Connecting MATLAB and ROS for Autonomous Driving

Harshita Bhurat
Robotics System Toolbox Product Manager

Carlos Santacruz-Rosero, Ph.D.
Senior Application Engineer – Robotics
Agenda

- Automated Driving at BMW
- ROS for Automated Driving
- Robotics Workflow at the MathWorks
Automated Driving with ROS at BMW

** MICHAEL AEBERHARD, BMW, ROSCON 2015 **
Automated Driving with ROS at BMW

Using Robotics System Toolbox

USING MATLAB/SIMULINK WITH ROS.

- MathWorks released the Robotics System Toolbox this year for ROS integration with Matlab/Simulink.
- Easily read and analyze data from ROS Bags → useful for evaluating the system.
- Some of our software is implemented as a Simulink model.
- Use the Toolbox to easily integrate this software into the ROS eco-system.

http://www.mathworks.com/products/robotics/
What is ROS (Robot Operating System)?

- An architecture for distributed inter-process communication
- Packages for common algorithms and drivers
- Multilanguage interface (C++, Python, Lua, Java and MATLAB)
ROS Trends in Robotics Development

- #1 middleware for robotics applications development
- Popular in research and gaining great momentum in industry

http://rosindustrial.org/ric-americas/
Benefits of ROS for Automated Driving

• Sensor data acquisition
• Reliable distributed architecture
• Lots of “off the shelf” algorithms for autonomy
  • Path planning
  • Collision avoidance
  • Sensor processing
• Simplified component compatibility through standalone interfaces
• Integration with simulation environments
Autonomous Car as an Advanced Robotics System

- **Planning**
- **Localization**
- **Obstacle avoidance**
- **Global Map**

**NODE**

**Motion control**

**LIDAR** **Camera** **RADAR** **GPS/IMU**

**Motion Controllers**

- Actuator ECUs
- Steering Actuator
- GAS/Brake Actuator

ROS: communication framework and stack of libraries
Robotics Development Workflow with Hand Code

- Traceability to requirements
- Analyzing logged data
- Designing algorithm

- Translating to C/C++ , Python
- Visualizing and debugging

- Changing open-source code
- Integration into production
Robotics Development Workflow with MATLAB and MBD

RAPID ITERATIVE PROCESS

Idea

Requirements
- System Level Simulation
- Leverage built-in algorithms and Apps
- Co-simulation with ROS-Enabled Systems

Design and Simulation

Prototype
- C/C++ automatic code generation
- SIL with ROS Enabled Systems
- Debug C/C++ with MATLAB Engine

Implementation

Product

Design independent of communication framework!
Robotics Development Workflow with MATLAB and ROS

Simulate & Prototype

Visualize
Experiment
Test

ROS
Let’s solve a real problem: Sign Detection System

Clearpath Husky Robot
- ROS Enabled
- Kinect (RGB and Point cloud)
- Hokuyo 2D Lidar

Simulation model in Gazebo

Track and Park
Reduce Speed
Collision Avoidance
System Level Design

- ROS as Communication Framework
- State Machine
- Obstacle Avoidance
- Object Classifier
Sign Recognition with Collision Avoidance
Robotics Development Workflow with MATLAB and ROS

- Import image data from ROS bags
- Threshold and detect features in images
- Train a classifier
- Test algorithm
Importing Simulation and Experimental Data

Import ROS Data

Filter your logged field data by topic or time interval

Experimental data or simulation Data

Data ready to design algorithms

Robotics System Toolbox™

```matlab
%% Load the file
carData = rosbag('\car_field_test_042016.bag');

%% Select all messages on the scan topic
odomMsg = select(carData, 'Topic', '/scan');

%% Get all RGB camera points
imagMsg = select(carData, 'Topic', '/camera/rgb/image_raw');
```
Visualize, Analyze, and Process Data: Classifier

Input

Training data

Preprocessing

Feature Extraction

Training

Classifier

stop
speedlimit
membrane
neg

% Detect red regions
BW = createMask(videoFrame);

% Fill image regions
BW = imfill(BW, 'holes');

% Get bounding boxes
stats = regionprops('table', BW, 'BoundingBox', 'Area');

% Filter based on area size
targetIndex = stats.Area > 500;

% Get bounding boxes from detected regions
testFeatures(k,:) = extractHOGFeatures(Icr);
Visualize, Analyze, and Process Data: Classifier

