mmWave sensors in robotics: technical deep dive

Texas Instruments

Detailed agenda

- mmWave Sensing in Robotics – how do robots “see” using mmWave?
  - Overview and market differentiation
  - mmWave Demo Visualizer
  - ROS (Robot OS) Point Cloud Visualizer lab on TI Resource Explorer
- Autonomous robot demonstration using ROS + TI mmWave sensor (IWR1443)
- Technical Deep Dive
  - Tuning the mmWave sensor configuration for a specific application
  - How the “Autonomous Robotics with ROS for mmWave” demo works
  - Tuning Robot OS parameters in the demo

Technical deep dive

- **Tuning the mmWave sensor configuration for a specific application**
- How the “Autonomous Robotics with ROS for mmWave” demo works
- Tuning Robot OS parameters in the demo

Tuning the mmWave sensor configuration for a specific application

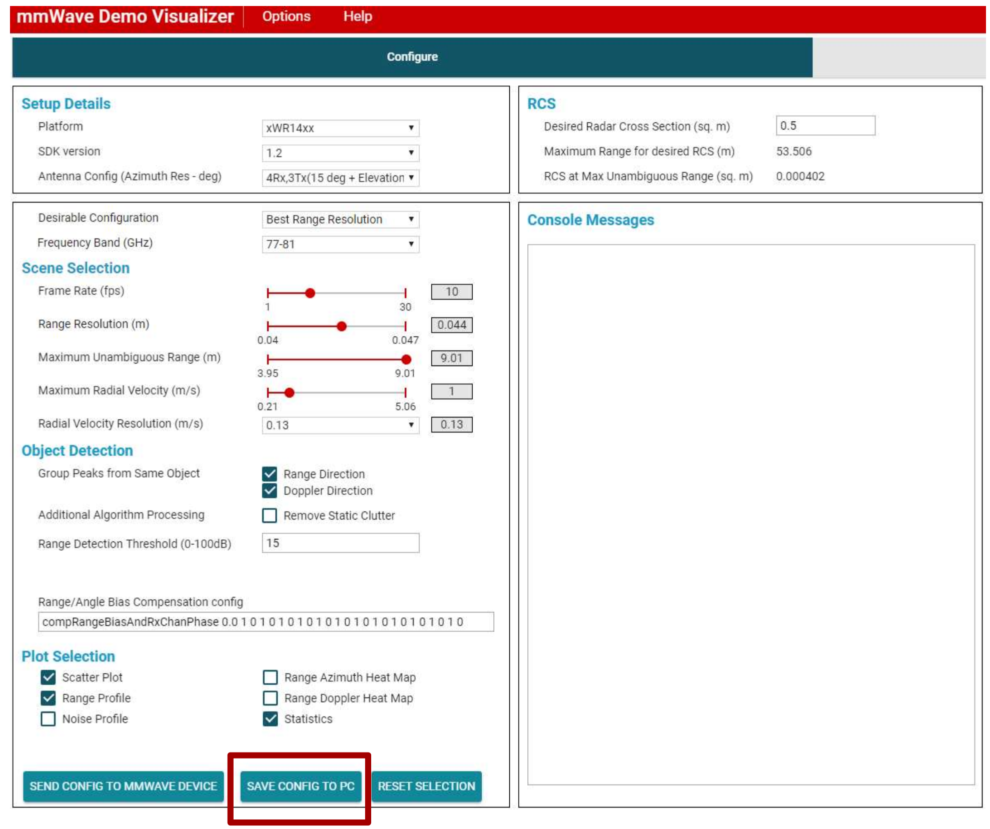
- Goal is to create a mmWave sensor chirp configuration that satisfies the sensing needs of the application

Development Stage	Method
Discover/Evaluate ↓	**mmWave Demo Visualizer** can be used to create and test a chirp configuration that will work with the mmWave SDK out-of-box demo which is used in the “Autonomous Robotics with ROS for mmWave” lab
Evaluate/Design	**mmWave Sensing Estimator** can be used to create more advanced/customized chirp configurations that may not work with the out-of-box demo and instead require a custom processing chain

Creating a chirp configuration with mmWave Demo Visualizer

Steps to create a chirp configuration:

- Select the desired parameters on the Configure tab in the mmWave Demo Visualizer
- Verify that the config works by sending the config to the mmWave device and review the live plots on the Plots tab
- Press the “Save Config to PC” button to save the chirp config to a file
- If you are on a Windows computer, copy the chirp config file to the Linux PC connected to the mmWave EVM

