G-Mapping with PIONEER-3DX and HOKUYO

By,
Mingyuan Shen,
Melvin P Manuel

G-Mapping with PIONEER P-3Dx and Hokuyo

1. Introduction

This document discusses about the various steps we need to follow for implementing G-Mapping on a pioneer 3Dx robot. The Robot Operating System (ROS) provides operating system-like services to operate robots. Mapping, localization, and autonomous navigation in an indoor environment are popular issues in the field of autonomous robots. Autonomous navigation in a dynamic environment is not only challenging but also uncovers many indoor environmental factors which affects the process of mapping and navigation. In this document we will be discussing how a ROS-based system is used with a Pioneer 3-DX robot for indoor mapping (G Mapping). Here we are depending mainly on LIDAR and odometer for acquiring data from environment for constructing the map.

The map of the environment is a basic need of a robot to perform indoor services like moving room to room, gripping and picking an object from one place and taking it to another place. To perform such type of services, the robot should not only know about the environment but while it is moving it should also be aware of its own location in that environment. Moreover, proper representation of the robot itself in the environment also plays a vital role to solve many issues related to automatic navigation.

The steps to be followed to implement G-mapping on a Pioneer P3dx robot with the help of data from odometer and HOKUYO UST-10LX is discussed in details below.

2. ROS setup for Pioneer 3-DX

The Pioneer 3-DX robot is one of the most popular research robots. Because of its modest and balanced size combined with reasonable hardware, it is most suitable for in-door navigation. Pioneer 3-DX robots use a differential drive for locomotion. To make the robot fully capable of mapping and localization, one laser range finder (HOKUYO UST-10LX) is also used. Linux (Ubuntu 14.4) based computer system equipped with ROS Indigo is used to control the robot. In our case, the HOKUYO UST-10LX is connected with the onboard computer through the Ethernet port “eth0”. It is a best practice to verify the connectivity between the LIDAR and computer by “ping” to the LIDAR with its assigned IP address. The various steps to configure the LIDAR (if it is not configured) are explained below.

You should always keep in mind that all the things you installed should be in the work space (ex. ~/catkin_ws/src) and remember catkin_make after you installed any new things. Go into the workspace (cd ~/catkin_ws) then sourece devel/setup.bash before you hope to run something.

3. Installed the LIDAR drive

Step 1: Clone the urg_node (which is the new hokuyo package) from github.

Go to the source (src) folder which is inside the workspace you created by executing the below command in a new terminal (In my case “catkin_ws” is the workspace name)

$ cd ~/catkin_ws/src

Execute the below command in the same terminal and wait for clone to complete

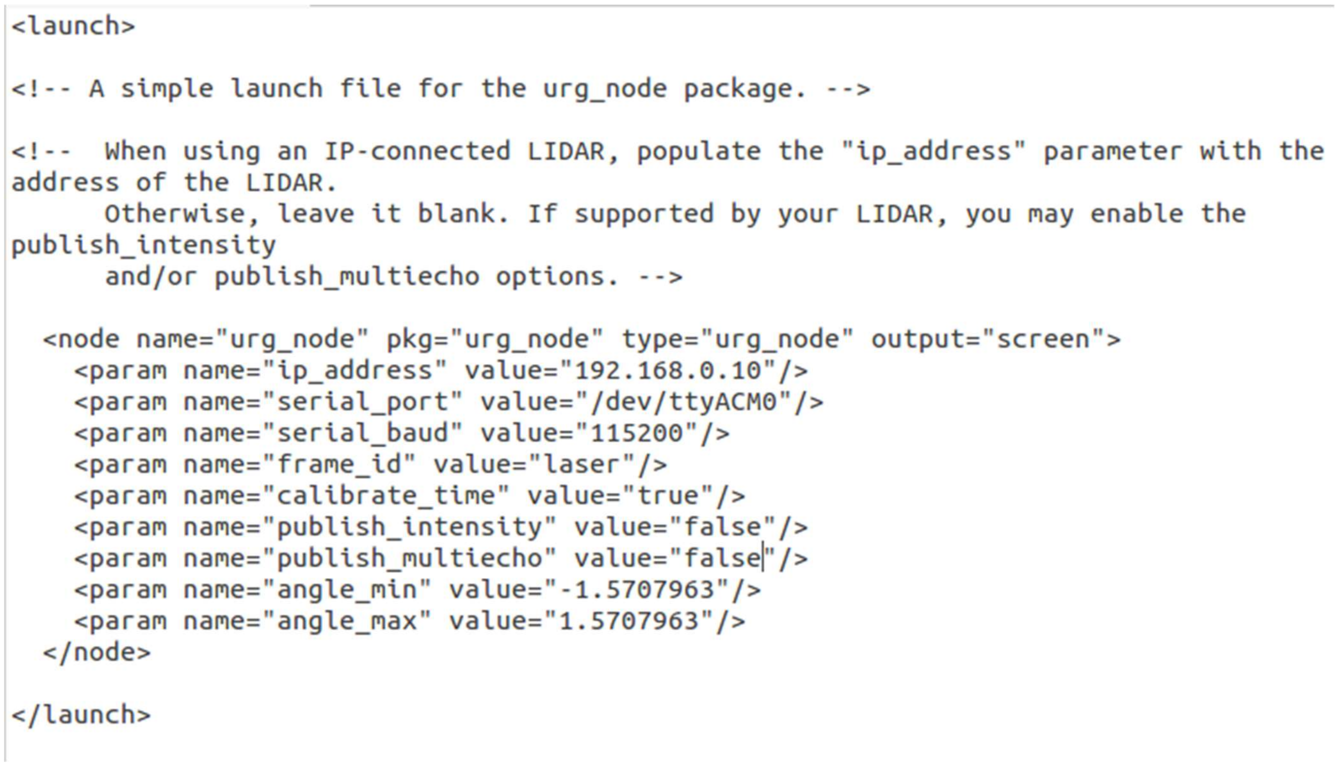
$ git clone https://github.com/ros-drivers/urg_node.git

$ git clone https://github.com/ros-perception/laser_proc.git

$ git clone https://github.com/ros-drivers/urg_c.git

Once you have done with the clone go the source (src) folder inside your workspace and ensure that three packages named “urg_node” , “laser_proc”, “urg_c” exist.

Step 2: Open the “ urg_node “ folder and open the launch folder . Inside the launch folder you can see a .launch file named “ urg_lidar.launch” file. Update the details of the file with the below details (Figure 1)

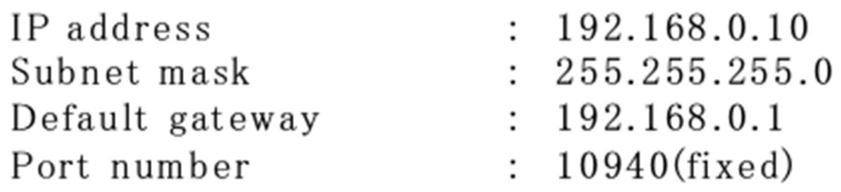


```
<launch>

<!-- A simple launch file for the urg_node package. -->

<!--  When using an IP-connected LIDAR, populate the "ip_address" parameter with the
address of the LIDAR.
      Otherwise, leave it blank. If supported by your LIDAR, you may enable the
publish_intensity
      and/or publish_multiecho options. -->

  <node name="urg_node" pkg="urg_node" type="urg_node" output="screen">
    <param name="ip_address" value="192.168.0.10"/>
    <param name="serial_port" value="/dev/ttyACM0"/>
    <param name="serial_baud" value="115200"/>
    <param name="frame_id" value="laser"/>
    <param name="calibrate_time" value="true"/>
    <param name="publish_intensity" value="false"/>
    <param name="publish_multiecho" value="false"/>
    <param name="angle_min" value="-1.5707963"/>
    <param name="angle_max" value="1.5707963"/>
  </node>

</launch>
```

