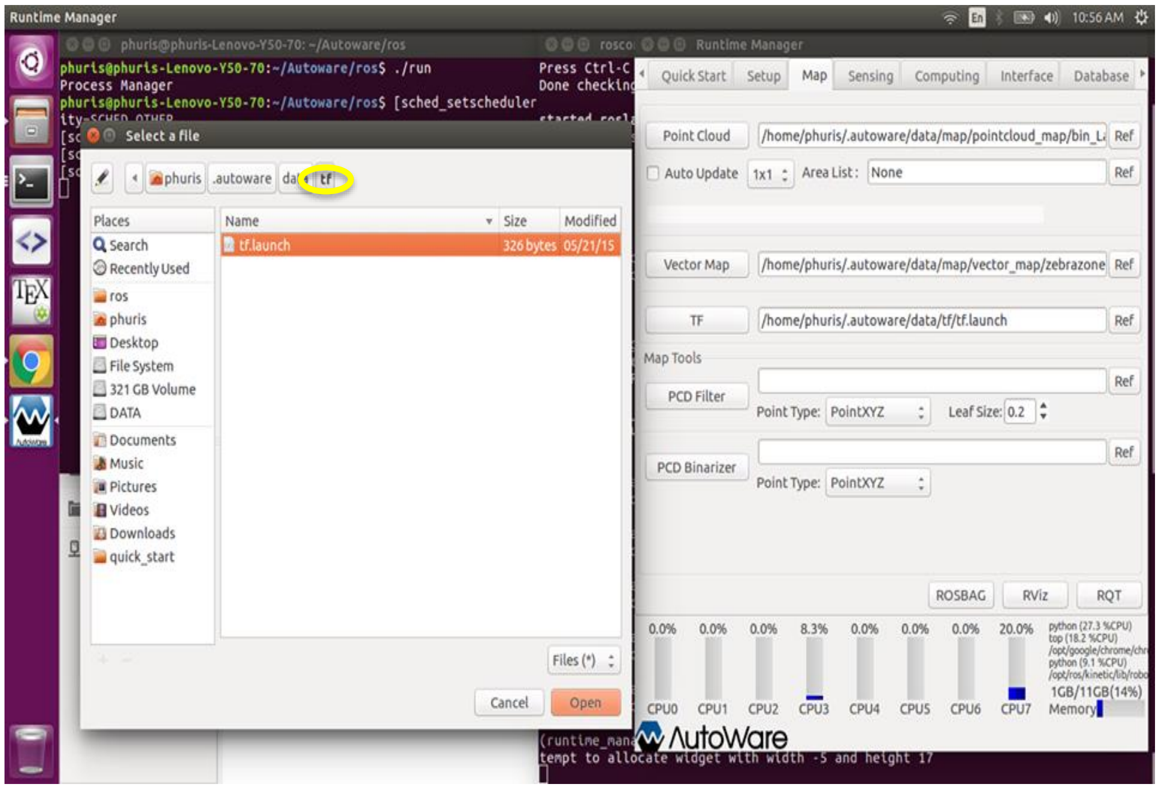
```
Press Ctrl-C
Done checking
tempt to allc
(runtime_mana
tempt to allocate widget with width -5 and height 17
```

Name	Size	Modified
zebrazone.csv	62 bytes	08/24/15
whiteline.csv	141.0 kB	08/24/15
vector.csv	6.8 kB	08/24/15
utilitypole.csv	2.0 kB	08/24/15
streetlight.csv	2.6 kB	08/24/15
stopline.csv	1.4 kB	08/24/15
signaldata.csv	2.1 kB	08/24/15
roadsign.csv	932 bytes	08/24/15
roadedge.csv	27.2 kB	08/24/15
road_surface_mark.csv	5.1 kB	08/24/15
poledata.csv	825 bytes	08/24/15
pole.csv	3.9 kB	08/24/15
point.csv	3.6 MB	08/24/15
node.csv	163.4 kB	08/24/15
line.csv	541.1 kB	08/24/15
lane.csv	972.0 kB	08/24/15
idx.csv	306 bytes	08/24/15

Point Cloud: /home/phuris/.autoware/data/map/pointcloud_map/bin_L
Auto Update 1x1 Area List: None
Vector Map: /home/phuris/.autoware/data/map/vector_map/zebrazone
TF: /home/phuris/.autoware/data/tf/tf.launch
PCD Filter — Point Type: PointXYZ Leaf Size: 0.2
PCD Binarizer — Point Type: PointXYZ

CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7	Memory
0.0%	9.1%	0.0%	8.3%	0.0%	8.3%	0.0%	0.0%	1GB/11GB(14%)

Figure 10: Loading Vector Map Data



```
phuris@phuris-Lenovo-Y50-70:~/Autoware/ros$ ./run
Process Manager
phuris@phuris-Lenovo-Y50-70:~/Autoware/ros$ [sched_setscheduler
ity-SCHED_OTHER
```

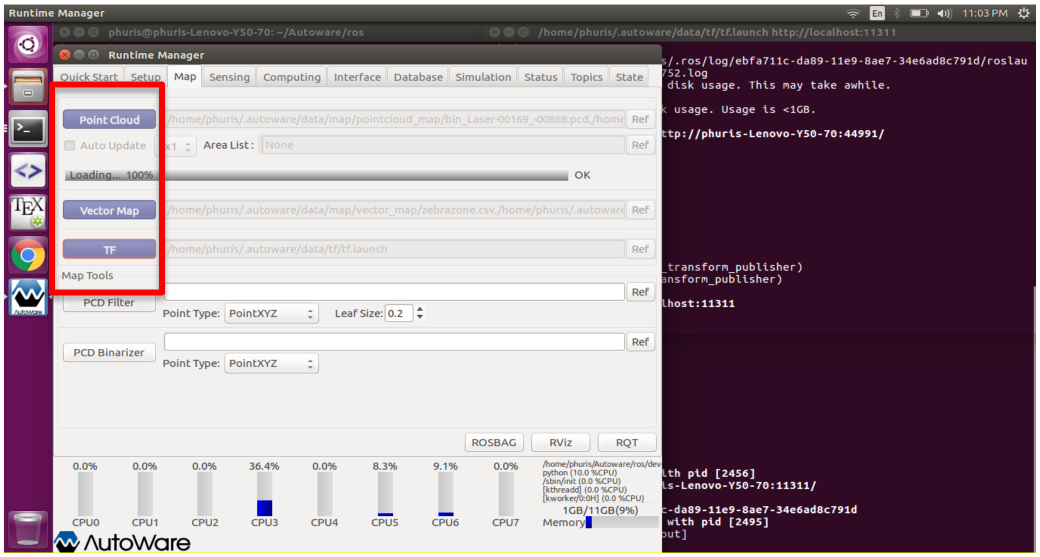
```
Press Ctrl-C
Done checking
(runtime_mana
tempt to allocate widget with width -5 and height 17
```

Name	Size	Modified
tf.launch	326 bytes	05/21/15

Point Cloud: /home/phuris/.autoware/data/map/pointcloud_map/bin_L
Auto Update 1x1 Area List: None
Vector Map: /home/phuris/.autoware/data/map/vector_map/zebrazone
TF: /home/phuris/.autoware/data/tf/tf.launch
PCD Filter — Point Type: PointXYZ Leaf Size: 0.2
PCD Binarizer — Point Type: PointXYZ

CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7	Memory
0.0%	0.0%	0.0%	8.3%	0.0%	0.0%	0.0%	20.0%	1GB/11GB(14%)

Figure 11: Loading TF Data



/home/phuris/.autoware/data/tf/tf.launch http://localhost:11311

Point Cloud: /home/phuris/.autoware/data/map/pointcloud_map/bin_Laser-00169_-00868.pcd,/home

Auto Update x1 Area List : None

Loading... 100% OK

Vector Map: /home/phuris/.autoware/data/map/vector_map/zebrazone.csv,/home/phuris/.autoware

TF: /home/phuris/.autoware/data/tf/tf.launch

PCD Filter — Point Type: PointXYZ Leaf Size: 0.2

PCD Binarizer — Point Type: PointXYZ

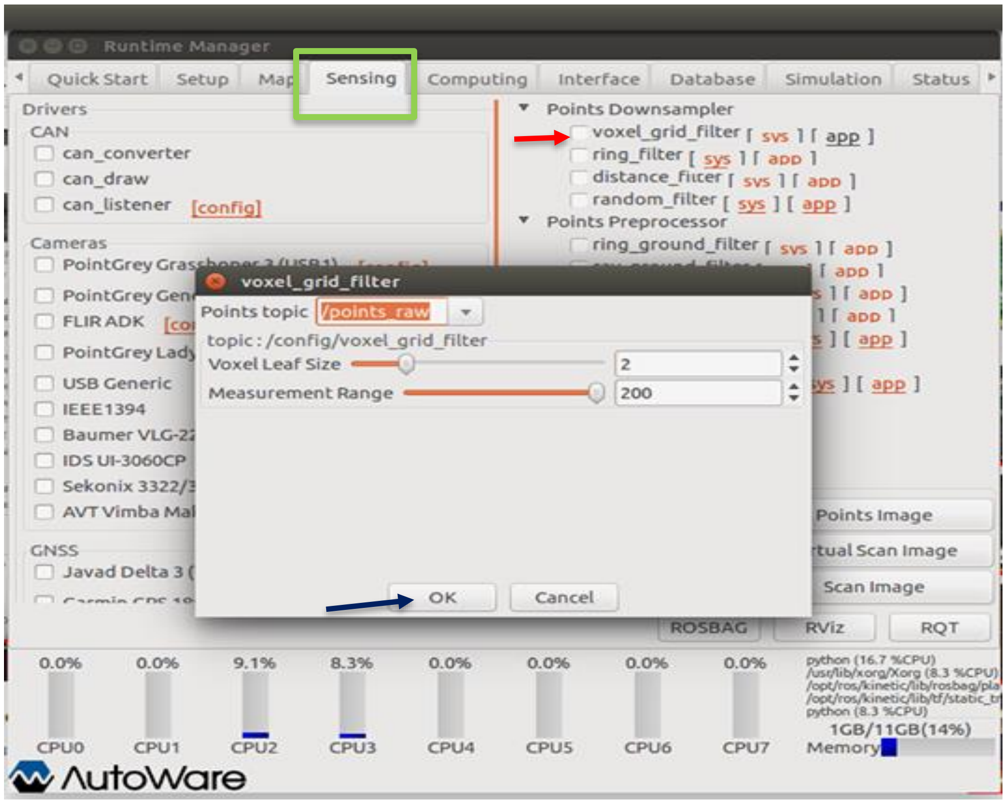
CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7
0.0%	0.0%	0.0%	36.4%	0.0%	8.3%	9.1%	0.0%

```
/home/phuris/Autoware/ros/dev
python (10.0 %CPU)
/sbin/init (0.0 %CPU)
[kthreadd] (0.0 %CPU)
[kworker/0:0H] (0.0 %CPU)
1GB/11GB(9%)
Memory
```

```
s/.ros/log/ebfa711c-da89-11e9-8ae7-34e6ad8c791d/roslau
752.log
 disk usage. This may take awhile.

k usage. Usage is <1GB.

ttp://phuris-Lenovo-Y50-70:44991/


_transform_publisher)
ansform_publisher)

lhost:11311


ith pid [2456]
is-Lenovo-Y50-70:11311/

c-da89-11e9-8ae7-34e6ad8c791d
 with pid [2495]
out]
```

Figure 12: Turning on Point Cloud, Vector Map, and TF by simply clicking on them

Step 6: Go to Sensing tab and click the voxel_grid_filter app in the Points Downsampler section and set the value as shown in the picture below. Then click ok to close the app window and turn the voxel_grid_filter on by tick the square box in front of it.



voxel_grid_filter

Points topic /points_raw

topic : /config/voxel_grid_filter

Voxel Leaf Size 2

Measurement Range 200

CPU0	CPU1	CPU2	CPU3	CPU4	CPU5	CPU6	CPU7
0.0%	0.0%	9.1%	8.3%	0.0%	0.0%	0.0%	0.0%

```
python (16.7 %CPU)
/usr/lib/xorg/Xorg (8.3 %CPU)
/opt/ros/kinetic/lib/rosbag/pla
/opt/ros/kinetic/lib/tf/static_tf
python (8.3 %CPU)
1GB/11GB(14%)
Memory
```

Figure 13: A screenshot of Sensing Tab

Step 7: Go to Computing tab, turn the nmea2tfpose on.

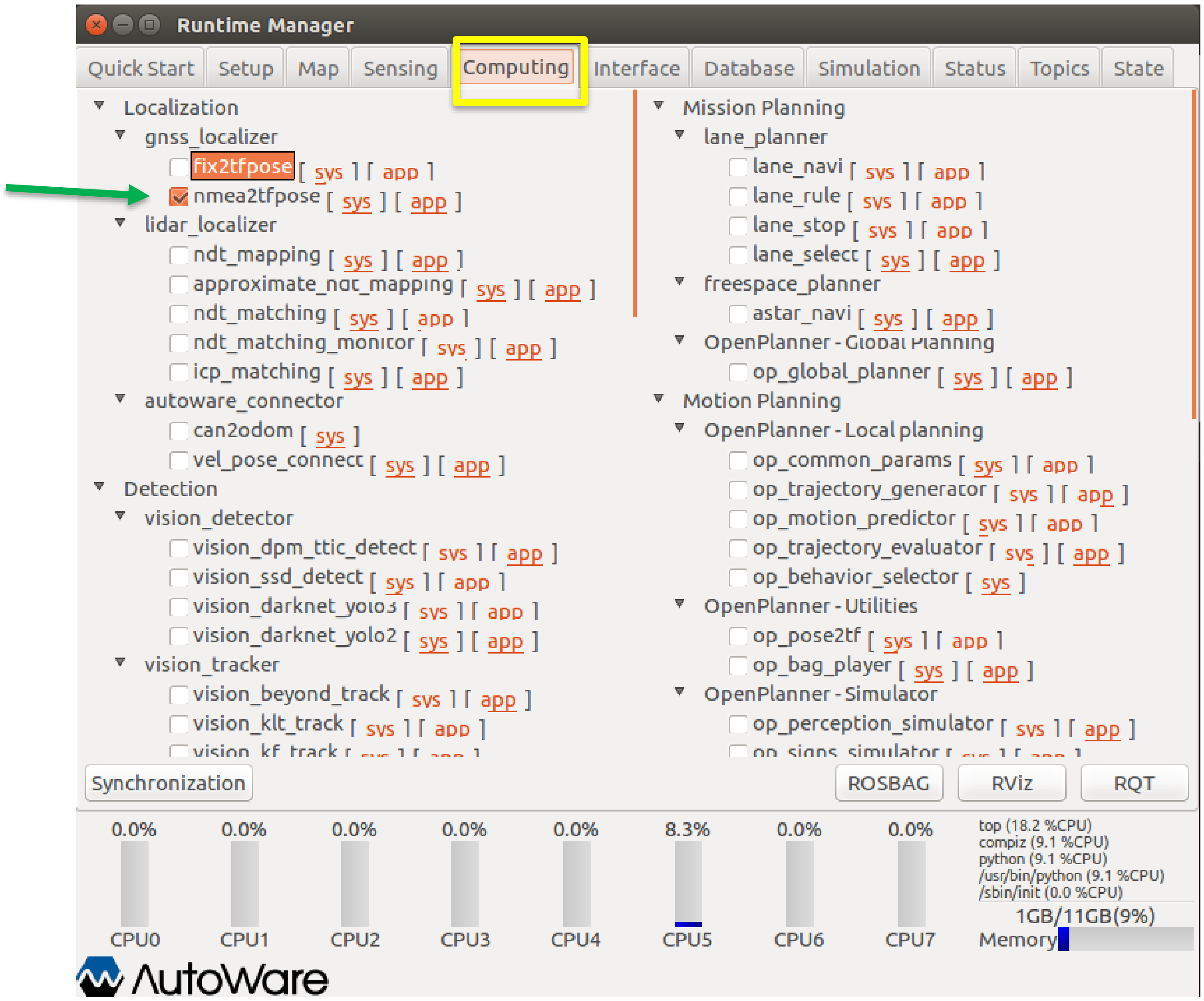

Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Status
Topics
State
Localization
gnss_localizer
fix2tfpose [sys] [app]
✓ nmea2tfpose [sys] [app]
lidar_localizer
ndt_mapping [sys] [app]
approximate_ndt_mapping [sys] [app]
ndt_matching [sys] [app]
ndt_matching_monitor [sys] [app]
icp_matching [sys] [app]
autoware_connector
can2odom [sys]
vel_pose_connect [sys] [app]
Detection
vision_detector
vision_dpm_ttic_detect [sys] [app]
vision_ssd_detect [sys] [app]
vision_darknet_yolo3 [sys] [app]
vision_darknet_yolo2 [sys] [app]
vision_tracker
vision_beyond_track [sys] [app]
vision_klt_track [sys] [app]
vision_kf_track [sys] [app]
Mission Planning
lane_planner
lane_navi [sys] [app]
lane_rule [sys] [app]
lane_stop [sys] [app]
lane_select [sys] [app]
freespace_planner
astar_navi [sys] [app]
OpenPlanner - Global Planning
op_global_planner [sys] [app]
Motion Planning
OpenPlanner - Local planning
op_common_params [sys] [app]
op_trajectory_generator [sys] [app]
op_motion_predictor [sys] [app]
op_trajectory_evaluator [sys] [app]
op_behavior_selector [sys]
OpenPlanner - Utilities
op_pose2tf [sys] [app]
op_bag_player [sys] [app]
OpenPlanner - Simulator
op_perception_simulator [sys] [app]
op_signs_simulator [sys] [app]
Synchronization
ROSBAG
RViz
RQT
0.0% CPU0
0.0% CPU1
0.0% CPU2
0.0% CPU3
0.0% CPU4
8.3% CPU5
0.0% CPU6
0.0% CPU7
top (18.2 %CPU)
compiz (9.1 %CPU)
python (9.1 %CPU)
/usr/bin/python (9.1 %CPU)
/sbin/init (0.0 %CPU)
1GB/11GB(9%)
Memory
AutoWare

Figure 14: A screenshot of Computing Tab

Step 8: Still in the Computing tab, click the ndt_matching app in the lidar_localizer section. Set all the values and settings as shown in the picture below then click ok to close the window. Then turn the ndt_matching on.

