Autonomous Robotics with ROS for mmWave Setup Guide

Version 1.2

Table of Contents

Purpose

This lab allows for the TI mmWave sensor to be used with popular mapping and navigation libraries in the Robot Operating System (ROS) environment, familiar to many robotics engineers. The lab uses the Octomap server and move_base libraries with TI's mmWave mmWave ROS Driver Package software interface to the TI mmWave sensor IWR1443BOOST EVM or the TI mmWave sensor IWR6843ISK EVM. With this TI driver and software from the ROS community (ros.org) engineers may evaluate robot navigation and object avoidance quickly and easily.

Requirements

Hardware

- TurtleBot2 with plate and standoff kit
- Two computers (preferably laptops) – one for Turtlebot2, one for remote operation and visualization
- USB 2.0 printer-style cable (A-Male to B-Male) to connect laptop to Turtlebot2
- TI mmWave sensor IWR1443BOOST EVM or TI mmWave sensor IWR6843ISK EVM (with MMWAVEICBOOST carrier board)
- USB to microUSB cable to connect laptop to the mmWave EVM (cable comes with the EVM and should be connected to 'XDS110 USB' port on EVM)
- 12V to 5V DC to DC converter (must be able to output at least 2.5Amps at 5V) to allow the EVM to be powered from the Turtlebot2 (this converter is required since the normal 5V output port on the TurtleBot2 cannot supply 2.5Amps)
- 2-pin miniFit JR connector/cable to go from Turtlebot2 12V output port to the 12V input on the converter (for example, Molex cable part number 245135-0210 or 245135-0220 can be used by cutting it in half so the connector end goes to the TurtleBot2 12V output port and the cut wire end goes to the 12V input of the converter)
- 2.1mm barrel jack connector (center positive) with cable/wire to go from the 5V output on the converter to the EVM
- Misc. small bolts and nuts and brackets for mounting mmWave sensor and DC converter to TurtleBot platform (not included with EVM or TurtleBot)

Software

TI mmWave EVM

- Must be running TI mmWave SDK Out-of-Box Demo firmware from mmWave SDK version 2.1 for IWR1443 ES3.0 or mmWave SDK version 3.0 for IWR6843 ES1.0

Laptops

- Linux Ubuntu 16.04 natively installed (Ubuntu 16.04 Virtual Machine running on Windows can be used for remote control laptop if desired)
- ROS Kinetic Kame LTS with specified ROS packages

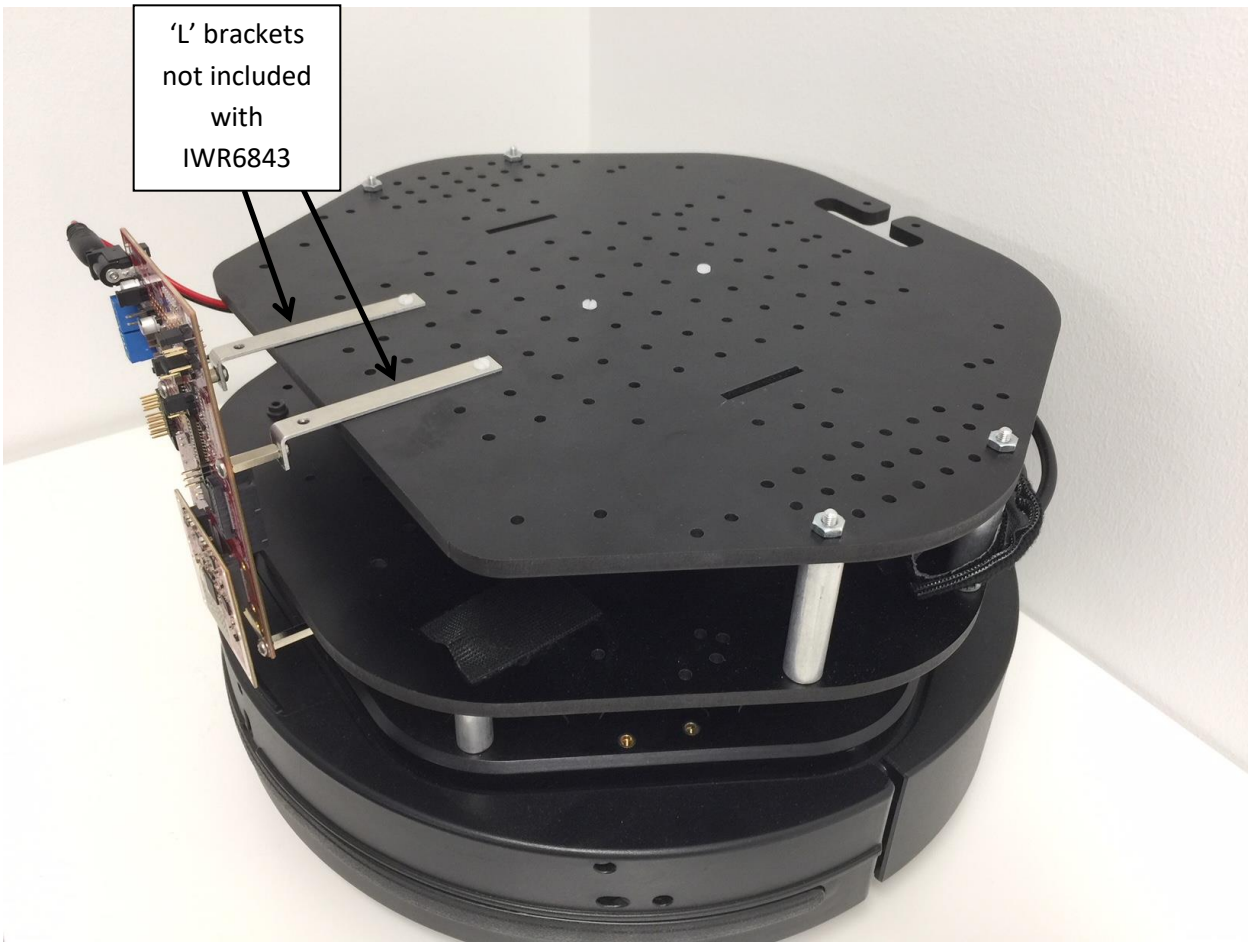
- TI mmWave ROS Driver
- Additional ROS packages supplied with this lab

