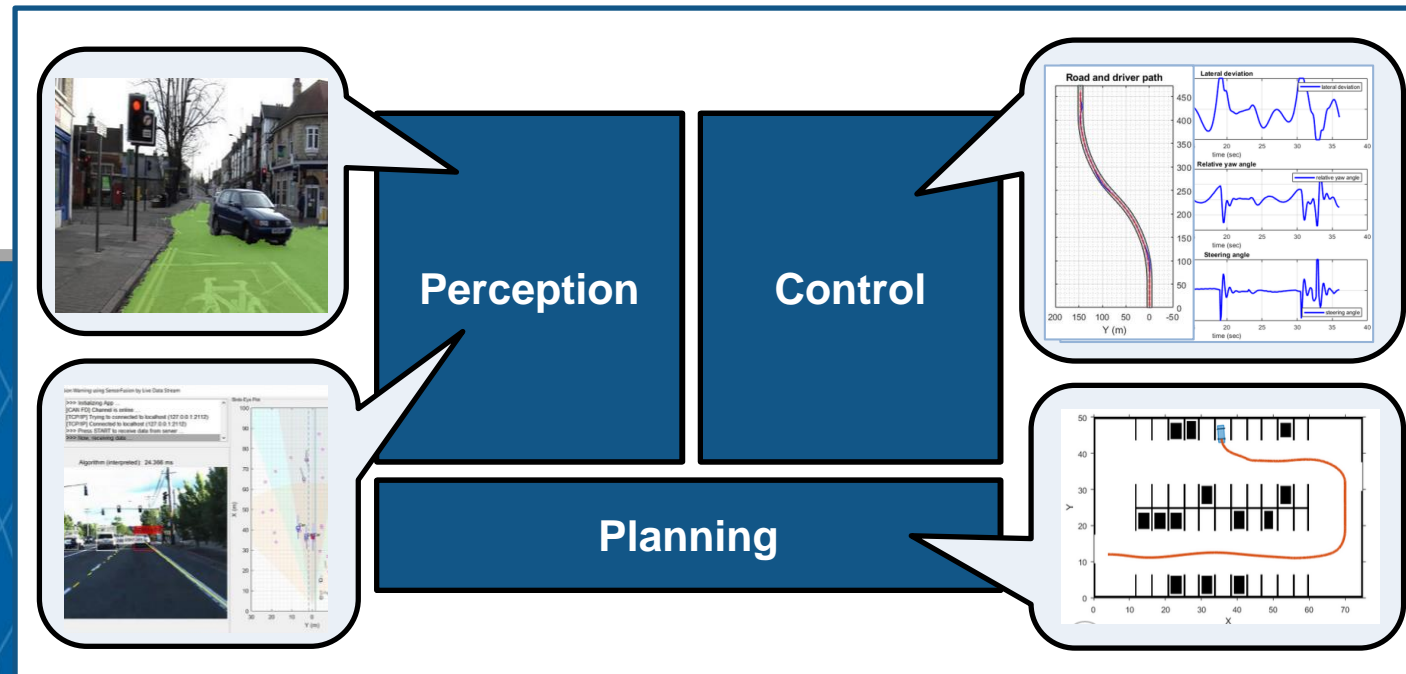


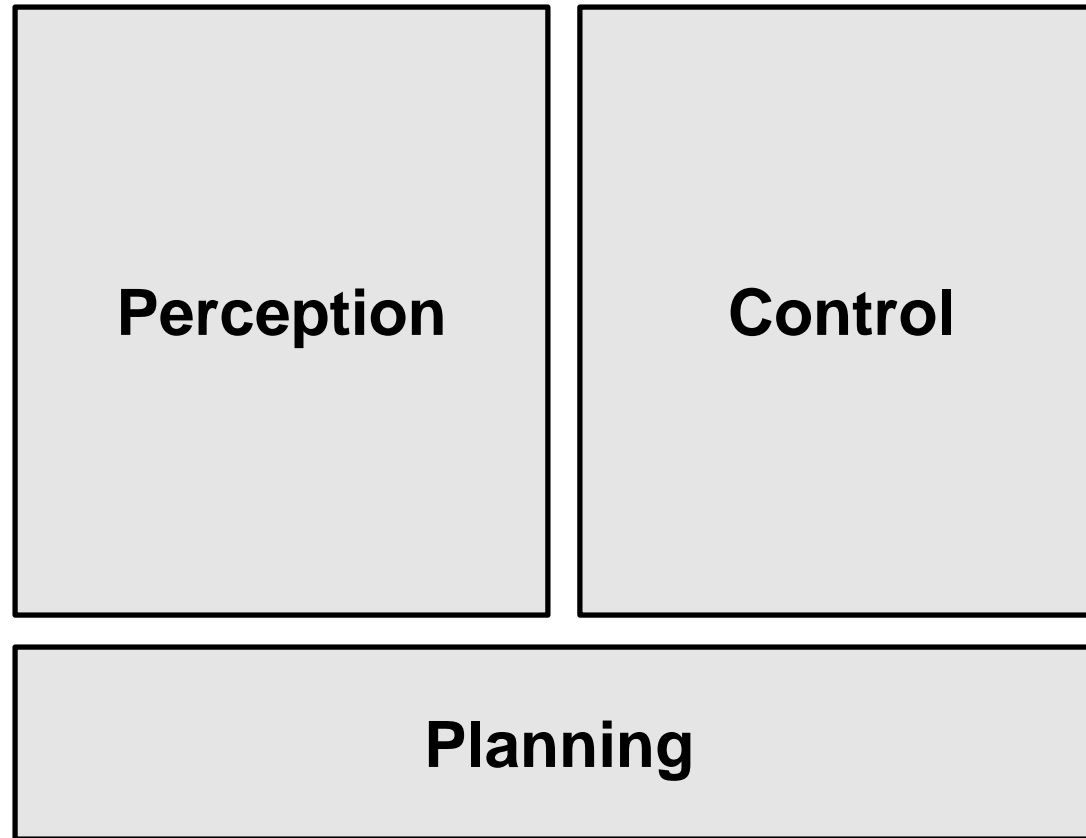
Automated Driving

with MATLAB® and Simulink®

Mark Corless
Automated Driving Segment Manager
Industry Marketing
2018-04-05



How can you use MATLAB and Simulink to develop automated driving algorithms?



Examples of how you can use MATLAB and Simulink to develop automated driving algorithms

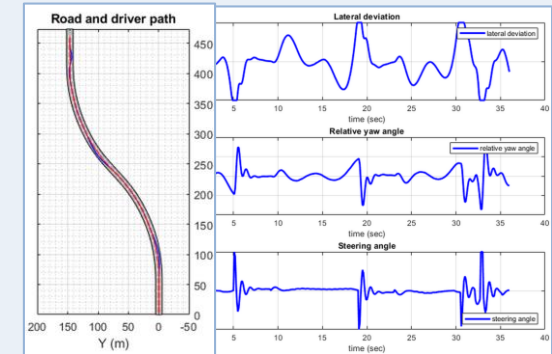
Deep learning



Perception

Control

Sensor models & model predictive control

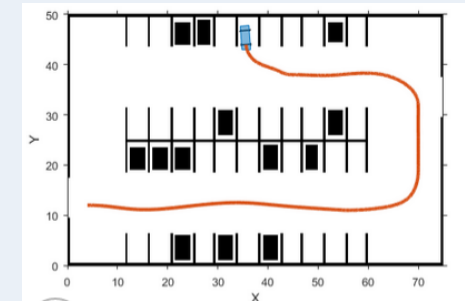


Sensor fusion with live data



Planning

Path planning



How can you use MATLAB and Simulink to develop perception algorithms?

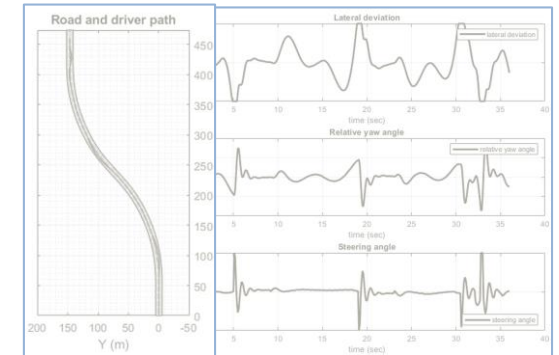
Deep learning



Perception

Control

Sensor models & model predictive control

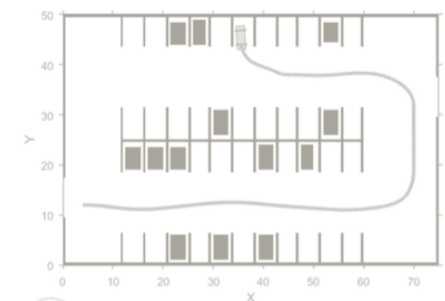


Sensor fusion with live data



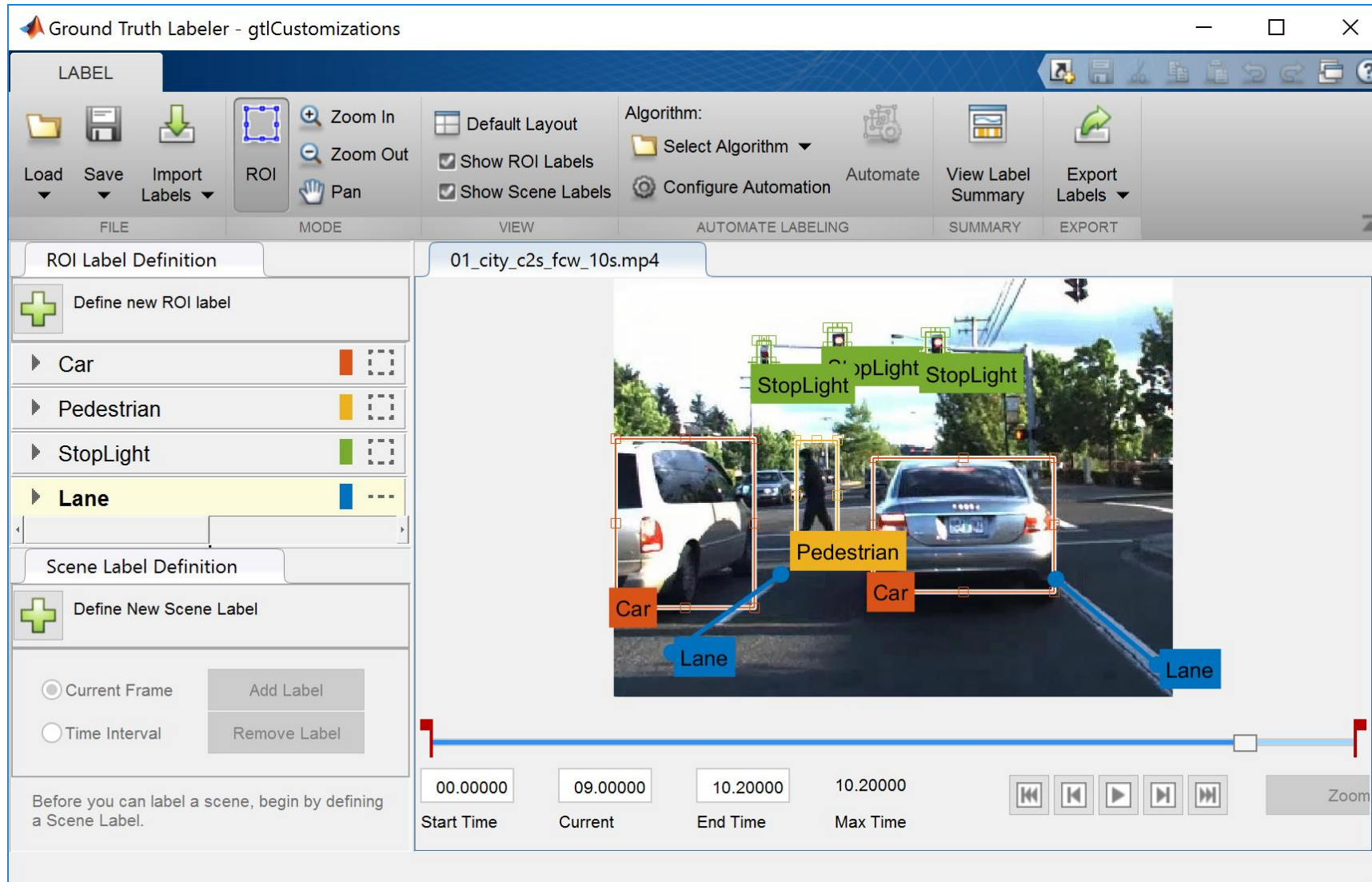
Planning

Path planning

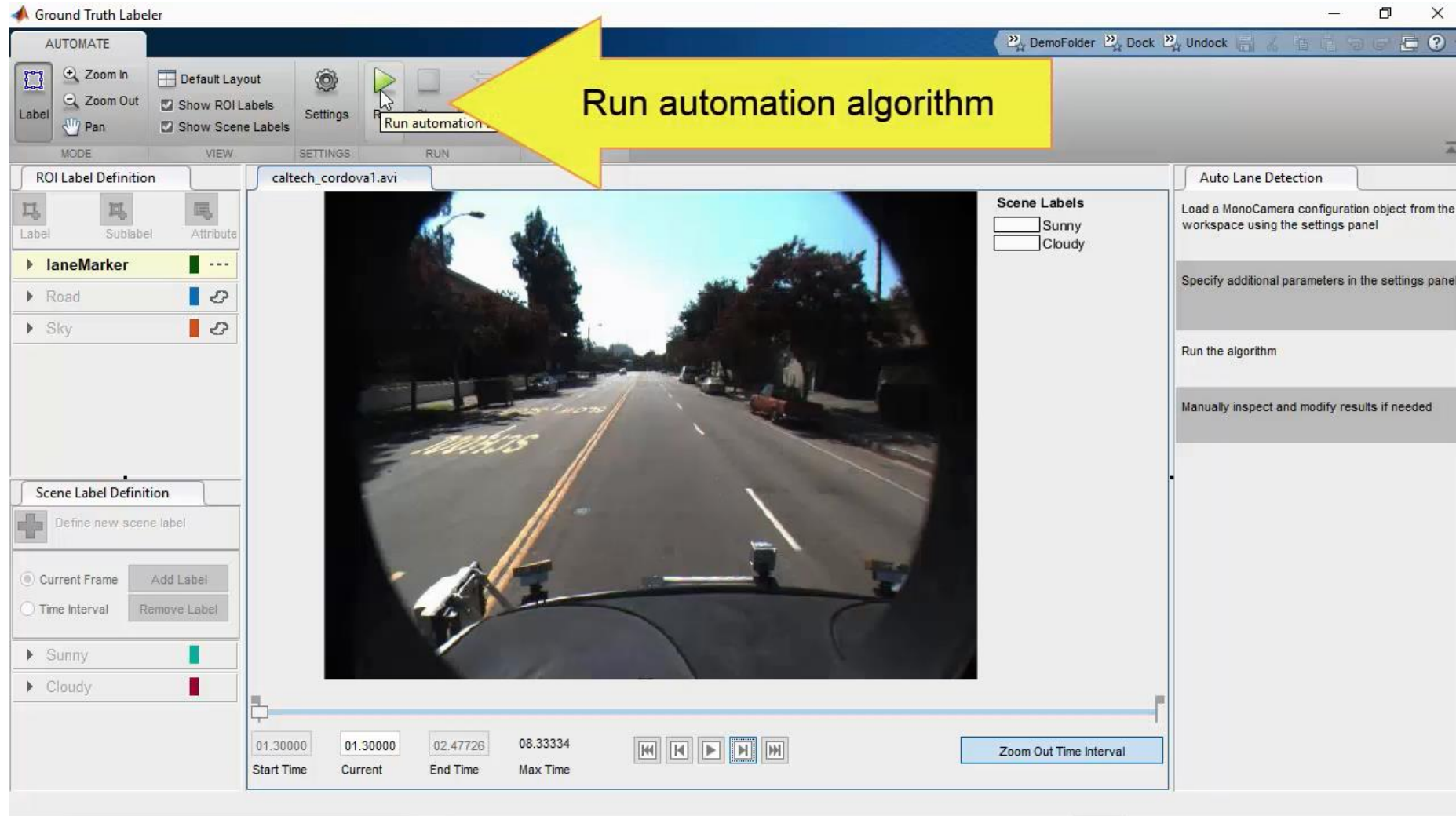


Automated Driving System Toolbox introduced: Ground Truth Labeling App to label video data

R2017a



Automate labeling lanes with Ground Truth Labeler



Specify attributes and sublabels in Ground Truth Labeler App

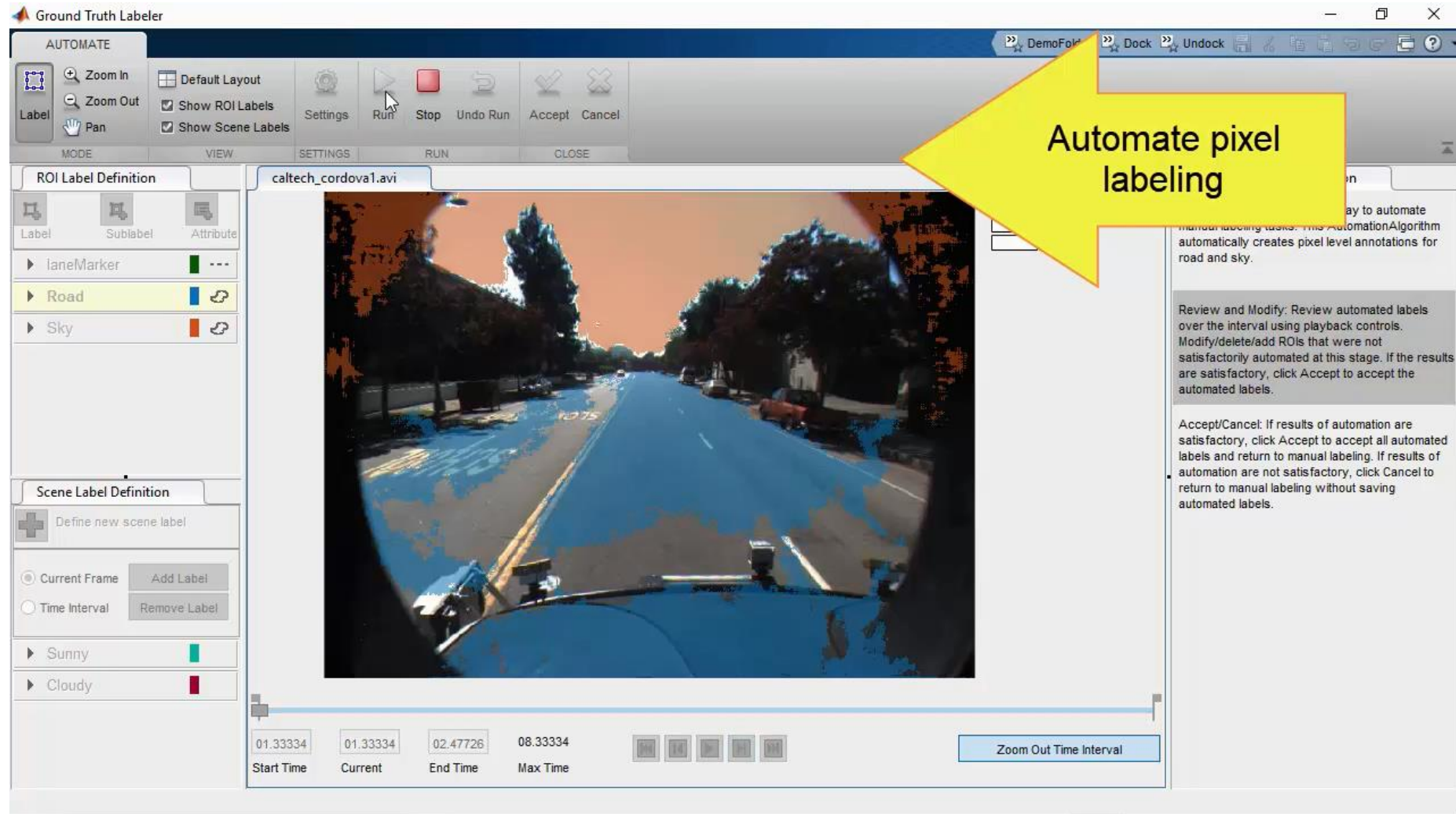
R2018a

The screenshot illustrates the Ground Truth Labeler App interface, which is used for defining and labeling video data. The interface is divided into several sections:

- ROI Label Definition:** This section allows users to define labels and sublabels. It includes a list of labels (cyclist, bicycle, vehicle) and their corresponding colors and bounding box styles. A blue arrow points from this section to the video playback area.
- Scene Label Definition:** This section allows users to define scene labels. It includes a list of scene labels (bicycle, cyclist) and their corresponding colors and bounding box styles. A blue arrow points from this section to the video playback area.
- Video Playback:** The central area shows a video frame with labeled objects. A blue arrow points from the video playback area to the ROI Label Definition section.
- Attributes and Sublabels:** This section allows users to specify attributes and sublabels for the labeled objects. It includes a list of attributes (bikeType, action) and their corresponding values (bicycle, inMotion). A blue arrow points from this section to the video playback area.