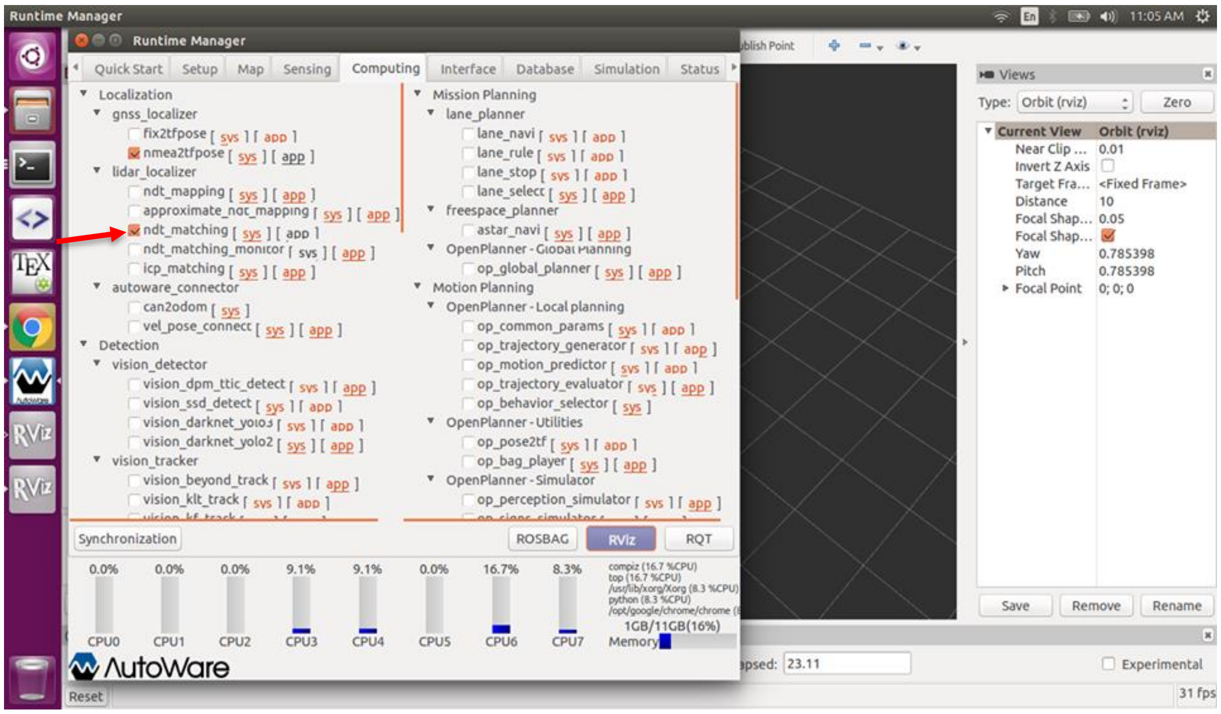

Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Status
Localization
gnss_localizer
fix2tfpose [sys] [app]
nmea2tfpose [sys] [app]
lidar_localizer
ndt_mapping [sys] [app]
approximate_ndt_mapping [sys] [app]
ndt_matching [sys] [app]
ndt_matching_monitor [sys] [app]
icp_matching [sys] [app]
autoware_connector
can2odom [sys]
vel_pose_connect [sys] [app]
Detection
vision_detector
vision_dpm_ttic_detect [sys] [app]
vision_ssd_detect [sys] [app]
vision_darknet_yolo3 [sys] [app]
vision_darknet_yolo2 [sys] [app]
vision_tracker
vision_beyond_track [sys] [app]
vision_klt_track [sys] [app]
Mission Planning
lane_planner
lane_navi [sys] [app]
lane_rule [sys] [app]
lane_stop [sys] [app]
lane_select [sys] [app]
freespace_planner
astar_navi [sys] [app]
OpenPlanner - Global Planning
op_global_planner [sys] [app]
Motion Planning
OpenPlanner - Local planning
op_common_params [sys] [app]
op_trajectory_generator [sys] [app]
op_motion_predictor [sys] [app]
op_trajectory_evaluator [sys] [app]
op_behavior_selector [sys]
OpenPlanner - Utilities
op_pose2tf [sys] [app]
op_bag_player [sys] [app]
OpenPlanner - Simulator
op_perception_simulator [sys] [app]
Synchronization
ROSBAG
RViz
RQT
0.0% 0.0% 0.0% 9.1% 9.1% 0.0% 16.7% 8.3%
CPU0 CPU1 CPU2 CPU3 CPU4 CPU5 CPU6 CPU7
compiz (16.7 %CPU)
top (16.7 %CPU)
/usr/lib/xorg/Xorg (8.3 %CPU)
python (8.3 %CPU)
/opt/google/chrome/chrome
1GB/11GB(16%)
Memory
AutoWare
Reset
Views
Type: Orbit (rviz)
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 10
Focal Shap... 0.05
Focal Shap...
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0
Save
Remove
Rename
Experimental
31 fps
11:05 AM

Figure 15: ndt Matching App

Step 9: Open RVIZ from Autoware

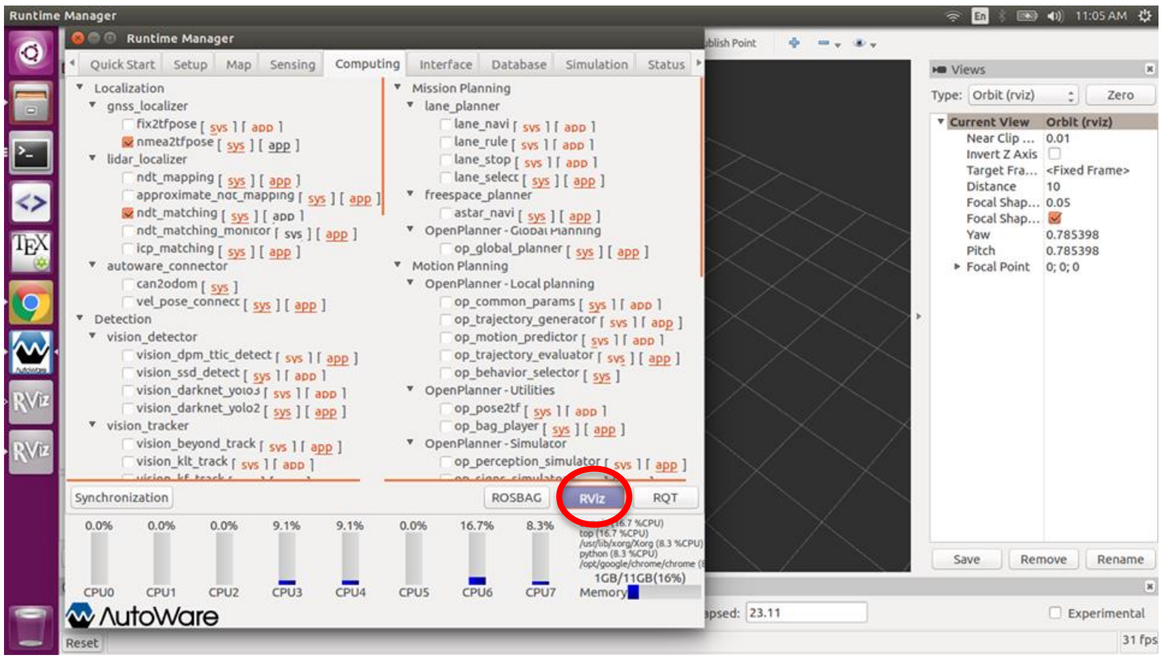
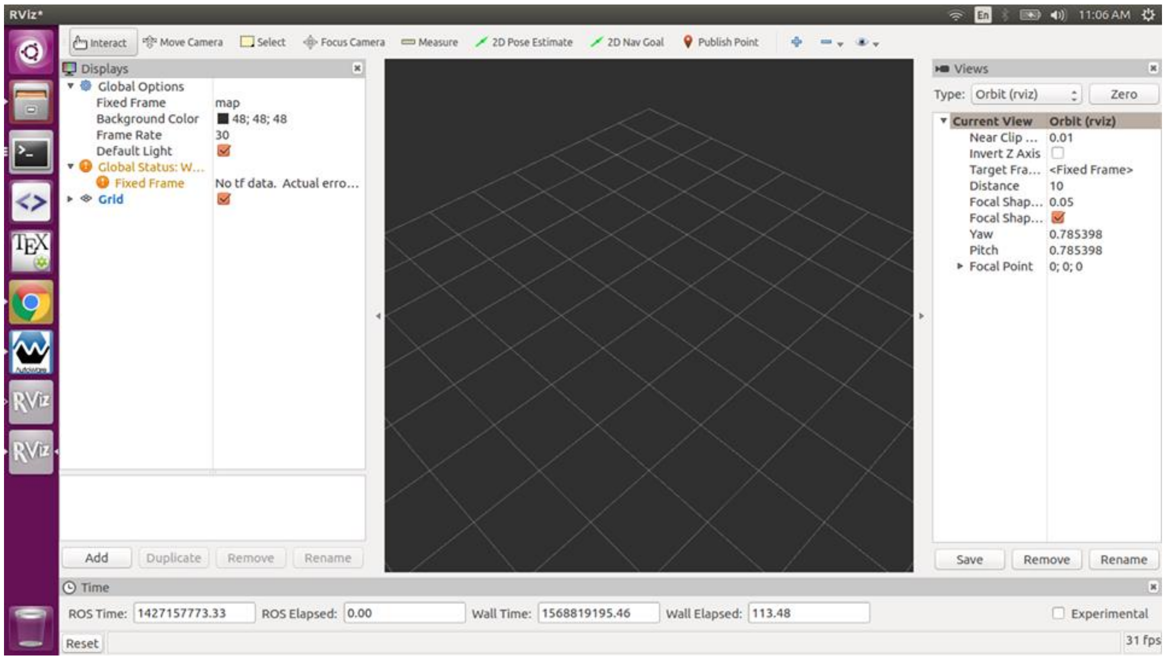


Figure 16: Opening Rviz



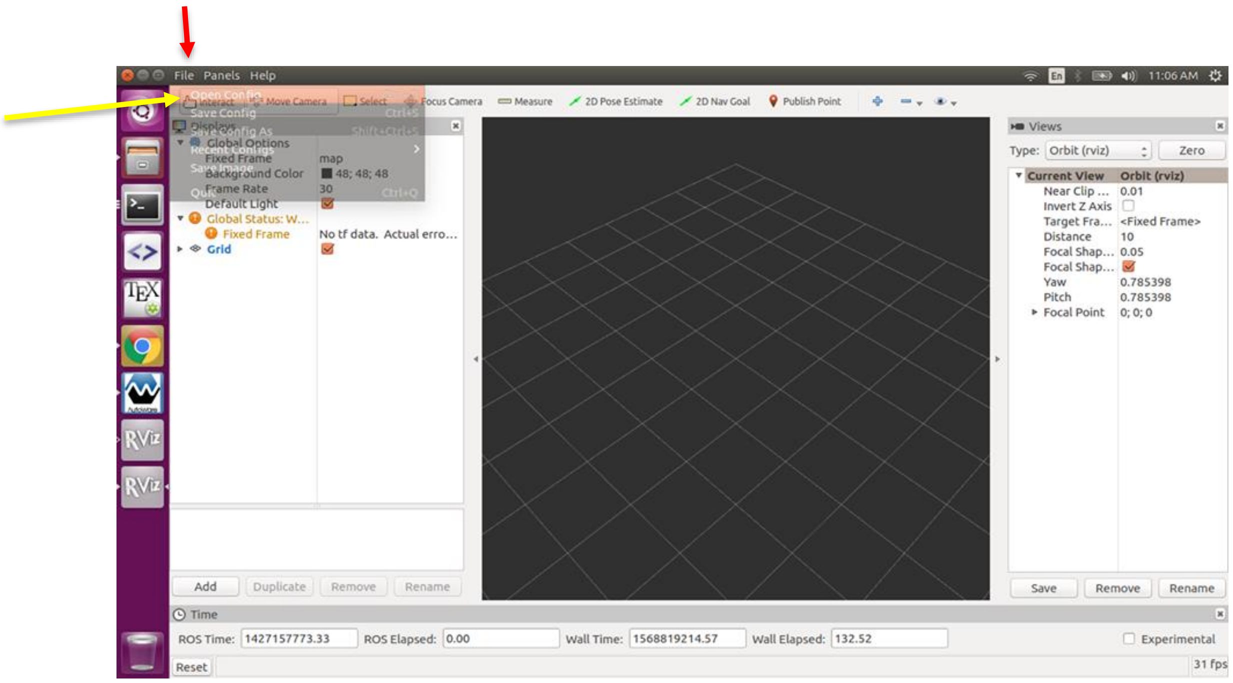
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light ✓
Global Status: W...
Fixed Frame No tf data. Actual erro...
Grid ✓

Type: Orbit (rviz)
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis ☐
Target Fra... <Fixed Frame>
Distance 10
Focal Shap... 0.05
Focal Shap... ✓
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0

ROS Time: 1427157773.33 ROS Elapsed: 0.00 Wall Time: 1568819195.46 Wall Elapsed: 113.48 ☐ Experimental
31 fps

Figure 17: Appearing of RViz Window

Step 10: Once the RVIZ shows up, open the default configuration by clicking on File, Open Config.



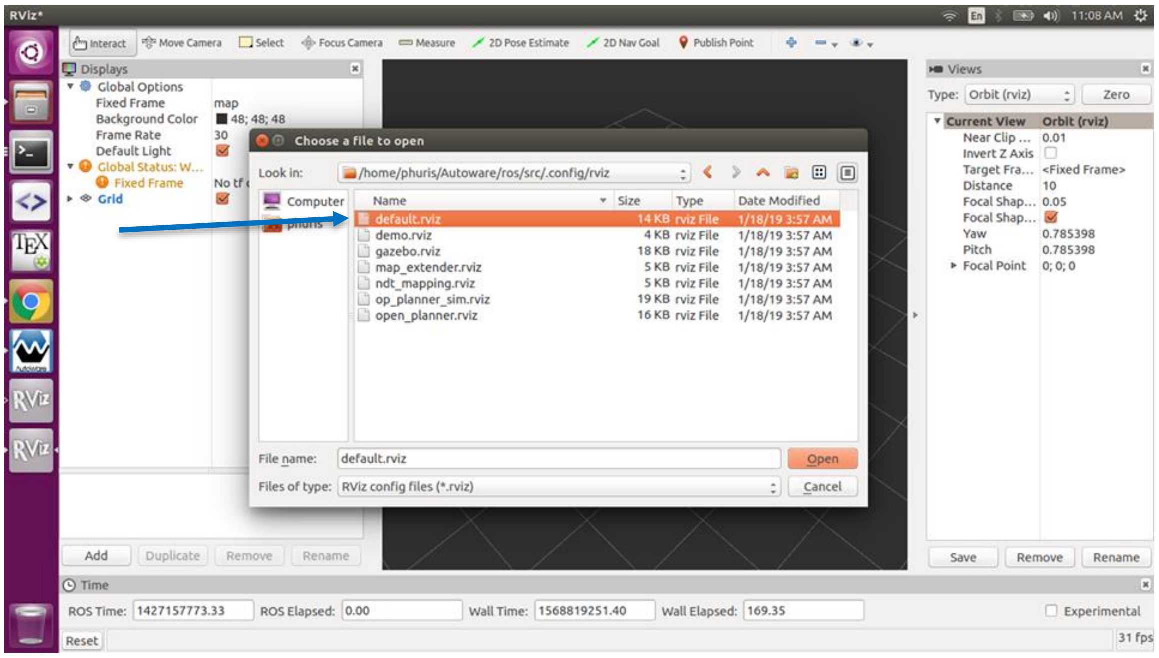
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light ✓
Global Status: W...
Fixed Frame No tf data. Actual erro...
Grid ✓

Type: Orbit (rviz)
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis ☐
Target Fra... <Fixed Frame>
Distance 10
Focal Shap... 0.05
Focal Shap... ✓
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0

ROS Time: 1427157773.33 ROS Elapsed: 0.00 Wall Time: 1568819214.57 Wall Elapsed: 132.52 ☐ Experimental
31 fps

Figure 18: Open Config in RViz by clicking on File

Step 11: Open the default.rviz by selecting the path where the Autoware folder is : Autoware/ros/src/.config/rviz. In the folder src, if you cannot see the folder .config, right click and shows the hidden folders.



