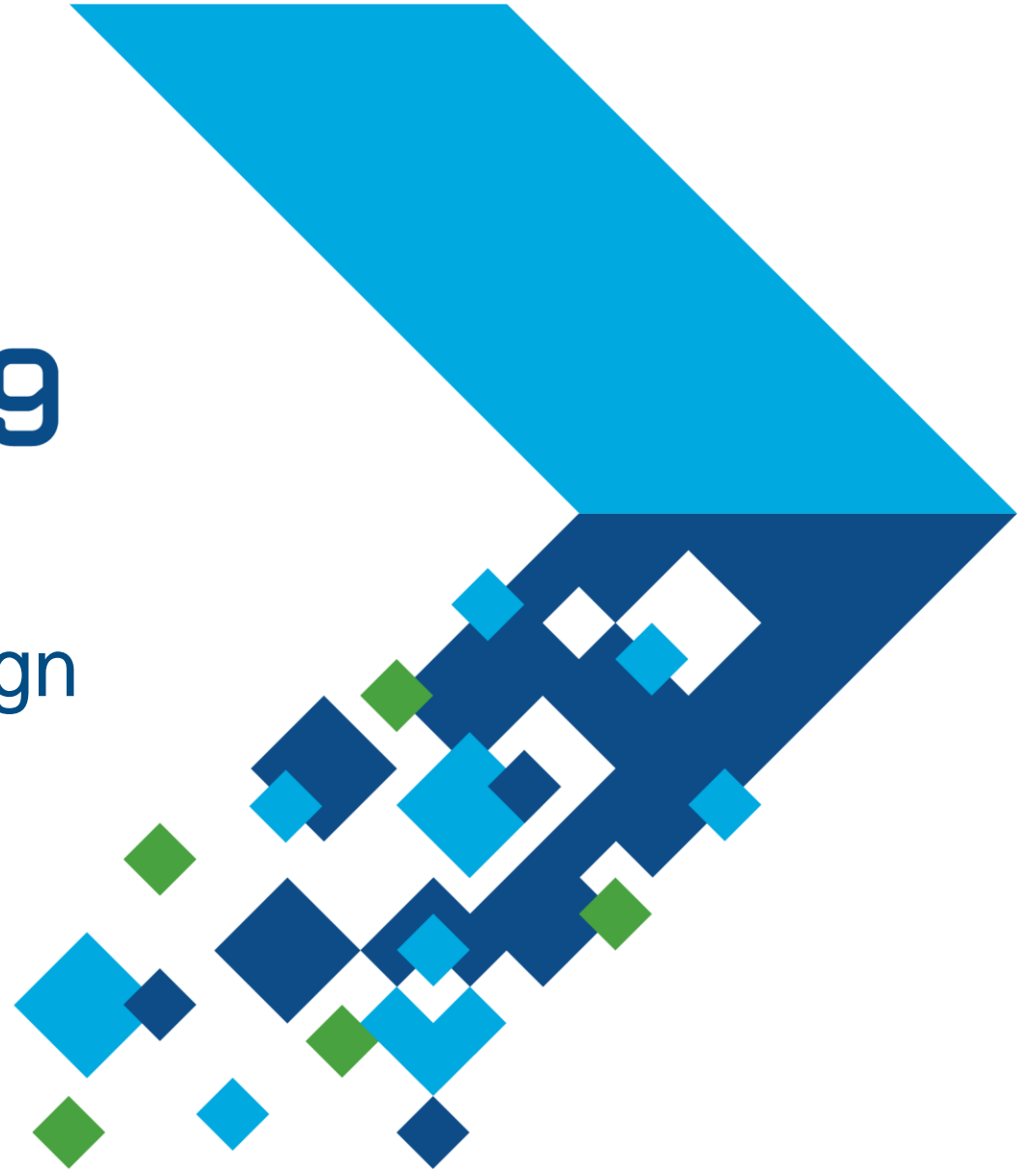


# MATLAB EXPO 2019

## Automated Driving System Design and Simulation

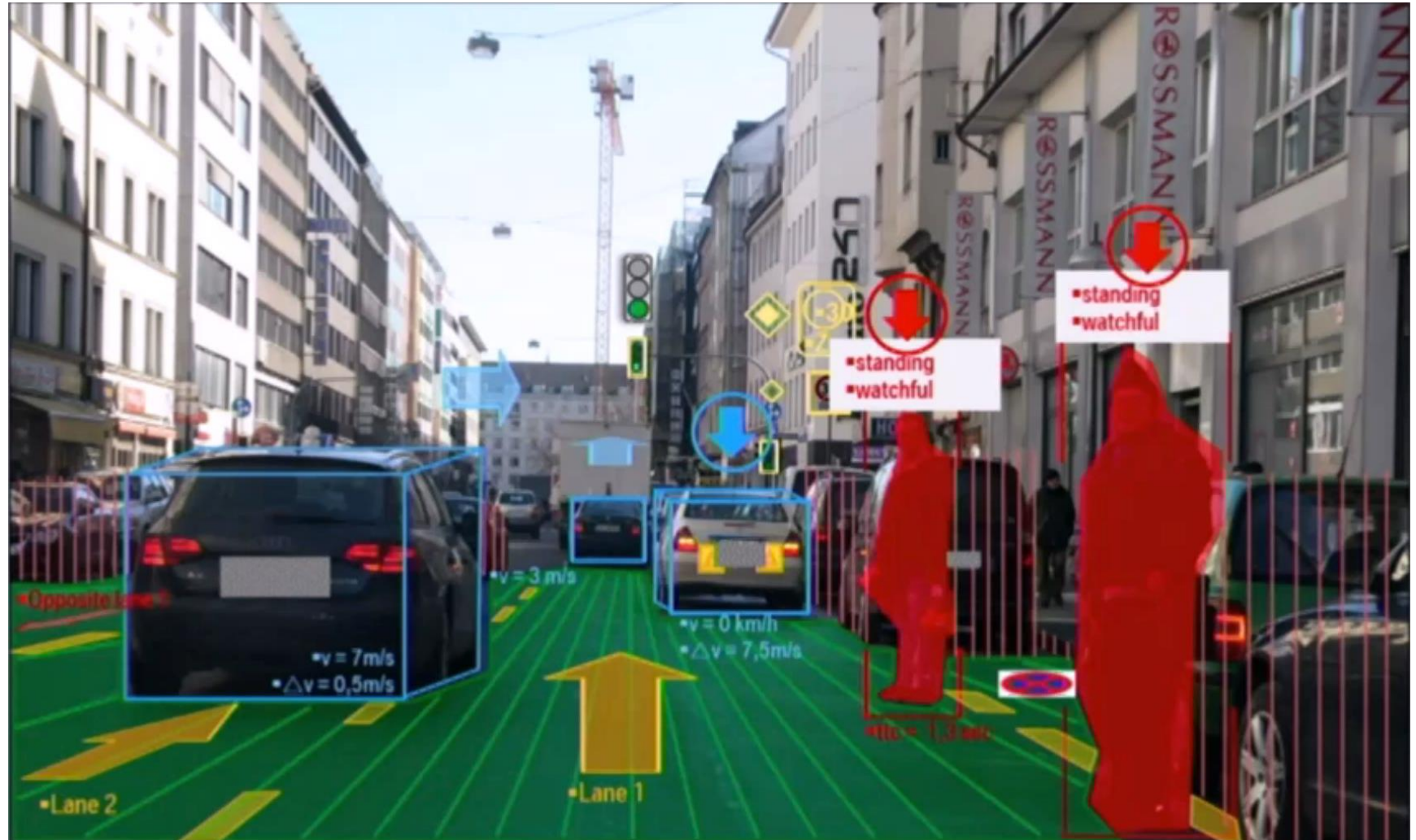
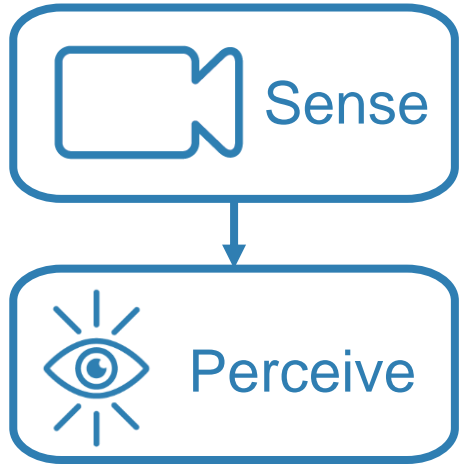
**Dr. Amod Anandkumar**  
*MathWorks India*



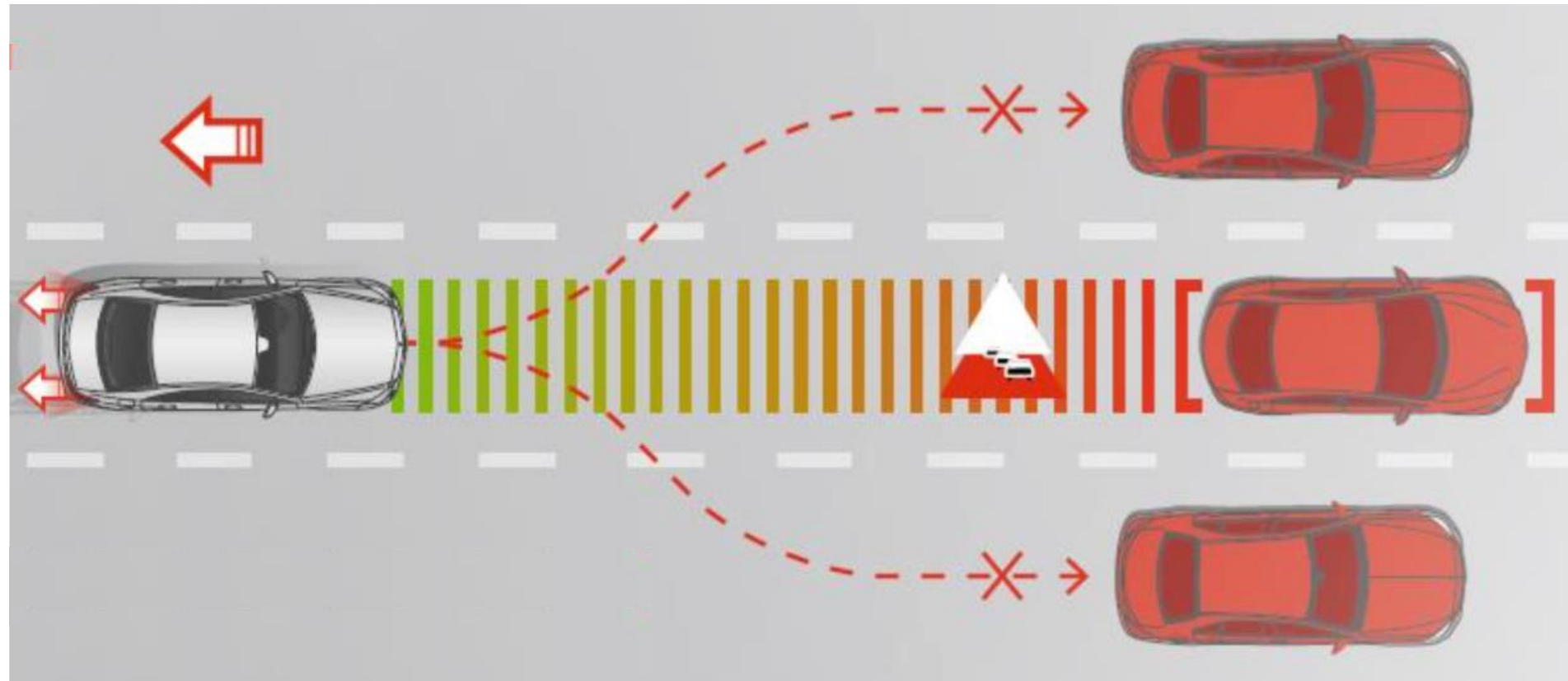
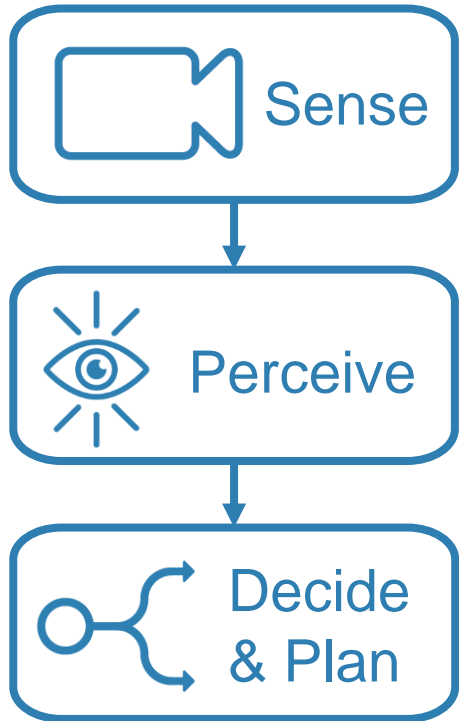
# Capabilities of an Autonomous Vehicle



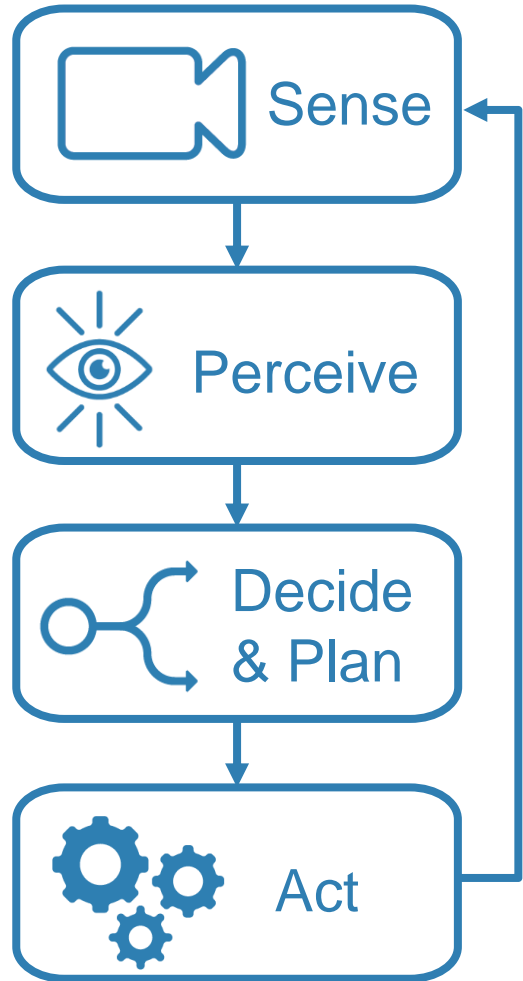
# Capabilities of an Autonomous Vehicle



# Capabilities of an Autonomous Vehicle

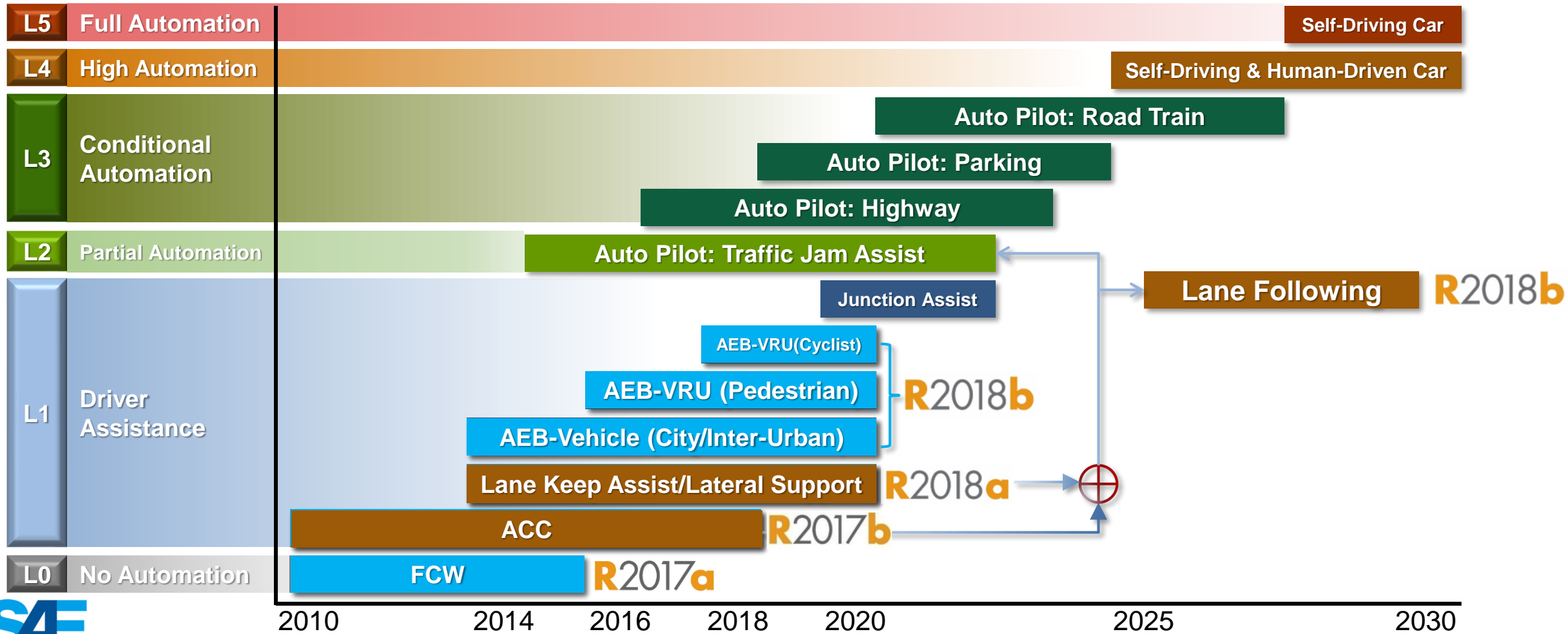


# Capabilities of an Autonomous Vehicle

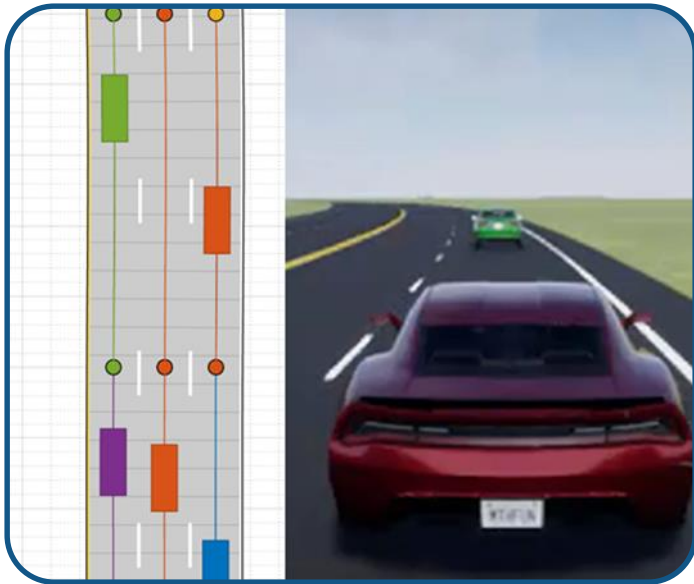


# Evolution of ADAS and Autonomous Driving Car Technologies

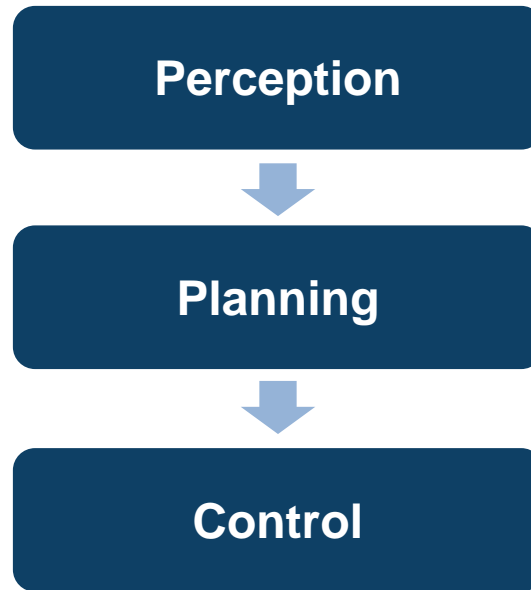
Reference examples using **Automated Driving Toolbox™**