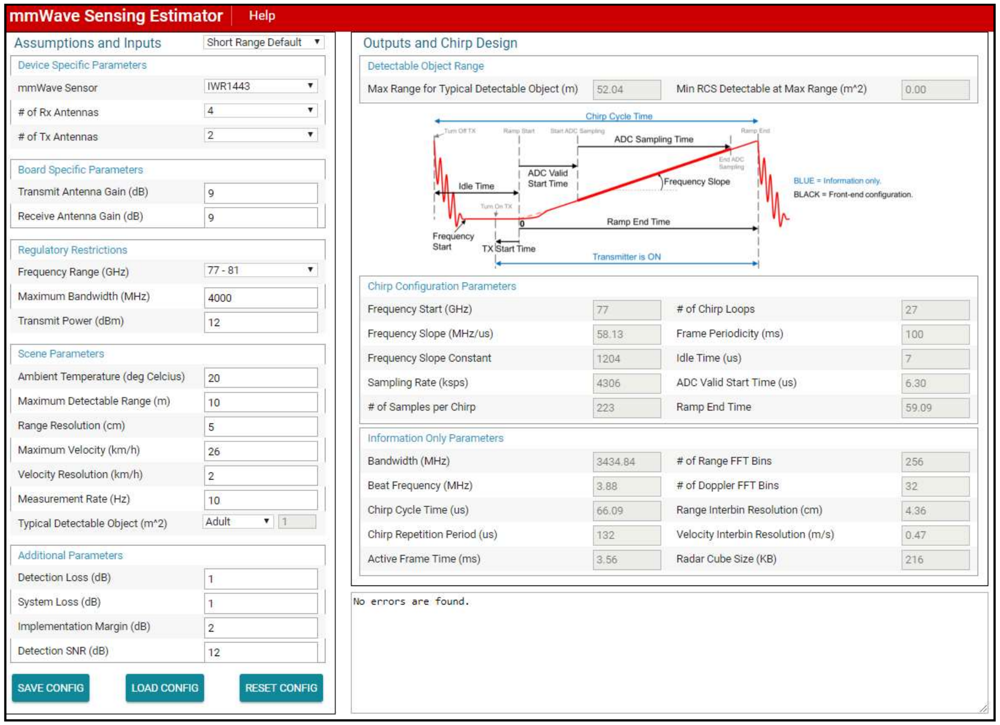


Using the chirp configuration from the mmWave Demo Visualizer

- Chirp configurations generated from the mmWave Demo Visualizer can be used directly by the Robot OS mmWave labs
- It is possible to reconfigure the mmWave sensor after it is already running as long as the previous config used the same number of TX and RX antennas (command is given in ROS Point Cloud Visualizer user guide)
- It is also possible to replace the default config so that the new config is loaded when the lab is started (required if the number of TX/RX antennas differs)
  – Place the new chirp config file in the "~/catkin_ws/src/ti_mmwave_rospkg/cfg" directory
  – Rename the original default config file to a different name
  – Rename the new config file to match the original default config file

Creating a chirp configuration with mmWave Sensing Estimator

- mmWave Sensing Estimator can be used to create more advanced / customized chirp configurations
- Inputs
  - Device type and number of antennas
  - Board-specific antenna gains
  - Regulatory Restrictions
  - Scene Parameters
  - Additional Parameters
- Outputs
  - Chirp Configuration Parameters
  - Information Only Parameters
  - Detectable Object Range (also for information only)
  - Errors (if config is not valid)



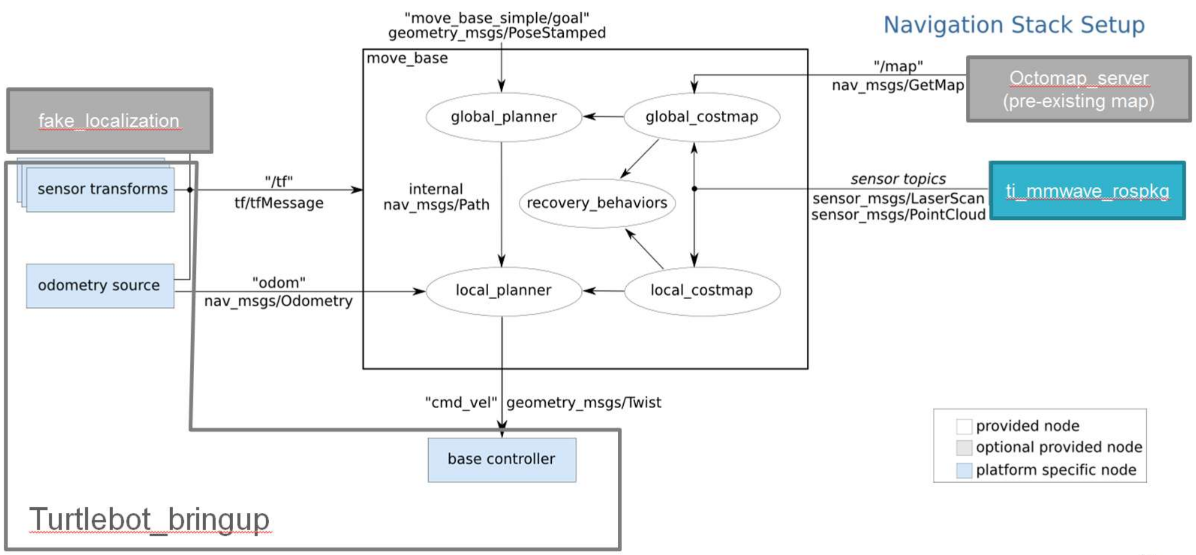
Using the chirp configuration from the mmWave Sensing Estimator

