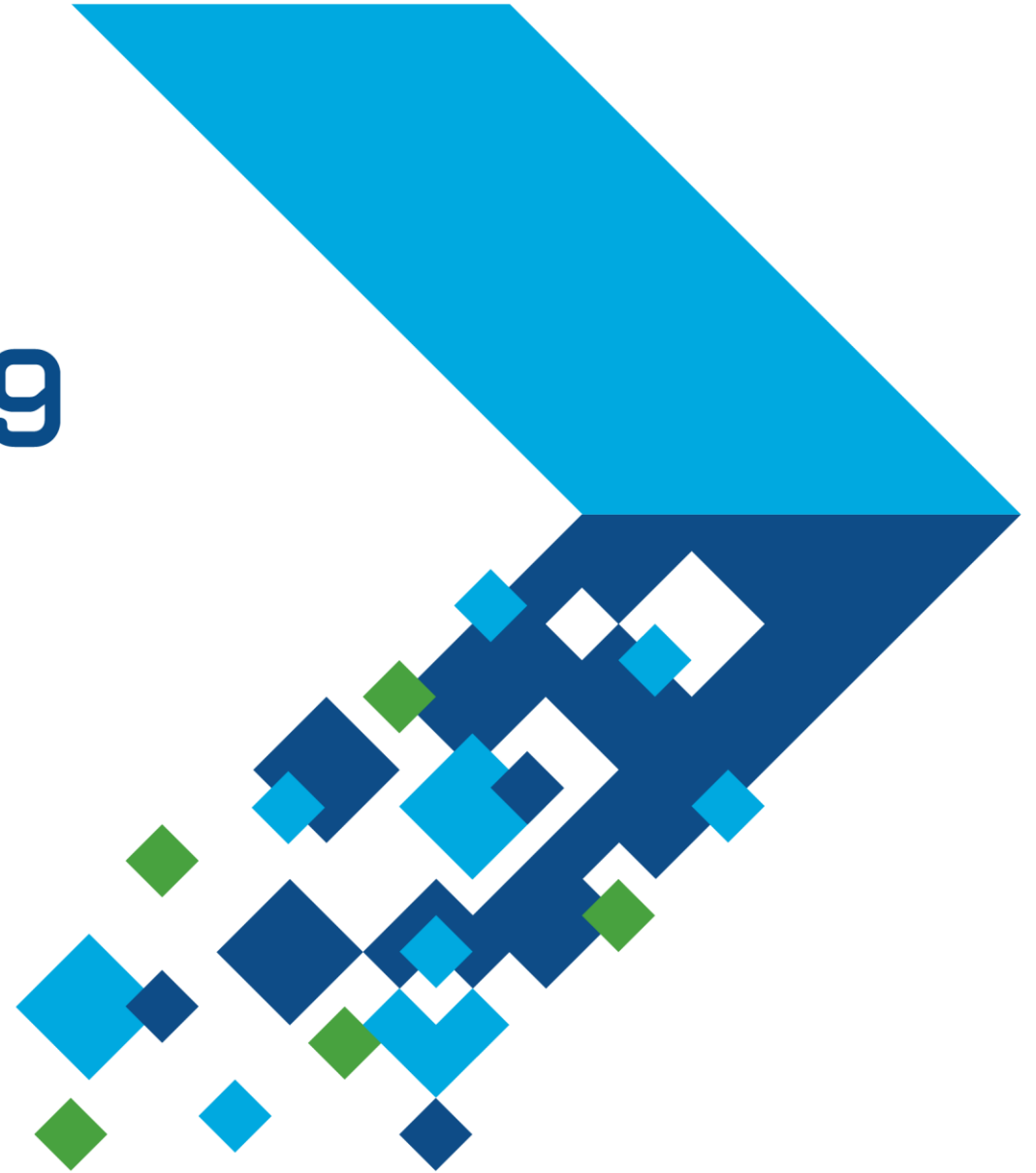


MATLAB EXPO 2019

Automated Driving with MATLAB and Simulink

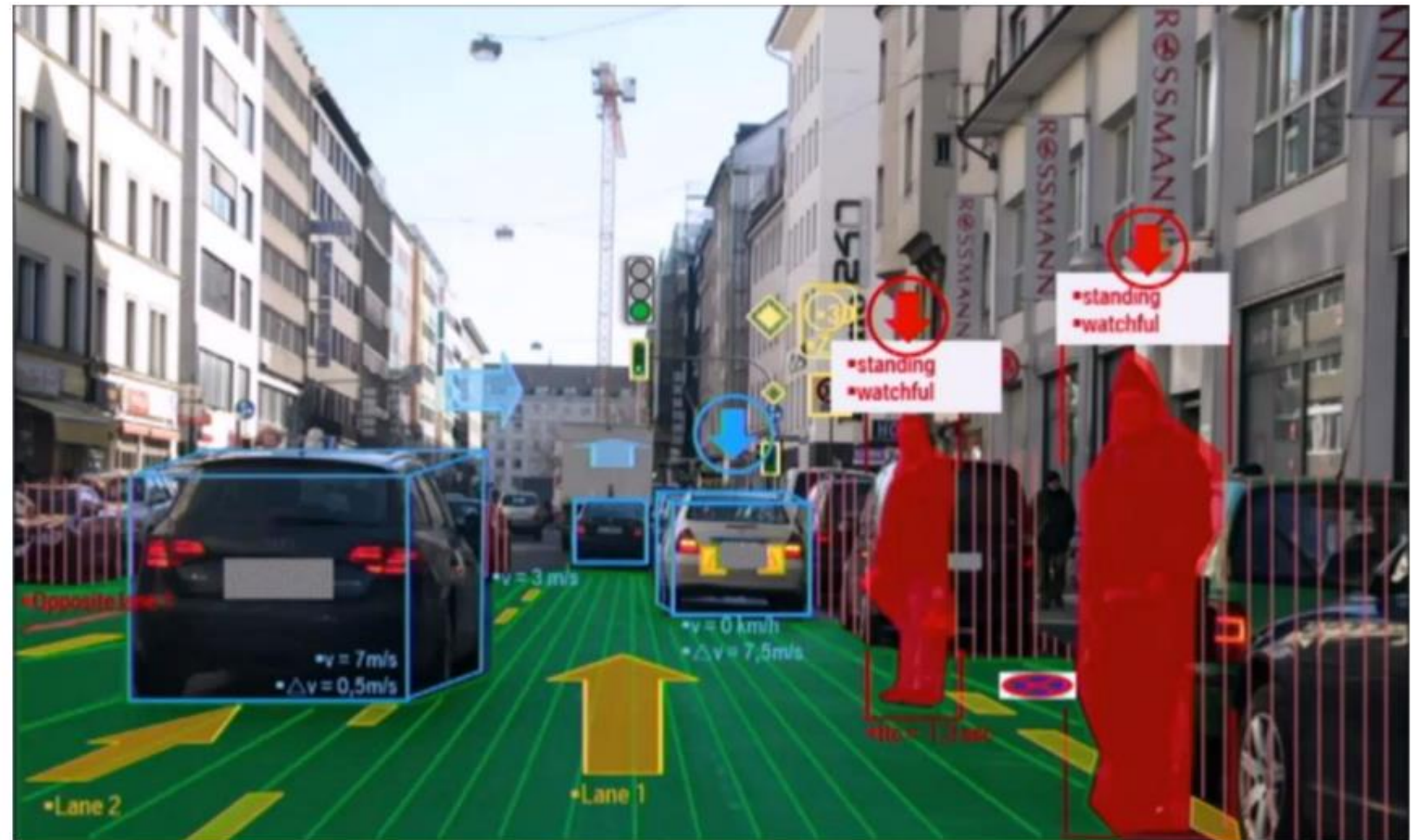
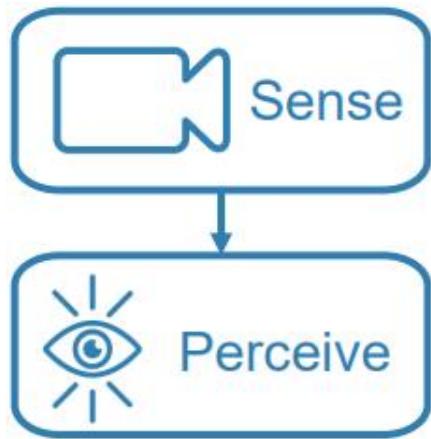
GianCarlo Pacitti
Senior Application Engineer, MathWorks



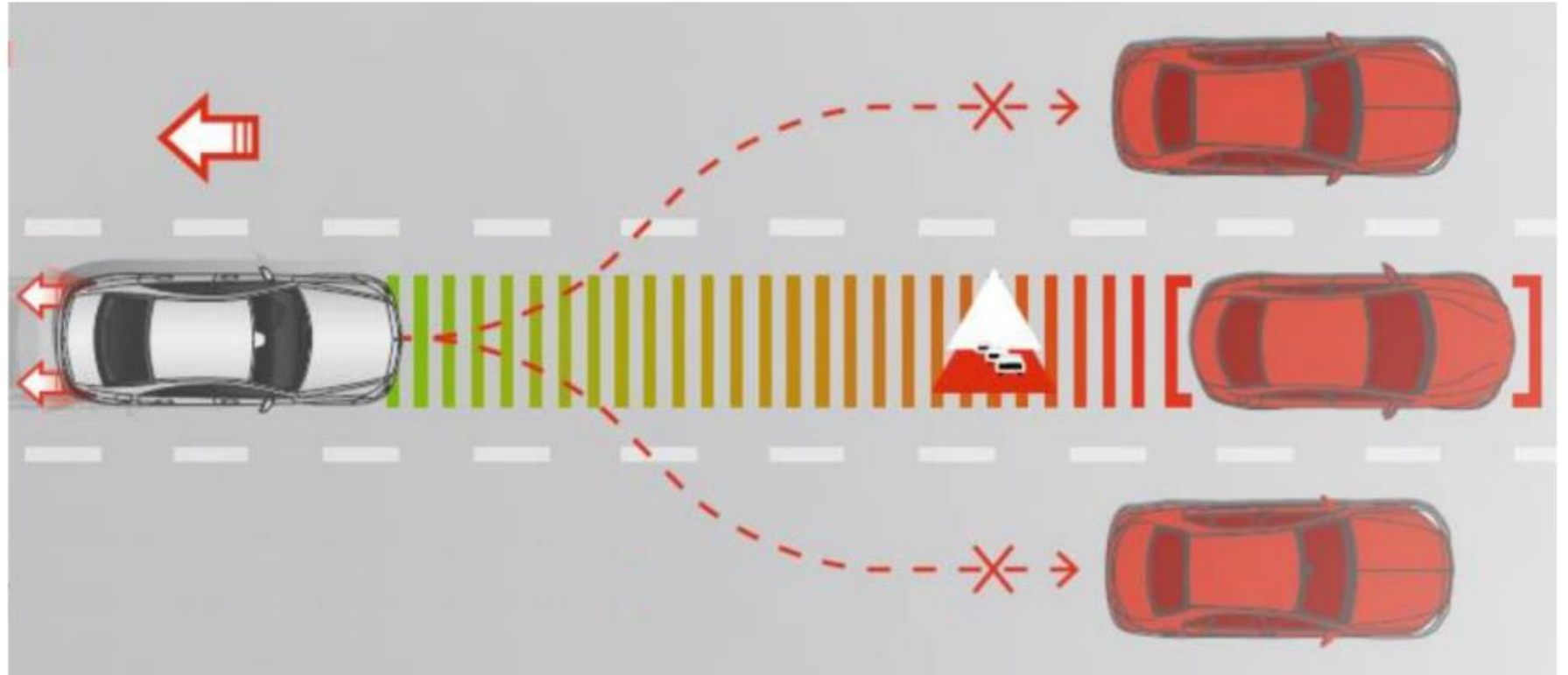
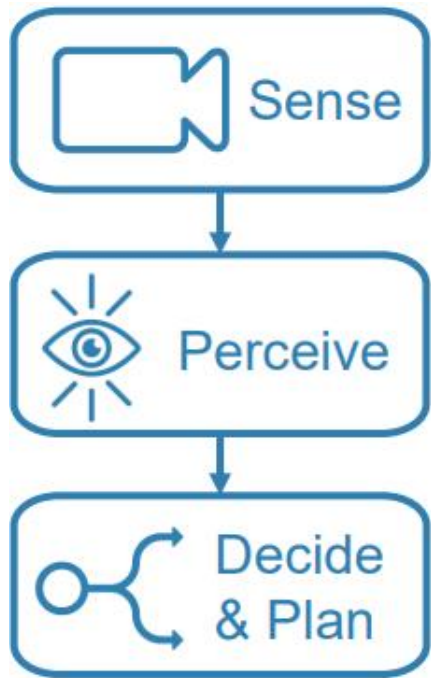
Capabilities of an Autonomous Vehicle



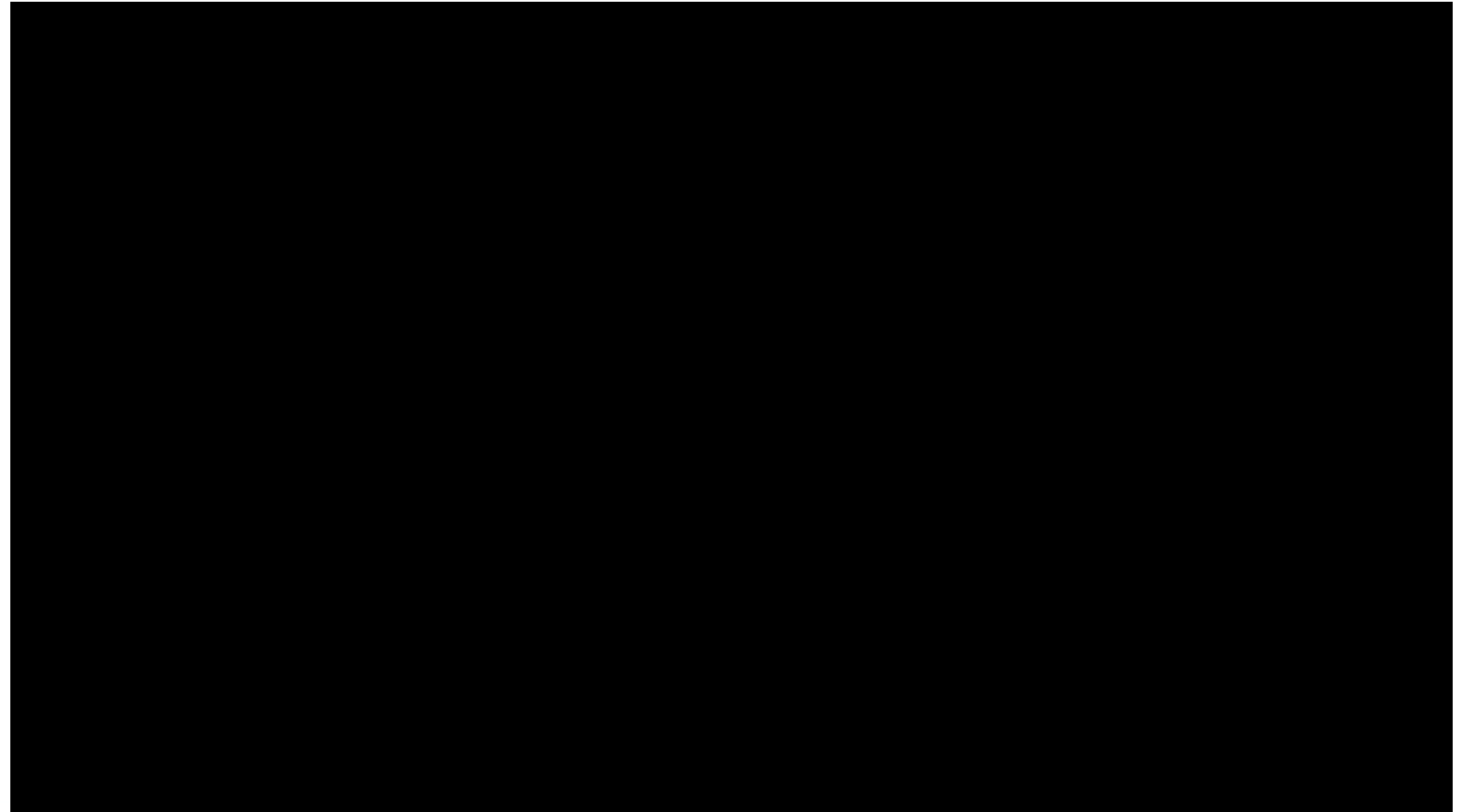
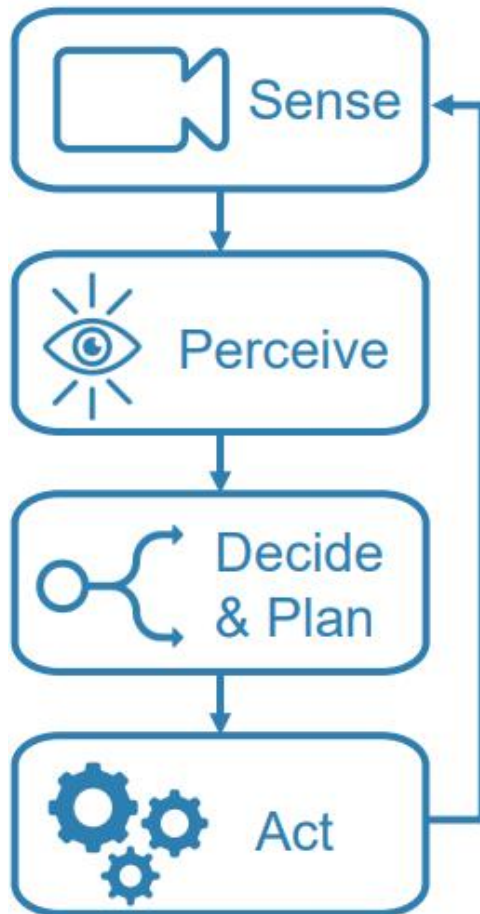
Capabilities of an Autonomous Vehicle



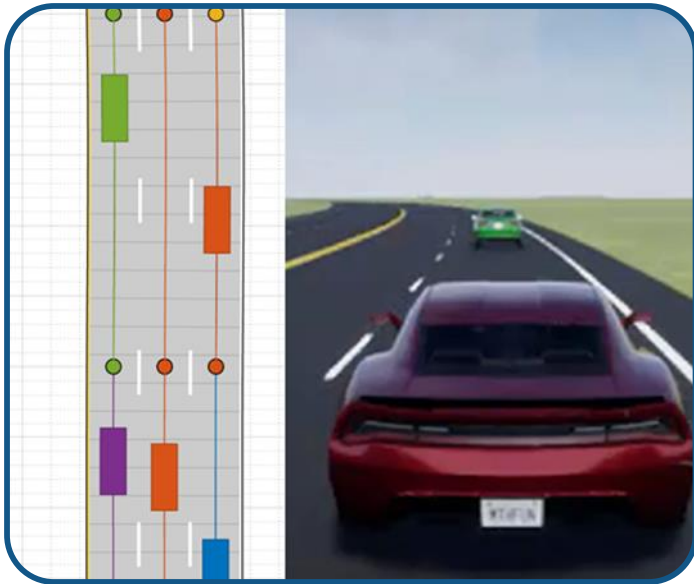
Capabilities of an Autonomous Vehicle



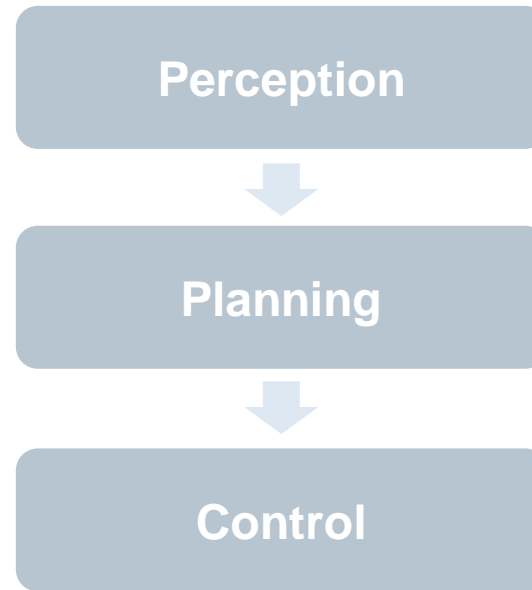
Capabilities of an Autonomous Vehicle



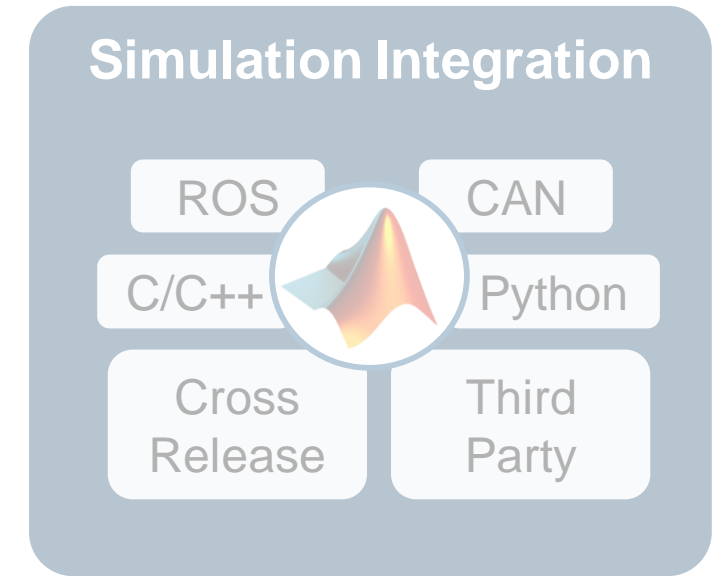
Some common questions from automated driving engineers



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

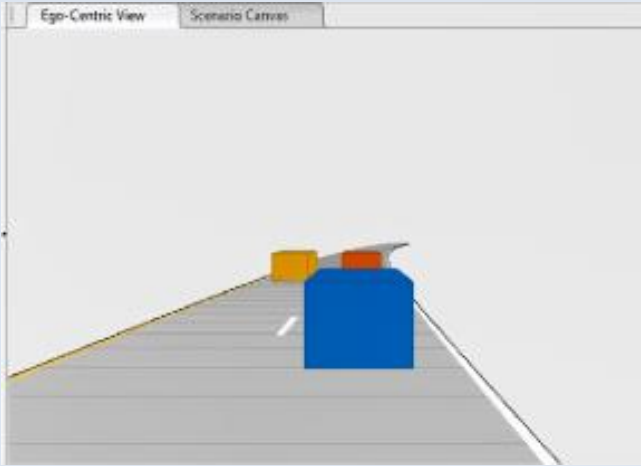


How can I **discover and design** in multiple domains?



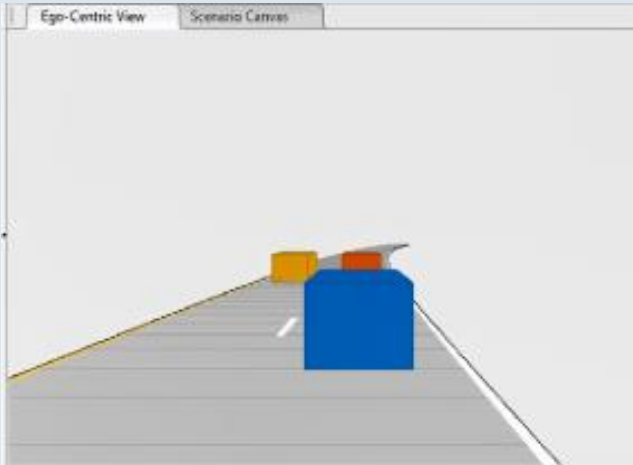
How can I **integrate** with other environments?

How can I design with virtual driving scenarios?

| | |
|------------------|---|
| <p>Scenes</p> | <p>Cuboid</p>  |
| <p>Testing</p> | <p>Controls, sensor fusion, planning</p> |
| <p>Authoring</p> | <p>Driving Scenario Designer App Programmatic API (drivingScenario)</p> |
| <p>Sensing</p> | <p>Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list)</p> |

How can I design with virtual driving scenarios?

| Scenes | Cuboid | 3D Simulation |
|-----------|--|--|
| Testing | Controls, sensor fusion, planning | Controls, sensor fusion, planning, perception |
| Authoring | Driving Scenario Designer App Programmatic API (drivingScenario) | Unreal Engine Editor |
| Sensing | Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list) | Probabilistic radar (detection list) Monocular camera (image, labels, depth) Fisheye camera (image) Lidar (point cloud) |



Simulate controls with perception

Lane-Following Control with Monocular Camera Perception

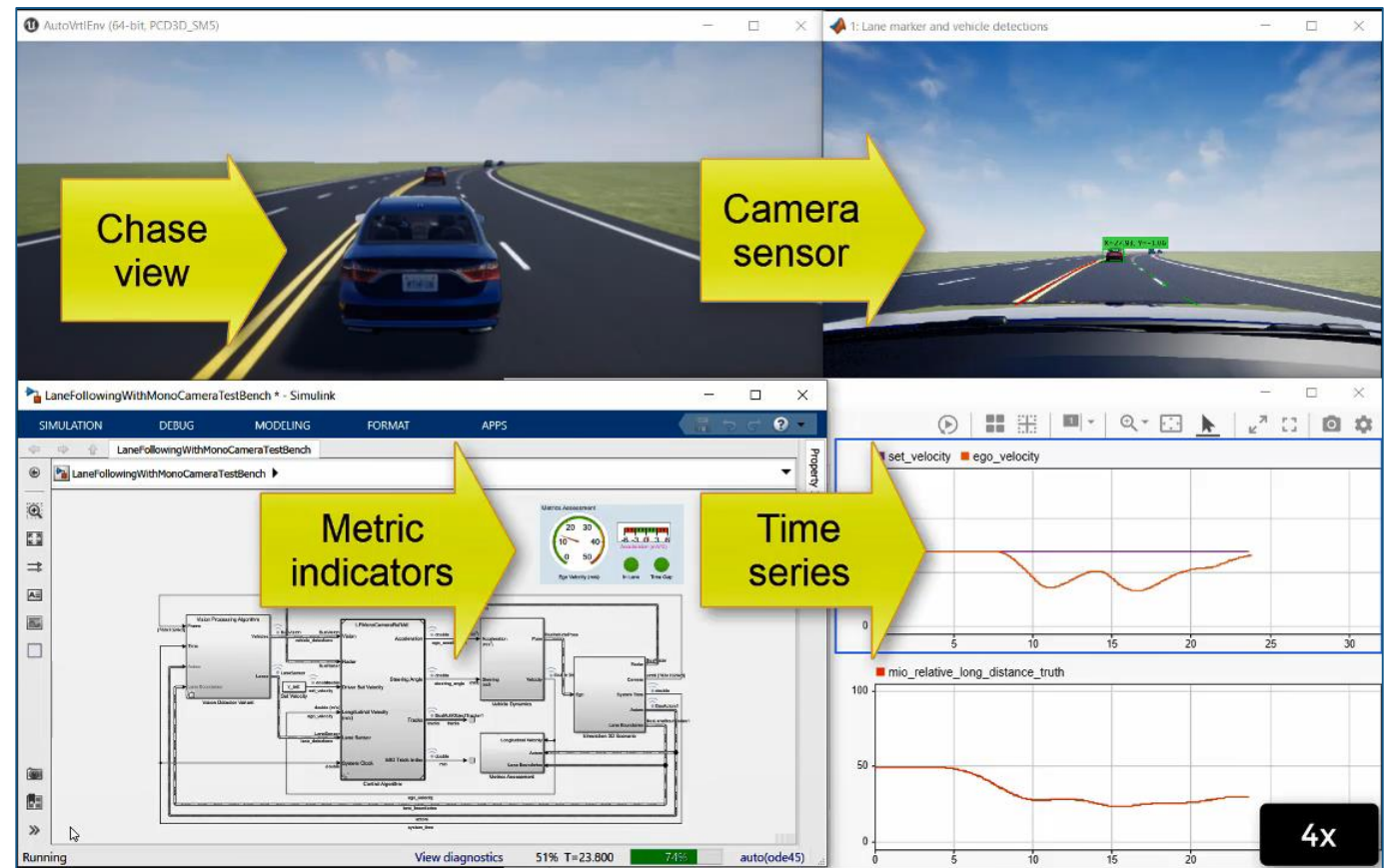
- Author target vehicle trajectories
- Synthesize monocular camera and probabilistic radar sensors
- Model lane following and spacing control in Simulink
- Model lane boundary and vehicle detectors in MATLAB code

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

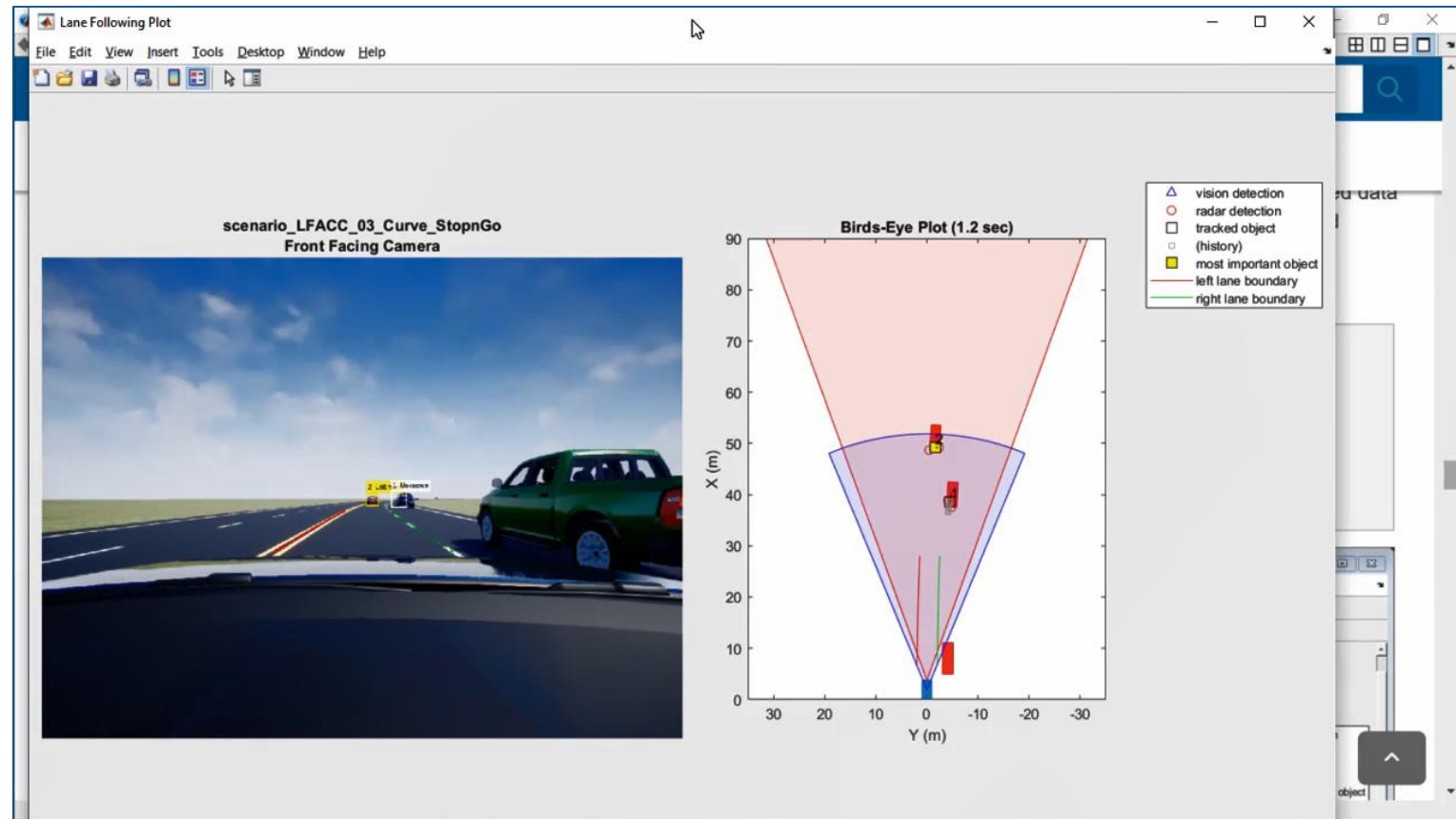
- Author target vehicle trajectories
- Synthesize monocular camera and probabilistic radar sensors
- Model lane following and spacing control in Simulink
- Model lane boundary and vehicle detectors in MATLAB code

Model Predictive Control Toolbox™

Automated Driving Toolbox™

Vehicle Dynamics Blockset™

Updated **R2019b**



How can I design with virtual driving scenarios?

