MATLAB EXPO 2019

Automated Driving with MATLAB and Simulink

GianCarlo Pacitti
Senior Application Engineer, MathWorks
Capabilities of an Autonomous Vehicle
Capabilities of an Autonomous Vehicle
Capabilities of an Autonomous Vehicle
Capabilities of an Autonomous Vehicle
Some common questions from automated driving engineers

How can I synthesize scenarios to test my designs?

How can I discover and design in multiple domains?

How can I integrate with other environments?

Simulation Integration

- Perception
- Planning
- Control

ROS
C/C++
Python
Cross Release
Third Party
CAN
### How can I design with virtual driving scenarios?

<table>
<thead>
<tr>
<th>Scenes</th>
<th>Cuboid</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Diagram of Cuboid Scene" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Testing</th>
<th>Controls, sensor fusion, planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring</td>
<td>Driving Scenario Designer App Programmatic API (drivingScenario)</td>
</tr>
<tr>
<td>Sensing</td>
<td>Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list)</td>
</tr>
</tbody>
</table>
How can I design with virtual driving scenarios?

<table>
<thead>
<tr>
<th>Scenes</th>
<th>Cuboid</th>
<th>3D Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>Controls, sensor fusion, planning</td>
<td>Controls, sensor fusion, planning, perception</td>
</tr>
<tr>
<td>Authoring</td>
<td>Driving Scenario Designer App Programmatic API (drivingScenario)</td>
<td>Unreal Engine Editor</td>
</tr>
<tr>
<td>Sensing</td>
<td>Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list)</td>
<td>Probabilistic radar (detection list) Monocular camera (image, labels, depth) Fisheye camera (image) Lidar (point cloud)</td>
</tr>
</tbody>
</table>
Simulate controls with perception

**Lane-Following Control with Monocular Camera Perception**
- Author target vehicle trajectories
- Synthesize monocular camera and probabilistic radar sensors
- Model lane following and spacing control in Simulink
- Model lane boundary and vehicle detectors in MATLAB code

*Model Predictive Control Toolbox™*
*Automated Driving Toolbox™*
*Vehicle Dynamics Blockset™*

*Updated R2019b*

Visit the Demo Station to see more…
Visualize logged simulation detection and camera data

Lane-Following Control with Monocular Camera Perception
- Author target vehicle trajectories
- Synthesize monocular camera and probabilistic radar sensors
- Model lane following and spacing control in Simulink
- Model lane boundary and vehicle detectors in MATLAB code

Model Predictive Control Toolbox™
Automated Driving Toolbox™
Vehicle Dynamics Blockset™
Updated R2019b
How can I design with virtual driving scenarios?

<table>
<thead>
<tr>
<th>Scenes</th>
<th><strong>Cuboid</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>Controls, sensor fusion, planning</td>
</tr>
<tr>
<td>Authoring</td>
<td>Driving Scenario Designer App Programmatic API (drivingScenario)</td>
</tr>
<tr>
<td>Sensing</td>
<td>Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list)</td>
</tr>
</tbody>
</table>

| 3D Simulation |
| Controls, sensor fusion, planning, perception |
| Unreal Engine Editor |
| Probabilistic radar (detection list) Monocular camera (image, labels, depth) Fisheye camera (image) Lidar (point cloud) |
Synthesize driving scenarios to test sensor fusion algorithms

Sensor Fusion Using Synthetic Radar and Vision Data
- Create scenario
- Add probabilistic radar and vision sensors
- Create tracker
- Visualize coverage area, detections, and tracks

Automated Driving Toolbox™

Vehicle passes through detector coverage areas
Graphically author driving scenarios

**Driving Scenario Designer**
- Create roads and lane markings
- Add actors and trajectories
- Specify actor size and radar cross-section (RCS)
- Explore pre-built scenarios
- Import OpenDRIVE roads

**Automated Driving Toolbox™**

R2018a
Programmatically author driving scenarios

```matlab
scenario = drivingScenario;

road( scenario, [0 0; 10 0; 53 -20], ... 
    'lanes', lanespec(2) );

plot( scenario,'Waypoints','on' );

idleCar = vehicle( scenario, ... 
    'Position',[25 -5.5 0], ... 
    'Yaw',-22 );

passingCar = vehicle( scenario, 'ClassID', 1 );

waypoints = [1 -1.5; 16.36 -2.5; 17.35 -2.765; ... 
    23.83 -2.01; 24.9 -2.4; 50.5 -16.7];

velocity = 15;

trajectory( passingCar, waypoints, velocity );
```
Synthesize driving scenarios from recorded data

**Scenario Generation from Recorded Vehicle Data**
- Visualize video
- Import OpenDRIVE roads
- Import GPS
- Import object lists

*Automated Driving Toolbox™*

**R2019a**
Enhancements to driving scenarios

**Create Driving Scenario Variations Programmatically**

- Export the scenario code to MATLAB® and generate scenario variations programmatically
- Export the scenario and sensors to Simulink® and use them to test your driving algorithms.