The video frame shows a street scene with a cyclist and a car. The cyclist is labeled with a yellow bounding box and the text "bicycle cyclist". The car is labeled with a blue bounding box and the text "vehicle".

Automate labeling pixels with Ground Truth Labeler



Learn how to train a deep learning network using this example

R2017b



Semantic Segmentation Using Deep Learning

- Train free space detection network using deep learning

Computer Vision
System Toolbox™

Examples

Search Help



Semantic Segmentation Using Deep Learning

This example shows how to train a semantic segmentation network using deep learning.

A semantic segmentation network classifies every pixel in an image, resulting in an image that is segmented by class. Applications for semantic segmentation include road segmentation for autonomous driving and cancer cell segmentation for medical diagnosis. To learn more, see [Semantic Segmentation Basics](#).

To illustrate the training procedure, this example trains SegNet [1], one type of convolutional neural network (CNN) designed for semantic image segmentation. Other types networks for semantic segmentation include [fully convolutional networks \(FCN\)](#) and [U-Net](#). The training procedure shown here c

This example also uses:
[Neural Network Toolbox](#)
[vgg16](#)

[Open Script](#)

This example uses the C...
street-level views obtaine

Learn more about

Setup

This example creates the
[Toolbox™ Model for VG](#)

```
vgg16();
```

Add-On Explorer

Manage Add-Ons

Search for add-ons



Neural Network Toolbox Model for VGG-16 Network

version 17.2.0.0 by [MathWorks Neural Network Toolbox Team](#)

Pretrained VGG-16 network model for image classification

MathWorks Feature

★★★★★ 4 Ratings

125 Downloads ⓘ

Updated 14 Jun 2017

[Install ▾](#)
[Overview](#)

Load and plot training images

```
% Create datastore for images  
imds = imageDatastore(imgDir);  
I = readimage(imds, 1);  
I = histeq(I);  
imshow(I)
```



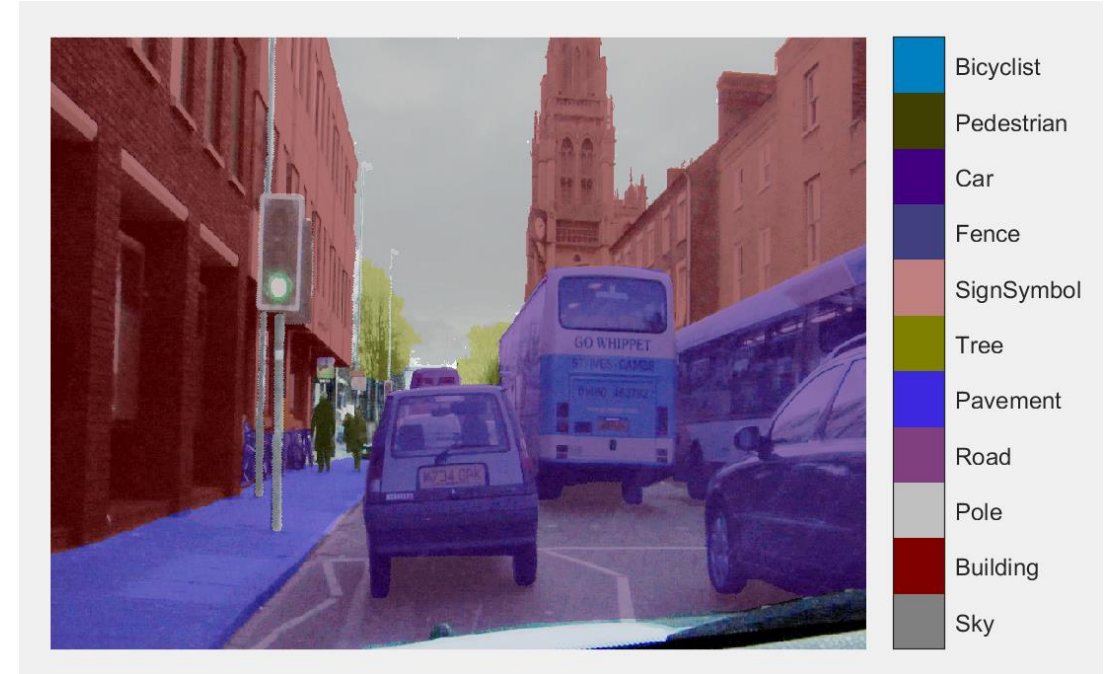
imageDatastore
manages large collections
of images

Load and overlay pixel labels

```
% Load pixel labels
classes = ["Sky"; "Building";...
          "Pole"; "Road"; "Pavement"; "Tree";...
          "SignSymbol"; "Fence"; "Car";...
          "Pedestrian"; "Bicyclist"];

pxds = pixelLabelDatastore(...
    labelDir, classes, labelIDs);

% Display labeled image
C = readimage(pxds, 1);
cmap = camvidColorMap;
B = labeloverlay(I, C, 'ColorMap', cmap);
imshow(B)
```



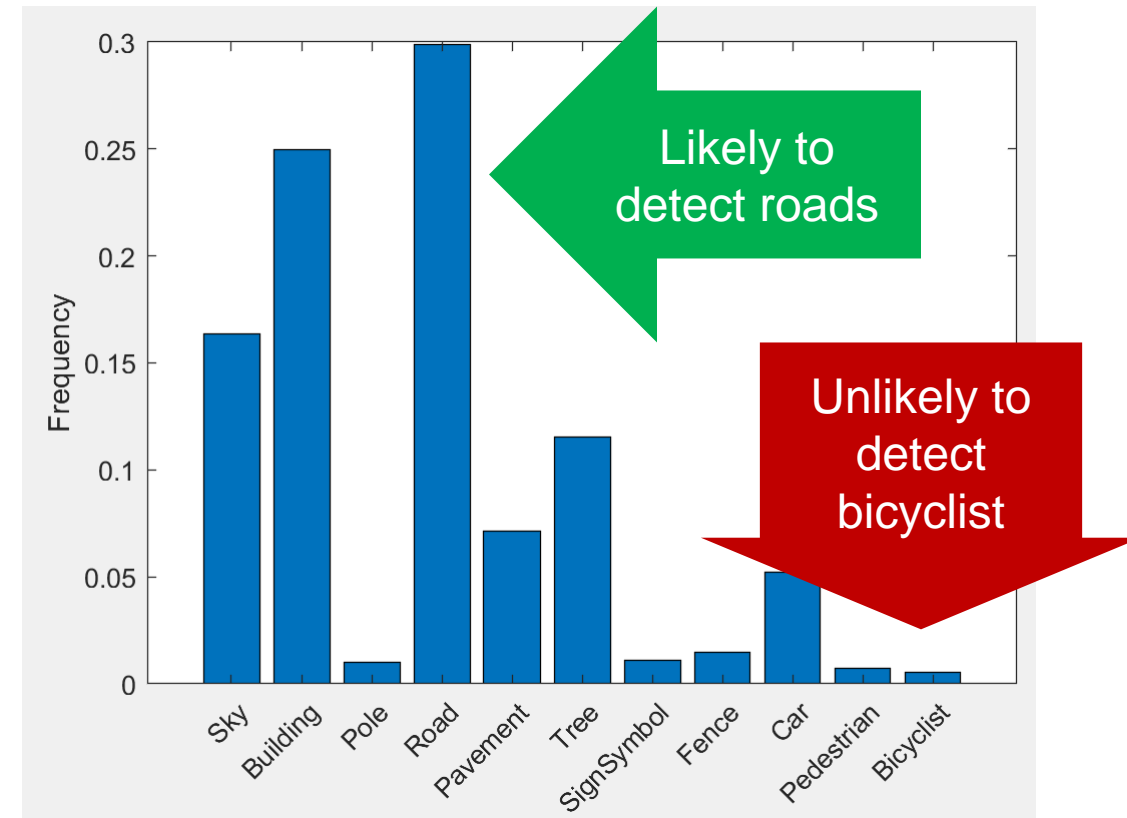
pixelLabelDatastore
manages large collections
of pixel labels

Visualize distribution of labeled pixels

```
% Visualize label count by class
tbl = countEachLabel(pxds)

frequency = tbl.PixelCount / ...
           sum(tbl.PixelCount);

bar(1:numel(classes), frequency)
xticks(1:numel(classes))
xticklabels(tbl.Name)
xtickangle(45)
ylabel('Frequency')
```



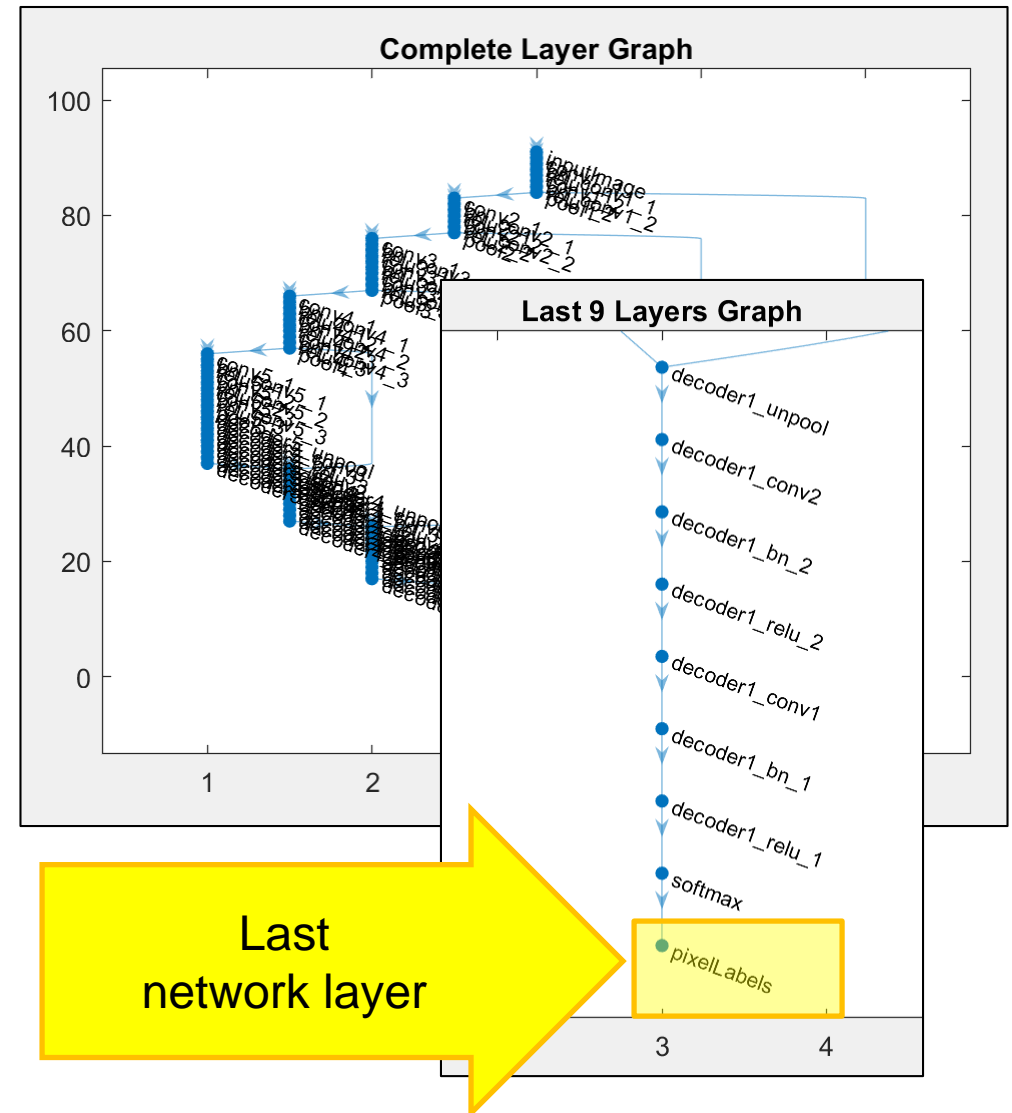
Labeled pixels in this set are imbalanced

Create and visualize baseline network

```
% Create SegNet architecture
lgraph = segnetLayers(...
    imageSize, numClasses,...
    'vgg16');

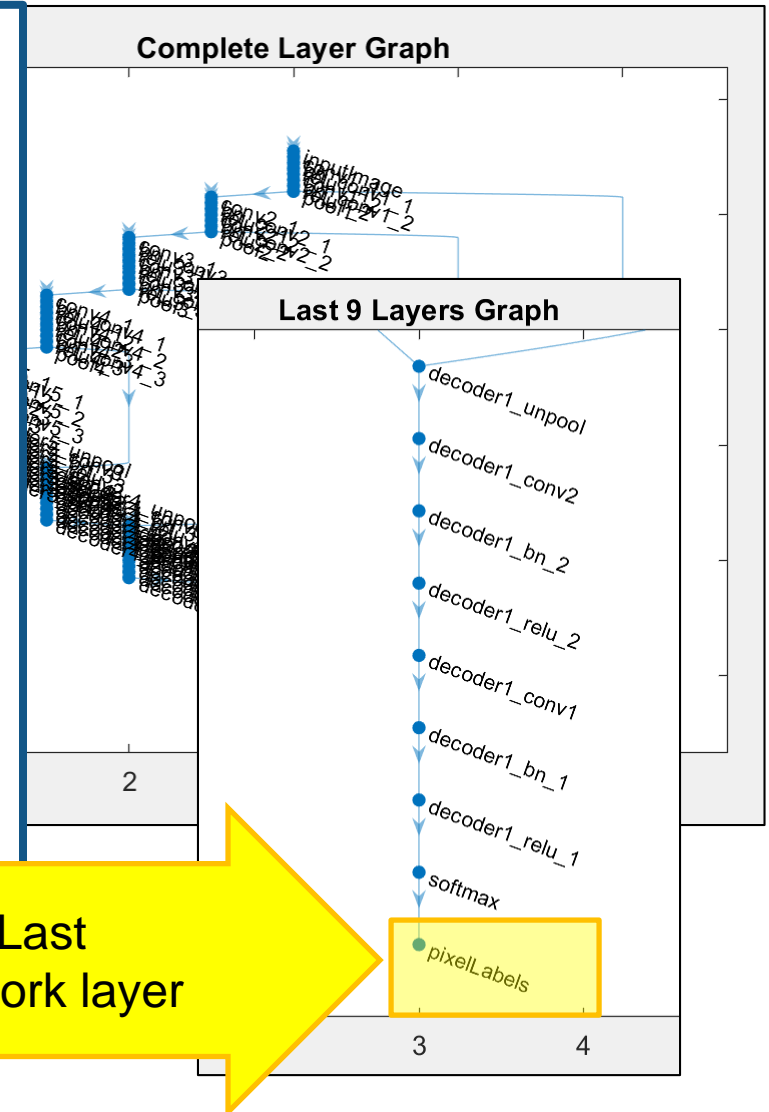
% Display network structure
plot(lgraph)
title('Complete Layer Graph')%

% Display last layers
plot(lgraph); ylim([0 9.5])
title('Last 9 Layers Graph')
```



Add weighted layer to compensate for imbalanced data set

```
% Create weighted layer
pxLayer = pixelClassificationLayer(...
    'Name','weightedLabels', 'ClassNames',tbl.Name,...
    'ClassWeights',classWeights)
```

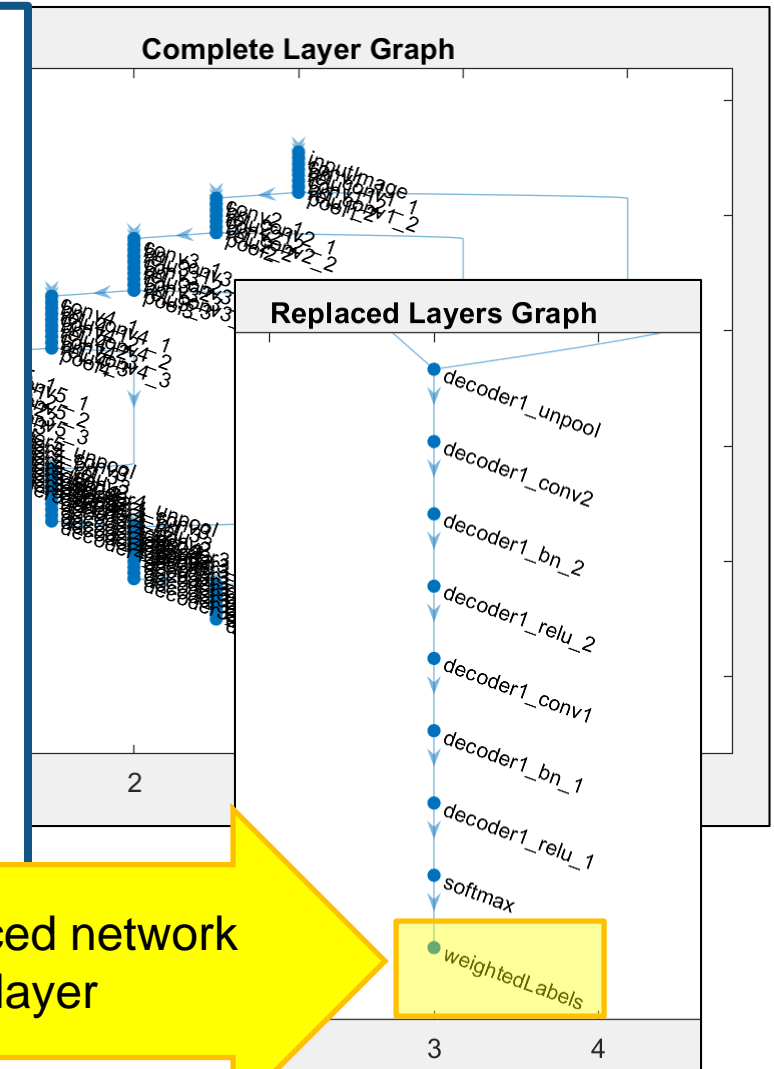


Add weighted layer to compensate for imbalanced data set

```
% Create weighted layer
pxLayer = pixelClassificationLayer(...
    'Name', 'weightedLabels', 'ClassNames', tbl.Name, ...
    'ClassWeights', classWeights)

% Replace layer
lgraph = removeLayers(lgraph, 'pixelLabels');
lgraph = addLayers(lgraph, pxLayer);
lgraph = connectLayers(lgraph, ...
    'softmax', 'weightedLabels');

% Display network structure
plot(lgraph); ylim([0 9.5])
title('Replaced Layers Graph')
```



Augment images to expand training set

```
augmenter = imageDataAugmenter(...  
    'RandXReflection', true,...  
    'RandRotation',    [-30 30],... % degrees  
    'RandXTranslation', [-10 10],... % pixels  
    'RandYTranslation', [-10 10]); % pixels  
  
datasource = pixelLabelImageSource(...  
    imdsTrain,... % Image datastore  
    pxdsTrain,... % Pixel datastore  
    'DataAugmentation', augmenter)
```



Deep learning on CPU, GPU, multi-GPU and clusters

```
options = trainingOptions('sgdm', ...  
    'InitialLearnRate', 1e-3, ...  
    'MaxEpochs', 100, ...  
    'MiniBatchSize', 4, ...  
    'Shuffle', 'every-epoch', ...  
    'VerboseFrequency', 2, ...  
    'ExecutionEnvironment', 'auto');
```



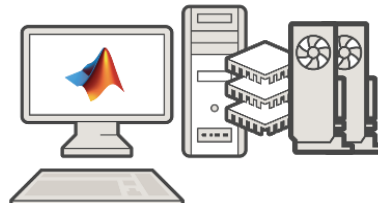
Single CPU

'auto'



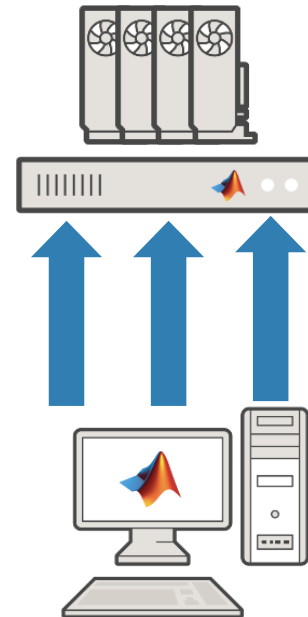
Single CPU
Single GPU

'auto'