TurtleBot2

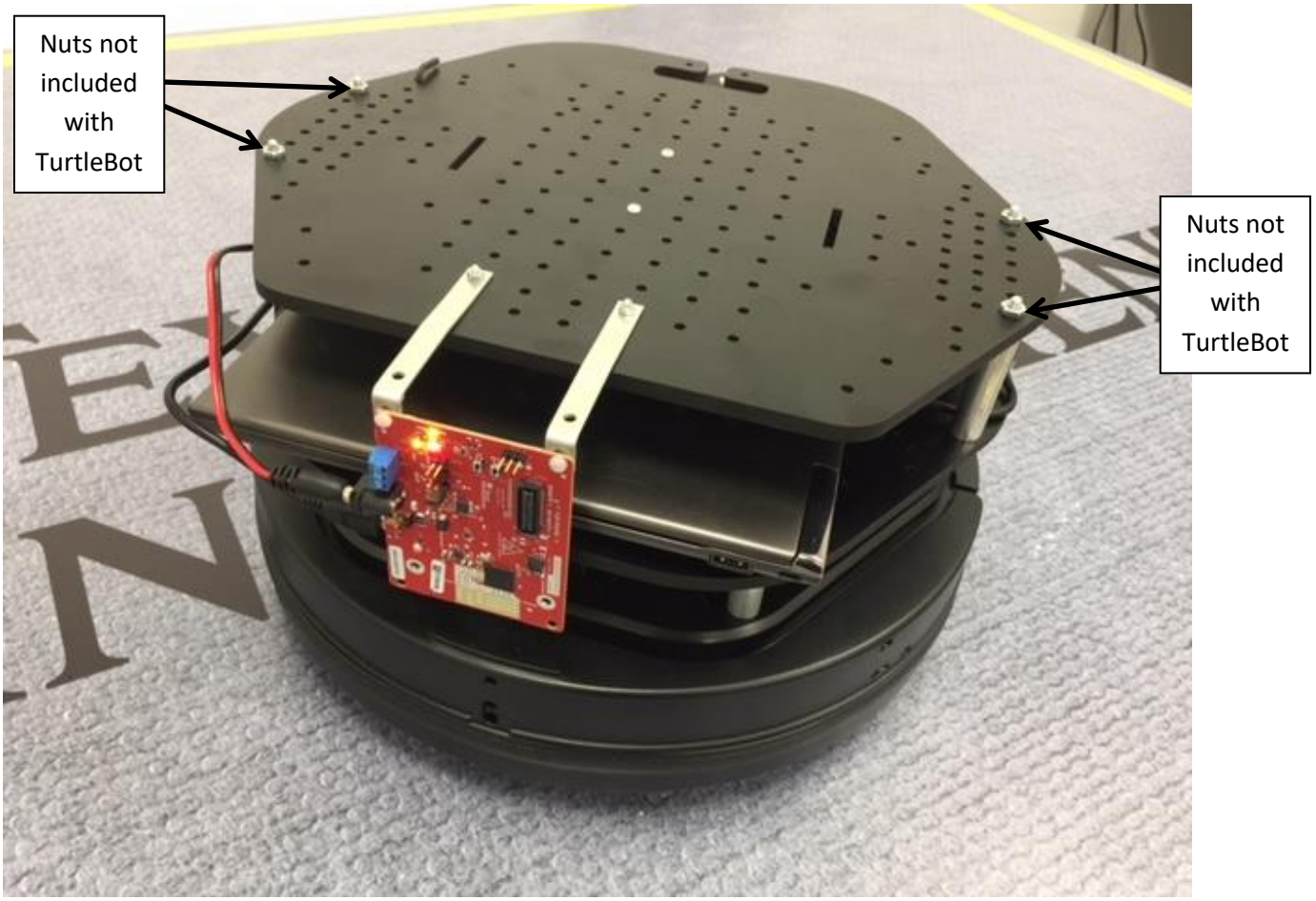
The TurtleBot2 is a low-cost, personal robotics platform that is well supported within the ROS community. There are many existing demos that work out-of-the-box with the TurtleBot2 including teleoperation (remote control), mapping, and navigation. In this guide we will have a look at how to modify these demos to integrate the TI mmWave sensor as the 3-D sensor. The TurtleBot2 should be assembled and mmWave EVM mounted as shown in the following picture to work with this lab. In the example shown, the 12V to 5V converter is mounted underneath the center of the top plate.