*Automated Driving Toolbox™ R2019b*
Integrate driving scenario into closed loop simulation

**Lane Following Control with Sensor Fusion**
- Integrate scenario into system
- Design lateral (lane keeping) and longitudinal (lane spacing) model predictive controllers
- Visualize sensors and tracks
- Generate C/C++ code
- Test with software in the loop (SIL) simulation

*Model Predictive Control Toolbox™*
*Automated Driving Toolbox™*
*Embedded Coder®*
Design lateral and longitudinal controls

Lane Following Control with Sensor Fusion
- Integrate scenario into system
- Design lateral (lane keeping) and longitudinal (lane spacing) model predictive controllers
- Visualize sensors and tracks
- Generate C/C++ code
- Test with software in the loop (SIL) simulation

*Model Predictive Control Toolbox™*
*Automated Driving Toolbox™*
*Embedded Coder®*
Automate testing against driving scenarios

Testing a Lane Following Controller with Simulink Test
- Specify driving scenario

Simulink Test™
Automated Driving Toolbox™
Model Predictive Control Toolbox™

R2018b
### How can I design with virtual driving scenarios?

<table>
<thead>
<tr>
<th>Scenes</th>
<th>Cuboid</th>
<th>3D Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>Controls, sensor fusion, planning</td>
<td>Controls, sensor fusion, planning, perception</td>
</tr>
<tr>
<td>Authoring</td>
<td>Driving Scenario Designer App Programmatic API (drivingScenario)</td>
<td>Unreal Engine Editor</td>
</tr>
<tr>
<td>Sensing</td>
<td>Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list)</td>
<td>Probabilistic radar (detection list) Monocular camera (image, labels, depth) Fisheye camera (image) Lidar (point cloud)</td>
</tr>
</tbody>
</table>
Select from prebuilt 3D simulation scenes

**3D Simulation for Automated Driving**
- Straight road
- Curved road
- Parking lot
- Double lane change
- Open surface
- US city block
- US highway
- Virtual Mcity

*Automated Driving Toolbox™ R2019b*
Customize 3D simulation scenes

Support Package for Customizing Scenes

- Install Unreal Engine
- Set up environment and open Unreal Editor
- Configure configuration Block for Unreal Editor co-simulation
- Use Unreal Editor to customize scenes

Vehicle Dynamics Blockset™ R2019b
Model sensors in 3D simulation environment

3D Simulation for Automated Driving
- Monocular camera
- Fisheye camera
- Lidar
- Probabilistic radar

Automated Driving Toolbox™
Synthesize monocular camera sensor data

Visualize Depth and Semantic Segmentation Data in 3D Environment
- Synthesize RGB image
- Synthesize depth map
- Synthesize semantic segmentation

Automated Driving Toolbox™
Synthesize fisheye camera sensor data

Simulate a Simple Driving Scenario and Sensor in 3D Environment

- Scaramuzza camera model
  - parameters for distortion center, image size and mapping coefficients

Automated Driving Toolbox™
Synthesize lidar sensor data

Simulate Lidar Sensor Perception Algorithm
- Record and visualize
- Develop algorithm
- Build a 3D map
- Use algorithm within simulation environment

Automated Driving Toolbox™
Synthesize radar sensor data

Simulate Radar Sensors in 3D Environment

- Extract the center locations
- Use center location for road creation using driving scenario
- Define multiple moving vehicles
- Export trajectories from app
- Configure multiple probabilistic radar models
- Calculate confirmed track

Automated Driving Toolbox™

R2019b
Communicate with the 3D simulation environment

**Send and Receive Double-Lane Change Scene Data**

- Simulation 3D Message Set
  - Send data to Unreal Engine
  - Traffic light color

- Simulation 3D Message Get
  - Retrieve data from Unreal Engine
  - Number of cones hit

*Vehicle Dynamics Blockset™*

*R2019b*
New Examples for 3D Simulation in Automated Driving Toolbox

**Unreal Engine Driving Scenario Simulation**

Select waypoints for 3D Simulation

Select waypoints from a scene and visualize the path of a vehicle following these waypoints in a 3D simulation environment.