Figure 1. Launch file details

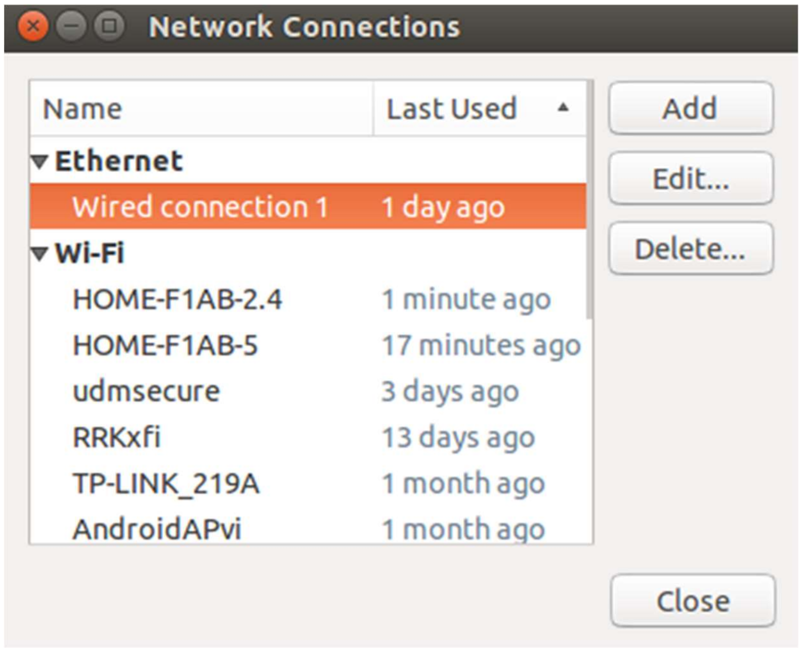
Once you update the details, especially the IP address of your lidar, save the launch file. The IP address of the LIDAR can be obtained from the operators manual of the Hokuyo Lidar. Our case it is “192.168.0.10”. If you want to change the IP address you need to download the urg network tool from link ***https://sourceforge.net/projects/urgnetwork/***

The default IP settings are listed below. We are proceeding further with the default IP address.



IP address	:	192.168.0.10
Subnet mask	:	255.255.255.0
Default gateway	:	192.168.0.1
Port number	:	10940(fixed)

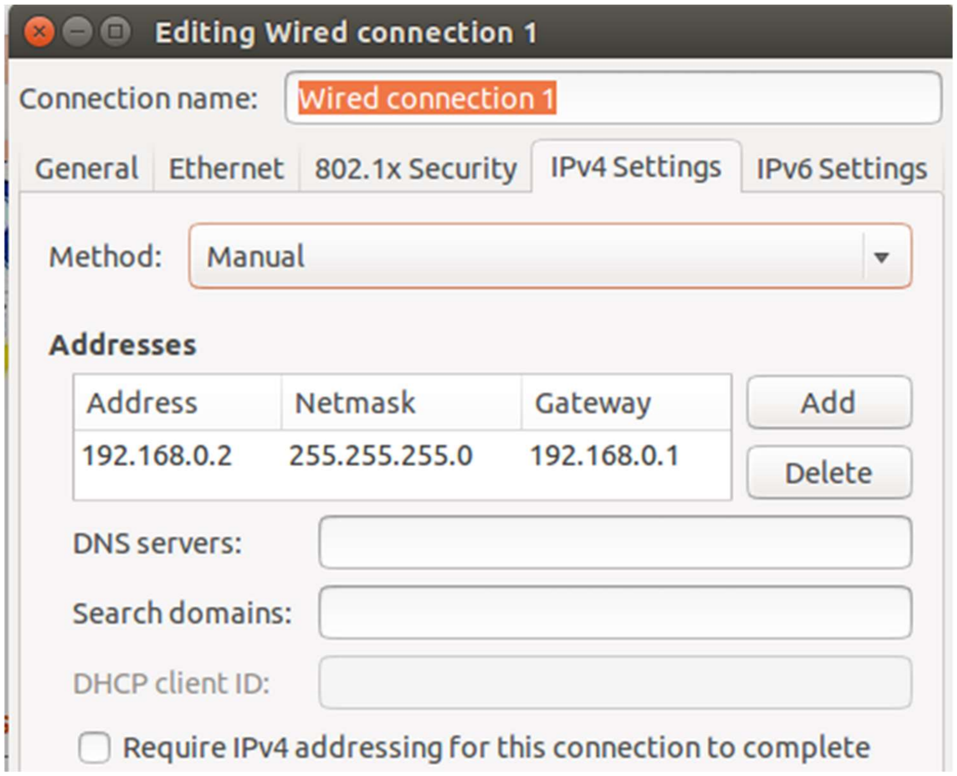
Step 3: Now we need to update the network setting of the Ethernet port of our computer. Go to the search TAB and type “Network Connections” in it. Update the network settings with the below details.



Name	Last Used
Ethernet	
Wired connection 1	1 day ago
Wi-Fi	
HOME-F1AB-2.4	1 minute ago
HOME-F1AB-5	17 minutes ago
udmsecure	3 days ago
RRKxfi	13 days ago
TP-LINK_219A	1 month ago
AndroidAPvi	1 month ago

Figure 2. Network connections tab

Select the connection under Ethernet (wired connection 1) and click the edit button. Select the IPv4 Settings and update the addresses as shown below. Save the settings.



Connection name: Wired connection 1

Method: Manual

Addresses

Address	Netmask	Gateway
192.168.0.2	255.255.255.0	192.168.0.1

DNS servers:

Search domains:

DHCP client ID:

Require IPv4 addressing for this connection to complete

Figure 3. Editing window inside network settings

And I really advise you click this link and set as he taught so that you can use the ethernet and WIFI at the same time. By this way, you don't need to pluck the HOKUYO every time when you hope to use the Internet.

