MathWorks Toolchain for Low-Velocity Maneuvering Development at General Motors

Alon Davidi
Jonathan Naor
The last hundred meters of an autonomous drive entail unique challenges: there is variability in direction (reverse, three-point turns, etc.) there are often no road markings, no GPS signal, and no map. This is where a Self-Driving Car is at its most autonomous.
The goal of the Low Velocity Maneuvering (LVM) team at General Motors is to drive the vehicle in GPS-denied environments with high accuracy, to enable a variety of autonomous features.
In this talk we review our LVM development cycle, emphasizing MathWorks tools utilization, starting from the architecture management, models base design development, requirements & testing coverage, MIL, SIL, HIL simulations and code generation for multiple platforms.
Development Process
Motivation

• Fast development cycle: fast transition from proof of concept to production-mature software
• Utilize the best of all worlds for simulation, development, validation and deployment:
  • CarSim for precise vehicle dynamics
  • Unreal-based fisheye photo-realistic image rendering
  • ROS2 for visualization and debugging
  • dSpace for real-time validation
  • Simulink for Model-based design
Easy System and Architecture management – Model-based design approach
Simulink/CarSim co-simulation (SIL)

In-house full vehicle system simulation
RT HIL/Vehicle Validation – co-simulation (Simulink, Dspace & CarSim / Vehicle)
Unit-test process flowchart using Test Manager for requirements coverage analysis

Model and Code validation – “white box”
Model Validation “Black box” (Monte-Carlo)

System-level module in the loop verification using Simulation Manager
Adding vision perception to system simulation

Photorealism
Requirements: Fisheye Image Generation
Requirements: Simulated Weather Effects
Requirements:
Ground Truth
Requirements: Recording and Visualization
Photorealism (Unreal Engine Interface)

Interface with Unreal Engine enables state of the art photorealistic images in a variety of scenarios
Fisheye Image Generation

Unique ability to generate fisheye images from Unreal (configurable intrinsics and extrinsics)
Ground Truth

Generate route in MATLAB / Simulink and execute in Unreal
Broadcast sensor data and ground truth in standard ROS2 messages; utilize debugging and visualization tools
Wrap up

Using MATLAB/Simulink we were able to quickly:

• Manage
  • Architecture management using System Composer

• Develop
  • Control System
  • Vision Perception System

• Validate
  • Requirement-based and back-to-back testing using Simulink Test

• Deploy
  • Code generation

Next Steps:

• Unify control and vision development platforms
• Move to Adaptive AutoSAR architecture
MathWorks and GM Collaborative Effort

Thank you!