Open Script

**Design of Lane Marker Detector in 3D Simulation Environment**

Use a 3D simulation environment to record synthetic sensor data and develop and test a lane marker detection system.

Open Script

**Visualize Automated Parking Valet Using 3D Simulation**

Visualize vehicle motion in a 3D simulation environment using an automated parking valet system constructed in Simulink.

Open Script

**Simulate Lidar Sensor Perception Algorithm**

Develop a lidar perception algorithm using data recorded from a 3D simulation environment, and simulate within that environment.

Open Script

**Simulate Radar Sensors in 3D Environment**

Implement a synthetic data simulation for tracking and sensor fusion using Simulink and a 3D simulation environment.

Open Model

**Simulate a Simple Driving Scenario and Sensor in 3D Environment**

Learn the basics of configuring and simulating scenes, vehicles, and sensors in a 3D environment powered by the Unreal Engine from

Open Model

**Visualize Depth and Semantic Segmentation Data in 3D Environment**

Visualize depth and semantic segmentation data captured from a camera sensor in a 3D simulation environment.

Open Model
Simulating automated driving systems with MATLAB and Simulink

Simulation Environment

Perception

Planning

Control
Simulating automated driving systems with MATLAB and Simulink

Simulation Environment

- Scenarios
  - Scenery
- Actions & Events
  - Actors
- Goals & Metrics
  - Sensors

Perception
Planning
Control
Integrate components and model scenarios

Monocular camera lane detector

Lane following controller

Actions & Events

Actors

Scenery

Sensors

Goals & Metrics
Specify equivalent 3D Simulation scenery

Scenery:
- Equivalent straight and curved roads in Simulation 3D
Scene Configuration and Scenario Reader

Supported Scenery:
✓ Straight road
✓ Curved road segment
✓ Curved road (not exposed in example, but available)
Specify 3D Simulation actor trajectories

Action & Events:
- Scenario Reader describes trajectories of target vehicles
- Target trajectories are converted to world coordinates
Specify 3D Simulation vehicles

Actors:
- Target vehicle positions specified from Scenario Reader block

Actors:
- Ego vehicle position specified based on vehicle dynamics
Synthesize scenarios to test your design

**Lane Following Control with Sensor Fusion**
*Model Predictive Control Toolbox™*
*Automated Driving Toolbox™*
*Embedded Coder®*

---

**Design of Lane Marker Detector in 3D Simulation Environment**

Use a 3D simulation environment to record synthetic sensor data and develop and test a lane marker detection system.

---

**Lane-Following Control with Monocular Camera Perception**
*Model Predictive Control Toolbox™*
*Automated Driving Toolbox™*
*Vehicle Dynamics Blockset™*

---

*Updated R2019b*
Some common questions from automated driving engineers

How can I synthesize scenarios to test my designs?

How can I discover and design in multiple domains?

How can I integrate with other environments?

Control
Planning
Perception

Simulation Integration
ROS
CAN
C/C++
Python
Cross Release
Third Party
Design camera, lidar, and radar perception algorithms

- Detect vehicle with camera
- Detect ground with lidar
- Detect pedestrian with radar

**Object Detection Using YOLO v2 Deep Learning**
*Computer Vision Toolbox™*
*Deep Learning Toolbox™*

**Segment Ground Points from Organized Lidar Data**
*Computer Vision Toolbox™*

**Introduction to Micro-Doppler Effects**
*Phased Array System Toolbox™*

**Versions:**
- R2019a
- R2018b
- R2019a
Interoperate with neural network frameworks

Interoperability with various deep learning frameworks:
- PyTorch
- Caffe2
- MXNet
- Core ML
- CNTK
- Keras-Tensorflow
- MATLAB

Open Neural Network Exchange

Visit the Demo Stations to see more…
Simulate lane detection and lane following system with MATLAB and Simulink

Simulation Environment

- Scenarios
  - Scenery
- Actions & Events
  - Actors
- Goals & Metrics
  - Sensors

Monocular camera lane detector

Lane following controller
Monocular camera lane detector
- Based on shipping example
- Lane rejection and tracking added to improve performance
Track Vehicles Using Lidar: From Point Cloud to Track List

