Bridging ROS with MATLAB and Simulink: From Algorithms to Deployment

Shashank Sharma
Application Engineer
MathWorks, Munich, Germany

YJ Lim, Ph.D
Sr. Product Manager
Robotics and Autonomous Systems
MathWorks, Natick, USA
Components of Autonomous Systems

Localization → Perception → Planning → Control

Perception

Localization

Planning

Control

Stateflow
Managing Complexity in Autonomous Systems
Common Challenges of Autonomous Robotics Development

Applying Multidomain Expertise

Features to design complex Algorithms

End-to-End workflows
In one development environment

Evaluate robot performance and operation before connecting to hardware
What does success look like?
Autonomous System Examples

German Aerospace Center (DLR)

Autonomous Robots
Senses the environment using stereo cameras and tactile sensors on his skin
Performs human-like tasks
Autonomous System Examples

Eelume snake robot in action
Bridging ROS with MATLAB & Simulink
From Algorithm Development To Deployment
Model-Based Design Value Proposition

- Requirements
- System Architecture
- Component Design
- Code Generation
- Modeling & Simulation
- Test & Verification
- Component Implementation
- V&V
- HIL Simulation
Autonomous Car as an Advanced Robotics System

ROS: communication framework and stack of libraries

NODE
Planning

NODE
Motion control

NODE
Localization

NODE
Obstacle avoidance

NODE
Global Map

LIDAR
Camera
RADAR
GPS/IMU

Motion Controllers
Actuator ECUs
Steering Actuator
GAS/Brake Actuator

NODE
NODE
NODE
NODE
NODE
Bridging ROS with MATLAB & Simulink
From Algorithm Development To Deployment

ROS bags (Log files)

Data analysis and playback

Desktop prototyping

Simulators
Hardware

ROS Nodes (Software)

Standalone node deployment

Controls
Perception
Planning and decision making...

ROS
ROS 2
Data: Logging, Streaming, Processing, and Analysis

- **Read ROS bag files into MATLAB**
  - MATLAB is good at data analysis (time series, geometry, sensors, etc.)

- **Play back ROS bag data into Simulink**
  - Simulink is good at time-based simulation and algorithm prototyping → deployment

- **ROS Toolbox handles specialized sensor message types**
  (images, lidar, point clouds, occupancy maps)

- **Many customers use ROS bags for offline design even if they don’t use ROS in production!**

https://www.slideshare.net/jwiegelmann/deep-learning-for-autonomous-driving
User Story: Clearpath Robotics
Accelerates Algorithm Development for Industrial Robots

Challenge
Shorten development times for laser-based perception, computer vision, fleet management, and control algorithms used in industrial robots

Solution
Use MATLAB to analyze and visualize ROS data, prototype algorithms, and apply the latest advances in robotics research

Results
- Data analysis time cut by up to 50%
- Customer communication improved
- Cutting-edge SDV algorithms quickly incorporated

“ROS is good for robotics research and development, but not for data analysis. MATLAB, on the other hand, is not only a data analysis tool, it’s a data visualization and hardware interface tool as well, so it’s an excellent complement to ROS in many ways.”
- Ilia Baranov, Clearpath Robotics

Link to user story
Desktop Simulation for Algorithm Development

- Live connectivity from MATLAB and Simulink to ROS and ROS2 (external simulators or hardware)
- Support desktop simulation with MATLAB and Simulink
- Leverage MATLAB, Simulink and Stateflow for Model-Based Design
MATLAB ↔ ROS Workflows

**SETUP**
- rosinit('ipAddress')
- odomSub = rossubscriber('/odom');
- [velPub, velMsg] = ...
- rospublisher('/mobile_base/commands/velocity');

**PROCESS**
- r = rateControl(10);
- while(r.TotalElapsedTime < 20)
  - odomMsg = odomSub.LatestMessage;
  - % INSERT YOUR ALGORITHM CODE HERE
  - velMsg.angular.z = ctrlOut;
  - send(velPub, velMsg);
  - plot(r.TotalElapsedTime, ctrlOut)
  - waitfor(r)
end