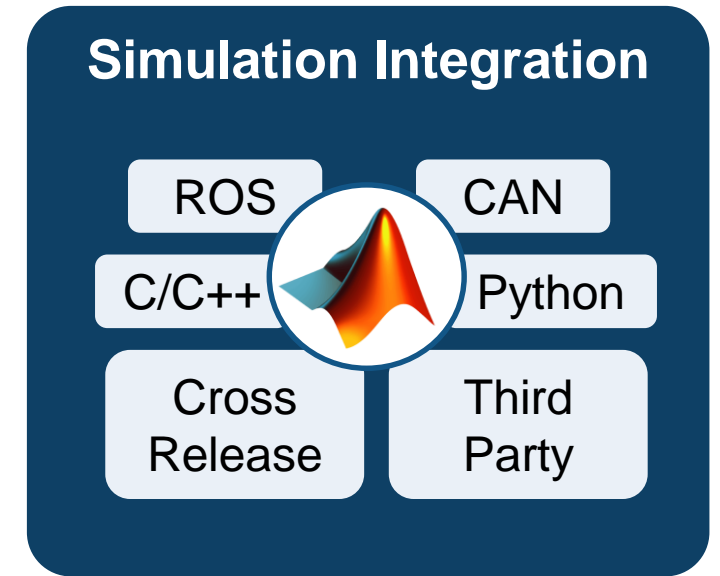
# Some common questions from automated driving engineers



How can I **synthesize scenarios** to test my designs?

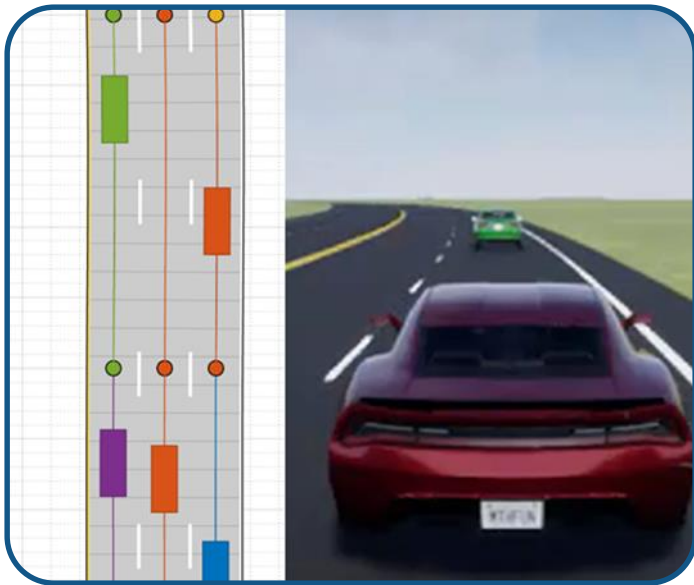


How can I **discover and design** in multiple domains?

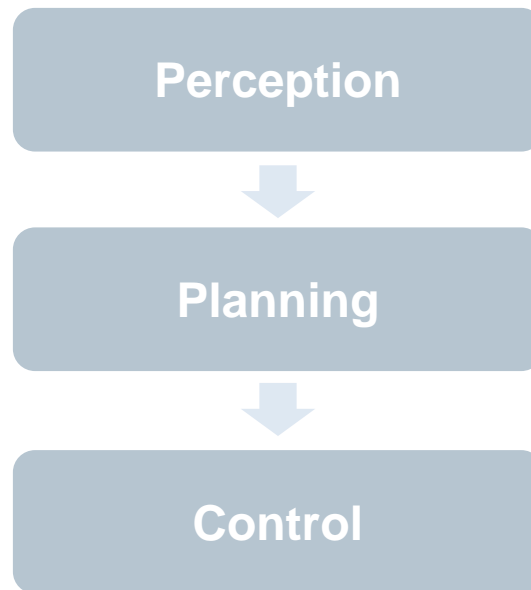


How can I **integrate** with other environments?

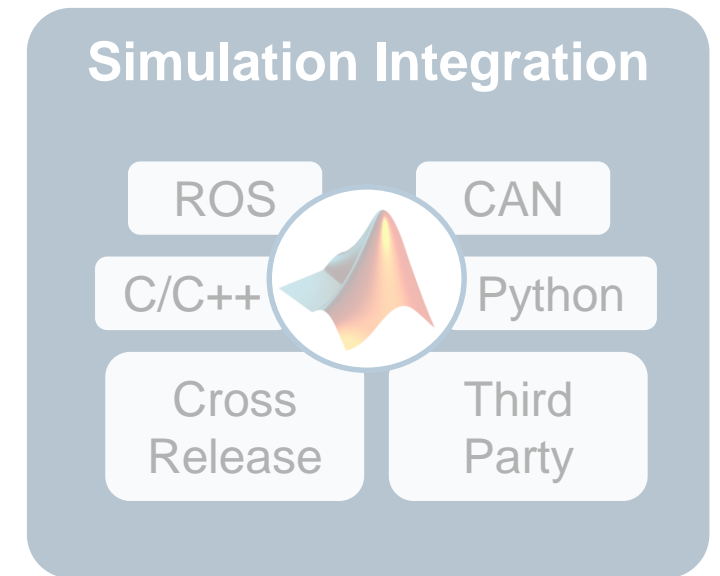
# Some common questions from automated driving engineers



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?



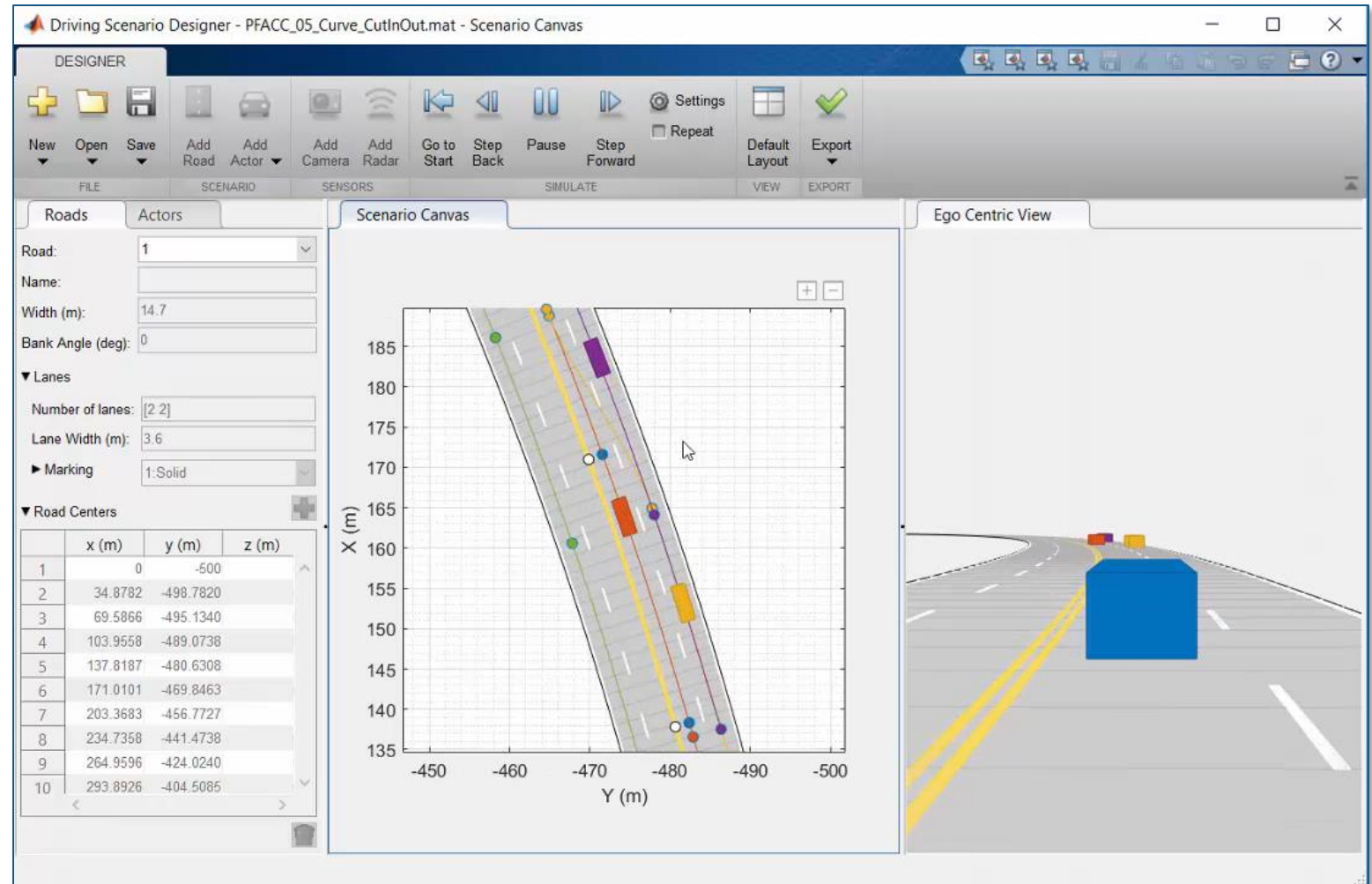
# Graphically author driving scenarios

## 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™*

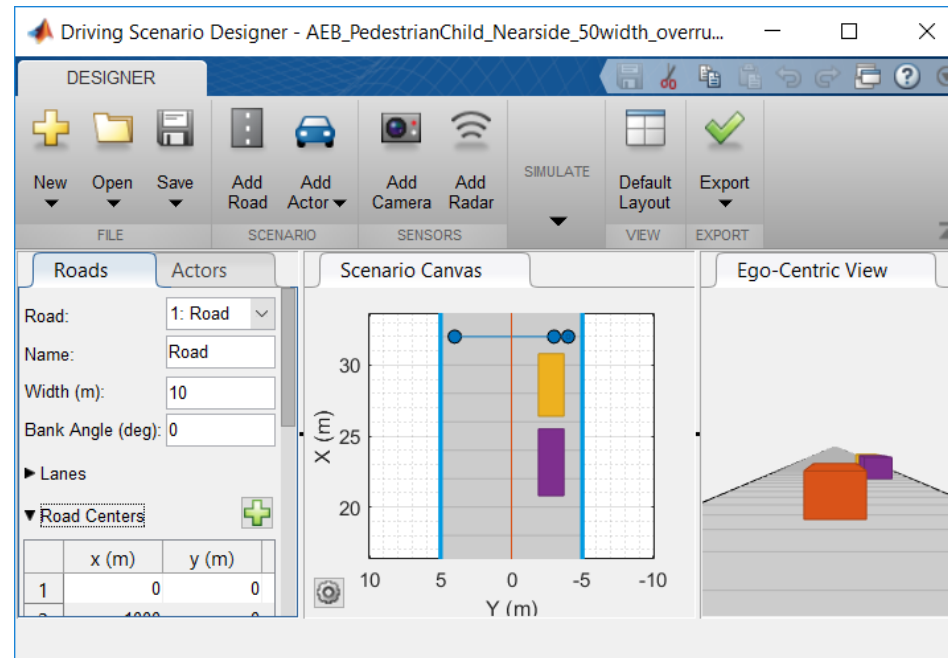
**R2018a**



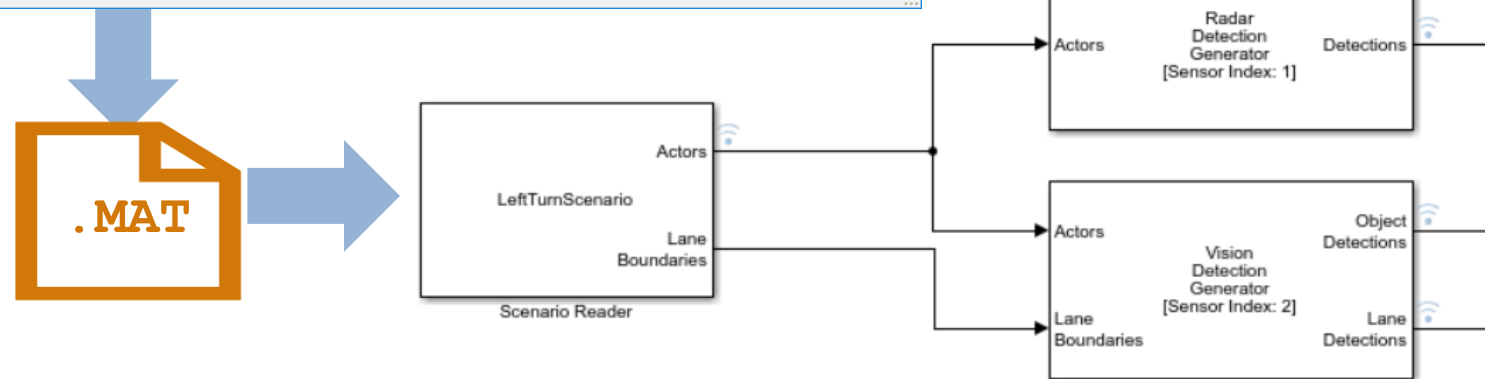
# Integrate driving scenarios into Simulink simulations

## Test Open-Loop ADAS Algorithm Using Driving Scenario

- Edit driving scenario
- Integrate into Simulink
- Add sensor models
- Visualize results
- Pace simulation



Automated Driving Toolbox™  
**R2019a**



# Simulate driving scenarios into closed loop simulations

## Automatic Emergency Braking (AEB) with Sensor Fusion

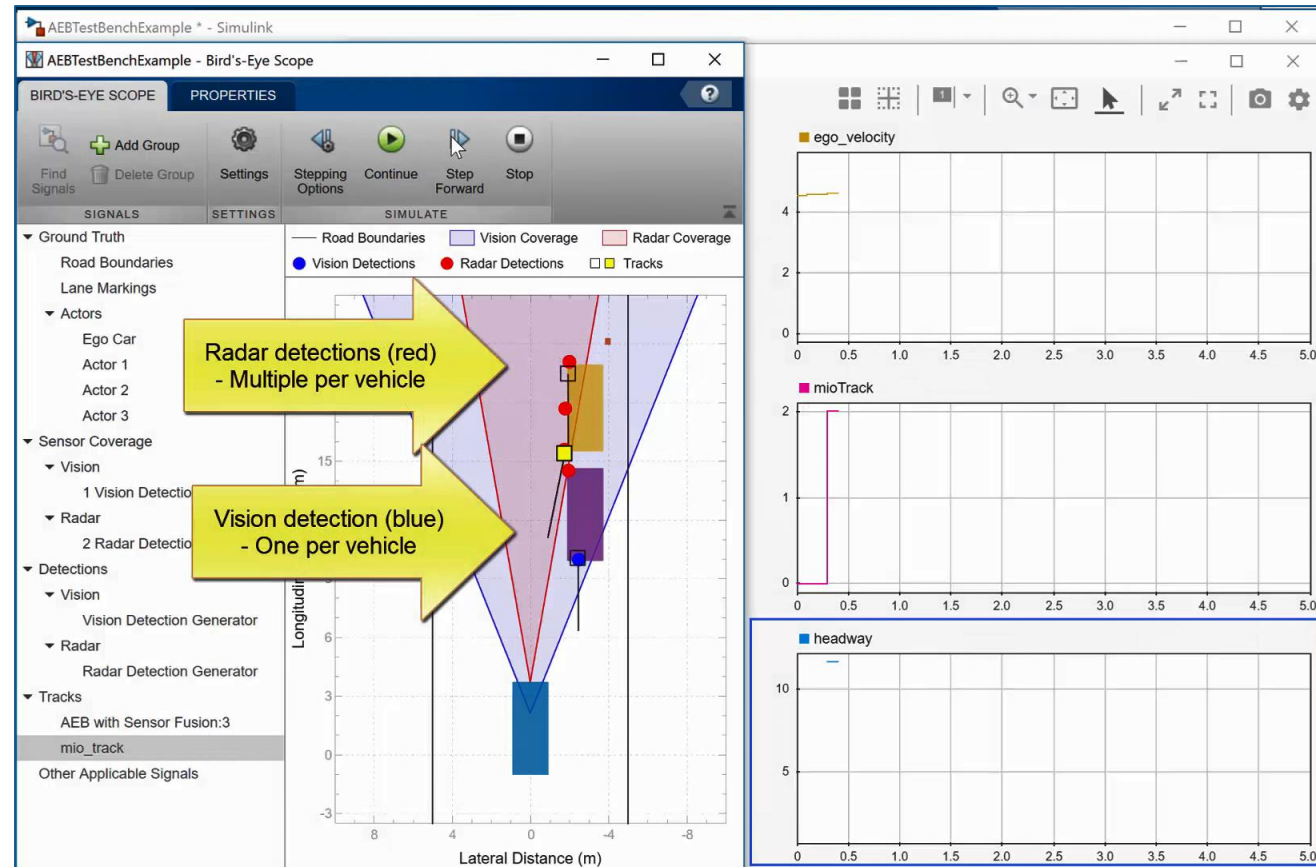
- Specify driving scenario
- Design AEB logic
- Integrate sensor fusion
- Simulate system
- Generate C/C++ code
- Test with software in the loop (SIL) simulation

Automated Driving Toolbox™

Stateflow®

Embedded Coder®

R2018b



Develop and Test Vehicle Controllers for ADAS and Automated Driving Applications Through System Simulation

15:00–15:30

# 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**

The screenshot shows the Simulink Test Manager interface. The 'TESTS' toolbar at the top has a red box around the 'Run' button. The 'Test Browser' on the left shows a tree view of 'LaneFollowingTestScenarios' with a blue box around the 'Scenarios' folder, which contains a list of test scenarios. The 'ACC\_ISO\_TargetDiscriminationTest' is selected. The right pane shows the configuration for this test, including a 'Requirements link' pointing to 'ACC\_ISO\_TargetDiscriminationTest (LaneFollowingTestRequirements#1)', a 'Simulink Model' field set to 'LaneFollowingTestBenchExample', and a 'CALLBACKS\*' section with a code block:
 

```
1 scenarioId = 1;
2 helperLFSetUp;
```

 A callout points to this code with the text 'Define scenario ID and data initialization'. At the bottom, the 'CLEANUP\*' section contains:
 

```
1 plotLFResults(sltest_simout.logout);
```

 A callout points to this line with the text 'Plot the results'. A property table at the bottom left shows details for the selected test:
 

PROPERTY	VALUE
Name	ACC_ISO_TargetDiscriminationTest
Type	Simulation Test
Model	LaneFollowingTestBenchExample
Simulation Mode	Normal
Location	C:\02_ADST2018b\Demos\...
Enabled	<input checked="" type="checkbox"/>
Hierarchy	LaneFollowingTestScenario...
Tags	Type comma or space separa...