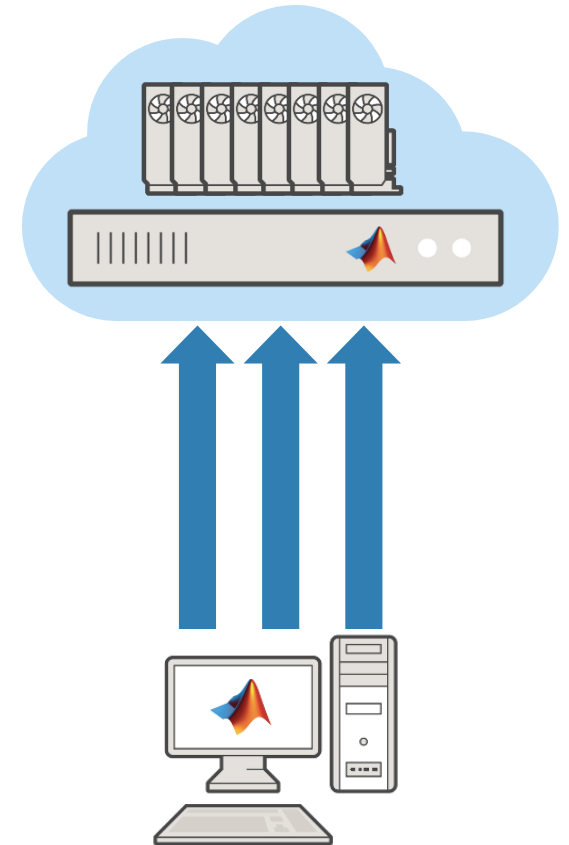
Single CPU
Multiple GPUs

'multi-gpu'



On-prem server with
GPUs

'parallel'

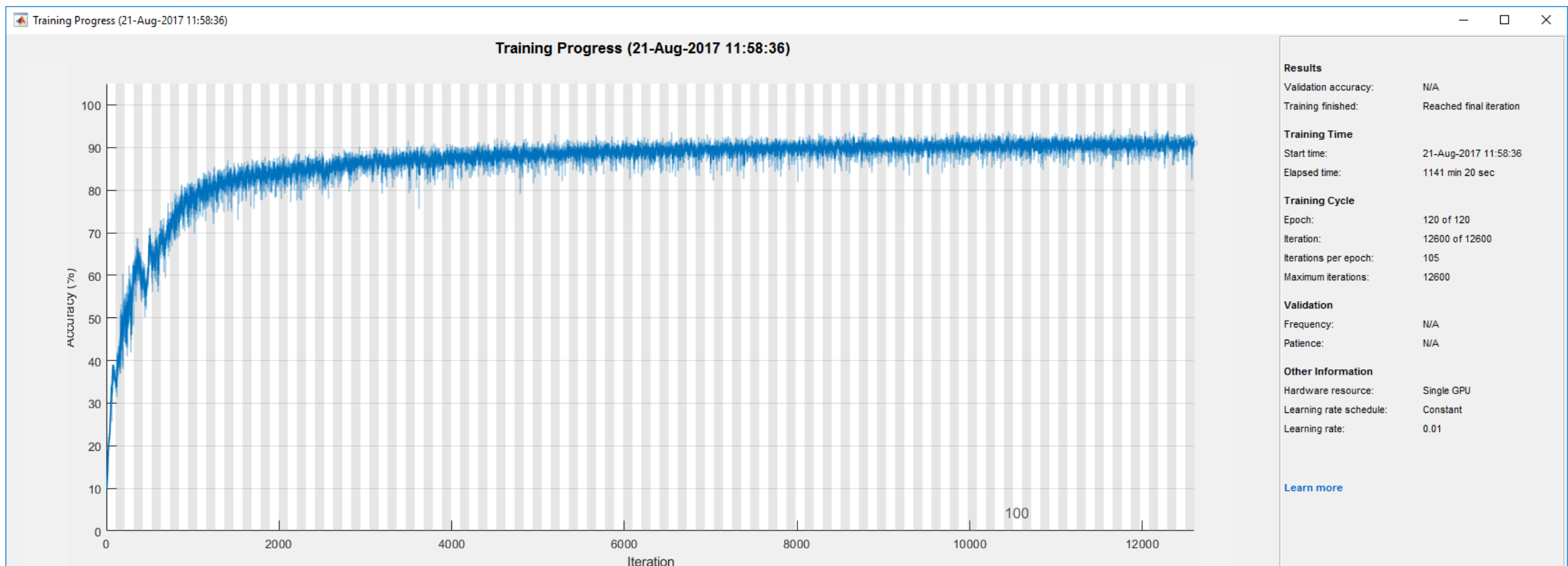


Cloud GPUs
(AWS, Azure, etc.)

'parallel'

Train network and view progress

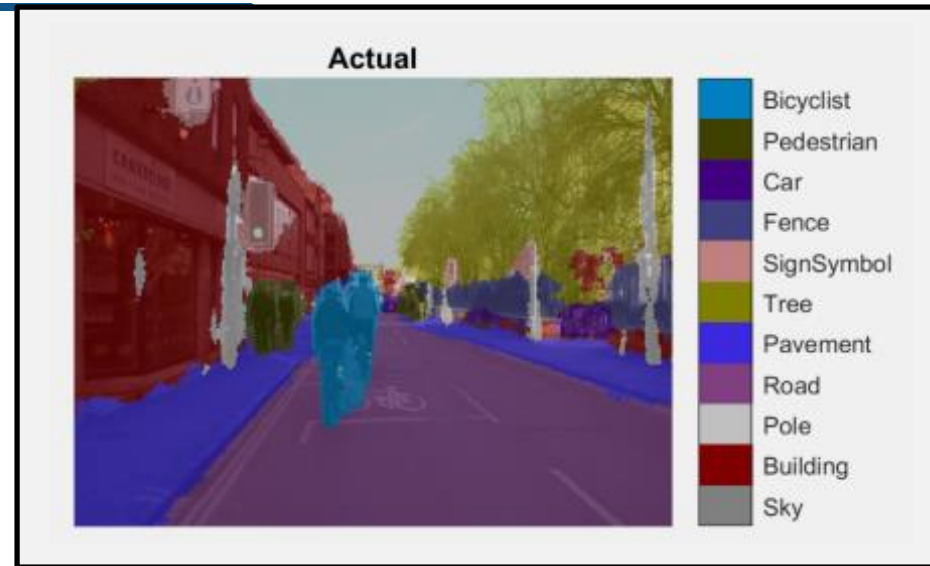
```
[net, info] = trainNetwork(datasource, lgraph, options);
```



Evaluate trained network on image

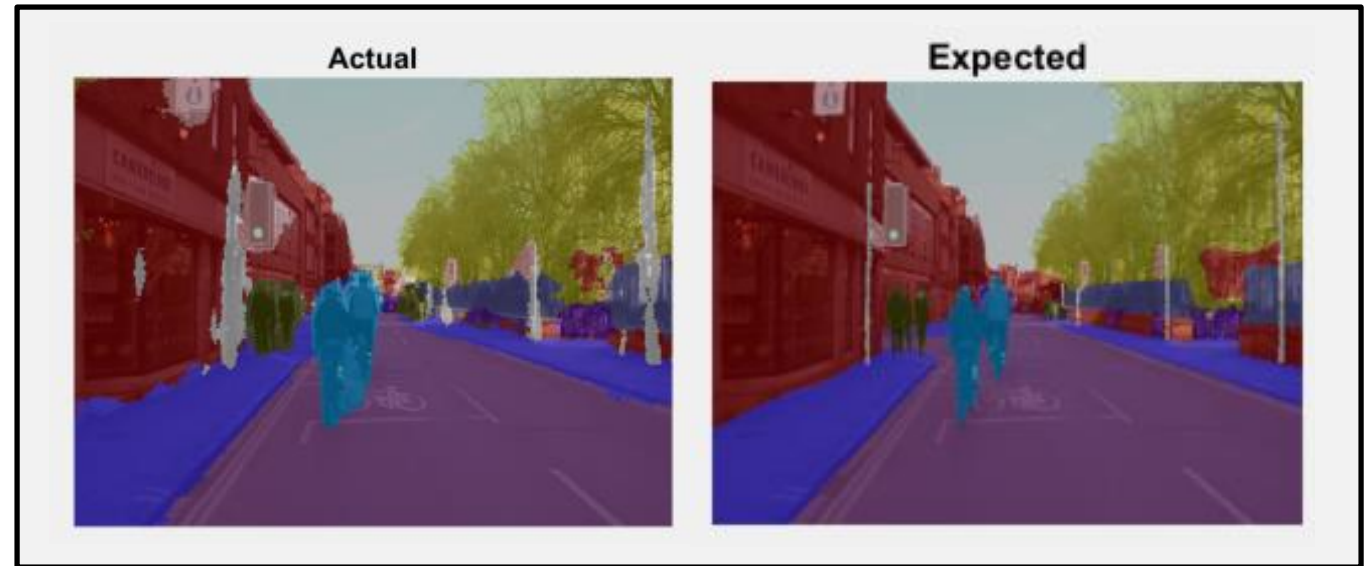
```
% Plot actual results
I = read(imdsTest);
actual = semanticseg(I, net);

B = labeloverlay(I, ...
    actual,...
    'Colormap', cmap,...
    'Transparency',0.4);
imshow(B)
pixelLabelColorbar(cmap, classes);
title('Actual')
```



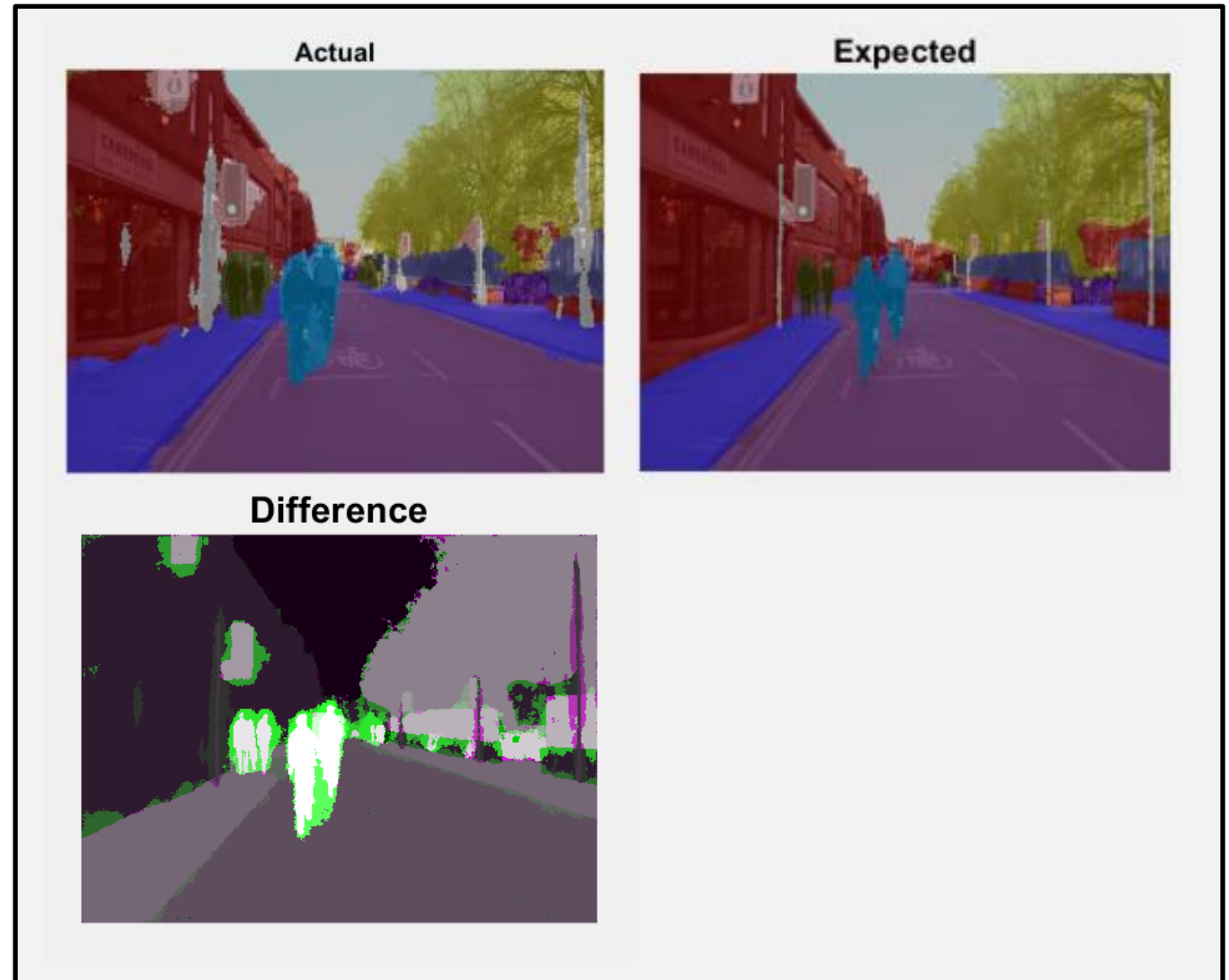
Visually compare actual with original labeled results

```
% Plot expected results  
% using original labels  
expected = read(pxdstest);  
E = labeloverlay(I,...  
    expected,...  
    'Colormap', cmap,...  
    'Transparency',0.4);  
imshow(E)  
title('Expected');
```



Visually compare actual with original labeled results

```
% Plot differences  
imshowpair(...  
    uint8(actual),...  
    uint8(expected));  
title('Difference');
```

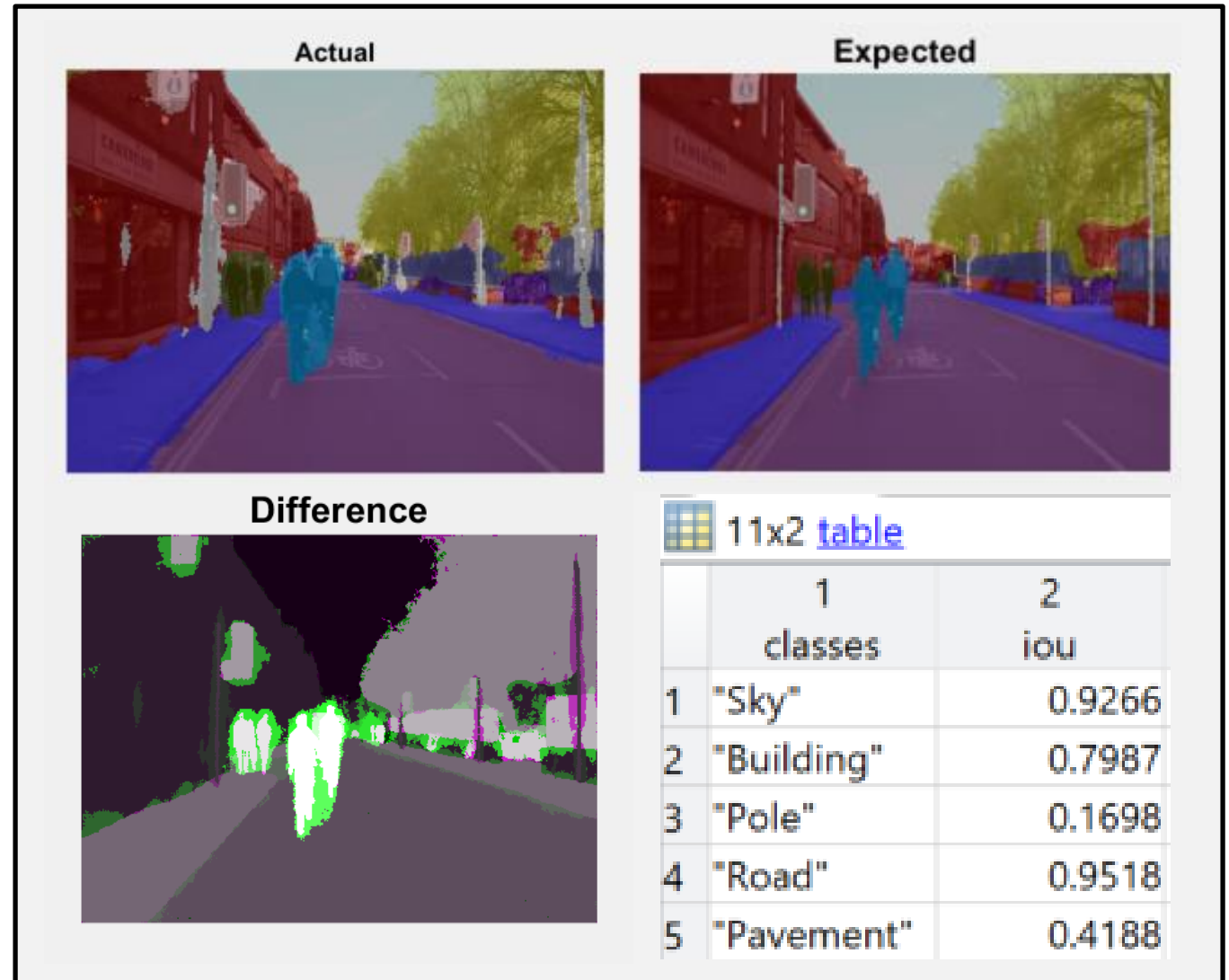


Assess similarity using intersection-over-union (IoU) metric

```
iou = jaccard(actual,...
               expected);
table(classes,iou)
```

```
ans =
    11x2 table
      classes      iou
    _____  _____

    "Sky"         0.92659
    "Building"     0.7987
    "Pole"         0.16978
    "Road"         0.95177
    "Pavement"     0.41877
    "Tree"         0.43401
    "SignSymbol"   0.32509
    "Fence"        0.492
    "Car"          0.068756
    "Pedestrian"   0
    "Bicyclist"    0
```



Evaluate trained network statistics

```
pxdsResults = ...
    semanticseg(...
        imdsTest,net,...
        'WriteLocation',tempdir,...
        'Verbose',false);

metrics = ...
    evaluateSemanticSegmentation(...
        pxdsResults,pxdsTest,...
        'Verbose',false);

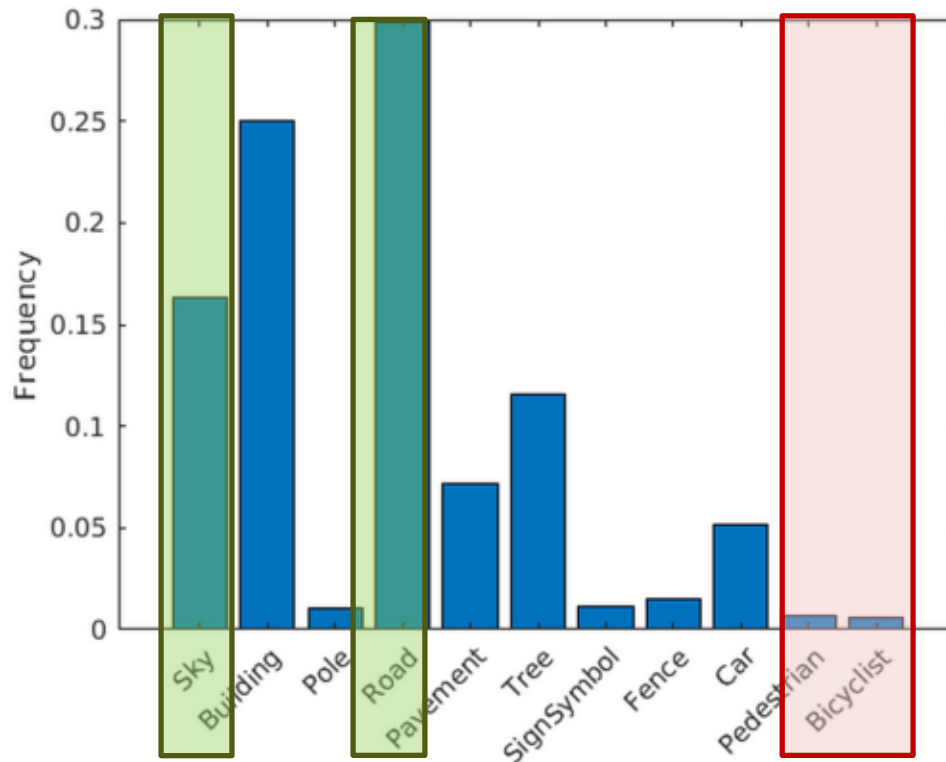
metrics.ClassMetrics
```

Evaluation metrics of network

	Accuracy	IoU	MeanBFScore
	—————	—————	—————
Sky	0.93544	0.89279	0.88239
Building	0.79978	0.75543	0.59861
Pole	0.73166	0.18361	0.51426
Road	0.93644	0.90663	0.7086
Pavement	0.90624	0.72932	0.70585
Tree	0.86587	0.73694	0.67097
SignSymbol	0.76118	0.35339	0.44175
Fence	0.83258	0.49648	0.50265
Car	0.90961	0.75263	0.64837
Pedestrian	0.83751	0.35409	0.46796
Bicyclist	0.84156	0.5472	0.46933