Global Options	
Fixed Frame	map
Background Color	48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: W...	
Fixed Frame	No tf
Grid	✓

Choose a file to open

Look in: /home/phuris/Autoware/ros/src/.config/rviz

Name	Size	Type	Date Modified
default.rviz	14 KB	rviz File	1/18/19 3:57 AM
demo.rviz	4 KB	rviz File	1/18/19 3:57 AM
gazebo.rviz	18 KB	rviz File	1/18/19 3:57 AM
map_extender.rviz	5 KB	rviz File	1/18/19 3:57 AM
ndt_mapping.rviz	5 KB	rviz File	1/18/19 3:57 AM
op_planner_sim.rviz	19 KB	rviz File	1/18/19 3:57 AM
open_planner.rviz	16 KB	rviz File	1/18/19 3:57 AM

File name: default.rviz

Files of type: RViz config files (*.rviz)

Type: Orbit (rviz)

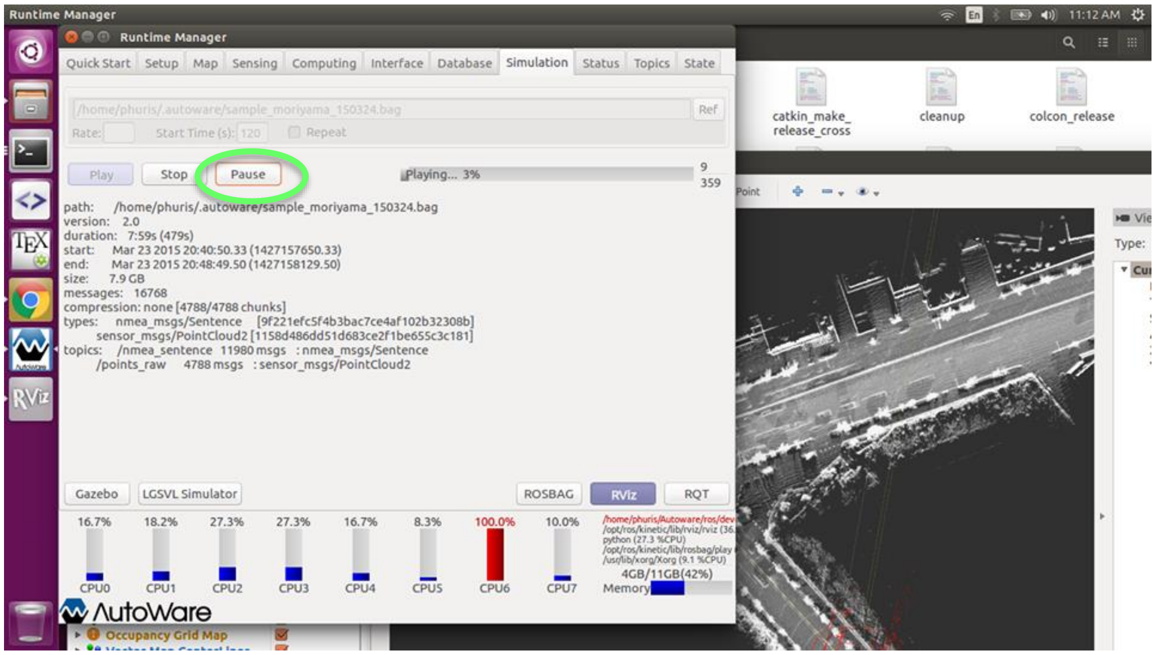
Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	☐
Target Fra...	<Fixed Frame>
Distance	10
Focal Shap...	0.05
Focal Shap...	✓
Yaw	0.785398
Pitch	0.785398
Focal Point	0; 0; 0

ROS Time: 1427157773.33 ROS Elapsed: 0.00 Wall Time: 1568819251.40 Wall Elapsed: 169.35 ☐ Experimental

31 fps

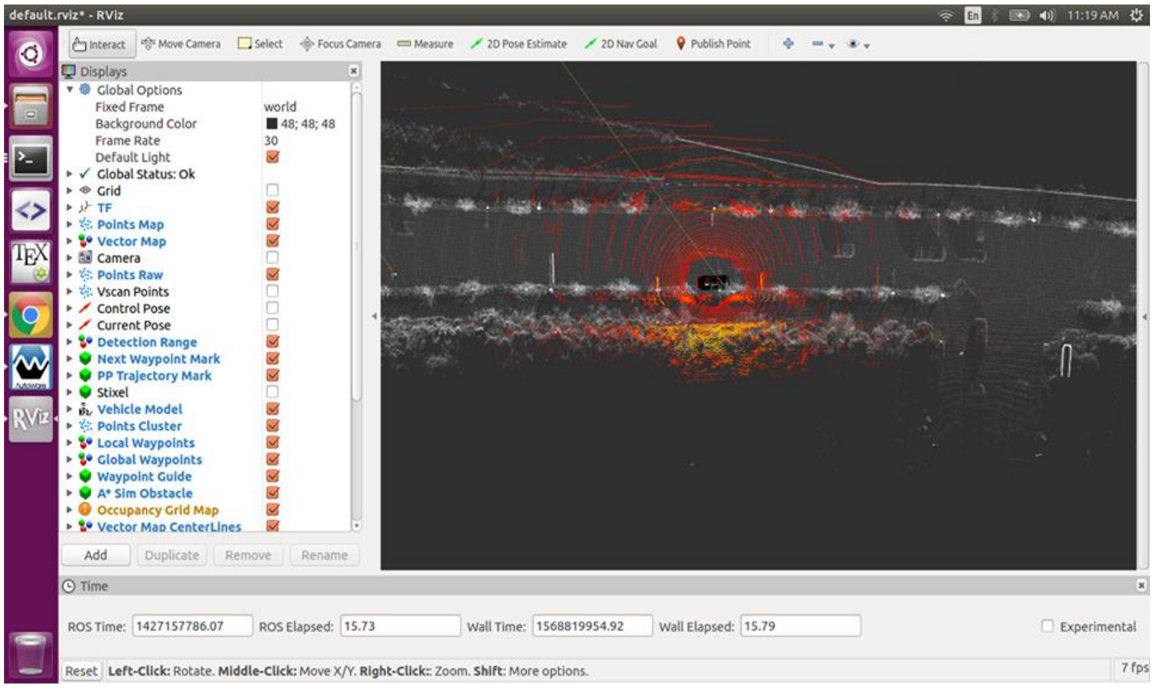
Figure 19: Selecting file default.rviz

Step 12: Unpause the simulation in the Simulation tab in Autoware and check whether the map is being shown up and whether the vehicle is moving or not.



```
/home/phuris/.autoware/sample_moriyama_150324.bag
Rate:      Start Time (s): 120    Repeat

path:     /home/phuris/.autoware/sample_moriyama_150324.bag
version:  2.0
duration: 7:59s (479s)
start:    Mar 23 2015 20:40:50.33 (1427157650.33)
end:      Mar 23 2015 20:48:49.50 (1427158129.50)
size:     7.9 GB
messages: 16768
compression: none [4788/4788 chunks]
types:    nmea_msgs/Sentence      [9f221efc5f4b3bac7ce4af102b32308b]
          sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:   /nmea_sentence  11980 msgs   : nmea_msgs/Sentence
          /points_raw     4788 msgs    : sensor_msgs/PointCloud2
```

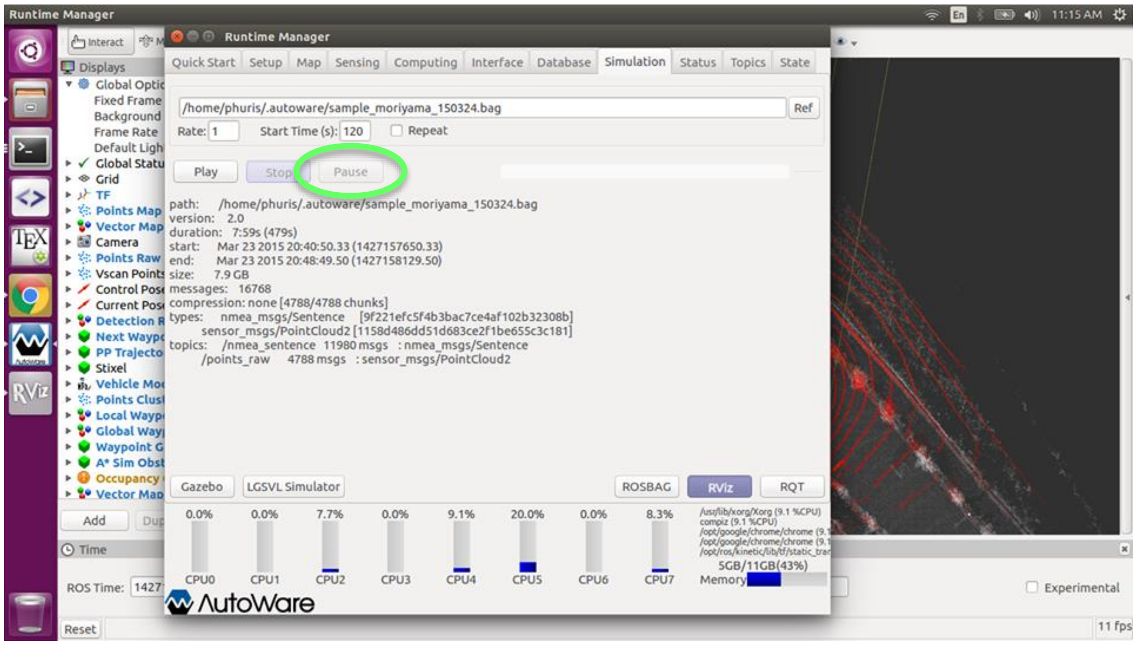
Figure 20: Unpause the Simulation



```
Global Options
  Fixed Frame        world
  Background Color   48; 48; 48
  Frame Rate         30
  Default Light      ✓
Global Status: Ok
Grid
TF                   ✓
Points Map           ✓
Vector Map           ✓
Camera
Points Raw           ✓
Vscan Points
Control Pose
Current Pose
Detection Range      ✓
Next Waypoint Mark   ✓
PP Trajectory Mark   ✓
Stixel
Vehicle Model        ✓
Points Cluster       ✓
Local Waypoints      ✓
Global Waypoints     ✓
Waypoint Guide       ✓
A* Sim Obstacle      ✓
Occupancy Grid Map   ✓
Vector Map CenterLines ✓

ROS Time: 1427157786.07  ROS Elapsed: 15.73  Wall Time: 1568819954.92  Wall Elapsed: 15.79  Experimental
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.  7 fps
```

Figure 21: The Movement of Vehicle

Step 13: If everything goes well, then pause the simulation again.



```
path:     /home/phuris/.autoware/sample_moriyama_150324.bag
version:   2.0
duration:   7:59s (479s)
start:    Mar 23 2015 20:40:50.33 (1427157650.33)
end:      Mar 23 2015 20:48:49.50 (1427158129.50)
size:     7.9 GB
messages:   16768
compression: none [4788/4788 chunks]
types:    nmea_msgs/Sentence     [9f221efc5f4b3bac7ce4af102b32308b]
          sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:   /nmea_sentence  11980 msgs   : nmea_msgs/Sentence
          /points_raw      4788 msgs   : sensor_msgs/PointCloud2
```

Figure 22: Pausing the Simulation

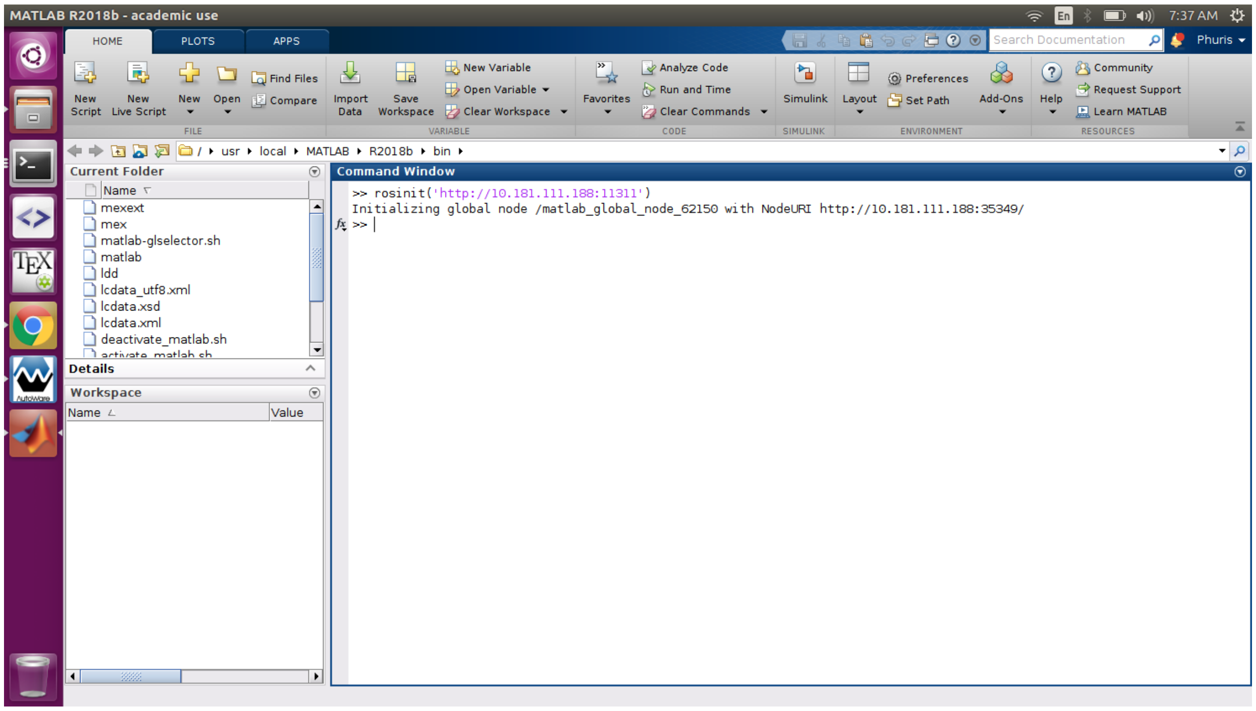
Step 14: Open MATLAB and enter the command

```
>> rosinit('ip_address')
```

for example:

```
>> rosinit ('http://192.168.1.1:12000')
```

This is to start a ROS node in MATLAB and register this node to the existing ROS network in the ip_address. If Autoware and MATLAB are running on the same computer, you may use only `rosinit`.



```
>> rosinit('http://10.181.111.188:11311')
Initializing global node /matlab_global_node_62150 with NodeURI http://10.181.111.188:35349/
>> |
```

Figure 23: Running the rosinit('ip_address') command on MATLAB

Step 15: Run the command

`>> rosrun rqt_graph rqt_graph` in the Ubuntu terminal to show the ROS network graph. You will see that there is a node from MATLAB showing in the graph.

Note: if you are not familiar with the rqt tool and your computer doesn't have this tool installed yet, please check the link below.