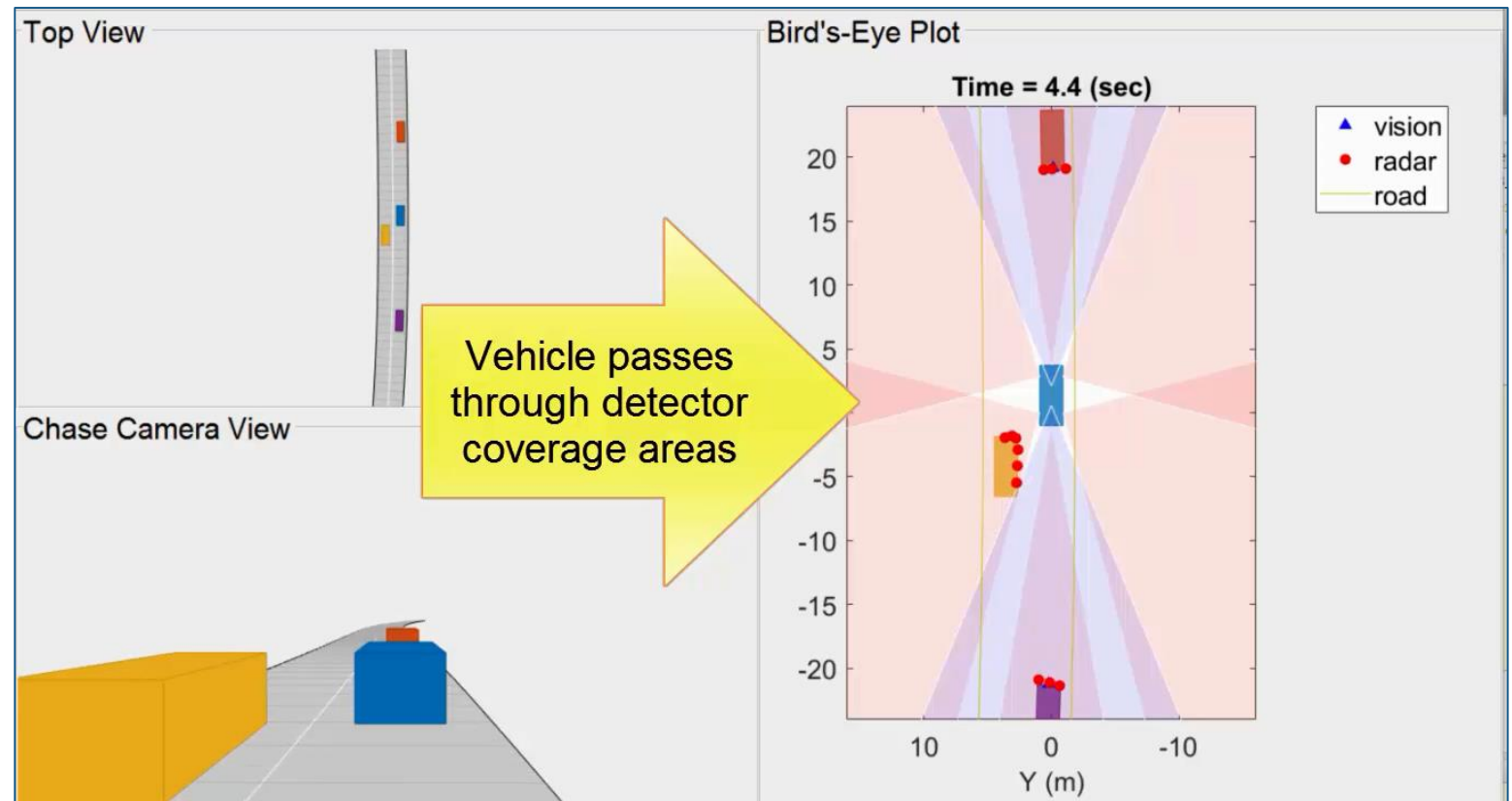
| Scenes | Cuboid | 3D Simulation |
|-----------|--|--|
| Testing | Controls, sensor fusion, planning | Controls, sensor fusion, planning, perception |
| Authoring | Driving Scenario Designer App Programmatic API (drivingScenario) | Unreal Engine Editor |
| Sensing | Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list) | 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™
R2017a



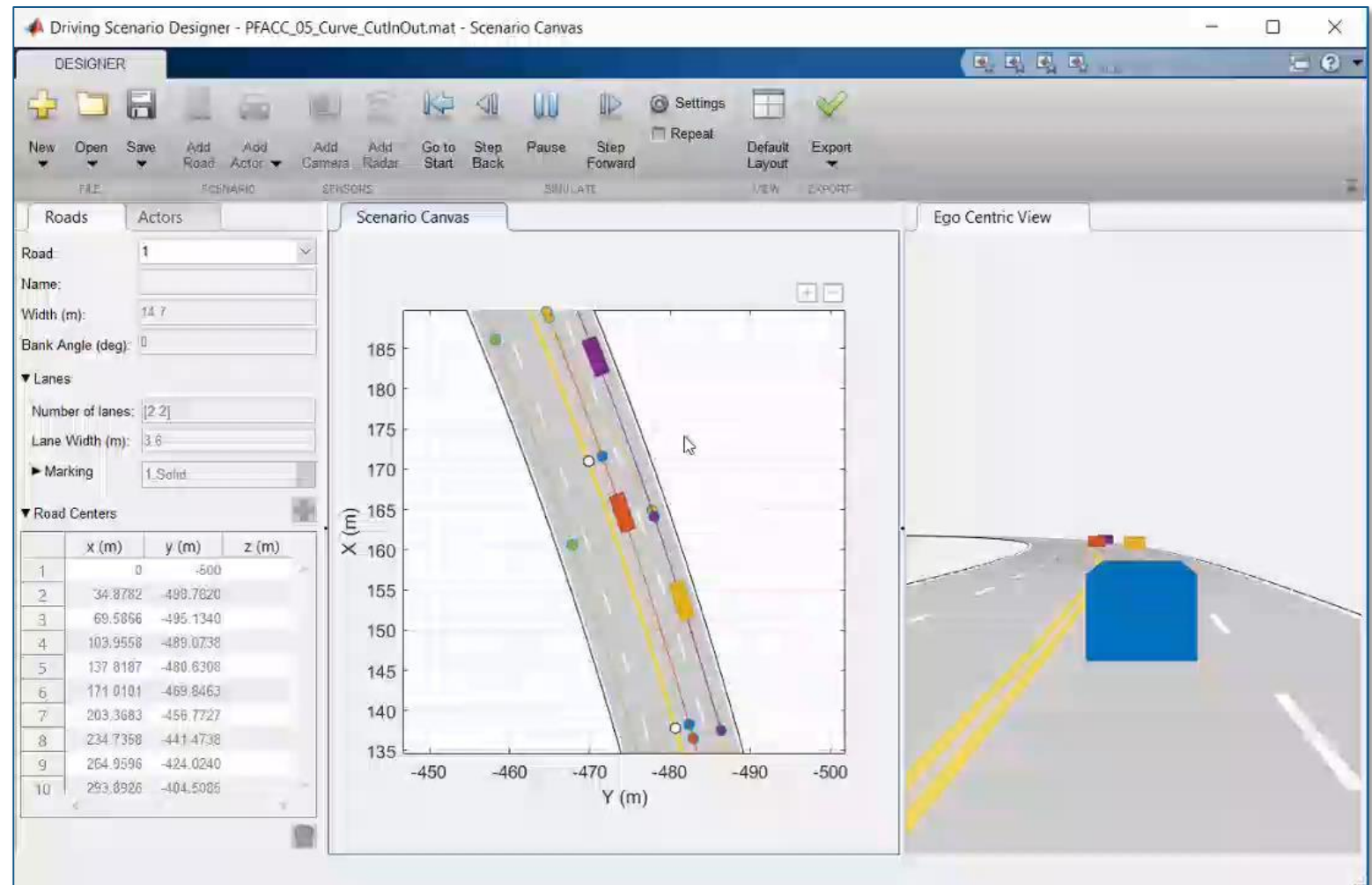
Graphically author driving scenarios

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

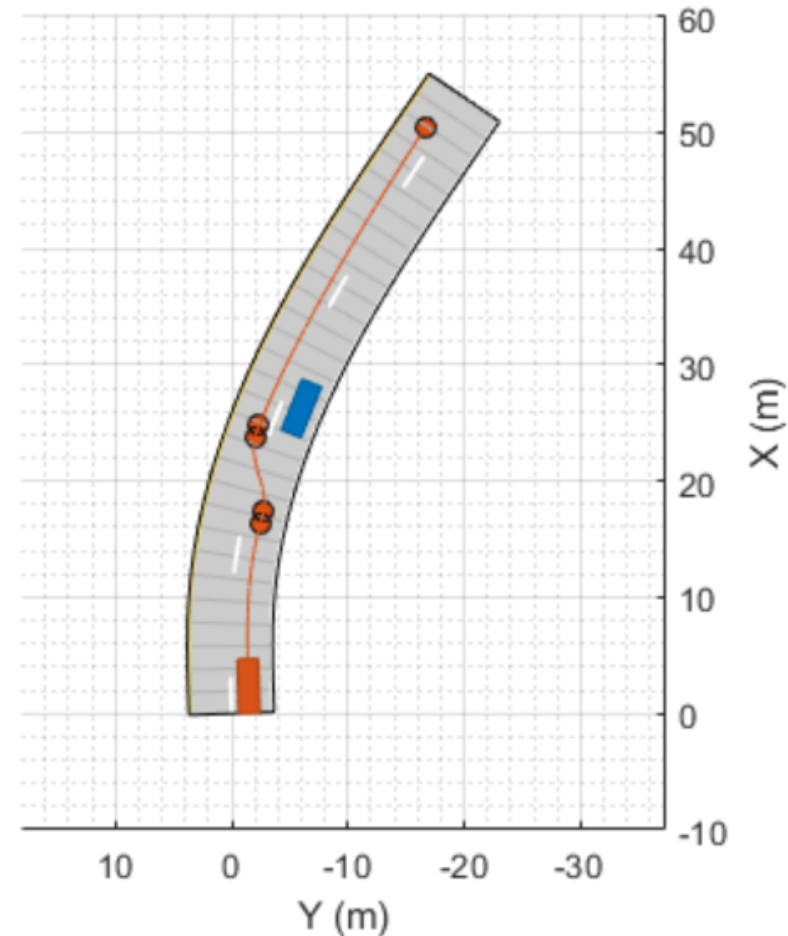
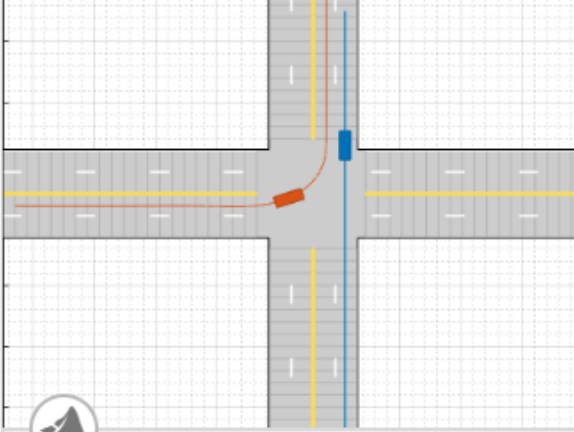


Programmatically author driving scenarios

```

1  scenario = drivingScenario;
2
3  road( scenario, [0 0; 10 0; 53 -20], ...
4         'lanes', lanespec(2) );
5
6  plot( scenario, 'Waypoints', 'on' );
7
8  idleCar = vehicle( scenario, ...
9                   'Position', [25 -5.5 0], ...
10                  'Yaw', -22 );
11
12  passingCar = vehicle( scenario, 'ClassID', 1 );
13
14  waypoints = [1 -1.5; 16.36 -2.5; 17.35 -2.765; ...
15              23.83 -2.01; 24.9 -2.4; 50.5 -16.7];
16
17  velocity = 15;
18
19  trajectory( passingCar, waypoints, velocity );

```

Create Driving Scenario Variations Programmatically

Programmatically create variations of a driving scenario that was built using the Driving Scenario Designer app.

[Open Live Script](#)

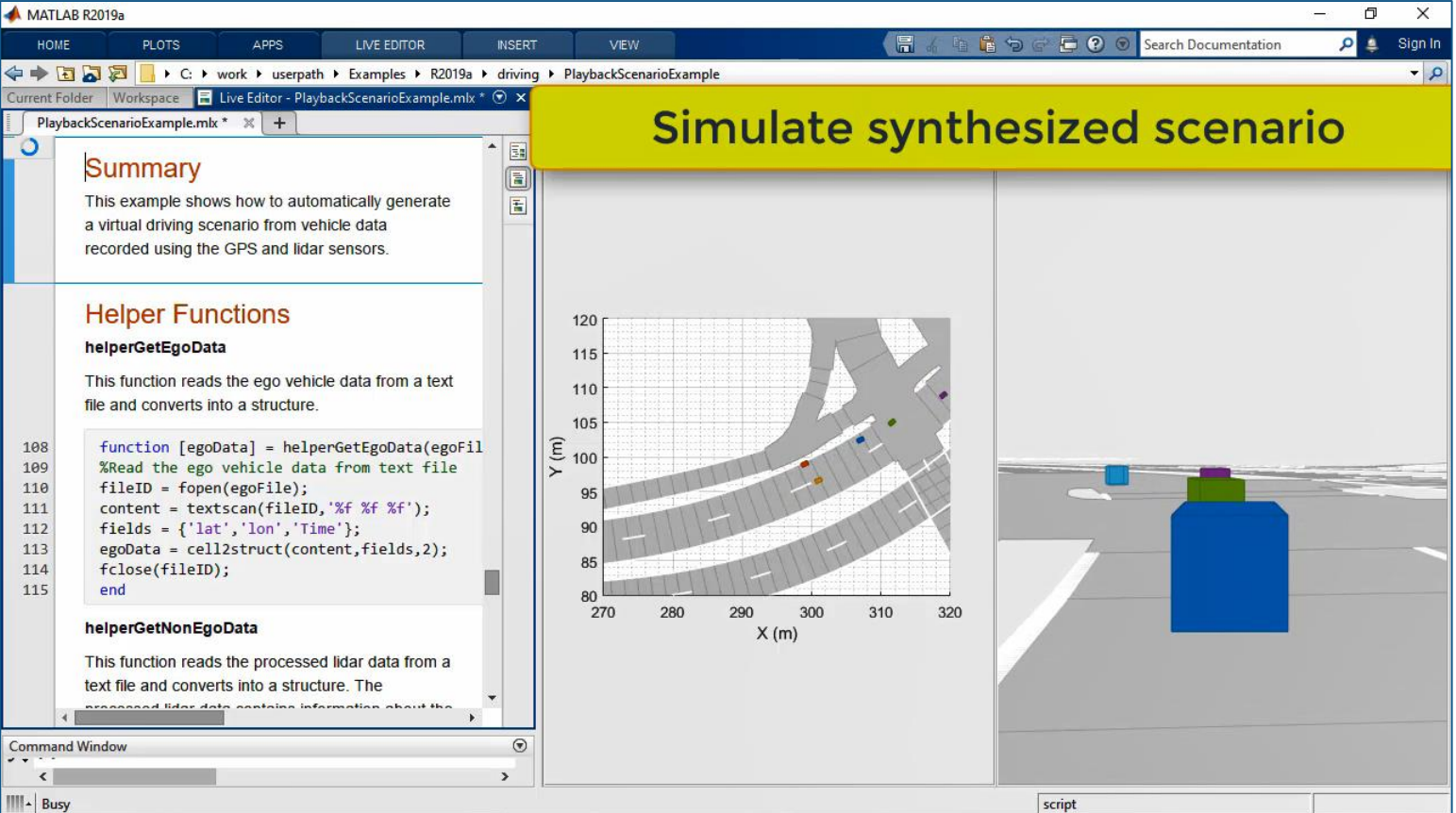
Synthesize driving scenarios from recorded data

Scenario Generation from Recorded Vehicle Data

- Visualize video
- Import OpenDRIVE roads
- Import GPS
- Import object lists

Automated Driving Toolbox™

R2019a



Simulate synthesized scenario

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

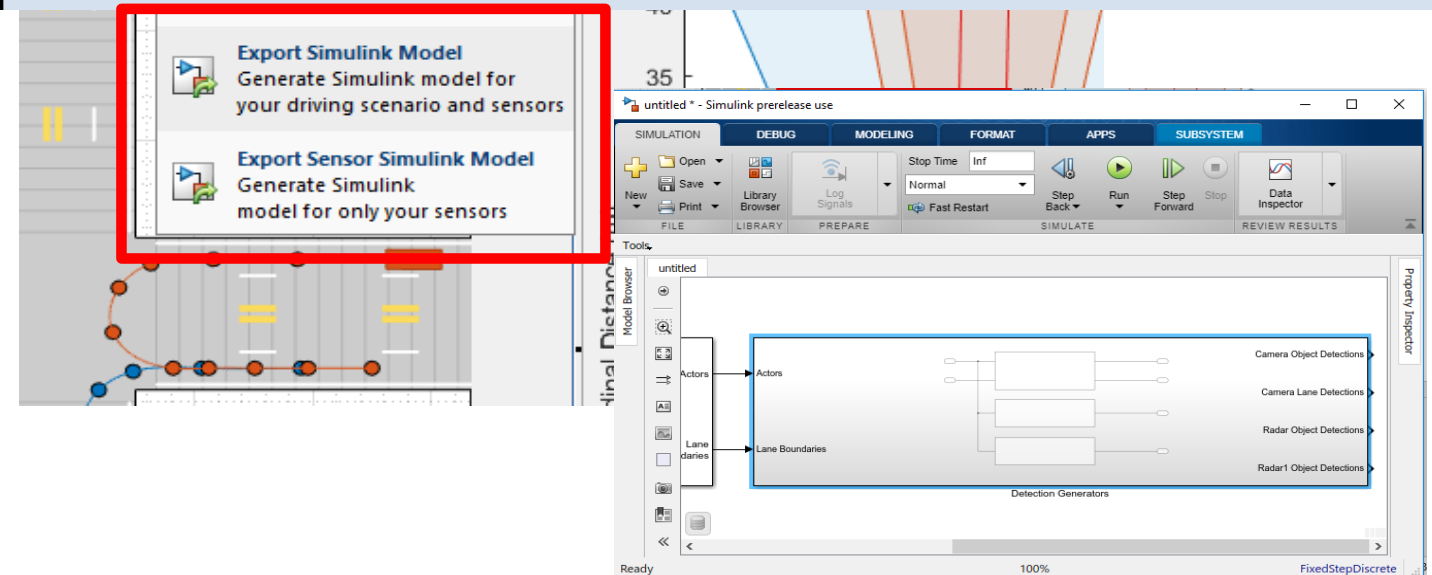
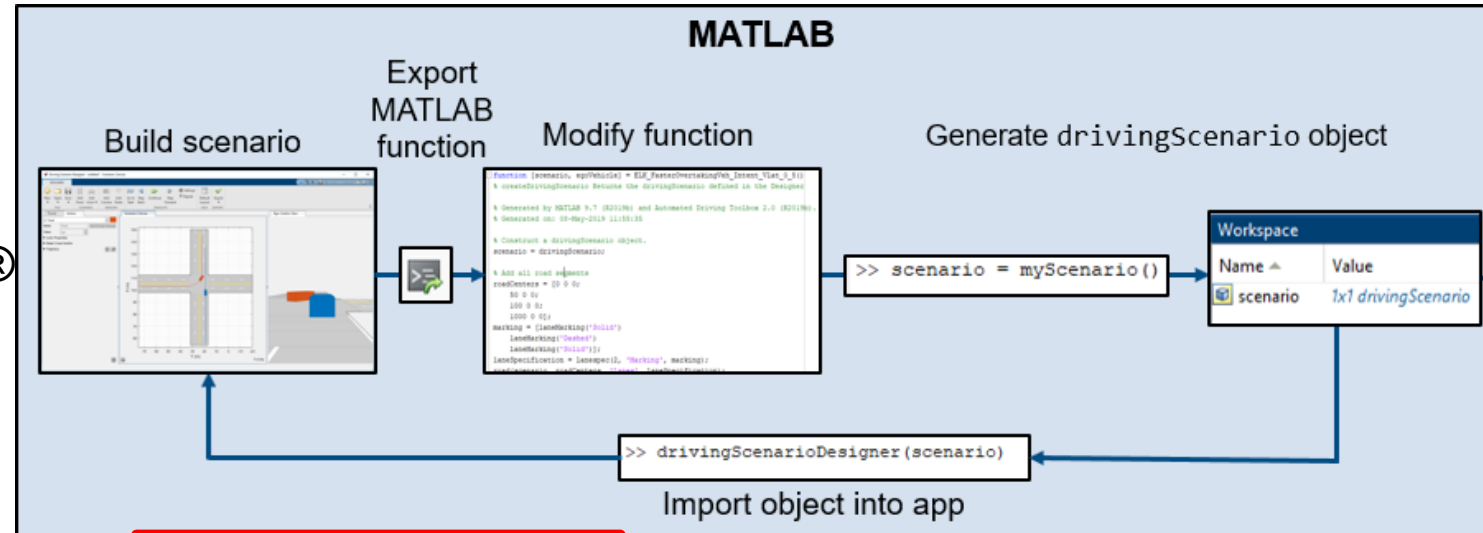
Command Window
Busy

script

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

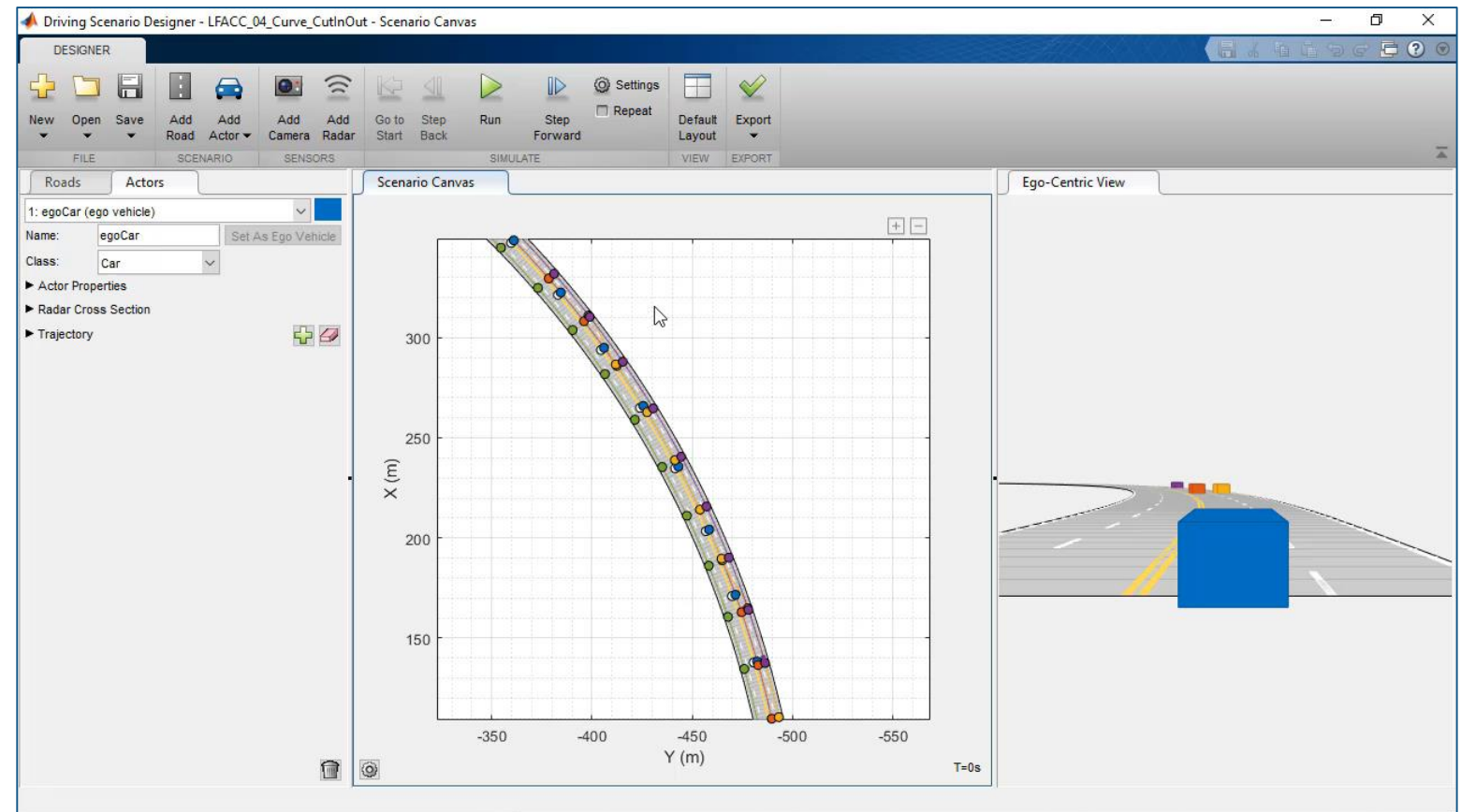
- Integrate scenario into system
- Design lateral (lane keeping) and longitudinal (lane spacing) model predictive controllers
- Visualize sensors and tracks
- Generate C/C++ code
- Test with software in the loop (SIL) simulation

Model Predictive Control Toolbox™

Automated Driving Toolbox™

Embedded Coder®

R2018b



Design lateral and longitudinal controls

Lane Following Control with Sensor Fusion

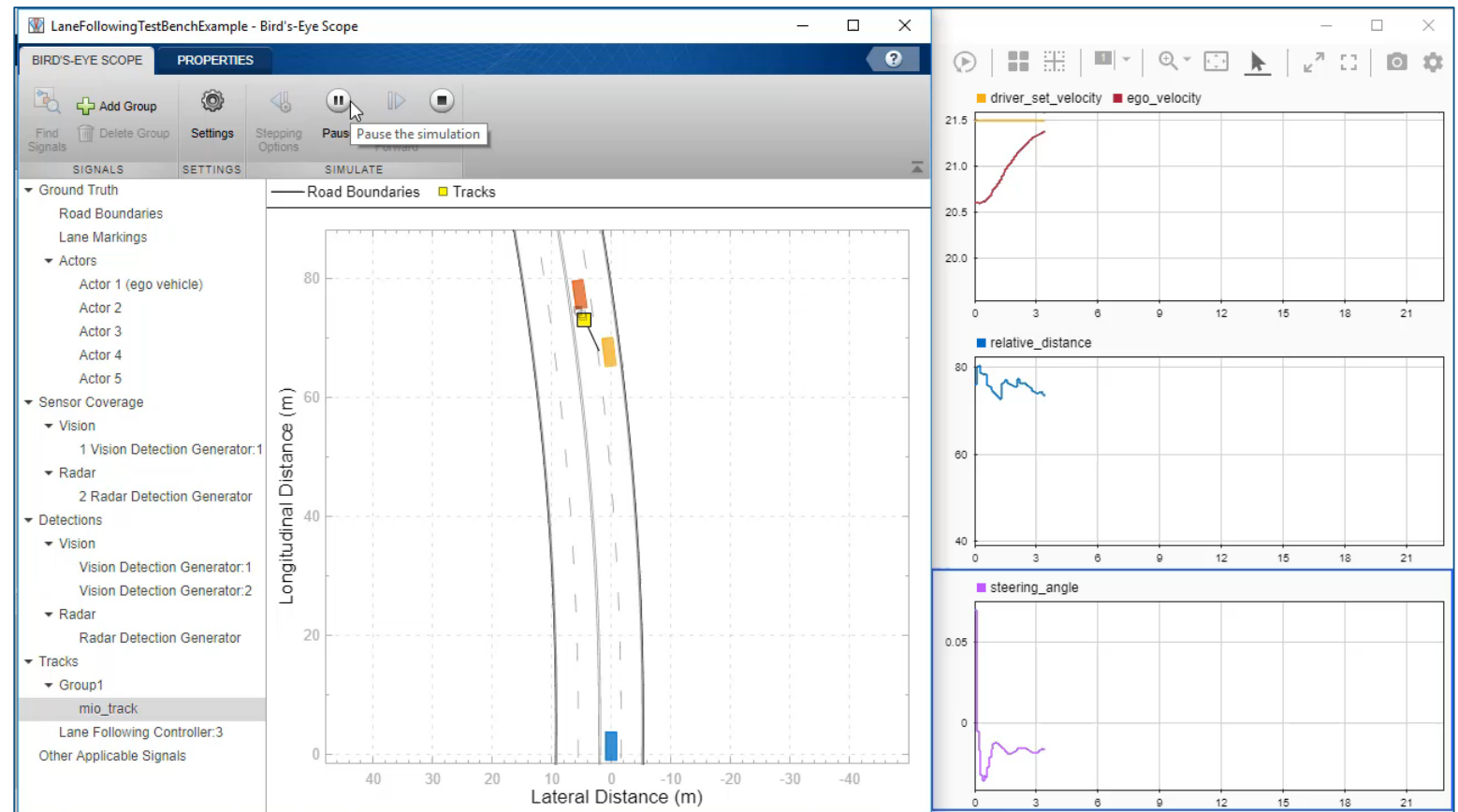
- Integrate scenario into system
- Design lateral (lane keeping) and longitudinal (lane spacing) model predictive controllers
- Visualize sensors and tracks
- Generate C/C++ code
- Test with software in the loop (SIL) simulation

