Capabilities of an Autonomous Vehicle
Capabilities of an Autonomous Vehicle
Capabilities of an Autonomous Vehicle

- Sense
- Perceive
- Decide & Plan
Capabilities of an Autonomous Vehicle

- Sense
- Perceive
- Decide & Plan
- Act
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?

Perception
Planning
Control

Simulation Integration
ROS
CAN
C/C++
Python
Cross Release
Third Party

Control
Planning
Perception

MathWorks
How can I design with virtual driving scenarios?

<table>
<thead>
<tr>
<th>Scenes</th>
<th>Cuboid</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>
## How can I design with virtual driving scenarios?

<table>
<thead>
<tr>
<th>Scenes</th>
<th><strong>Cuboid</strong></th>
<th><strong>3D Simulation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Cuboid Diagram" /></td>
<td><img src="image2.png" alt="3D Simulation Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Testing</th>
<th>Controls, sensor fusion, planning</th>
<th>Controls, sensor fusion, planning, perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoring</td>
<td>Driving Scenario Designer App Programmatic API (<code>drivingScenario</code>)</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™

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

Create Driving Scenario Variations Programmatically

Programmatically create variations of a driving scenario that was built using the Driving Scenario Designer app.
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>Scenarios</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™

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™*
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 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.

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.

Visualize Automated Parking Valet Using 3D Simulation
Visualize vehicle motion in a 3D simulation environment using an automated parking valet system constructed in Simulink.

Simulate Lidar Sensor Perception Algorithm
Develop a lidar perception algorithm using data recorded from a 3D simulation environment, and simulate within that environment.

Simulate Radar Sensors in 3D Environment
Implement a synthetic data simulation for tracking and sensor fusion using Simulink and a 3D simulation environment.

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

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.
Simulating automated driving systems with MATLAB and Simulink
Simulating automated driving systems with MATLAB and Simulink
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

- 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
- Target vehicle positions specified from Scenario Reader block
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?
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™
Interoperate with neural network frameworks

PyTorch

Caffe2

MXNet

Core ML

ONNX

Keras-Tensorflow

MATLAB

Caffe

(...)

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

Visual Perception Using Monocular Camera
Automated Driving Toolbox™
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
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™**
Design trackers

Multi-Object Tracker

- Association & Track Management
- Tracking Filter

Detections → Multi-object tracker

- Linear, extended, and unscented Kalman filters

Tracks

From various sensors at various update rates

Automated Driving Toolbox™
Design trackers

- 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

Detection → Multi-Object Tracker → Tracks

Vehicle 2

Detection → Multi-Object Tracker → Tracks
Design track level fusion systems

Vehicle 1
- Detections
- Multi-Object Tracker
- Track Fusion
- Tracks

Vehicle 2
- Detections
- Multi-Object Tracker
- Tracks
Design track level fusion systems
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
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</td>
<td>Synchronous Machine Modelling Using Simscape</td>
<td>Sensor Fusion and Tracking for Autonomous Systems</td>
<td>Simplifying Requirements-Based Verification with Model-Based Design</td>
<td>Predictive Maintenance with MATLAB</td>
</tr>
<tr>
<td></td>
<td>Samuel Bailey, Skyrad Consulting</td>
<td>Peeni Reni, Cummins Generator Technologies</td>
<td>Marc Willerton, MathWorks</td>
<td>Fraser Macmillan, MathWorks</td>
<td>Phil Rother, MathWorks</td>
</tr>
<tr>
<td>16:15</td>
<td>Industrial IoT and Digital Twins</td>
<td>Developing Fit-For-Purpose Simscape Models to Support System and Control Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coorous Mohtadi, MathWorks</td>
<td>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?

Perception

Planning

Control

Simulation Integration

ROS
CAN
C/C++
Python
Cross Release
Third Party

ROS
C/C++
Python
Cross Release
Third Party
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?

Perception
Planning
Control

Simulation Integration
- ROS
- CAN
- C/C++
- Python
- Cross Release
- Third Party

MathWorks
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
Lane Following Controller Algorithm
Components of lane following with spacing control algorithm
Goal

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

\[ V_{ego}, \ a \]

\[ D_{relative} \]

\[ V_{mio} \]

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

- Slow down the ego vehicle when road is curvy.
Model predictive control (MPC)

- Measured outputs
- Manipulated variables
- References
- Measured disturbances
- Optimizer
- Plant Model
- MPC controller
- Ego Vehicle
MPC for Lane Following Control

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

**Manipulated variables**
- Acceleration ($a$)
- Steering angle ($\delta$)

**References**
- Ego velocity set point ($V_{\text{set}}$)
- Target lateral deviation (=0)

**Measured disturbances**
- MIO velocity ($V_{\text{mio}}$)
- Previewed road curvature ($\rho$)

**minimize:**
$$w_1|V_{\text{ego}} - V_{\text{set}}|^2 + w_2|E_{\text{lateral}}|^2$$

**subject to:**
- $D_{\text{relative}} \geq D_{\text{safe}}$
- $a_{\text{min}} \leq a \leq a_{\text{max}}$
- $\delta_{\text{min}} \leq \delta \leq \delta_{\text{max}}$
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)

Inputs to MPC: For lateral control

- MPC: Path following controller
Path Following Control Block

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

- Measurement
  - Longitudinal acceleration
  - Steering angle

Virtual lane
- For lane change

Model Predictive Control Toolbox™

- Bicycle model parameters
- Delay or latency in the system
- Disable distance keeping

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)]

- Vehicle parameters
  - Total mass (kg): 1575
  - Yaw moment of inertia (mN·m²): 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): 13

- Transport lag between model inputs and outputs (s): 0

- Spacing Control
  - Maintain safe distance between lead vehicle and ego vehicle
    - Default spacing (m): 10
Path Following Control Block

Actuator limits

Tune MPC performance
Path Following Control Block

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

Change MPC design
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®
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++**

```
.hpp -> .mlx
```

**Call Python**

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

**Call OpenCV & OpenCV GPU**

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

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

MATLAB®

**Call Python from MATLAB**

MATLAB®

R2014a

**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

Create buses from C structs

Test and verify C code

Bring Custom Image Filter Algorithms as Reusable Blocks in Simulink

Simulink®

R2017b

Import Structure and Enumerated Types

Simulink®

R2017a

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
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

Perception

Planning

Control

Simulation Integration

ROS

CAN

C/C++

Python

Cross Release

Third Party

MathWorks
Thank You!