IWR6843ISK with MMWAVEICBOOST carrier board – front view (shown without required laptop):

IWR6843ISK with MMWAVEICBOOST carrier board – side view (shown without required laptop):



IWR1443BOOST (shown with laptop):



Installing ROS and the TI mmWave ROS Driver

Please follow the instructions in the TI mmWave ROS Driver Setup Guide (available on the TI Resource Explorer) to install ROS and the TI mmWave ROS Driver on each laptop before continuing. ROS must be installed on both the TurtleBot laptop and the Remote Control laptop. It is a good idea to test out the installation on each laptop by connecting the TI mmWave EVM and trying out the point cloud visualization.

Installing the mmWave Mapping and Navigation Demo Packages

After installing ROS and the TI mmWave ROS Driver, follow the steps below on both laptops for interchangeability.

Install the Required Dependent ROS Packages

1. Install the following ROS packages which are required dependencies to run the demos:

```
$ sudo apt-get install ros-kinetic-navigation \
  ros-kinetic-hector-slam \
  ros-kinetic-octomap-server \
  ros-kinetic-kobuki \
  ros-kinetic-octomap-rviz-plugins \
  ros-kinetic-vision-opencv \
  ros-kinetic-depth-image-proc \
  ros-kinetic-joy
```

2. Download the source code for the specific version of the ROS fake_localization package required to run the demos:

```
$ mkdir -p ~/catkin_ws/src/navigation/fake_localization
$ cd ~/catkin_ws/src/navigation/fake_localization
$ wget https://raw.githubusercontent.com/ros-planning/navigation/1.14.2/fake_localization/fake_localization.cpp
$ wget https://raw.githubusercontent.com/ros-planning/navigation/1.14.2/fake_localization/CMakeLists.txt
$ wget https://raw.githubusercontent.com/ros-planning/navigation/1.14.2/fake_localization/package.xml
```

Download the TurtleBot mmWave Mapping and Navigation Packages

1. Download the ti_mmwave_ros_map_nav_<ver>.tar.gz Linux archive file from the TI Resource Explorer and copy it to your ${WORKSPACE}/src directory.

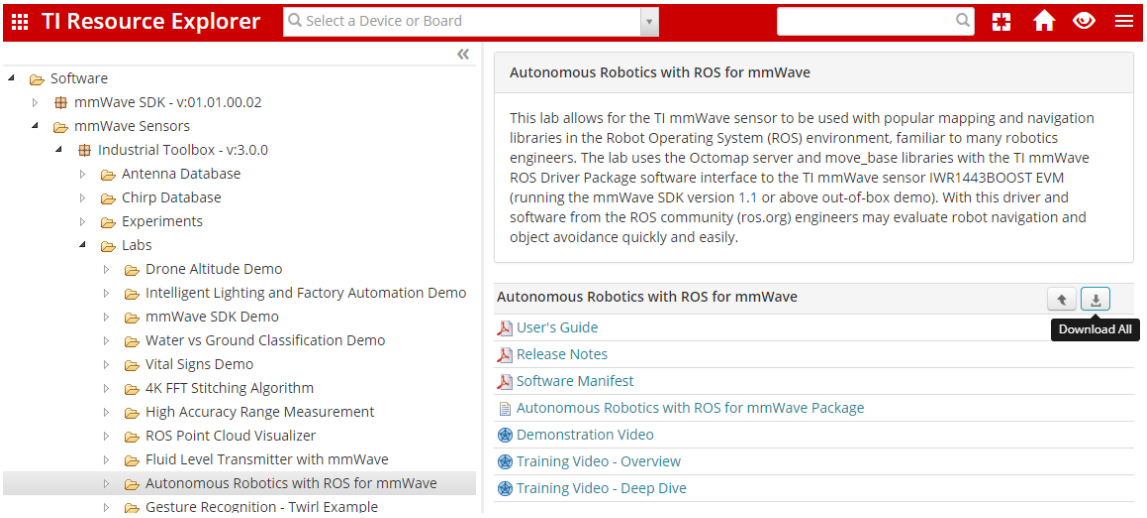
    a) Point your web browser to the following link: http://dev.ti.com/tirex/#/?link=Software

    b) Click on “mmWave Sensors” and then click on “Industrial Toolbox”

