MATLAB EXPO 2016
Robotics Development Workflow with MATLAB and Simulink

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

• Introduction
• Advanced Robotics Systems
• Robotics Development Workflow with MATLAB and Simulink
• Takeaways
Car as an Advanced Robotics System

PLANNING

Localization

Global Map

Motion control

Obstacle avoidance

PERCEIVE

SENSE

LIDAR
Camera
RADAR
GPS/IMU

Motion Controllers
Actuator ECUs
Steering Actuator
GAS/Brake Actuator

CONTROL

MATLAB EXPO 2016
Collaborative Robot as an Advanced Robotics System

Motion Planning → Optimization

Obstacle Avoidance → Object Detection

World

Torque → Camera → RGB-D → Temperature

SENSE

Motion Controllers

Actuator Control → Impedance Control → Safety Guards

PERCEIVE

PLAN

CONTROL
Architecture - Advanced Robotics System

Motion Planning → Optimization
Obstacle Avoidance → Object Detection

World

Middleware
Data synchronization
CPU/GPU Cluster, Server, Cloud

Real-time requirements
Microcontroller, FPGA, Real-time PC
Technology Solutions for Autonomous Systems

Control System Tbx

Computer Vision

Phased Array

HW Support Packages

Robots System Tbx

WLAN System Toolbox

Simscape Toolboxes

Data Acquisition Tbx

Simulink Real-Time

Communications Tbx

Stateflow

Statistics & Machine Learning

Robotics System Tbx

ROS
Success Stories

MATLAB EXPO 2016
Technology Solutions for Autonomous Systems

- Control System Tbx
- Simscape Toolboxes
- Simulink Real-Time
- HW Support Packages
- Phased Array
- Data Acquisition Tbx
- Statistics & Machine Learning
- Robotics System Tbx
- Communications Tbx
- WLAN System Toolbox
- Stateflow
- ROOS
Robotics System Toolbox

Environment for **prototyping**, **simulating**, and **deploying** robotics applications
Robotics Applications with Robotics System Toolbox

MATLAB EXPO 2016
Workflow Convergence is Needed

- Trace to requirements
- Design/debug algorithm
- System integration

- Testing on physical system
- Manual Coding
- Verification and Validation
Robotics Development Workflow with MATLAB and Simulink

RAPID ITERATIVE PROCESS

- Idea
- Requirements
  - Built-in algorithms and apps
  - System-Level Simulation MBD
  - Co-simulation
- Design and Simulation
  - C/C++ automatic code generation
  - Processor-in-the-loop (PIL)
  - Debug C/C++ with MATLAB Engine
- Prototype
- Implementation
  - Design independent of hardware implementation!
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

Sensor Communication Layer

Application Layer

Actuator Communication Layer

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

RAPID ITERATIVE PROCESS

- Import logged data
- Train a classifier
- Test component
Importing Simulation and Experimental Data

Experimental data or simulation Data

Import ROS Data

Filter your logged field data by topic or time interval

```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
imgMsg = select(carData, 'Topic','/camera/rgb/image_raw');
```

Data ready to design algorithms

Robotics System Toolbox™
% 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(k,:));
Visualize, Analyze, and Process Data: Classifier

Training data → Preprocessing → Feature Extraction → Training → Classifier

Input → Output

Computer Vision System Toolbox™ Statistics and Machine Learning Toolbox™
Design and Test Algorithm

```matlab
%% Create classifier object
sd = SignDetector();

%% Test algorithm with two stops signs
Iin = imread('gazeboTwoStopsTest.png');

%% Show algorithm result
[Iout,~,~,~,~] = sd.step(Iin);
imshow(Iout);
```
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 components
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
rosim '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
Robotics Development Workflow with MATLAB and ROS

RAPID ITERATIVE PROCESS

Requirements → Design and Simulation → Prototype → Implementation

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
Part1: Object Detection and Position Estimation
Part3: System Integration
Advanced Robotics Application Requires Multiple Technologies

MATLAB and Simulink: very powerful tools to design advanced robotics applications
Trajectory Planning with RGB-D Sensor
System Level Design

Pick and Place Controller

Motion Planner

Object Detector

State Controller
Robotics Development Workflow with MATLAB and Simulink

- **Visualize**
- **Experiment**
- **Prototype**
- **Simulate**
- **Deploy**

**Middleware**

ROS
Questions

% Thank you