Model Predictive Control Toolbox™

Automated Driving Toolbox™

Embedded Coder®

R2018b



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, with callouts pointing to various sections:

- Requirements link:** Points to the 'REQUIREMENTS*' section, which contains a link to 'scenarioId #1: ACC_ISO_TargetDiscriminationTest (LaneFollowingTestRequirements#1)'.
- Simulink Model:** Points to the 'SYSTEM UNDER TEST*' section, where the 'Model' is set to 'LaneFollowingTestBenchExample'.
- Define scenario ID and data initialization:** Points to the 'CALLBACKS*' section, specifically to the 'POST-LOAD*' callback which contains the code:


```
1 scenarioId = 1;
2 helperLFSetup;
```
- Plot the results:** Points to the 'CLEANUP*' section, which contains the code:


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

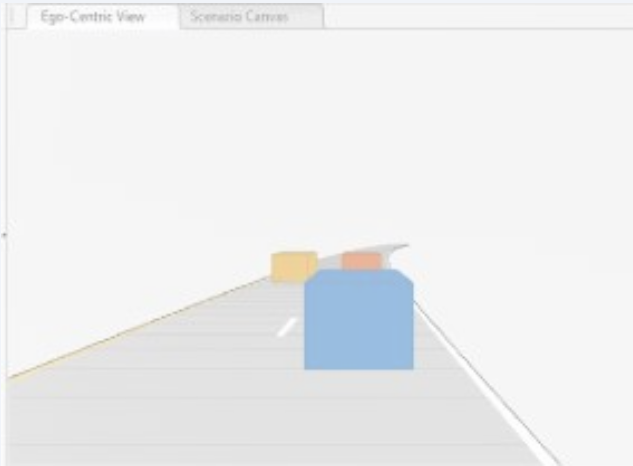
At the bottom left of the interface, a 'PROPERTY' table is visible:

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

How can I design with virtual driving scenarios?

| Scenes | Cuboid | 3D Simulation |
|-----------|--|--|
| Testing | Controls, sensor fusion, planning | Controls, sensor fusion, planning, perception |
| Authoring | Driving Scenario Designer App Programmatic API (drivingScenario) | Unreal Engine Editor |
| Sensing | Probabilistic radar (detection list) Probabilistic vision (detection list) Probabilistic lane (detection list) | Probabilistic radar (detection list) Monocular camera (image, labels, depth) Fisheye camera (image) Lidar (point cloud) |

Cuboid



3D Simulation



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



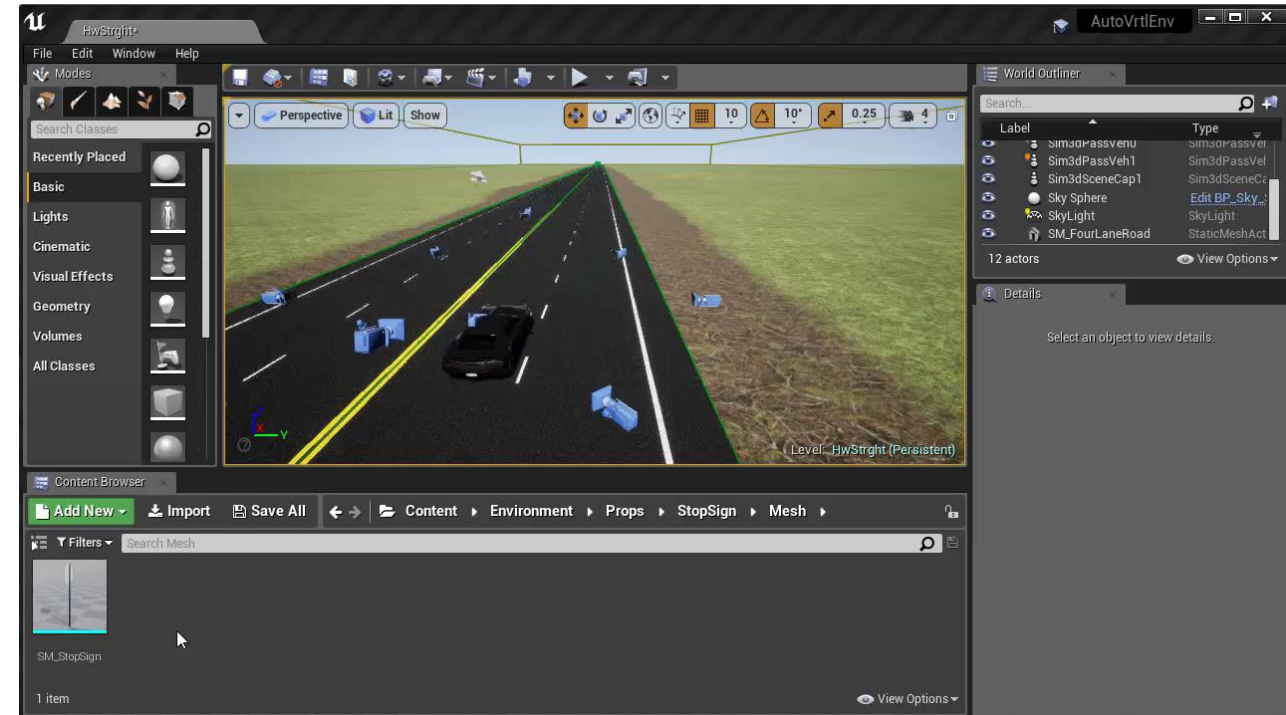
Automated Driving Toolbox™

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



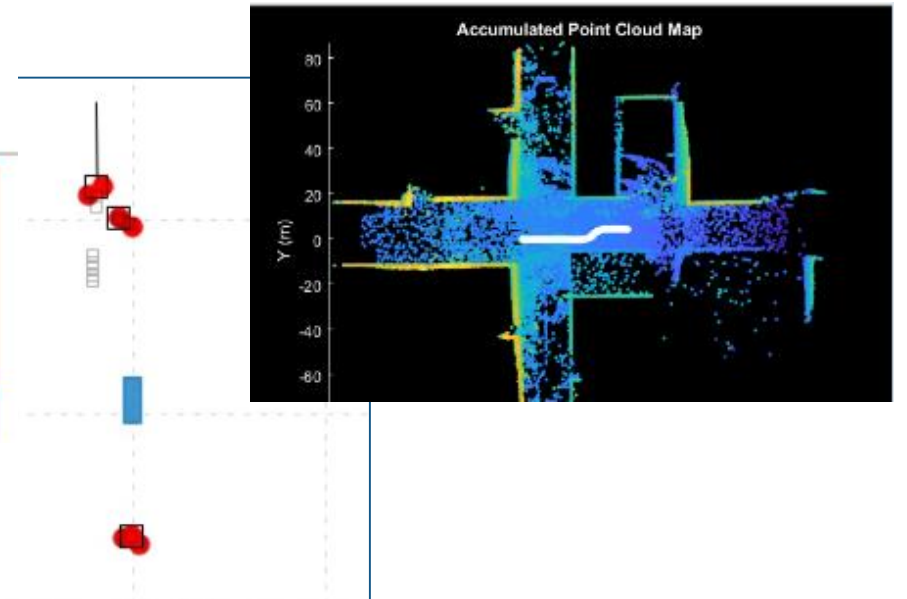
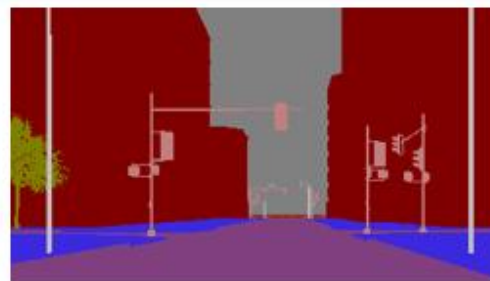
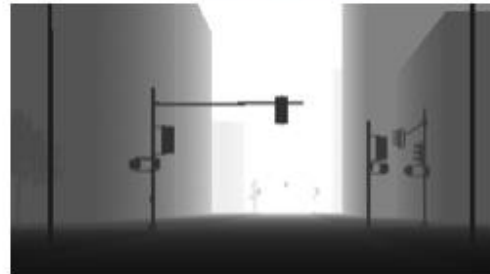
Vehicle Dynamics Blockset™

R2019b

Model sensors in 3D simulation environment

3D Simulation for Automated Driving

- Monocular camera
- Fisheye camera
- Lidar
- Probabilistic radar



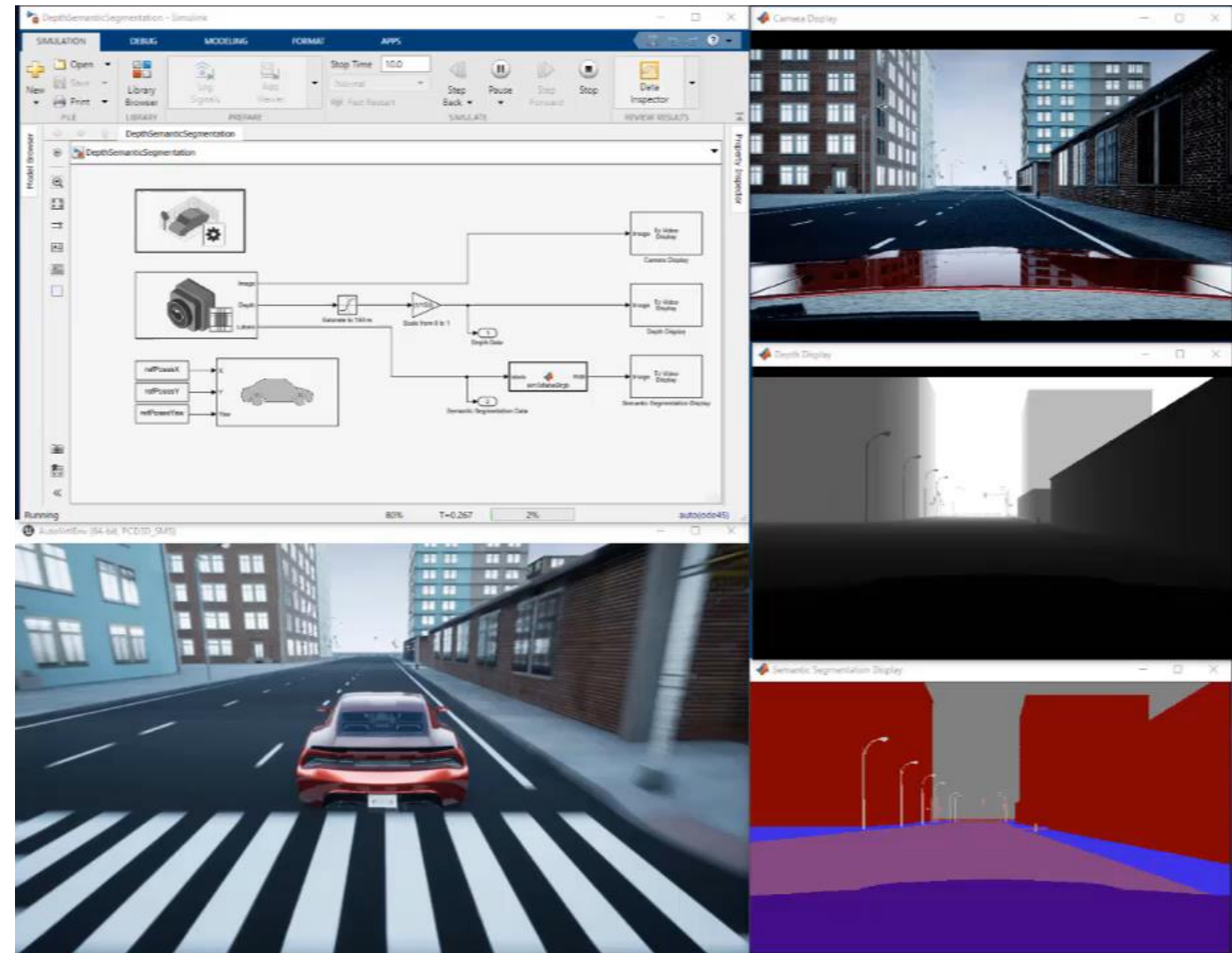
Automated Driving Toolbox™

R2019b

Synthesize monocular camera sensor data

Visualize Depth and Semantic Segmentation Data in 3D Environment

- Synthesize RGB image
- Synthesize depth map
- Synthesize semantic segmentation



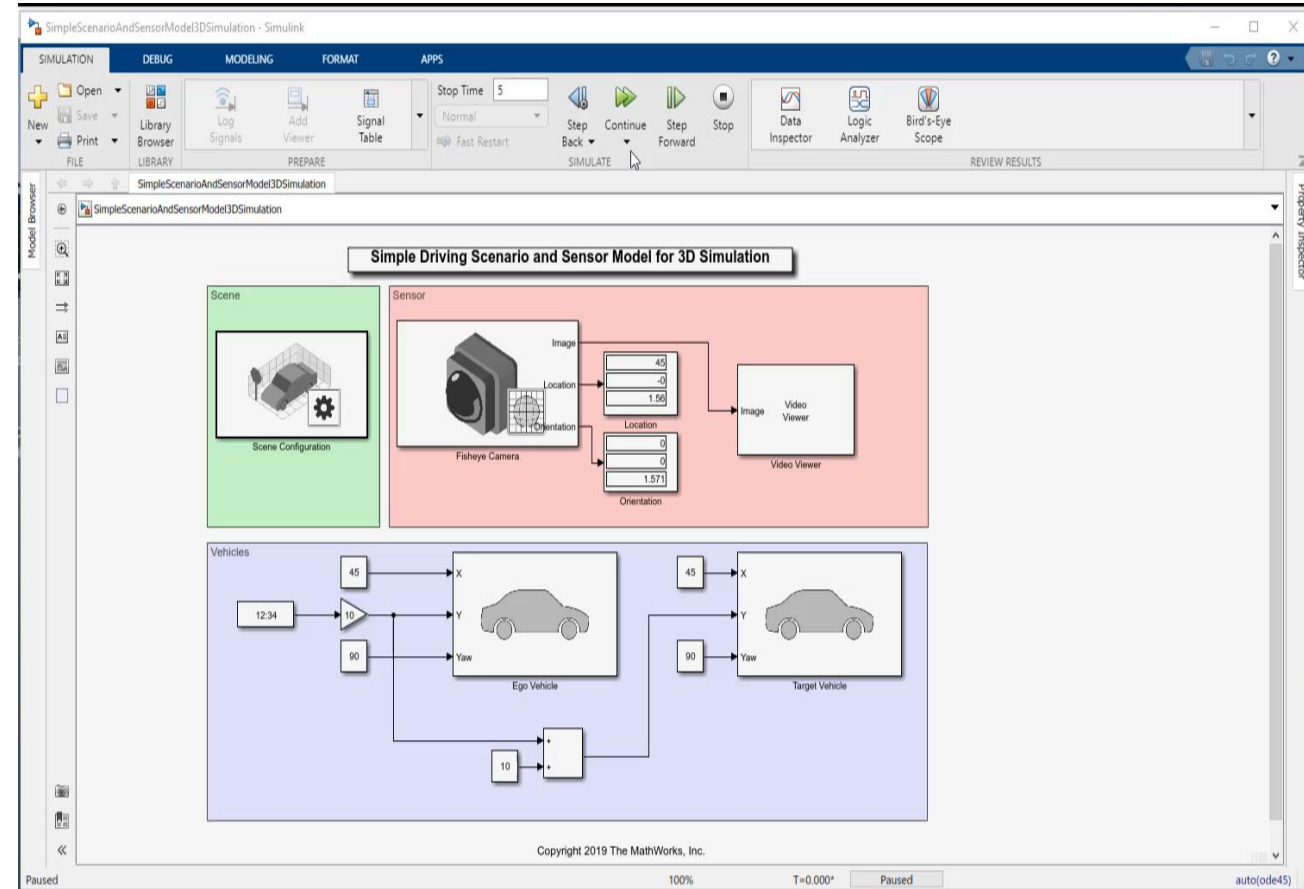
Automated Driving Toolbox™

R2019b

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™

R2019b

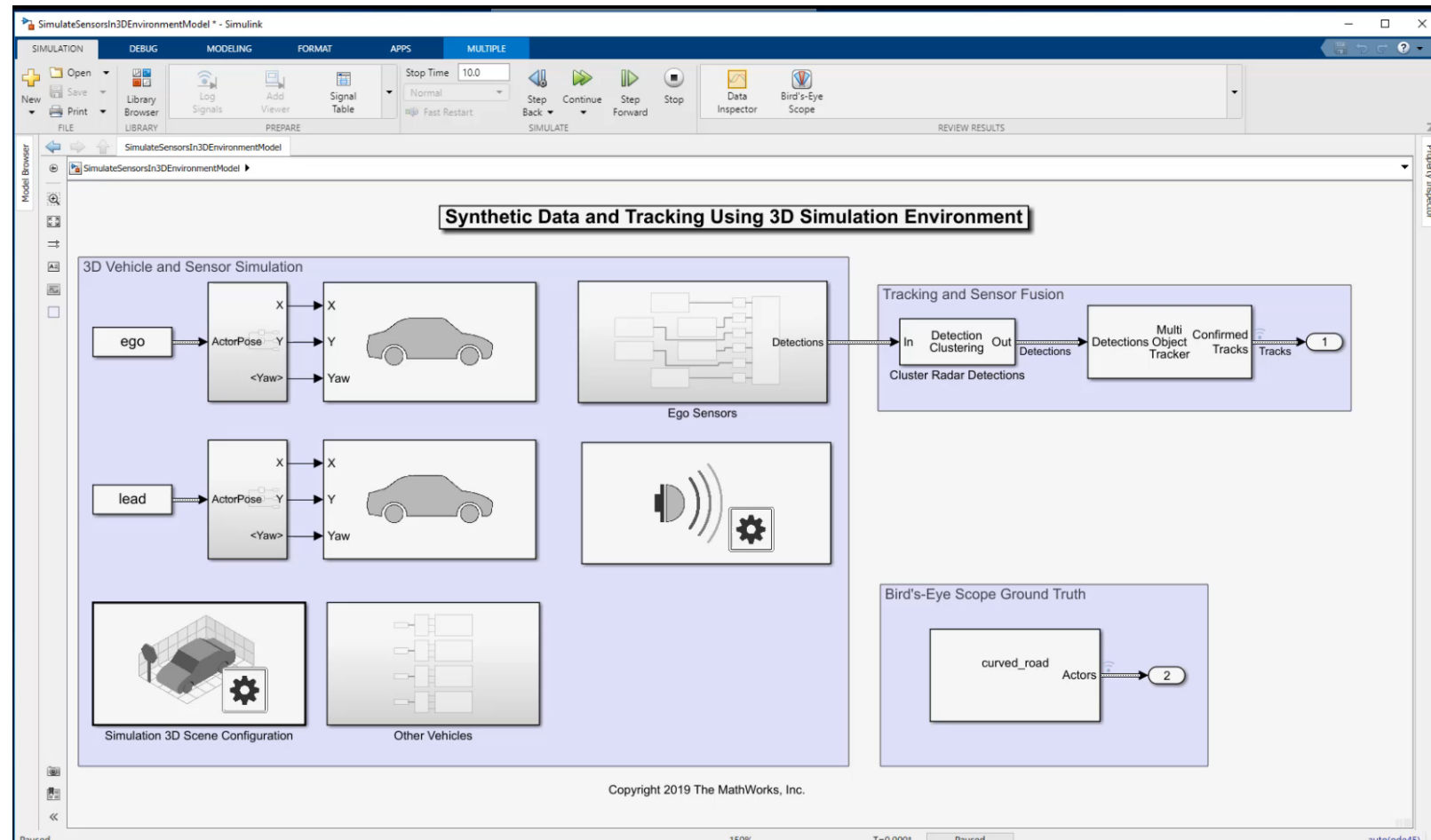
Synthesize radar sensor data

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

The screenshot displays the Simulink environment with a 3D visualization of a red car on a road. The interface includes a menu bar (SIMULATION, DEBUG, MODELING, FORMAT, APPS), a toolbar, and a project browser. A 'Vehicle Position' window shows a 2D plot of the car's position with 'X Distance [m]' on the y-axis (ranging from -10 to 10) and 'X' on the x-axis (ranging from -5 to 10). The car is represented by a small icon with a green arrow pointing upwards.

Below the plot, a data exchange diagram is shown. It features a 'ReadMsg' block with a matrix $\begin{bmatrix} 0 & 5 & 4 \\ 2 & 2 & 3 \\ 5 & 8 & 1 \end{bmatrix}$ and a 'WriteMsg' block with the same matrix. An 'int32' block is also present, connected to the 'WriteMsg' block. The diagram is connected to a list of variables: RearLeft Roll, RearRight Roll, FrontLeft Yaw, FrontRight Yaw, RearLeft Yaw, and RearRight Yaw.

Vehicle Dynamics Blockset™

R2019b

New Examples for 3D Simulation in Automated Driving Toolbox

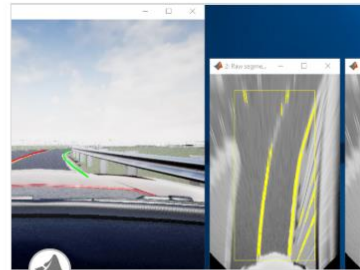
Unreal Engine Driving Scenario Simulation



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.

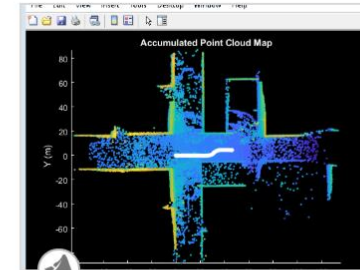
[Open Script](#)



Visualize Automated Parking Valet Using 3D Simulation

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

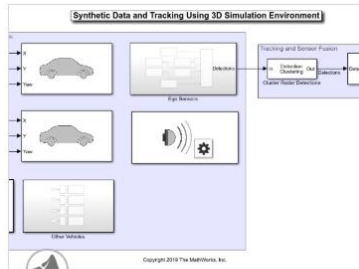
[Open Script](#)



Simulate Lidar Sensor Perception Algorithm

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

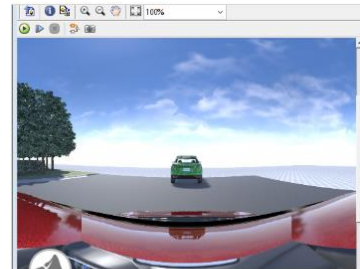
[Open Script](#)



Simulate Radar Sensors in 3D Environment

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

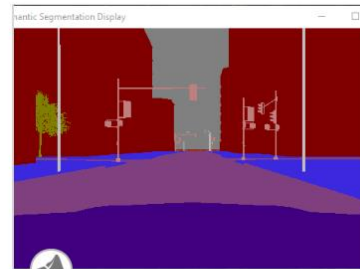
[Open Model](#)



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](#)

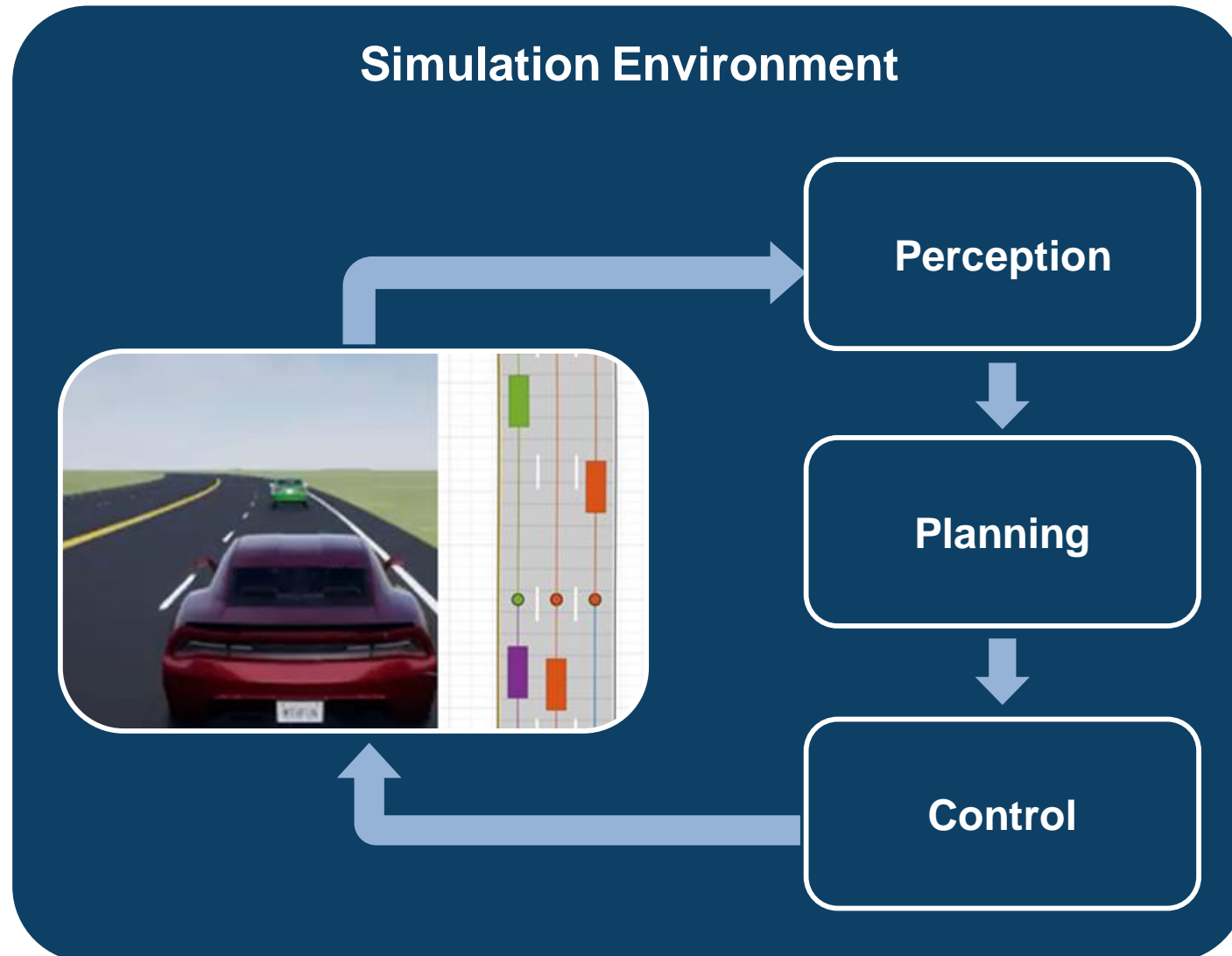


Visualize Depth and Semantic Segmentation Data in 3D Environment

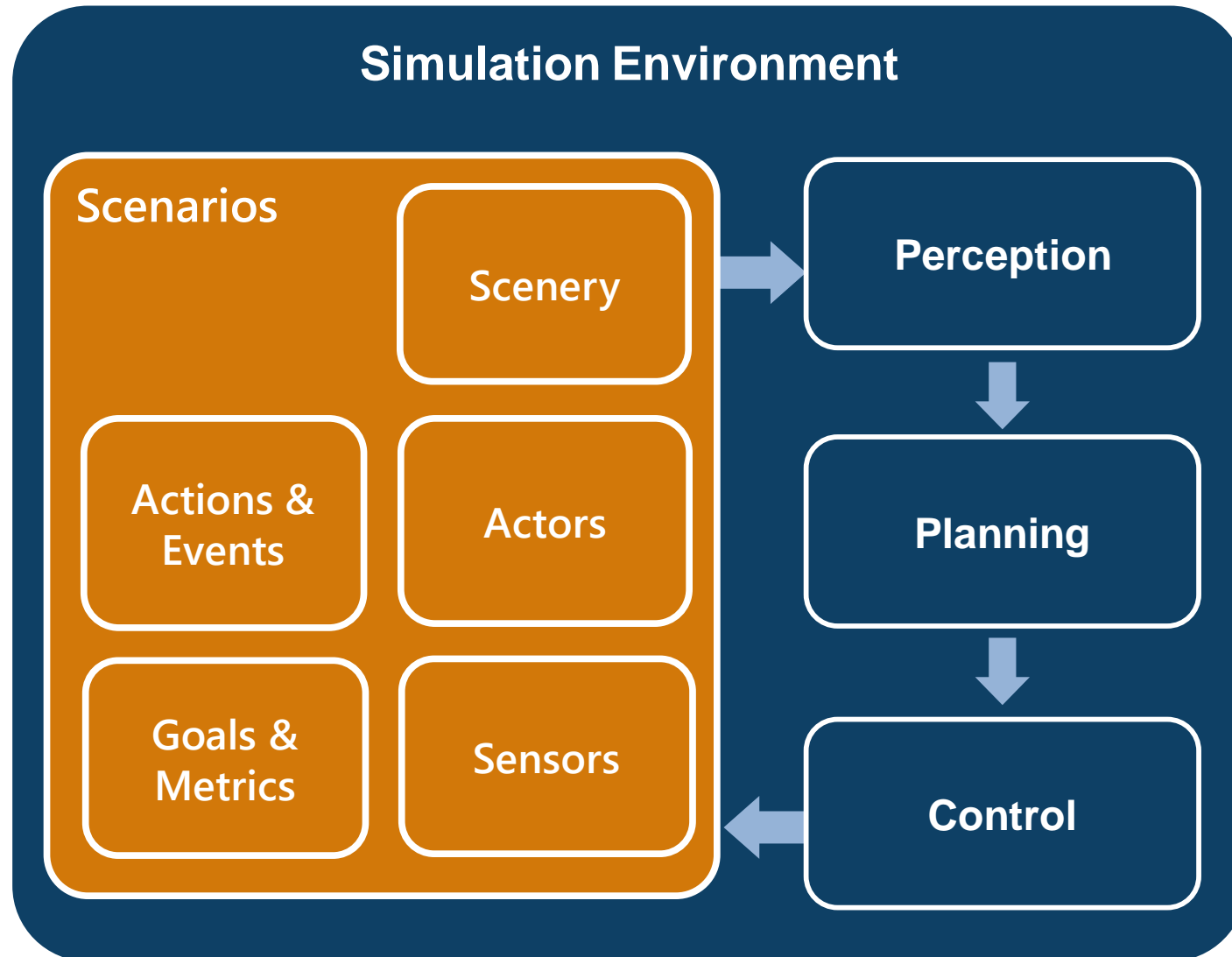
Visualize depth and semantic segmentation data captured from a camera sensor in a 3D simulation environment.