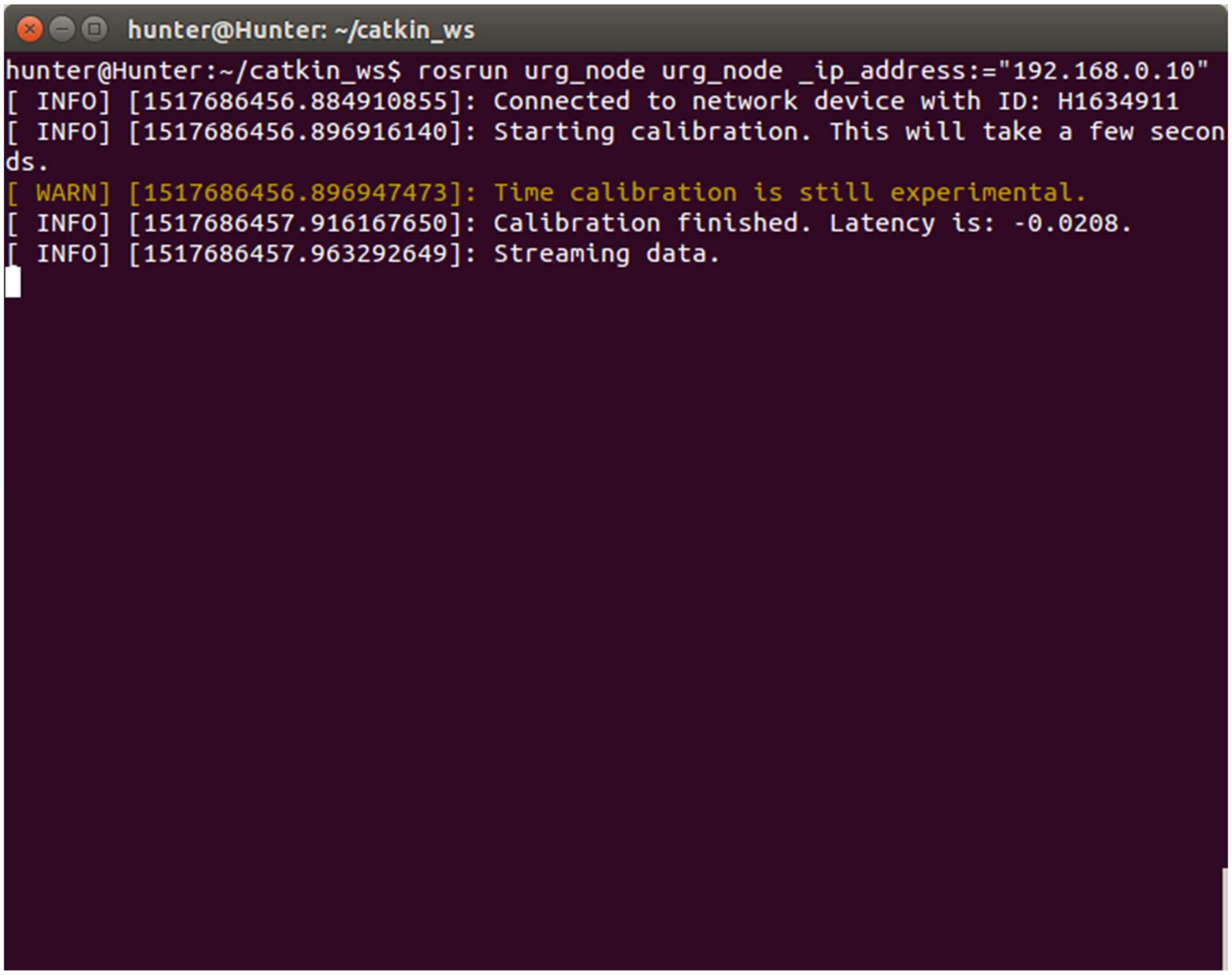
https://askubuntu.com/questions/639100/how-to-get-connection-to-both-wifi-as-well-as-lan-in-ubuntu-14-04-lts

Step 4: Connect the Hokuyo to the Ethernet port of your computer. Then, switch on the power supply. Open a new terminal and type the below command.

$ roscore

Open another terminal and execute the launch file. Please don't forget to update the command with the IP address you have configured for your LIDAR. I have used the default IP address which is 192.168.0.10

$ rosrun urg_node urg_node _ip_address:="192.168.0.10"

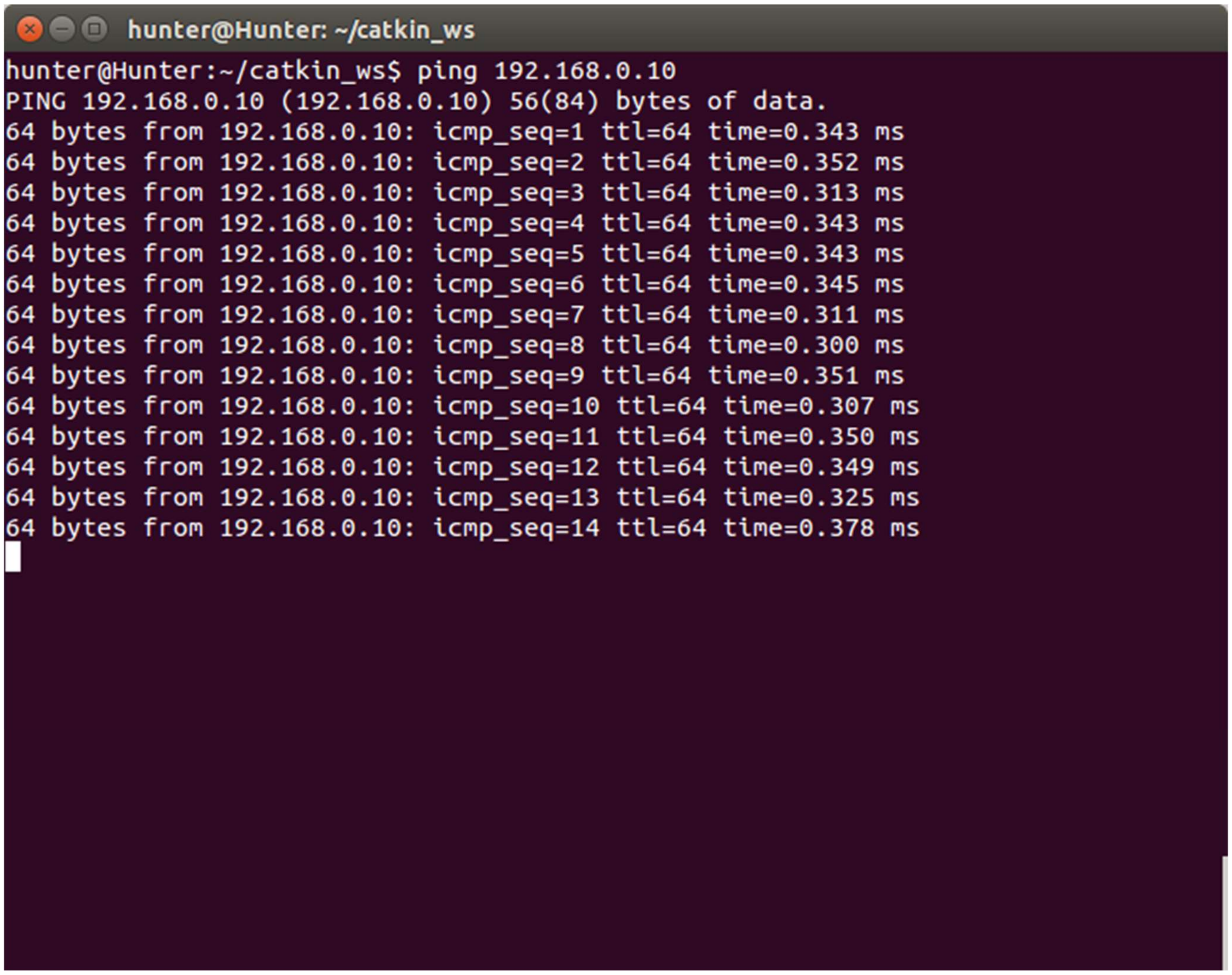


```
hunter@Hunter:~/catkin_ws$ rosrun urg_node urg_node _ip_address:="192.168.0.10"
[ INFO] [1517686456.884910855]: Connected to network device with ID: H1634911
[ INFO] [1517686456.896916140]: Starting calibration. This will take a few secon
ds.
[ WARN] [1517686456.896947473]: Time calibration is still experimental.
[ INFO] [1517686457.916167650]: Calibration finished. Latency is: -0.0208.
[ INFO] [1517686457.963292649]: Streaming data.
```

Figure 4. Streaming data

By the way, you can also use the following command to check the connection of the HOKUYO is good or not.