- Design 3-D bounding box detector
- Design tracker (target state and measurement models)
- Generate C/C++ code for detector and tracker

Sensor Fusion and Tracking Toolbox™

Computer Vision Toolbox™
Design tracker for lidar point cloud data

Track Vehicles Using Lidar: From Point Cloud to Track List
- Design 3-D bounding box detector
- Design tracker (target state and measurement models)
- Generate C/C++ code for detector and tracker

Sensor Fusion and Tracking Toolbox™
Computer Vision Toolbox™

Visualize detections and tracks
Design trackers

Multi-Object Tracker

Association & Track Management

Tracking Filter

Tracks

Detections

- Multi-object tracker
- Linear, extended, and unscented Kalman filters

From various sensors at various update rates
Design trackers

Multi-Object Tracker

- Association & Track Management
- Tracking Filter

From various sensors at various update rates

- Multi-object tracker
- Global Nearest Neighbor (GNN) tracker
- Joint Probabilistic Data Association (JPDA) tracker
- Track-Oriented Multi-Hypothesis Tracker (TOMHT)
- Probability Hypothesis Density (PHD) tracker

- Linear, extended, and unscented Kalman filters
- Particle, Gaussian-sum, IMM filters

Automated Driving Toolbox™
Sensor Fusion and Tracking Toolbox™
Evaluate error metrics

**Extended Object Tracking**
- Design multi-object tracker
- Design extended object trackers
- Evaluate tracking metrics
- Evaluate error metrics
- Evaluate desktop execution time

**Sensor Fusion and Tracking Toolbox™**

**Automated Driving Toolbox™**

*Updated R2019a*
Compare relative execution times of object trackers

**Extended Object Tracking**
- Design multi-object tracker
- Design extended object trackers
- Evaluate tracking performance
- Evaluate error metrics
- Evaluate desktop execution time

**Sensor Fusion and Tracking Toolbox™**

**Automated Driving Toolbox™**

Updated R2019a

- Multi-object tracker
- Probability Hypothesis Density tracker
- Extended object (size and orientation) tracker
Design track level fusion systems

Vehicle 1

Detections -> Multi-Object Tracker -> Tracks

Vehicle 2

Detections -> Multi-Object Tracker -> Tracks
Design track level fusion systems
Design track level fusion systems

Vehicle 1

Detections → Multi-Object Tracker → Track Fusion → Tracks

Vehicle 2

Detections → Multi-Object Tracker → Track Fusion → Tracks
Track-level fusion

Track-to-Track Fusion for Automotive Safety Applications

Parked vehicles observed by vehicle 1
Pedestrian observed by vehicle 1

Occluded vehicle fused from vehicle 1
Occluded pedestrian fused from vehicle 1
Pedestrian observed by vehicle 1

Rumor control: the fused track is dropped by vehicle 1 because vehicle 2 is coasting and there is no update by vehicle 1 sensors

Sensor Fusion and Tracking Toolbox™
Automated Driving Toolbox™
For more on Sensor Fusion and Tracking…

Visit Marc Willerton’s presentation later this afternoon

<table>
<thead>
<tr>
<th>Time</th>
<th>Technical Computing</th>
<th>Model-Based Design</th>
<th>Getting Started with MATLAB and Simulink</th>
<th>Master Classes</th>
<th>Innovation Auditorium</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:15</td>
<td>Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:45</td>
<td>Developing Smart IoT Sensors Using the MathWorks Toolchain Samuel Bailey, Skyred Consulting</td>
<td>Synchronous Machine Modelling Using Simscape Peeni Reni, Cummins Generator Technologies</td>
<td>Sensor Fusion and Tracking for Autonomous Systems Marc Willerton, MathWorks</td>
<td>Simplifying Requirements-Based Verification with Model-Based Design Fraser Macmillen, MathWorks</td>
<td>Predictive Maintenance with MATLAB Phil Rother, MathWorks</td>
</tr>
<tr>
<td>16:15</td>
<td>Industrial IoT and Digital Twins Coorous Mohtadi, MathWorks</td>
<td>Developing Fit-For-Purpose Simscape Models to Support System and Control Design Rick Hyde, MathWorks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:00</td>
<td>End of Day</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Some common questions from automated driving engineers

How can I synthesize scenarios to test my designs?

How can I discover and design in multiple domains?