[Open Model](#)

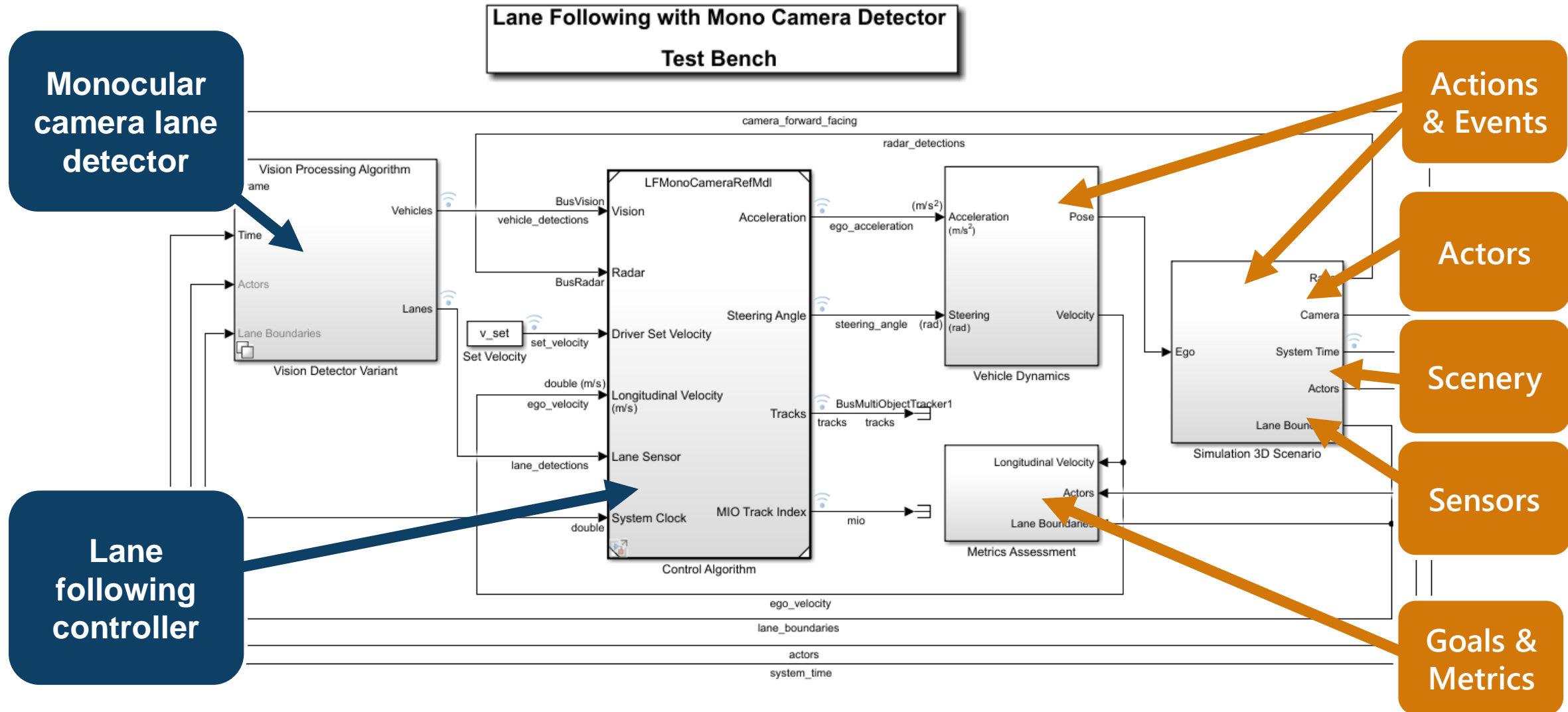
Simulating automated driving systems with MATLAB and Simulink



Simulating automated driving systems with MATLAB and Simulink



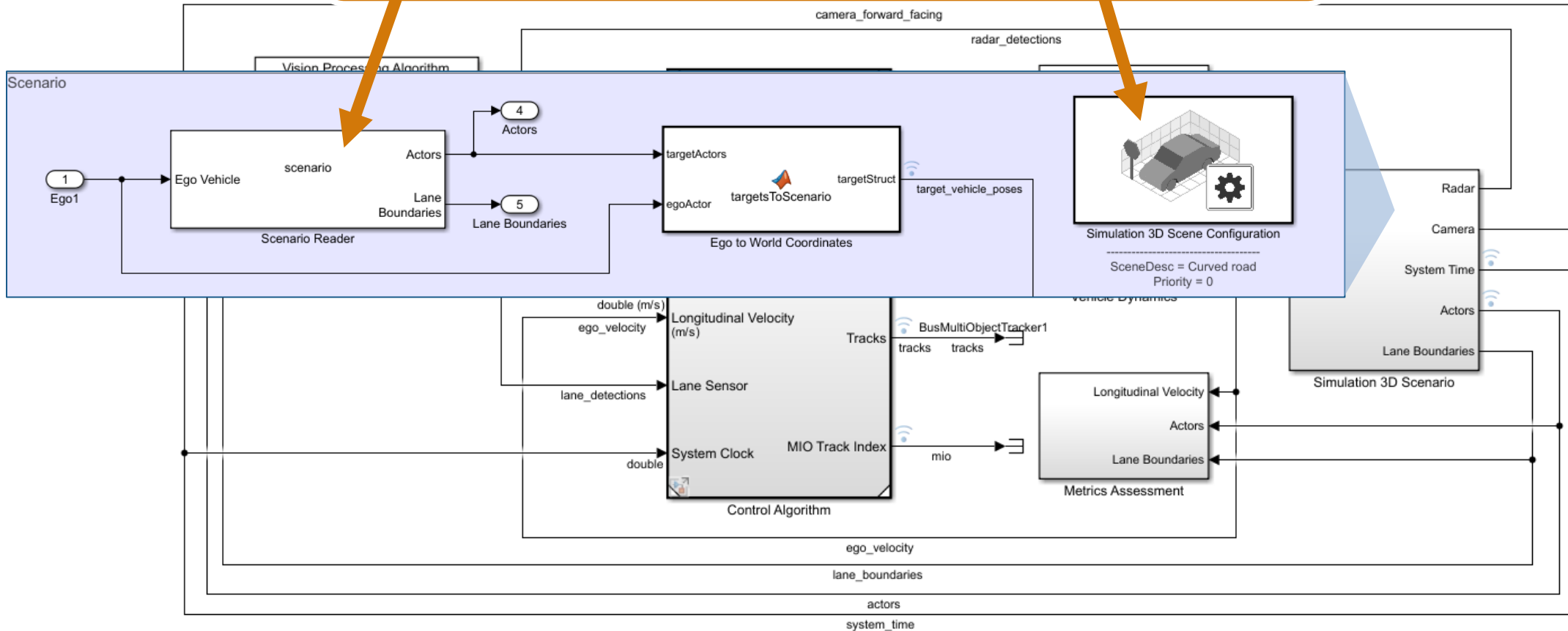
Integrate components and model scenarios



Specify equivalent 3D Simulation scenery

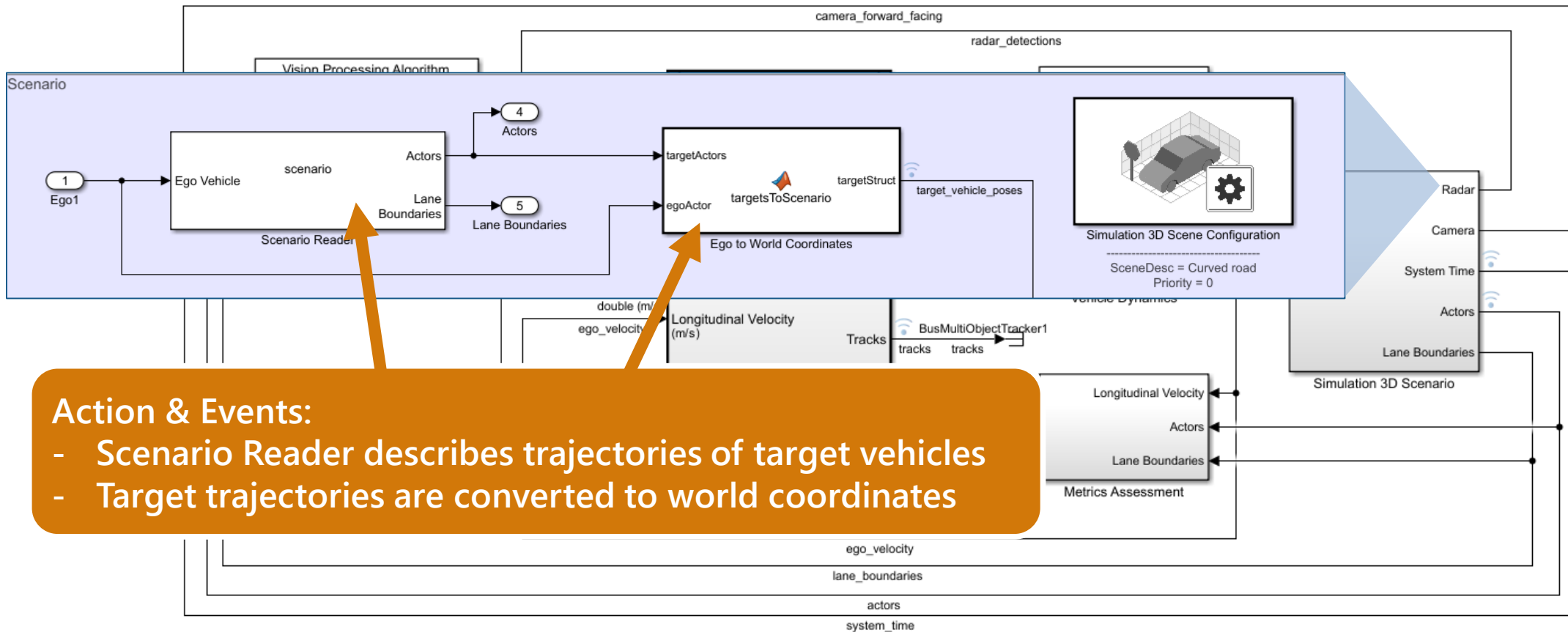
Scenery:
 - Equivalent straight and curved roads in Simulation 3D Scene Configuration and Scenario Reader

- Supported Scenery:
- ✓ Straight road
 - ✓ Curved road segment
 - ✓ Curved road (not exposed in example, but available)



Specify 3D Simulation actor trajectories

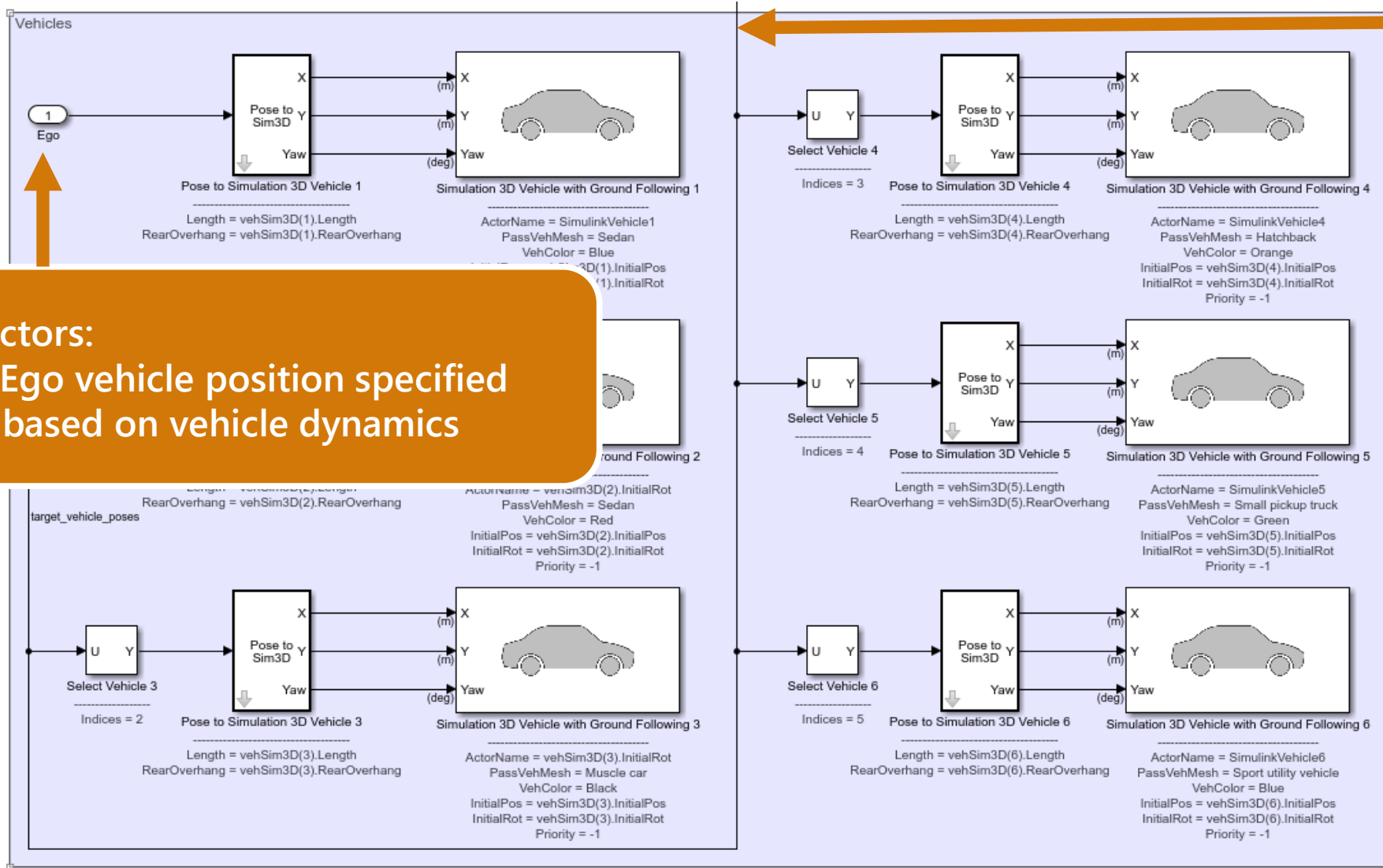
Lane Following with Mono Camera Detector Test Bench



Action & Events:

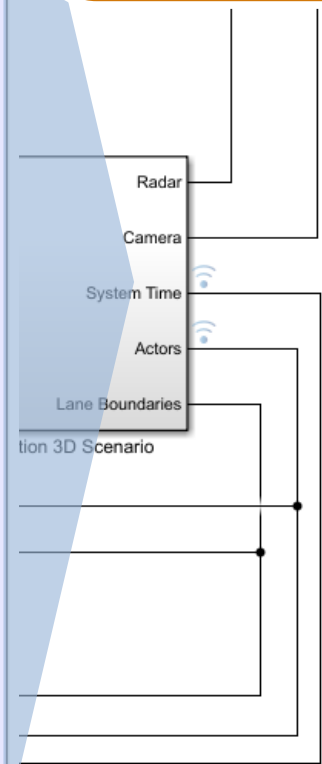
- Scenario Reader describes trajectories of target vehicles
- Target trajectories are converted to world coordinates

Specify 3D Simulation vehicles

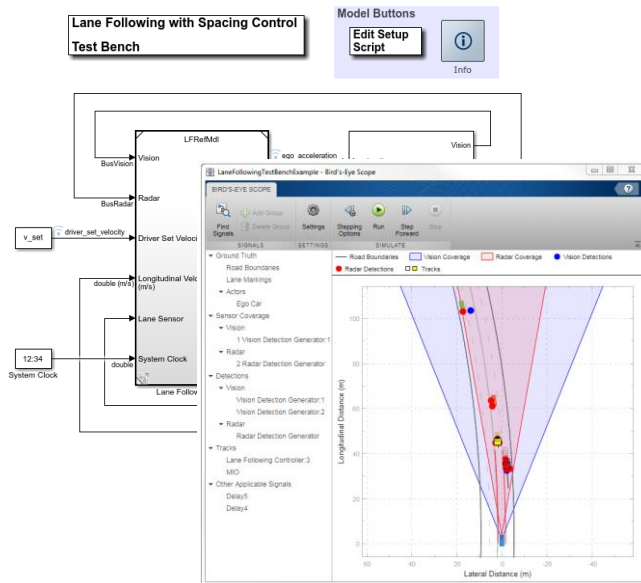


Actors:
- Ego vehicle position specified based on vehicle dynamics

Actors:
- Target vehicle positions specified from Scenario Reader block

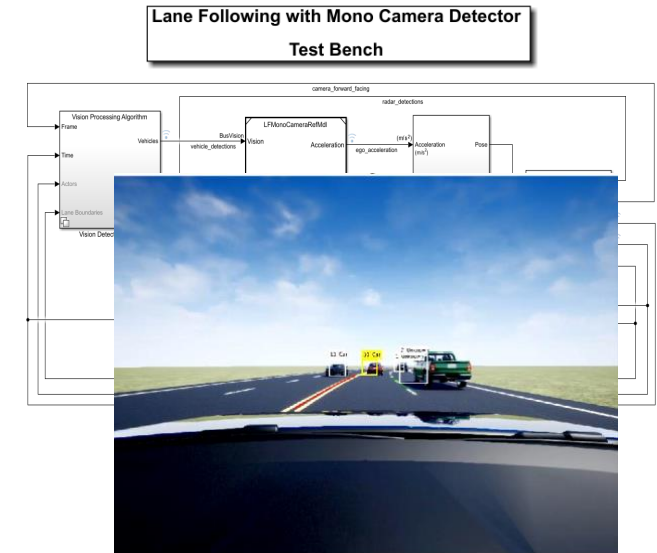


Synthesize scenarios to test your design



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 Sensor Fusion

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

R2018b

Design of Lane Marker Detector in 3D Simulation Environment

Automated Driving Toolbox™

R2019b

Lane-Following Control with Monocular Camera Perception

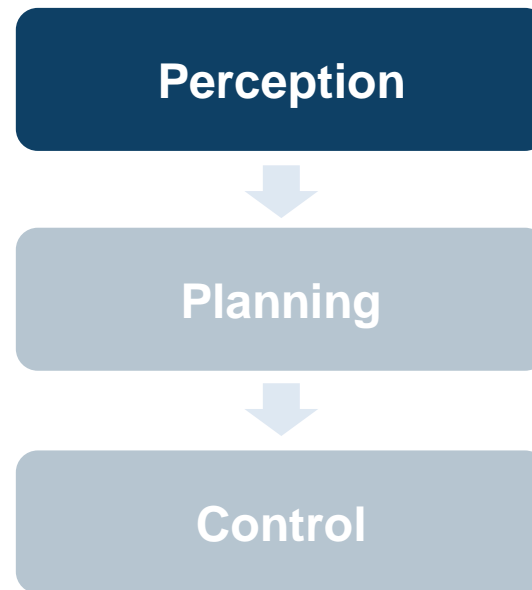
Model Predictive Control Toolbox™
Automated Driving Toolbox™
Vehicle Dynamics Blockset™

Updated **R2019b**

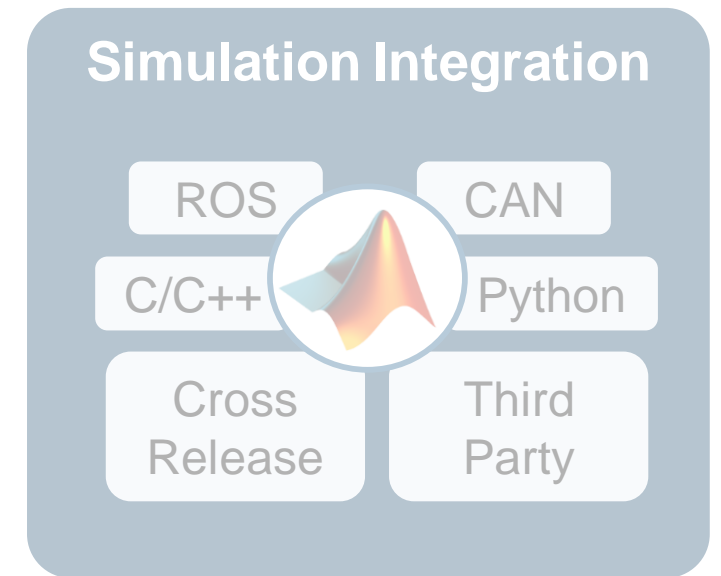
Some common questions from automated driving engineers



How can I
synthesize scenarios
to test my designs?



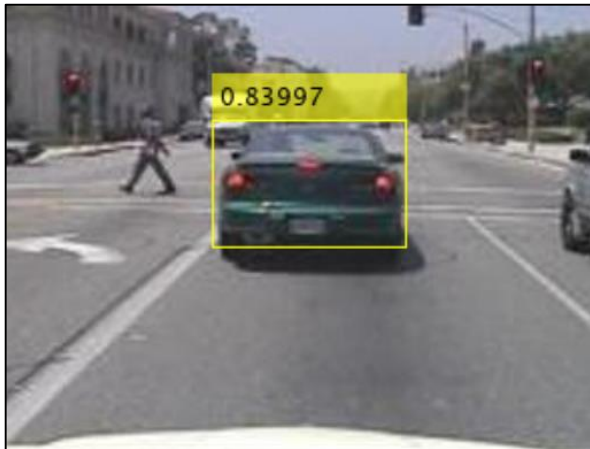
How can I
discover and design
in multiple domains?



How can I
integrate
with other environments?

Design camera, lidar, and radar perception algorithms

Detect vehicle with camera

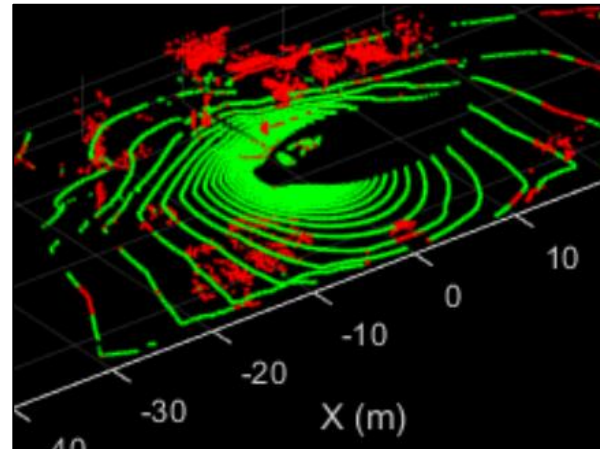


[Object Detection Using YOLO v2 Deep Learning](#)

*Computer Vision Toolbox™
Deep Learning Toolbox™*

R2019a

Detect ground with lidar

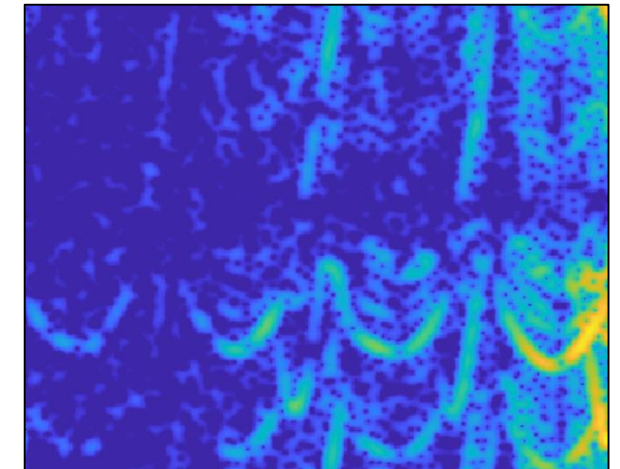


[Segment Ground Points from Organized Lidar Data](#)

Computer Vision Toolbox™

R2018b

Detect pedestrian with radar

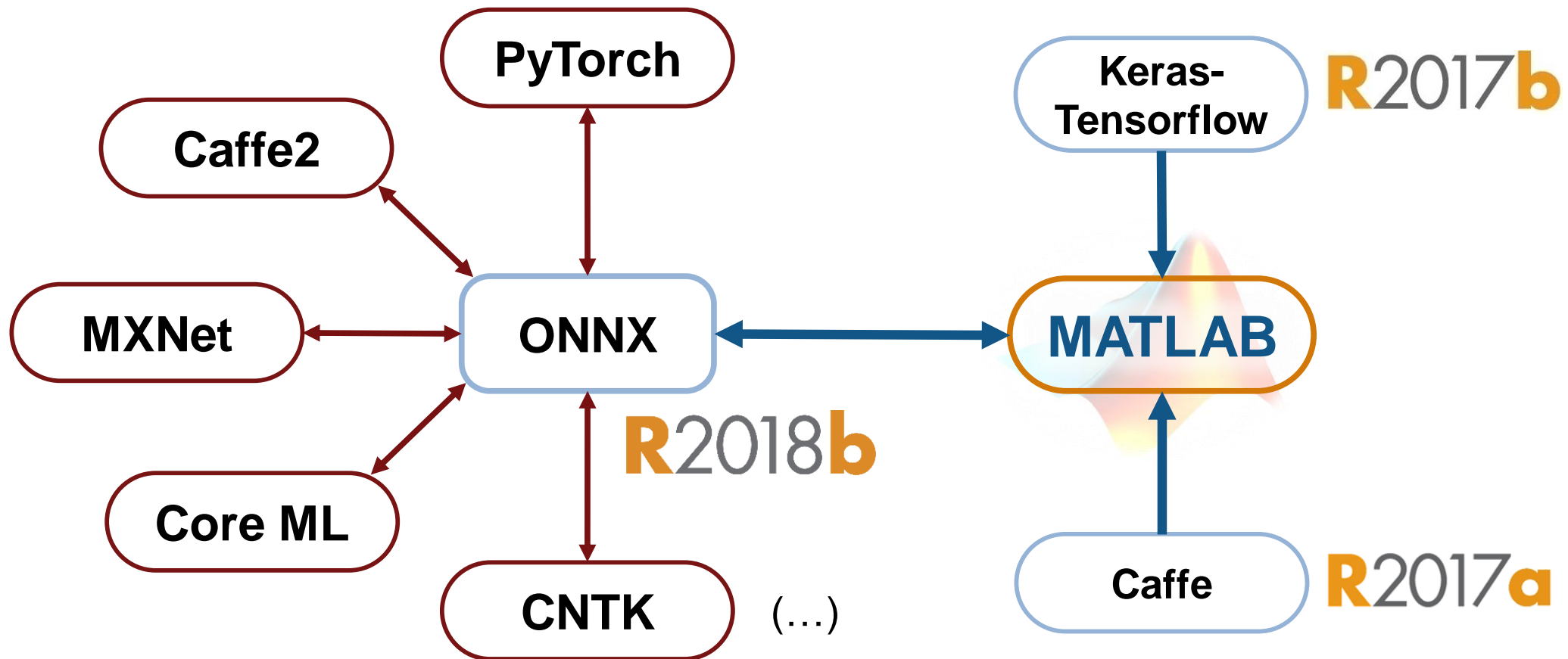


[Introduction to Micro-Doppler Effects](#)

Phased Array System Toolbox™

R2019a

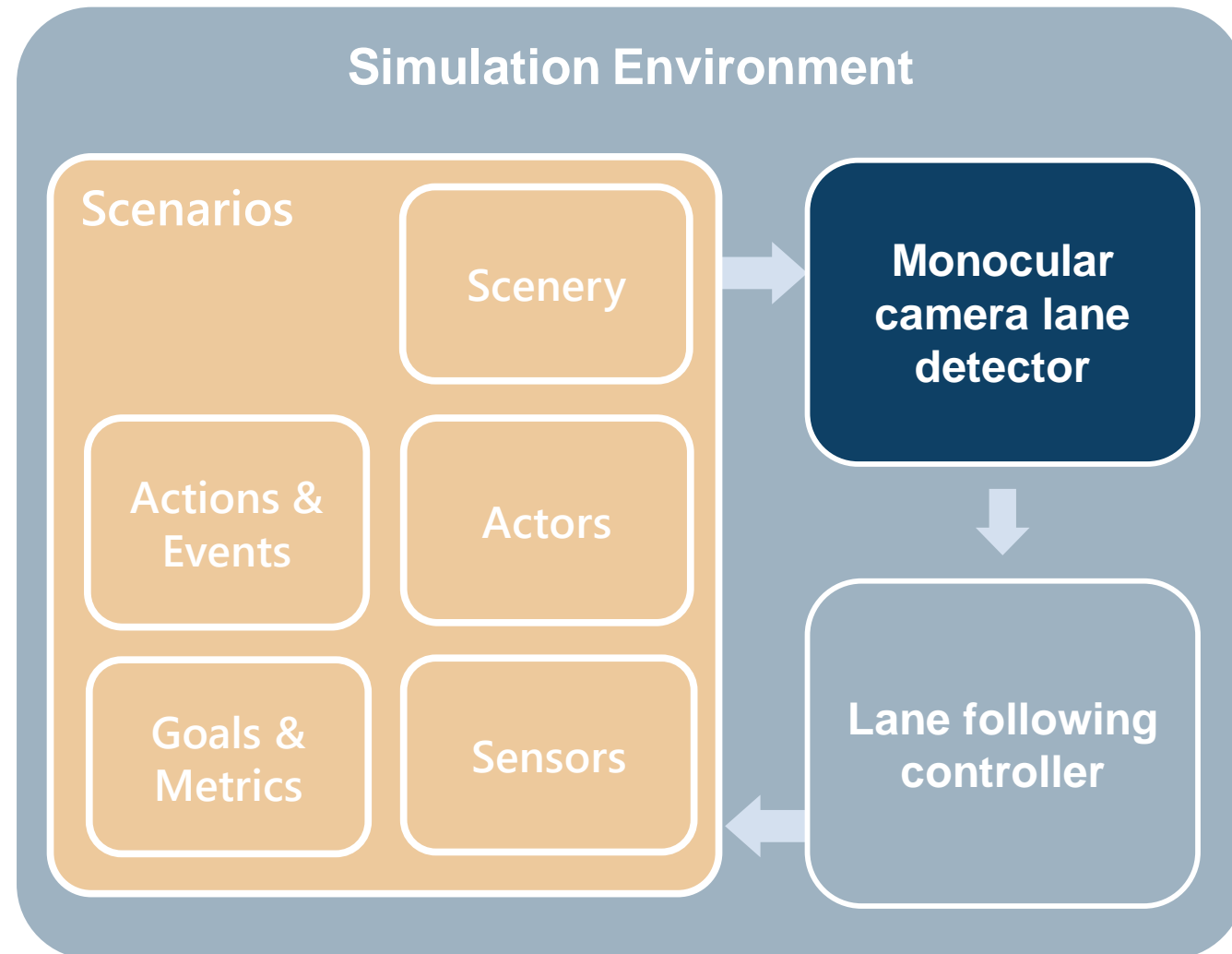
Interoperate with neural network frameworks