- Chirp configurations generated from the mmWave Sensing Estimator may require a custom processing chain
- In that case, the mmWave SDK out-of-box demo code would need to be modified as follows:
  - If the command-line interface (CLI) rejects the new chirp config parameters, it would need to be modified
  - The mmWave SDK processing chain would need to be modified to support the dataflow, timing, and signal processing requirements for the new chirp configuration
- In order to work with the Robot OS mmWave labs, the modified mmWave SDK demo must still output the same detected object data format over the UART

Technical deep dive

- Tuning the mmWave sensor configuration for a specific application
- **How the "Autonomous Robotics with ROS for mmWave" demo works**
- Tuning Robot OS parameters in the demo

How the "Autonomous Robotics with ROS for mmWave" demo works



- ROS move_base navigation package
  - Global costmap/planner to plot path
  - Local costmap/planner to account for robot movement capabilities
  - Added recovery behavior to clear obstacle map and re-scan when no path found
- ROS fake_localization package
  - Allows user to set initial location and desired destination on map
- Sensors
  - Odometry to track location
  - TI mmWave for obstacle detection
- Map server
  - Made static so it can be used to define constrained area for robot to stay within

Technical deep dive

- Tuning the mmWave sensor configuration for a specific application
- How the “Autonomous Robotics with ROS for mmWave” demo works
- **Tuning Robot OS parameters in the demo**

Tuning Robot OS parameters in the demo

- ROS navigation stack is described at: http://wiki.ros.org/navigation
- In the demo, Robot OS navigation stack parameter files are located at: ~/catkin_ws/src/turtlebot/turtlebot_apps/turtlebot_navigation/param/
- Parameter files:
  - costmap_common_params.yaml
  - global_costmap_params.yaml
  - global_planner_params.yaml, navfn_global_planner_params.yaml
  - local_costmap_params.yaml
  - dwa_local_planner_params.yaml
  - move_base_params.yaml
  - radar_costmap_params.yaml (used to override settings in other param files)

Important tuning parameters for local planner

- dwa_local_planner_params.yaml
  - Parameters described at: http://wiki.ros.org/dwa_local_planner?distro=kinetic
  - max_vel_x = maximum forward velocity (m/s)
  - acc_lim_x = maximum forward acceleration (m/s^2)
  - max_rot_vel = maximum rotational velocity (rad/s)
  - acc_lim_theta = maximum rotational acceleration (rad/s^2)
  - xy_goal_tolerance = tolerance (in meters) when trying to reach goal

Important tuning parameters for costmaps

- costmap_common_params.yaml (radar_costmap_params.yaml overrides the global costmap inflation_layer parameters)
  - Parameters described at: http://wiki.ros.org/costmap_2d?distro=kinetic (in section "8.2 Layer Specifications")
  - robot_radius = radius of robot (m)
  - obstacle_layer
    - z_voxels = number of cells in z-axis in 3D costmap occupancy "voxel" grid (max = 16)
    - z_resolution = z resolution of 3D occupancy "voxel" grid (meters/cell), height of the grid is z_resolution * z_voxels
    - mark_threshold = maximum number of marked cells allowed in a column considered to be "free“ (i.e. if more cells in column are marked, then the X/Y grid location is considered to be occupied)
    - obstacle_range = maximum distance from the robot at which an obstacle will be inserted into the cost map (m)
  - inflation_layer
    - inflation_radius = max distance from an obstacle at which costs are incurred for planning paths
    - cost_scaling_factor = exponential rate at which the obstacle cost drops off

Customer collateral

Content type	Content title	Link to content or more details
Labs on TI CCS Resource Explorer	ROS Point Cloud Visualizer lab and Autonomous Robotics with ROS for mmWave lab	http://dev.ti.com/tirex/#/?link=Software%2FmmWave%20Sensors%2FIndustrial%20Toolbox (under Labs)
Customer training series	mmWave Training Series	https://training.ti.com/mmwave-training-series
Technical blog content or white paper	mmWave radar sensors in robotics applications	http://www.ti.com/lit/wp/spry311/spry311.pdf
Selection and design tools and models	mmWave Sensing Estimator	https://dev.ti.com/mmWaveSensingEstimator
	mmWave Demo Visualizer	https://dev.ti.com/mmWaveDemoVisualizer
Videos	mmWave Demo Visualizer video	https://training.ti.com/mmwave-sdk-evm-out-box-demo
	ROS Point Cloud Visualizer lab video	https://youtu.be/lNEGT10Mk9k
	Autonomous Robotics with ROS for mmWave lab video	https://training.ti.com/robotics-sense-and-avoid-demonstration-using-ti-mmwave-sensors
Product and EVM pages	IWR1443 product page IWR1443BOOST EVM page IWR1642 product page IWR1642BOOST EVM page	http://www.ti.com/product/IWR1443 http://www.ti.com/tool/IWR1443BOOST http://www.ti.com/product/IWR1642 http://www.ti.com/tool/IWR1642BOOST