# 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

The image displays the MATLAB R2019a interface. The main window shows a script titled "PlaybackScenarioExample.mlx" with the following content:

```
Summary
This example shows how to automatically generate a virtual driving scenario from vehicle data recorded using the GPS and lidar sensors.

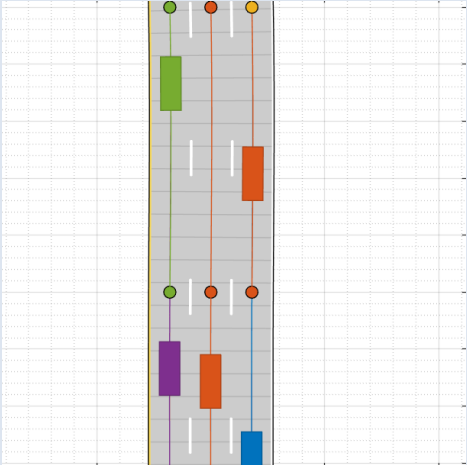
Helper Functions
helperGetEgoData
This function reads the ego vehicle data from a text file and converts into a structure.

108 function [egoData] = helperGetEgoData(egoFile)
109 %Read the ego vehicle data from text file
110 fileID = fopen(egoFile);
111 content = textscan(fileID, '%f %f %f');
112 fields = {'lat', 'lon', 'Time'};
113 egoData = cell2struct(content, fields, 2);
114 fclose(fileID);
115 end

helperGetNonEgoData
This function reads the processed lidar data from a text file and converts into a structure. The processed lidar data contains information about the
```

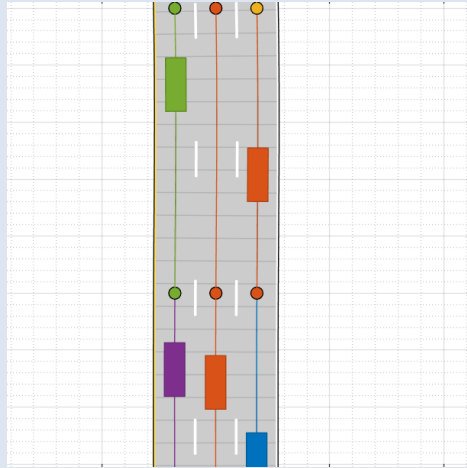
The right side of the interface features a yellow header "Simulate synthesized scenario" above a 3D visualization of a road scene. The scene includes a blue car, a green car, and a purple car on a road with lane markings. A 2D plot on the left shows the road layout with X (m) on the horizontal axis (ranging from 270 to 320) and Y (m) on the vertical axis (ranging from 80 to 120). The Command Window at the bottom shows "Busy".

# How can I design with virtual scenarios?

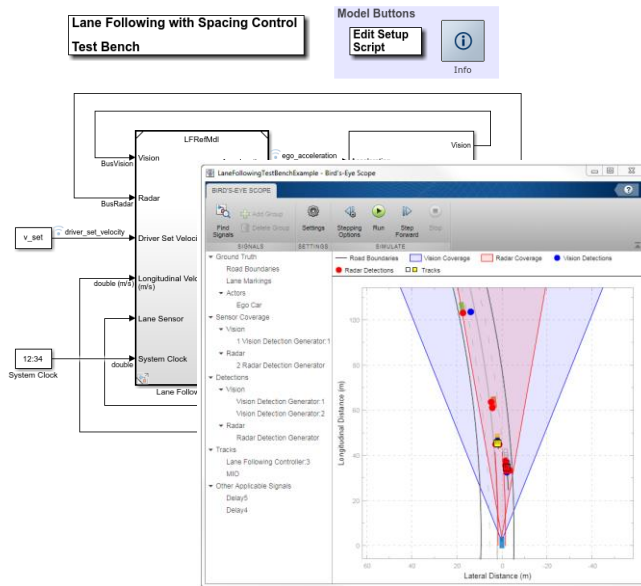
Scenes	<b>Driving Scenarios (cuboid)</b>  
Testing	Controls Controls + sensor fusion
Authoring	Driving Scenario Designer App drivingScenario programmatic API
Sensing	Probabilistic radar detections Probabilistic vision detections Probabilistic lane detections

# How can I design with virtual scenarios?

Scenes	Driving Scenarios (cuboid)	Unreal Engine
Testing	Controls Controls + sensor fusion	Controls Controls + vision
Authoring	Driving Scenario Designer App drivingScenario programmatic API	Unreal Editor
Sensing	Probabilistic radar detections Probabilistic vision detections Probabilistic lane detections	Ideal camera (viewer)

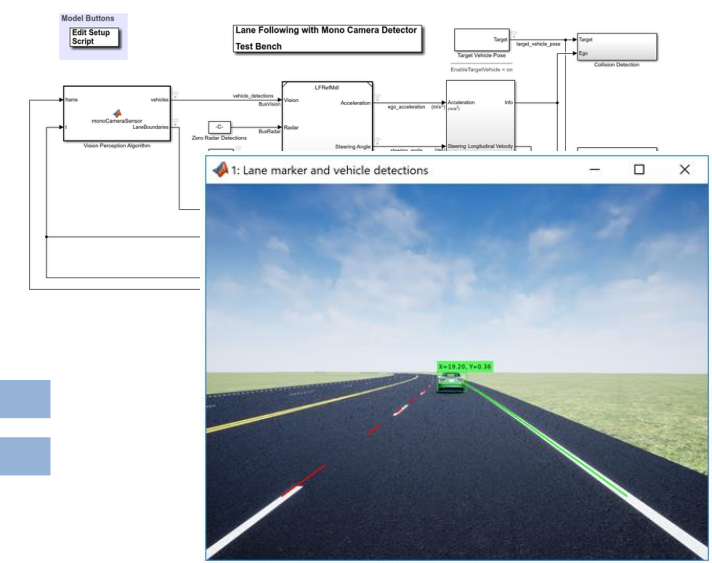


# Simulate controls and perception systems



```

40
41 classdef helperMonoSensor < handle
42
43 properties
44     % Sensitivity for the lane segmentation
45     LaneSegmentationSensitivity = 0.25;
46
47
48
49
50
51
  
```



## Lane Following Control with Sensor Fusion

*Model Predictive Control Toolbox™*  
*Automated Driving Toolbox™*  
*Embedded Coder®*

**R2018b**

## Visual Perception Using Monocular Camera

*Automated Driving Toolbox™*

**R2017a**

## Lane-Following Control with Monocular Camera Perception

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

**R2018b**



# Simulate lane controls with vision based perception

## Lane-Following Control with Monocular Camera Perception

- Integrate Simulink controller
  - Lane follower
  - Spacing control
- Integrate MATLAB perception
  - Lane boundary detector
  - Vehicle detector
- Synthesize ideal camera image from Unreal Engine

*Model Predictive Control Toolbox™*

*Automated Driving Toolbox™*

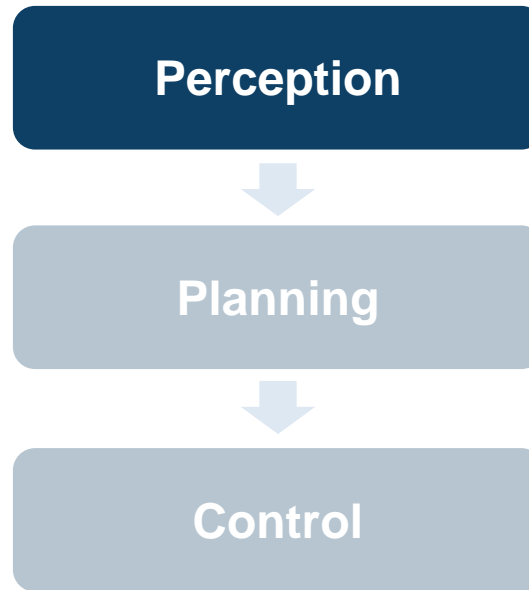
*Vehicle Dynamics Blockset™*



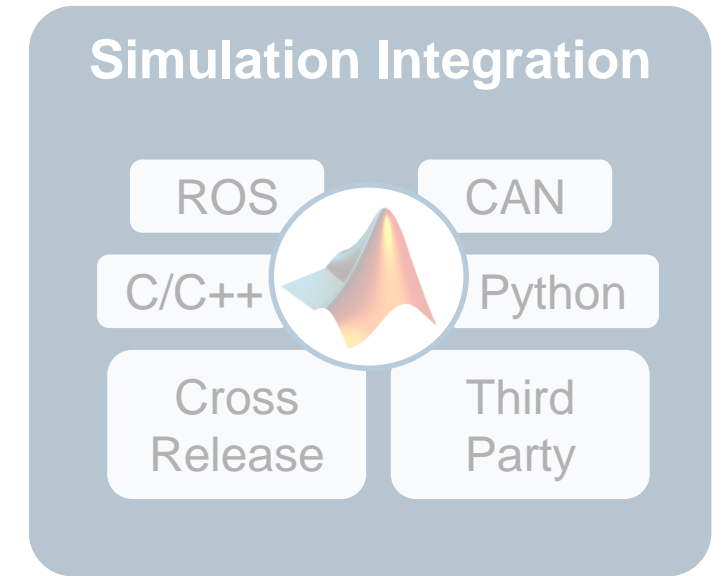
# Some common questions from automated driving engineers



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?

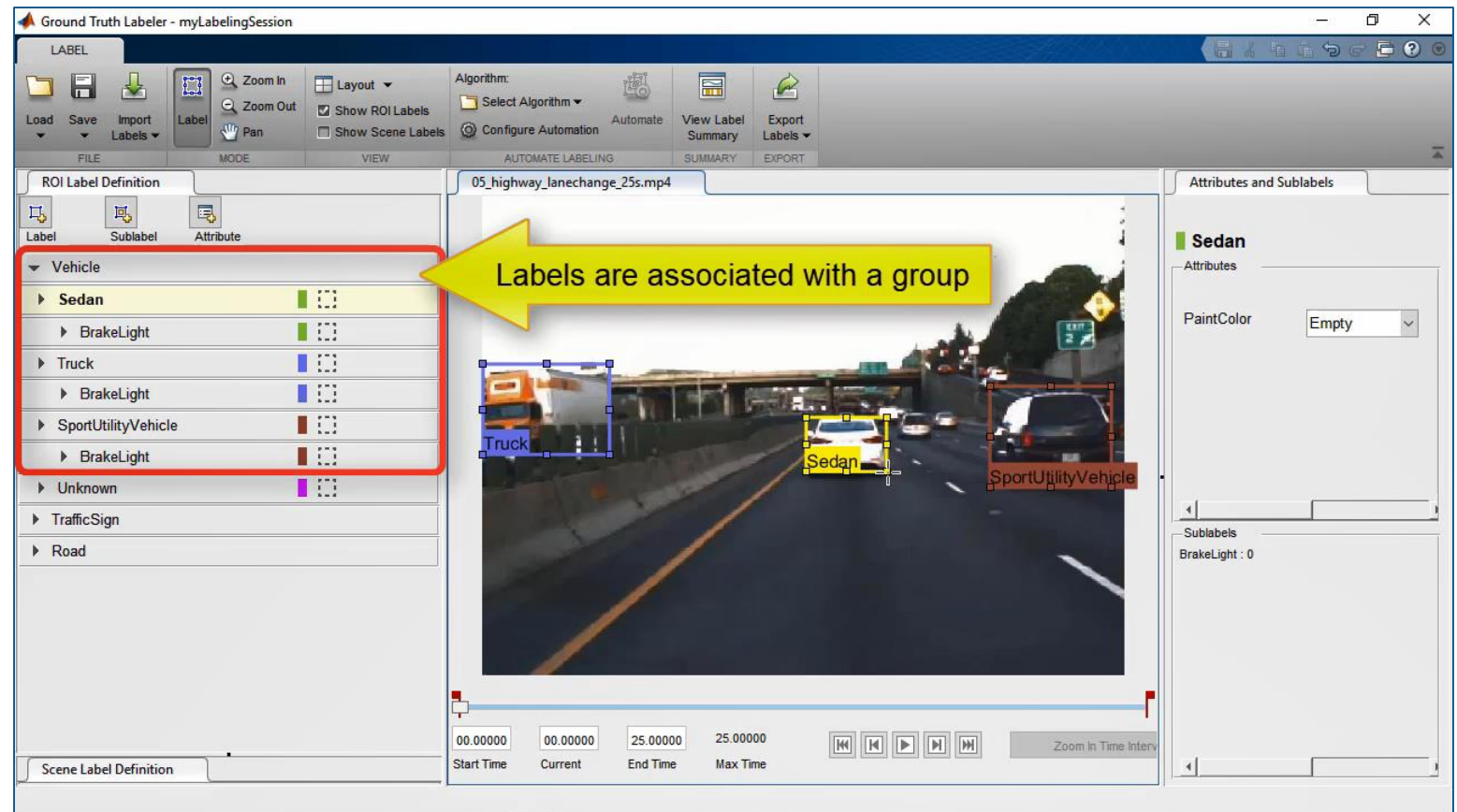
# Interactively label sensor data

## Get Started with the Ground Truth Labeler

- Label rectangles
- Label lane markings
- Label pixels
- Label scenes
- Create label groups
- Create sublabels
- Add label attributes

Automated Driving Toolbox™

Updated **R2019a**



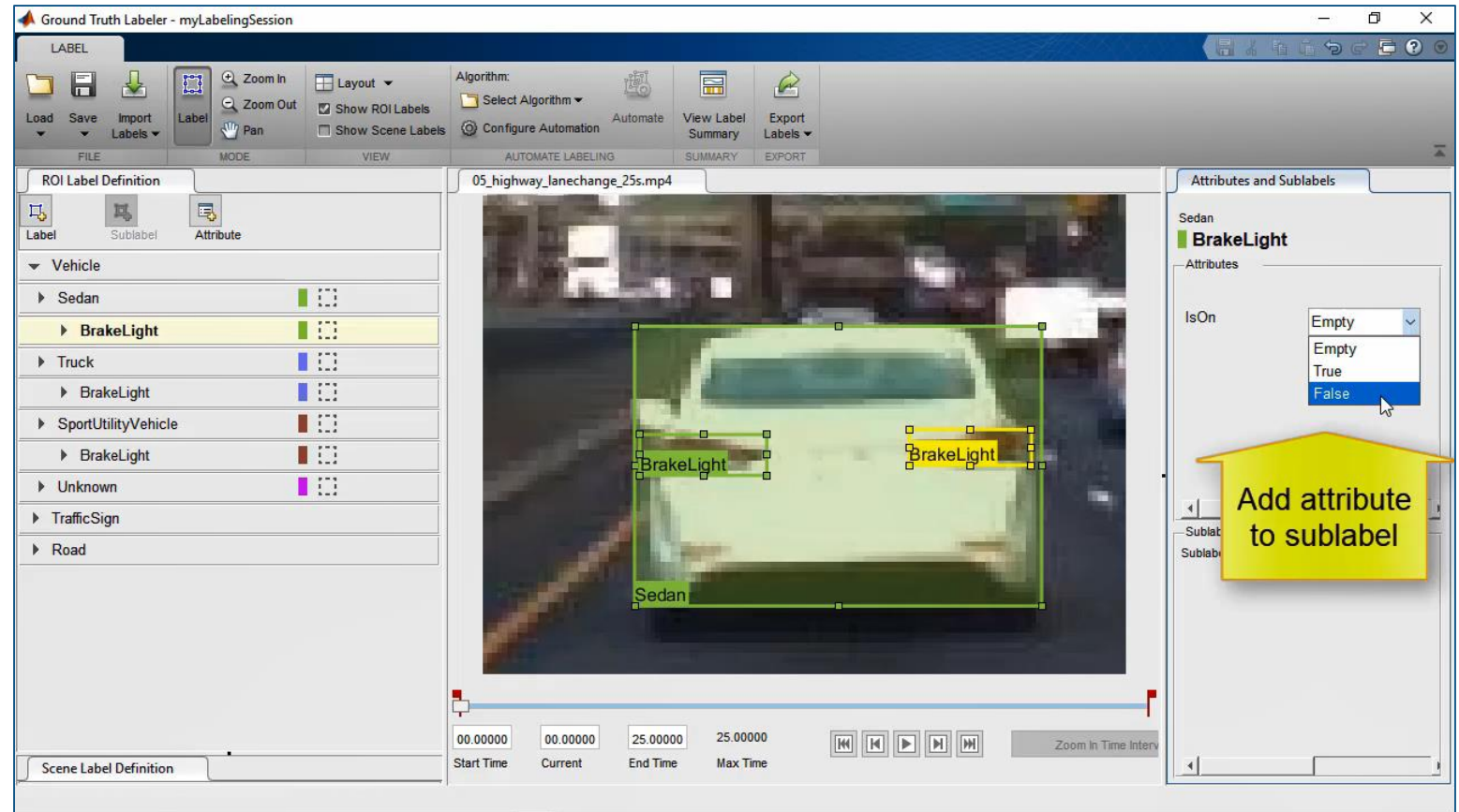
# Create sublabels and add attributes

## Get Started with the Ground Truth Labeler

- Label rectangles
- Label lane markings
- Label pixels
- Label scenes
- Create label groups
- Create sublabels
- Add label attributes