Open Neural Network Exchange

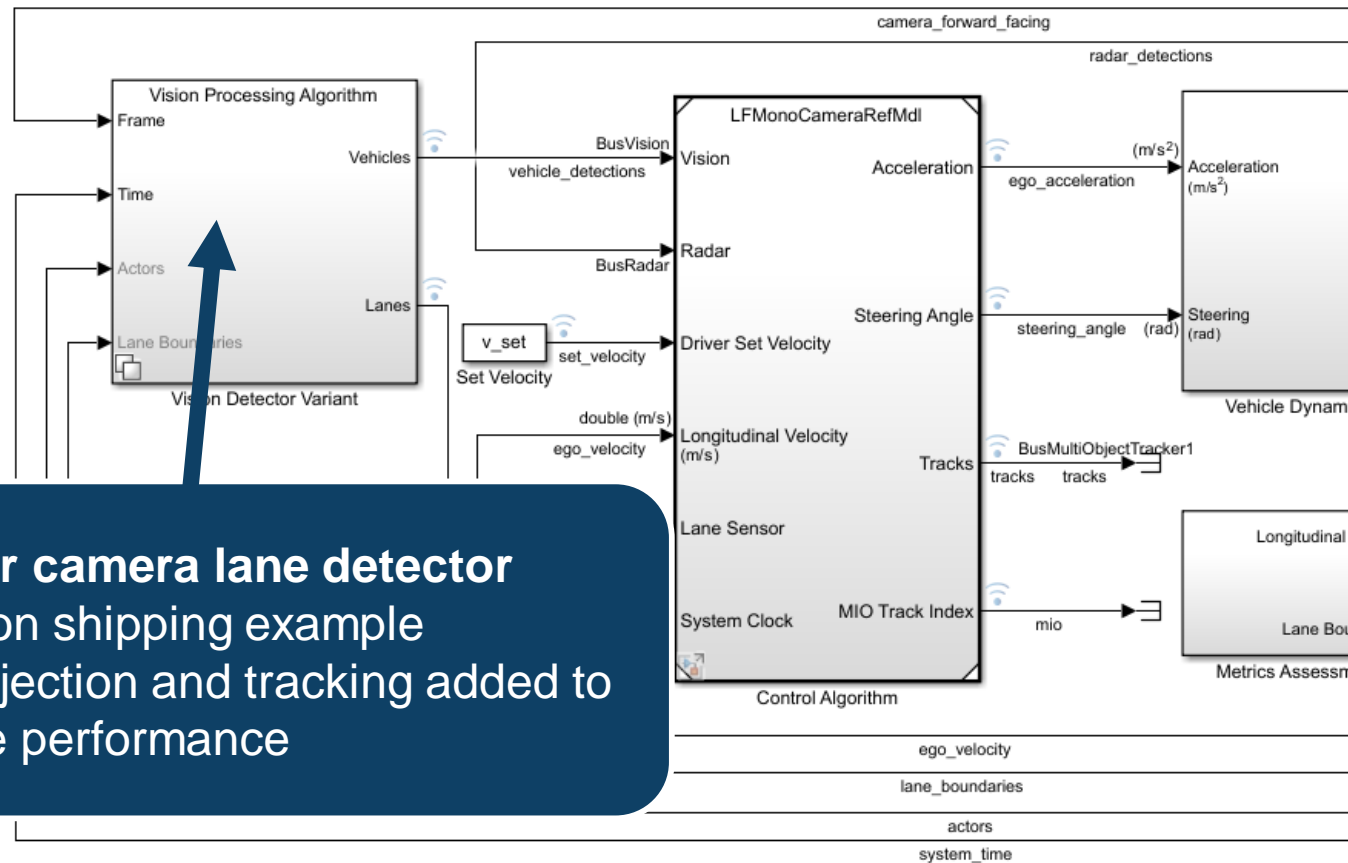
Visit the Demo Stations to see more...

Simulate lane detection and lane following system with MATLAB and Simulink



Monocular Camera Lane detector example

Lane Following with Mono Camera Detector Test Bench



Monocular camera lane detector

- Based on shipping example
- Lane rejection and tracking added to improve performance

```

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

[Visual Perception Using Monocular Camera](#)

Automated Driving Toolbox™

R2017a

Design detector 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™

R2019a

The screenshot shows the MATLAB R2019a environment. The Editor window displays the code for the `HelperBoundingBoxDetector.m` file. The code includes comments for various processing steps: cropping the point cloud, removing the ground plane, forming clusters, and generating bounding boxes. The Command Window shows the execution of `pcshow(pcObstacles)` and the resulting output of a 6x10 matrix of bounding box parameters.

```

methods (Access = protected)
function [bboxDets, obstacleI] = HelperBoundingBoxDetector(pcObstacles)
% Crop point cloud
[pcSurvived, survivedIndices] = cropPointCloud(pcObstacles);
% Remove ground plane
[pcObstacles, obstacleIndices] = removeGroundPlane(pcSurvived, survivedIndices);
% Form clusters and generate bounding boxes
detBBBoxes = getBoundingBoxes(pcObstacles, obstacleIndices);
% Assemble bounding boxes
bboxDets = detBBBoxes;
end
end
end

```

Command Window output:

```

K>> pcshow(pcObstacles)
fx K>>
detBBBoxes: 6x10 single matrix =
Columns 1 through 4
12.8921 -22.6758 -46.8280 -21.1414
-3.9148 -3.7233 -3.5872 0.0260
0.7299 0.6966 -0.6705 0.7558
2.8747 2.6390 0.0816 2.2517
1.7510 1.7391 0.8562 1.6446
1.0838 0.5916 0.0068 0.5503

```

A 3D visualization of the point cloud is shown in the Figure window, with a yellow callout box indicating the parameters of the bounding boxes: X-Center, Y-Center, Z-Center, Length, Width, and Height.

Design tracker for lidar point cloud data

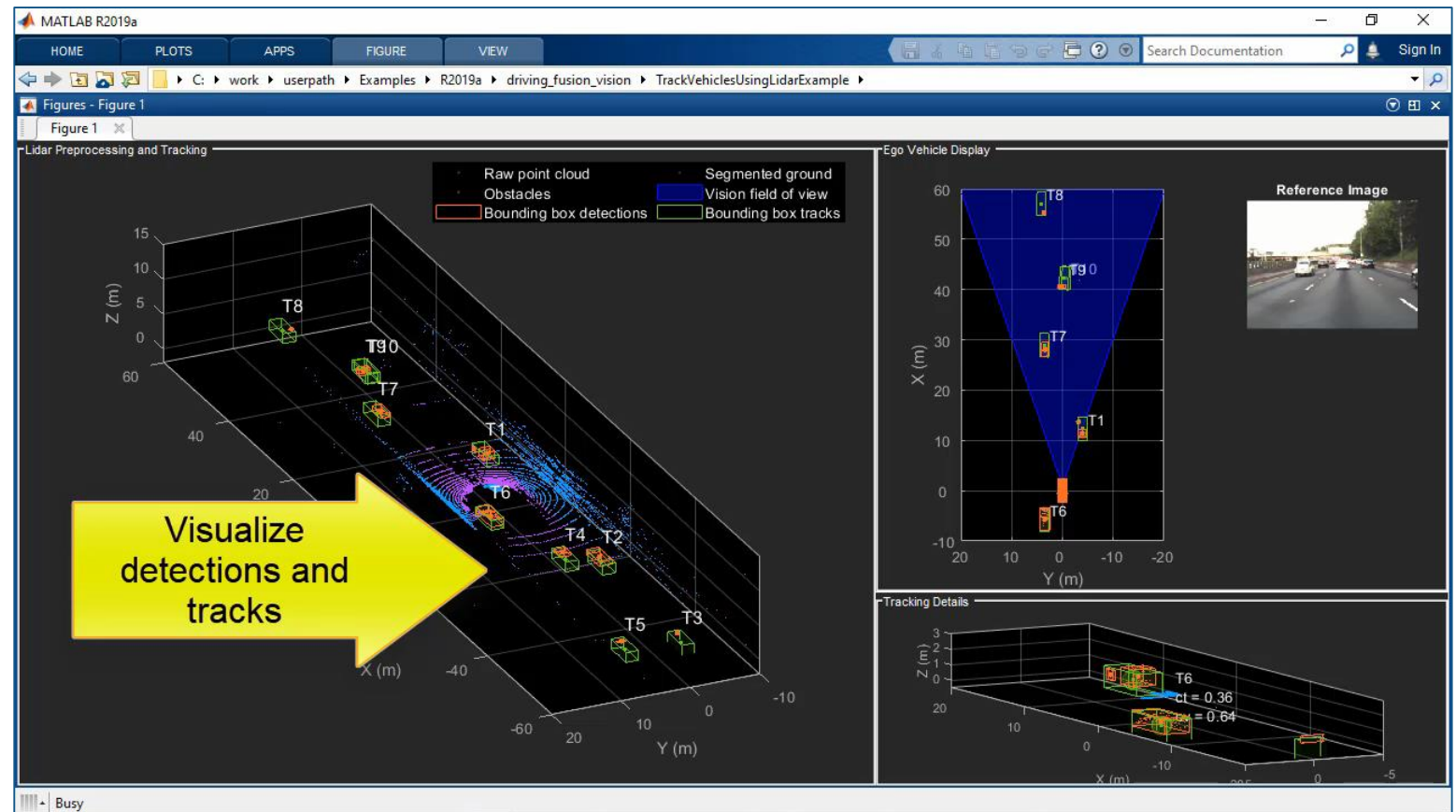
Track Vehicles Using Lidar: From Point Cloud to Track List

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

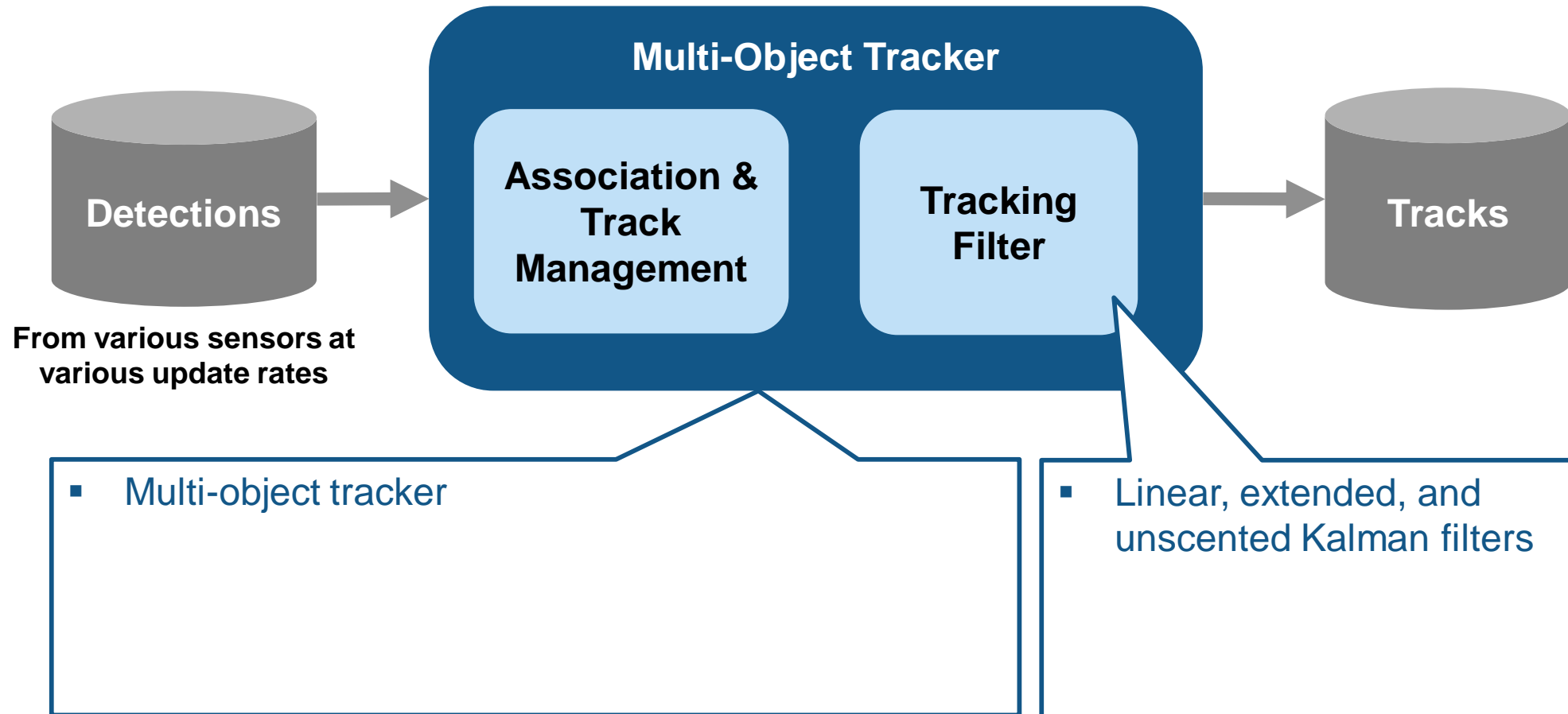
*Sensor Fusion and Tracking
Toolbox™*

Computer Vision Toolbox™

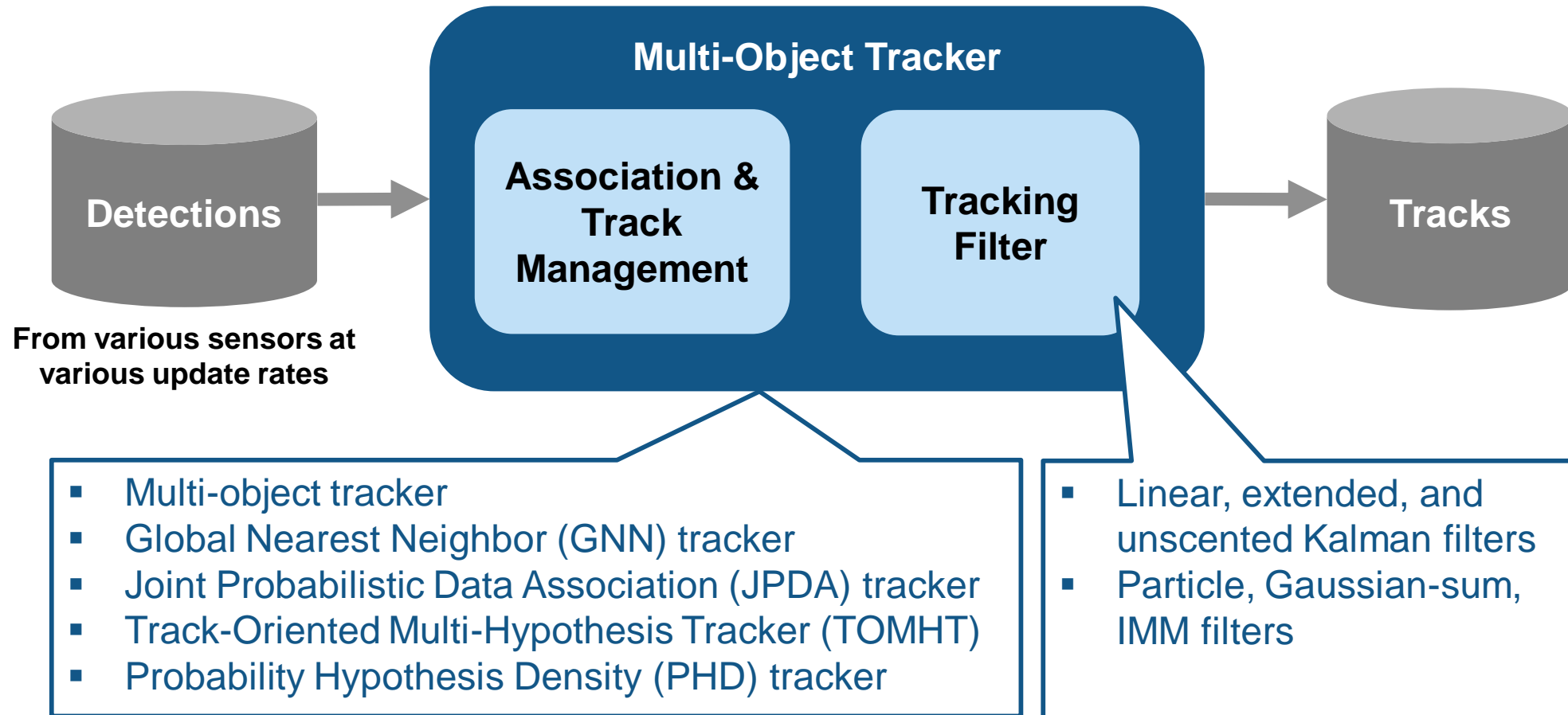
R2019a



Design trackers



Design trackers



Automated Driving Toolbox™

Sensor Fusion and Tracking Toolbox™

R2019a

Evaluate error metrics

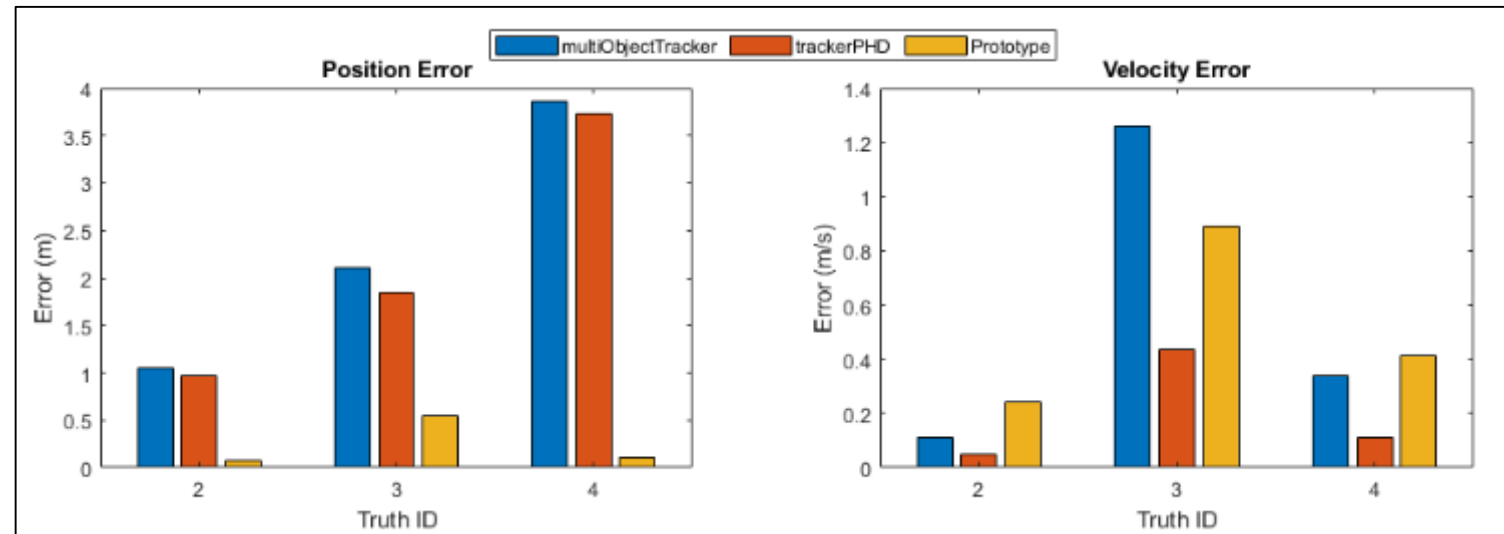