**CONTROL**
- node_1 = ros2node('node_1', domainId);
- node_2 = ros2node('node_2', domainId);

**SENSE**
-odomSub = ros2subscriber(node_1, '/odom');
- [velPub, velMsg] = ...
- ros2publisher(node_2, '/mobile_base/commands/velocity');

**VISUALIZE**
- r = rateControl(10);
- while(r.TotalElapsedTime < 20)
  - odomMsg = odomSub.LatestMessage;
  - % INSERT YOUR ALGORITHM CODE HERE
  - velMsg.angular.z = ctrlOut;
  - send(velPub, velMsg);
  - plot(r.TotalElapsedTime, ctrlOut)
  - waitfor(r)
end
Simulink ↔ ROS Workflows

**SETUP** when model loads or runs

- **SENSE**
  - Extract data from ROS subscribers, if new data available

- **PROCESS**
  - Run perception and control algorithms

- **CONTROL**
  - Package control outputs into ROS messages and publish them

**Slow Rate**

**Fast Rate**

16
Algorithm Development Example
Traffic Sign Recognition and Collision Avoidance

ROS as Communication Framework

State Machine
Object Classifier
Obstacle Avoidance

Algorithm Development Example
Traffic Sign Recognition and Collision Avoidance

ROS as Communication Framework

State Machine
Object Classifier
Obstacle Avoidance
Algorithm Development Example
Traffic Sign Recognition and Collision Avoidance

Gazebo Co-simulation
Video Display
Application Example

- Sign-following Robot
  - Detect the color of the sign and send the velocity commands to turn the robot
  - Connect with ROS-enabled simulator, i.e., Gazebo
  - And connect with hardware
ROS Node Generation

- Can automatically generate, transfer, build, and run ROS nodes from Simulink
- Deploy algorithms as standalone C/C++ ROS nodes
- Once a ROS node is generated, you can:
  - Start and stop node from MATLAB
  - Use External mode to access data and tune parameters from Simulink
  - Use ROS to communicate with node
Application Examples

- Automated Valet parking using ROS 2
  - Distribute automated parking Valet application among various nodes in a ROS 2 network
  - Deploy as standalone ROS 2 nodes to speed up a simulation
User Story - Voyage
Develops longitudinal controls for self-driving taxis

Challenge
Develop a controller for a self-driving car to follow a target velocity and maintain a safe distance from obstacles

Solution
Use Simulink to design a longitudinal model predictive controller and tuned parameters based on experimental data imported into MATLAB using Robotics System Toolbox. Deploy the controller as a ROS node using ROS Toolbox. Generate source code using Simulink Coder into a Docker Container.

Results
- Development speed tripled
- Easy integration with open-source software
- Simulink algorithms delivered as production software

“We were searching for a prototyping solution that was fast for development and robust for production. We decided to go with Simulink for controller development and code generation, while using MATLAB to automate development tasks.”
- Alan Mond, Voyage
Concluding Remarks

Challenges in autonomous system development

- Applying Multi-domain Expertise
- Features to design complex Algorithms
- End-to-End workflows
- Evaluate robot performance and operation

Develop Software with Model-Based Design

Bridge ROS with MATLAB and Simulink

- Data Analysis
- Desktop Simulation
- ROS Node Deployment

End-to-End workflows
Key Takeaway of This Talk

• MATLAB and Simulink capabilities to prototype new algorithms through the ROS interface

• With ROS interface from MATLAB and Simulink users can connect to a live ROS network to access ROS messages

• Robot algorithms can be verified on desktop simulation and by connecting to external robot simulators

• Code generation tools automatically generate ROS nodes and deploy to simulated or physical hardware

• MATLAB and Simulink provide additional design tools, such as Technical computing tools, Simulation tools, Control design from low-level to supervisory logic, Algorithm design, and MBD.