➢ http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

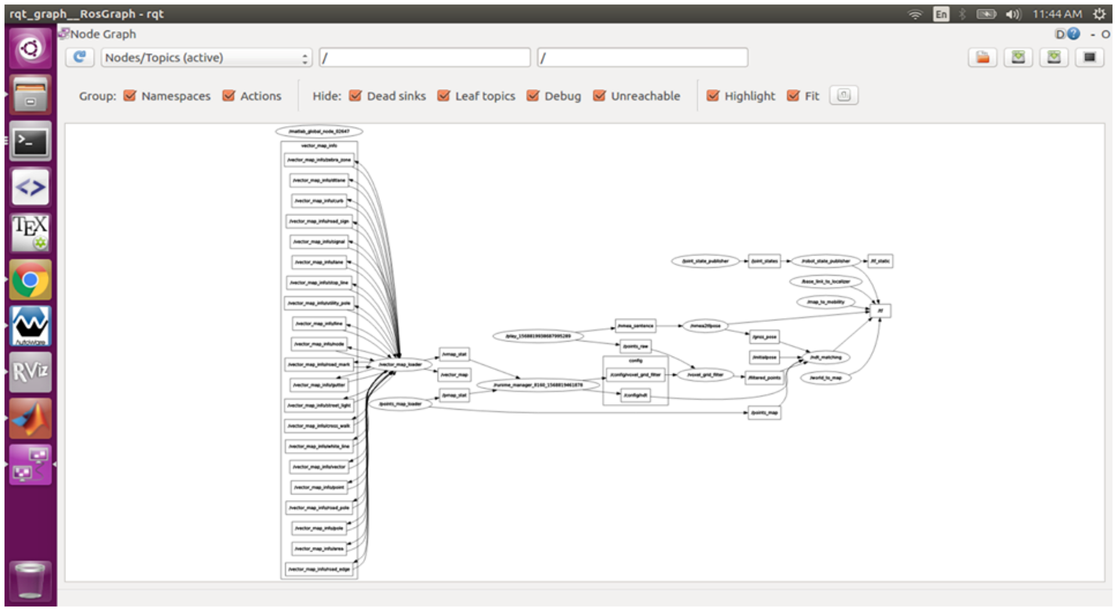

rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (active)
Group: Namespaces Actions
Hide: Dead sinks Leaf topics Debug Unreachable
Highlight Fit

Figure 24: A rqt Window

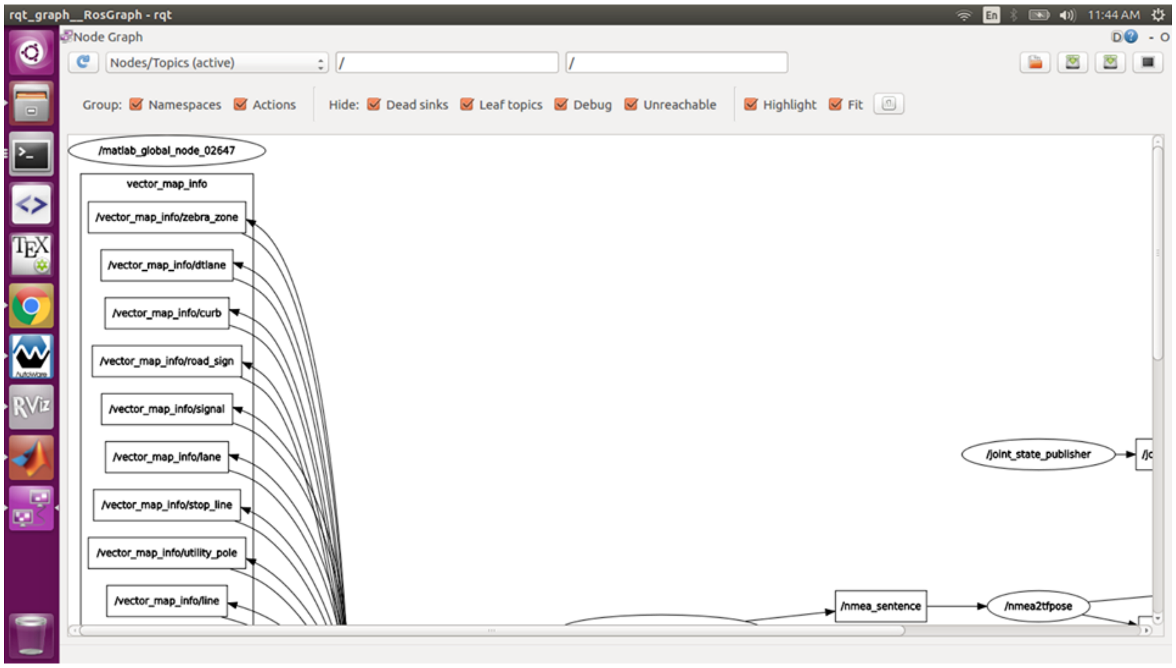
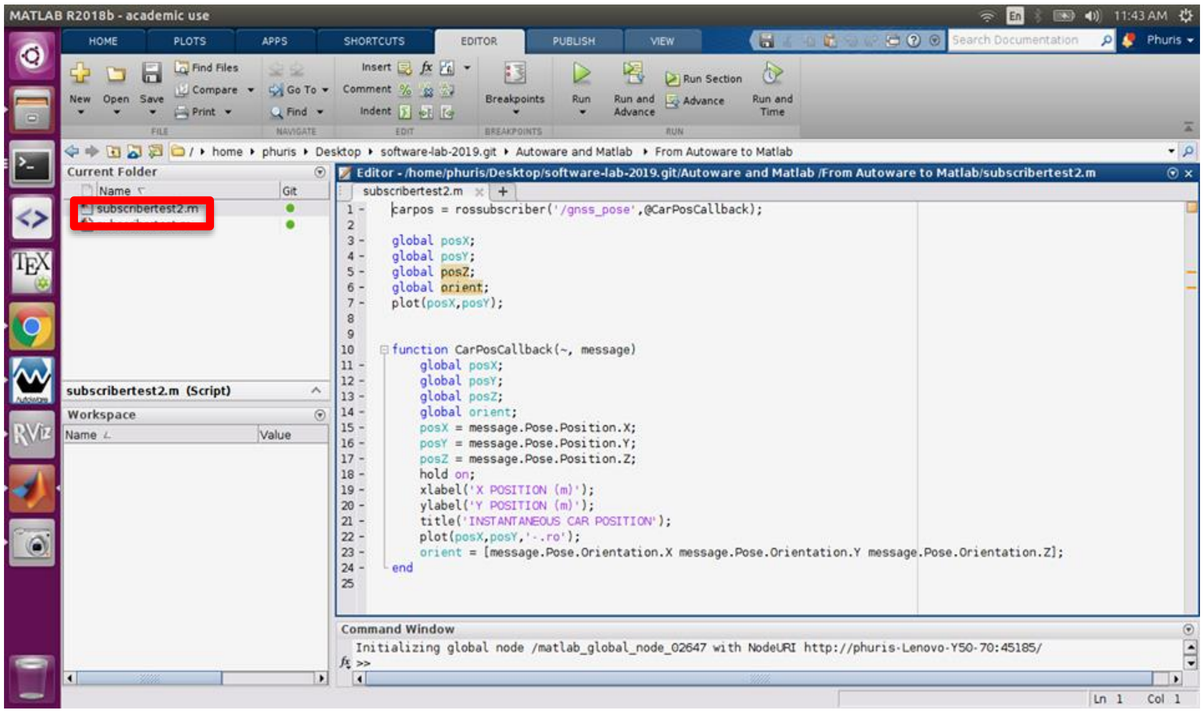

rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (active)
Group: Namespaces Actions
Hide: Dead sinks Leaf topics Debug Unreachable
Highlight Fit
/matlab_global_node_02647
vector_map_info
/vector_map_info/zebra_zone
/vector_map_info/dtlane
/vector_map_info/curb
/vector_map_info/road_sign
/vector_map_info/signal
/vector_map_info/lane
/vector_map_info/stop_line
/vector_map_info/utility_pole
/vector_map_info/line
/joint_state_publisher
/nmea_sentence
/nmea2tfpose

Figure 25: A rqt Window

Step 16: Lead MATLAB to the directory that store the subsribertest2.m file and add this directory to path. Double click the subscribertest2.m file in the Current Folder panel and run the script.



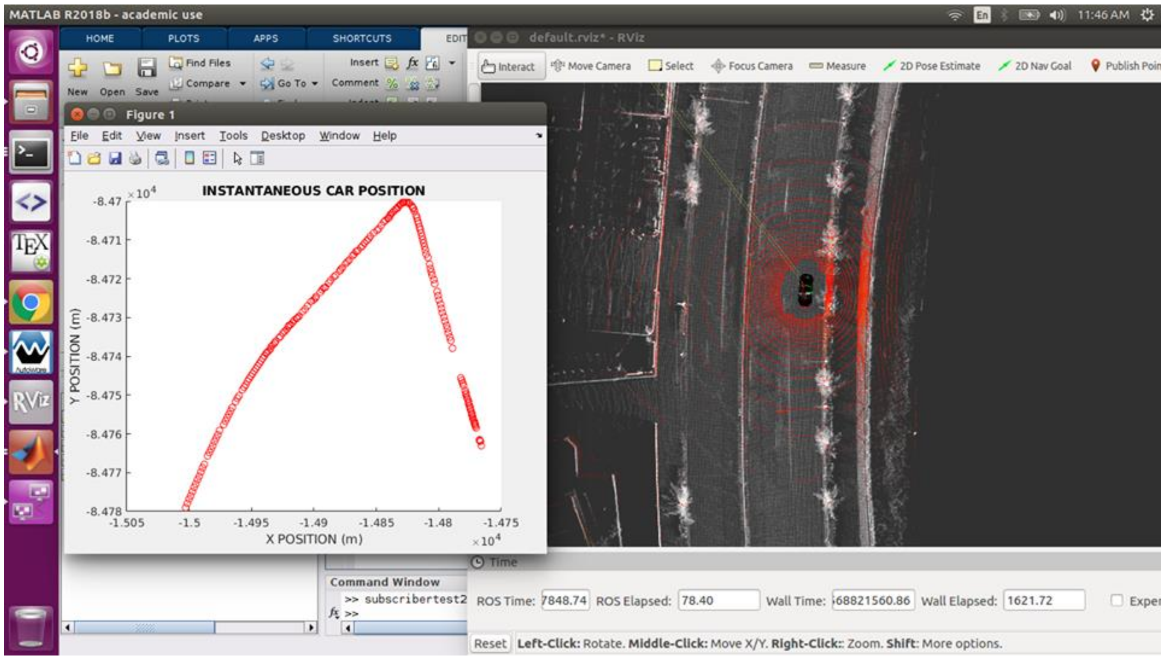
```
carpos = rossubscriber('/gnss_pose',@CarPosCallback);

global posX;
global posY;
global posZ;
global orient;
plot(posX,posY);


function CarPosCallback(~, message)
    global posX;
    global posY;
    global posZ;
    global orient;
    posX = message.Pose.Position.X;
    posY = message.Pose.Position.Y;
    posZ = message.Pose.Position.Z;
    hold on;
    xlabel('X POSITION (m)');
    ylabel('Y POSITION (m)');
    title('INSTANTANEOUS CAR POSITION');
    plot(posX,posY,'-.ro');
    orient = [message.Pose.Orientation.X message.Pose.Orientation.Y message.Pose.Orientation.Z];
end
```

```
Initializing global node /matlab_global_node_02647 with NodeURI http://phuris-Lenovo-Y50-70:45185/
>>
```

MATLAB R2018b - academic use
HOME
PLOTS
APPS
SHORTCUTS
EDITOR
PUBLISH
VIEW
Current Folder
subscribertest2.m
subscribertest2.m (Script)
Workspace
Editor - /home/phuris/Desktop/software-lab-2019.git/Autoware and Matlab /From Autoware to Matlab/subscribertest2.m
Command Window

Figure 26: Running subscribertest2.m file

Step 17: Go to Autoware and play the simulation. The instantaneous car position plot will be shown.



```
>> subscribertest2
>>
```

ROS Time: 7848.74 ROS Elapsed: 78.40 Wall Time: 68821560.86 Wall Elapsed: 1621.72

Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.

Figure 27: Instantaneous Car Position Plot

Step 18: Go to the ROS network graph, refresh the graph, you will see that now the MATLAB ROS node is subscribing the data from the topic `/gnss_pose` from Autoware.

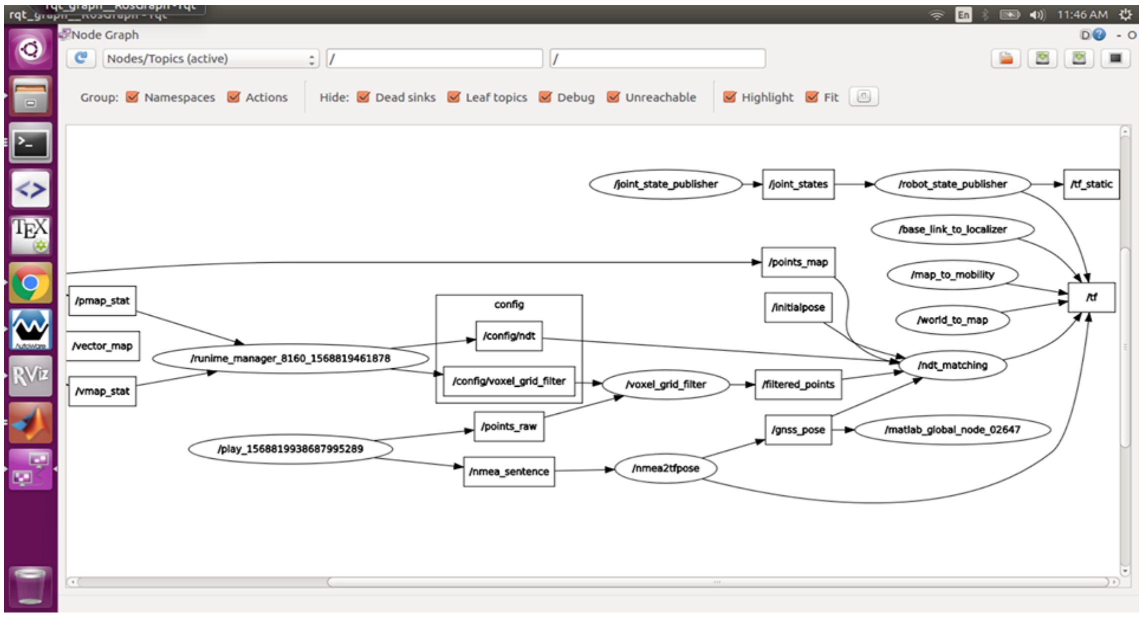
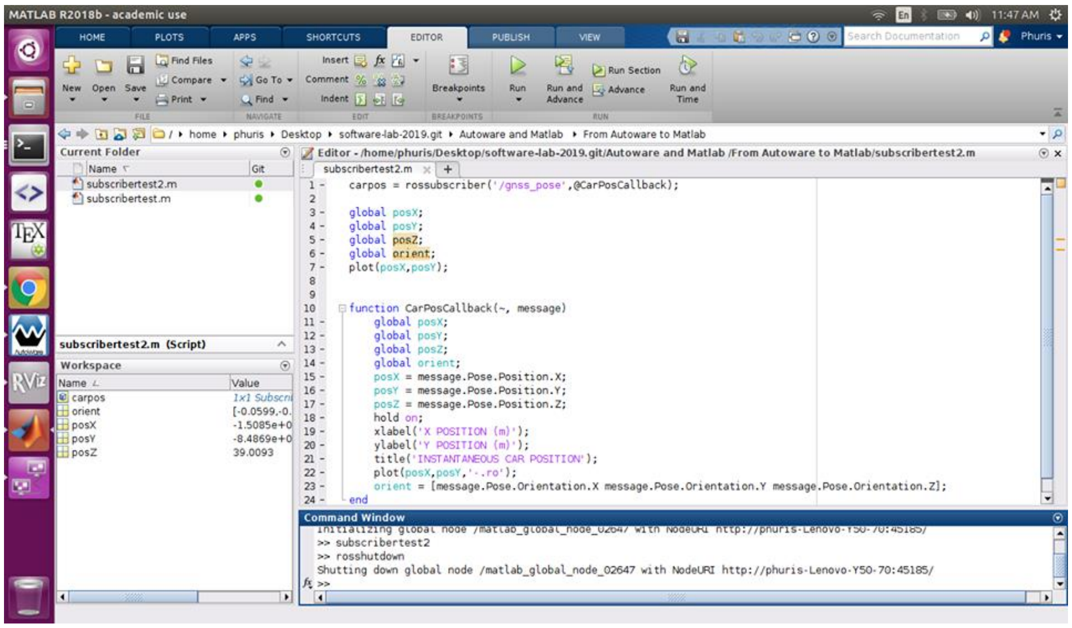


Figure 28: Subscription of Data from the topic /gnss_pose from Autoware

Step 19: To turn off all the program, stop the simulation in Autoware. In MATLAB, type the command

```
>> rosshutdown
```

and close MATLAB window. Then close RVIZ and Autoware respectively. Now if you refresh the ROS network graph, there should be nothing.



```
Editor - /home/phuris/Desktop/software-lab-2019.git/Autoware and Matlab /From Autoware to Matlab/subscribertest2.m
```

```
carpos = rossubscriber('/gnss_pose',@CarPosCallback);

global posX;
global posY;
global posZ;
global orient;
plot(posX,posY);


function CarPosCallback(~, message)
    global posX;
    global posY;
    global posZ;
    global orient;
    posX = message.Pose.Position.X;
    posY = message.Pose.Position.Y;
    posZ = message.Pose.Position.Z;
    hold on;
    xlabel('X POSITION (m)');
    ylabel('Y POSITION (m)');
    title('INSTANTANEOUS CAR POSITION');
    plot(posX,posY,'-.ro');
    orient = [message.Pose.Orientation.X message.Pose.Orientation.Y message.Pose.Orientation.Z];
end
```

Name	Value
carpos	1x1 Subscri
orient	[-0.0599,-0.
posX	-1.5085e+0
posY	-8.4869e+0
posZ	39.0093

```
Initializing global node /matlab_global_node_02647 with NodeURI http://phuris-Lenovo-Y50-70:45185/
>> subscribertest2
>> rosshutdown
Shutting down global node /matlab_global_node_02647 with NodeURI http://phuris-Lenovo-Y50-70:45185/
>>
```

Figure 29: Stopping the Simulation

Chapter 6: Co-simulation Between Autoware and MATLAB with a Simulink Model

In this chapter, a detailed guide will be provided which is intended to demonstrate co-simulation between Autoware and MATLAB & Simulink. A Simulink model consisting of subsystems for twist filter, vel pose connect, and plot of instantaneous ego vehicles' position along with MATLAB file for wf simulator were run together in conjunction with Autoware to demonstrate interconnectivity between MATLAB and Autoware.

It should be noted that Autoware's version 1.8.0 has been utilized for this task in order to avoid compatibility issues. Furthermore, Autoware Toolbox was specifically implemented for this version.