Automated Driving Toolbox™

Updated **R2019a**



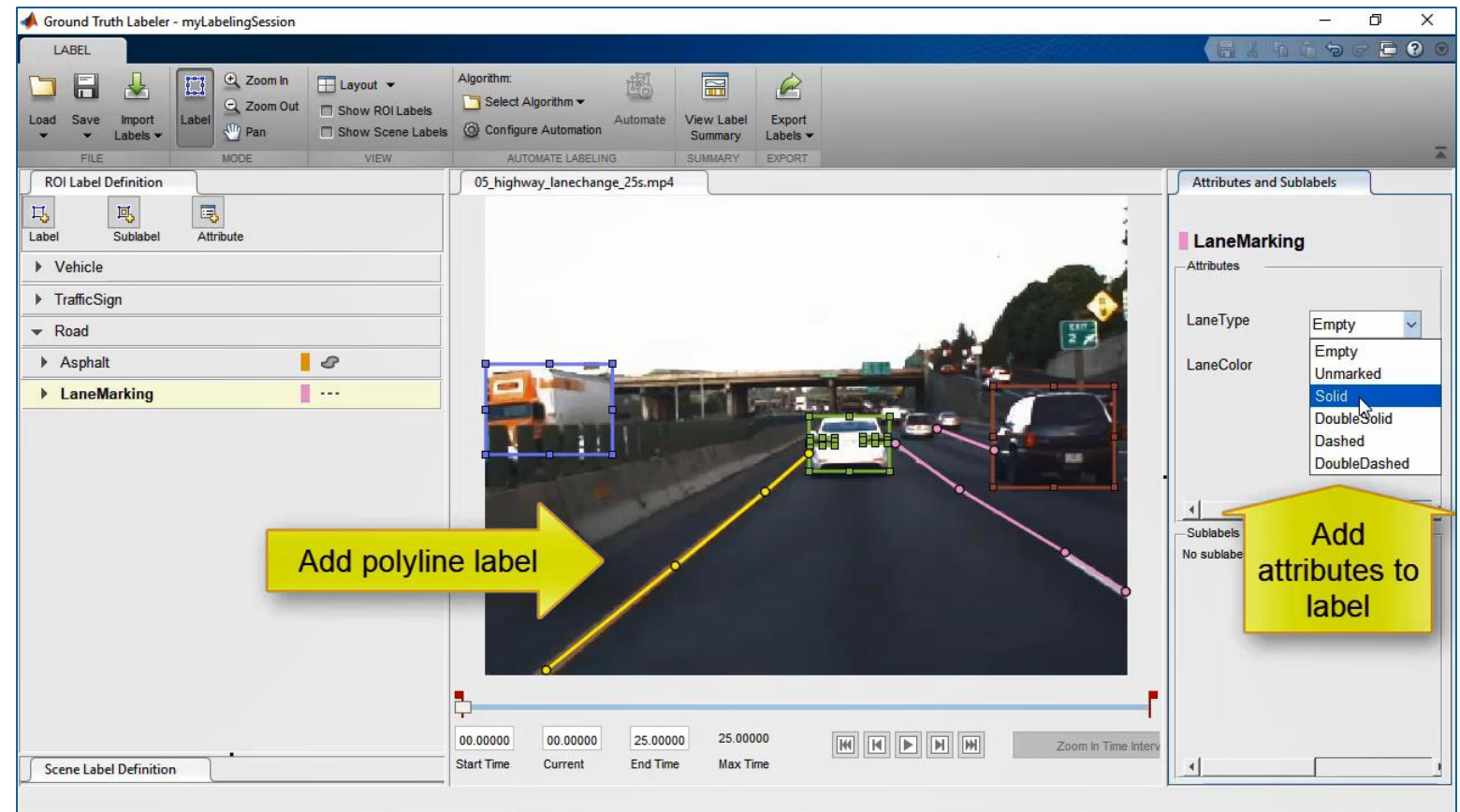
# Create polyline labels and add attributes

## Get Started with the Ground Truth Labeler

- Label rectangles
- Label lane markings
- Label pixels
- Label scenes
- Create label groups
- Create sublabels
- Add label attributes

Automated Driving Toolbox™

Updated **R2019a**



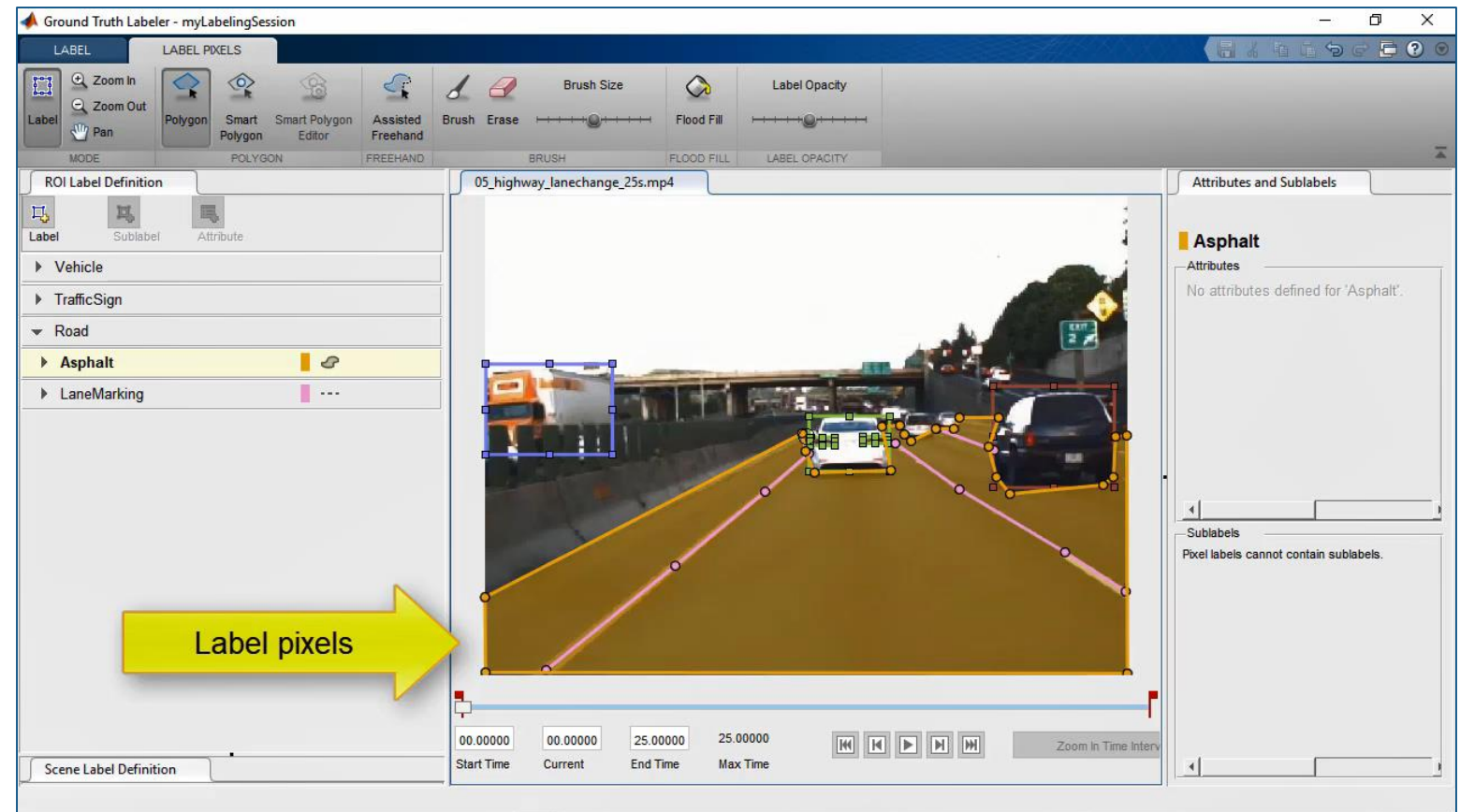
# Create pixel labels

## Get Started with the Ground Truth Labeler

- Label rectangles
- Label lane markings
- Label pixels
- Label scenes
- Create label groups
- Create sublabels
- Add label attributes

Automated Driving Toolbox™

Updated **R2019a**



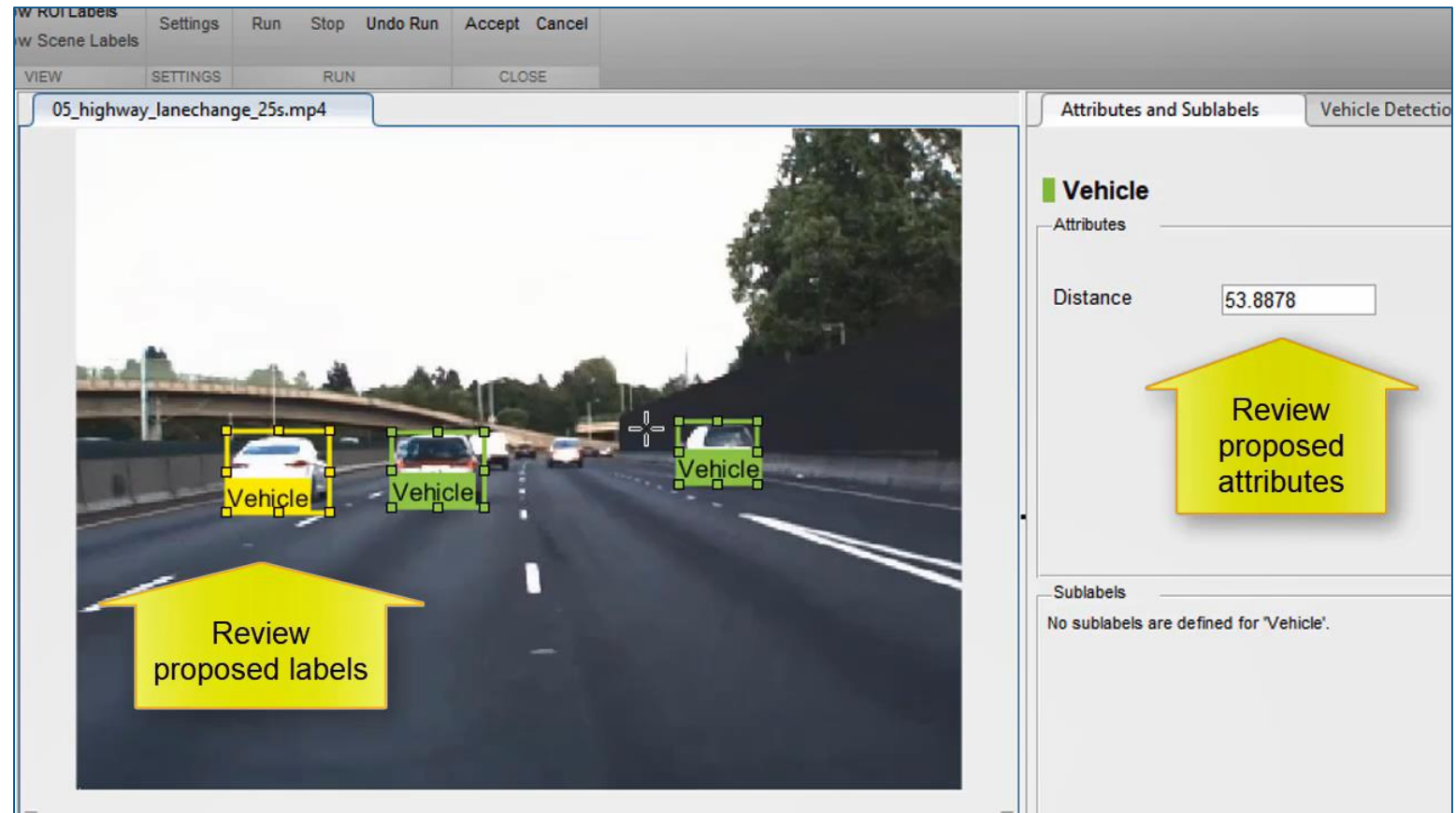
# Import custom automation algorithms

## Automate Attributes of Labeled Objects

- Import automation algorithm into Ground Truth Labeling app
- Detect vehicles from monocular camera
- Estimate distance to detected vehicles
- Run automation algorithm and interactively validate labels

Automated Driving Toolbox™

R2018b

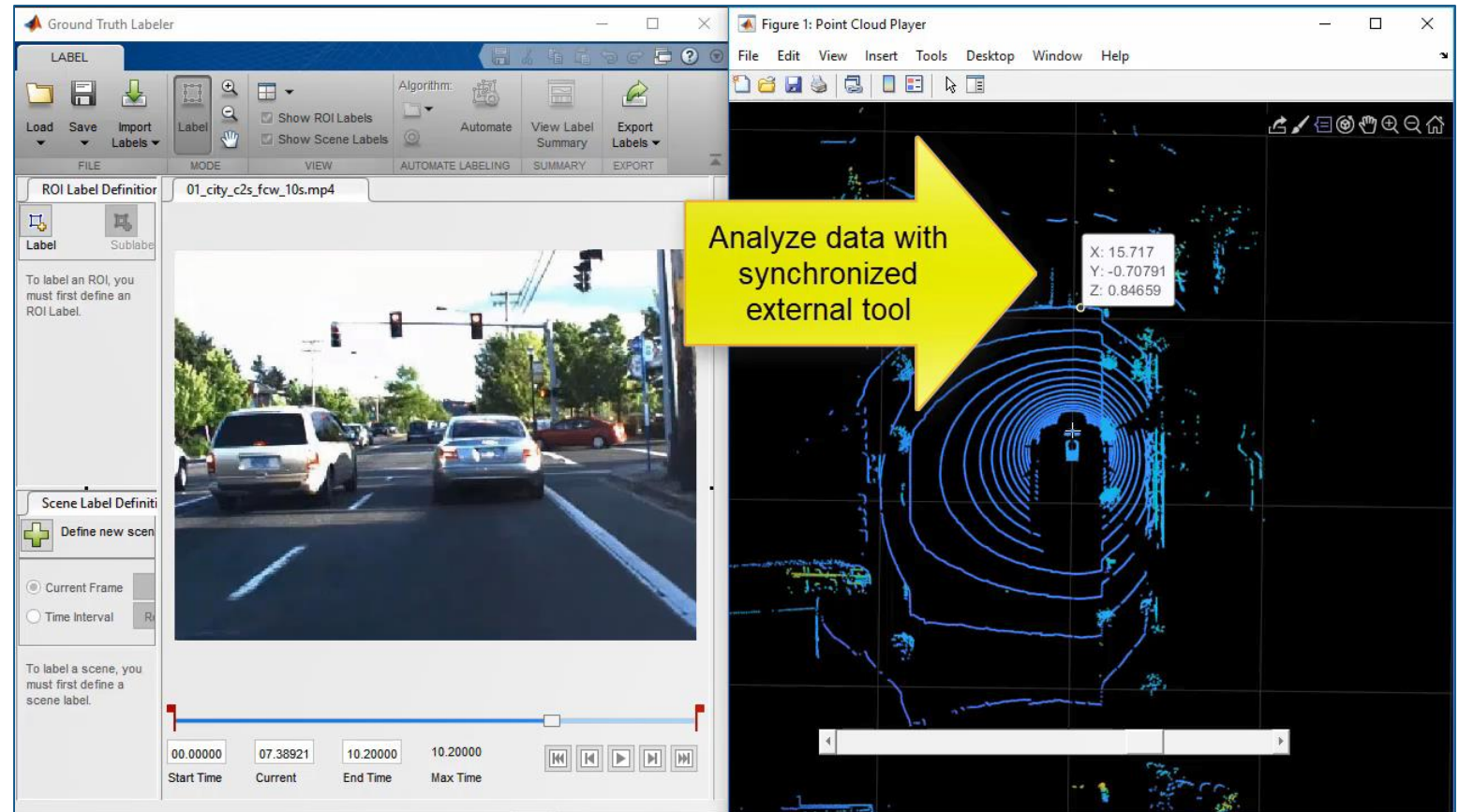


# Add custom visualizations for multi-sensor data

## Connect Lidar Display to Ground Truth Labeler

- Sync external tool to each frame change
- Control external tool through playback controls

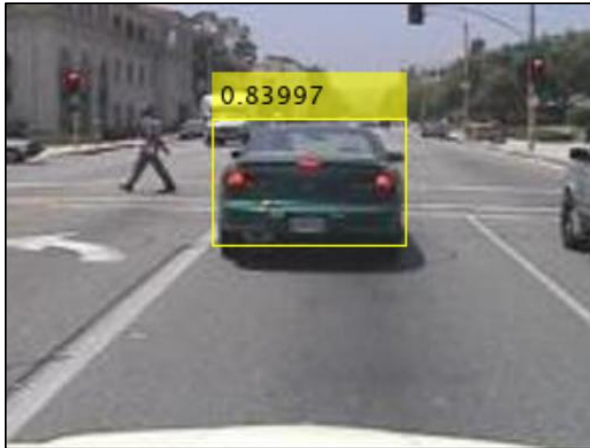
Automated Driving Toolbox™  
**R2017a**





# Design camera, lidar, and radar perception algorithms

## Detect vehicle with camera



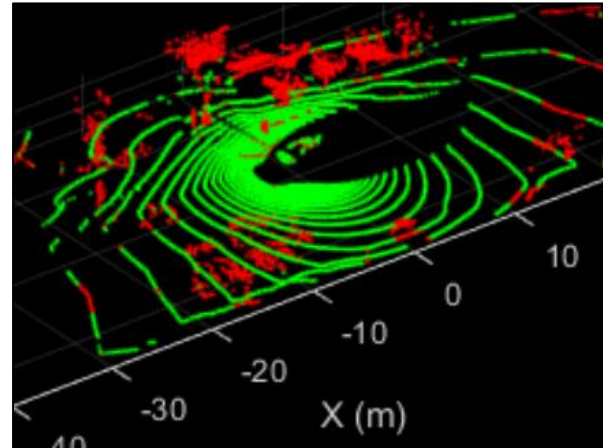
[Object Detection Using YOLO v2 Deep Learning](#)