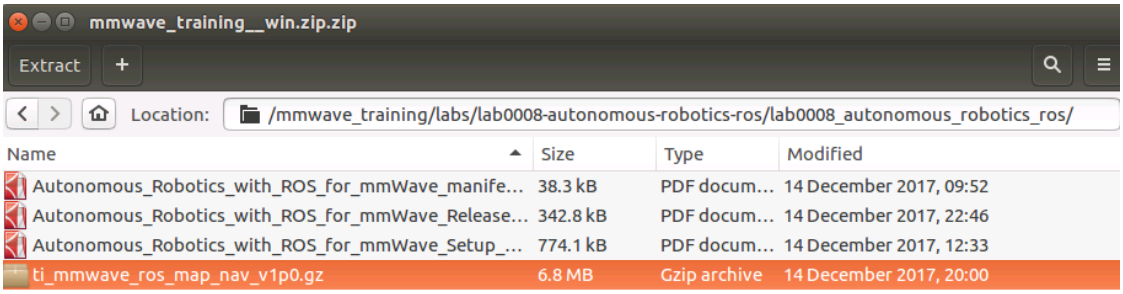
    c) Click on the link for “Labs”

d) Click on the “Autonomous Robotics with ROS for mmWave” lab in the table in the center

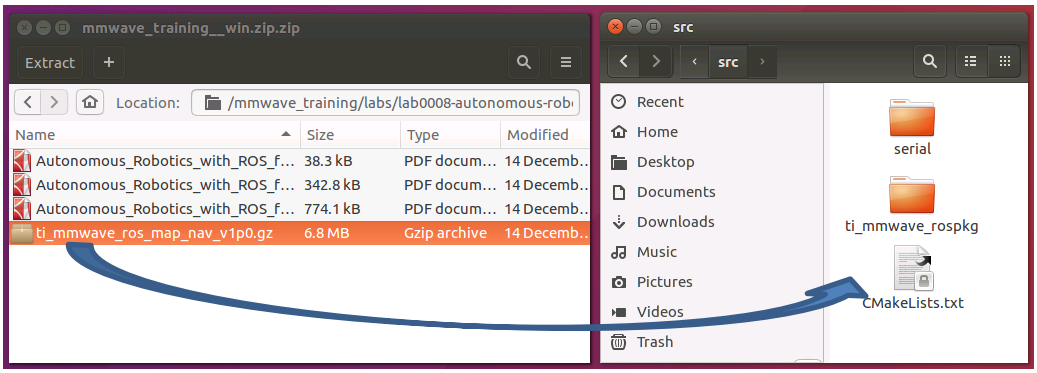
e) Click on the “Download All” button  on the right side as shown in the following image and accept the user license agreement when prompted. Choose to save the downloaded zip file if prompted or save it to disk first and then open it by double-clicking it from the downloaded location.



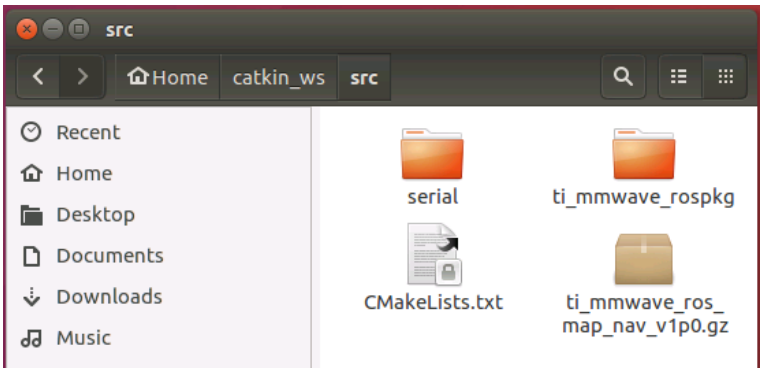
f) Navigate into the folder structure of the opened zip file to the path shown in the following figure. (Note that the path may start with mmwave_sensors/industrial_toolbox_<ver> depending on package version.)



g) Open a new file browser window and navigate to your <workspace_dir>/src directory as shown in the right window in the following screenshot. Copy the ti_mmwave_ros_map_nav_<ver>.tar.gz file to your <workspace_dir>/src directory by dragging it from the source (left) window to the destination (right) window.



h) You should now see the ti_mmwave_ros_map_nav_<ver>.tar.gz file in your <workspace_dir>/src directory as shown in the following screenshot.



