Latest Features in Robotics System Toolbox

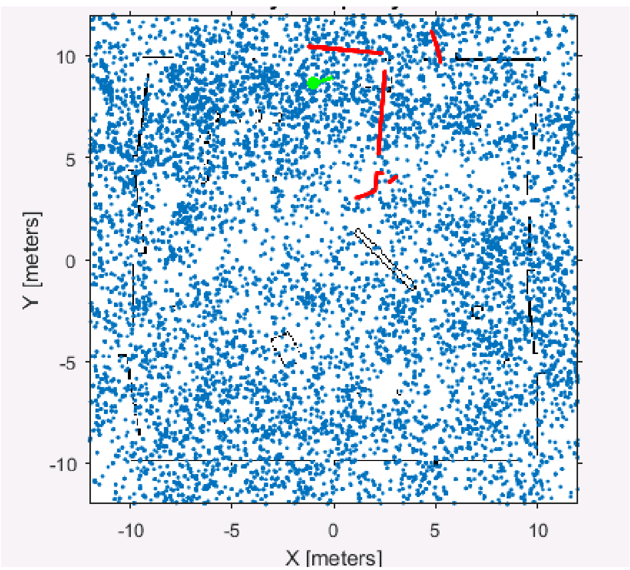
March 2016

R2016a

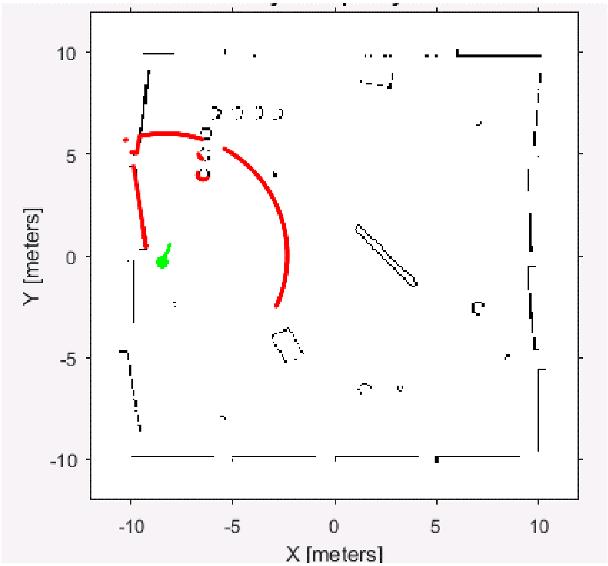
Monte Carlo Localization Algorithm

Estimate robot location in a known map

- Estimate pose (location and orientation) of a differential drive robot in a known environment using sensor data
- Provide BinaryOccupancyGrid object of your map and range sensor data from the robot to the `robotics.MonteCarloLocalization` object
- Use global localization or specify an initial pose to improve performance



Initial Distribution - Unknown Robot Position



Converged Distribution - Localized Robot

```
>> mcl = robotics.MonteCarloLocalization

>> [~, pose] = step(mcl, odom, ranges, angles)
```

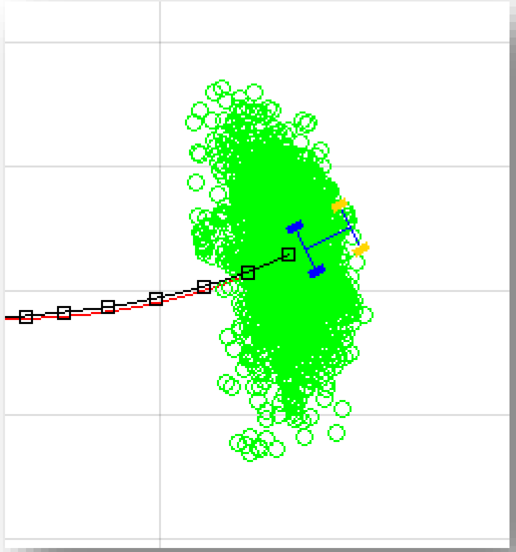
```
» edit TurtleBotMonteCarloLocalizationExample
```

Particle Filter Algorithm

Estimate state for nonlinear systems

- Estimate state for arbitrary non-linear systems and non-Gaussian noise distributions
- Apply particle filter to diverse applications such as robot pose estimation, object tracking, and sensor fusion
- Customize your particle filter by giving a state transition function and measurement likelihood model to match your system