*Computer Vision Toolbox™*

*Deep Learning Toolbox™*

**R2019a**

## Detect ground with lidar

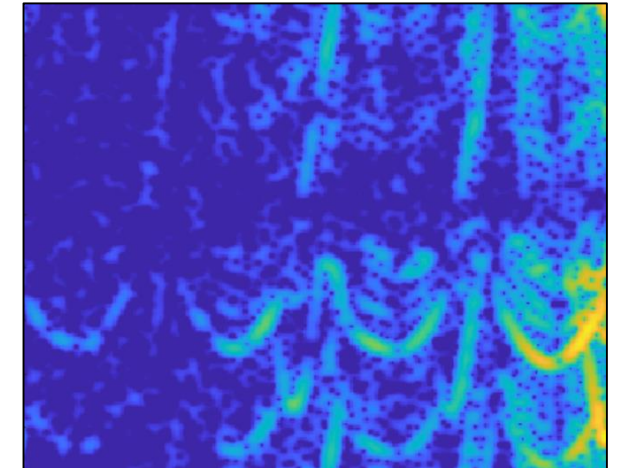


[Segment Ground Points from Organized Lidar Data](#)

*Computer Vision Toolbox™*

**R2018b**

## Detect pedestrian with radar



[Introduction to Micro-Doppler Effects](#)

*Phased Array System Toolbox™*

**R2019a**

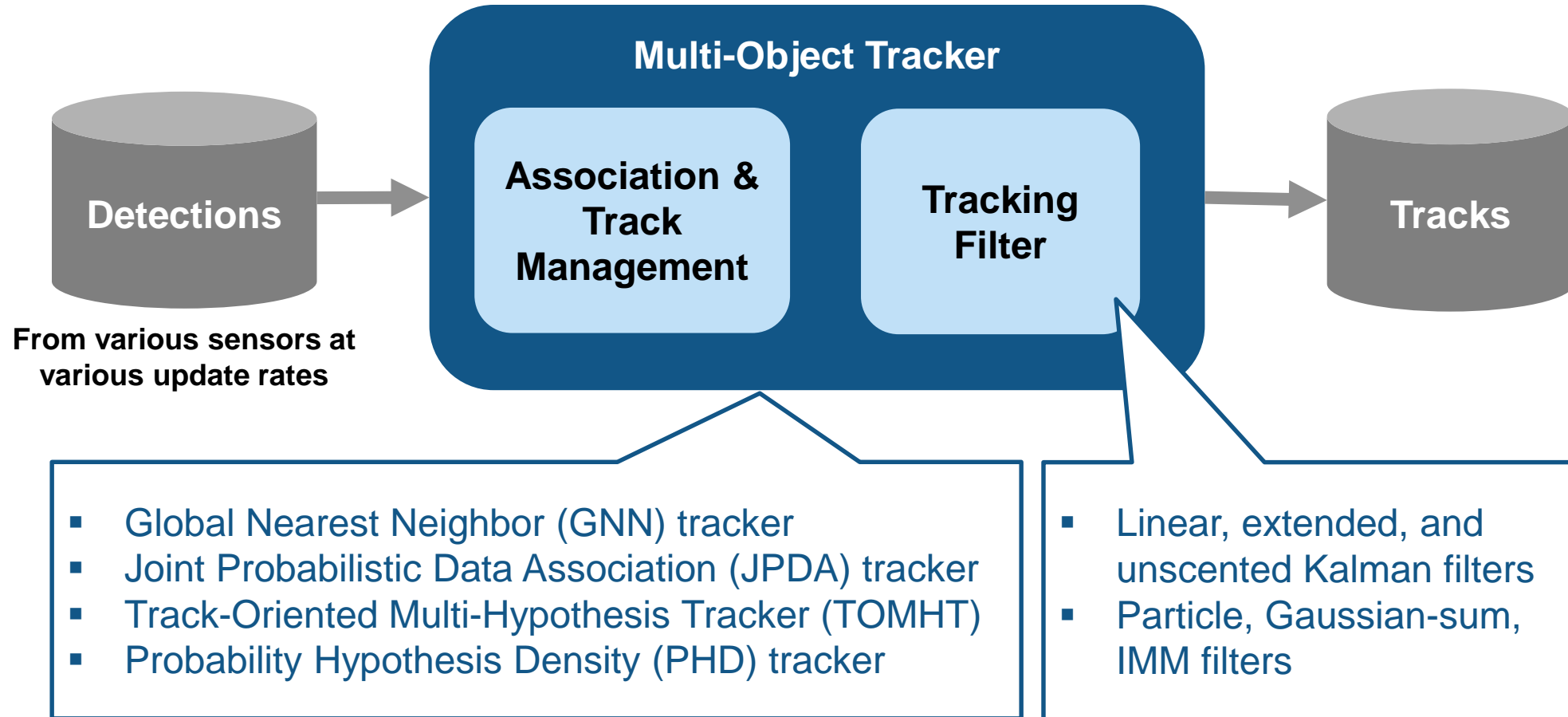
Deep Learning and Reinforcement Learning Workflows in AI

16:15–16:45

LiDAR Processing for Automated Driving

12:45–13:15

# Design multi-object trackers



*Sensor Fusion and Tracking Toolbox™*

*Automated Driving Toolbox™*

# Design multi-object trackers

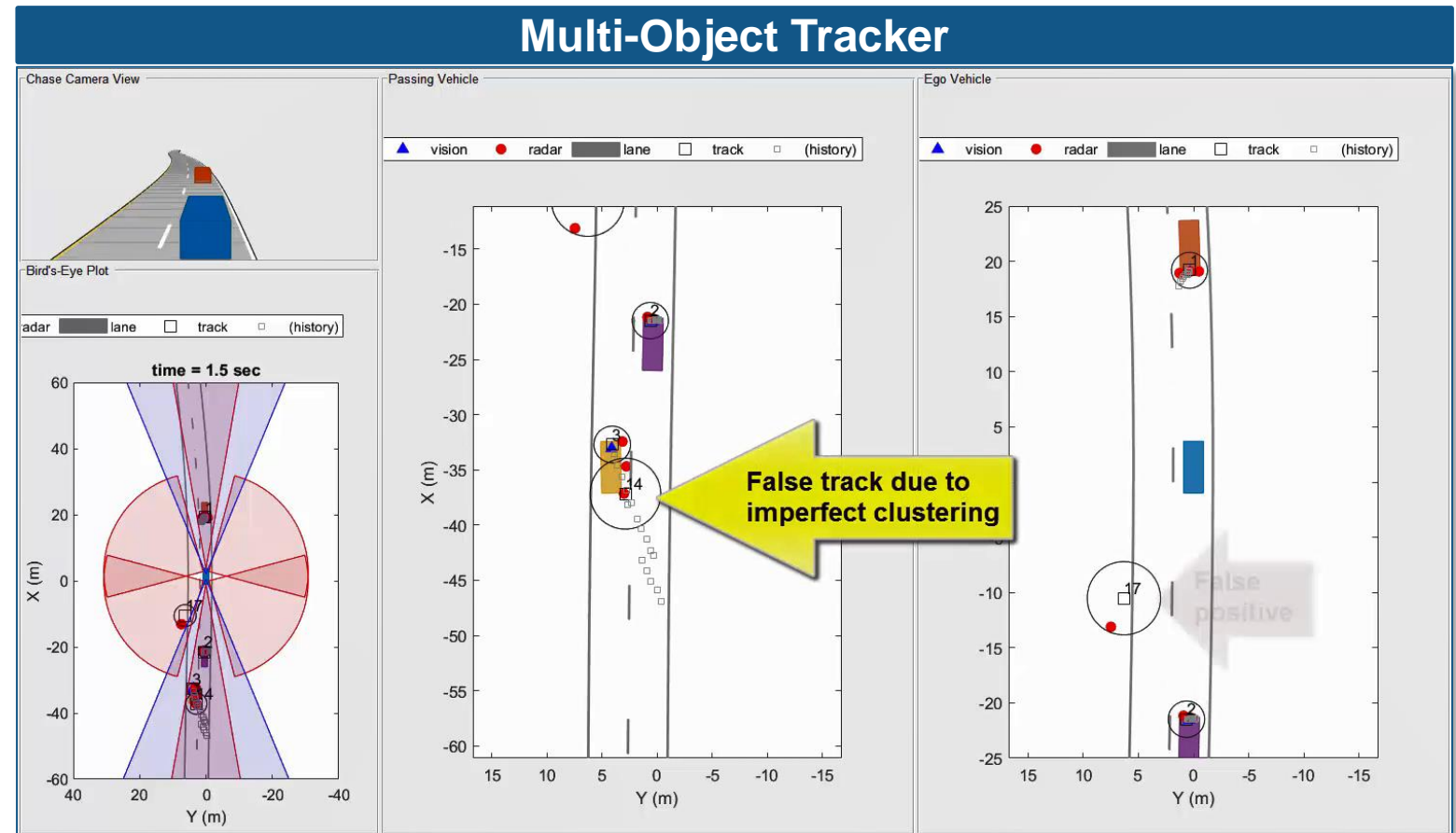
## Extended Object Tracking

- 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**



# Design extended object trackers

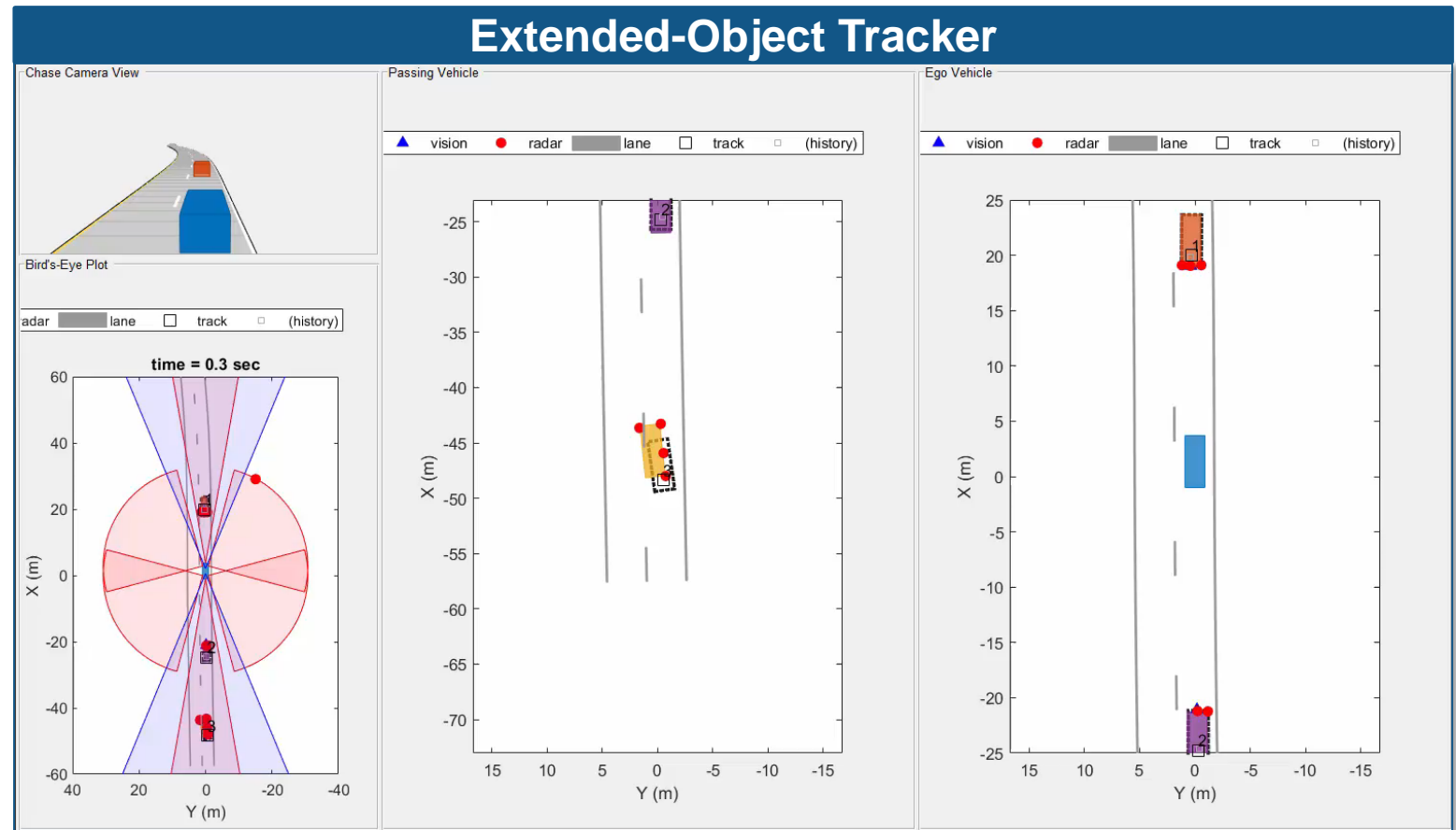
## Extended Object Tracking

- 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**



# Evaluate tracking performance

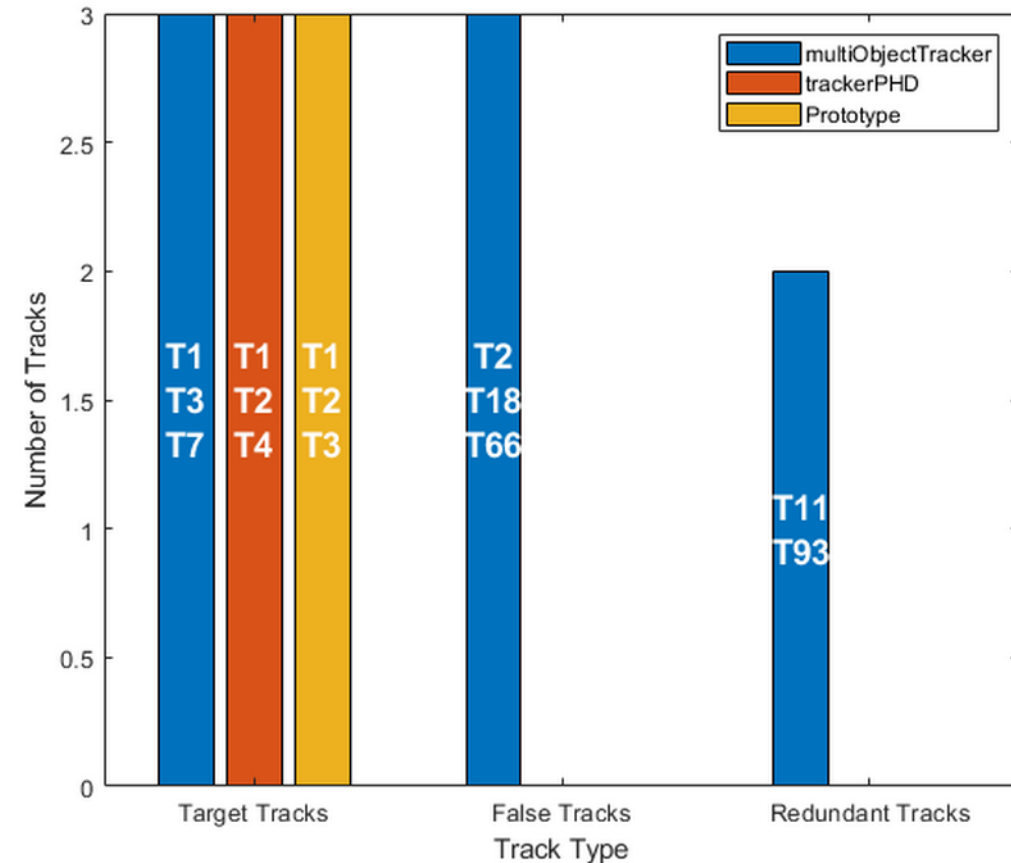
## Extended Object Tracking

- 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**



- Multi-object tracker
- Probability Hypothesis Density tracker
- Extended object (size and orientation) tracker

# Evaluate error metrics

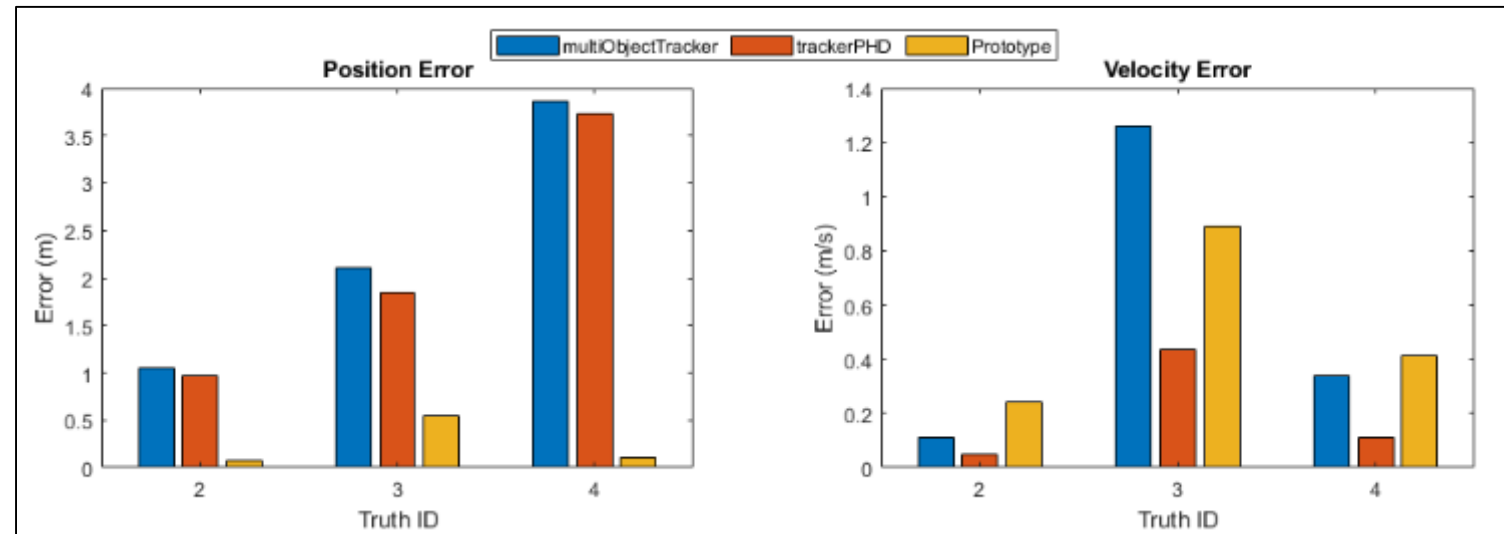
## Extended Object Tracking




- 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**



-  Multi-object tracker
-  Probability Hypothesis Density tracker
-  Extended object (size and orientation) tracker