2. Extract the turtlebot, navigation, and turtlebot_mmwave_launchers folders (as well as a custom mmWave chirp config file which gets placed in the ti_mmwave_rospkg/cfg directory) from the archive (.tar.gz) file using the following command executed from the ${WORKSPACE}/src directory. Change <ver> to match the actual filename.

```
$ tar xzf ti_mmwave_ros_map_nav_<ver>.tar.gz
```

3. Go back to your ${WORKSPACE} directory and build the workspace:

```
$ cd <workspace_dir>
$ catkin_make
```

If all of the installation steps were followed and both ROS environment scripts were sourced the driver should build successfully and you should see [100%] on the lines at the end of the build output.

Networking

ROS is a distributed system, meaning that it can communicate over a local network with other ROS components. For this demo, both laptops mentioned above must be on the exact same network and must be able to ping (icmp) each other by IP address. The Remote Control laptop must also be able to ssh (tcp/ip) into the TurtleBot laptop by IP address. You may need to install ssh on the laptops using the following command:

```
$ sudo apt-get install ssh
```

For more information regarding ROS's networking visit the link below:

http://wiki.ros.org/ROS/NetworkSetup

Additionally a ROS system may only have one "roscore" across all machines. In order for machines to recognize this "roscore" they must have an environment variable defined which specifies the IP address of the "roscore" machine.

On the TurtleBot machine

Edit your ~/.bashrc file to include the following lines at the bottom:

```
export ROS_MASTER_URI=http://localhost:11311

export ROS_IP=<IP_OF_THIS_MACHINE>
```

You can check your IP by running `$ ifconfig` on the command line. Note that the line exporting the ROS_IP environment variable may not be required if your network is setup where each machine can contact/ping the other by hostname. You must close and re-open the shell for the updated ~/.bashrc file to take effect.

On the Remote machine

Edit your ~/.bashrc file to include the following lines at the bottom:

```
export ROS_MASTER_URI=http://<IP_OF_TURTLEBOT_MACHINE>:11311

export ROS_IP=<IP_OF_THIS_MACHINE>
```

You can check your IP by running `$ ifconfig` on the command line. Note that if your network is setup where each machine can contact/ping the other by hostname then you can use `<NAME_OF_TURTLEBOT_MACHINE>` instead of `<IP_OF_TURTLEBOT_MACHINE>` in the first line and the line exporting the ROS_IP environment variable may not be required. You must close and re-open the shell for the updated ~/.bashrc file to take effect.

TurtleBot Bring-up and Teleoperation (Remote Control)

These steps must take place from the remote machine, "ssh-ing" into the TurtleBot laptop when necessary.

TurtleBot Bring-up

To start the TurtleBot, open a terminal window on the remote machine, ssh into the TurtleBot laptop and run the following command.

For IWR6843:

```
$ roslaunch turtlebot_bringup minimal.launch mmwave_device:=6843
```

For IWR1443:

```
$ roslaunch turtlebot_bringup minimal.launch mmwave_device:=1443
```

If the EVM was not in a good state the roslaunch command will fail. Try resetting the EVM by pressing the 'NRST' button on the EVM and then run the desired roslaunch command again.

After the Turtlebot and mmWave sensor are configured, you may see periodic "Kobuki : malformed sub-payload detected" errors. These errors appear to come from the Turtlebot driver and do not affect the operation of the demo.

Teleoperation (Remote Control)

Open a new terminal window on the remote machine and run the following command to bring up the teleoperation (remote control) of the TurtleBot:

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

Follow the instructions shown in the window to control the TurtleBot. You can exit out of the remote control application by pressing CTRL-C.

Mapping Demo

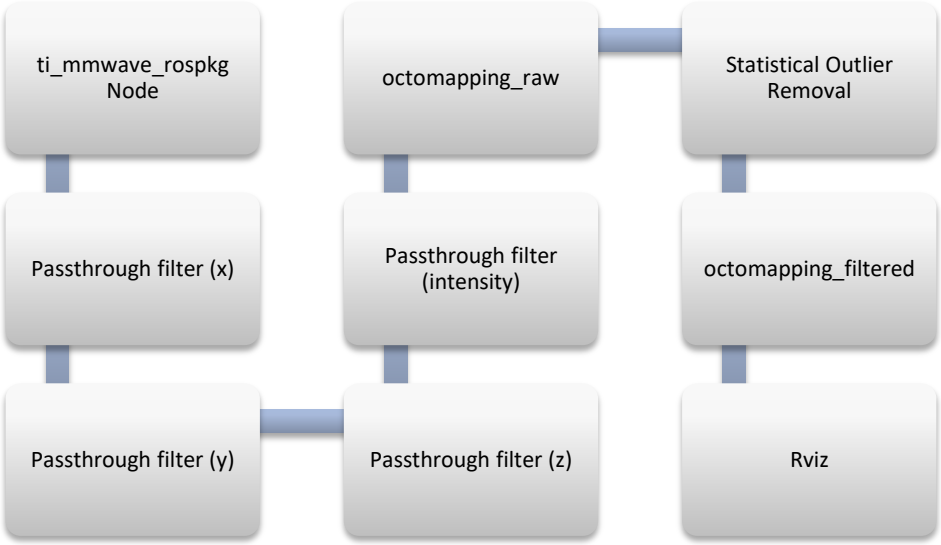
The Mapping Demo is an example of how to use TI's mmWave Radar EVMs within the ROS framework on a robot to build a map.