Distribution of labels in data affects intersection-over-union (IoU)

Distribution of labels in original data set



Evaluation metrics of network

	Accuracy	IoU	MeanBFScore
Sky	0.93544	0.89279	0.88239
Building	0.79978	0.75543	0.59861
Pole	0.73166	0.18361	0.51426
Road	0.93644	0.90663	0.7086
Pavement	0.90624	0.72932	0.70585
Tree	0.86587	0.73694	0.67097
SignSymbol	0.76118	0.35339	0.44175
Fence	0.83258	0.49648	0.50265
Car	0.90961	0.75263	0.64837
Pedestrian	0.83751	0.35409	0.46796
Bicyclist	0.84156	0.5472	0.46933

Underrepresented classes such as Pedestrian and Bicyclist are not segmented as well as classes such as Sky and Road

Generate CUDA Code for Embedded Deployment

```
% Save network to MAT file  
save('SegNet.mat', 'net')
```

```
function out = segnet_predict(in) %#codegen  
persistent mynet;  
if isempty(mynet)  
    mynet = coder.loadDeepLearningNetwork('SegNet.mat');  
end  
out = predict(mynet,in);
```

```
% Generate CUDA code  
cfg = coder.config('lib');  
cfg.TargetLang = 'C++';  
codegen -config cfg segnet_predict -args  
{ones(360,480,3,'uint8')} -report
```

Free Space Detection Using Semantic Segmentation



Learn more about developing deep learning perception algorithms with these examples

R2017b



**Semantic Segmentation
Using Deep Learning**

- **Train free space detection network** using deep learning

*Computer Vision
System Toolbox™*

R2018a



**Code Generation for
Semantic Segmentation
Network**

- **Generate CUDA® code** to execute directed acyclic graph network on an NVIDIA GPU

GPU Coder™

R2018a

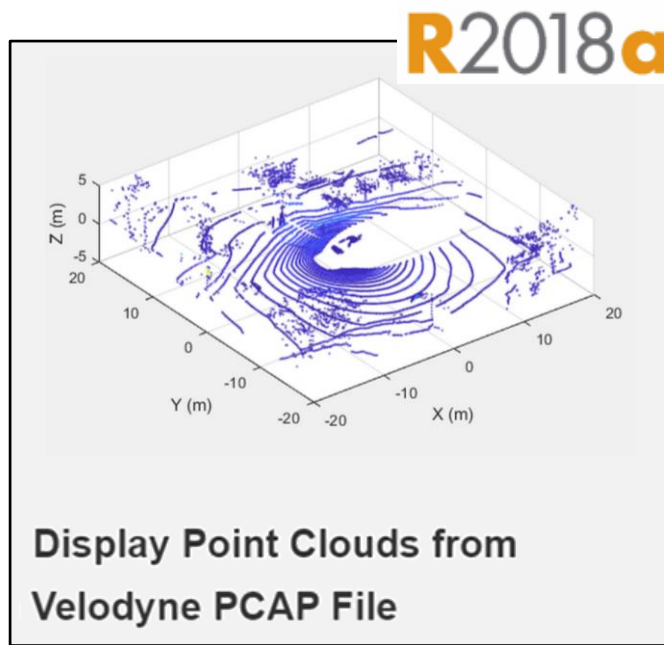


**Automate Ground Truth
Labeling for Semantic
Segmentation**

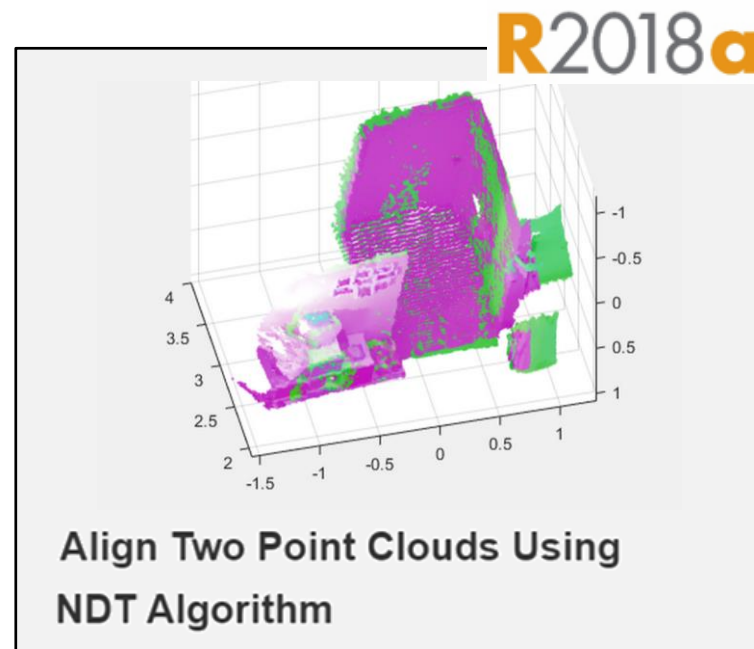
- **Add semantic segmentation automation algorithm** to Ground Truth Labeler App

*Automated Driving
System Toolbox™*

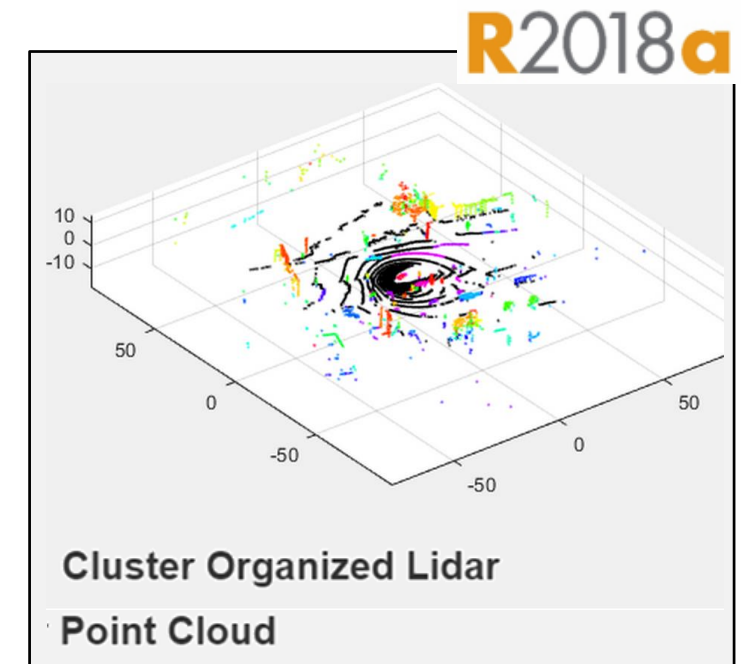
Learn about developing lidar perception algorithms with these examples



- **Read Velodyne files**
`velodyneFileReader`
Automated Driving System Toolbox™



- **Register** point clouds with Normal Distributions Transform
`pcregisterndt`
Computer Vision System Toolbox™



- **Segment** lidar point cloud
`segmentLidarData`
Automated Driving System Toolbox™

How can you use MATLAB and Simulink to develop perception algorithms?

Deep learning



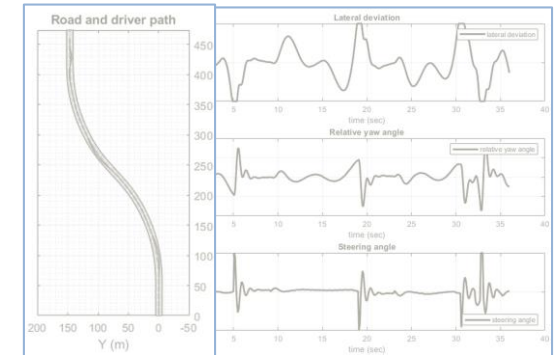
Perception

Sensor fusion
with live data



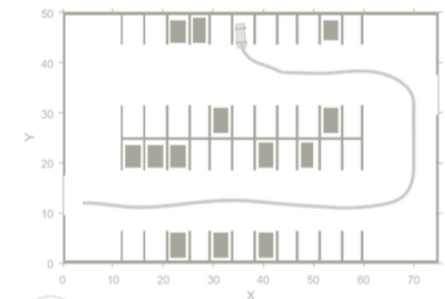
Control

Sensor models &
model predictive control

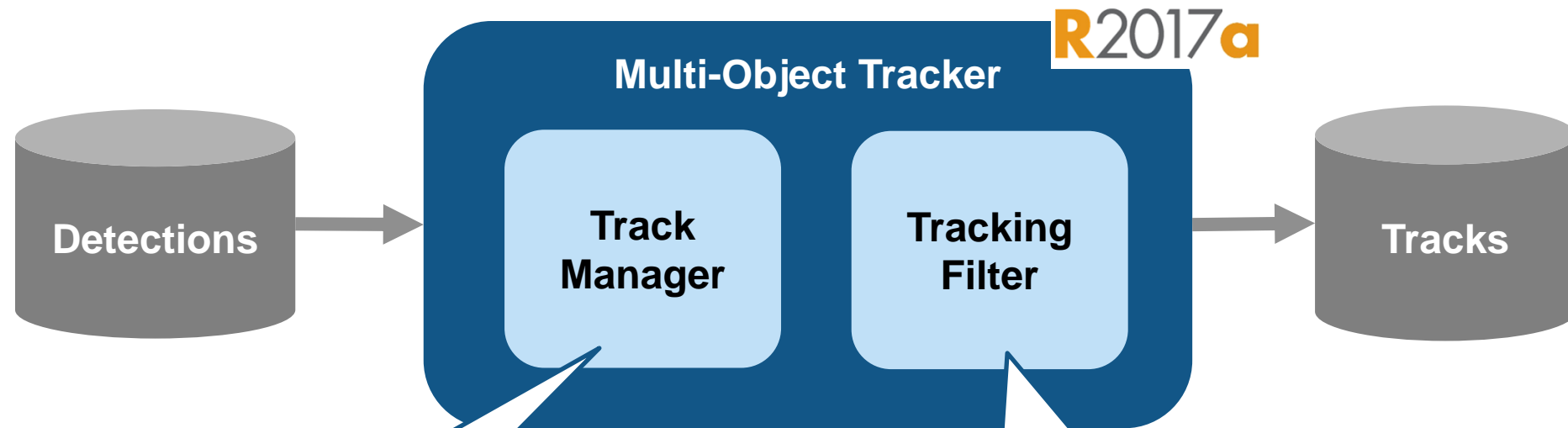


Planning

Path planning



Automated Driving System Toolbox introduced: Multi-object tracker to develop sensor fusion algorithms



- Assigns detections to tracks
- Creates new tracks
- Updates existing tracks
- Removes old tracks

- Predicts and updates state of track
- Supports linear, extended, and unscented Kalman filters

Videos and Webinars

Some common questions from automated driving engineers

How can I visualize vehicle data?

How can I fuse multiple detections?

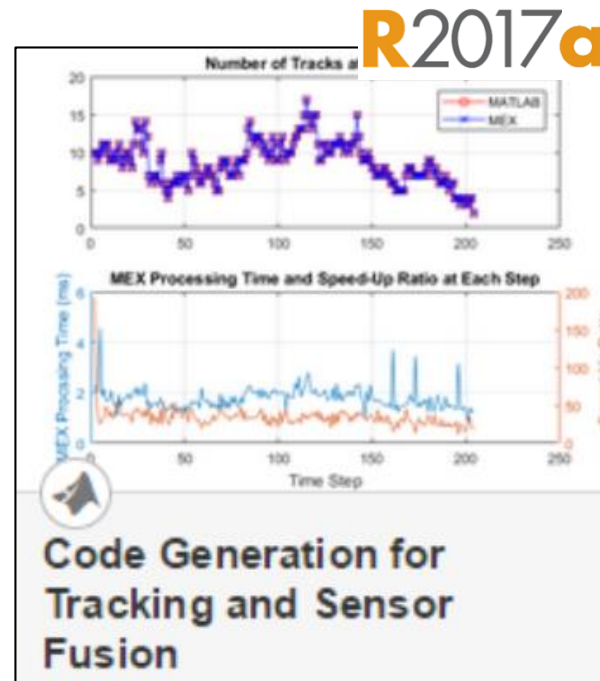
19:27

Introduction to Automated Driving System Toolbox

Automated Driving System Toolbox introduced examples to: **Develop sensor fusion algorithms with recorded data**