# Compare relative execution times of object trackers

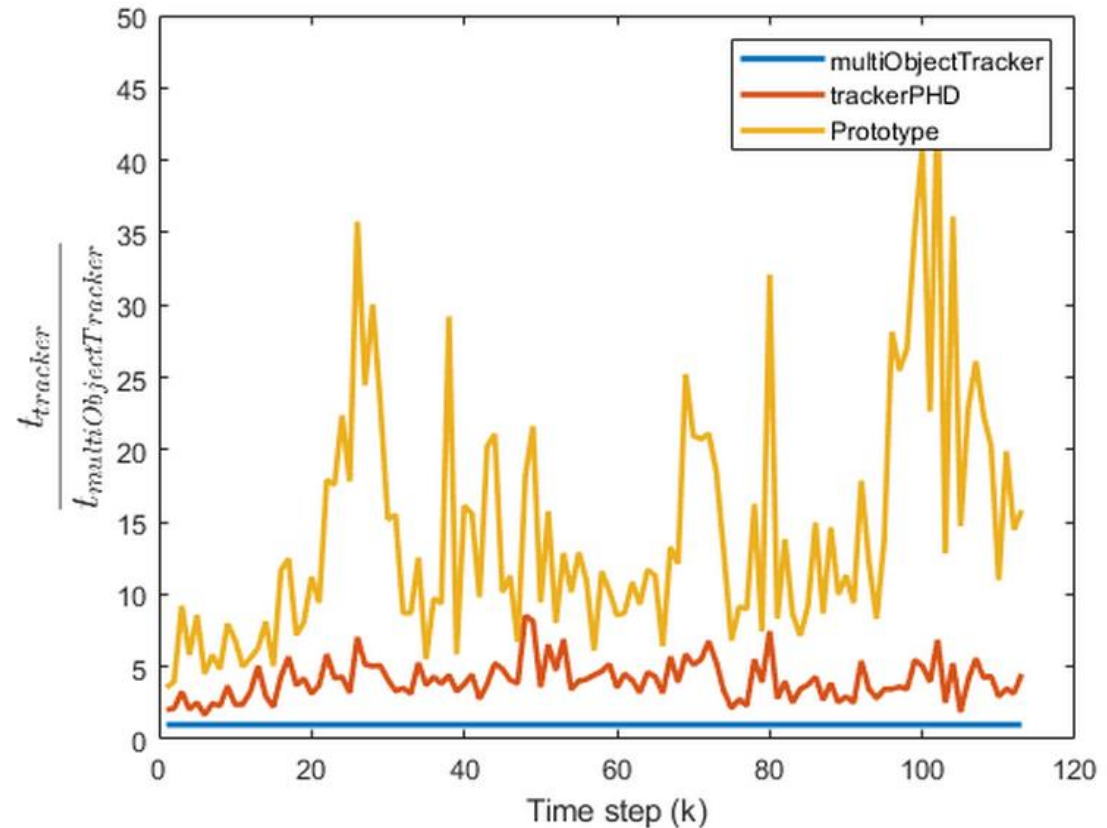
## Extended Object Tracking

- 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 tracker for lidar point cloud data

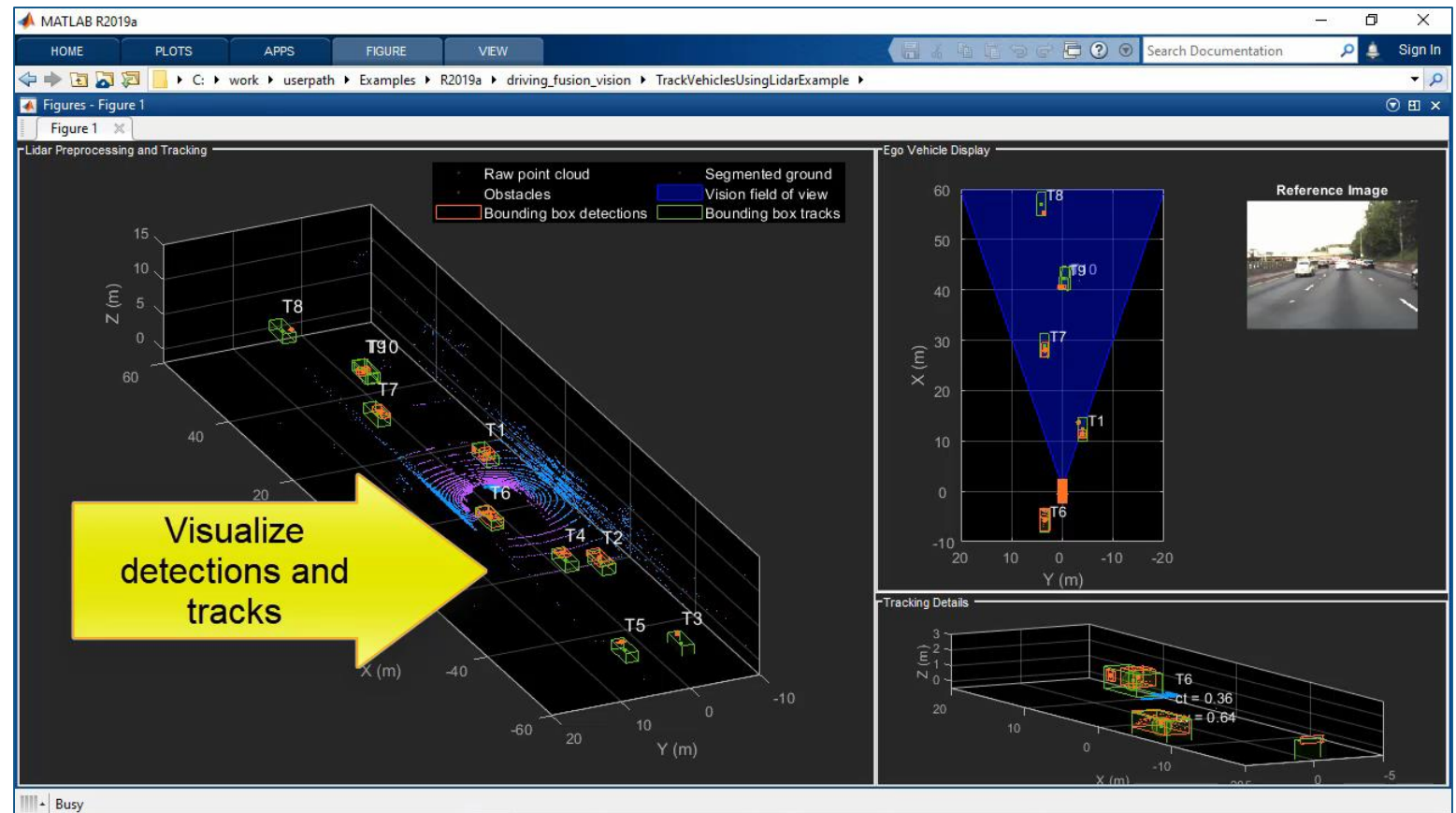
## Track Vehicles Using Lidar: From Point Cloud to Track List

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

*Sensor Fusion and Tracking  
Toolbox™*

*Computer Vision Toolbox™*

**R2019a**



LiDAR Processing for Automated Driving

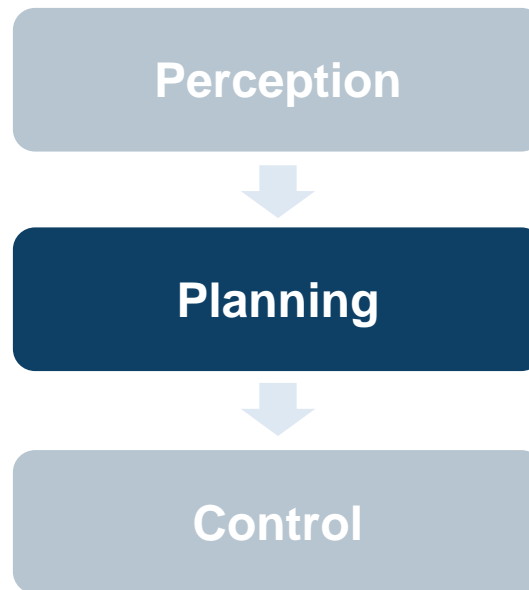
12:45–13:15



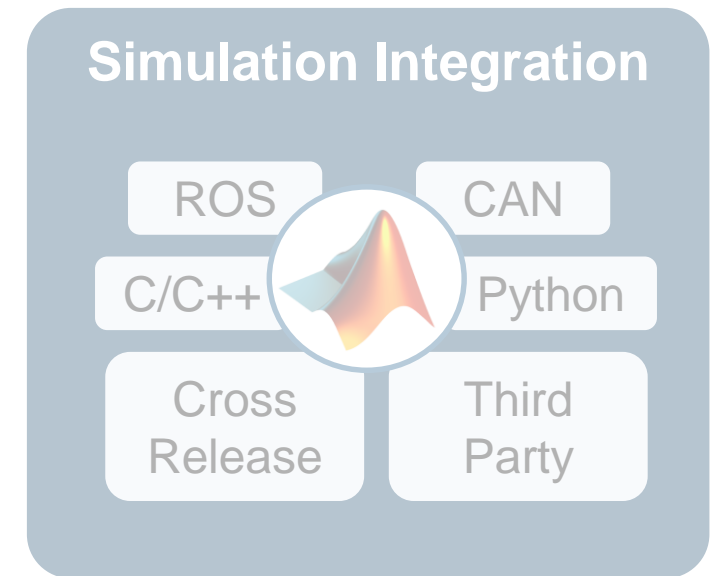
# Some common questions from automated driving engineers



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?

# 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

The figure window displays a camera view of a road with a traffic jam, a map view of the same road, and a data panel showing a timestamp of 22:11:25 and a speed limit of 35. The map view highlights a lane group and a link on N San Antonio Rd.

```

286
287 % Visu
288 % The m
289 % lane
290 % coord
291 % link
292 % confi
293
294 % The
295 % |<mat
296 % Help
297 % a rec
298 % HD Li
299 hdlmUI
300
301 % Synch
302 synchro
303 videoRe
  
```

Timestamp: 22:11:25  
Speed Limit: 35

Lane Types and Boundaries: ↑ - - - ↑ BICYCLE

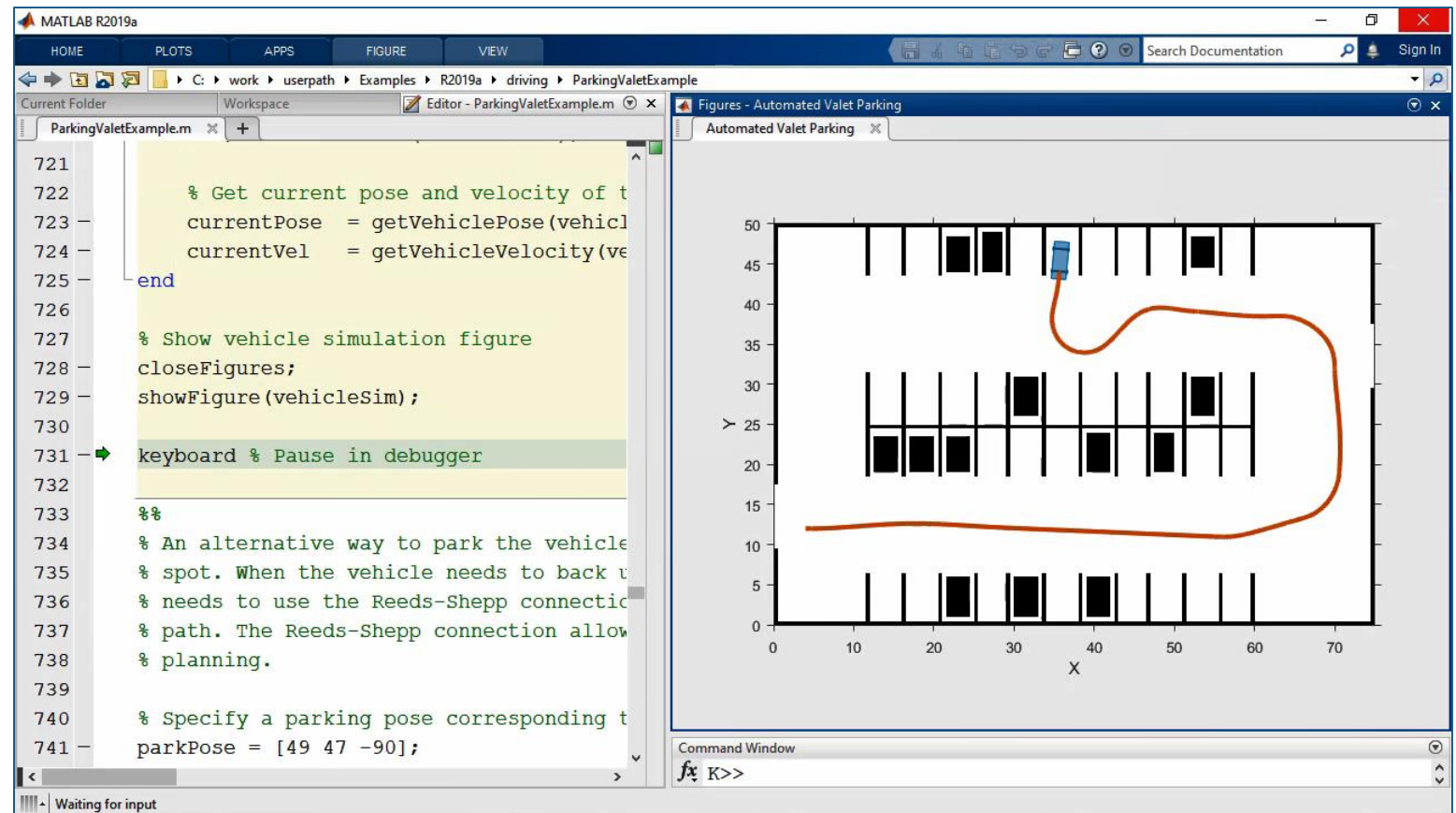
# Design path planner

## 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



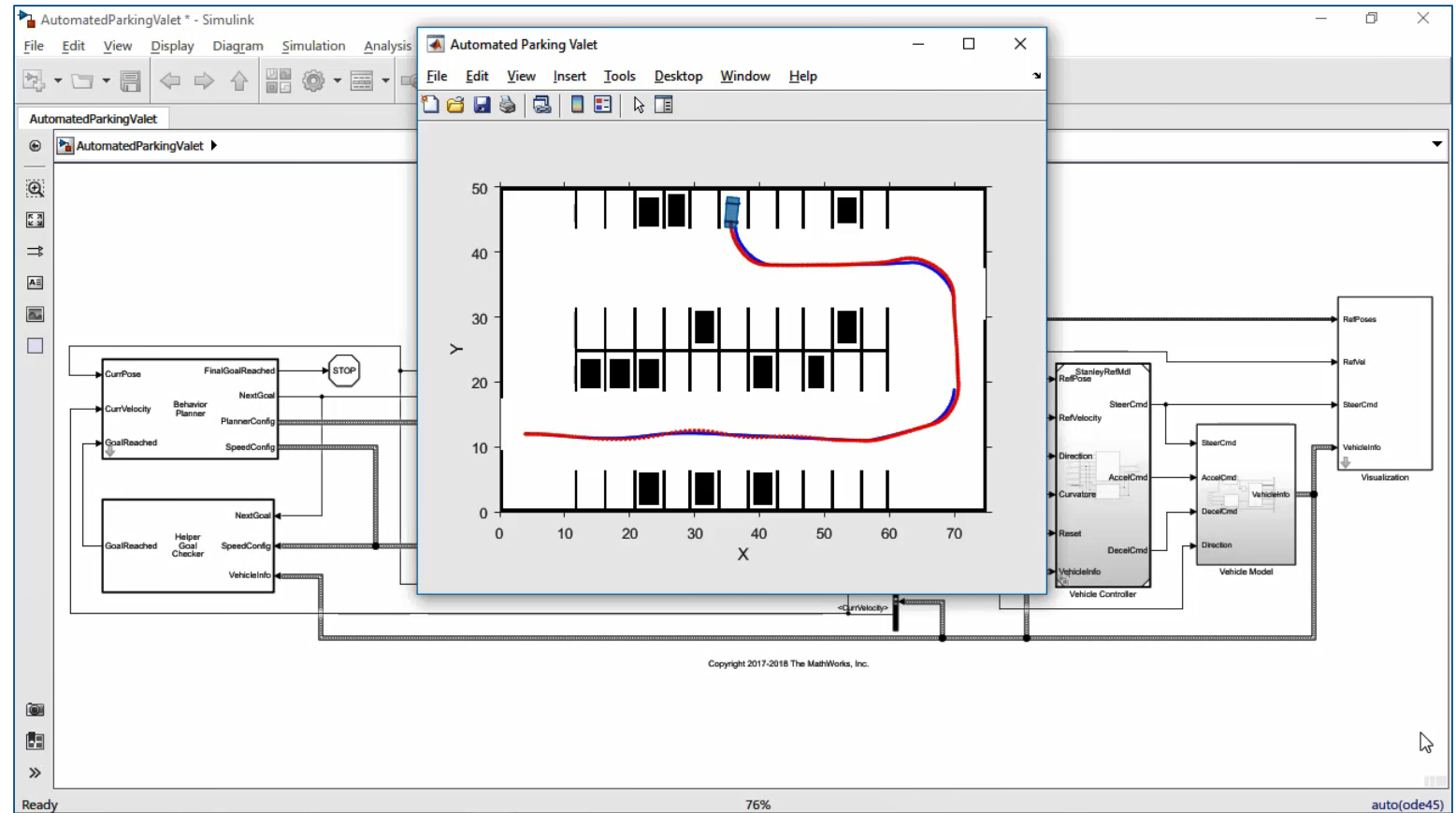
# Design path planner and controller

## 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

Automated Driving Toolbox™

R2018b



# Generate C/C++ code for path planner and controller

## Code Generation for Path Planning and Vehicle Control

- Simulate system
- Configure for code generation
- Generate C/C++ code
- Test using Software-In-the-Loop
- Measure execution time of generated code