Training data

Input

Preprocessing

Feature Extraction

Training

Classifier

Output

Computer Vision System Toolbox™

Statistics and Machine Learning Toolbox™
Design and Test Algorithm

%% Create classifier object
sd = SignDetector();

%% Test algorithm with two stops signs
In = imread('gazeboTwo StopsTest.png');

%% Show algorithm result
[out,~,~,~,~] = sd.step(In);
imshow(out);
Design and Test Algorithm
Robotics Development Workflow with MATLAB and ROS

RAPID ITERATIVE PROCESS

- Test algorithm with an external simulator
- Tune your algorithm
- Integrate with other algorithms
MATLAB and Simulink connect to the ROS network

- Multiple master support
- ROS publishers/subscribers
- ROS services
- ROS TF tree
- ROS Parameter server
Co-simulation with ROS

%% Connect to ROS
rosinit '192.168.204.144';

%% Create subscribers
imSub = rossubscriber('/camera/rgb/image_raw');
scanSub = rossubscriber('/scan');

%% Create publisher
[velPub, velMsg] = rospublisher('/husky_velocity_controller/cmd_vel');
Co-simulation with ROS
Leverage Built-in Algorithms in MATLAB

% Sensor fusion and pose estimation
>> pf = robotics.ParticleFilter;

% Robot localization
>> mcl = robotics.MonterCarloLocalization;

% Loops at system time or ROS time
>> r = robotics.Rate(5);

% Obstacle Avoidance
>> vfh = robotics.VectorFieldHistogram;
## Optional Products for Robotics Applications Development

<table>
<thead>
<tr>
<th>Image Processing Toolbox™</th>
<th>Computer Vision System Toolbox™</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Contrast adjustment</td>
<td>- High-speed video I/O</td>
</tr>
<tr>
<td>- Geometric transformations</td>
<td>- Point Cloud processing</td>
</tr>
<tr>
<td>- Various filters</td>
<td>- Tracking</td>
</tr>
<tr>
<td>- Segmentation</td>
<td>- Stereo vision</td>
</tr>
<tr>
<td>- Object analysis</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Acquisition Toolbox™</th>
<th>Statistics and Machine Learning Toolbox™</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Image capture from standard H/W</td>
<td>- Multivariate statistics</td>
</tr>
<tr>
<td>- Analog, Camera Link, DCAM, GigE Vision, USB camera, etc</td>
<td>- Probability distribution</td>
</tr>
<tr>
<td>- Microsoft Kinect Support</td>
<td>- Machine learning</td>
</tr>
<tr>
<td></td>
<td>- Experimental design</td>
</tr>
<tr>
<td></td>
<td>- Statistical process control</td>
</tr>
</tbody>
</table>
### Optional Products for Robotics Applications Development

<table>
<thead>
<tr>
<th>Control System Toolbox™</th>
<th>Simulink Design Optimization™</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Linear analysis</td>
<td>▪ Model parameter estimation from test data</td>
</tr>
<tr>
<td>▪ Classical control design</td>
<td>▪ Optimization of parameters</td>
</tr>
<tr>
<td>▪ Modern control design</td>
<td>▪ Response optimization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Robust Control Toolbox™</th>
<th>Simulink Control Design™</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Robust control design</td>
<td>▪ Automatic tuning of PID Controller blocks</td>
</tr>
<tr>
<td>▪ Automatic tuning of gain-scheduled controllers</td>
<td>▪ Linearization of Simulink models</td>
</tr>
<tr>
<td></td>
<td>▪ Continuous-Discrete time conversions</td>
</tr>
</tbody>
</table>
Final Design
Robotics Development Workflow with MATLAB and ROS

RAPID ITERATIVE PROCESS

Idea → Requirements

Requirements → Design and Simulation

Design and Simulation → Prototype

Prototype → Implementation

Implementation → Product

System level design to target a different middleware or framework
Deployment

- Generate ROS Node with Simulink and Embedded Coder™
- Generate a shared library with MATLAB Coder™
- Create a Stand Alone Executable with MATLAB Compiler™

Determine deployment methods based on application
Implementation
Takeaways

1. Robotics development workflow with MATLAB and Simulink for autonomous driving with ROS.

2. Rapid iterative process based on powerful visualization tools, interactive apps, built-in algorithms, debugging capabilities.

3. Design is independent of communication framework.

Call to Action! Check Demo
% Thank you