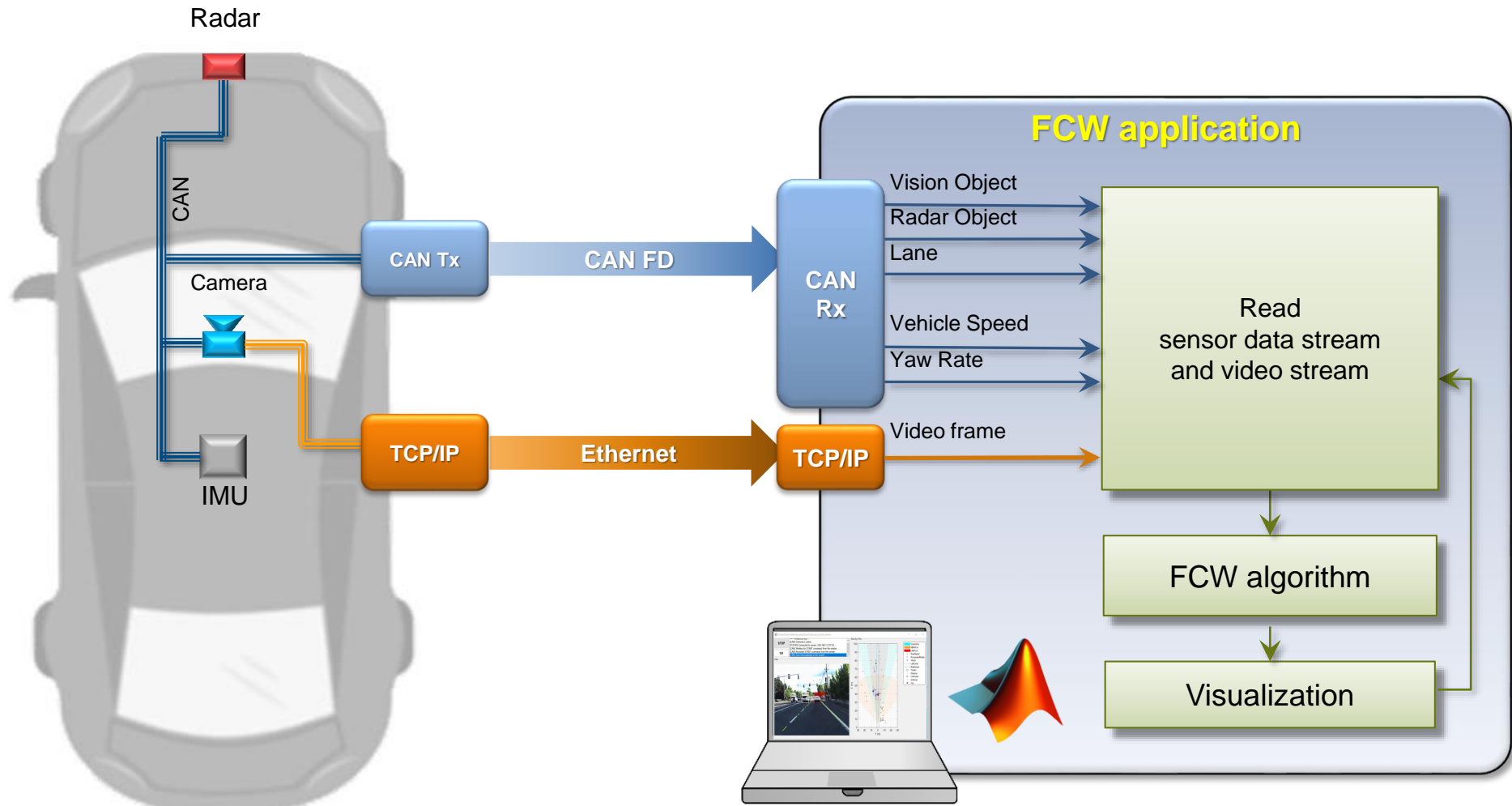


- **Design**
multi-object tracker
based on logged
vehicle data

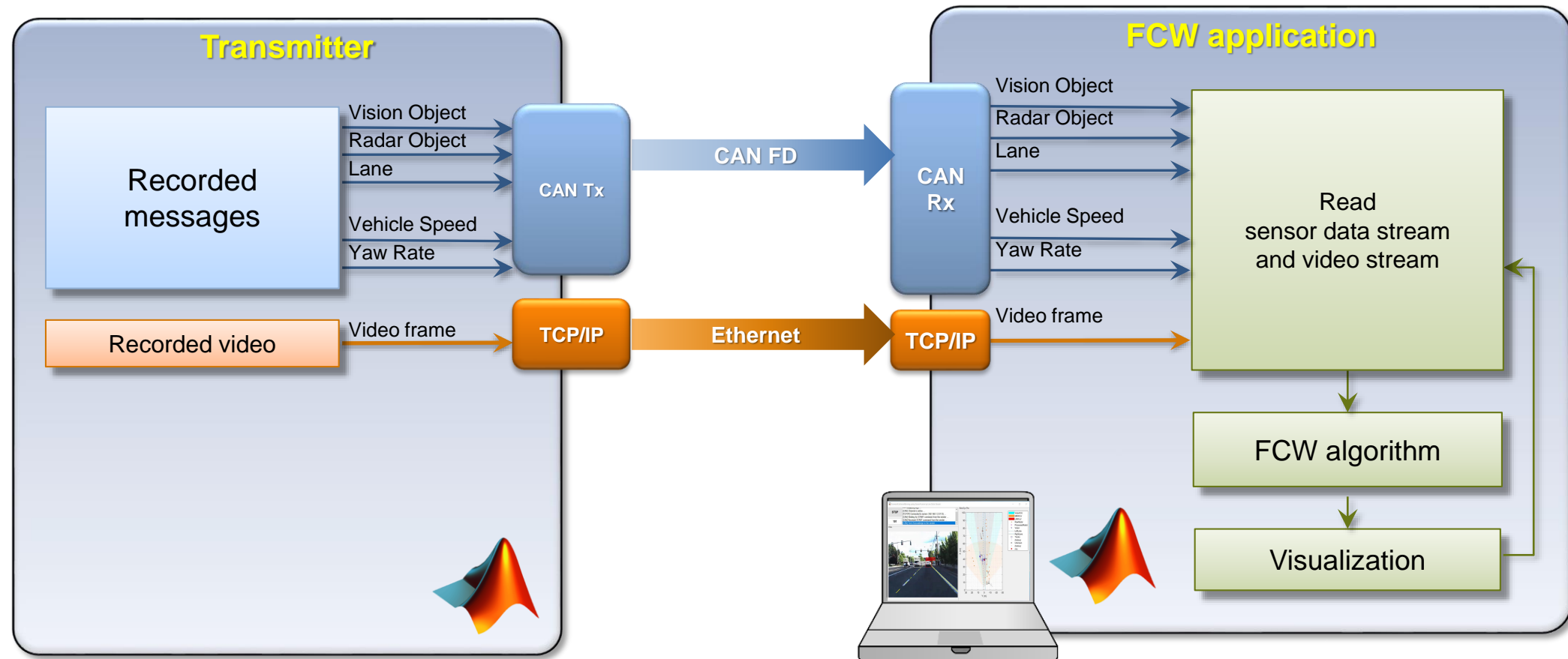


- **Generate C/C++**
code from algorithm
which includes a
multi-object tracker

Test forward collision warning algorithm with live data from vehicle

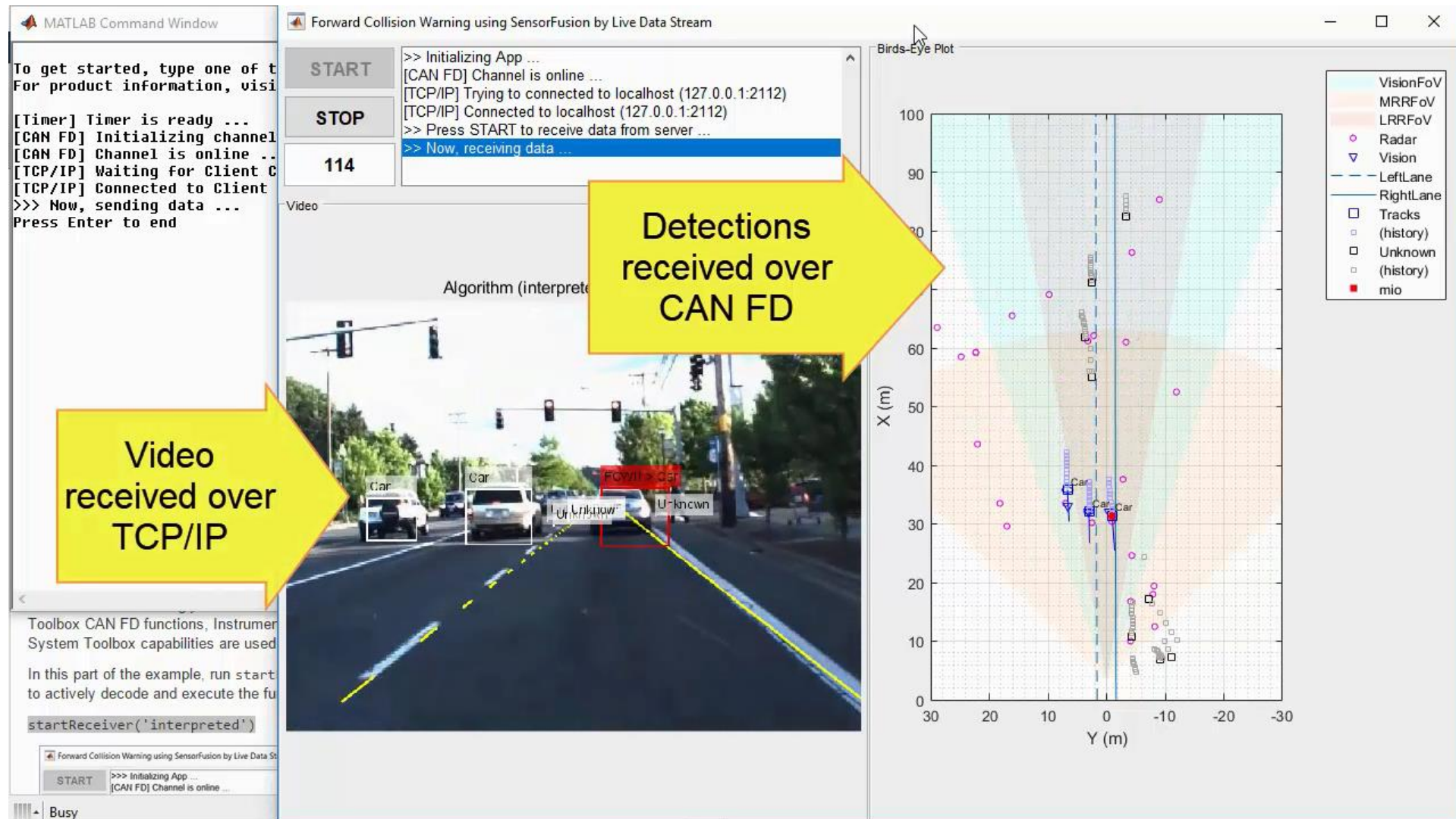


Test forward collision warning algorithm with live data from “surrogate” vehicle





Receive live CAN FD and TCP/IP data



Generate C/C++ code for algorithm

MATLAB Coder Report Viewer - C:\MATLABExamples\VNT\codegen\lib\trackingForFCW_kernel\html\report.mldatx

REPORT

Back Forward Find Trace Code Edit In MATLAB Package Code

NAVIGATE TRACE EDIT SHARE

MATLAB SOURCE

Function List Call Tree

trackingForFCW_kernel.m

- trackingForFCW_kernel
- fx calculateGroundSpeed
- fx fcwmeas > 1
- fx fcwmeas > 2
- fx fcwmeasjac > 1
- fx fcwmeasjac > 2
- fx findMostImportantObject
- fx findNonClutterRadarObjec
- fx initConstantAccelerationFi
- fx processDetections

GENERATED CODE

- trackingEKF.h
- trackingForFCW_kernel.c
- trackingForFCW_kernel.h
- trackingForFCW_kernel_e
- trackingForFCW_kernel_e
- trackingForFCW_kernel_e
- trackingForFCW_kernel_e
- trackingForFCW_kernel_e
- trackingForFCW_kernel_ir
- trackingForFCW_kernel_ir
- trackingForFCW_kernel_rt
- trackingForFCW_kernel_rt
- trackingForFCW_kernel_te

```
1565 *      const struct5_T *laneReports
1566 *      struct7_T *egoLane
1567 *      double time
1568 *      const double positionSelector[12]
1569 *      const double velocitySelector[12]
1570 *      emxArray_struct8_T *confirmedTracks
1571 *      double *numTracks
1572 *      struct10_T *mostImportantObject
1573 * Return Type : void
1574 */
1575 void trackingForFCW_kernel(const struct0_T *visionObjects, const struct2_T
1576 *radarObjects, const struct4_T *inertialMeasurementUnit, const struct5_T
1577 *laneReports, struct7_T *egoLane, double time, const double positionSelector
1578 [12], const double velocitySelector[12], emxArray_struct8_T *confirmedTracks,
1579 double *numTracks, struct10_T *mostImportantObject)
1580 {
1581     emxArray_objectDetection *detections;
1582     emxInit_objectDetection(&detections, 2);
1583
1584
```

Generated C function

SUMMARY ALL MESSAGES (1) BUILD LOGS CODE INSIGHTS (0) VARIABLES

Code generation successful

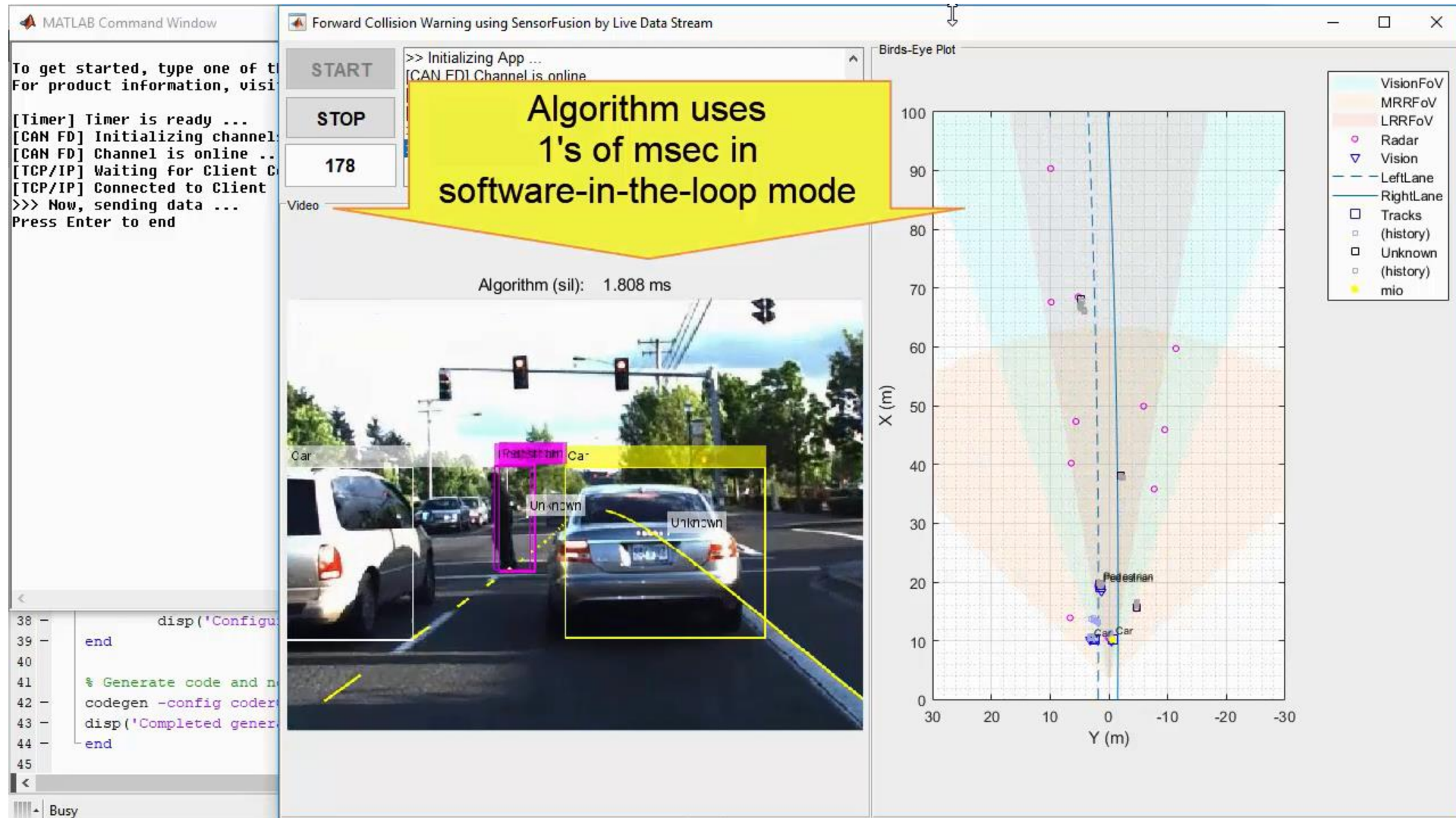
Generated on: 17-Mar-2018 19:07:16

Build type: Static Library

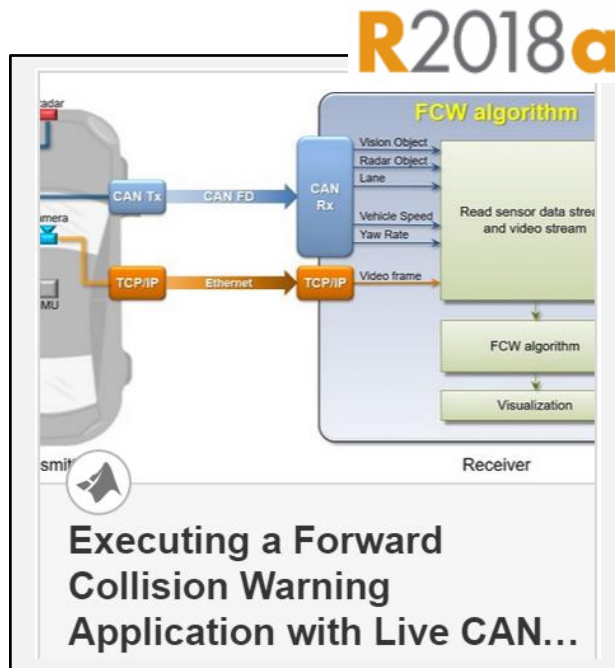
Output file: C:\MATLABExamples\VNT\codegen\lib\trackingForFCW_kernel\trackingForFCW_kernel.lib

Processor: Generic->MATLAB Host Computer

Stream live CAN FD and TCP/IP data into compiled algorithm code

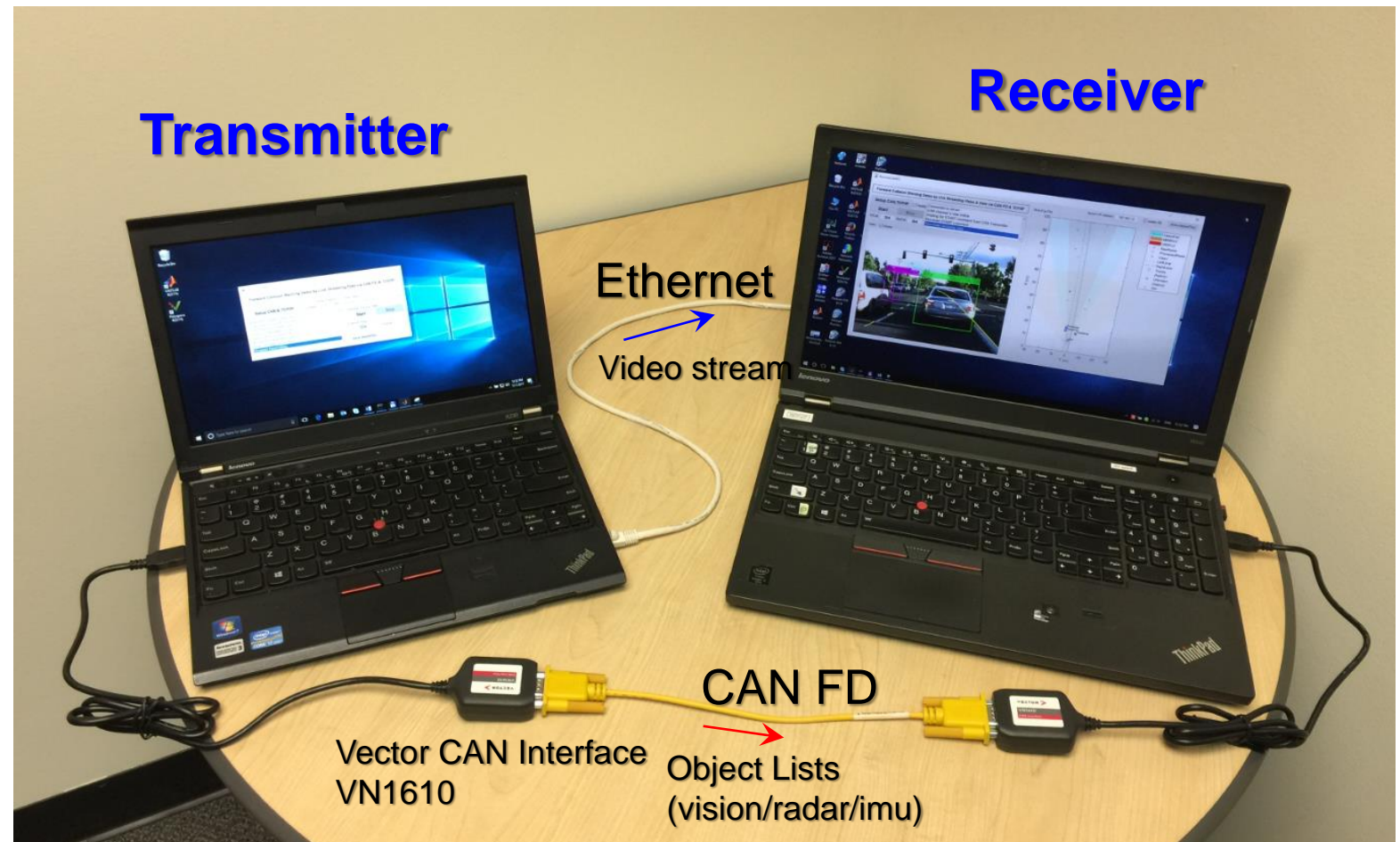


Learn about developing sensor fusion algorithms with live data using this example



- Stream CAN FD data to prototype algorithms on your laptop

Vehicle Network Toolbox™



How can you use MATLAB and Simulink to develop control algorithms?

Deep learning



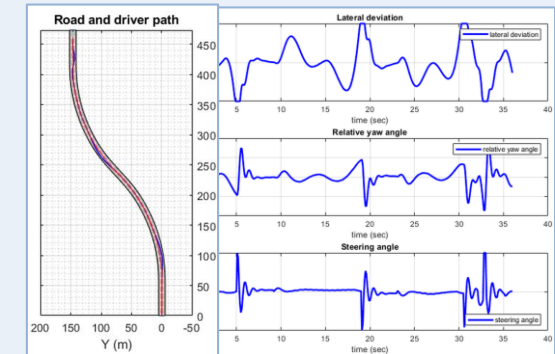
Perception

Sensor fusion
with live data



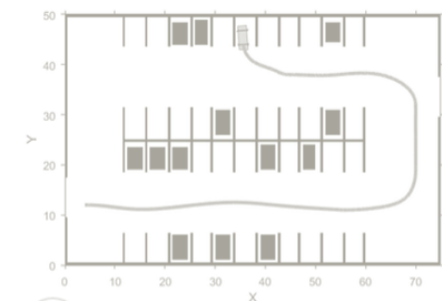
Control

Sensor models &
model predictive control

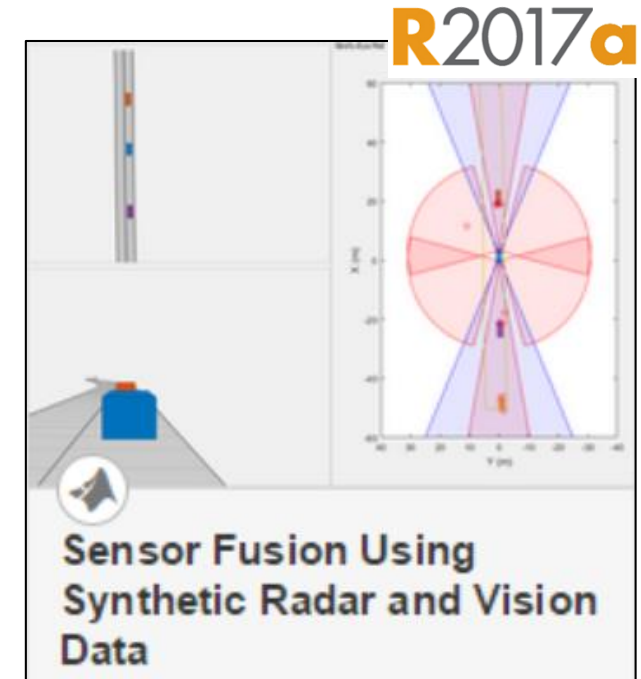
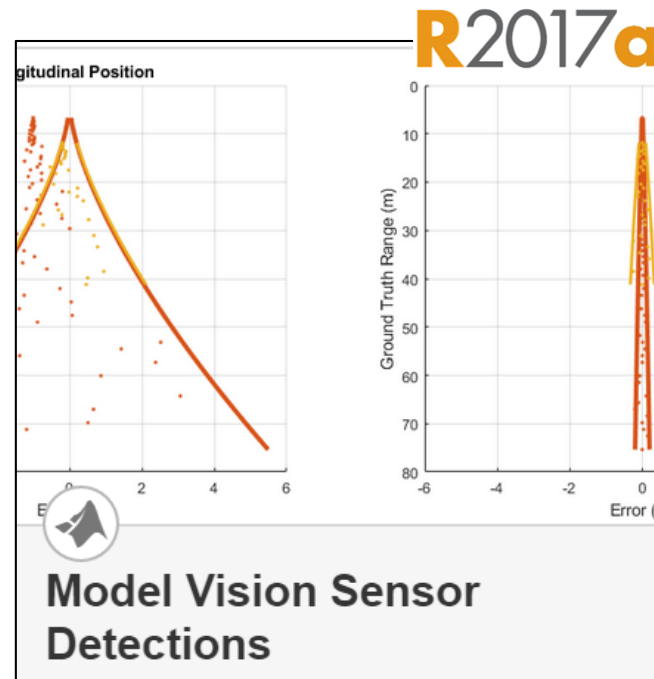
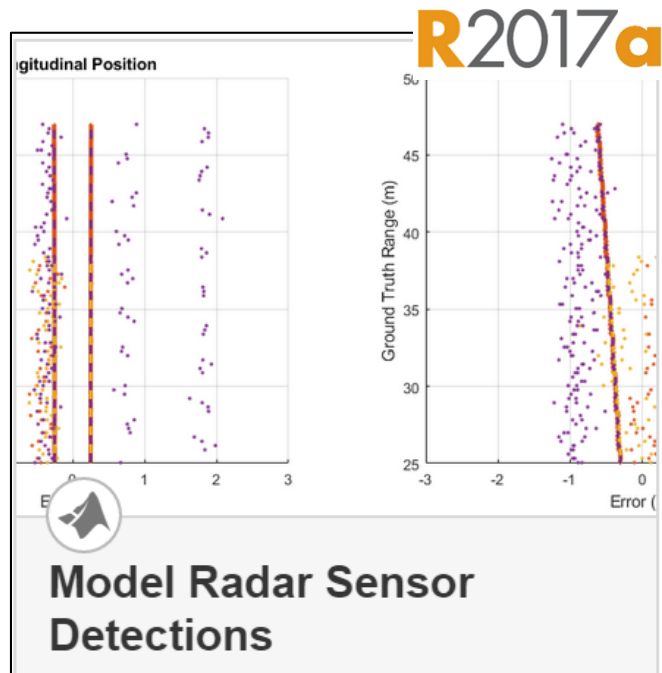