$ ping 192.168.0.10

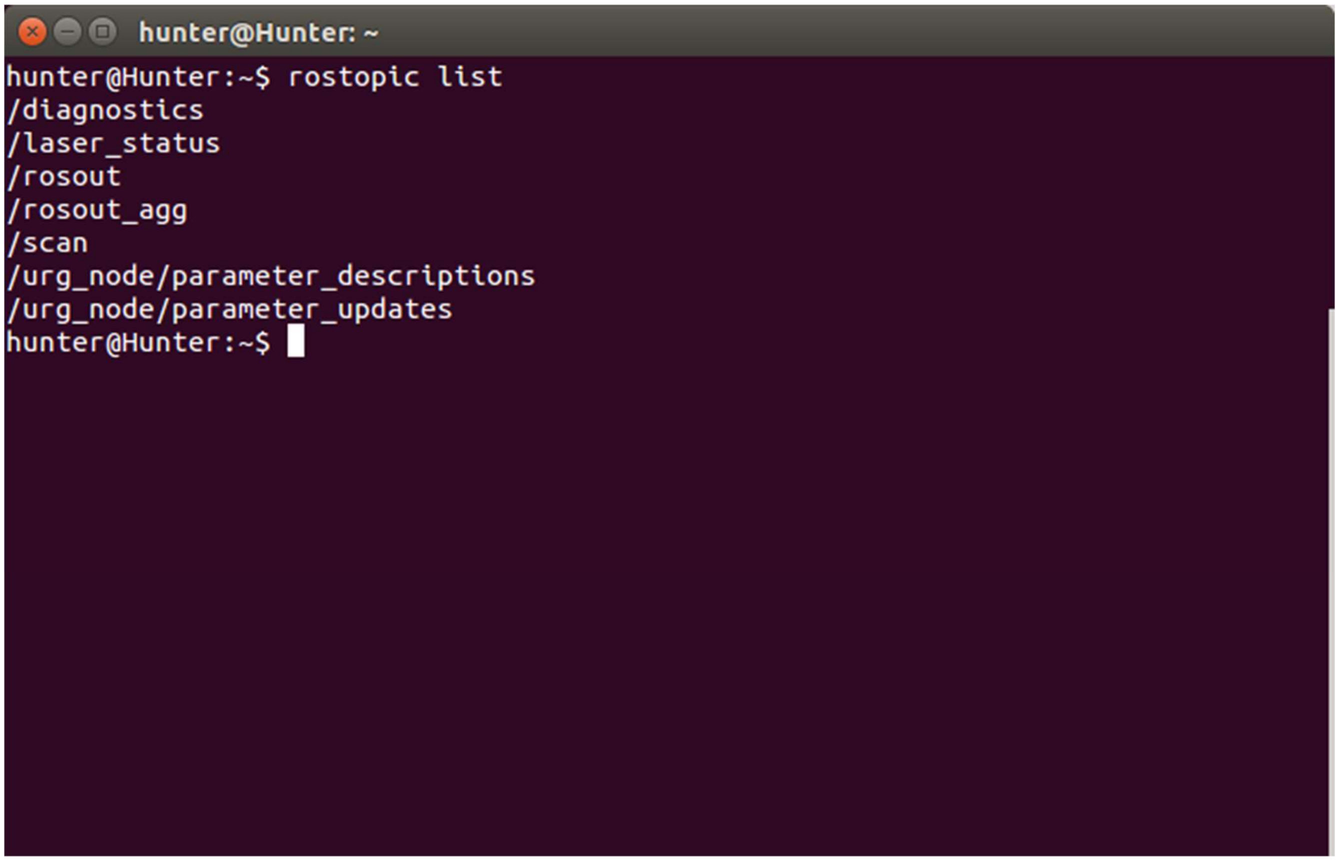


```
hunter@Hunter: ~/catkin_ws
hunter@Hunter:~/catkin_ws$ ping 192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
64 bytes from 192.168.0.10: icmp_seq=1 ttl=64 time=0.343 ms
64 bytes from 192.168.0.10: icmp_seq=2 ttl=64 time=0.352 ms
64 bytes from 192.168.0.10: icmp_seq=3 ttl=64 time=0.313 ms
64 bytes from 192.168.0.10: icmp_seq=4 ttl=64 time=0.343 ms
64 bytes from 192.168.0.10: icmp_seq=5 ttl=64 time=0.343 ms
64 bytes from 192.168.0.10: icmp_seq=6 ttl=64 time=0.345 ms
64 bytes from 192.168.0.10: icmp_seq=7 ttl=64 time=0.311 ms
64 bytes from 192.168.0.10: icmp_seq=8 ttl=64 time=0.300 ms
64 bytes from 192.168.0.10: icmp_seq=9 ttl=64 time=0.351 ms
64 bytes from 192.168.0.10: icmp_seq=10 ttl=64 time=0.307 ms
64 bytes from 192.168.0.10: icmp_seq=11 ttl=64 time=0.350 ms
64 bytes from 192.168.0.10: icmp_seq=12 ttl=64 time=0.349 ms
64 bytes from 192.168.0.10: icmp_seq=13 ttl=64 time=0.325 ms
64 bytes from 192.168.0.10: icmp_seq=14 ttl=64 time=0.378 ms
```

Figure 5. Check the connection

Execute rostopic list to ensure that you can see the topic from lidar. The topic name is scan in this case. See the response after executing rostopic list in figure 6.

$ rostopic list



```
hunter@Hunter:~$ rostopic list
/diagnostics
/laser_status
/rosout
/rosout_agg
/scan
/urg_node/parameter_descriptions
/urg_node/parameter_updates
hunter@Hunter:~$ 
```

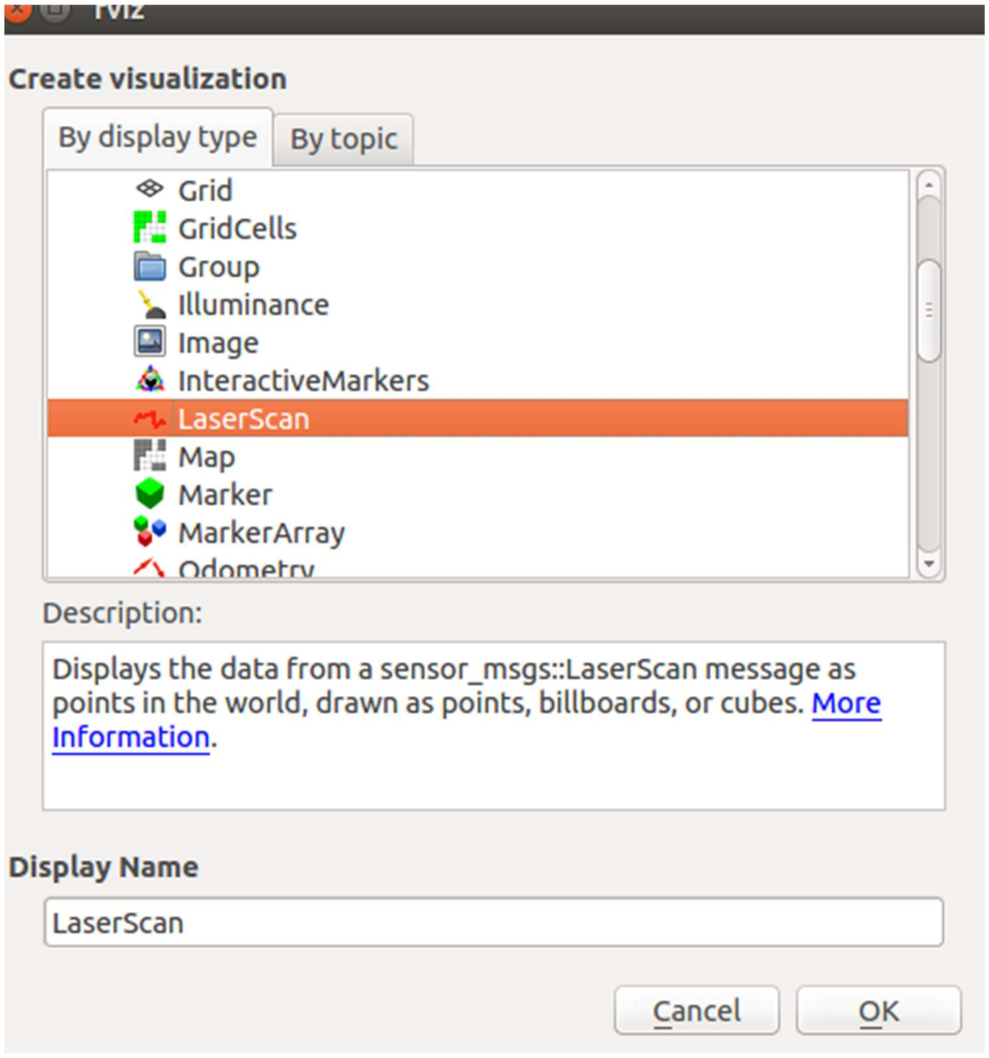
Figure 6. Response after executing rostopic list

4.How to see the LIDAR scan in RVIZ

Open RVIZ by executing the below command in a new terminal.