How can I integrate with other environments?
Read road and speed attributes from HERE HD Live Map data

**Use HERE HD Live Map Data to Verify Lane Configurations**
- Load camera and GPS data
- Retrieve speed limit
- Retrieve lane configurations
- Visualize composite data

**Automated Driving Toolbox™**

R2019a
Read lane attributes from HERE HD Live Map data

**Use HERE HD Live Map Data to Verify Lane Configurations**
- Load camera and GPS data
- Retrieve speed limit
- Retrieve lane configurations
- Visualize composite data

*Automated Driving Toolbox™*

R2019a
Visualize HERE HD Live Map recorded data

**Use HERE HD Live Map Data to Verify Lane Configurations**
- Load camera and GPS data
- Retrieve speed limit
- Retrieve lane configurations
- Visualize composite data

**Automated Driving Toolbox™ R2019a**
Design path planner

**Automated Parking Valet**

- Create cost map of environment
- Inflate cost map for collision checking
- Specify goal poses
- Plan path using rapidly exploring random tree (RRT*)

**Automated Driving Toolbox™**

R2018a
Design path planner and controller

**Automated Parking Valet with Simulink**
- Integrate path planner
- Design lateral controller (based on vehicle kinematics)
- Design longitudinal controller (PID)
- Simulate closed loop with vehicle dynamics

**Visualize Automated Parking Valet Using 3D Simulation**

**Automated Driving Toolbox™**

R2018b  R2019b
Some common questions from automated driving engineers

How can I synthesize scenarios to test my designs?

How can I discover and design in multiple domains?

How can I integrate with other environments?

Simulation Integration

Perception

Planning

Control

ROS

CAN

C/C++

Python

Cross Release

Third Party

Cross Release
Train reinforcement learning networks for ADAS controllers

Train Deep Deterministic Policy Gradient (DDPG) Agent for Adaptive Cruise Control
- Create environment interface
- Create agent
- Train agent
- Simulate trained agent

Reinforcement Learning Toolbox™

Visit the Demo Stations to see more…
Simulate lane detection and lane following system with MATLAB and Simulink

Simulation Environment

- Scenarios
  - Scenery
- Actions & Events
  - Actors
- Goals & Metrics
  - Sensors
- Monocular camera lane detector
- Lane following controller
Lane Following Controller Algorithm
Components of lane following with spacing control algorithm

- Path following control block
  - Estimate lane center
  - Estimate lead vehicle
Goal

- Maintain the driver-set velocity and keep a safe distance from lead vehicle.
- Keep the ego vehicle in the middle of the lane.
- Slow down the ego vehicle when road is curvy.
Model predictive control (MPC)
MPC for Lane Following Control

minimize:
\[ w_1 |V_{ego} - V_{set}|^2 + w_2 |E_{lateral}|^2 \]

References
• Ego velocity set point \( (V_{set}) \)
• Target lateral deviation (=0)

Measured disturbances
• MIO velocity \( (V_{mio}) \)
• Previewed road curvature \( (\rho) \)

subject to:
\[
\begin{align*}
D_{relative} & \geq D_{safe} \\
a_{min} & \leq a \leq a_{max} \\
\delta_{min} & \leq \delta \leq \delta_{max}
\end{align*}
\]

Measured outputs
• Relative distance \( (D_{relative}) \)
• Ego velocity \( (V_{ego}) \)
• Lateral deviation \( (E_{lateral}) \)
• Relative yaw angle \( (E_{yaw}) \)

Manipulated variables
• Acceleration \( (a) \)
• Steering angle \( (\delta) \)

Ego Vehicle

Optimizer

Ego Vehicle Model

MPC controller

Look ahead
Act early!
Components

- Estimate lane center
  
  - Inputs to MPC: For lateral control
    - Four cases are considered:
      1) Both left and right lanes are detected
      2) Left lane is detected
      3) Right lane is detected
      4) No lane is detected
  
  - Estimate MIO (lead vehicle)
    
    - Inputs to MPC: For longitudinal control
  
  - MPC: Path following controller
Path Following Control Block

**Driver setting**
- Set velocity
- Time gap
- Relative distance
- Relative velocity
- Longitudinal velocity
- Curvature
- Lateral deviation
- Relative yaw angle