Planning

Path planning



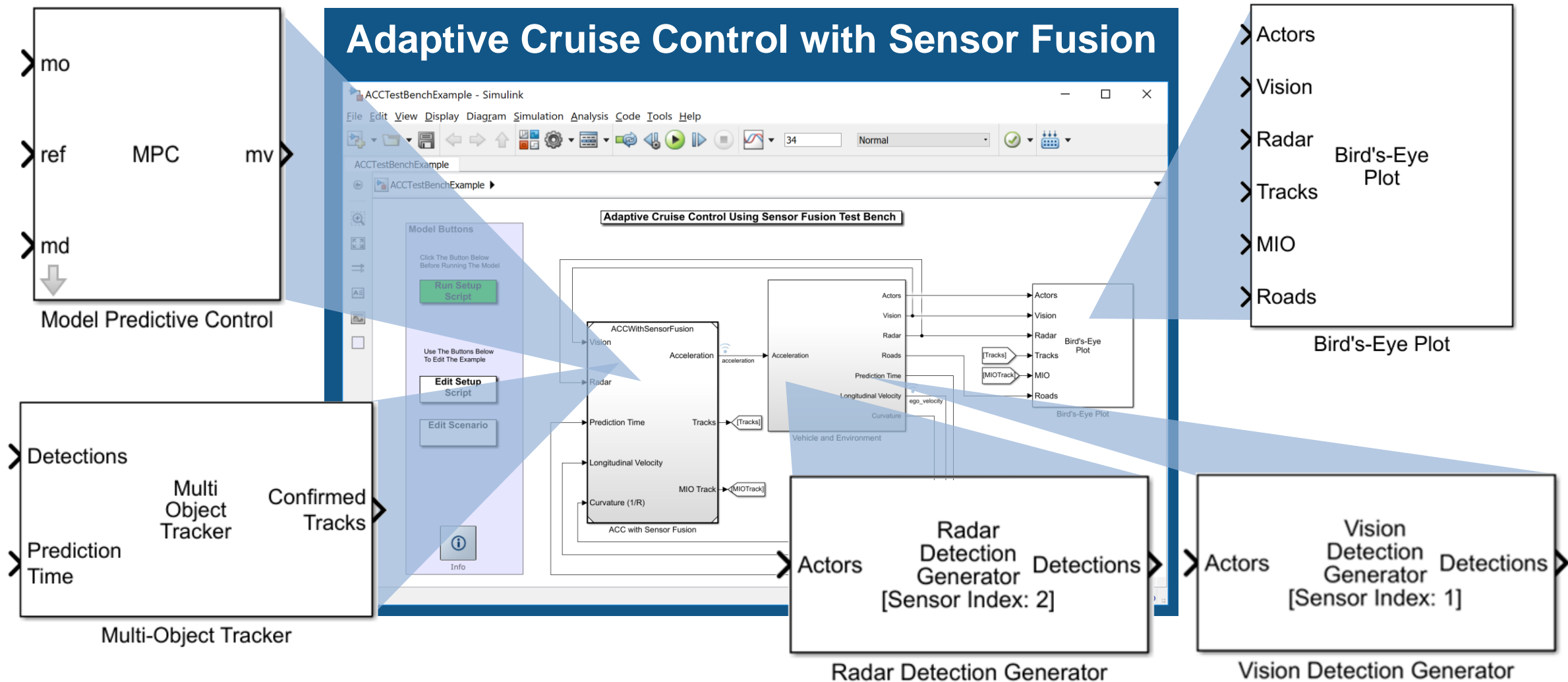
Automated Driving System Toolbox introduced examples to: **Synthesize detections to test sensor fusion algorithms**



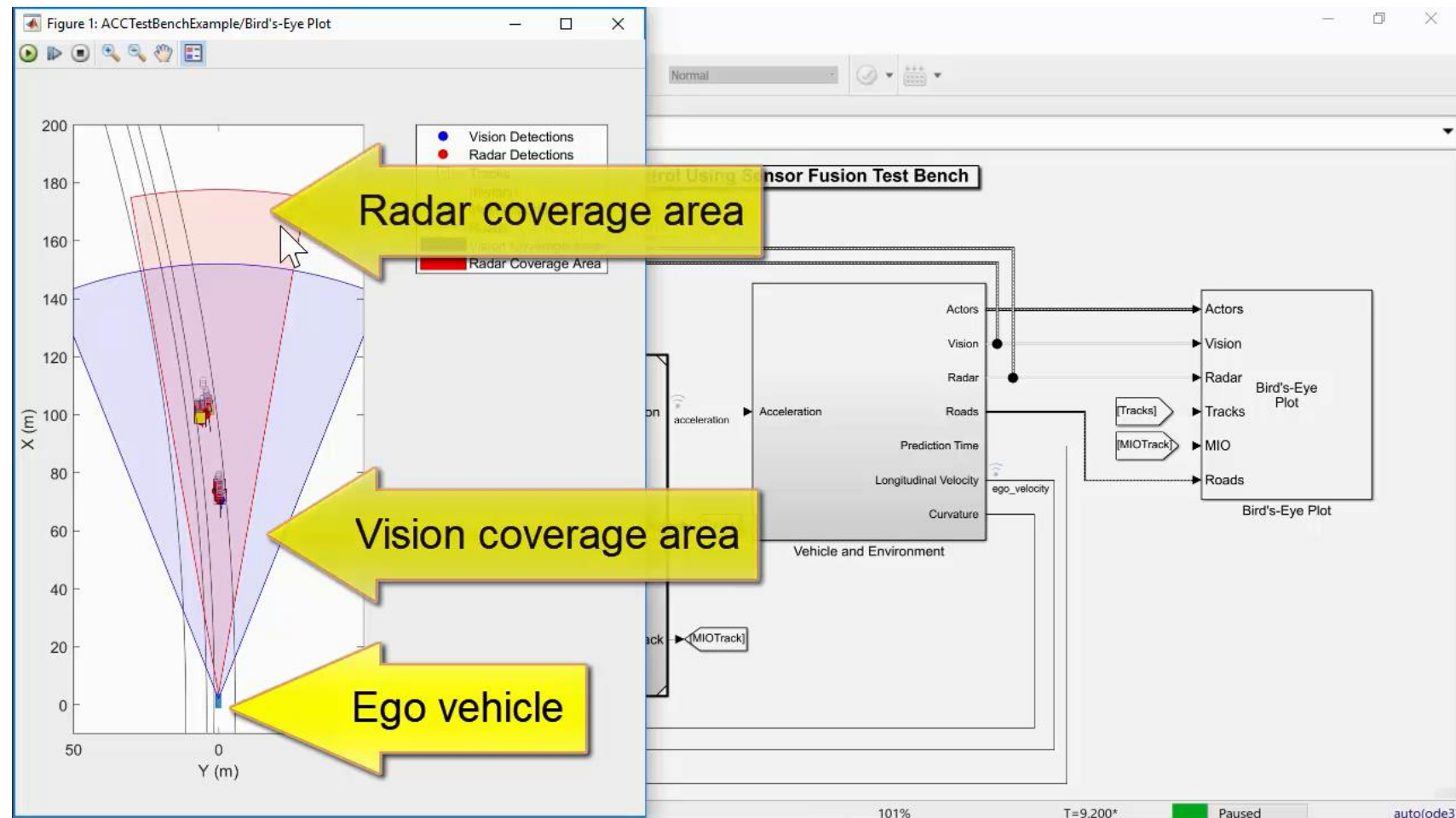
- **Synthesize radar** detections with probabilistic impairments
- **Synthesize vision** detections with probabilistic impairments
- **Synthesize scenario** to test multi-object tracker

Simulate closed loop system with radar/vision detections, sensor fusion, and model-predictive control

R2017b



Synthesize detections to test sensor fusion and model-predictive controller



How can MPC be applied to lane keeping control?

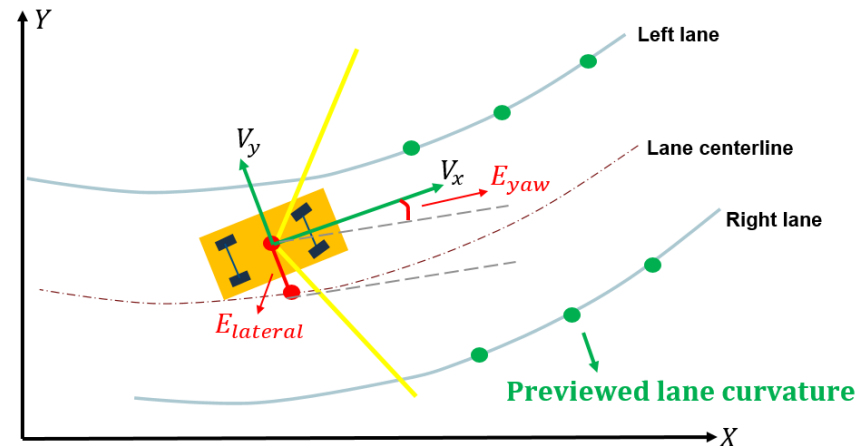
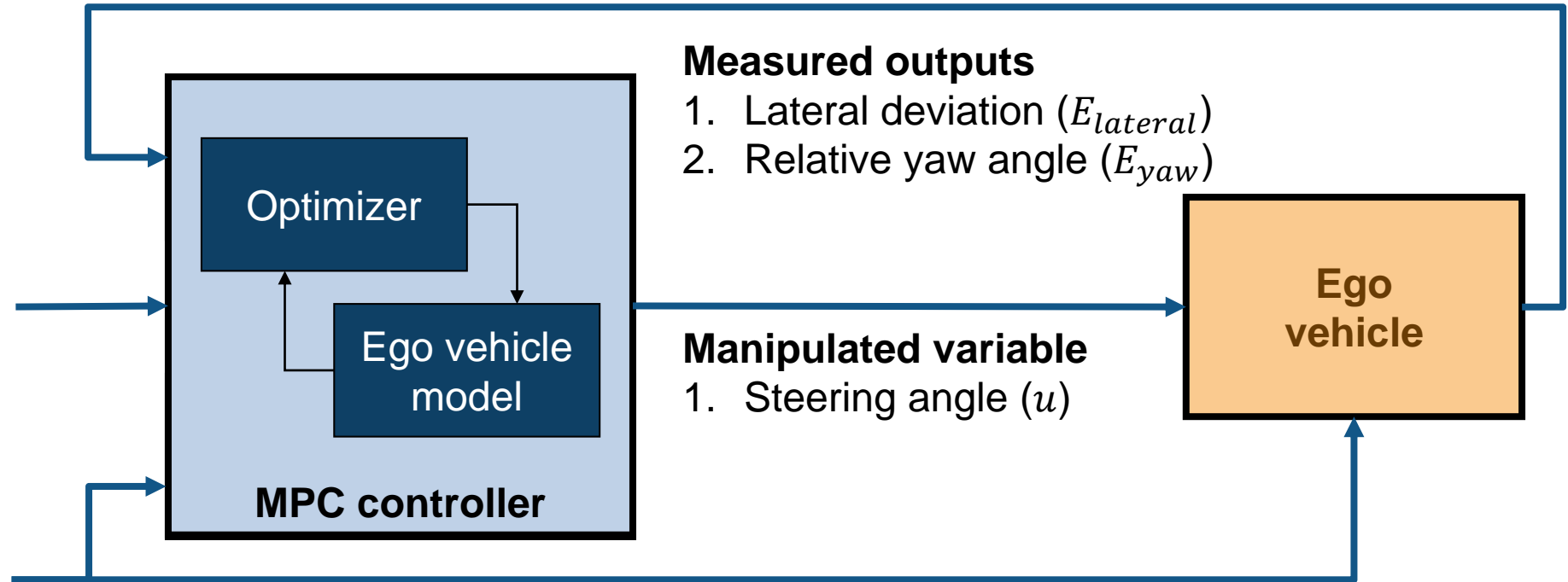
References

1. For $E_{lateral}(0)$
2. For $E_{yaw}(0)$

Measured disturbance

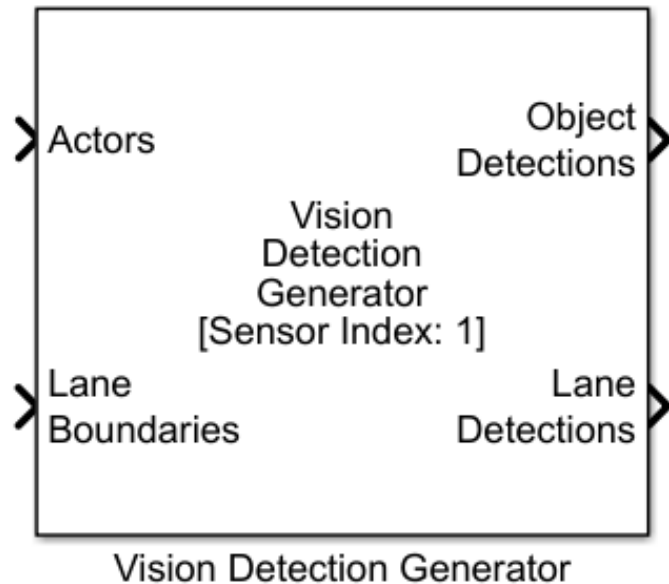
1. Previewed curvature

$$\begin{aligned} \text{minimize: } & |E_{lateral}|^2 + |E_{yaw}|^2 \\ \text{subject to: } & u_{min} \leq u \leq u_{max} \end{aligned}$$



Vision Detection Generator models lane detection sensor

R2018a



Block Parameters: Vision Detection Generator

Vision Detection Generator

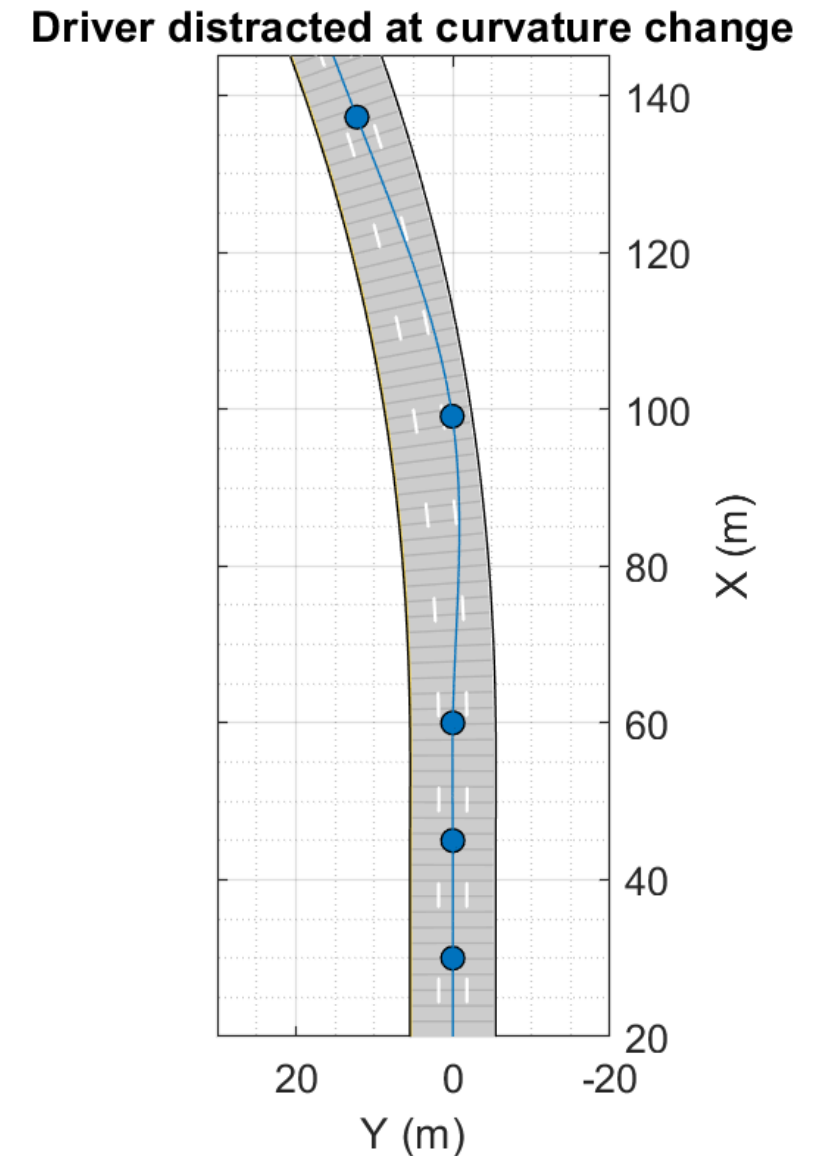
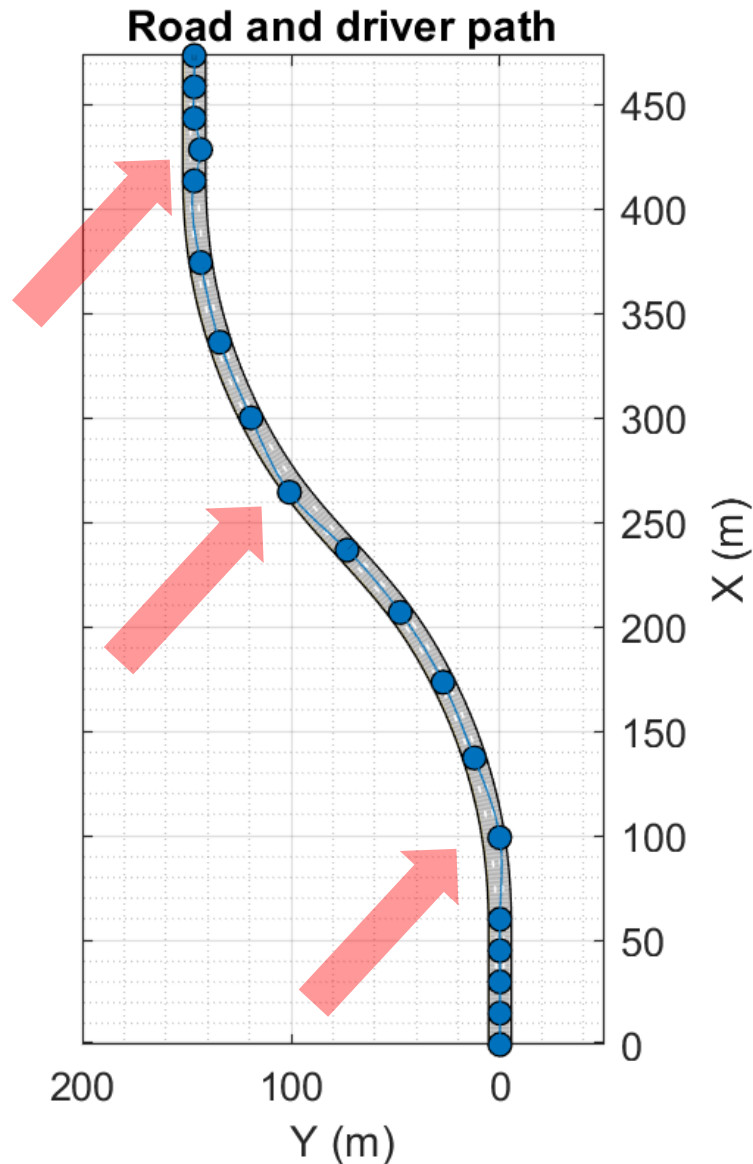
Sensor simulation block used to generate vision detections from simulated actor poses. Detections are generated at intervals of the sensor's update interval. A statistical model generates measurement noise, true detections, and false positives. The random numbers used by the statistical model are controlled by the random number generator settings on the Measurements tab.