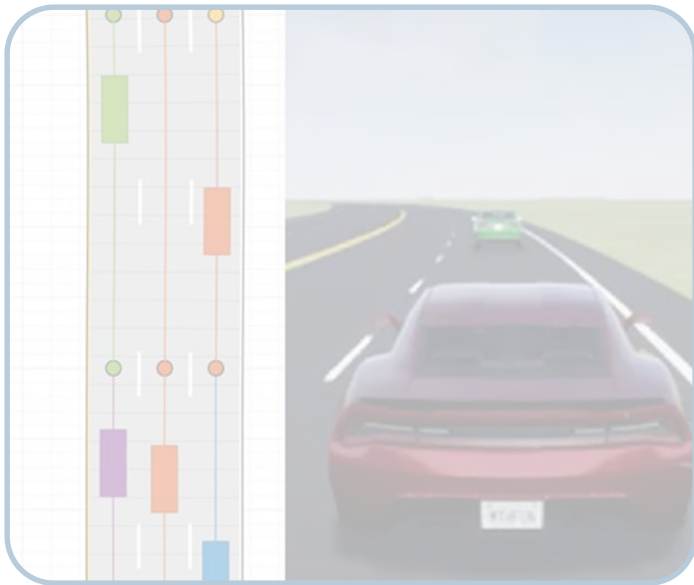
Automated Driving Toolbox™

Embedded Coder

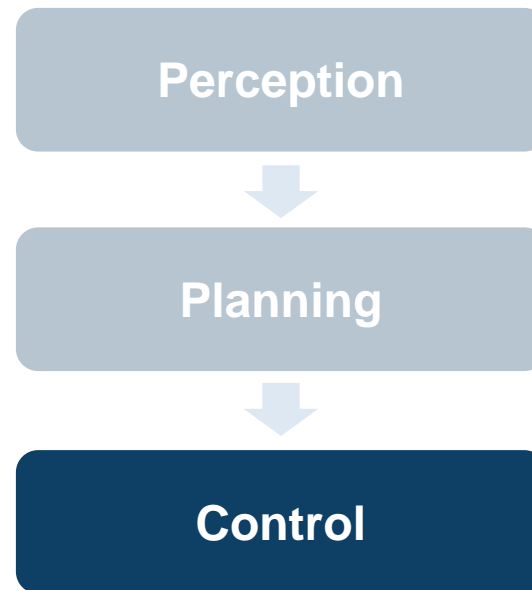
R2019a

```
186
187 // model step function
188 void step0();
189
190 // model step function
191 void step1();
192
193 // model terminate function
194 void terminate();
195
196 // Constructor
197 AutomatedParkingValetModelClass();
198
199 // Destructor
200 ~AutomatedParkingValetModelClass();
201
202 // Root inport: '<Root>/Costmap' set method
203 void setCostmap(costmapBus localArgInput);
204
205 // Root inport: '<Root>/GoalPose' set method
206 void setGoalPose(real_T localArgInput[3]);
207
```

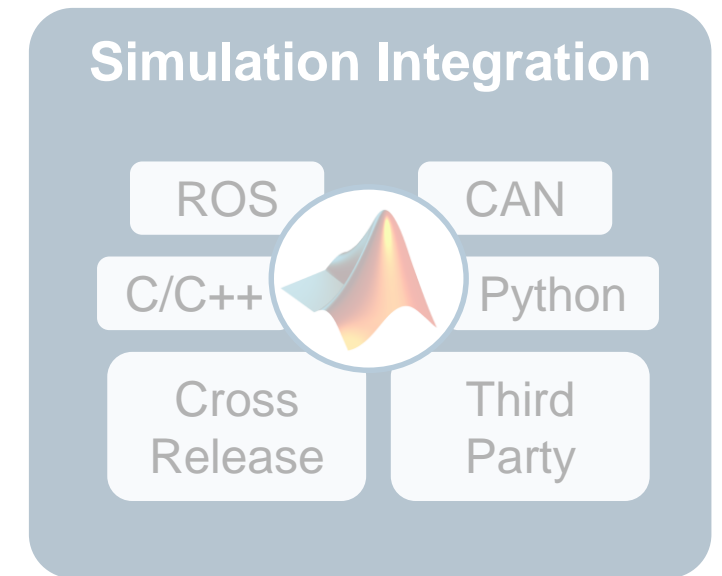
# Some common questions from automated driving engineers



How can I  
**synthesize scenarios**  
to test my designs?



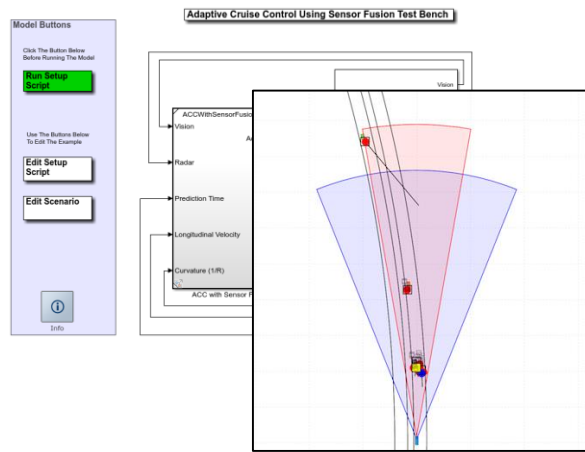
How can I  
**discover and design**  
in multiple domains?



How can I  
**integrate**  
with other environments?

# Design lateral and longitudinal Model Predictive Controllers

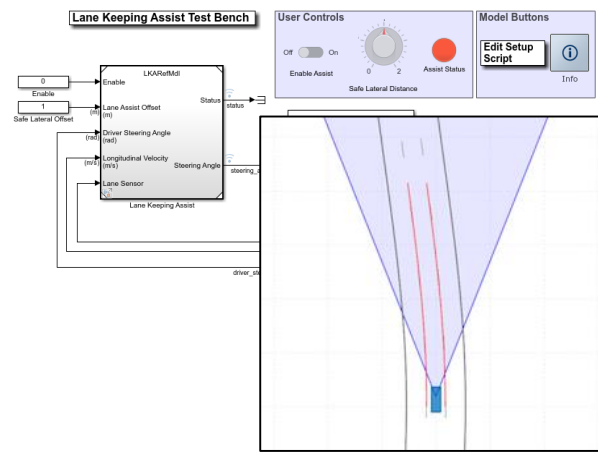
## Longitudinal Control



### Adaptive Cruise Control with Sensor Fusion

Automated Driving Toolbox™  
 Model Predictive Control  
 Toolbox™  
 Embedded Coder® **R2017b**

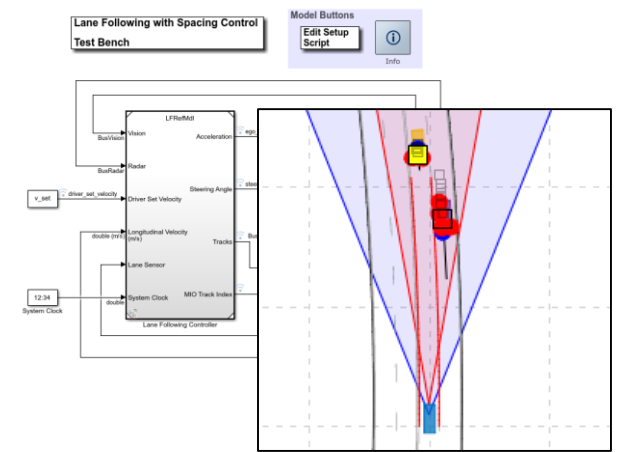
## Lateral Control



### Lane Keeping Assist with Lane Detection

Automated Driving Toolbox™  
 Model Predictive Control  
 Toolbox™  
 Embedded Coder® **R2018a**

## Longitudinal + Lateral



### Lane Following Control with Sensor Fusion and Lane Detection

Automated Driving Toolbox™  
 Model Predictive Control Toolbox™  
 Embedded Coder® **R2018b**

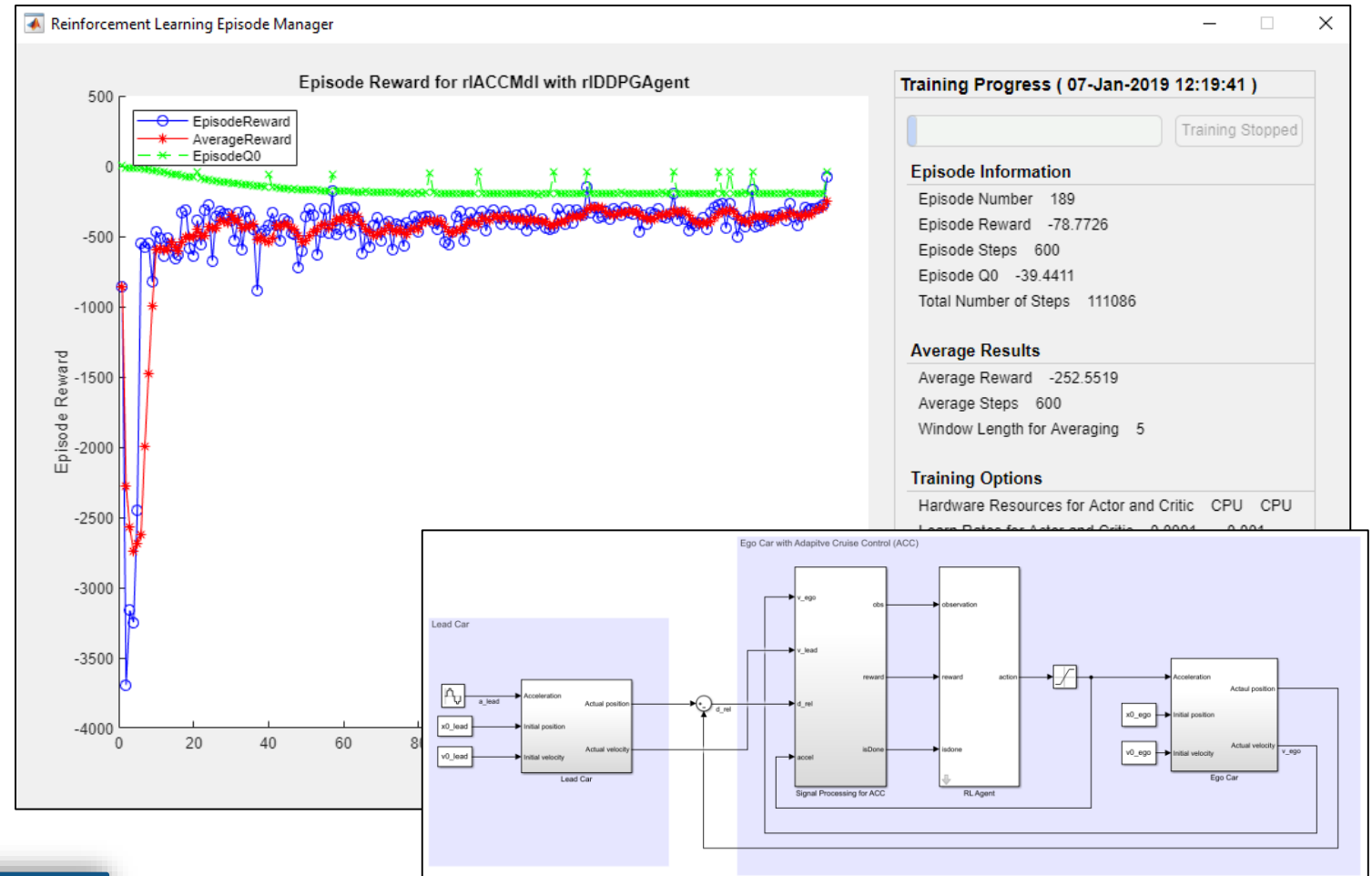
Develop and Test Vehicle Controllers for ADAS and Automated Driving Applications Through System Simulation  
 15:00–15:30

# 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™  
**R2019a**



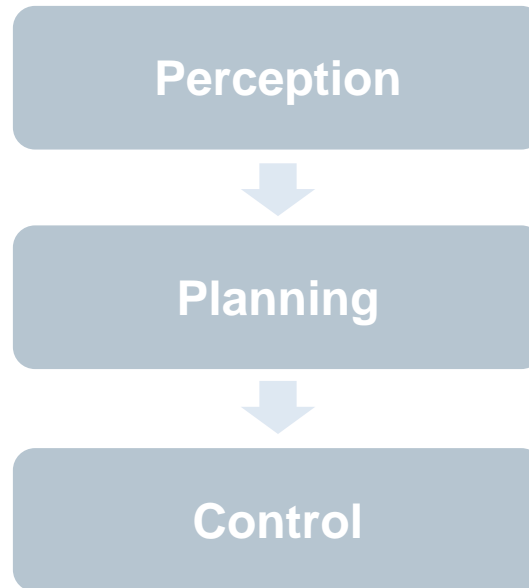
Deep Learning and Reinforcement Learning Workflows in AI  
 16:15–16:45



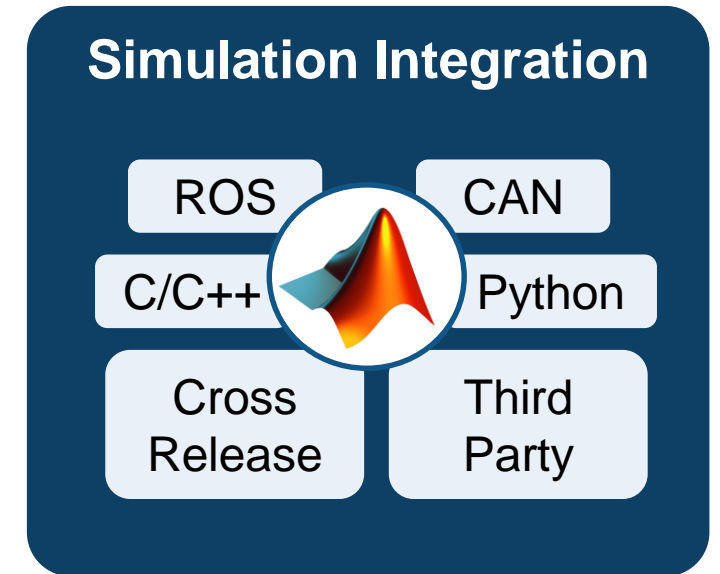
# Some common questions from automated driving engineers



How can I **synthesize scenarios** to test my designs?



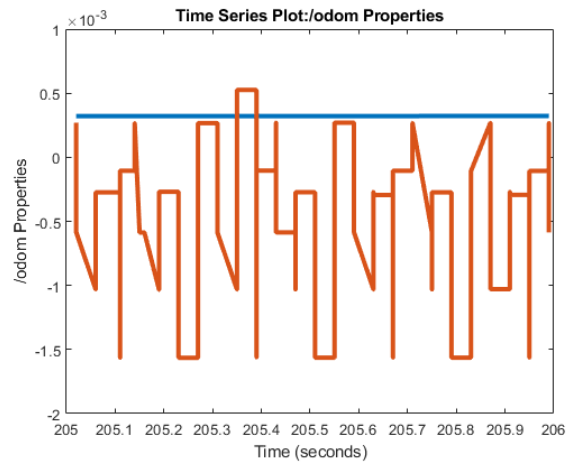
How can I **discover and design** in new domains?



How can I **integrate** with other environments?

# Integrate with ROS

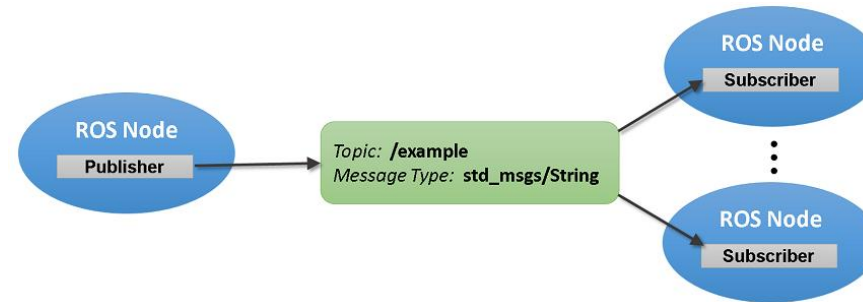
## Replay logged ROS data



[Work with rosbag Logfiles](#)

Robotic System Toolbox™

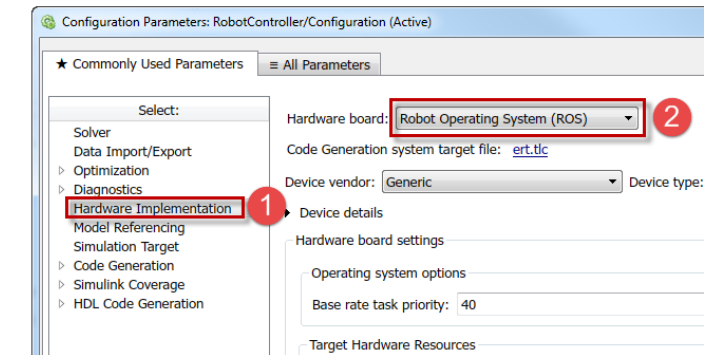
## Connect to live ROS data



[Exchange Data with ROS Publishers and Subscribers](#)

Robotic System Toolbox™

## Generate standalone ROS node



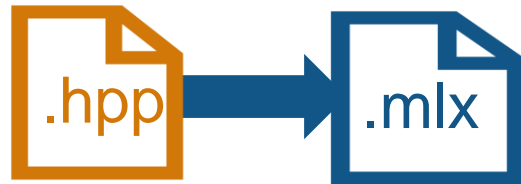
[Generate a Standalone ROS Node from Simulink](#)

Robotic System Toolbox™

Simulink Coder™

# Call C++, Python, and OpenCV from MATLAB

## Call C++



[Import C++ Library  
Functionality into MATLAB](#)

MATLAB®

**R2019a**

## Call Python

```
tw = ...
py.textwrap.TextWrapper(...
    pyargs(...
        'initial_indent', '% ',...
        'subsequent_indent', '% ',...
        'width', int32(30)))
```

[Call Python from MATLAB](#)

MATLAB®

**R2014a**

## Call OpenCV & OpenCV GPU

```
cv::Rect
cv::KeyPoint
cv::Size
cv::Mat
cv::Ptr
...
```



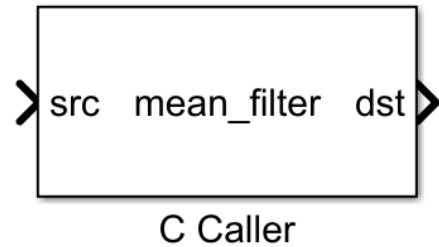
[Install and Use Computer Vision  
Toolbox OpenCV Interface](#)

Computer Vision System Toolbox™  
OpenCV Interface Support Package

Updated **R2018b**

# Call C code from Simulink

## Call C code



[Bring Custom Image Filter Algorithms as Reusable Blocks in Simulink](#)

Simulink®

R2017b

## Create buses from C structs

```
typedef struct {
    double coeff;
    double init;
    fault_T fault;
} params_T;
```

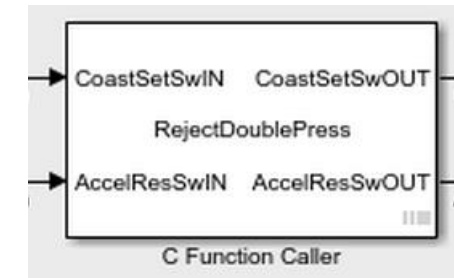
Name	DataType
-coeff	double
-init	double
-fault	Enum: fault_T