$ rviz

Once the RVIZ opened, click on the add button, which is available at the bottom left side and from the create visualization window (as shown in figure), select a laserScan display type from it and click on OK button.

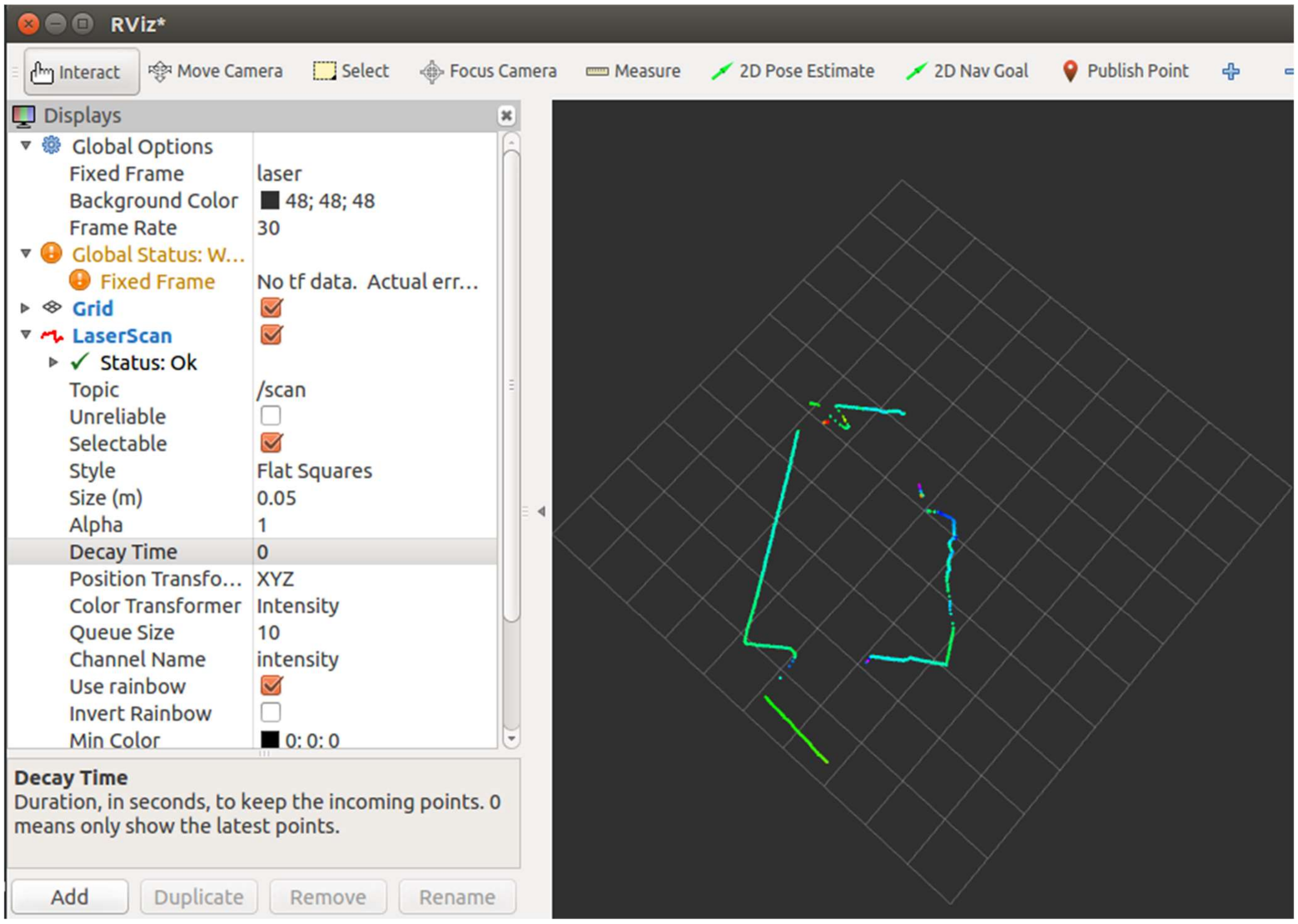


Displays the data from a sensor_msgs::LaserScan message as points in the world, drawn as points, billboards, or cubes. More Information.

Display Name: LaserScan

Figure 7: Rviz create visualization window

Under Displays tab update the fixed frame to laser. Then update the topic under LaserScan to /scan. You can see the LIDAR data now in the RVIZ window as shown in figure 8.



Global Options	
Fixed Frame	laser
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: W...	
Fixed Frame	No tf data. Actual err...
Grid	✓
LaserScan	✓
Status: Ok	
Topic	/scan
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.05
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0

Decay Time
Duration, in seconds, to keep the incoming points. 0 means only show the latest points.

Figure 8. Rviz window with LIDAR scan data

5. G-mapping with Pioneer P3Dx

Since we are ready with the LIDAR data and assume that we are confident to work with ROSARIA and ARIA package, we can now look in to the G-Mapping package for mapping an unknown environment.

Before proceeding further, we need to ensure that, your pioneer robot is able to move by using the basic ARIA package. That means, you should be able to run the Pioneer robot in different modes, like wandering mode, teleop mode etc. This step ensures that your pioneer is ready to go with its basic properties like publishing and subscribing the basic data to and from the pioneer board with the NUC computer.

We are using the g-mapping package available at the link below.

https://github.com/ros-perception/slam_gmapping

For more details about the g-mapping package you can browse through the below link http://wiki.ros.org/gmapping .