Video Object Tracking



Robot Pose Estimation

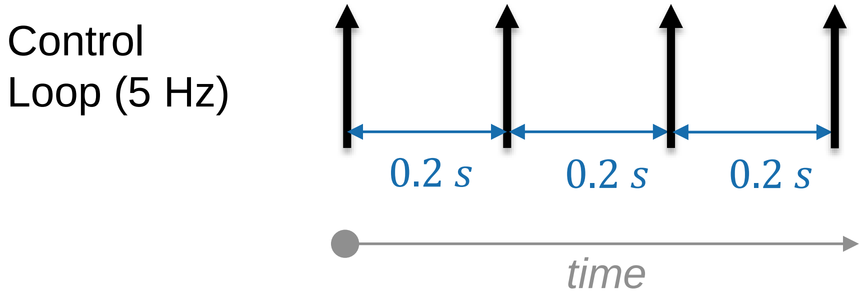
```
>> pf = robotics.ParticleFilter

>> predict(pf)
>> correct(pf, [0 0 pi])
```

```
» ParticleFilterExample
```

Fixed-Rate Execution

Run MATLAB code at a constant rate

- Execute loops at a constant rate based off either the system time or ROS time
  - Compensates for any user code to maintain the rate
  - Ensures that loops are run at a fixed rate when accurate timing of commands is required
- Collect statistics about the timing of loop iterations
- Use published simulation time when connected to a ROS network
  - Publish messages and control commands at a fixed rate to a ROS-enabled system



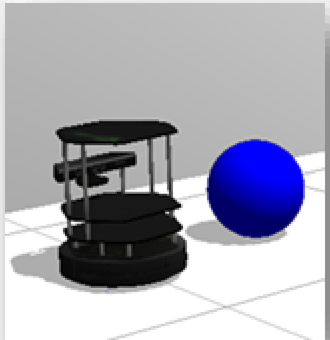

```
r = robotics.Rate(5);

% Run loop at 5 Hz
while(1)
    runUserCode();
    waitfor(r);
end
```

Robotics System Toolbox Support Package for TurtleBot Based Robots

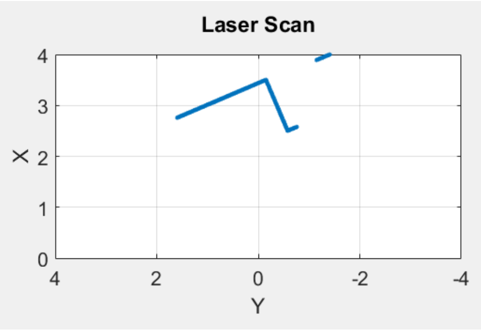
R2016a

Connect to TurtleBot hardware

- Acquire sensor data from TurtleBot based robots without explicitly calling ROS commands
  - Use the data for visualization and analysis, and send commands to control the robots
- Communicate with either simulated or physical robots

Communicate with a Physical or Simulated TurtleBot Robot



Visualize Sensor Data

```
>> tbot = turtlebot('192.168.2.100')

>> odom = getOdometry(tbot)
>> setVelocity(tbot, 0.2)
```