[Import Structure and Enumerated Types](#)

Simulink®

R2017a

## Test and verify C code



AGGREGATED COVERAGE RESULTS

ANALYZED MODEL	DECISION	CONDITION	MCDC
RejectDoublePress.c	100%	100%	100%

[Custom C Code Verification with Simulink Test](#)

Simulink Test™

Simulink Coverage™

R2019a

# Cross-release simulation through code generation

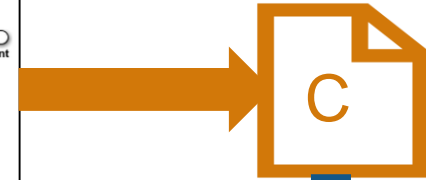
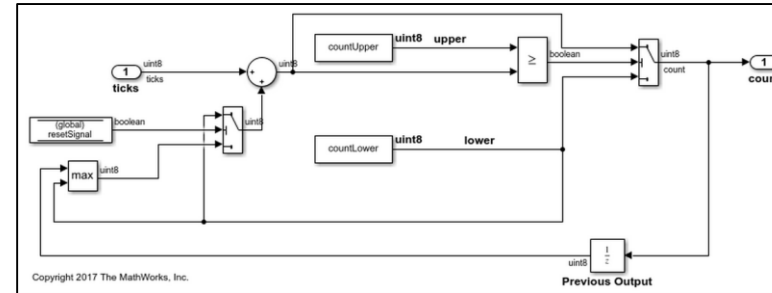
## Integrate Generated Code by Using Cross-Release Workflow

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

Embedded Coder

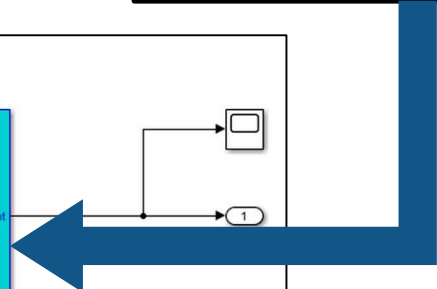
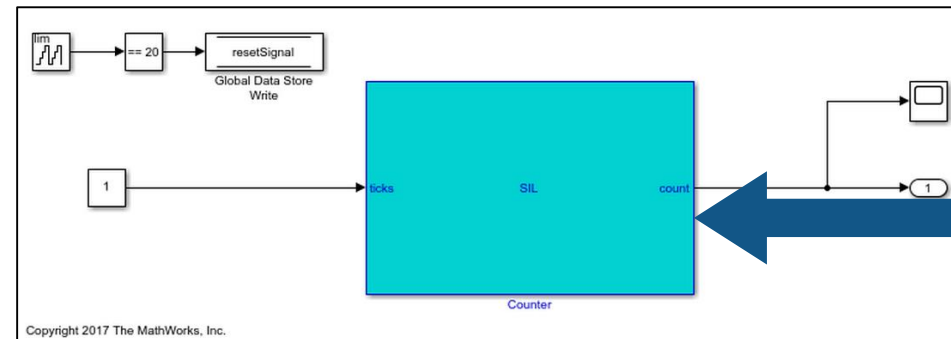
**R2016a**

## Previous Release

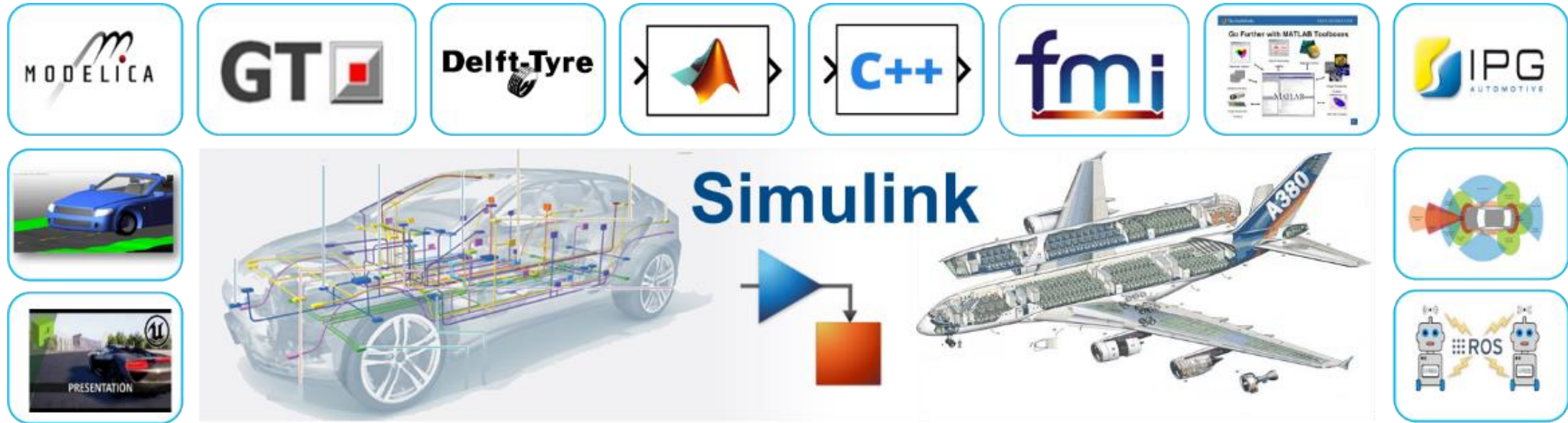


**crossReleaseImport**

## Current Release



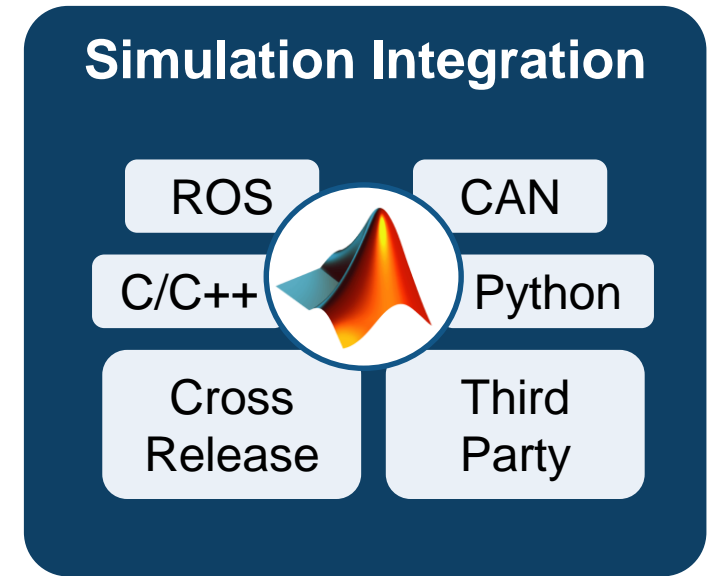
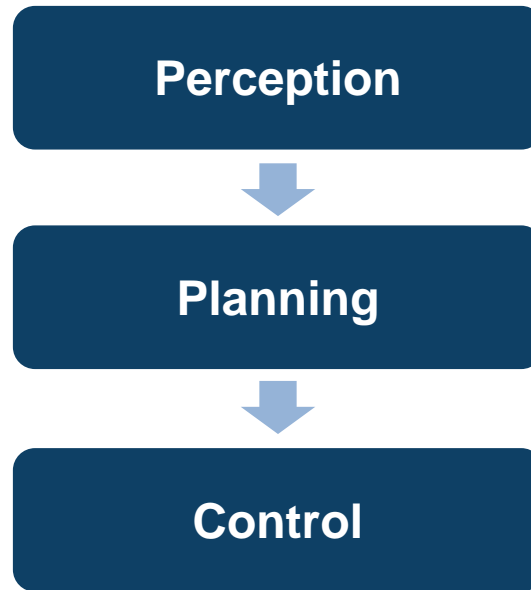
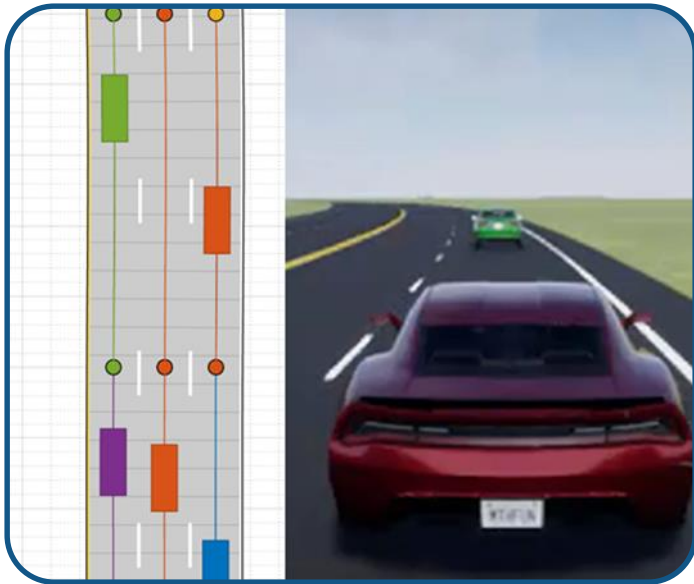
# Connect to third party tools



152 Interfaces to 3<sup>rd</sup> Party  
 Modeling and Simulation Tools  
 (as of March 2019)



# Some common questions from automated driving engineers

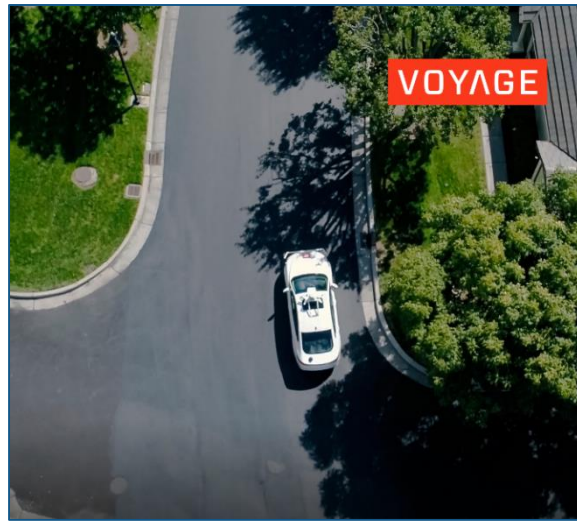


**Synthesize scenarios**  
to test my designs

**Discover and design**  
in multiple domains

**Integrate**  
with other environments

# MathWorks can help you customize MATLAB and Simulink for your automated driving application



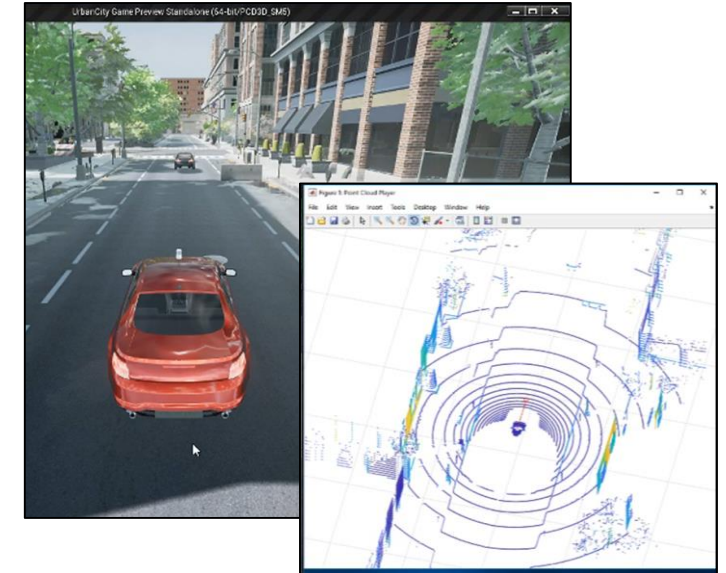
## Voyage develops MPC controller and integrates with ROS

- Developed & deployed in 3 months
- [2018 MathWorks Automotive Conference](#)



## Autoliv labels ground truth lidar data

- > 4x reduction in labeling effort
- Joint paper in SAE (2018-01-0043)
- [2018 MathWorks Automotive Conference](#)



## Ford synthesizes lidar data to test autonomous driving & active safety systems

- Joint paper with Ford
- SAE Paper 2017-01-0107

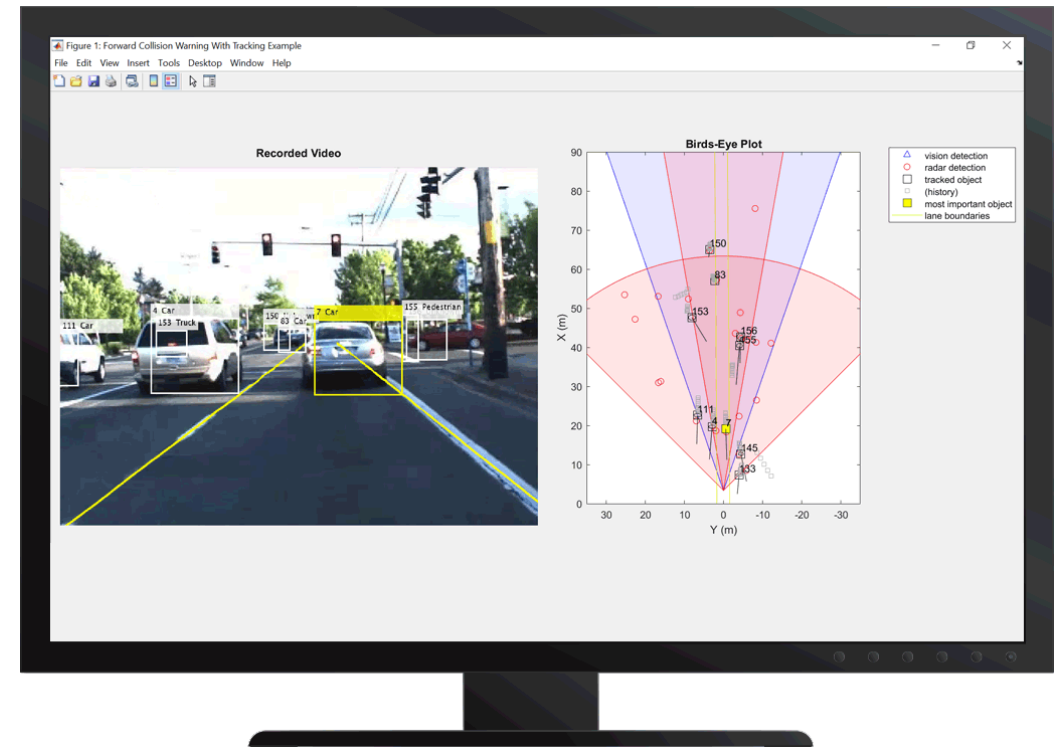


## Automated Driving with MATLAB

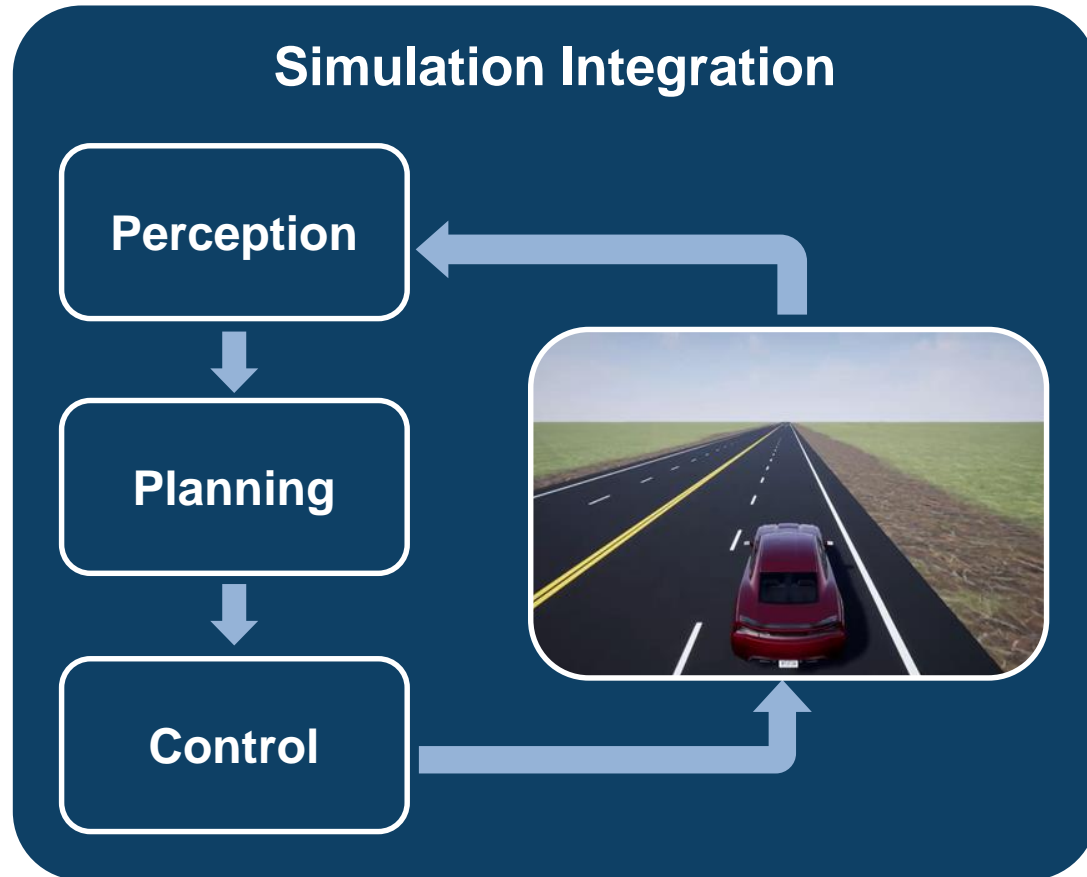
This one-day course provides hands-on experience with developing and verifying automated driving perception algorithms

### Topics include:

- Labeling of ground truth data
- Visualizing sensor data
- Detecting lanes and vehicles
- Fusing sensor detections
- Generating driving scenarios and modeling sensors



# Develop Automated Driving Systems with MATLAB and Simulink



Discuss your application with a MathWorks field engineer to help you structure your evaluation

- Understand your goals
- Recommend tasks
- Answer questions

**Visit us at demo booths**

- **Automated Driving**
- **Deep Learning**

# MATLAB EXPO 2019

Email: [Amod.Anandkumar@mathworks.in](mailto:Amod.Anandkumar@mathworks.in)

LinkedIn: <https://in.linkedin.com/in/ajga2>

Twitter: @\_Dr\_Amod