**Measurement**
- Longitudinal acceleration
- Steering angle

**Virtual lane**
For lane change

**Bicycle model parameters**

**Model Predictive Control Toolbox™**

**Delay or latency in the system**

**Disable distance keeping**
Path Following Control Block

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum steering angle</td>
<td>-0.26</td>
<td>Minimum steering angle (rad)</td>
</tr>
<tr>
<td>Maximum steering angle</td>
<td>0.26</td>
<td>Maximum steering angle (rad)</td>
</tr>
<tr>
<td>Minimum longitudinal acceleration</td>
<td>-3</td>
<td>Minimum longitudinal acceleration (m/s²)</td>
</tr>
<tr>
<td>Maximum longitudinal acceleration</td>
<td>2</td>
<td>Maximum longitudinal acceleration (m/s²)</td>
</tr>
</tbody>
</table>

Tune MPC performance:

- Sample time (s): 0.1
- Prediction horizon (steps): 30
- Control horizon (steps): 3
- Controller Behavior:
  - Weight on velocity tracking: 0.1
  - Weight on change of longitudinal acceleration: 0.1
  - Weight on lateral error: 1
  - Weight on change of steering angle: 0.1
Path Following Control Block

- Set velocity
- Time gap
- Relative distance
- Relative velocity
- Longitudinal velocity
- Curvature
- Lateral deviation
- Relative yaw angle

Change MPC design

Description:
Keep the ego vehicle traveling along the center of a straight or curved road, track a set velocity and maintain a safe distance from a lead vehicle by adjusting the longitudinal acceleration and the front steering angle of the ego vehicle.

Parameters:
- Optimization
  - Use suboptimal solution
  - Maximum iteration number: 10
- Data Type: double
- Optional Imports:
  - Use external signal to enable or disable optimization
  - Use external control signal for bumpless transfer between PFC and other controllers

Customization:
To customize your controller, generate an PFC subsystem from this block and modify it. The controller configuration data is exported as a structure in the MATLAB workspace.
Simulate controls with perception

**Lane-Following Control with Monocular Camera Perception**
- Author target vehicle trajectories
- Synthesize monocular camera and probabilistic radar sensors
- Model lane following and spacing control in Simulink
- Model lane boundary and vehicle detectors in MATLAB code

*Model Predictive Control Toolbox™*
*Automated Driving Toolbox™*
*Vehicle Dynamics Blockset™*

Updated R2019b
Design lateral and longitudinal Model Predictive Controllers

**Longitudinal Control**

Adaptive Cruise Control with Sensor Fusion
Automated Driving Toolbox™
Model Predictive Control Toolbox™
Embedded Coder®

**Lateral Control**

Lane Keeping Assist with Lane Detection
Automated Driving Toolbox™
Model Predictive Control Toolbox™
Embedded Coder®

**Longitudinal + Lateral**

Lane Following Control with Sensor Fusion and Lane Detection
Automated Driving Toolbox™
Model Predictive Control Toolbox™
Embedded Coder®

R2017b + R2018a = R2018b
Some common questions from automated driving engineers

How can I synthesize scenarios to test my designs?

How can I discover and design in new domains?

How can I integrate with other environments?
ROS Toolbox - NEW!

- Communicate with [ROS](#) and [ROS 2](#) nodes
- Multiplatform support
- Connect to live ROS data
- Replay logged data
- Generate standalone ROS nodes through code generation
Call C++, Python, and OpenCV from MATLAB

- **Call C++**
- **Call Python**
- **Call OpenCV & OpenCV GPU**

**Import C++ Library Functionality into MATLAB**
MATLAB®

**Call Python from MATLAB**
MATLAB®

**Install and Use Computer Vision Toolbox OpenCV Interface**

- **Computer Vision System Toolbox™**
- **OpenCV Interface Support Package**

**Updated** R2018b

- R2019a
- R2014a
Call C code from Simulink

Call C code

Create buses from C structs

Test and verify C code

Bring Custom Image Filter Algorithms as Reusable Blocks in Simulink
Simulink®

Import Structure and Enumerated Types
Simulink®

Custom C Code Verification with Simulink Test
Simulink Test™
Simulink Coverage™
Cross-release simulation through code generation

Integrate Generated Code by Using Cross-Release Workflow

- Generate code from previous release (R2010a or later)
- Import generated code as a block in current release
- Tune parameters
- Access internal signals

Embedded Coder

R2016a
Connect to third party tools

152 Interfaces to 3\textsuperscript{rd} Party Modeling and Simulation Tools

(as of March 2019)
Some common questions from automated driving engineers

Synthesize scenarios to test my designs

Discover and design in multiple domains

Integrate with other environments

Perception

Planning

Control

Simulation Integration

ROS

CAN

C/C++

Python

Cross Release

Third Party
Thank You!