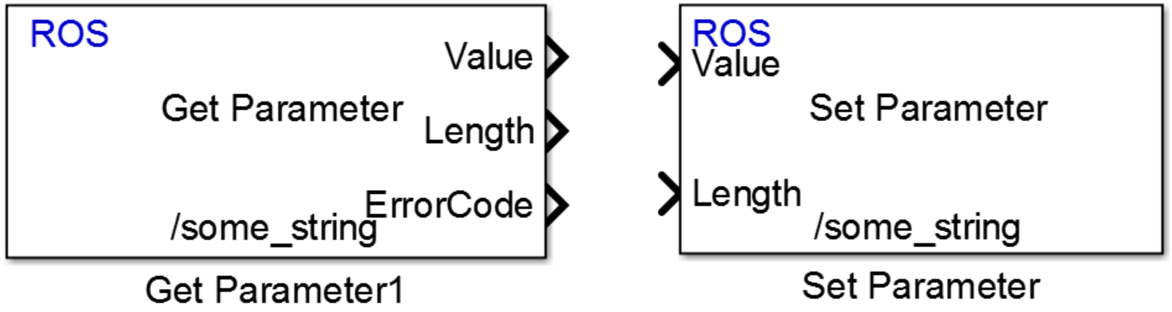
```
» TurtleBotSPGetStartedExample
```

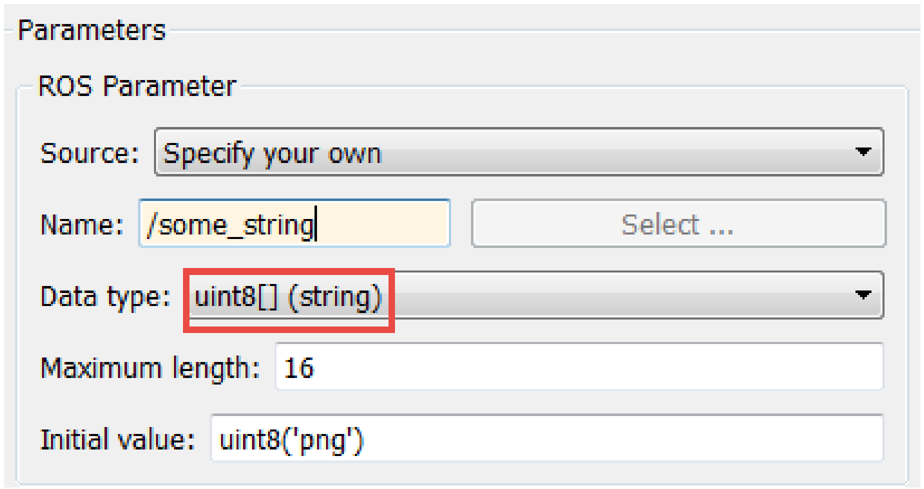
String Support for ROS Parameters in Simulink

R2016a

Support for using strings as ROS parameters

- Get ROS parameters that are strings and use them in your Simulink model
- Set ROS parameters that are strings
- When using strings, they must be cast as a uint8 array of ASCII values