The following steps outline the complete process of running a Simulink Model in conjunction with Autoware. For complete and detailed overview of Autoware installation, please visit the following link:

- https://gitlab.com/autowarefoundation/autoware.ai/autoware/wikis/Source-Build

In order to avoid installation error, after step 3 under "For version 1.11.1 or older" (namely install dependencies using rosdep) in the link above, please run the following additional command provided below (in red) which is missing from the documentation on Autoware's website.

```
$ rosdep update
$ git submodule update --init –recursive
$ rosdep install -y --from-paths src --ignore-src --rosdistro $ROS_DISTRO
```

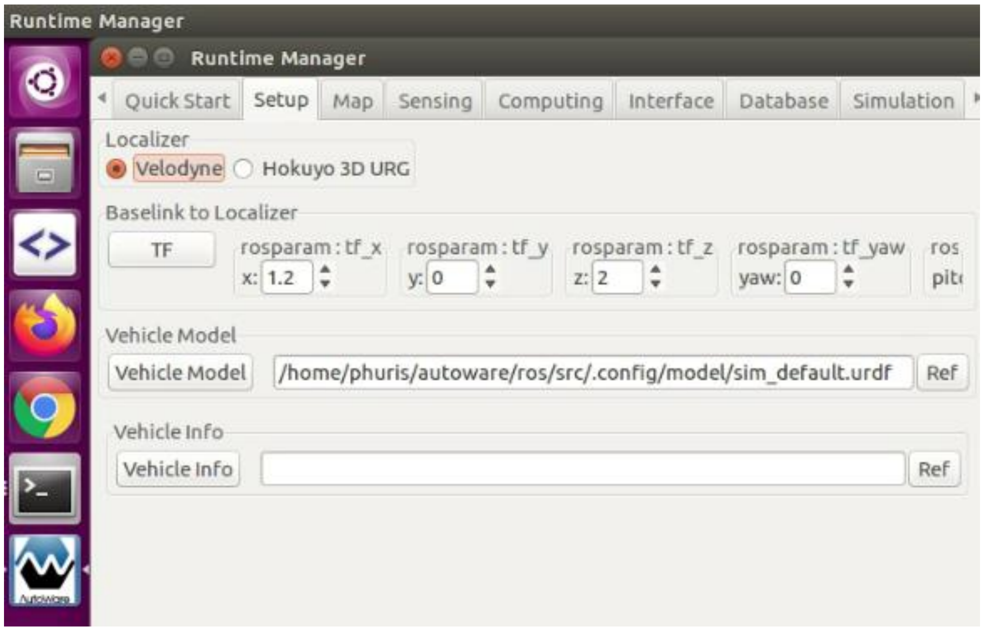
The rest of the process is the same as in the link above.

Furthermore, ***computing.yaml*** file must be modified slightly to include the two of the additional computing modules that are not available in the newer versions such as 1.8.0 or 1.12.0. These two modules are named as "way_planner" and "dp_planner". This ***computing.yaml*** file is found in Autoware folder, and then the parts related to "way_planner" and "dp_planner" must be uncommented so that they can appear in Autoware GUI.

Step 1: Run Autoware version 1.8.0 using the following commands:

```
$ cd autoware/ros
$ source devel/setup.bash
$ ./run
```

Step 2: After the Autoware Runtime manager window pops up on the screen, under the "Setup" tab, Vehicle Model will be selected. The path to the file for vehicle model can also be seen in the following screenshot. Vehicle Model button can be clicked to select it.



Localizer: [✓ Velodyne] Hokuyo 3D URG

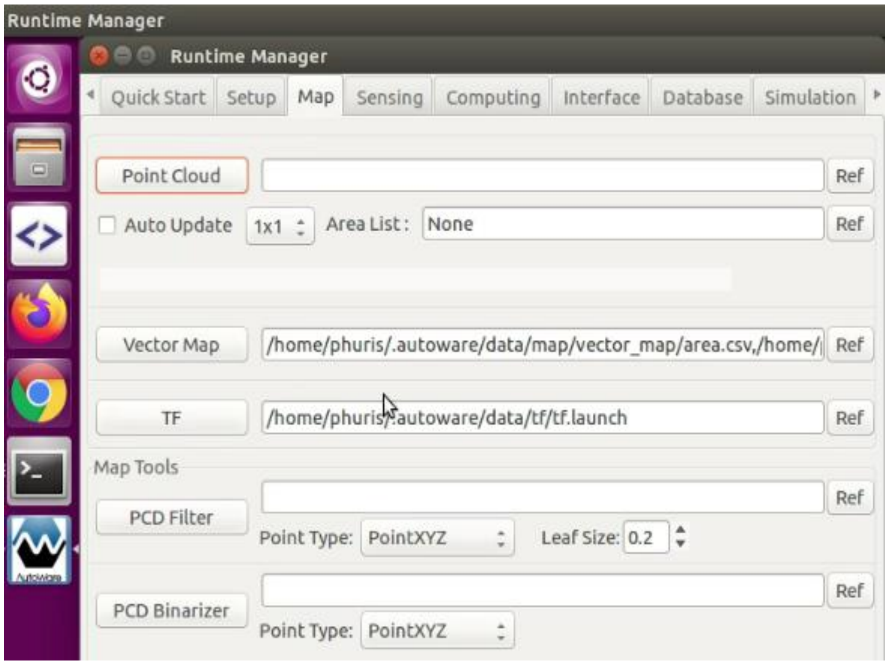
Baselink to Localizer: rosparam : tf_x x: 1.2; rosparam : tf_y y: 0; rosparam : tf_z z: 2; rosparam : tf_yaw yaw: 0

Vehicle Model: /home/phuris/autoware/ros/src/.config/model/sim_default.urdf

Vehicle Info:

Figure 30: A screenshot of the Setup Tab to select Vehicle Model

Step 3: Next step is to setup the "Map" tab. The file paths used for Vector Map and TF can be seen on the screenshot that follows.



Point Cloud:

Auto Update 1x1 Area List: None

Vector Map: /home/phuris/.autoware/data/map/vector_map/area.csv,/home/

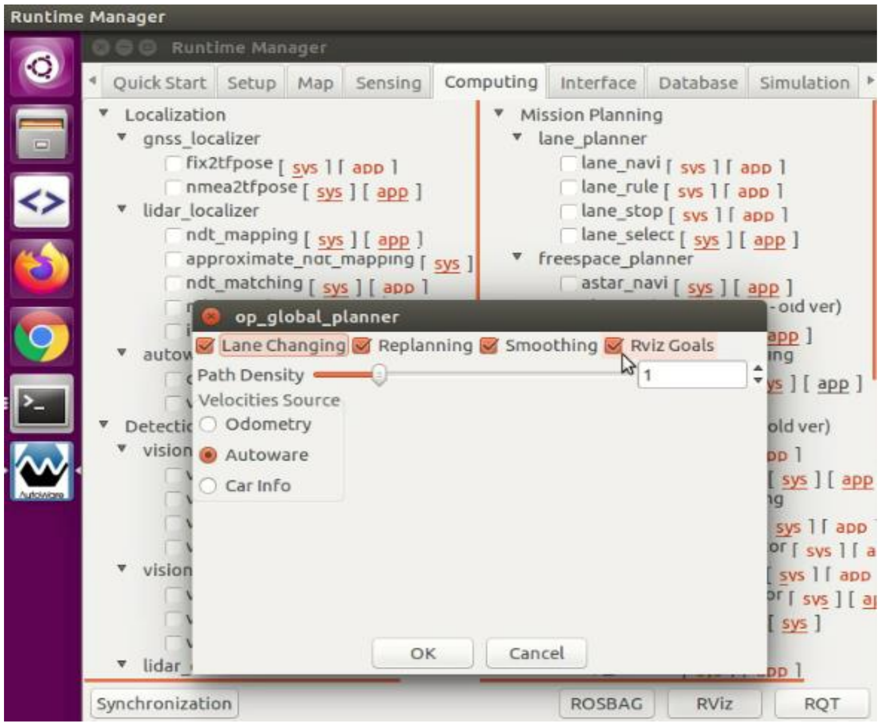
TF: /home/phuris/.autoware/data/tf/tf.launch

PCD Filter: Point Type: PointXYZ Leaf Size: 0.2

PCD Binarizer: Point Type: PointXYZ

Figure 31: A screenshot of Map tab

Step 4a: In the computing tab, we will first click on the *op_global_planner* app and perform following modifications so that the relevant options are selected.


Runtime Manager
Quick Start
Setup
Map
Sensing
Computing
Interface
Database
Simulation
Localization
gnss_localizer
fix2tfpose [sys] [app]
nmea2tfpose [sys] [app]
lidar_localizer
ndt_mapping [sys] [app]
approximate_ndt_mapping [sys]
ndt_matching [sys] [app]
Mission Planning
lane_planner
lane_navi [sys] [app]
lane_rule [sys] [app]
lane_stop [sys] [app]
lane_select [sys] [app]
freespace_planner
astar_navi [sys] [app]
OK
Cancel
Synchronization
ROSBAG
RViz
RQT

op_global_planner
[✓ Lane Changing] [✓ Replanning] [✓ Smoothing] [✓ Rviz Goals]
Path Density 1
Velocities Source
Odometry [✓ Autoware] Car Info

Figure 32: Selections for op_global_planner app

Step 4b: Now that the *op_global_planner* has been checked, we will open Rviz and set *2D Pose Estimate* and *2D Nav Goal* from the toolbar as shown below.

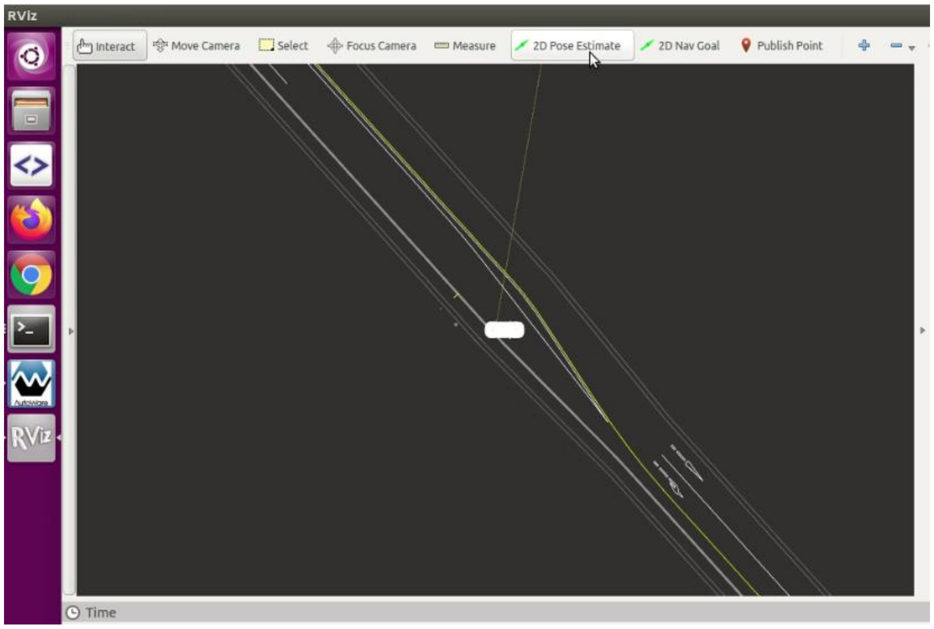
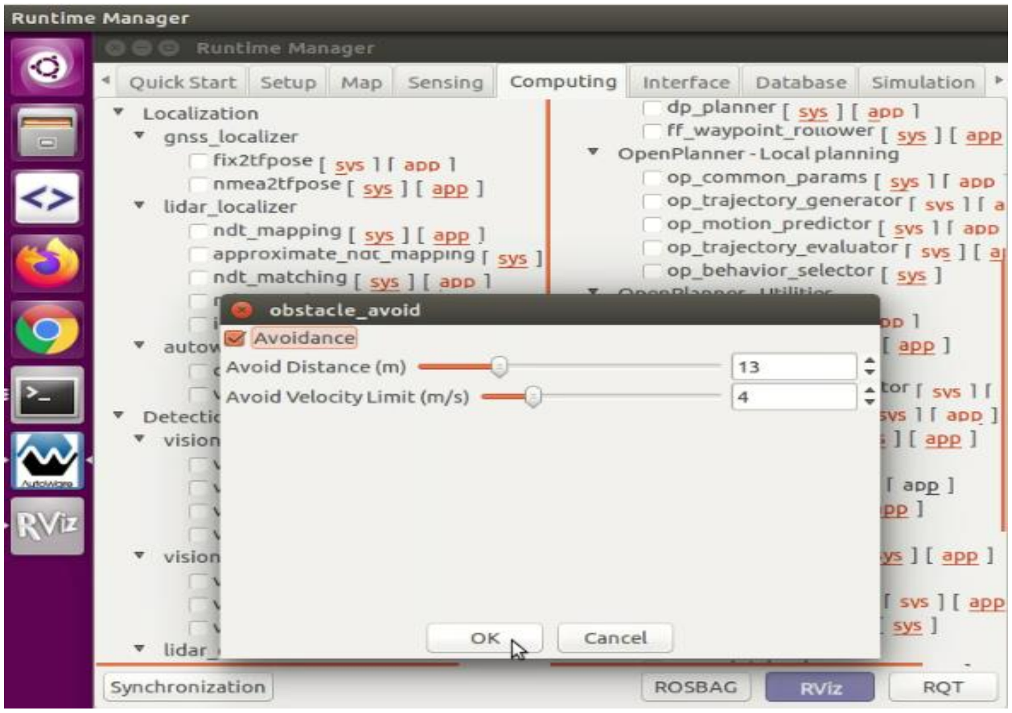


Figure 33: Initial RViz setup showing 2D Pose Estimate and 2D Nav Goal

By setting up 2D Pose Estimate and 2D Nav Goal, a three-lane path in blue will appear on the map in RViz.

Step 4c: Please now select the *obstacle_avoid* app and *velcoity_set* app, and make sure that the relevant options are selected similar to the photos below.



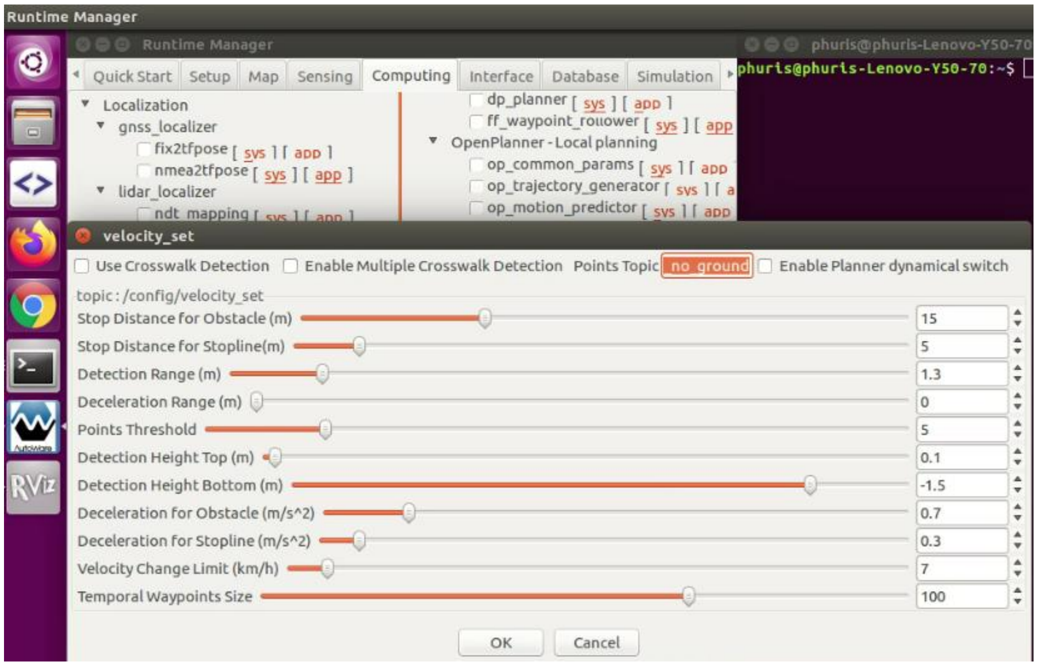
obstacle_avoid

[✓] Avoidance

Avoid Distance (m) 13

Avoid Velocity Limit (m/s) 4

Figure 34: Obstacle_avoid app



velocity_set

[ ] Use Crosswalk Detection [ ] Enable Multiple Crosswalk Detection Points Topic no_ground [ ] Enable Planner dynamical switch

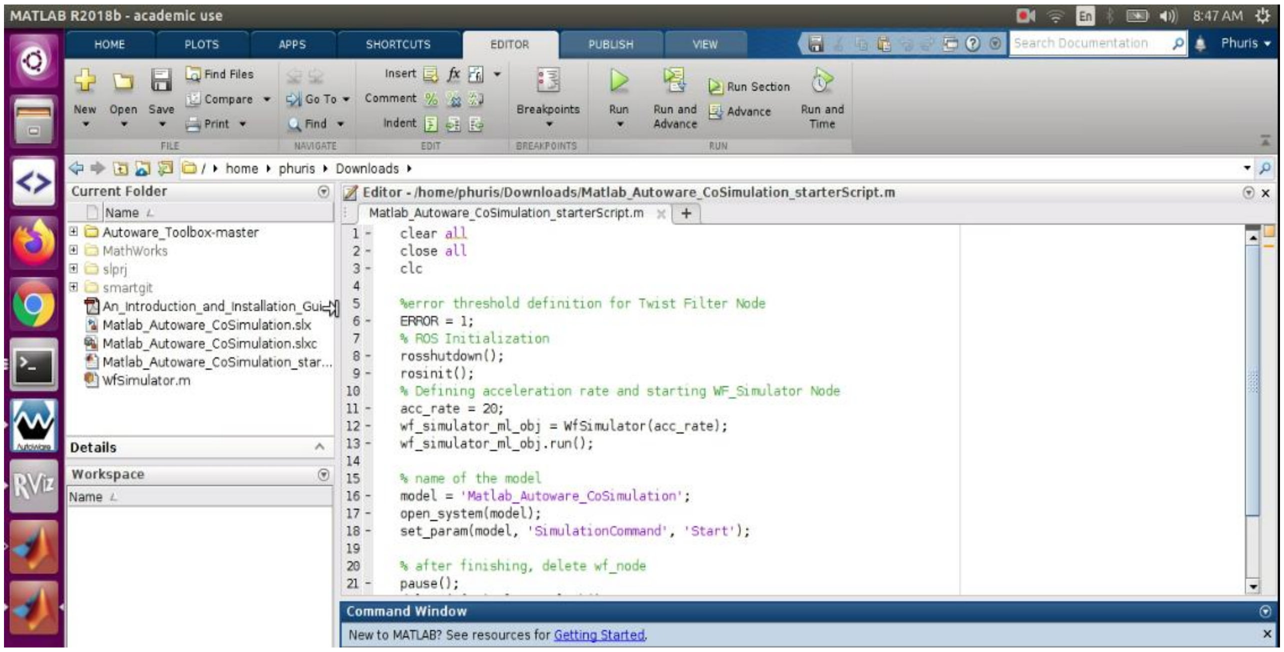
topic: /config/velocity_set

Stop Distance for Obstacle (m)	15
Stop Distance for Stopline(m)	5
Detection Range (m)	1.3
Deceleration Range (m)	0
Points Threshold	5
Detection Height Top (m)	0.1
Detection Height Bottom (m)	-1.5
Deceleration for Obstacle (m/s^2)	0.7
Deceleration for Stopline (m/s^2)	0.3
Velocity Change Limit (km/h)	7
Temporal Waypoints Size	100

Figure 35: velocity_set app

Step 5: Now that the previously mentioned three apps have already been selected, we will run MATLAB in the background.