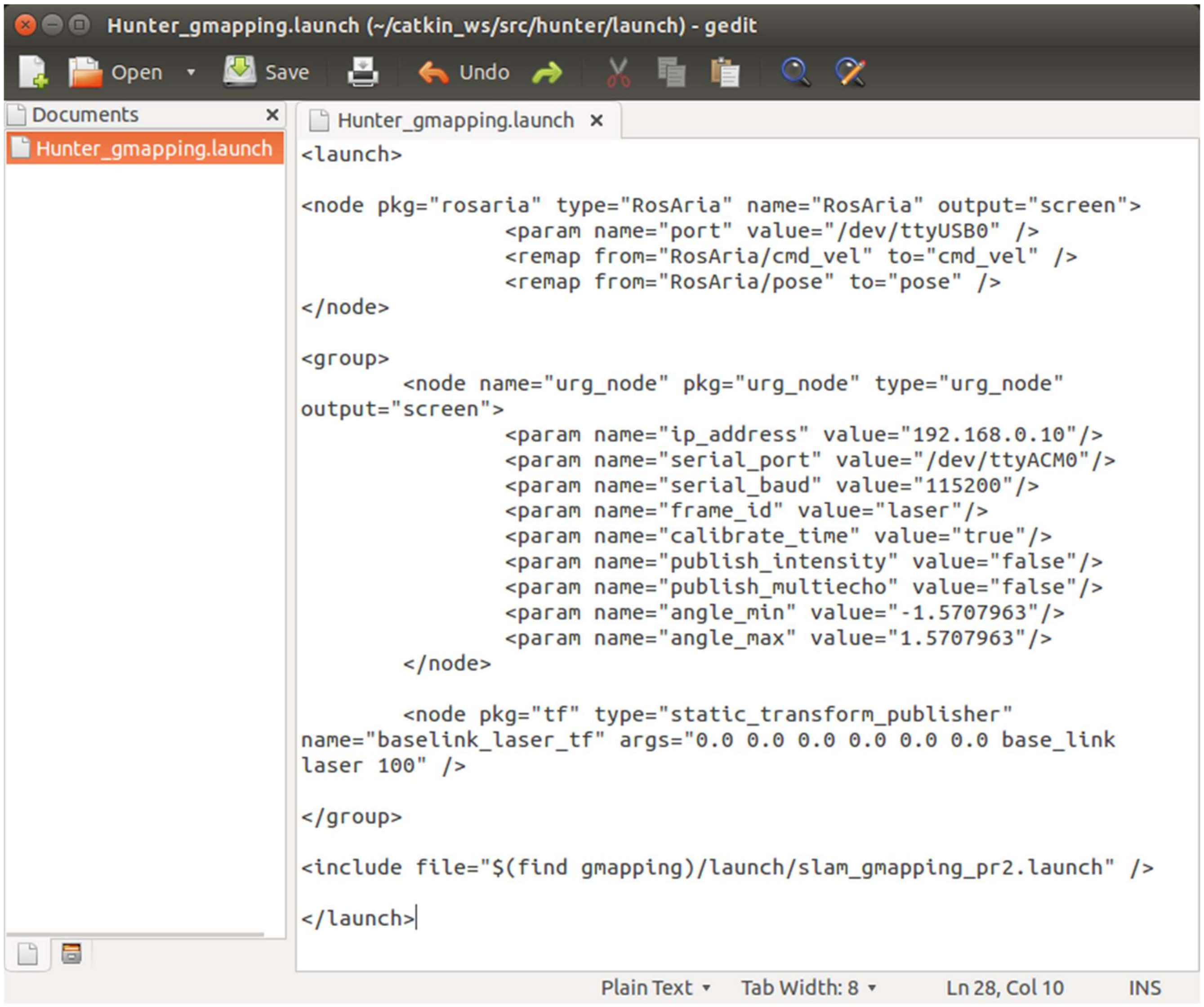
As we are doing with any other packages, clone the slam_gmapping package inside our workspace. Here in our case the workspace used is “catkin_ws”. Go to catkin_ws and download the github code using the following command:

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ros-perception/slam_gmapping.git
```

Then I advise you build your own package and launch file. You can do it as following steps.

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg “your_package_name” std_msgs rospy roscpp
$ cd “your_package_name”
$ mkdir “launch”
$ gedit “your_launchfile_name.launch”
```

And edit the launch file like the following figure.



Hunter_gmapping.launch (~/catkin_ws/src/hunter/launch) - gedit

```
<launch>

<node pkg="rosaria" type="RosAria" name="RosAria" output="screen">
                <param name="port" value="/dev/ttyUSB0" />
                <remap from="RosAria/cmd_vel" to="cmd_vel" />
                <remap from="RosAria/pose" to="pose" />
</node>

<group>
        <node name="urg_node" pkg="urg_node" type="urg_node"
output="screen">
                <param name="ip_address" value="192.168.0.10"/>
                <param name="serial_port" value="/dev/ttyACM0"/>
                <param name="serial_baud" value="115200"/>
                <param name="frame_id" value="laser"/>
                <param name="calibrate_time" value="true"/>
                <param name="publish_intensity" value="false"/>
                <param name="publish_multiecho" value="false"/>
                <param name="angle_min" value="-1.5707963"/>
                <param name="angle_max" value="1.5707963"/>
        </node>

        <node pkg="tf" type="static_transform_publisher"
name="baselink_laser_tf" args="0.0 0.0 0.0 0.0 0.0 0.0 base_link
laser 100" />

</group>

<include file="$(find gmapping)/launch/slam_gmapping_pr2.launch" />

</launch>
```

Figure 10. launch file of gmapping

I really hope you that you understand every lines in the above launch file because it will help a lot in your future work. You may find a lot of things you are familiar in this launch file because the main aim of this launch file is open other launch files or notes. And by this way you can just use one command instead of many in different terminals. So, you can also copy from the urg_note's launch file and gmapping's launch file. Don't need to type all of things.

Now, after catkin_make, you can use this launch file to build your own map.

$ cd catkin_ws

$ catkin_make

Once you are done with the catkin_make you can see a new folder created under the src (Source) folder with name slam_gmapping. For starting with g-mapping, Open a new terminal and type:

$ roscore

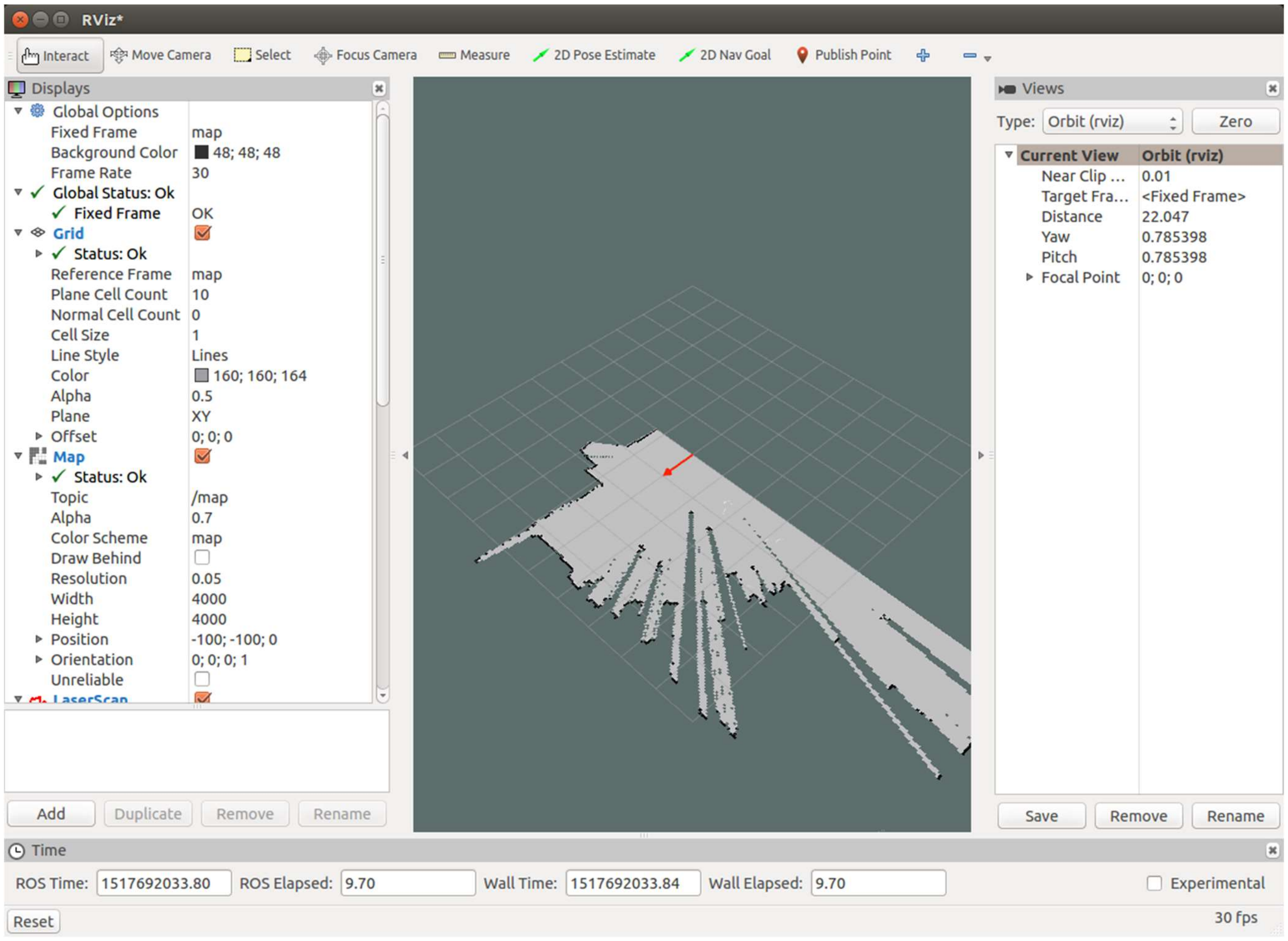
Then in another terminal:

$ cd catkin_ws

$ source /devel/setup.bash

$ roslaunch your_package_name your_launchfile_name.launch

Then you can open the RVIZ to see the simulation like figure 11.