The demo runs the octomap_server package in ROS. There are several filters that have been brought up for use: PassThrough filters for all the point cloud fields, and a Statistical Outlier Removal filter for filtering the raw map. The parameters/limits for these filters can be found and modified in the "turtlebot_mmwave_launchers/launch/radar_limit_filters.launch" file. The radar_mapping.launch file mentioned below must be re-started after changing the radar_limit_filters.launch file in order for the new parameters to take effect.

Pass Through Filters: remove values outside a certain range for a given field (X, Y, Z, intensity)

Statistical Outlier Removal: removes values based off their distance from their closest neighbors

The processing graph for the incoming point cloud data is shown below:



Start-up

To run the mapping demo, first follow the TurtleBot Bring-up and Remote Control instructions in the "TurtleBot Bring-up and Teleoperation (Remote Control)" section to bring up and remote control the TurtleBot. Then, to run the mapping demo open a new terminal window, ssh into the TurtleBot laptop and run the following command:

```
$ roslaunch turtlebot_mmwave_launchers radar_mapping.launch
```

You may see a few warnings immediately after the launch file is run. It may also output a warning saying "Nothing to publish, octree is empty" whenever there are no objects detected in front of the mmWave sensor.

Visualization

To view the TurtleBot, Radar data, and map data in Rviz open a new terminal window on the remote machine and run the following command which will open a pre-defined Rviz configuration customized for the mapping demo (the command is all one line):

```
$ rosrun rviz rviz -d ~/catkin_ws/src/
/turtlebot_mmwave_launchers/launch/mapping_visualization.rviz
```

Alternatively, you can run the following command to open a blank Rviz screen and then manually add the same topics to the visualization as follows: `$ rosrun rviz rviz`

Once Rviz has started add the radar data by selecting Add-> PointCloud2 and selecting /mmWaveDataHdl/RScan under the Topic dropdown menu for the PointCloud2. Changing the Size to 0.03, Style to Spheres, and Decay to 0.25 will improve the visualization.

To see the TurtleBot on the screen select Add->Robot Model and Rviz will automatically detect the robot description and display the TurtleBot. To view the path the TurtleBot is taking select Add->Path and choose /trajectory under the Topic dropdown for the Path.

To visualize the octomap output in Rviz, select Add->PointCloud2 and select <raw_or_filtered>_point_cloud_centers and change the ColorTransformer to AxisColor to color points by elevation. Changing the Size to 0.03 and Style to Spheres will improve the visualization.

You can also save a custom Rviz configuration and load it in the future for convenience.

Saving a Map

The TurtleBot laptop is now running the Radar and TurtleBot drivers, as well as all the filtering and mapping nodes shown in the flow chart above. Move the TurtleBot around your environment manually until you are satisfied with your map. Then open a new terminal window on the remote machine, ssh into the TurtleBot laptop, and run the following command to save the map:

```
$ rosrun octomap_server octomap_saver –f <your_file_name>.bt
```

Viewing a Previously Saved Map

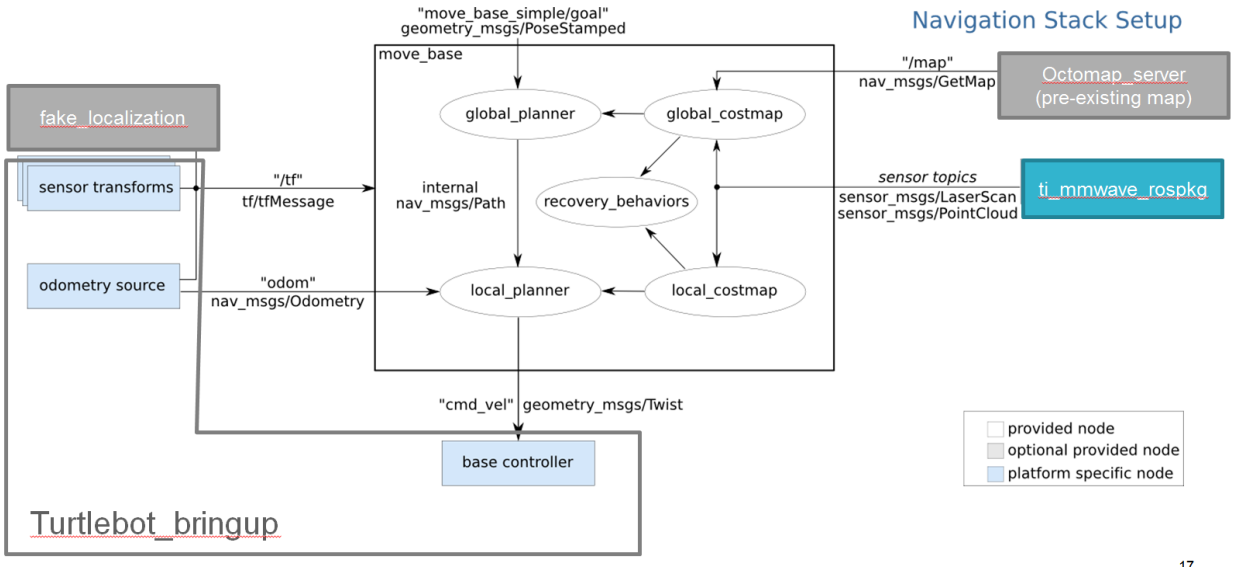
To view a saved <map>.bt file, you must first shutdown the mapping demo launch file if it was launched by pressing CTRL-C in the window where the radar_mapping.launch file was launched. Rviz will need to be opened and configured as mentioned in the Visualization section above if it is not already open. Then, open a new terminal window on the remote machine, ssh into the TurtleBot laptop, and run the following command to serve the saved map (the command is all one line):

```
$ rosrun octomap_server octomap_server_node /path/to/<map>.bt
octomap_point_cloud_centers:=filtered_point_cloud_centers
```