Extended Object Tracking

- 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

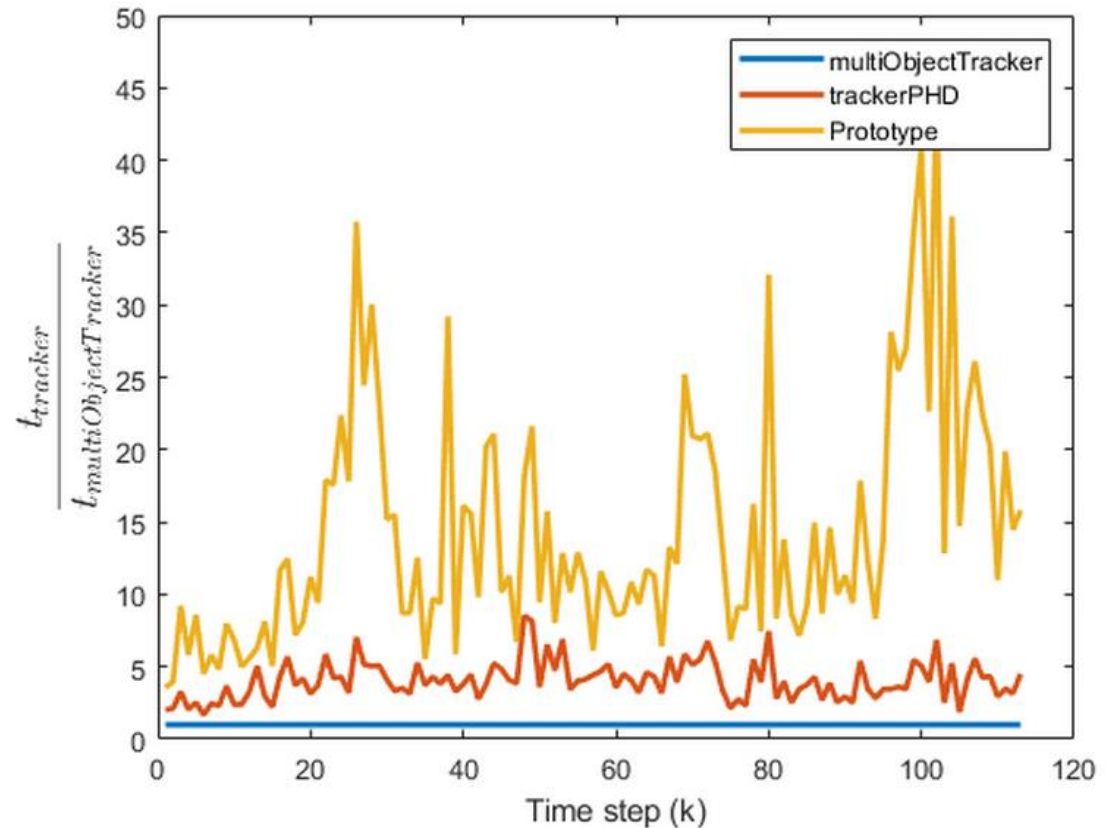
Extended Object Tracking

- Design multi-object tracker
- Design extended object trackers
- Evaluate tracking performance
- Evaluate error metrics
- Evaluate desktop execution time

*Sensor Fusion and
Tracking Toolbox™*

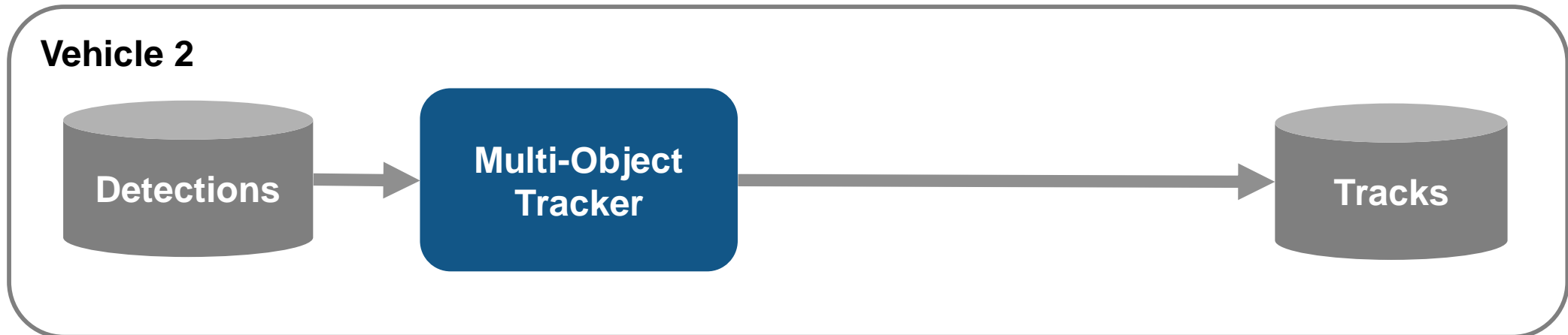
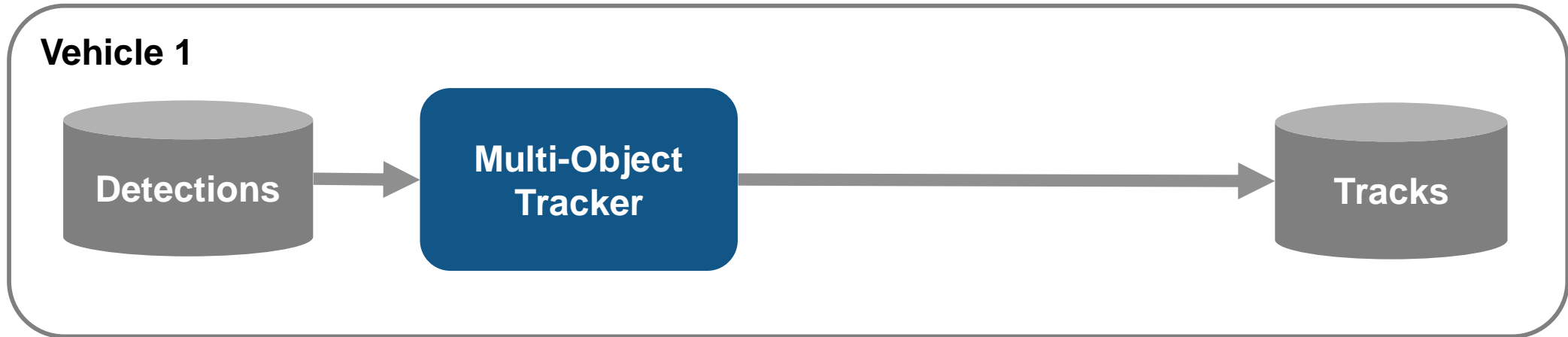
Automated Driving Toolbox™

Updated **R2019a**

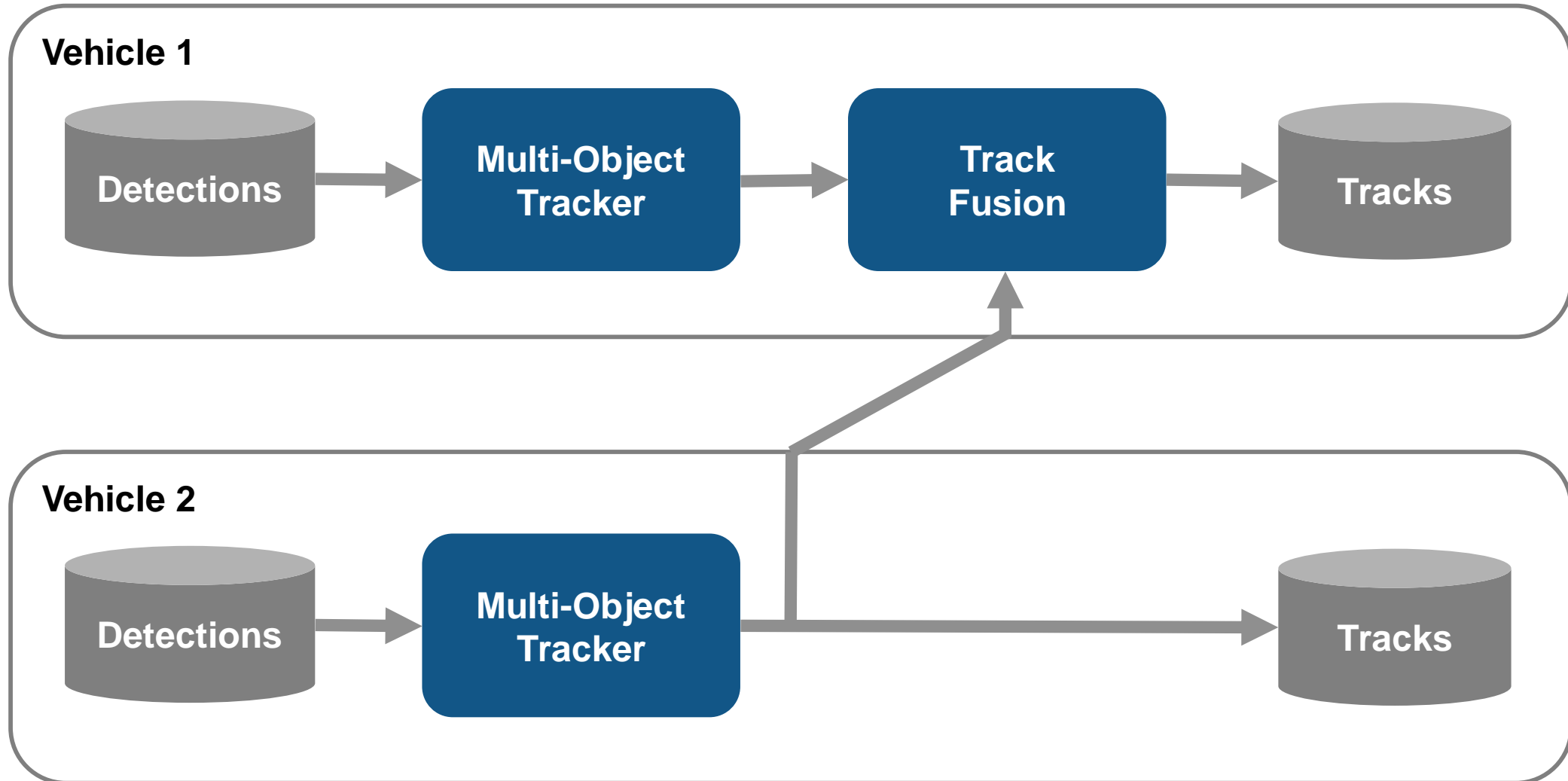


- Multi-object tracker
- Probability Hypothesis Density tracker
- Extended object (size and orientation) tracker

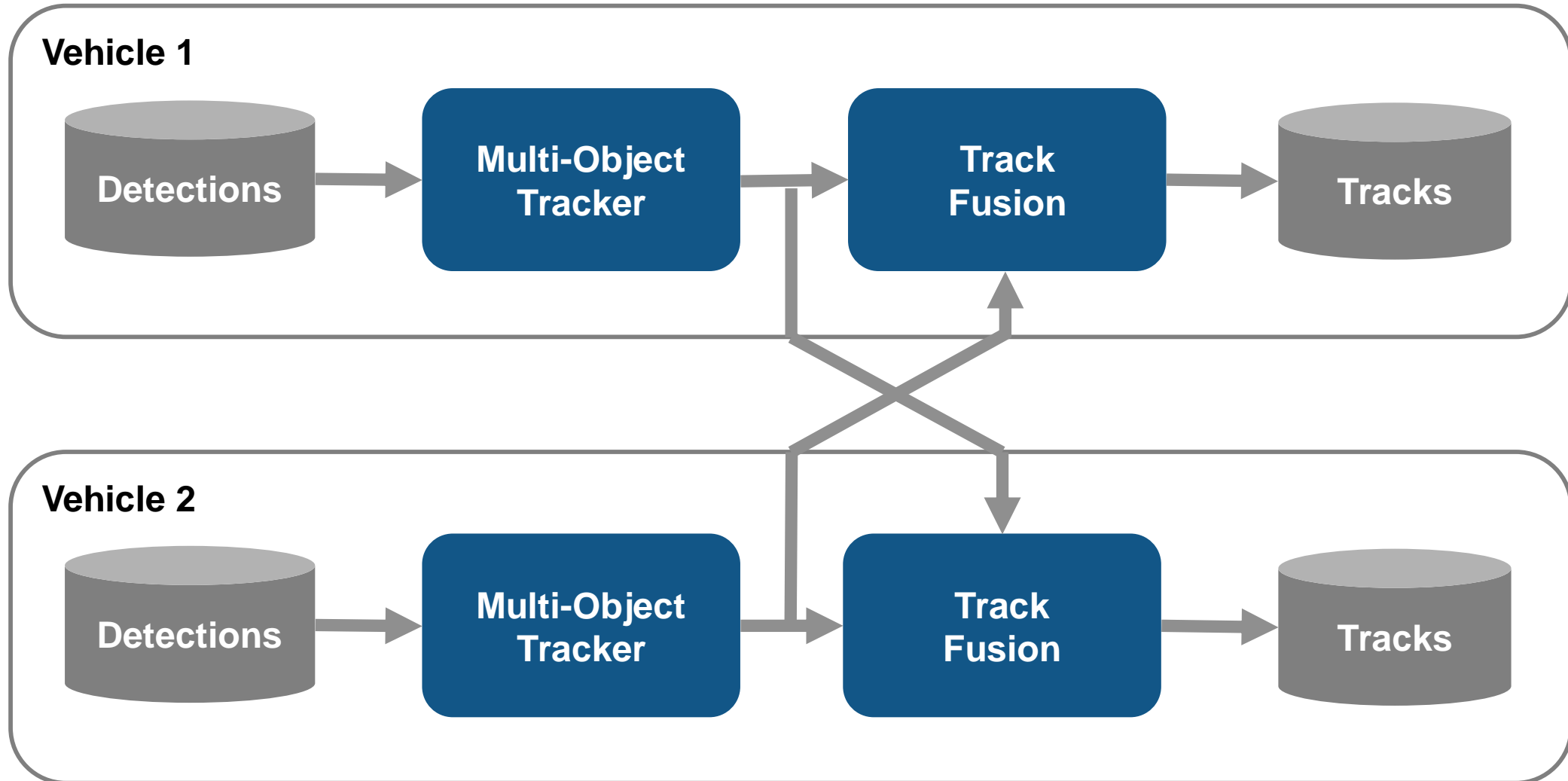
Design track level fusion systems



Design track level fusion systems



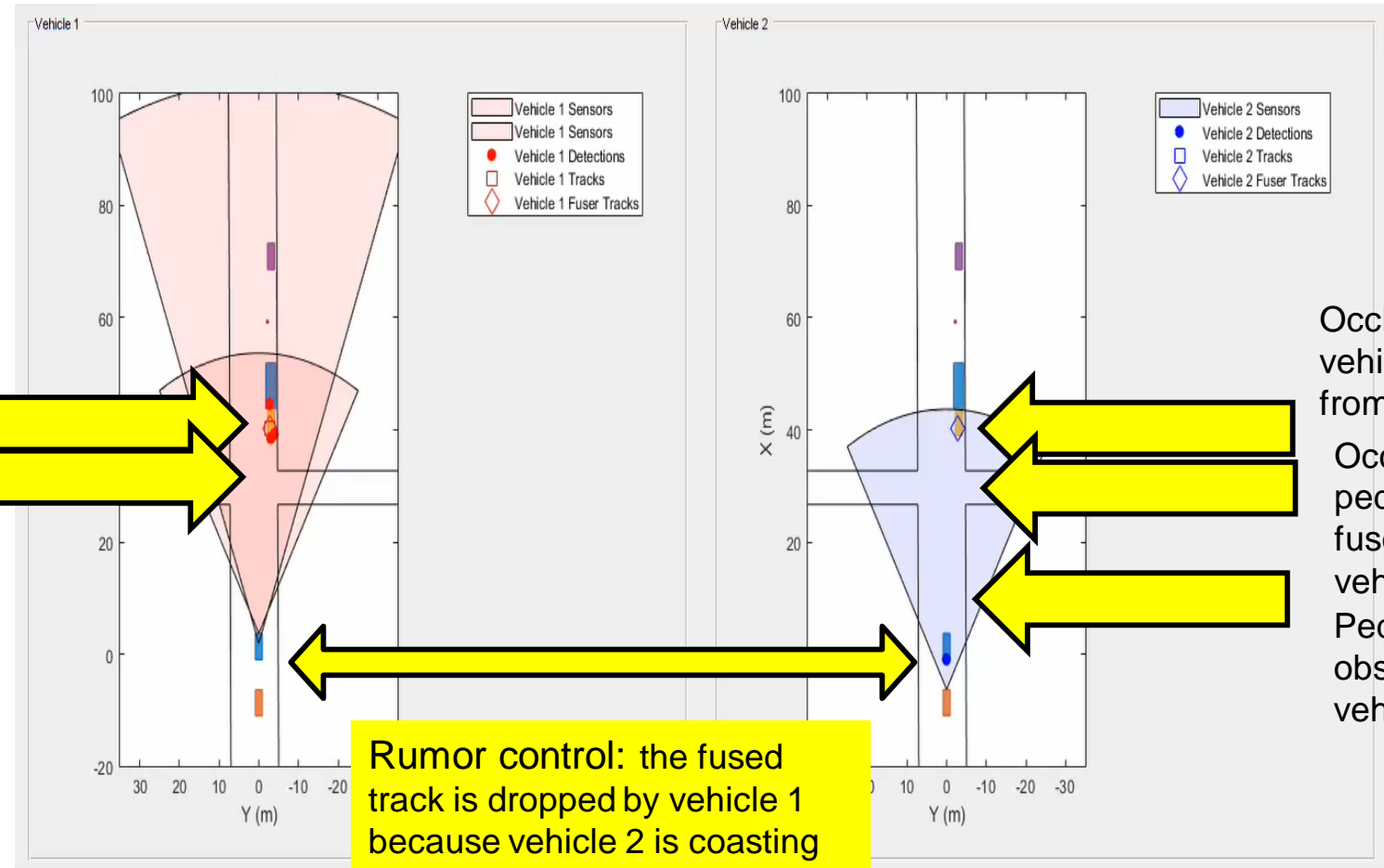
Design track level fusion systems



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 2

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™

R2019b

For more on Sensor Fusion and Tracking...

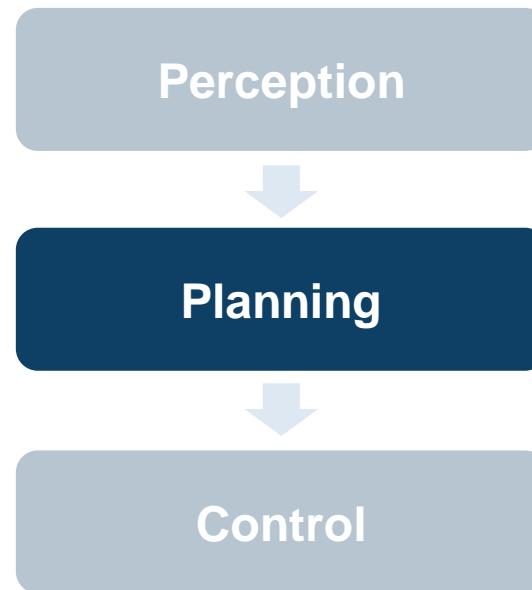
Visit Marc Willerton's presentation later this afternoon

| | Technical Computing | Model-Based Design | Getting Started with MATLAB and Simulink | Master Classes | Innovation Auditorium |
|-------|---|--|---|---|--|
| 15:15 | Break | | | | |
| 15:45 | Developing Smart IoT Sensors Using the MathWorks Toolchain <i>Samuel Bailey, Skyrad Consulting</i> | Synchronous Machine Modelling Using Simscape <i>Peenki Rani, Cummins Generator Technologies</i> | Sensor Fusion and Tracking for Autonomous Systems <i>Marc Willerton, MathWorks</i> | Simplifying Requirements-Based Verification with Model-Based Design <i>Fraser Macmillan, MathWorks</i> | Predictive Maintenance with MATLAB <i>Phil Rottier, MathWorks</i> |
| 16:15 | Industrial IoT and Digital Twins <i>Coorous Mohtadi, MathWorks</i> | Developing Fit-For-Purpose Simscape Models to Support System and Control Design <i>Rick Hyde, MathWorks</i> | | | |
| 17:00 | End of Day | | | | |

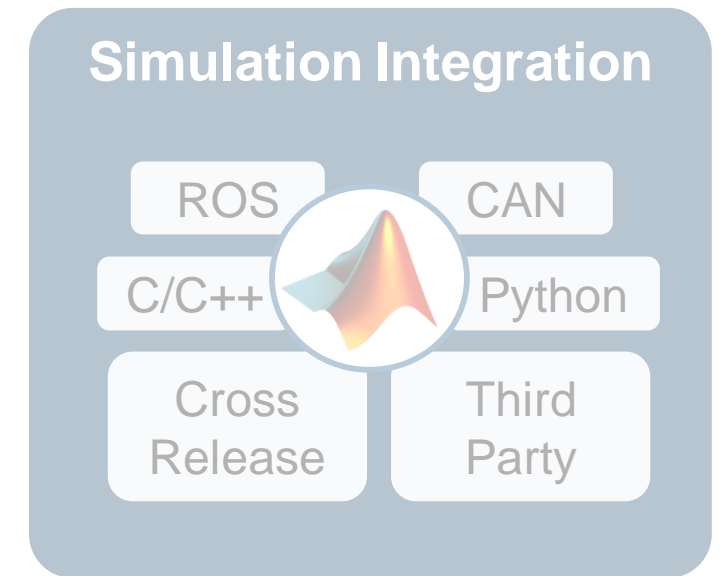
Some common questions from automated driving engineers



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?

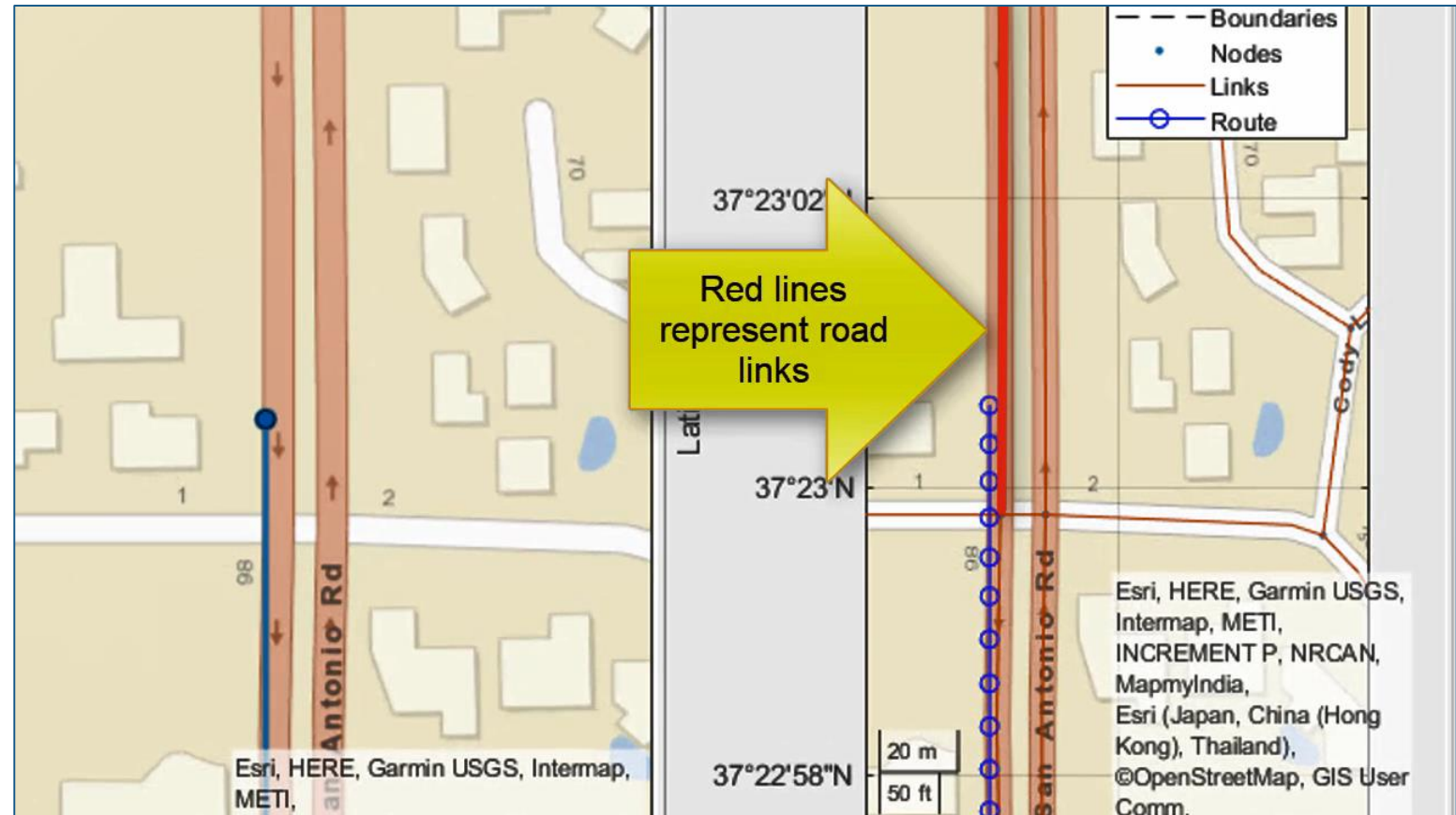
Read road and speed attributes from HERE HD Live Map data

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



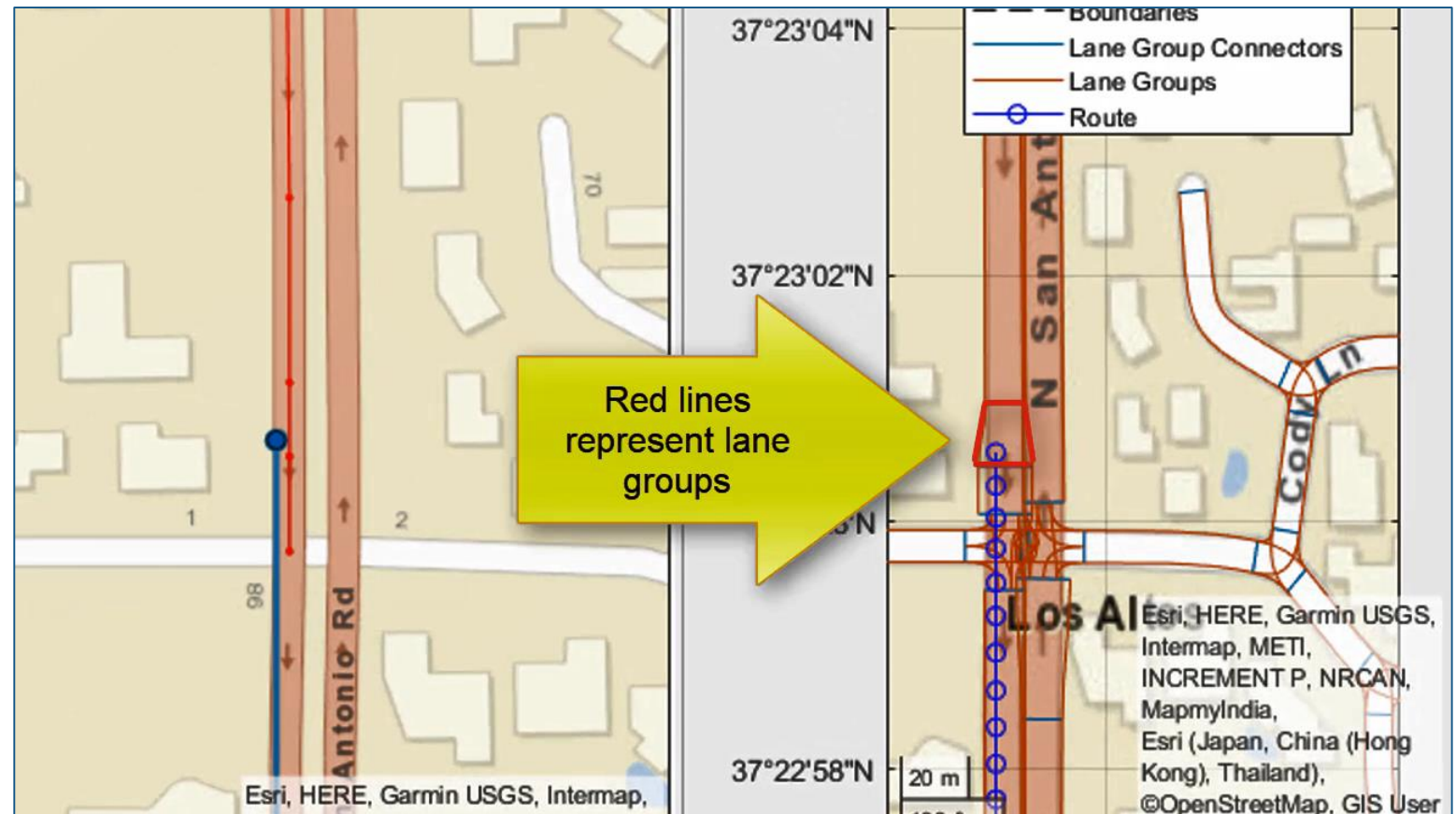
Read lane attributes from HERE HD Live Map data

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

The screenshot shows the MATLAB R2019a environment. The script editor on the left contains the following code:

```

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
  
```

The visualization window, titled 'HERE HD Live Map Example', displays a camera view of a road with a traffic jam. To the right of the camera view is a map view showing the road layout and lane configurations. Below the map view is a data panel with the following information:

- Timestamp: 22:11:25
- Speed Limit: 35
- Lane Types and Boundaries: A diagram showing lane configurations, including a 'BICYCLE' lane.

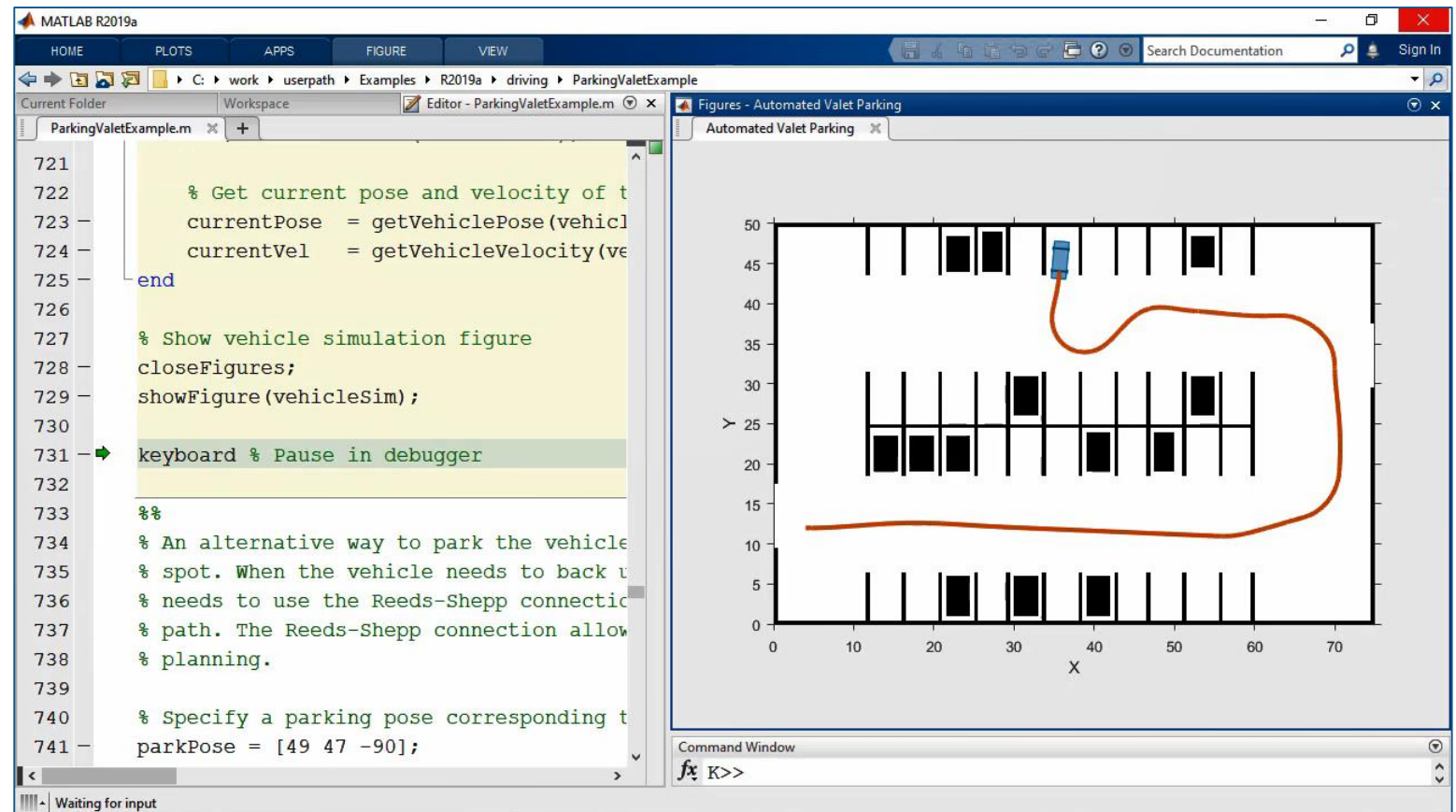
Design path planner

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

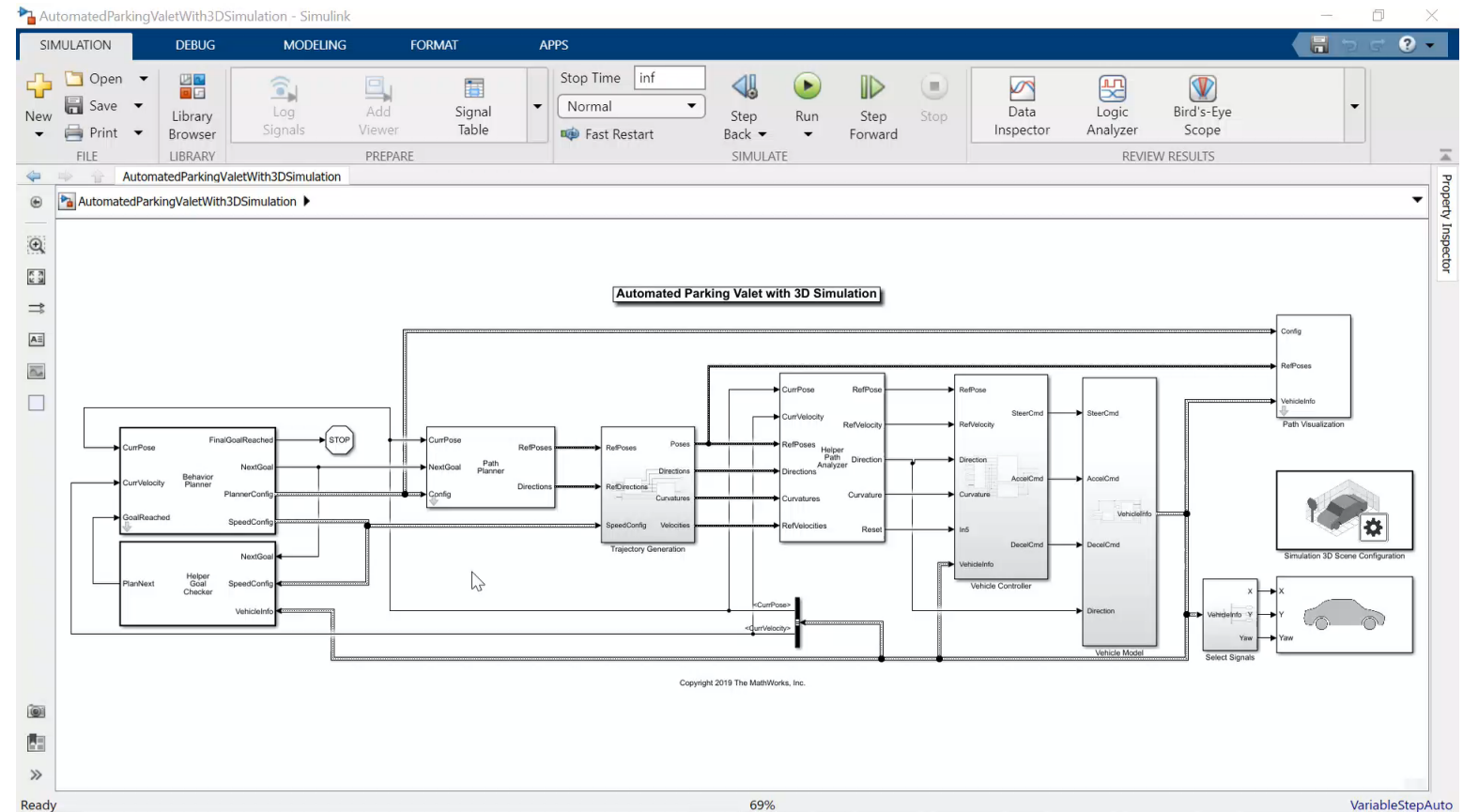


Design path planner and controller

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

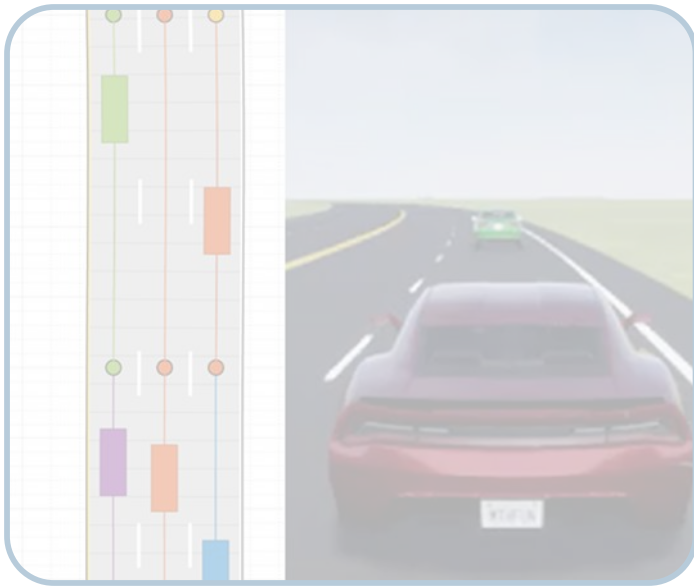
Visualize Automated Parking Valet Using 3D Simulation



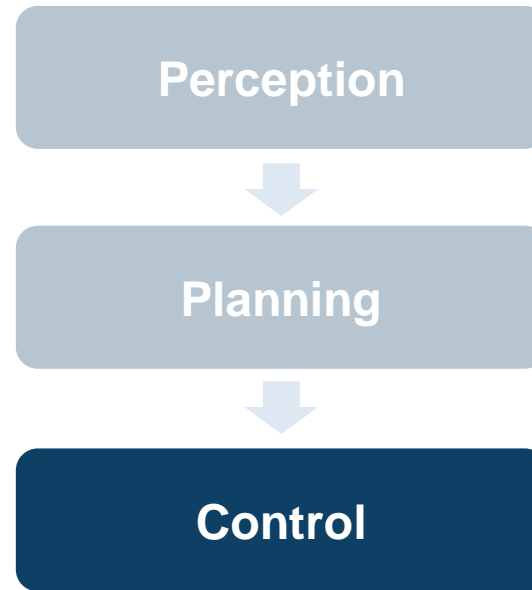
Automated Driving Toolbox™

R2018b **R2019b**

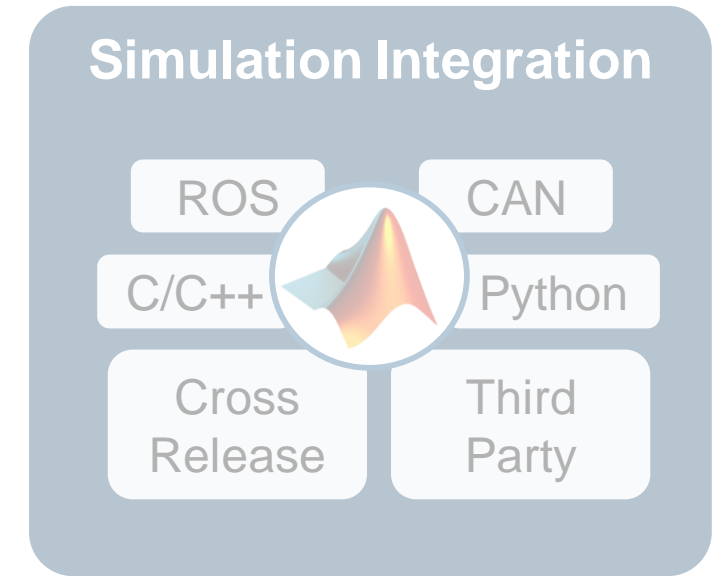
Some common questions from automated driving engineers



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?

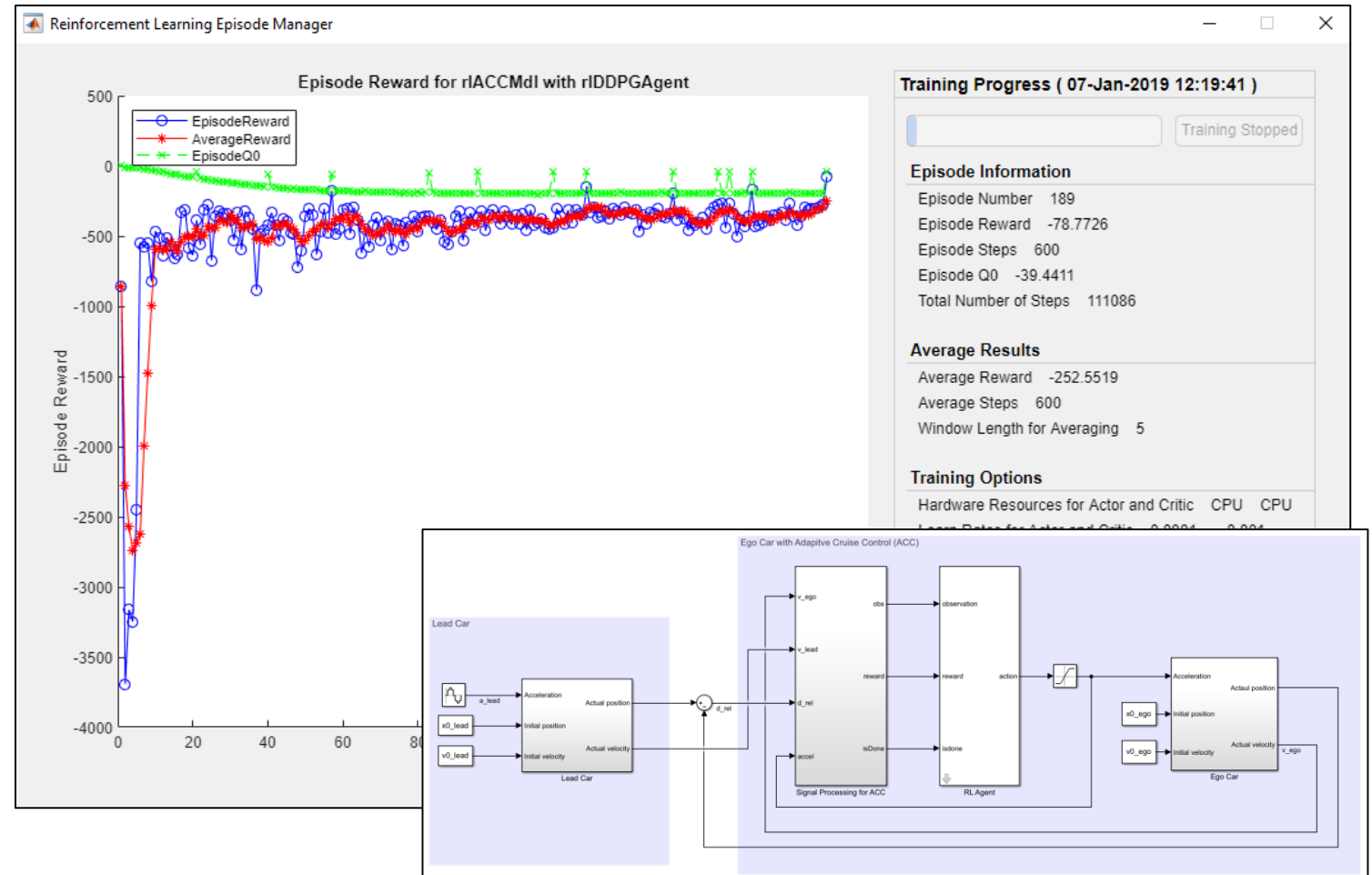
Train reinforcement learning networks for ADAS controllers

Train Deep Deterministic Policy Gradient (DDPG) Agent for Adaptive Cruise Control

- Create environment interface
- Create agent
- Train agent
- Simulate trained agent

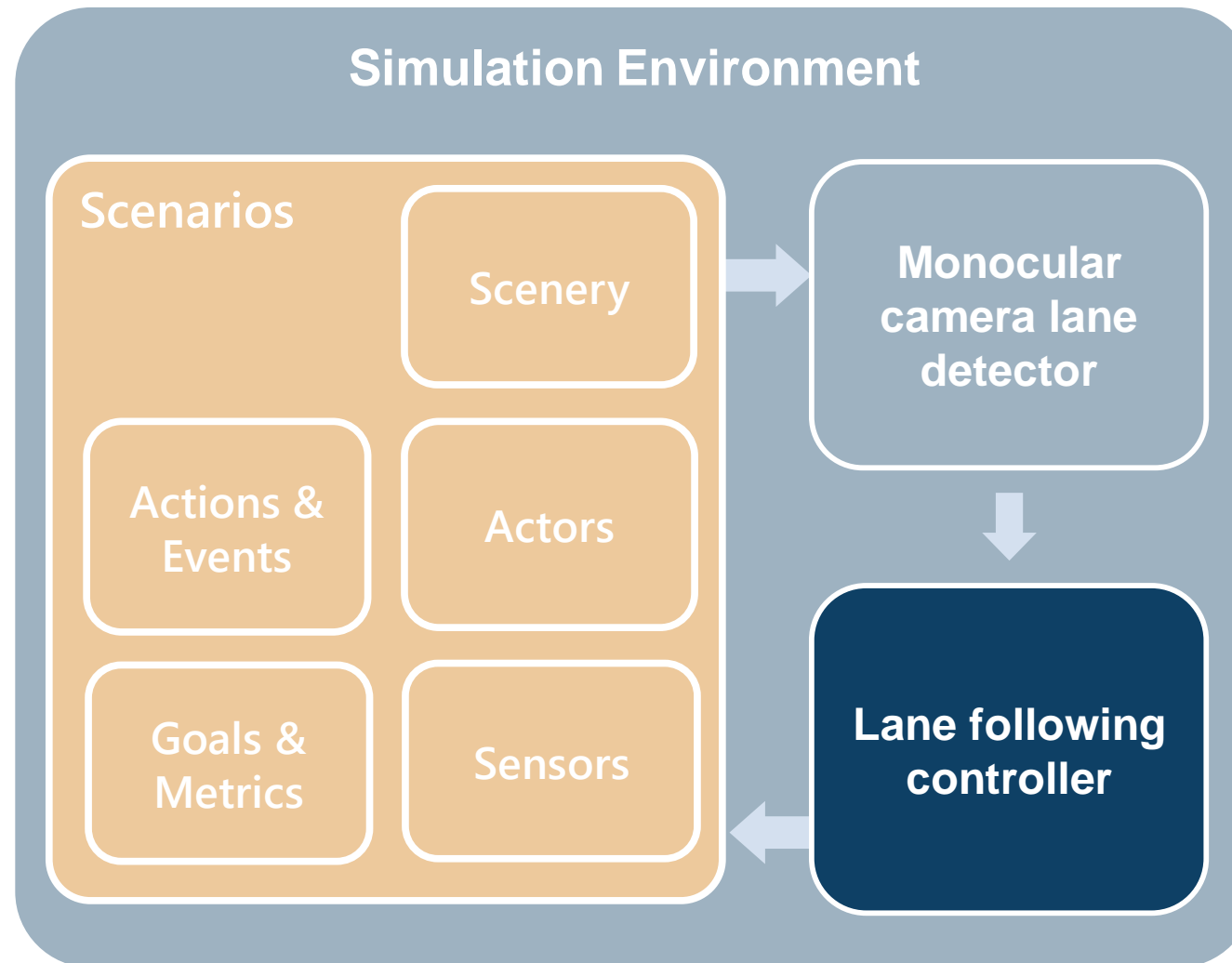
Reinforcement Learning Toolbox™

R2019a



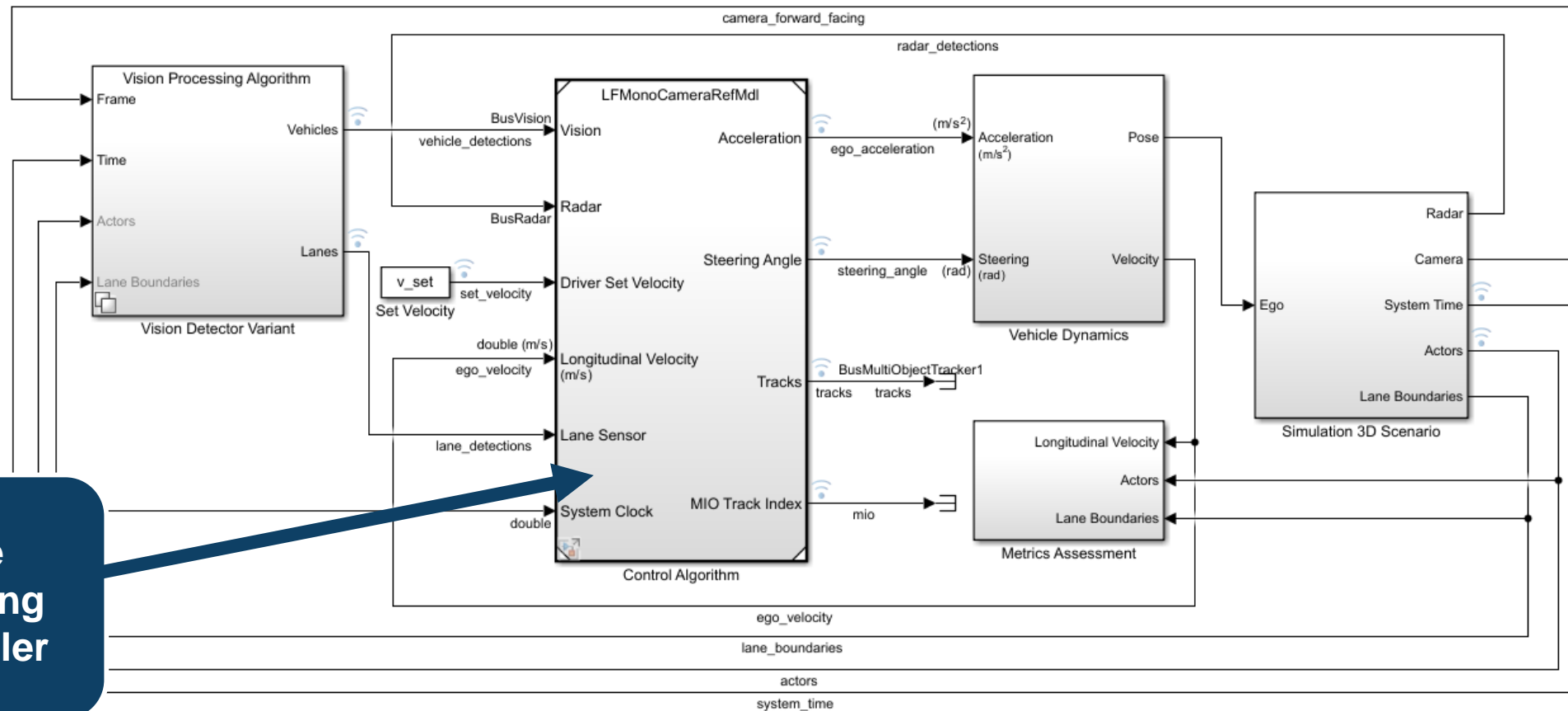
Visit the Demo Stations to see more...

Simulate lane detection and lane following system with MATLAB and Simulink



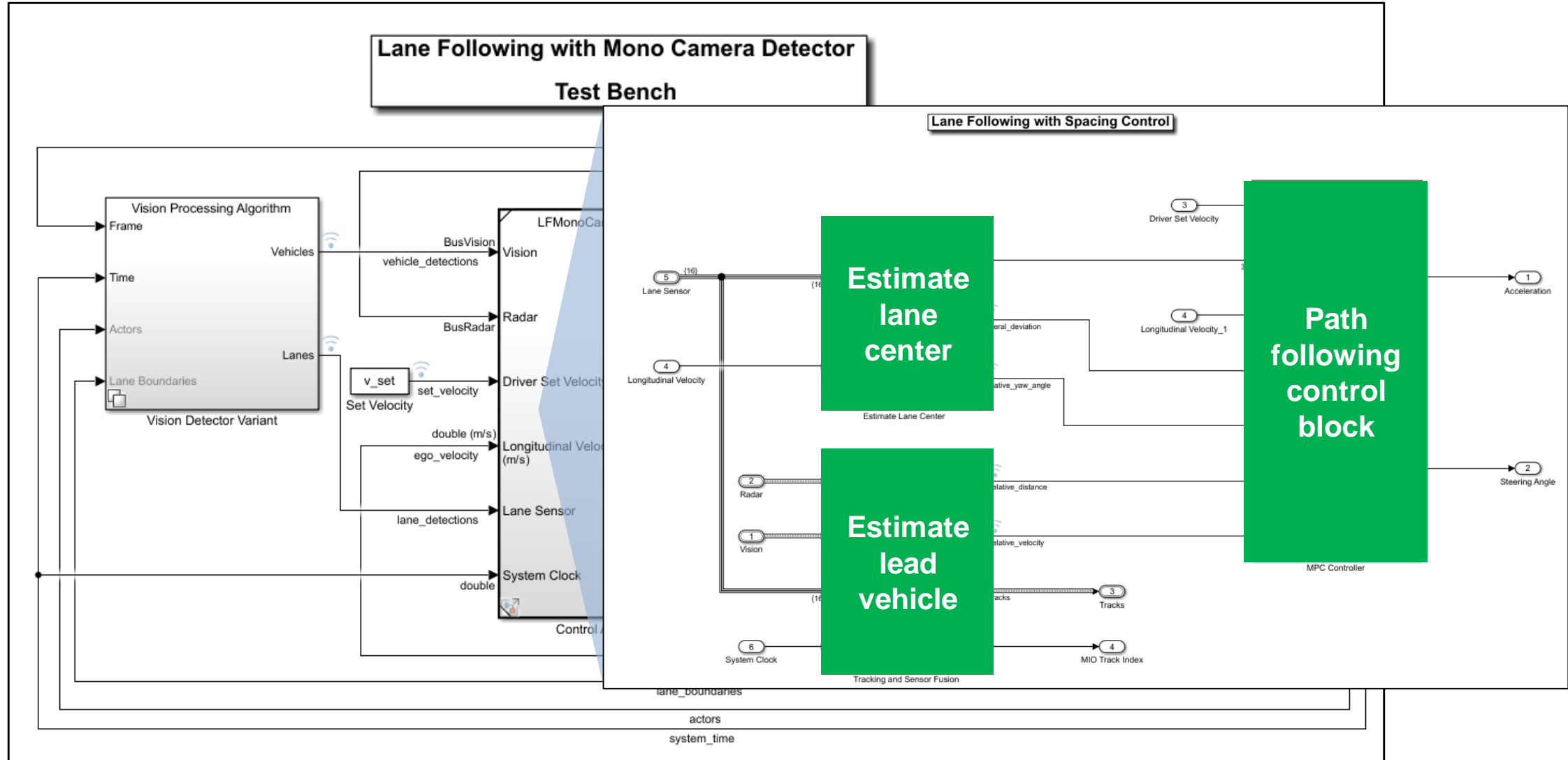
Lane Following Controller Algorithm

Lane Following with Mono Camera Detector Test Bench



Lane following controller

Components of lane following with spacing control algorithm

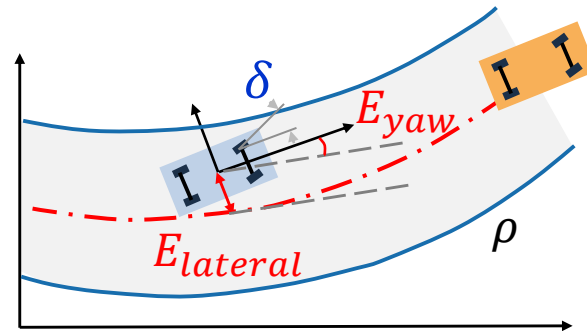


Goal

- Maintain the driver-set velocity and keep a safe distance from lead vehicle.

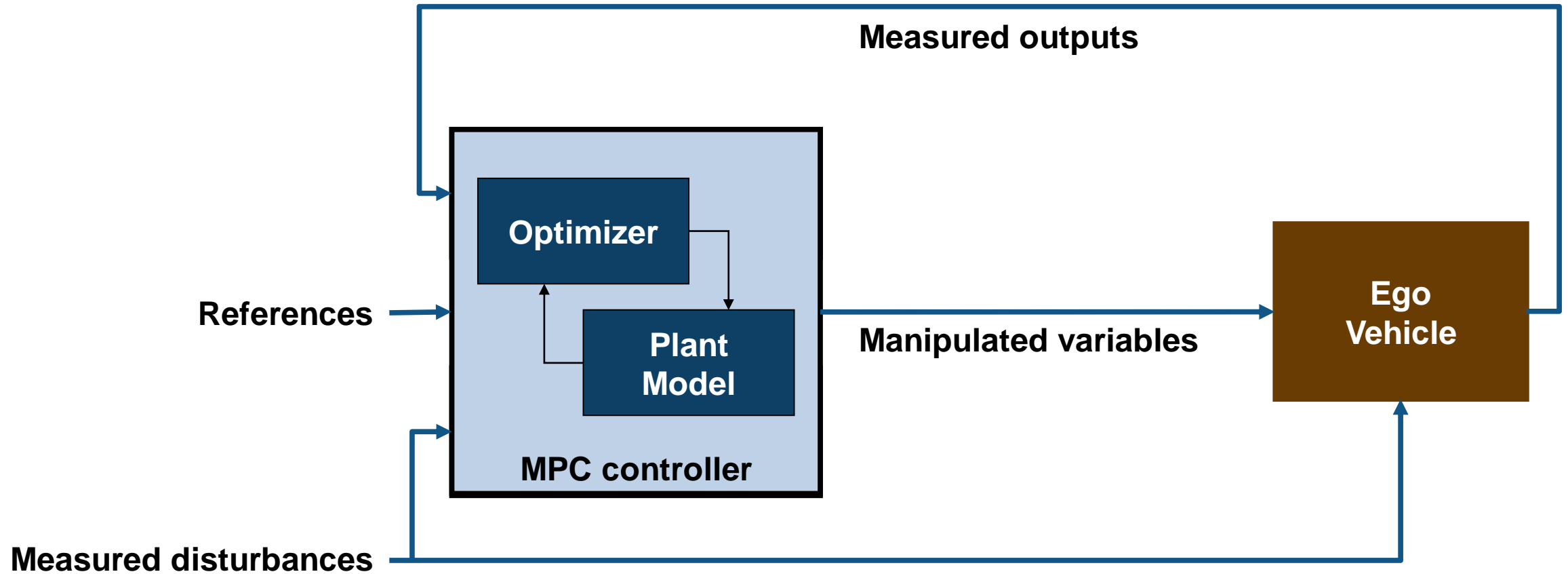


- Keep the ego vehicle in the middle of the lane.



- 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

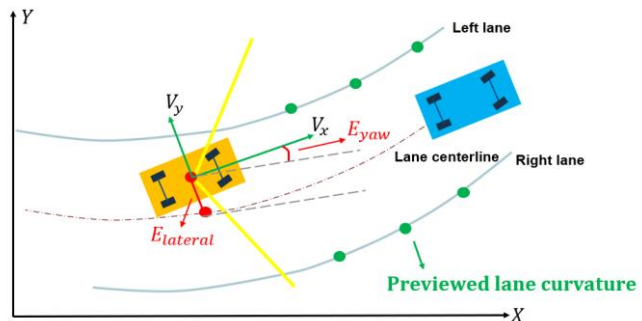
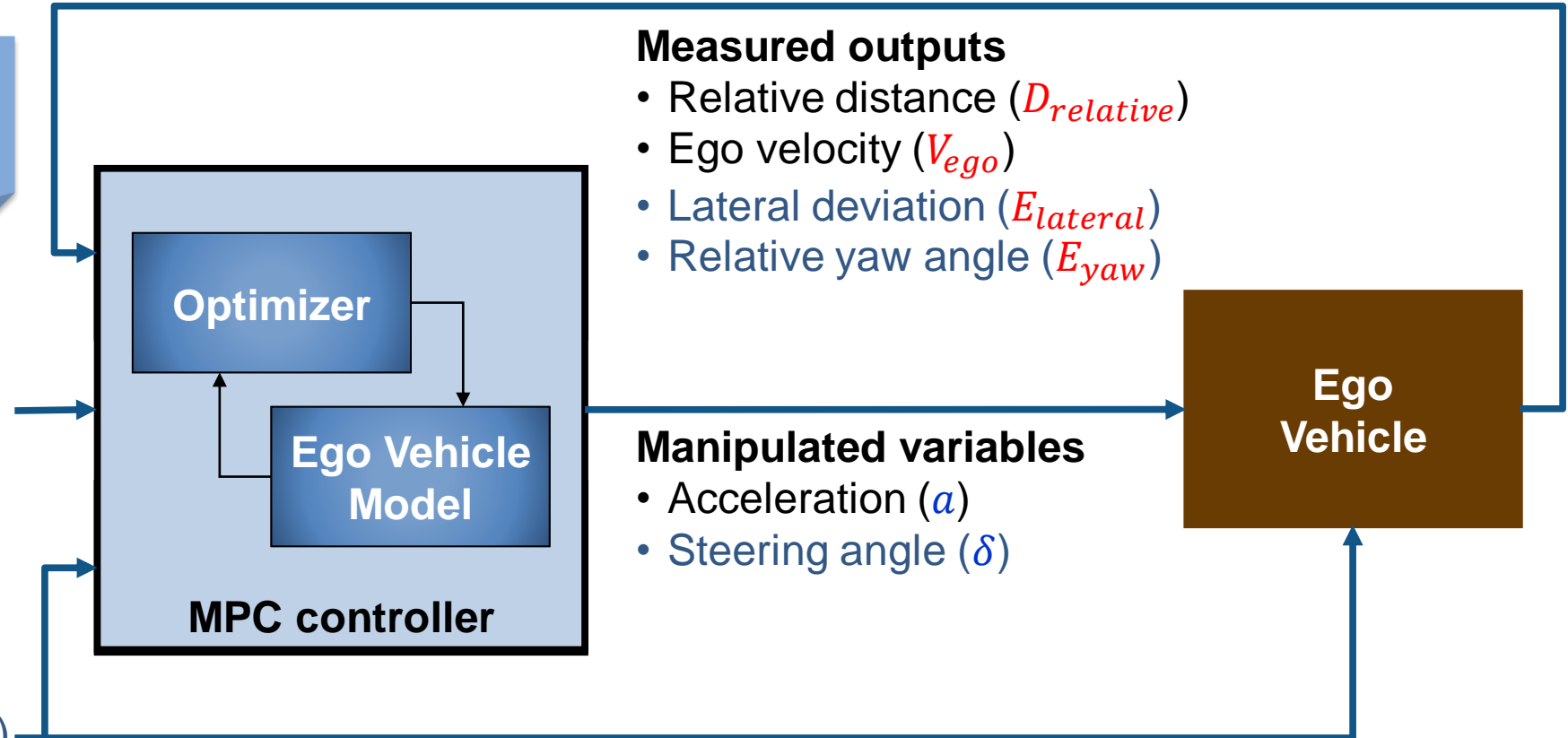
- MIO velocity (V_{mio})
- **Previewed** road curvature (ρ)

subject to:

$$D_{relative} \geq D_{safe}$$

$$a_{min} \leq a \leq a_{max}$$

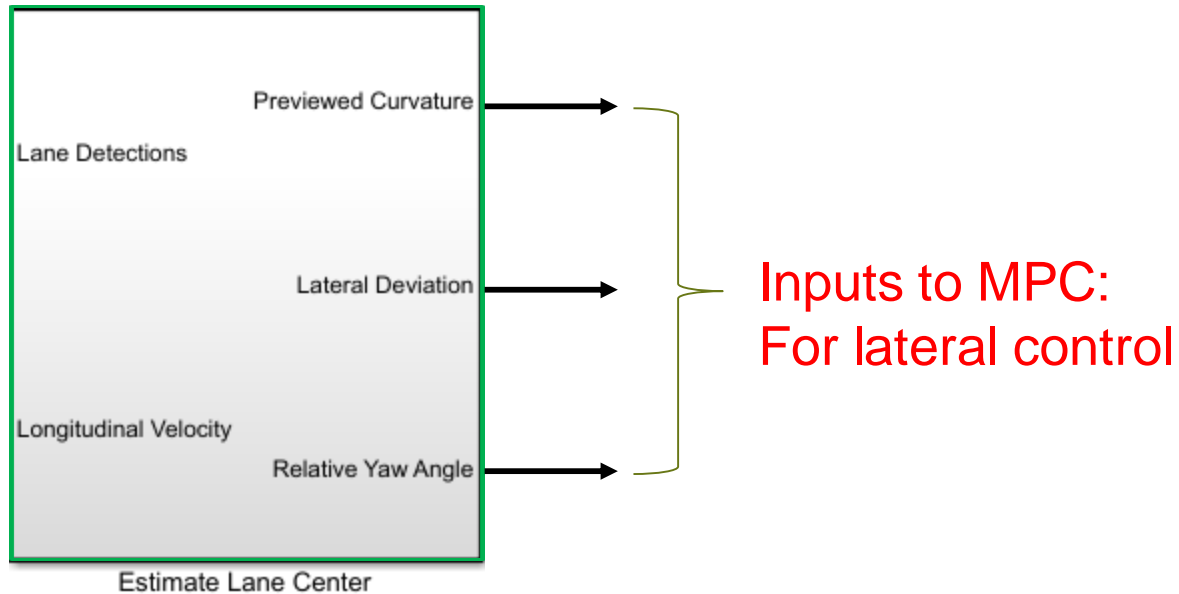
$$\delta_{min} \leq \delta \leq \delta_{max}$$



Look ahead → Act early!

Components

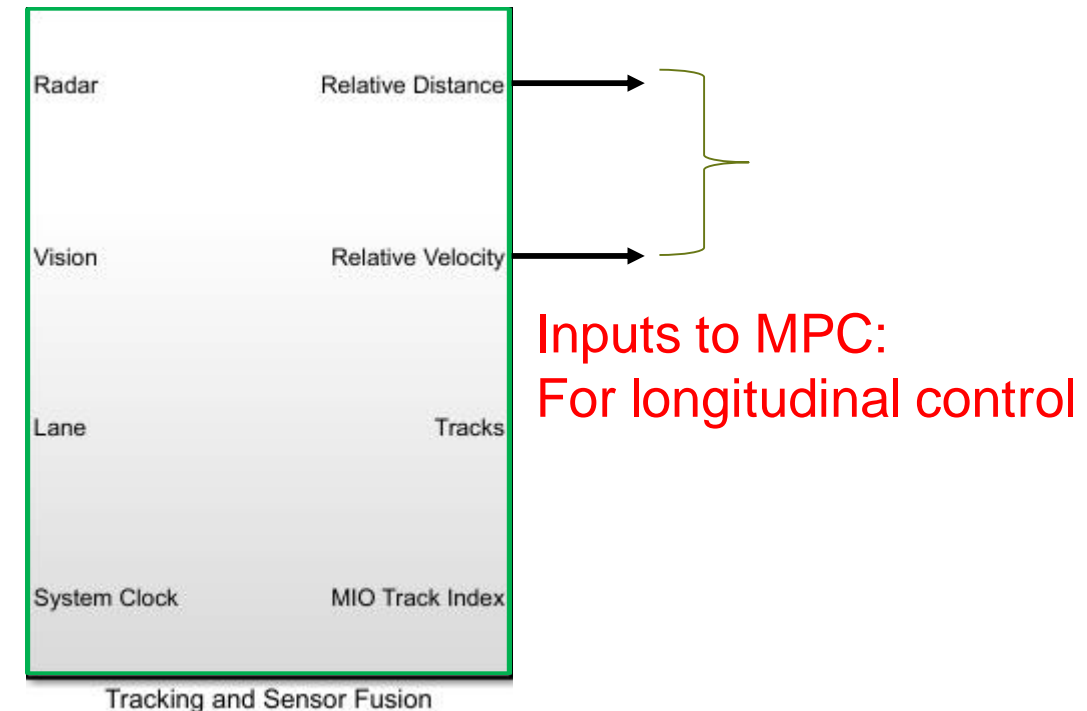
- Estimate lane center



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)

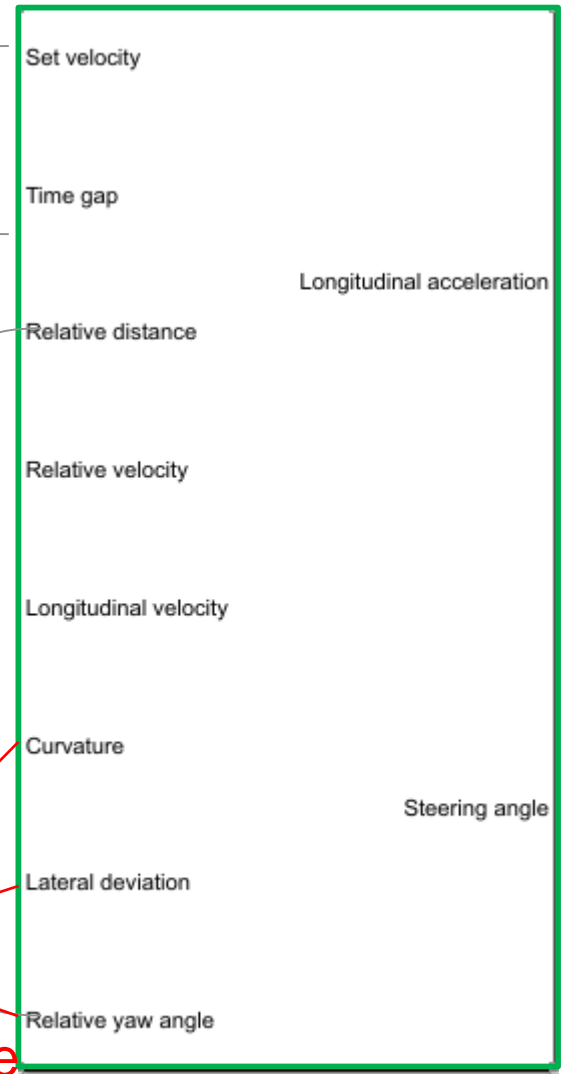


- MPC: Path following controller

Path Following Control Block

Model Predictive Control Toolbox™

R2019b



Block Parameters: Path Following Control System

Path following control (PFC) system (mask) (link)

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 Controller Block

Ego Vehicle

Linear model from [longitudinal acceleration (m/s²) and front steering angle (rad)] to [longitudinal velocity (m/s), lateral velocity (m/s) and yaw angle rate (rad/s)]

Use vehicle parameters

Use vehicle model

Vehicle parameters

- Total mass (kg) 1575
- Yaw moment of inertia (mNs²) 2875
- Longitudinal distance from center of gravity to front tires (m) 1.2
- Longitudinal distance from center of gravity to rear tires (m) 1.6
- Cornering stiffness of front tires (N/rad) 19000
- Cornering stiffness of rear tires (N/rad) 33000
- Longitudinal acceleration tracking time constant (s) 0.5

Initial longitudinal velocity (m/s) 15

Transport lag between model inputs and outputs (s) 0

Spacing Control

Maintain safe distance between lead vehicle and ego vehicle Default spacing (m) 10

OK Cancel Help Apply

Driver setting

Measurement

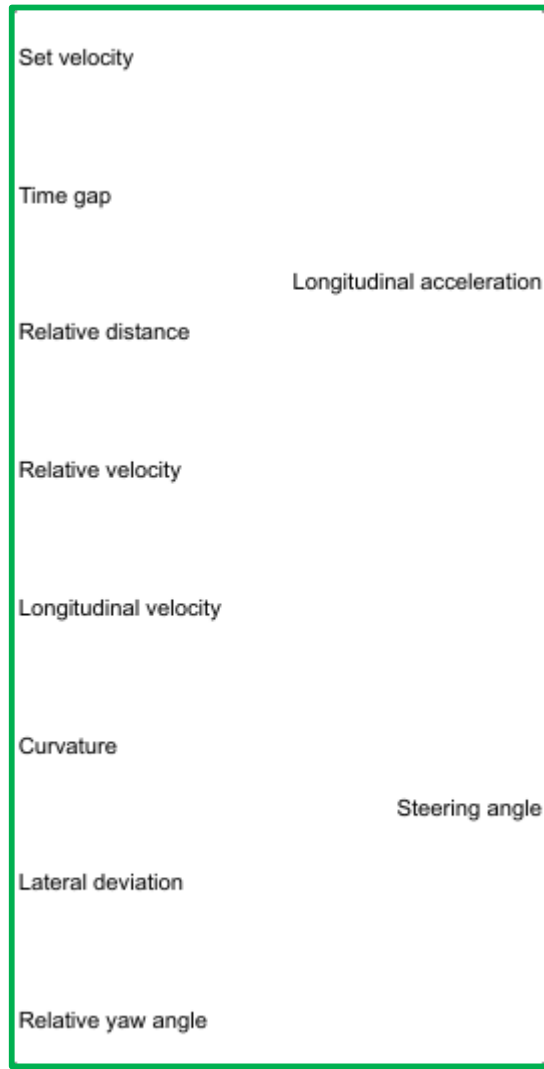
Virtual lane
For lane change