Step 6: Please open the MATLAB file named as *Matlab_Autoware_CoSimulation_starterScript.m* and run it. Now, the Simulink will be opened.



/ ▸ home ▸ phuris ▸ Downloads ▸

Autoware_Toolbox-master
MathWorks
slprj
smartgit
An_Introduction_and_Installation_Gui...
Matlab_Autoware_CoSimulation.slx
Matlab_Autoware_CoSimulation.slxc
Matlab_Autoware_CoSimulation_star...
WfSimulator.m

Editor - /home/phuris/Downloads/Matlab_Autoware_CoSimulation_starterScript.m

```
clear all
close all
clc

%error threshold definition for Twist Filter Node
ERROR = 1;
% ROS Initialization
rosshutdown();
rosinit();
% Defining acceleration rate and starting WF_Simulator Node
acc_rate = 20;
wf_simulator_ml_obj = WfSimulator(acc_rate);
wf_simulator_ml_obj.run();

% name of the model
model = 'Matlab_Autoware_CoSimulation';
open_system(model);
set_param(model, 'SimulationCommand', 'Start');

% after finishing, delete wf_node
pause();
```

New to MATLAB? See resources for Getting Started.

Figure 36: Matlab_Autoware_CoSimulation_starterScript.

We will notice that the Simulink is now being run.

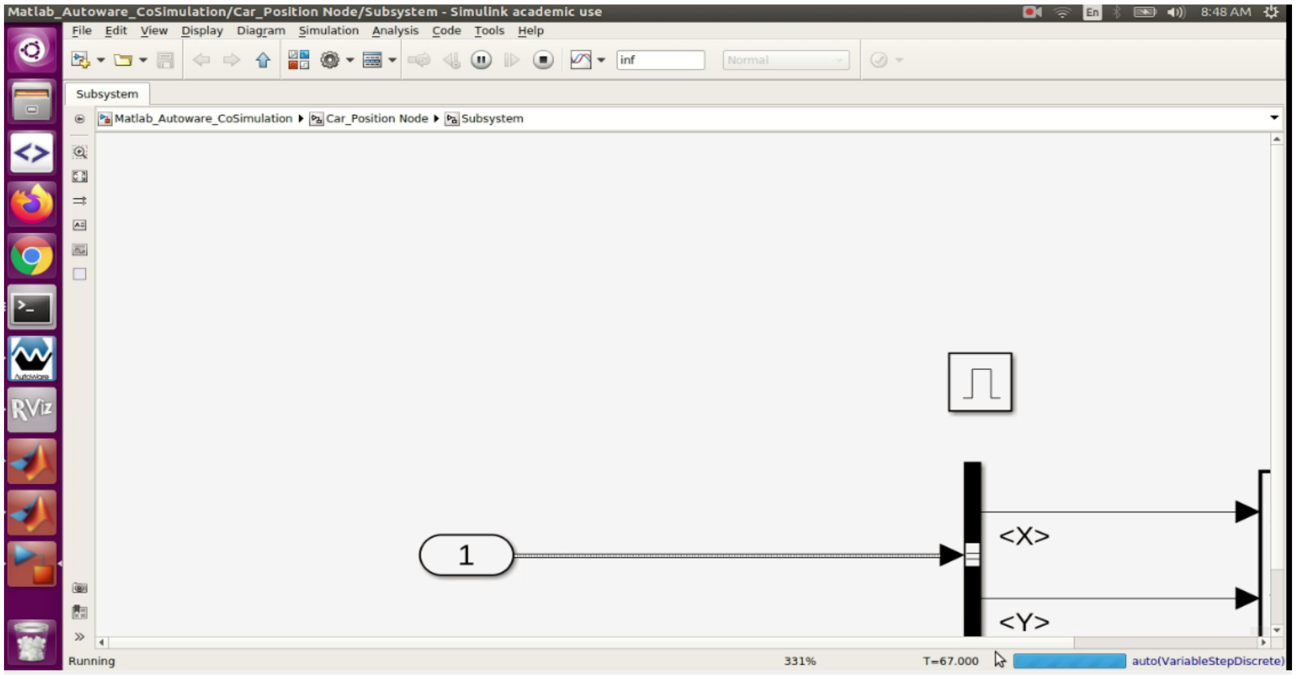

Matlab_Autoware_CoSimulation/Car_Position Node/Subsystem - Simulink academic use
File
Edit
View
Display
Diagram
Simulation
Analysis
Code
Tools
Help
inf
Normal
Subsystem
Matlab_Autoware_CoSimulation ▸ Car_Position Node ▸ Subsystem
1
<X>
<Y>
Running
331%
T=67.000
auto(VariableStepDiscrete)

Figure 37: Simulink Model being run

Step 7: While the Simulink is being run, we will go to RViz and visualize our car by setting up 2D Pose Estimate once again, and we will notice that a black car (ego vehicle) appears on the map as shown below.

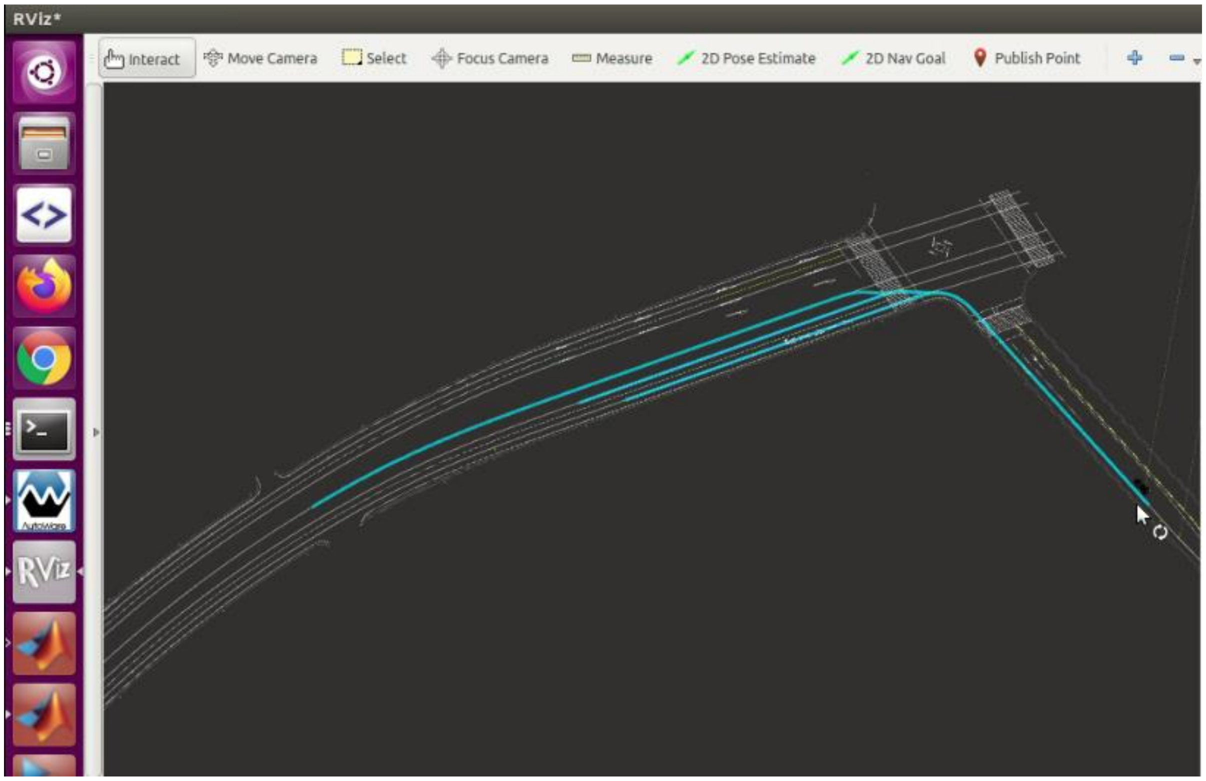
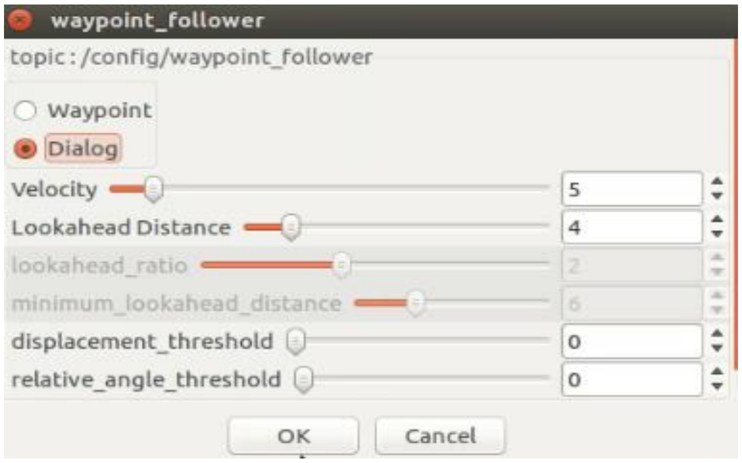


Figure 38: Visualizing our Car (Ego Vehicle)

Step 8: After visualizing our car, we will select the *pure_pursuit* app from *way point follower* under computing tab and make sure that it looks like the photo below and that all the options are selected as follows.


waypoint_follower

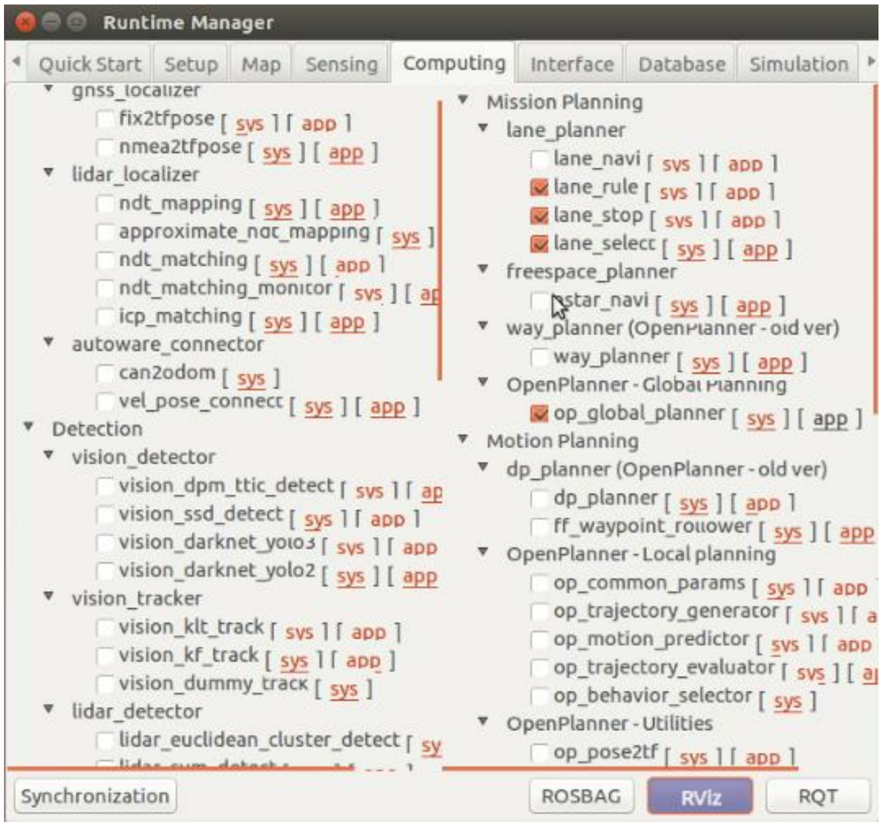
topic : /config/waypoint_follower

( ) Waypoint [✓ Dialog]

Velocity	5
Lookahead Distance	4
lookahead_ratio	2
minimum_lookahead_distance	6
displacement_threshold	0
relative_angle_threshold	0

Figure 39: Pure_Pursuit app

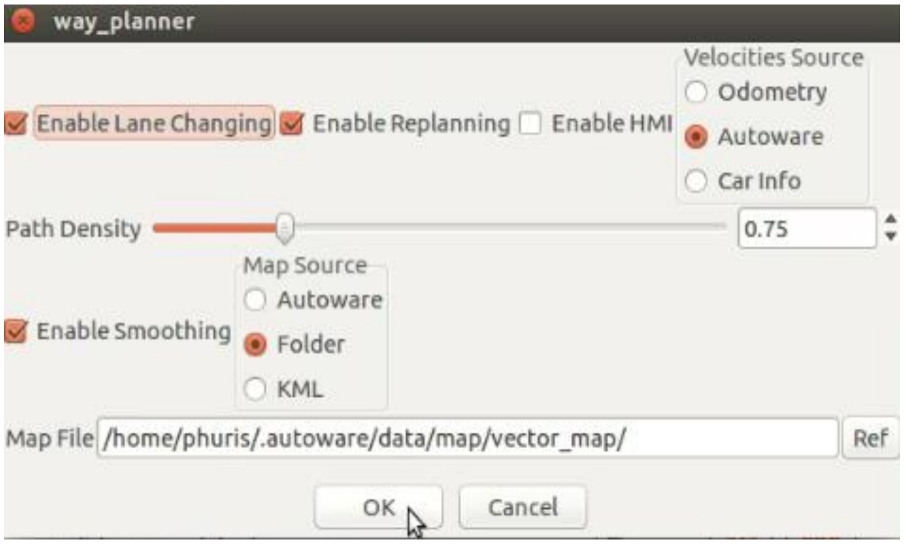
Step 9: Now we will select three *lane_planner* apps under computing tab namely *lane_rule, lane_stop,* and *lane_select.*



- gnss_localizer
  - [ ] fix2tfpose [sys] [app]
  - [ ] nmea2tfpose [sys] [app]
- lidar_localizer
  - [ ] ndt_mapping [sys] [app]
  - [ ] approximate_ndt_mapping [sys]
  - [ ] ndt_matching [sys] [app]
  - [ ] ndt_matching_monitor [sys] [ap
  - [ ] icp_matching [sys] [app]
- autoware_connector
  - [ ] can2odom [sys]
  - [ ] vel_pose_connect [sys] [app]
- Detection
  - vision_detector
    - [ ] vision_dpm_ttic_detect [sys] [ap
    - [ ] vision_ssd_detect [sys] [app]
    - [ ] vision_darknet_yolo3 [sys] [app
    - [ ] vision_darknet_yolo2 [sys] [app
  - vision_tracker
    - [ ] vision_klt_track [sys] [app]
    - [ ] vision_kf_track [sys] [app]
    - [ ] vision_dummy_track [sys]
  - lidar_detector
    - [ ] lidar_euclidean_cluster_detect [sy
- Mission Planning
  - lane_planner
    - [ ] lane_navi [sys] [app]
    - [x] lane_rule [sys] [app]
    - [x] lane_stop [sys] [app]
    - [x] lane_select [sys] [app]
  - freespace_planner
    - [ ] astar_navi [sys] [app]
  - way_planner (OpenPlanner - old ver)
    - [ ] way_planner [sys] [app]
  - OpenPlanner - Global Planning
    - [x] op_global_planner [sys] [app]
- Motion Planning
  - dp_planner (OpenPlanner - old ver)
    - [ ] dp_planner [sys] [app]
    - [ ] ff_waypoint_follower [sys] [app
  - OpenPlanner - Local planning
    - [ ] op_common_params [sys] [app
    - [ ] op_trajectory_generator [sys] [a
    - [ ] op_motion_predictor [sys] [app
    - [ ] op_trajectory_evaluator [sys] [a
    - [ ] op_behavior_selector [sys]
  - OpenPlanner - Utilities
    - [ ] op_pose2tf [sys] [app]

Figure 40: Selecting lane_planner apps

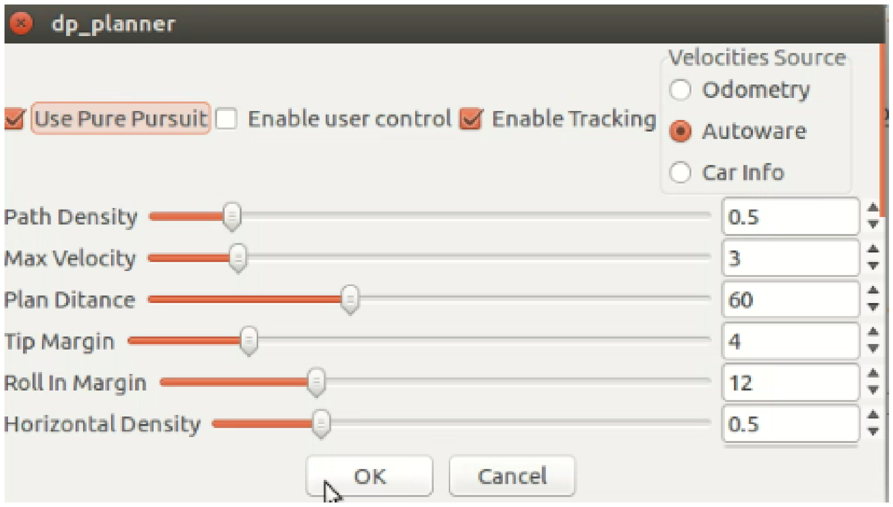
Step 10: We then select *way_planner* app.