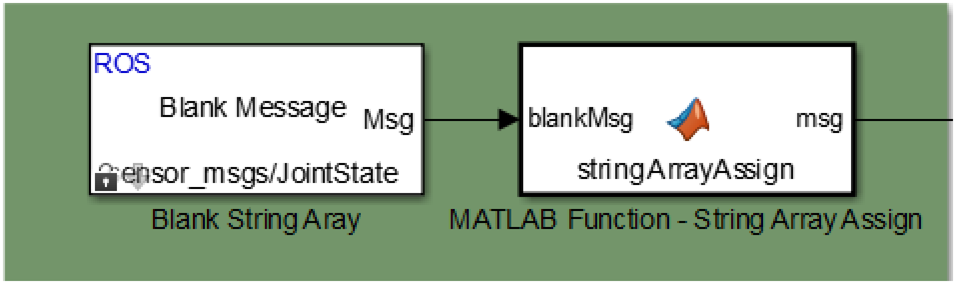



```
>> robotlib
```

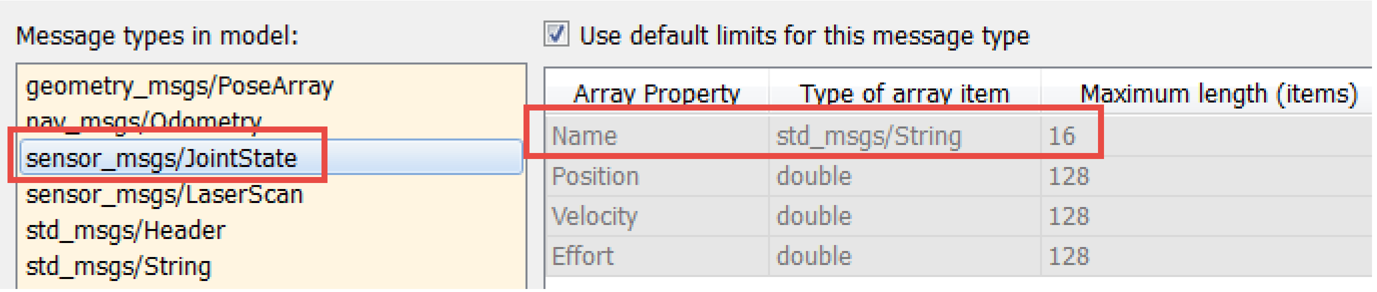
R2016a

String Array Support for ROS Messages in Simulink

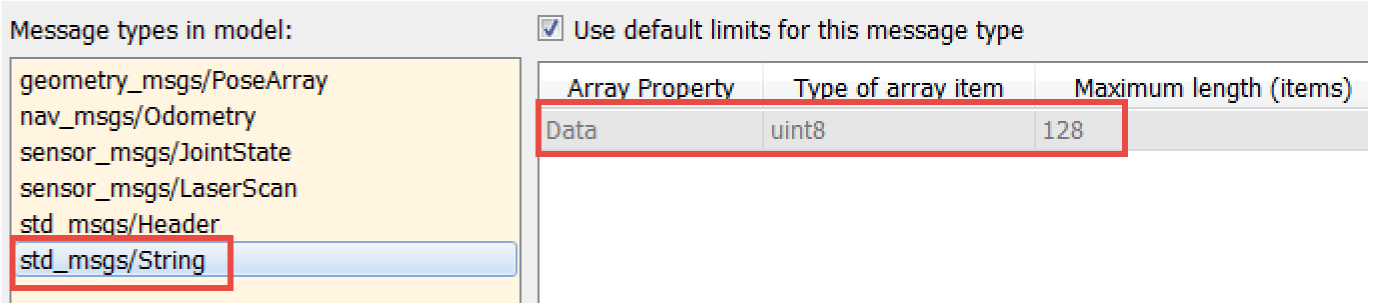
Support for using string arrays as ROS messages



- Use an array of strings when using the Publish, Subscribe, and Blank Message blocks to create, send, and receive messages using a ROS network in Simulink



- The size of variable-size arrays can be viewed and edited
  - *Tools > Robot Operating System > Manage Array Sizes*

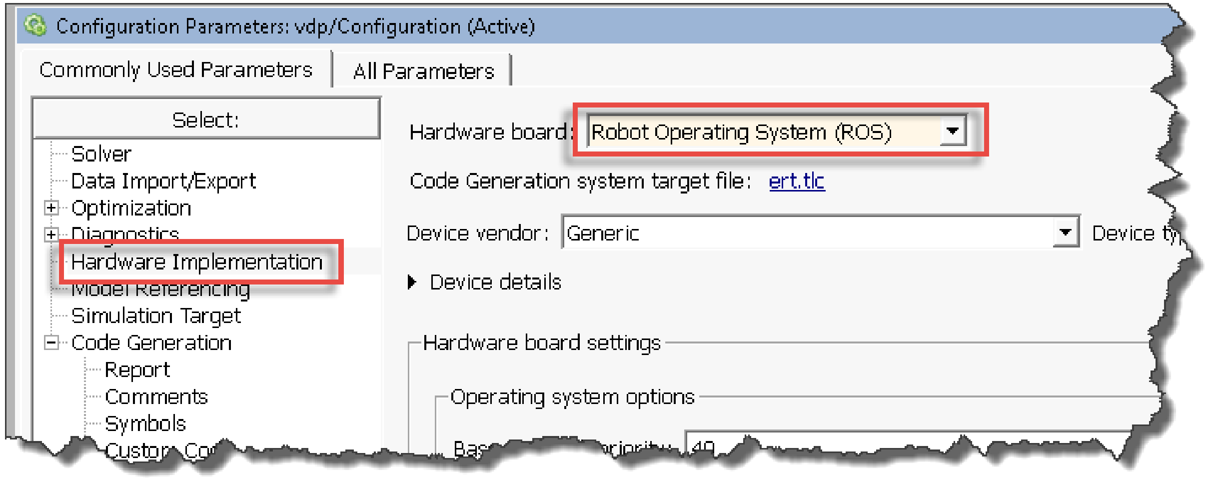


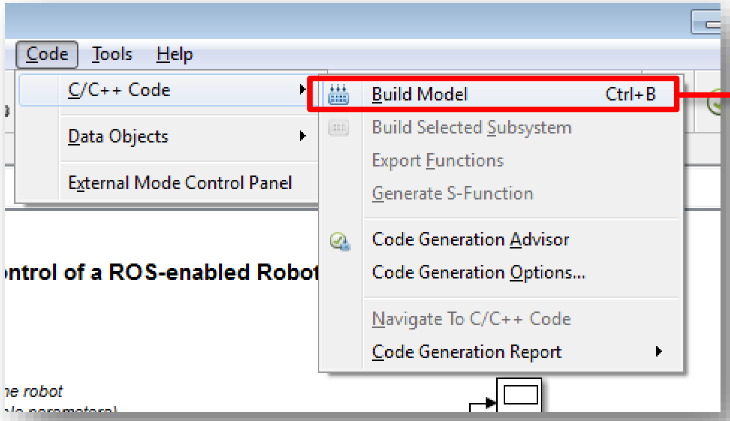
» `robotROSMessageUsageExample`

Code Generation from Simulink Using Simulink Coder

Generate standalone ROS nodes with Simulink Coder

- Generate standalone ROS nodes from Simulink models with just MATLAB Coder and Simulink Coder
- Embedded Coder can optionally be used to customize the generated code





» robotROSCodeGenerationExample