You should see the saved map displayed in Rviz. Note that the other topics in Rviz may show a warning/error since they are not active.

Navigation Demo

The TurtleBot navigation demo runs on the nodes mentioned in the image below.



Here are the steps to run the navigation demo.

1. Close all previous terminal windows if any were open

2. Open a new terminal window on the remote machine, ssh into the TurtleBot laptop and bring up the TurtleBot and mmWave EVM with the following command.

   For IWR6843:

   ```
   $ roslaunch turtlebot_bringup minimal.launch mmwave_device:=6843
   ```

   For IWR1443:

   ```
   $ roslaunch turtlebot_bringup minimal.launch mmwave_device:=1443
   ```

   If the EVM was not in a good state the roslaunch command will fail. Try resetting the EVM by pressing the 'NRST' button on the EVM and then run the desired roslaunch command again.

3. To bring up the move_base and fake_localization nodes and load a prebuilt map using the octomap_server, open a new terminal window on the remote machine, ssh into the TurtleBot laptop and run the following command:

```
$ roslaunch turtlebot_mmwave_launchers radar_navigation.launch
```

Note1: By default, this launch file loads a specific prebuilt map file containing a map of a simple rectangular space roughly 4ft x 6ft which can be used to make the robot stay within a space of that size for demo and testing purposes.

To load your own map, edit the radar_navigation.launch file and change the map filename shown in bold below to your own saved map file.

<node name="octomap_server" pkg="octomap_server" type="octomap_server_node" args**="$(find turtlebot_mmwave_launchers)/launch/map_4ft_by_6ft_border_large.bt** projected_map:=map" />

Alternatively, this static map can be completely disabled/removed as follows if desired to allow the robot to plan paths freely without any artificial boundaries.

- Remove the line containing “costmap_2d::StaticLayer” in the global_costmap_params.yaml and local_costmap_params.yaml param files
- Remove the line containing “octomap_server_node” in the ~catkin_ws/src/turtlebot_mmwave_launchers/launch/radar_navigation.launch file
- The size of the global and local costmaps can also be increased in the global_costmap_params.yaml and local_costmap_params.yaml param files to allow setting goals that are farther away

Note2: You may see “octree is empty” warnings when there are no objects detected in front of the mmWave sensor.

4. Open a new terminal window on the remote machine and run the following command which will open a pre-defined Rviz configuration customized for the navigation demo.

   To show the inflation layer which is used for path planning (the command is all one line):

   ```
   $ rosrun rviz rviz -d ~/catkin_ws/src/
   /turtlebot_mmwave_launchers/launch/navigation_visualization.rviz
   ```

   To not show the inflation layer which is used for path planning (the command is all one line):

   ```
   $ rosrun rviz rviz -d ~/catkin_ws/src/
   /turtlebot_mmwave_launchers/launch/navigation_visualization_2.rviz
   ```

   Alternatively, you can run the following command to open a blank Rviz screen and then manually add the same topics to the visualization as follows:

   ```
   $ rosrun rviz rviz
   ```

   “Add” three Map displays, one PointCloud2, the Robot Model, and one PosewithCovariance. Use the Topic dropdowns on the displays you added in Rviz to select topics for each one. For the maps,

choose “map” for one and the local and global cost maps for the other two. Select /initialpose for PosewithCovariance. You can select /mmWaveDataHdl/RScan or /xyzi_filt_out for the PointCloud2. Changing the Size to 0.03, Style to Spheres, and Decay to 0.25 will improve the PointCloud2 visualization. You can also add two Path topics (one for /move_base/DWAPlannerROS/local_plan and one for /move_base/DWAPlannerROS/global_plan) and a Polygon for /move_base/local_costmap/footprint.