Bicycle model parameters

Delay or latency in the system

Disable distance keeping

Path Following Control Block



Path Following Control System

Block Parameters: Path Following Control System

Path following control (PFC) system (mask) (link)

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 Controller Block

Path Following Controller Constraints

- Minimum steering angle (rad) Use external source
- Maximum steering angle (rad) Use external source
- Minimum longitudinal acceleration (m/s²) Use external source
- Maximum longitudinal acceleration (m/s²) Use external source

Model Predictive Controller Settings

- Sample time (s)
- Prediction horizon (steps) Control horizon (steps)

Controller Behavior

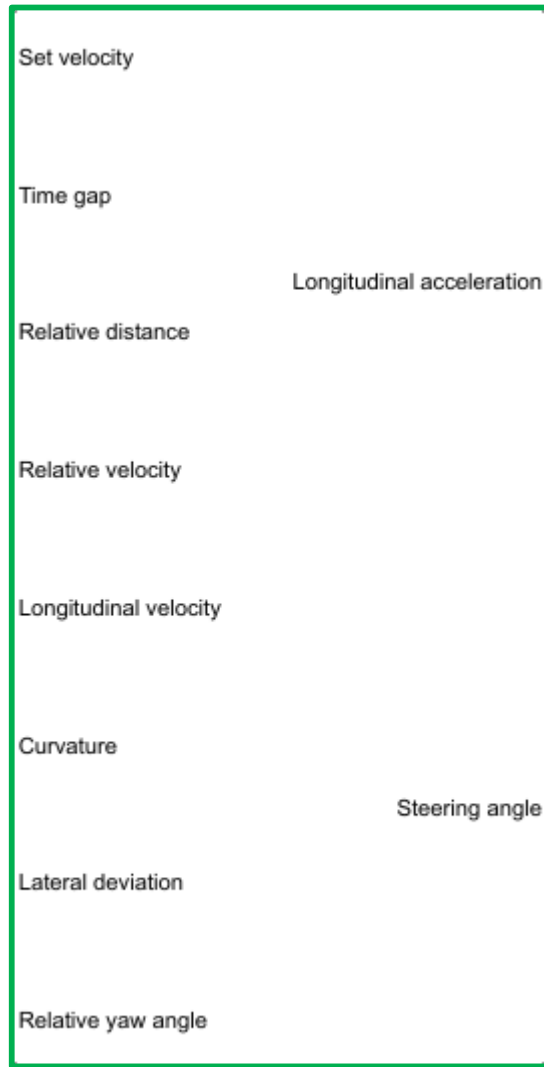
- Weight on velocity tracking Weight on change of longitudinal acceleration
- Weight on lateral error Weight on change of steering angle

OK Cancel Help Apply

Actuator limits

Tune MPC performance

Path Following Control Block



Path Following Control System

Block Parameters: Path Following Control System

Path following control (PFC) system (mask) (link)

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 Controller Block

Optimization

- Use suboptimal solution
- Maximum iteration number: 10

Data Type

- double
- single

Optional Inports

- 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.

Create PFC subsystem

Change MPC design

OK Cancel Help Apply

Simulate controls with perception

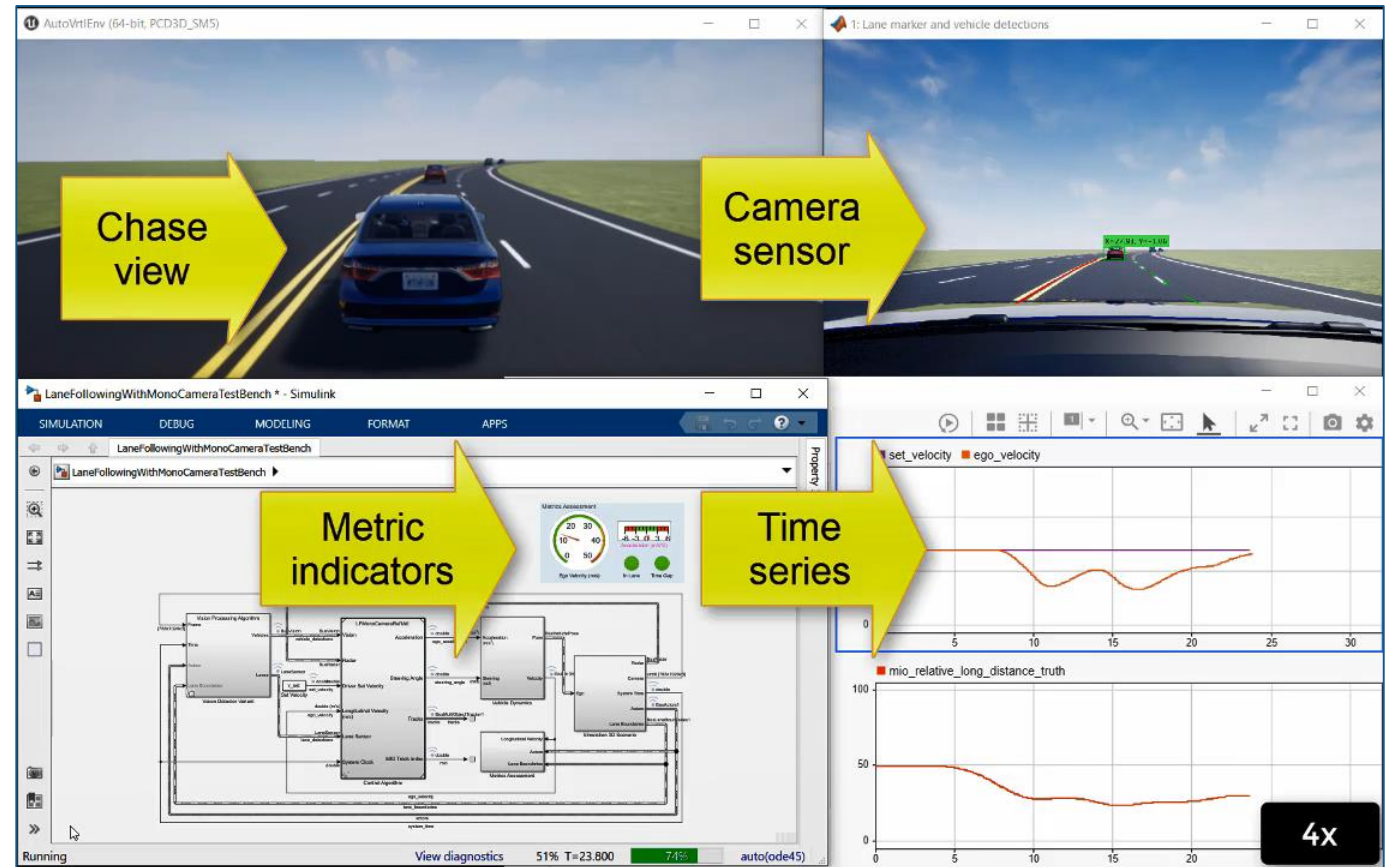
Lane-Following Control with Monocular Camera Perception

- 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™

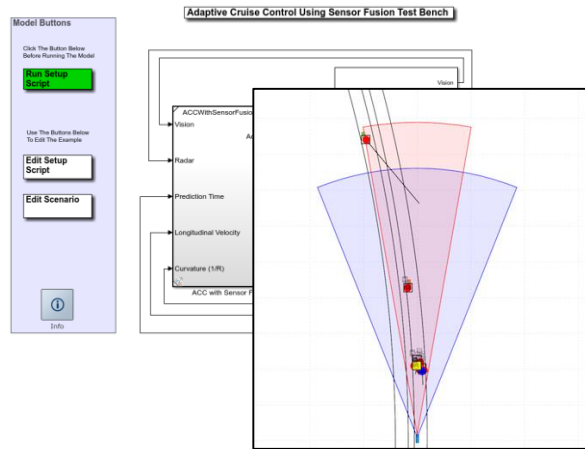
Vehicle Dynamics Blockset™



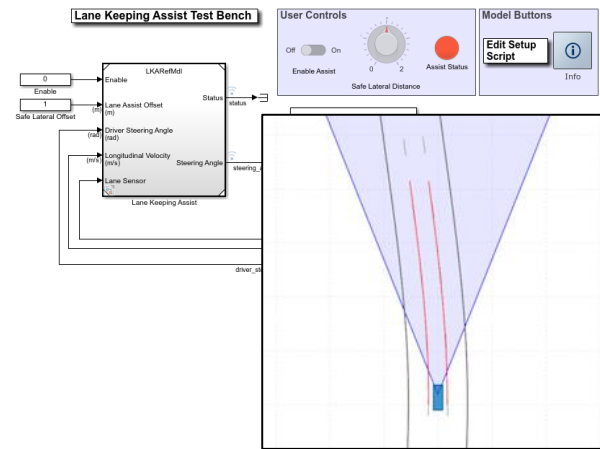
Updated **R2019b**

Design lateral and longitudinal Model Predictive Controllers

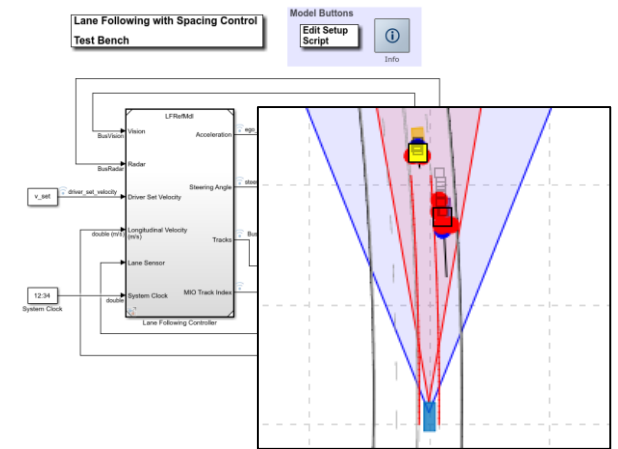
Longitudinal Control



Lateral Control



Longitudinal + Lateral



[Adaptive Cruise Control with Sensor Fusion](#)

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

R2017b

[Lane Keeping Assist with Lane Detection](#)

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

R2018a

[Lane Following Control with Sensor Fusion and Lane Detection](#)

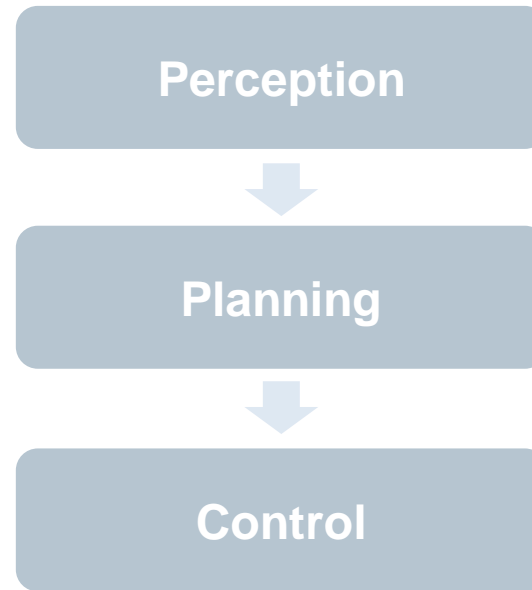
Automated Driving Toolbox™
 Model Predictive Control Toolbox™
 Embedded Coder®

R2018b

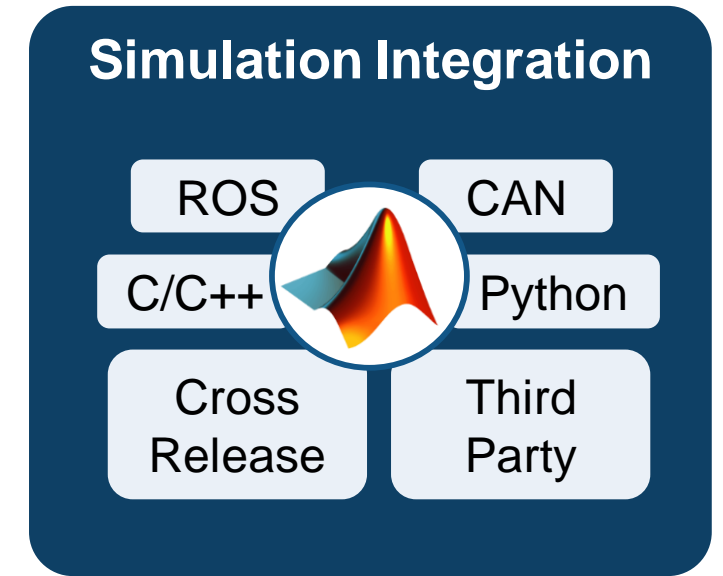
Some common questions from automated driving engineers



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



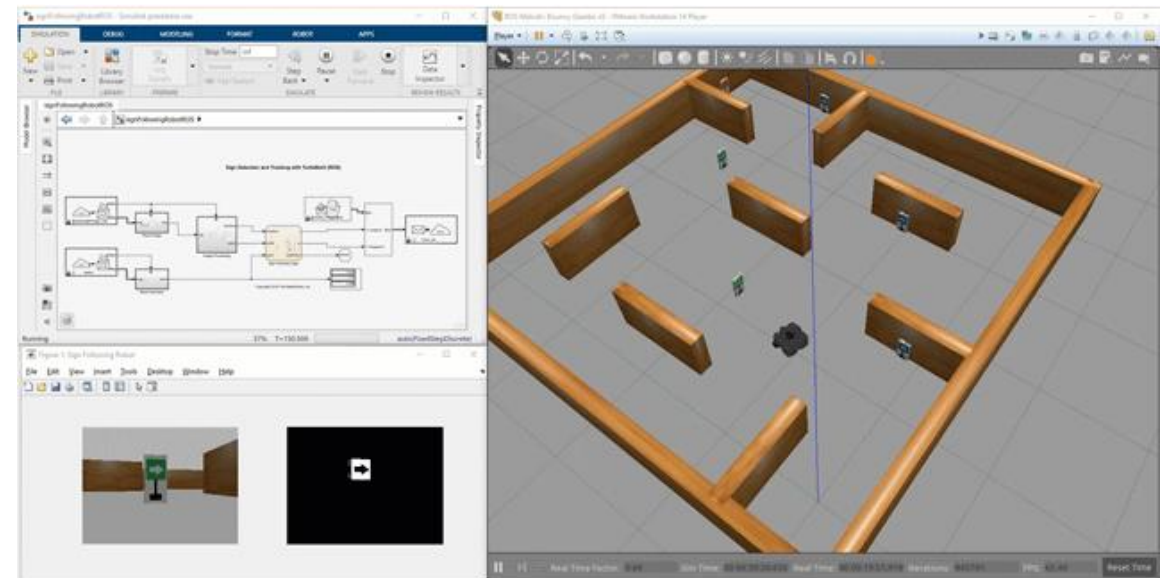
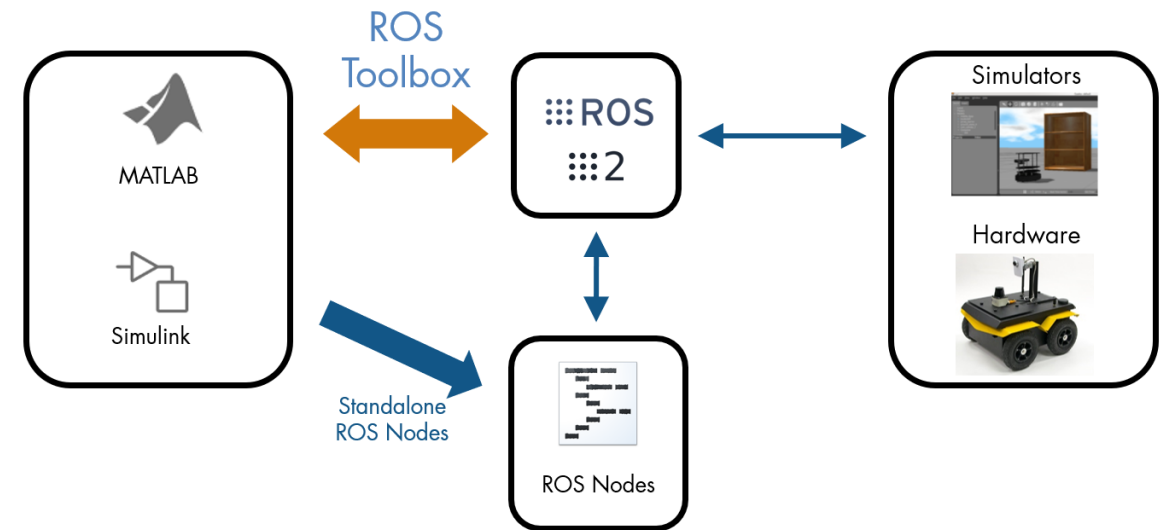
How can I **discover and design** in new domains?



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

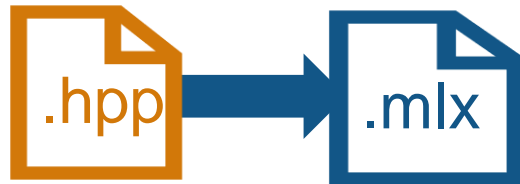
ROS Toolbox - NEW!

- 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++



[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](#)

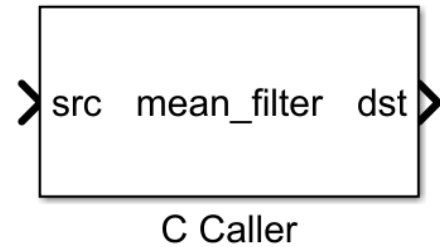
Computer Vision System Toolbox™

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

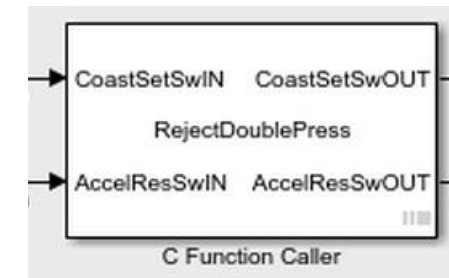
```
typedef struct {
    double coeff;
    double init;
    fault_T fault;
} params_T;
```

| 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

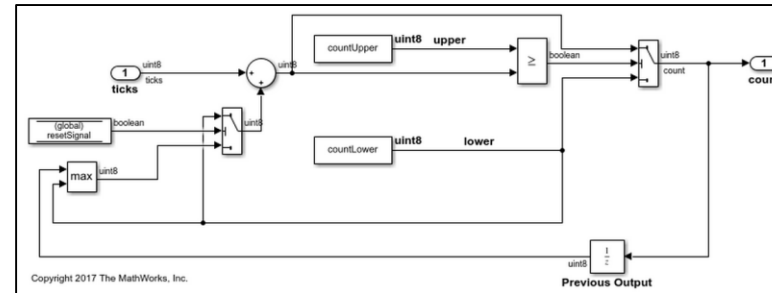
Integrate Generated Code by Using Cross-Release Workflow

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

Embedded Coder

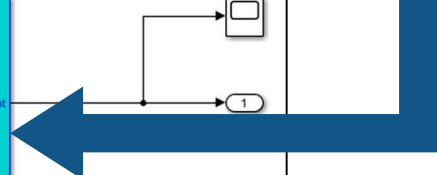
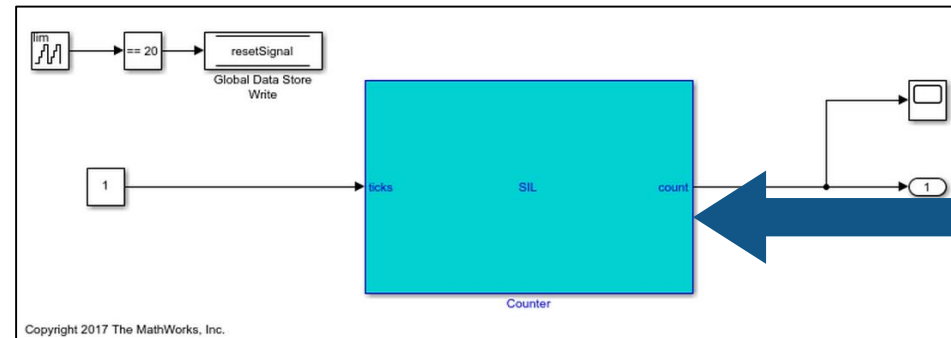
R2016a

Previous Release

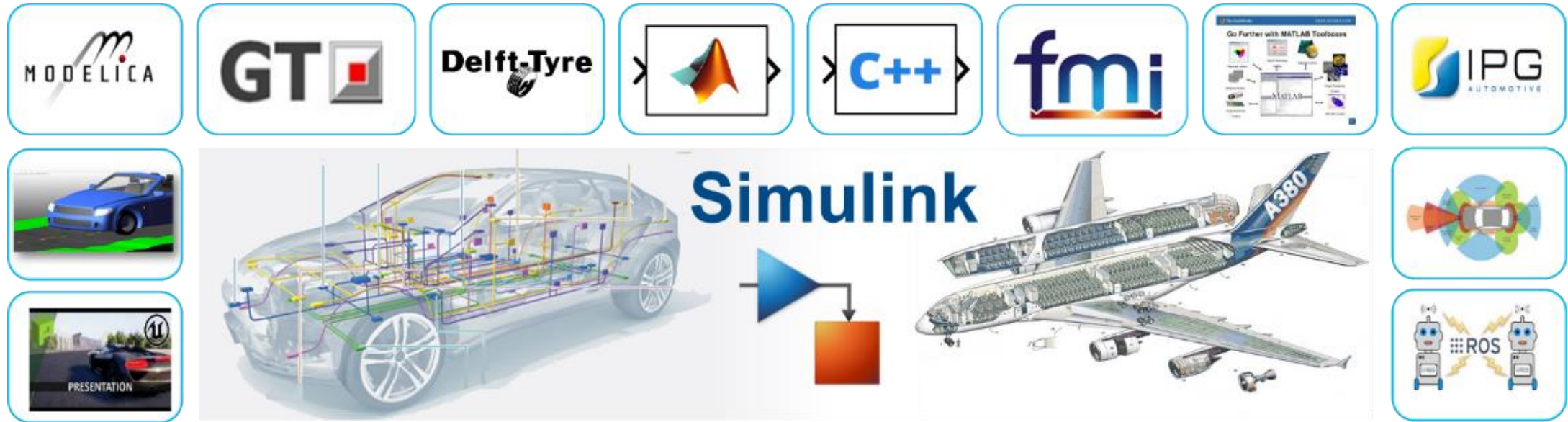


crossReleaseImport

Current Release



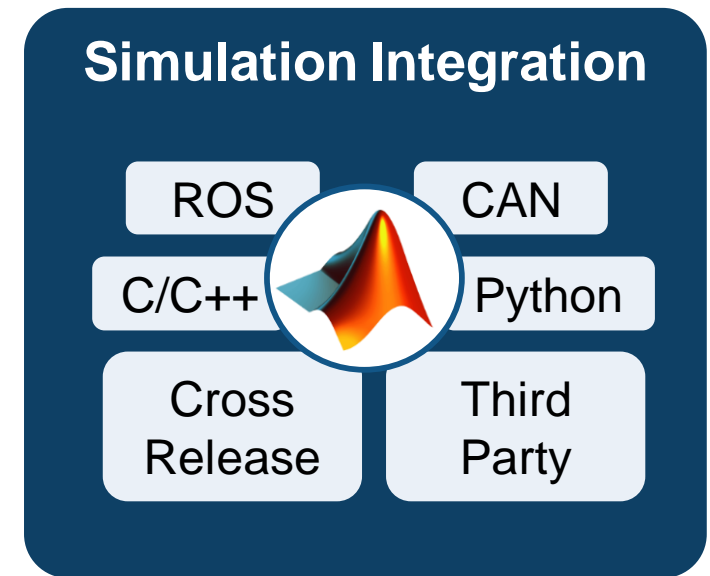
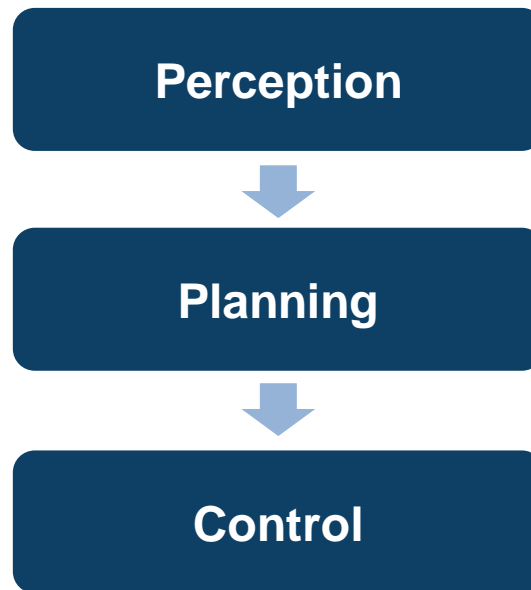
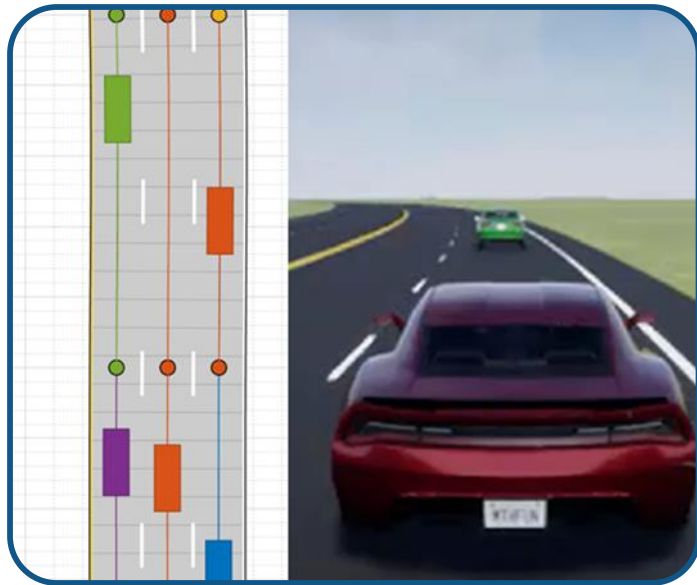
Connect to third party tools



152 Interfaces to 3rd 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

Thank You!