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
✓ Status: Ok	
Reference Frame	map
Plane Cell Count	10
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	▨ 160; 160; 164
Alpha	0.5
Plane	XY
Offset	0; 0; 0
Map	✓
✓ Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.05
Width	4000
Height	4000
Position	-100; -100; 0
Orientation	0; 0; 0; 1
Unreliable	☐
LaserScan	✓

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	22.047
Yaw	0.785398
Pitch	0.785398
Focal Point	0; 0; 0

ROS Time: 1517692033.80 ROS Elapsed: 9.70 Wall Time: 1517692033.84 Wall Elapsed: 9.70 ☐ Experimental

30 fps

Figure 11. Simulation in RVIZ

It's a good idea to add pioneer model, tf and odom display type to check the status of the robot and it will help a lot when something wrong.

You may use the following command to check your data transformer:

$ rosrun tf view_frames

And the correct one should be similar like following figure:

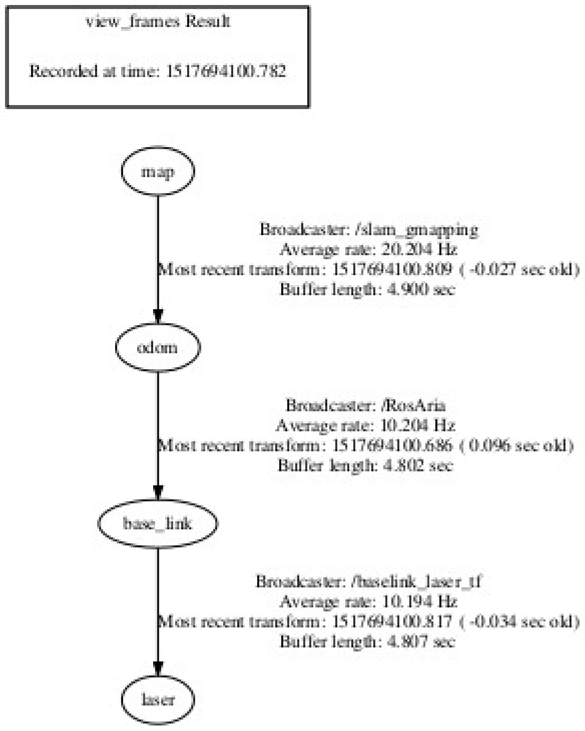
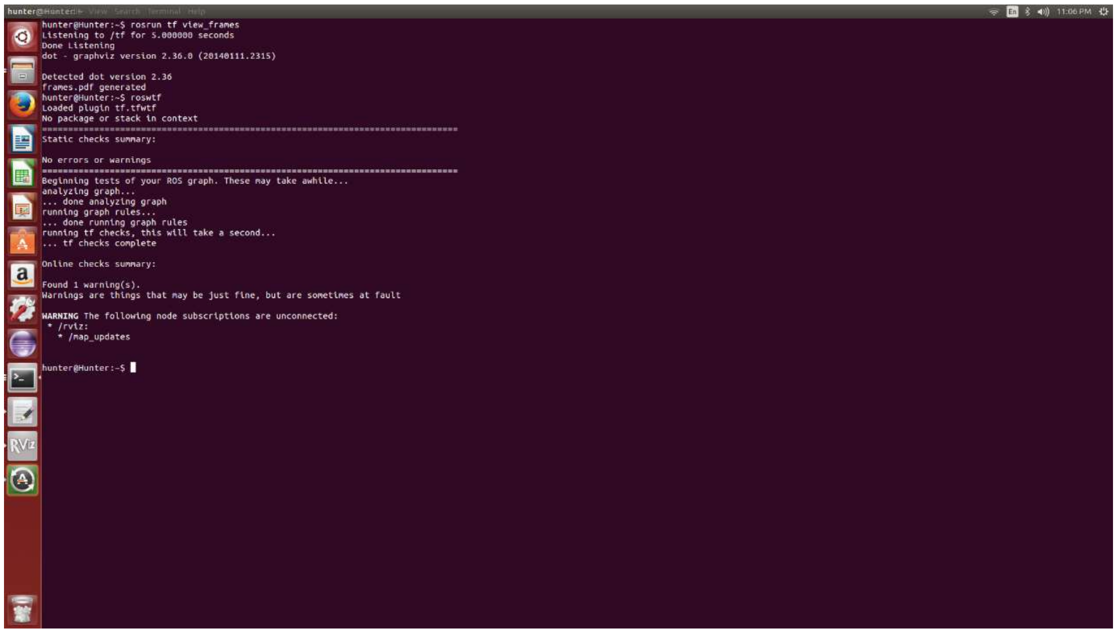

view_frames Result
Recorded at time: 1517694100.782
map
Broadcaster: /slam_gmapping
Average rate: 20.204 Hz
Most recent transform: 1517694100.809 (-0.027 sec old)
Buffer length: 4.900 sec
odom
Broadcaster: /RosAria
Average rate: 10.204 Hz
Most recent transform: 1517694100.686 (0.096 sec old)
Buffer length: 4.802 sec
base_link
Broadcaster: /baselink_laser_tf
Average rate: 10.194 Hz
Most recent transform: 1517694100.817 (-0.034 sec old)
Buffer length: 4.807 sec
laser

Figure 12. tf frame trees

And you may also use this command to let ROS check itself

$ roswtf



```
hunter@Hunter:~$ rosrun tf view_frames
Listening to /tf for 5.000000 seconds
Done Listening
dot - graphviz version 2.36.0 (20140111.2315)

Detected dot version 2.36
frames.pdf generated
hunter@Hunter:~$ roswtf
Loaded plugin tf.tfwtf
No package or stack in context
================================================================================
Static checks summary:

No errors or warnings
================================================================================
Beginning tests of your ROS graph. These may take awhile...
analyzing graph...
... done analyzing graph
running graph rules...
... done running graph rules
running tf checks, this will take a second...
... tf checks complete

Online checks summary:

Found 1 warning(s).
Warnings are things that may be just fine, but are sometimes at fault

WARNING The following node subscriptions are unconnected:
 * /rviz:
   * /map_updates


hunter@Hunter:~$ 
```

Figure 13. Check the ros graph

Additional tools:

1.Pioneer Model

For obtaining the Pioneer_Model(which is required for G-mapping and Navigation) you can get it from the below link:

http://wiki.lofarolabs.com/index.php/Moving_The_Pioneer_3-DX_In_Gazebo

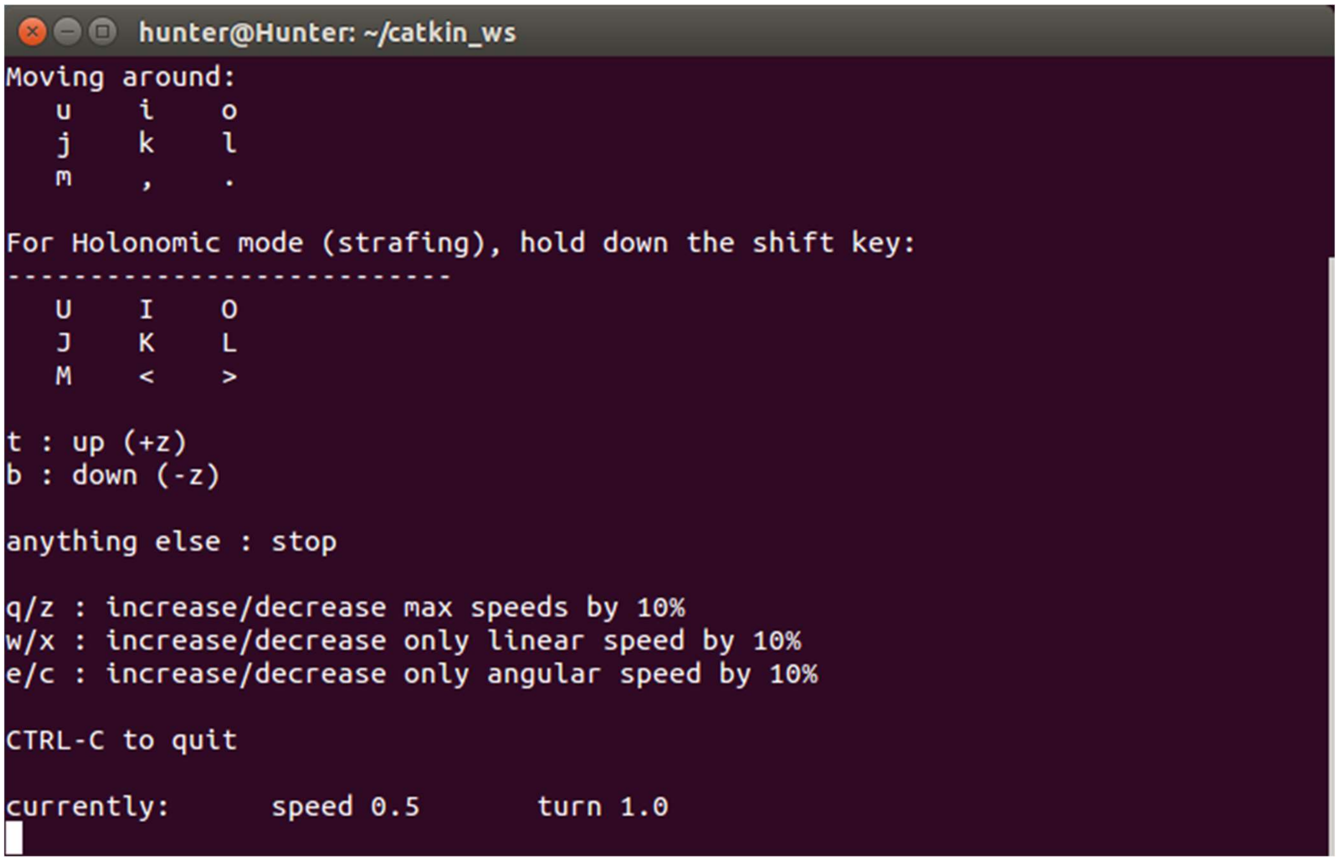
2.Keyboard Control

Once the rviz is configured you can drive the robot to the desired location where you need to map by using the teleop command. If you have not installed teleop_twist_keyboard package, you can install it by following the below steps. If you already installed teleop twist keyboard, you can jump to the next command for installing teleop package. Open a new terminal and copy the below command in it.

$ sudo apt-get install ros-indigo-teleop-twist-keyboard.

Once you are ready with teleop, open a new terminal and execute the below command. Keep in mind that you need to have this terminal active to drive the robot.

$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py



```
hunter@Hunter: ~/catkin_ws
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Figure 12. Keyboard Control tool

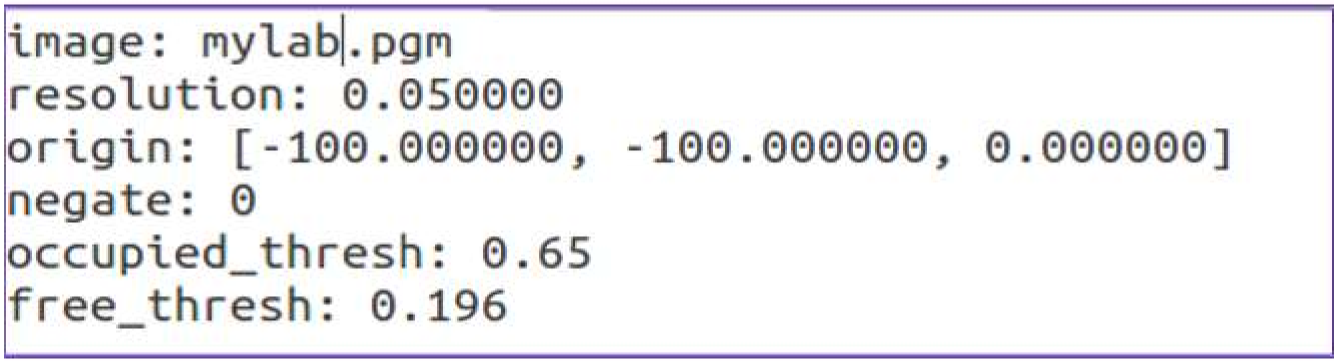
Below figure shows the teleop twist keyboard terminal. You can follow the keys on the below terminal to drive the robot. Reduce the speed of the robot by using the q/z keys which will help the algorithm to update the map without any errors.

3.Map saving

Teleop control keys and its details On the RVIZ screen you can see the updating map and once you are done with mapping open another terminal and save the map by executing the below command

$ rosrun map_server map_saver -f "map_name"

Give some name for your map in the filed "map_name". Now you can close the above terminal and also the terminals you launched for g mapping. Go to the workspace you created ie /home/amrl/catkin_ws directory and you can see two files created there with the name you specified for your map. If you gave the map_name as mylab, you will get two files named "mylab.pgm " and "mylab.yaml". The [map_name].yaml file only contains metadata about the map, which is given as a separate [map_name].pgm image file. You can create maps by using a SLAM algorithm and saving it using the map_saver tool. In principle, you can also edit maps by editing .pgm files with a graphics editing program of your choice. Fig 12 shows the contents inside the yaml file.



```
image: mylab.pgm
resolution: 0.050000
origin: [-100.000000, -100.000000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

Fig 12. Contents of a sample yaml file

Details of each parameter is discussed below:

image: Path to the image file containing the occupancy data; can be absolute, or relative to the location of the YAML file

Resolution: Resolution of the map, meters / pixel

Origin: The 2-D pose of the lower-left pixel in the map, as (x, y, yaw), with yaw as counterclockwise rotation (yaw=0 means no rotation). Many parts of the system currently ignore yaw.

Occupied_thresh: Pixels with occupancy probability greater than this threshold are considered completely occupied.

Free_thresh: Pixels with occupancy probability less than this threshold are considered completely free.

Negate: Whether the white/black free/occupied semantics should be reversed (interpretation of thresholds is unaffected)

4. Wireless Control tool

Teamviewer or Romanian or others. Either of them has its limit, such as Romanian can't use on a windows system computer ad teamviewer is conflict with rviz. So, choose one by your own concerning and maybe you can find some new tools.

References

http://ros.org/wiki/urg_node

http://wiki.ros.org/laser_proc

http://wiki.ros.org/urg_c

https://askubuntu.com/questions/639100/how-to-get-connection-to-both-wifi-as-well-as-lan-in-ubuntu-14-04-lts

http://robots.mobilerobots.com/wiki/ARIA

https://github.com/ros-perception/slam_gmapping

http://wiki.ros.org/slam_gmapping/Tutorials/MappingFromLoggedData#record

http://wiki.ros.org/map_server

http://wiki.ros.org/teleop_twist_keyboard

http://wiki.ros.org/tf/

http://wiki.ros.org/roswtf