[Source code](#)

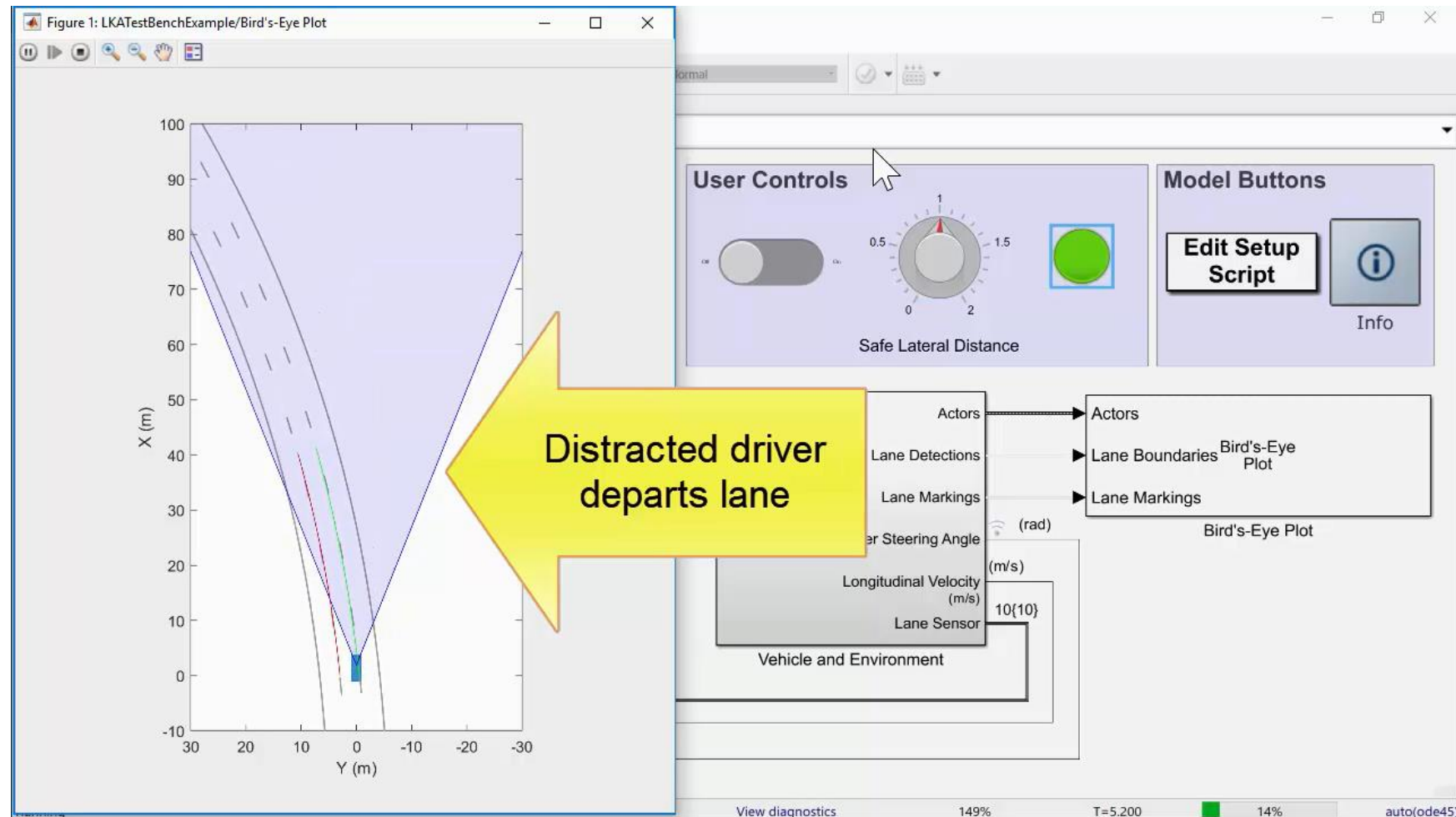
Parameters	Measurements	Actor Profiles	Camera Intrinsics
Sensor Identification			
Unique identifier of sensor:	1		
Types of detections generated by sensor:	Lanes and objects		
Required interval between sensor updates (s):			
Required interval between lane detection updates (s):	Lanes and objects		
Sensor Extrinsic			
Sensor's (x,y) position (m):	[1.9, 0]		
Sensor's height (m):	1.1		
Yaw angle of sensor mounted on ego vehicle (deg):	0		
Pitch angle of sensor mounted on ego vehicle (deg):	1		
Roll angle of sensor mounted on ego vehicle (deg):	0		

Create highway double curve with drivingScenario

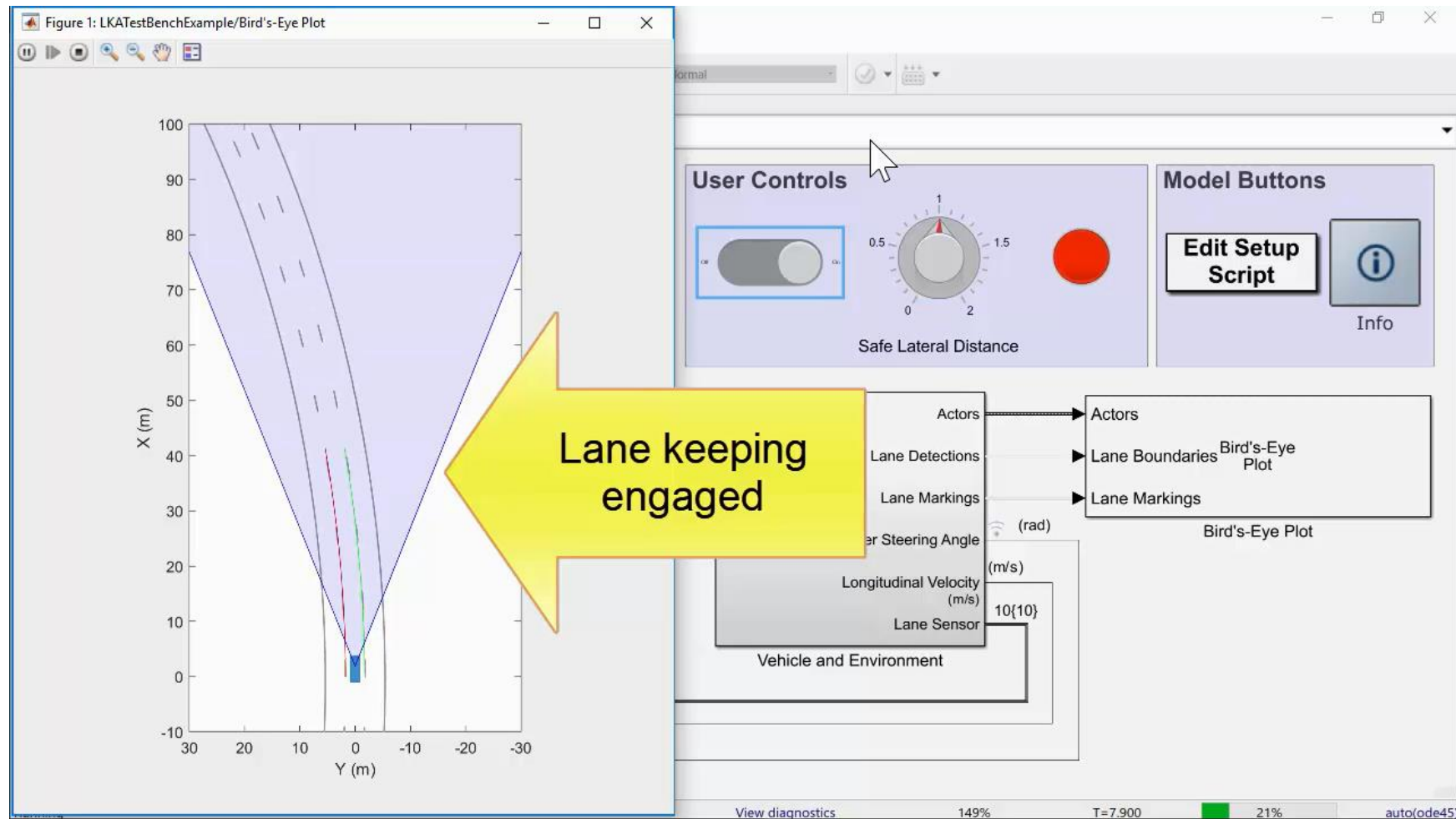
- Driver waypoints simulate distraction at curvature changes



Simulate distracted driver

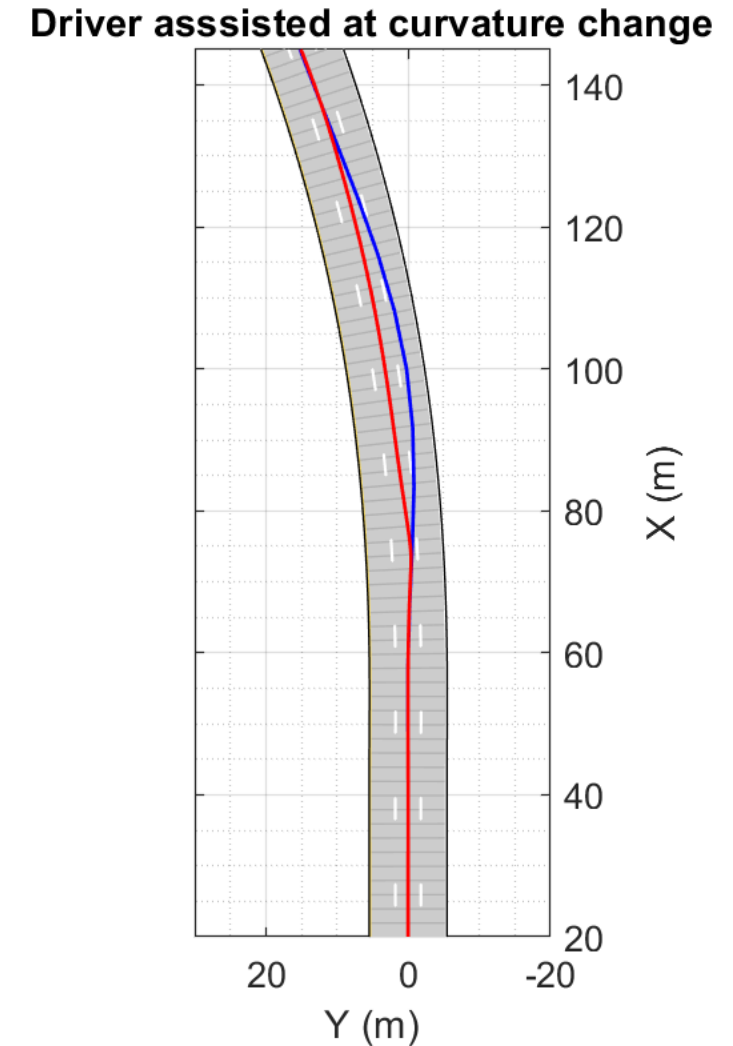
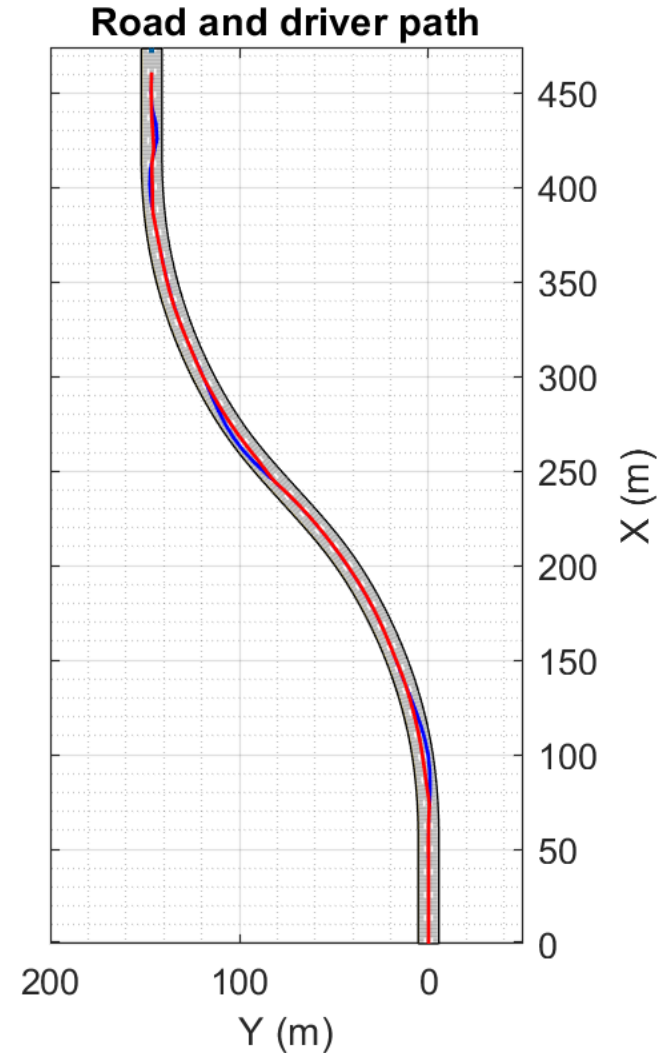


Simulate lane keep assist at distraction events

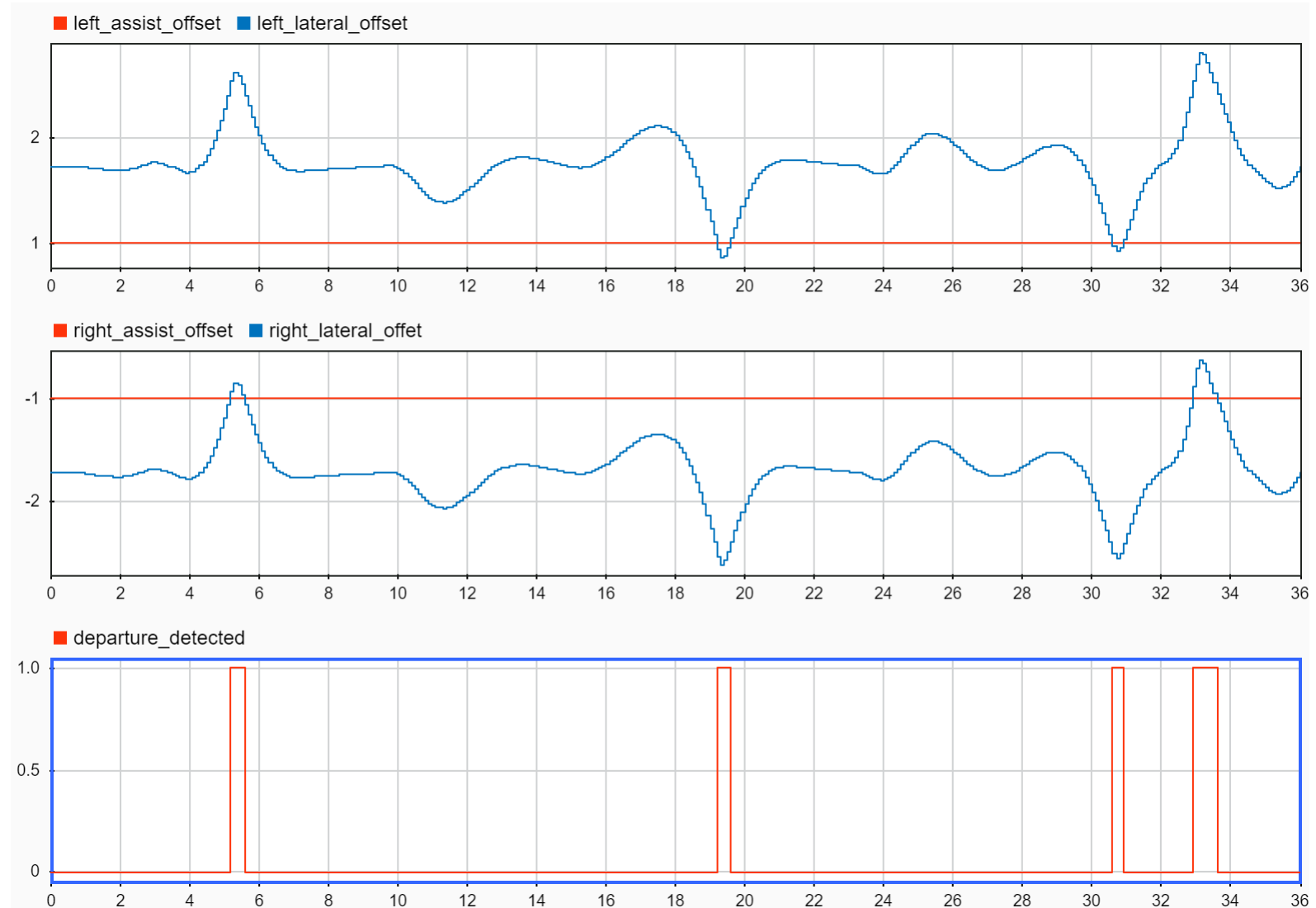


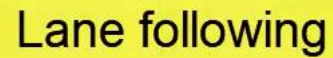
Compare distracted and assisted results

- Detect lane departure and maintain lane during distraction



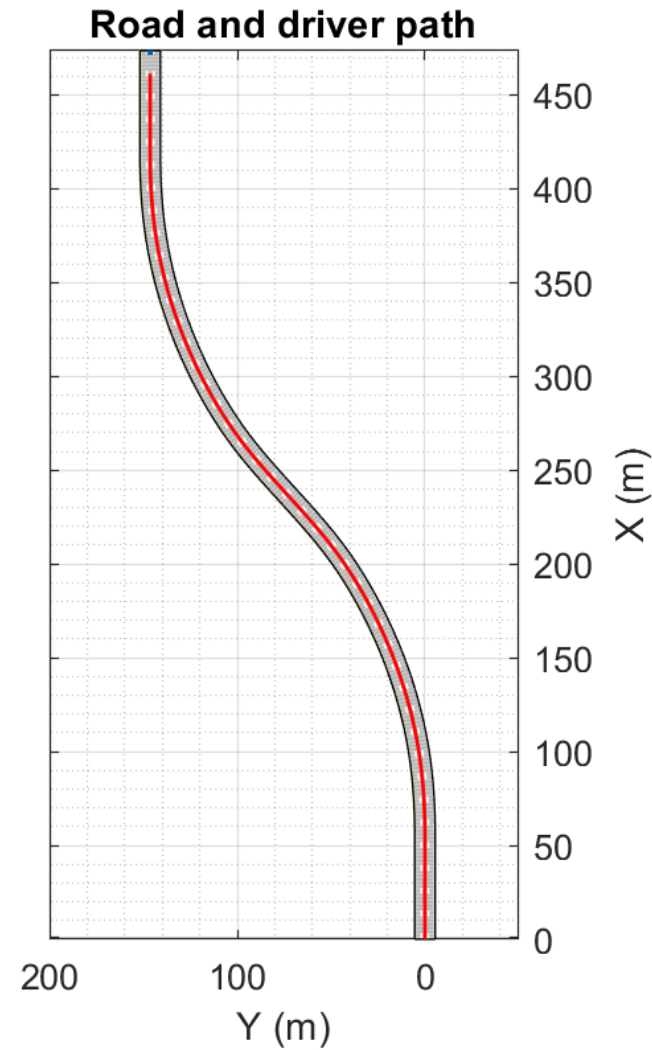
Detect departure based on lateral offset to lane boundary