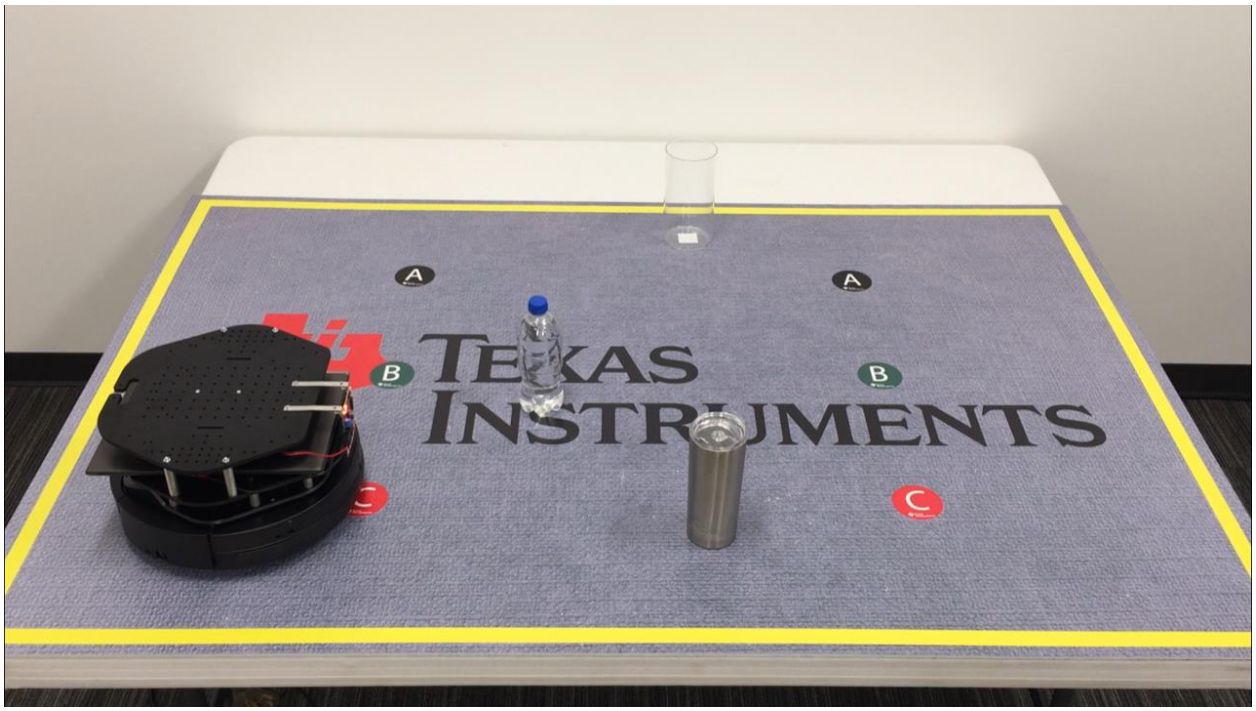
There is also an alternate pre-defined Rviz configuration file as an additional visualization example located at “turtlebot_mmwave_launchers/launch/navigation_visualization_2.rviz” that adds a PointCloud for the voxel grid (/move_base/local_costmap/obstacle_layer/marked_cloud) and disables the costmaps.

5. Start the navigation by first providing an initial pose estimate. Select 2D Pose Estimate (along the top of screen) and click on the location where the TurtleBot is within the map and drag in the direction it is facing. You should see the TurtleBot appear on the same spot you click immediately after releasing.

   Note: Please do not choose a starting position or goal that is too close to the boundaries of the current map as the navigation stack will not be able to create what it considers to be a valid path.

6. Now, give the TurtleBot a navigation goal by selecting 2D Nav Goal and clicking the location you would like the TurtleBot to navigate towards and dragging in the direction you would like it to face. The TurtleBot should then begin navigation to its goal. If you need to stop it, terminate the radar_navigation.launch roslaunch command by clicking on the terminal window that was used to launch it and pressing CTRL-C.

7. As an alternative to manually specifying the initial pose and goal using Rviz, there is also an example shell script in the “turtlebot_mmwave_launchers/scripts” directory that you can call from the Linux command prompt to set the initial pose and goal. It is intended to be used with the default 4ft x 6ft prebuilt map where you start the robot on one of three starting points (a, b, or c) at one end of the rectangular space facing the opposite end and then send it to one of the three goal points at the other end. This is useful for demonstrating obstacle avoidance in a 4ft x 6ft space as shown in the following picture. To run the script, go to the “turtlebot_mmwave_launchers/scripts” directory and type “./start_nav.sh” at the Linux prompt.


A
A
B
B
C
C
TEXAS
INSTRUMENTS