- [x] Enable Lane Changing
- [x] Enable Replanning
- [ ] Enable HMI
- Velocities Source: Odometry [✓ Autoware] Car Info
- Path Density: 0.75
- [x] Enable Smoothing
- Map Source: Autoware [✓ Folder] KML
- Map File: /home/phuris/.autoware/data/map/vector_map/

Figure 41: way_planner app

Step 11: Please now select *dp_planner* app as follows.



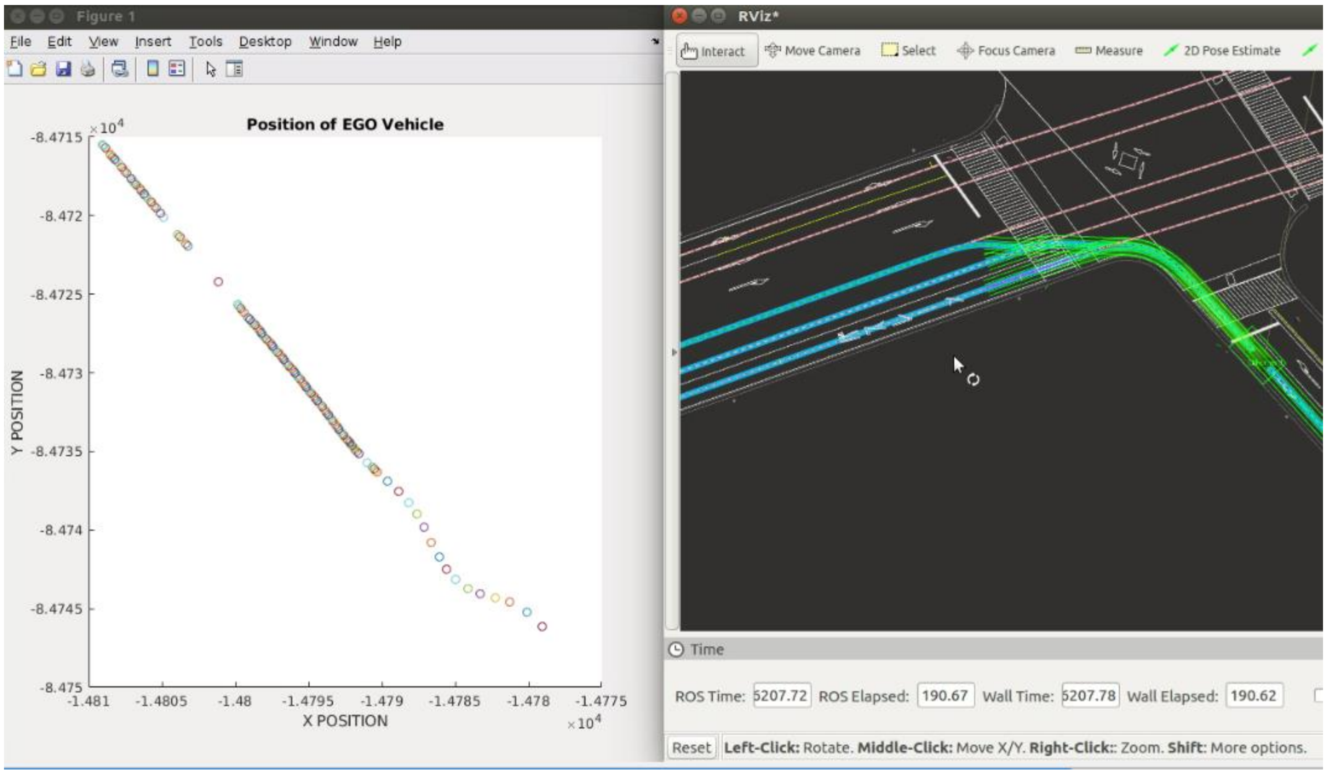
[✓ Use Pure Pursuit] Enable user control [✓ Enable Tracking]

Velocities Source: Odometry [✓ Autoware] Car Info

Path Density	0.5
Max Velocity	3
Plan Ditance	60
Tip Margin	4
Roll In Margin	12
Horizontal Density	0.5

Figure 42: dp_planner app

Step 12: Now, we will notice the movement of the car on the map in RViz, and the same time we can view the position of this ego vehicle on the plot as shown below. Next step would be to put an obstacle in the path of the vehicle to make sure that the car avoids the obstacle and takes the best possible path.

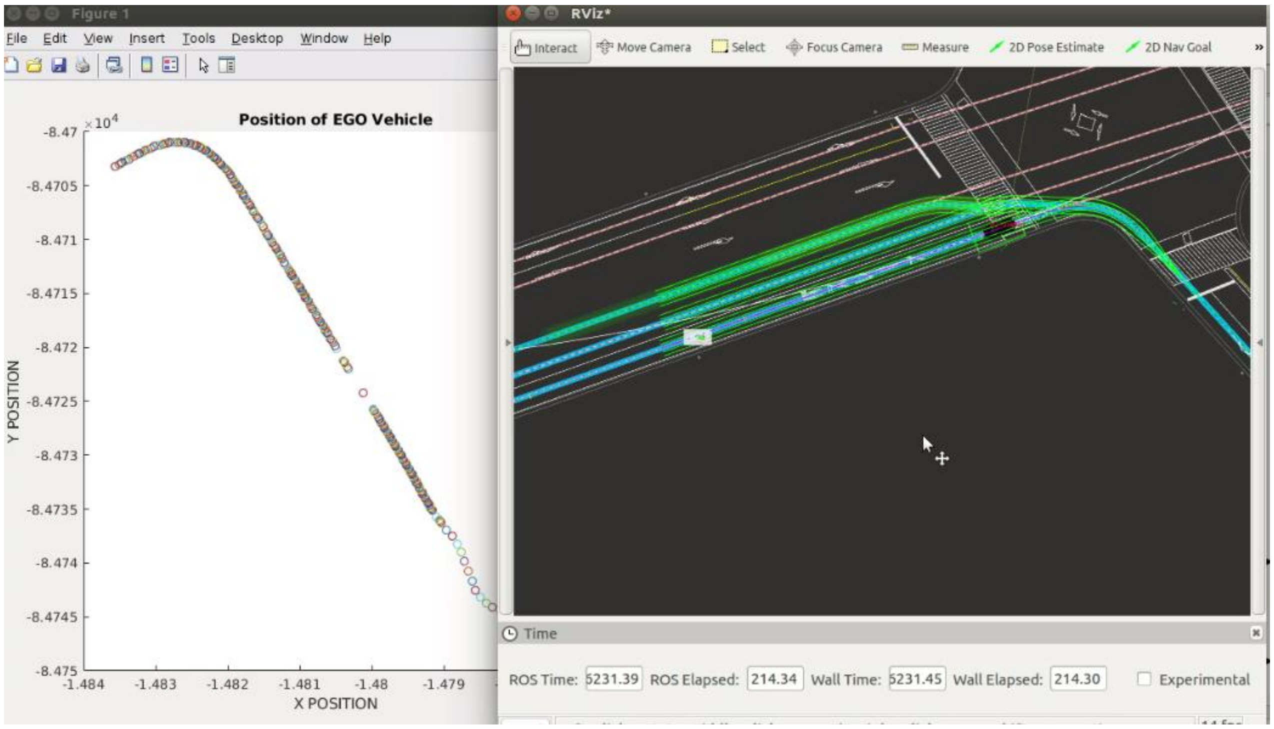


ROS Time: 5207.72 ROS Elapsed: 190.67 Wall Time: 5207.78 Wall Elapsed: 190.62

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift:** More options.

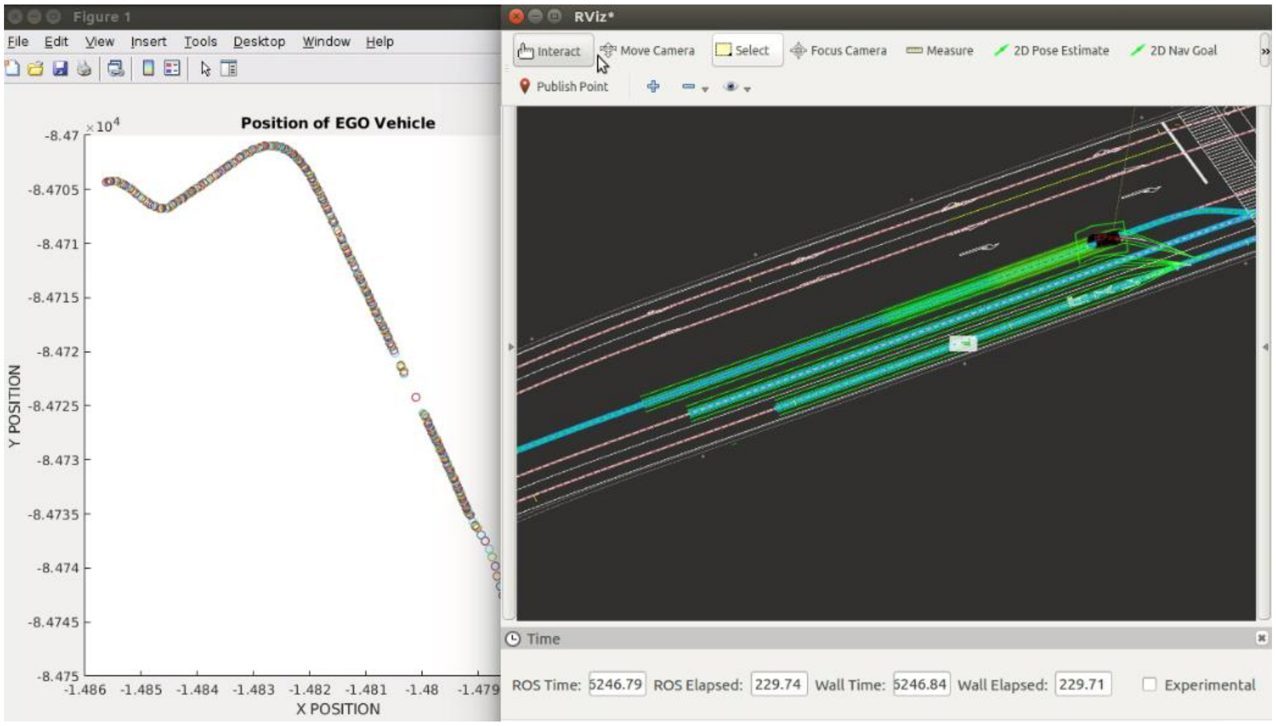
Figure 43: The car's movement visualized on RViz and MATLAB Simulink simultaneously

We can put an obstacle on the path by selecting “Publish Point” from the toolbar in RViz. We can notice that the car changes its path after few seconds, and this can also be seen from the Position of Ego vehicle plot.



ROS Time: 5231.39 ROS Elapsed: 214.34 Wall Time: 5231.45 Wall Elapsed: 214.30

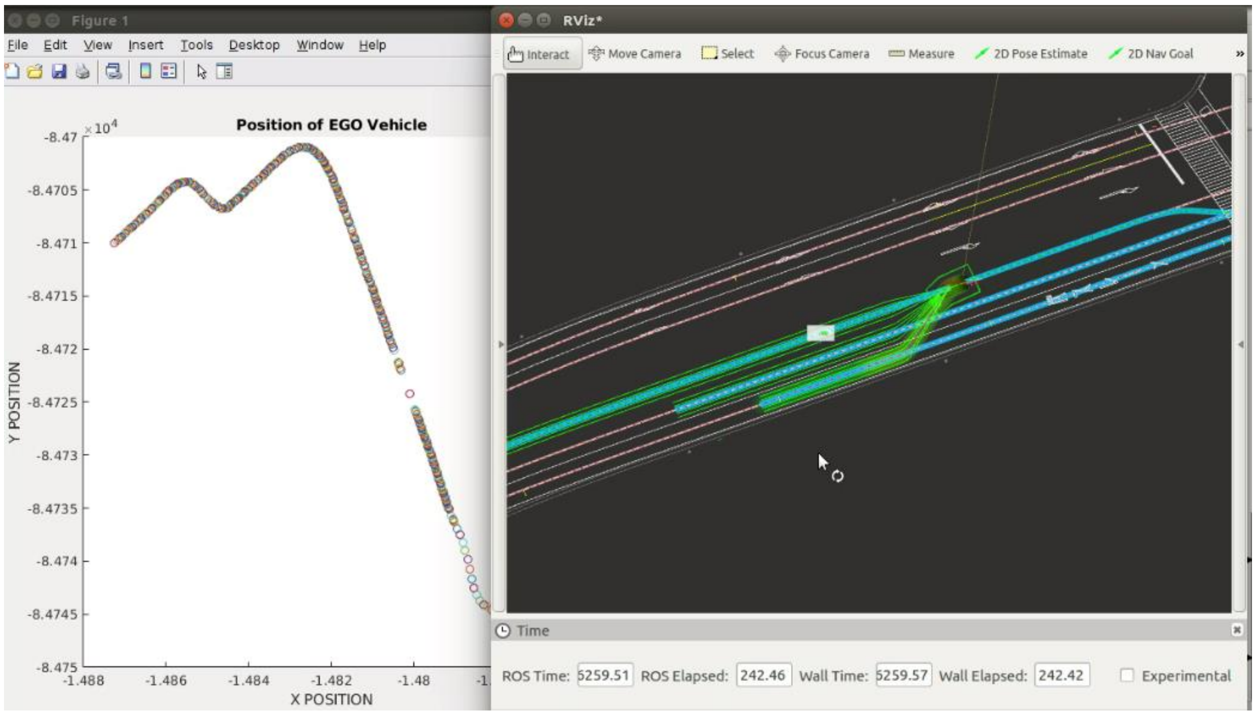
Figure 44: Putting an obstacle (in white) on the path



ROS Time: 5246.79 ROS Elapsed: 229.74 Wall Time: 5246.84 Wall Elapsed: 229.71

Figure 45: The car avoids the obstacle

We then put another obstacle on the path, and again the car avoids it as can be seen below on the plot of Position of ego vehicle.



ROS Time: 5259.51 ROS Elapsed: 242.46 Wall Time: 5259.57 Wall Elapsed: 242.42

Figure 46: Obstacle avoidance

Chapter 7: Implementation of a Manual Mode to Control the Ego Vehicle with a Keyboard

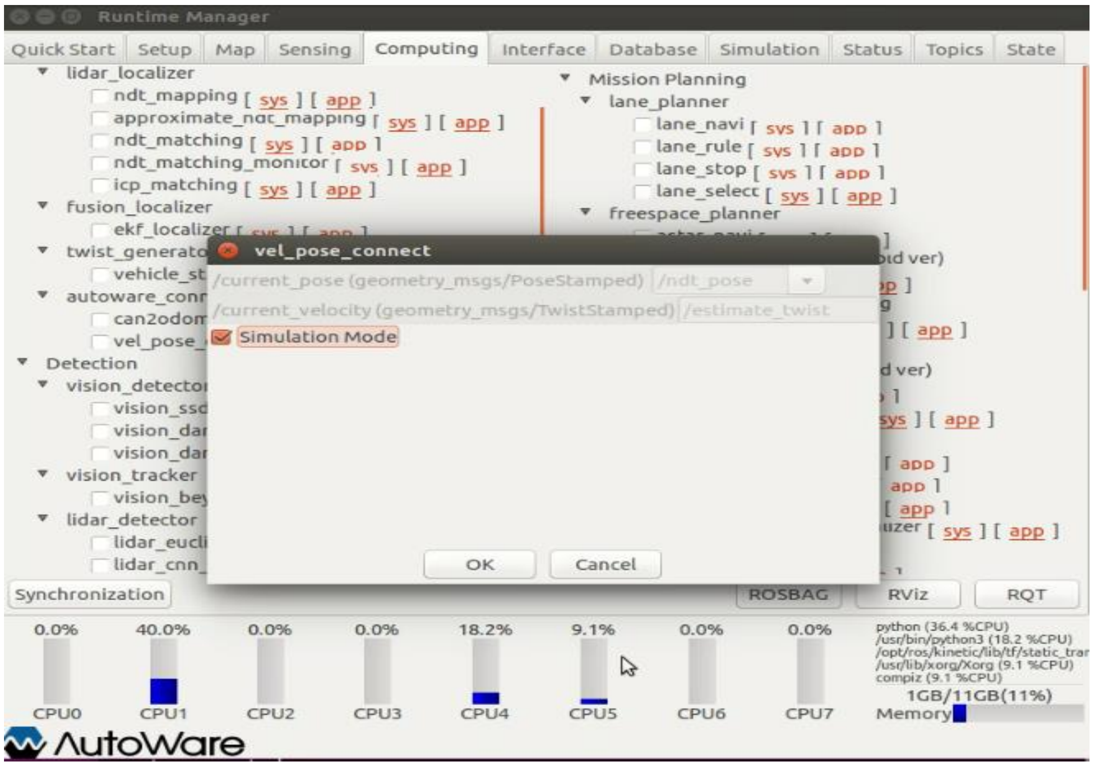
This chapter deals with an implementation of a manual mode so that an ego vehicle can be driven via a keyboard in RViz environment. This guide will provide the readers a detailed approach on how to control the vehicle from a keyboard. However, it should be noted that some initial commands that were run on the terminal are very similar to the ones explained in the previous chapter (Chapter 6), and they will not be repeated here to avoid repetition, but a reference to the figure number and step number will certainly be provided.