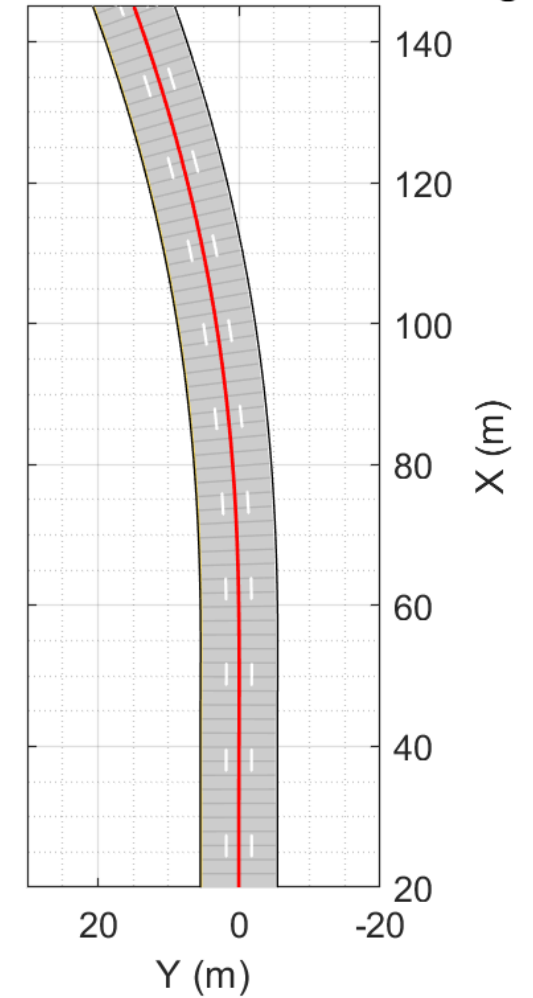


Explore lane following results

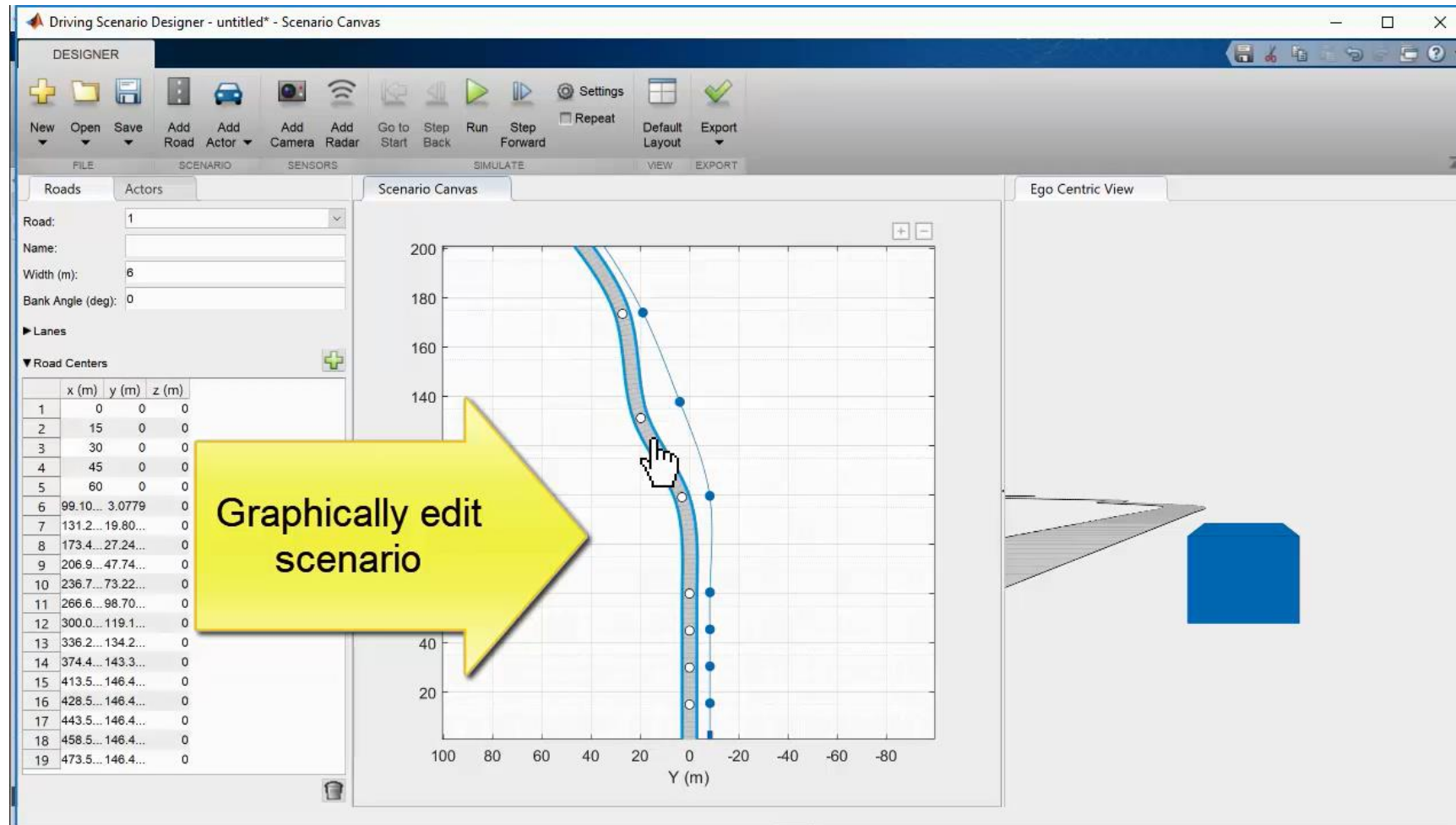
- Vehicle stays within lane boundaries



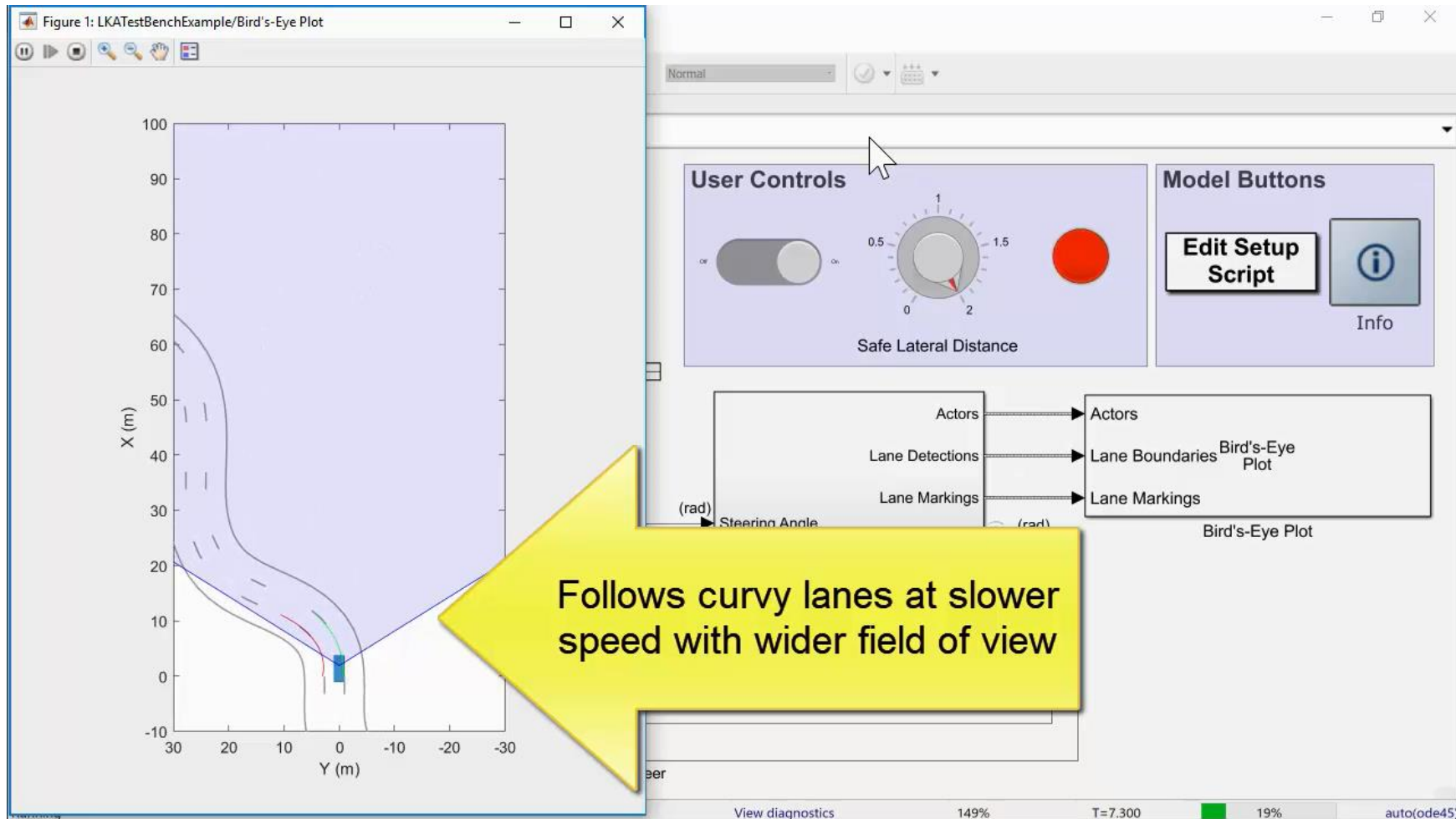
Driver assisted at curvature change



Graphically edit scenarios with Driving Scenario Designer



Explore what is required to follow high curvature paths





- **Edit roads, cuboid actors, and sensors** with
Driving Scenario Designer App
`drivingScenarioDesigner`

How can you use MATLAB and Simulink to develop planning algorithms?

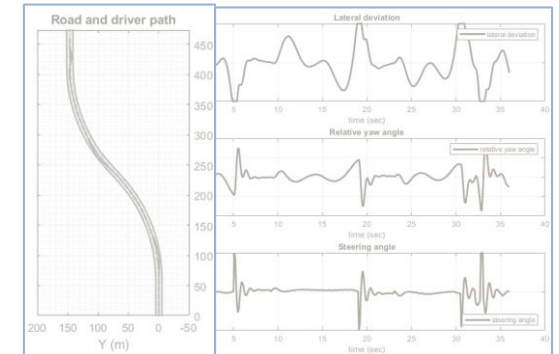
Deep learning



Perception

Control

Sensor models & model predictive control

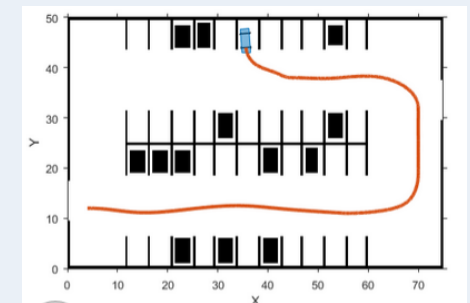


Sensor fusion with live data

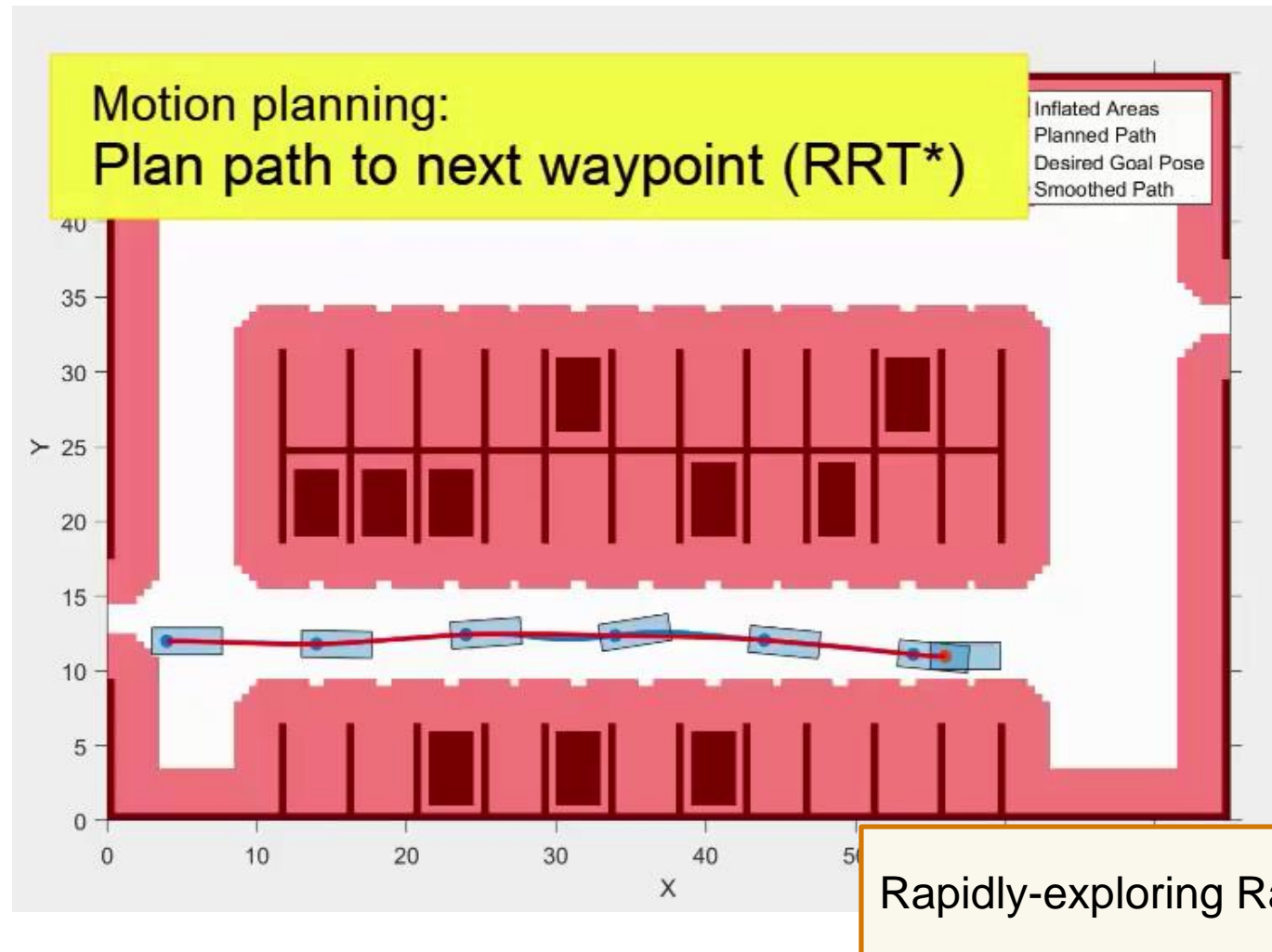


Planning

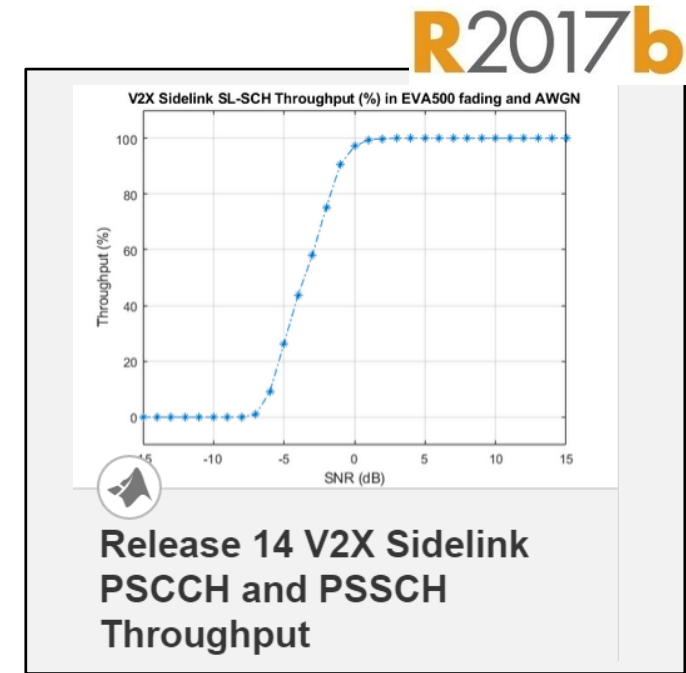
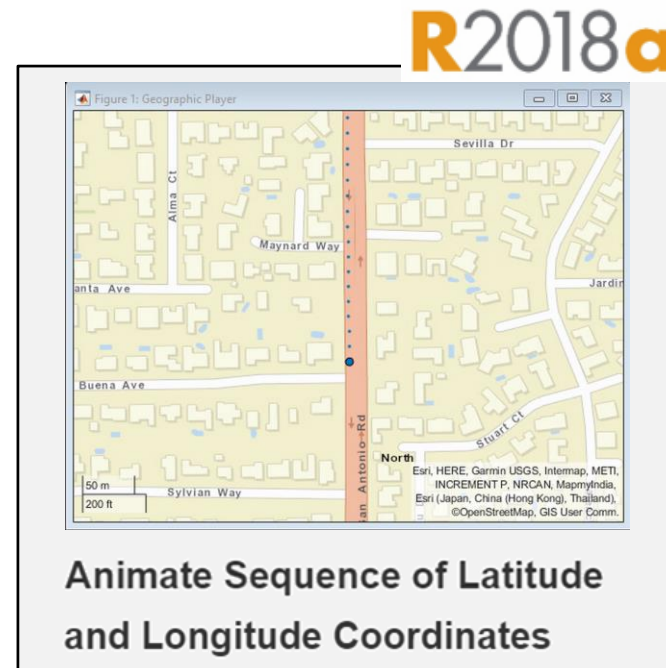
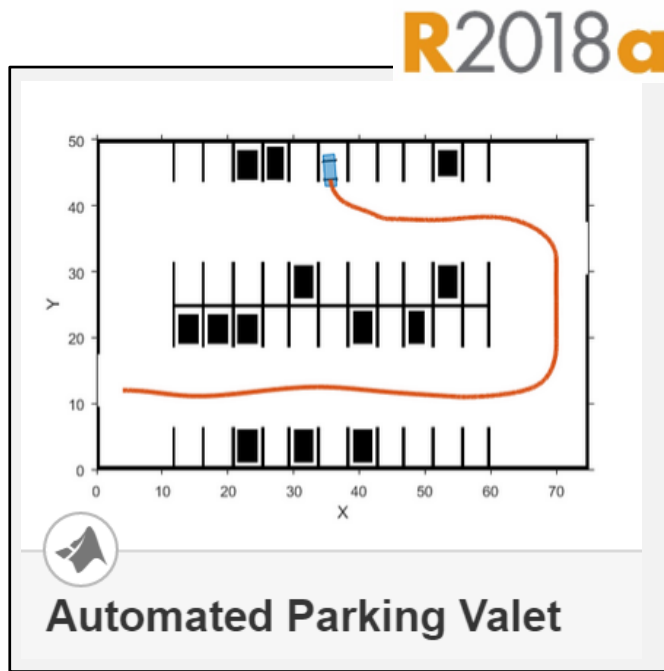
Path planning



We are investing in design and simulation of path planning for automobiles



Learn about developing path planning algorithms with these examples



- **Plan path** for automobile given pre-defined map
Automated Driving
System Toolbox™
- **Plot map tiles** using World Street Map (Esri)
Automated Driving
System Toolbox™
- **Simulate V2X communication** to assess channel throughput
LTE System Toolbox™

Examples of how you can use MATLAB and Simulink to develop automated driving algorithms

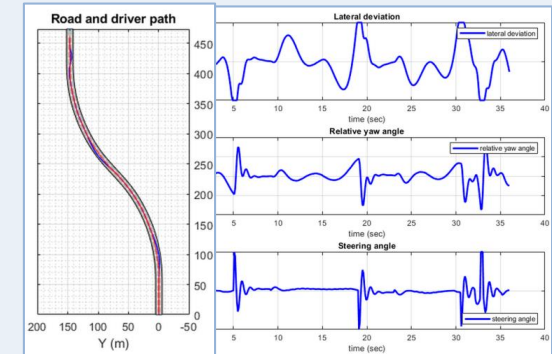
Deep learning



Perception

Control

Sensor models & model predictive control

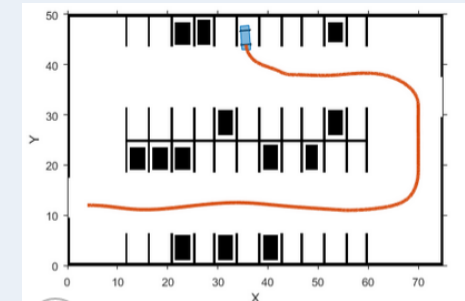


Sensor fusion with live data



Planning

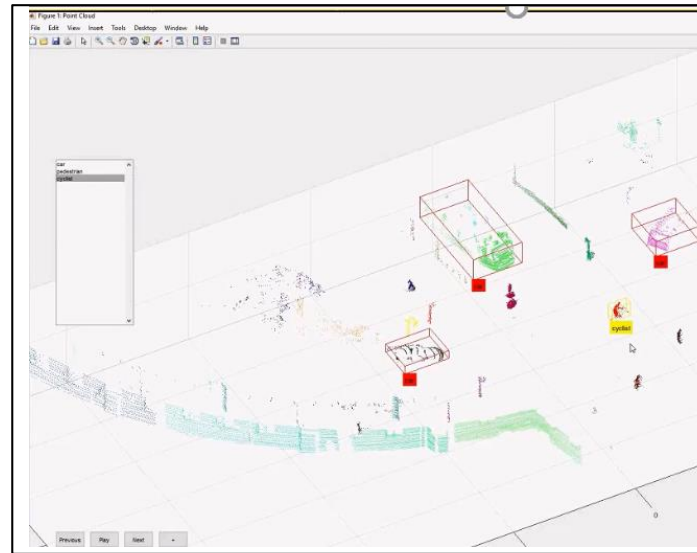
Path planning



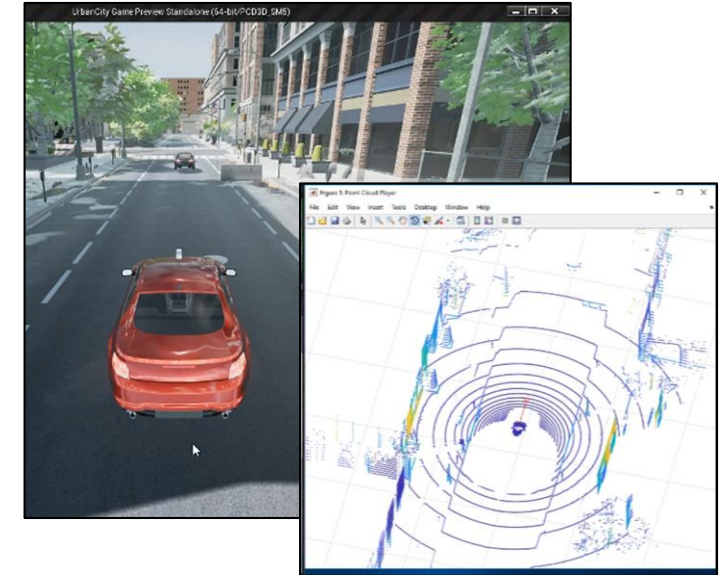
MathWorks can help you customize MATLAB and Simulink for your automated driving application



- **Web based ground truth labeling**
 - Consulting project with Caterpillar
 - [2017 MathWorks Automotive Conference](#)



- **Lidar ground truth labeling**
 - Joint presentation with Autoliv
 - 2018 MathWorks Automotive Conference (May 2nd, Plymouth MI)



- **Lidar sensor model for Unreal Engine**
 - Joint paper with Ford
 - SAE Paper 2017-01-0107

How can we help you can use MATLAB and Simulink to develop automated driving algorithms?

Perception

Control

Planning