Step 1: We will launch the Autoware runtime manager from the terminal. Please refer to ***Step1*** in chapter 6 for the detailed information on how to launch it.

Step 2: After the Autoware Runtime manager window pops up on the screen, under the "Setup" tab, Vehicle Model will be selected. The path to the file for vehicle model can also be seen in the figure 30. Vehicle Model button can be clicked to select it.

Step 3: Next step is to setup the "Map" tab. The file paths used for Vector Map and TF can be seen on figure 31.

Step 4: Now we will setup the "computing" tab. Firstly, the *vel_pose_connect* app will be selected. Please, make sure that the simulation mode is selected as below.



vel_pose_connect

/current_pose (geometry_msgs/PoseStamped) /ndt_pose

/current_velocity (geometry_msgs/TwistStamped) /estimate_twist

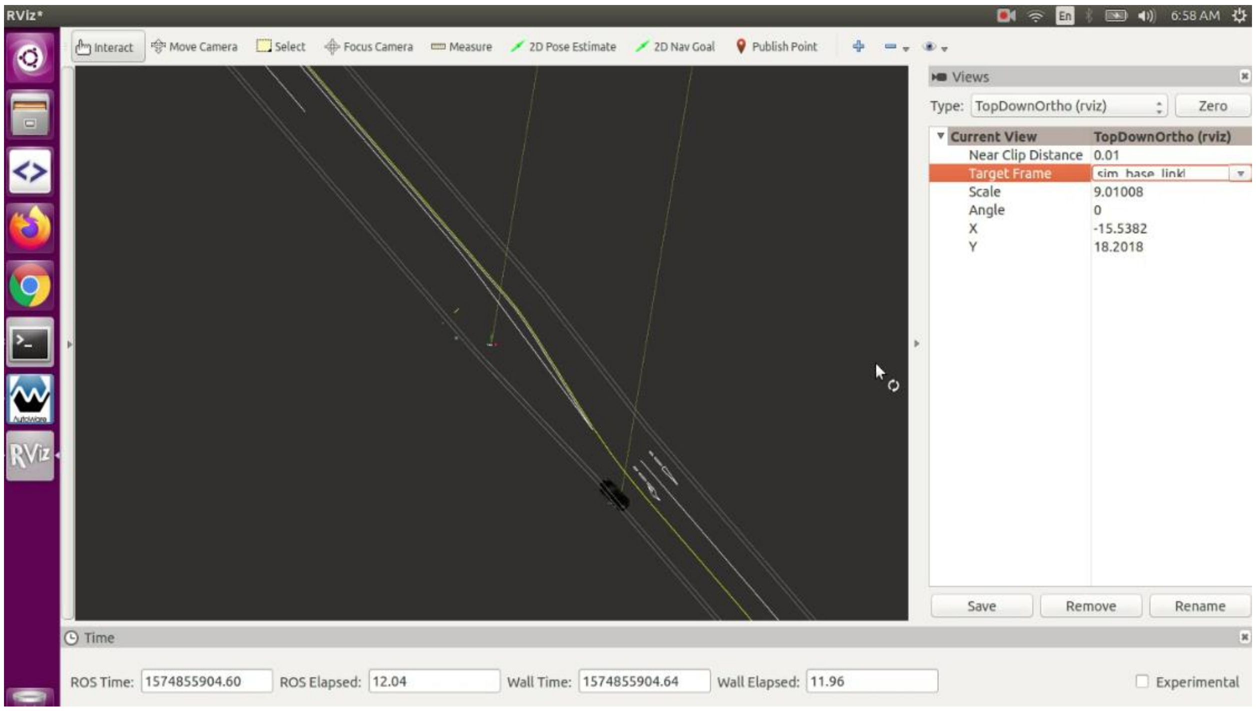
☑ Simulation Mode

OK Cancel

Figure 47: A screenshot of vel_pose_connect app

Step 5: Please turn on the wf_simulator and then click on RViz to open it.

Step 6: After launching RViz, set the 2D Pose Estimate, and the car will appear. After doing this, change the camera view to *TopDownOrtho* and target frame to *sim_base_link* located at the top right of the window.



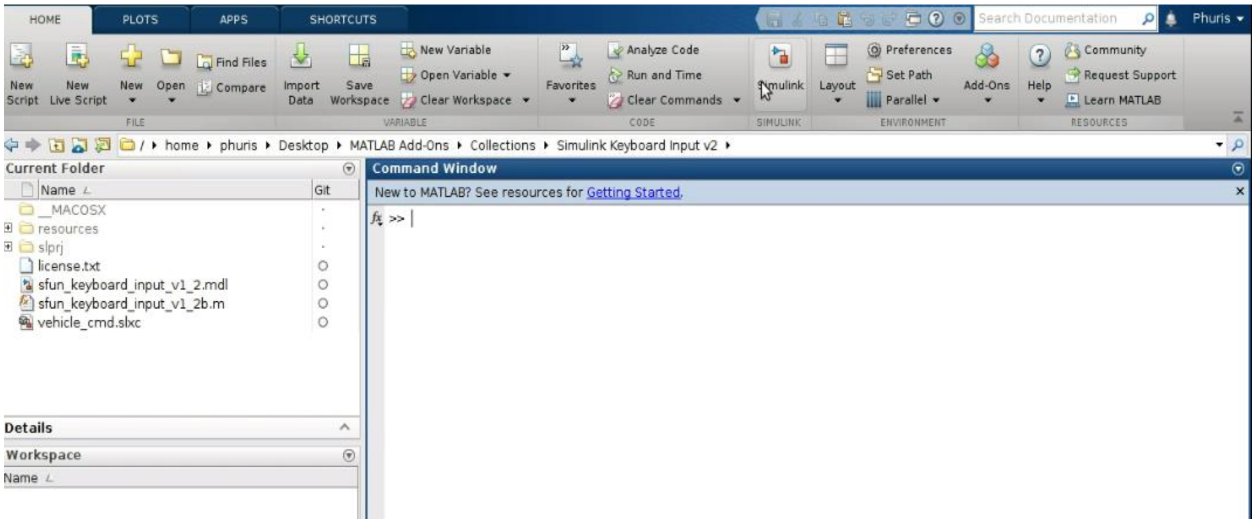
Type: TopDownOrtho (rviz) Zero

Current View	TopDownOrtho (rviz)
Near Clip Distance	0.01
Target Frame	sim_base_link
Scale	9.01008
Angle	0
X	-15.5382
Y	18.2018

ROS Time: 1574855904.60 ROS Elapsed: 12.04 Wall Time: 1574855904.64 Wall Elapsed: 11.96

Figure 48: Initial RViz window

Step 7: Run MATLAB from the terminal and change the path as shown below to locate the folder MATLAB Adds-Ons. For instance, in our case this looks like this: /home/phuris/Desktop/MATLAB Add-Ons/Collections/Simulink Keyboard Input v2



/ ▸ home ▸ phuris ▸ Desktop ▸ MATLAB Add-Ons ▸ Collections ▸ Simulink Keyboard Input v2 ▸

Name	Git
__MACOSX	
resources	·
slprj	·
license.txt	○
sfun_keyboard_input_v1_2.mdl	○
sfun_keyboard_input_v1_2b.m	○
vehicle_cmd.slxc	○

New to MATLAB? See resources for Getting Started.

```
>> |
```

Figure 49: Locating the folder on MATLAB

Step 8: Run the following command in MATLAB to initialize the ROS node.
>> rosinit
Now, launch Simulink.
After running the model, please make sure that Keyboard Window is shown up.

Step 9: Open *vehicle_cmd.slx* and run it

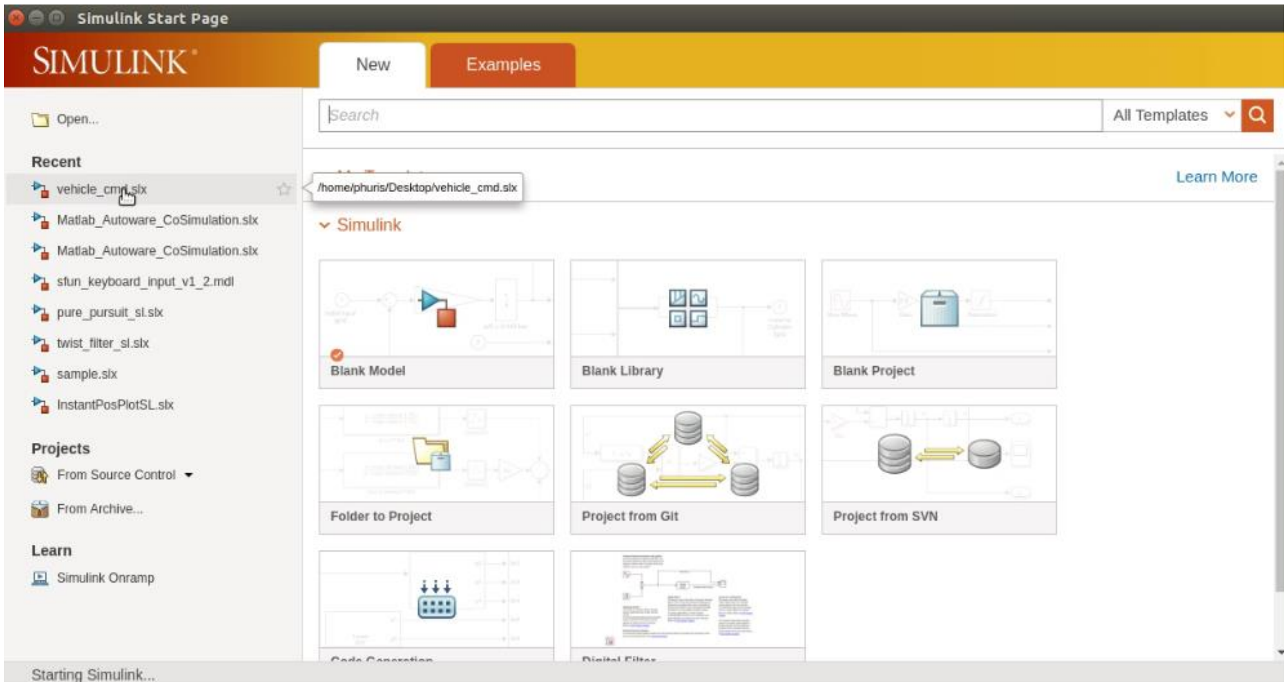


Figure 50: vehicle_cmd.slx

Step 10: Now, we are ready to control the car via keyboard.

- ➢ press 'up-arrow' to accelerate
- ➢ press 'down-arrow' to decelerate
- ➢ press 'left-arrow' to steer left
- ➢ press 'right-arrow' to steer right
- ➢ press 'spacebar' to stop turning the car

Some remarks on how to use the model:

- ➢ In the model, there exists two gain blocks for left steering angle and right steering angle which can be adjusted so that the response can be quicker.

- ➢ In addition to steering, there exists two gain blocks that determines until which velocity the vehicle should accelerate and decelerate. Those values are currently set to 60 km/h and 0 km/h respectively.

- Key control can be thought of as a switch. For instance, when up-arrow key is pressed, the vehicle will accelerate continuously even though key is not pressed anymore. Same condition applies to the left and right steering of the vehicle as well.

The video demonstration is found in our Git repository for the project and the link is provided below.

- https://gitlab.lrz.de/ge73xoh/software-lab---autoware.git

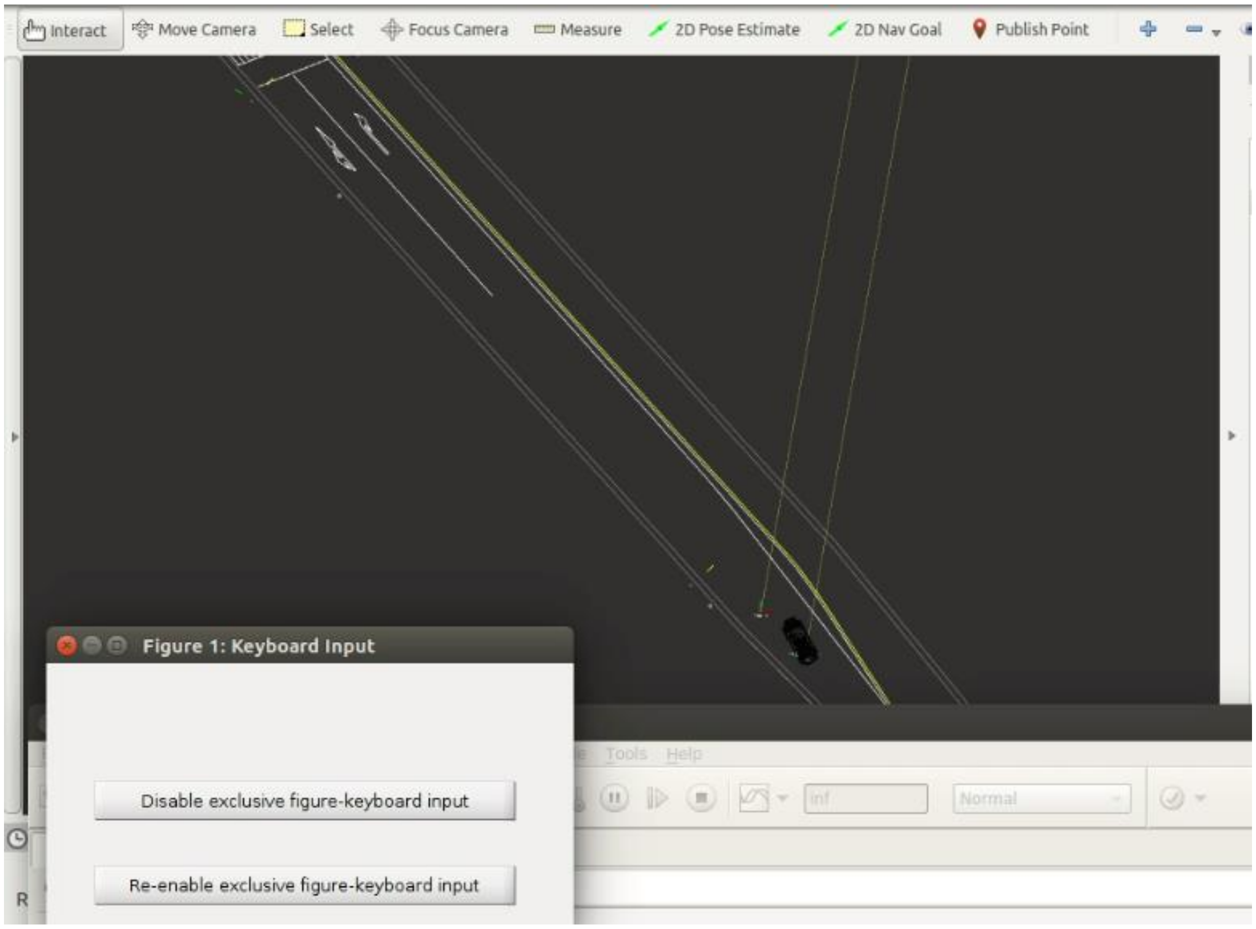


Figure 51: Keyboard Control

References

Autoware ToolBox. (n.d.). Retrieved June 06, 2019, from https://github.com/CPFL/Autoware_Toolbox

Autoware Foundation/Autoware. (n.d.). Retrieved June 06,2019, from https://github.com/autowarefoundation/autoware

Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Monrroy, A., . . . Azumi, T. (2018). Autoware on board: Enabling autonomous vehicles with embedded systems. Proc. of ICCPS 2018. 1-11. doi:10.1109/ICCPS.2018.00035

MATLAB/Simulink sample code suite for Autoware. (n.d.). Retrieved May 13, 2019, from https://github.com/CPFL/Autoware_Toolbox

Robotic System Toolbox. (n.d.). Retrieved July 15, 2019, from https://www.mathworks.com/products/robotics/code-examples.html

ROS Tutorials. (n.d.). Retrieved July 15, 2019, from http://wiki.ros.org/ROS/Tutorials

Source Build. (2019, March 18). Retrieved from https://github.com/autowarefoundation/autoware/wiki/Source-Build

Tokunaga, S., Horita, Y., Oda, Y., & Azumi, T. (2019). IDF – Autoware: Integrated development framework for ROS-Based self-driving systems using MATLAB/Simulink. ASD 2019. doi:10.4230/OASIcs.ASD.2019.3