

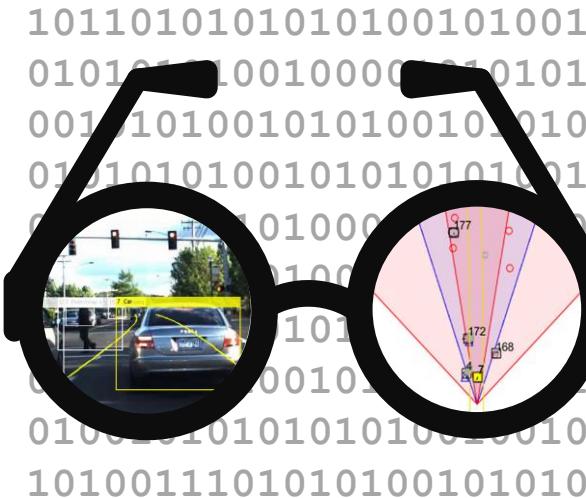
Introduction to Automated Driving System Toolbox: Design and Verify Perception Systems

MATLAB Expo Germany 2017

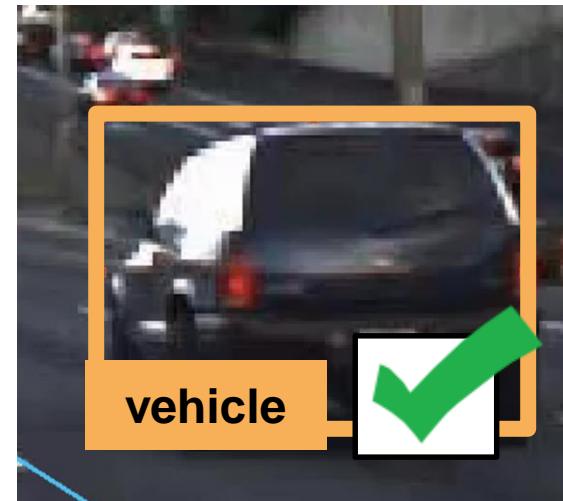
Anders Sollander
Principal Consultant



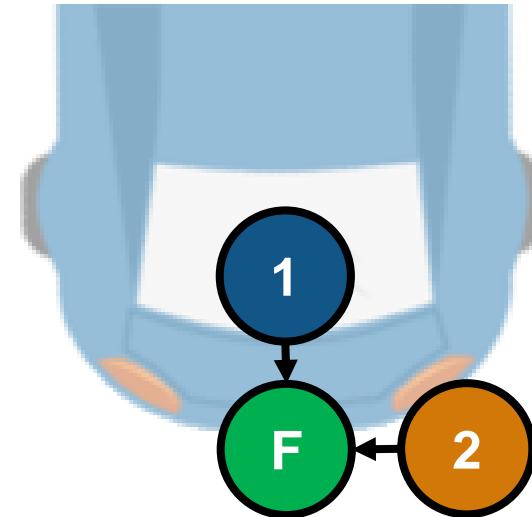
Some common questions from automated driving engineers



**How can I
visualize vehicle
data?**

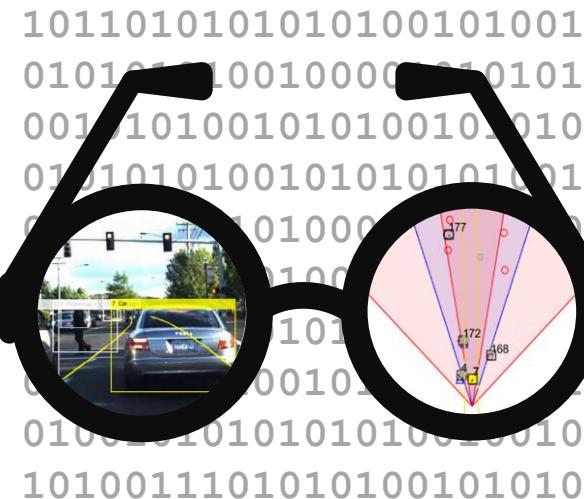


**How can I
detect objects in
images?**

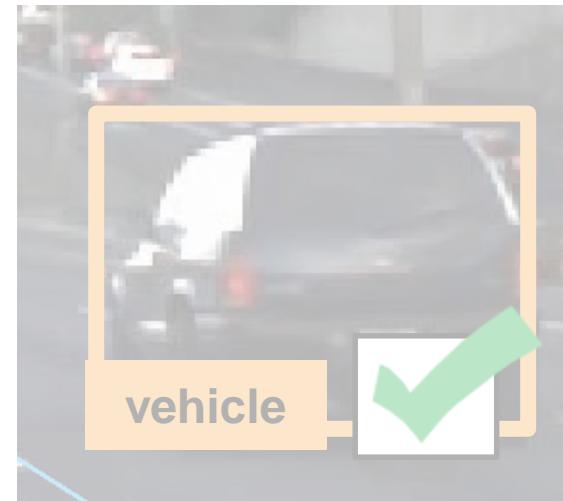


**How can I
fuse multiple
detections?**

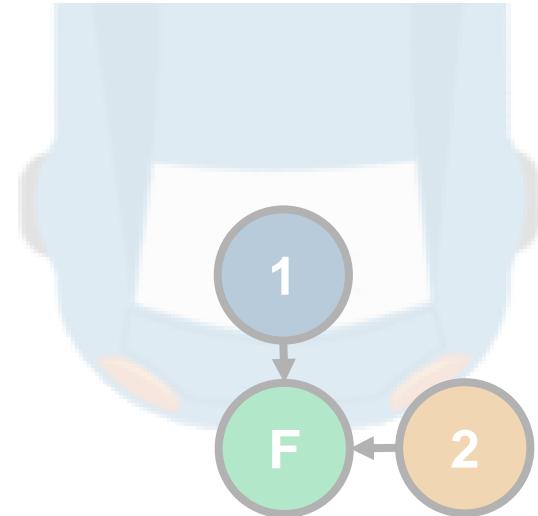
Some common questions from automated driving engineers



**How can I
visualize vehicle
data?**



**How can I
detect objects in
images?**



**How can I
fuse multiple
detections?**

Examples of automated driving sensors

Camera

Radar-based
object detector

Vision-based
object detector

Lidar

Lane detector

Inertial
measurement
unit



Examples of automated driving sensor data

Camera

(640 x 480 x 3)

239	239	237	238	241	241	241	242	243	243	243	243	243	243	243	243	243	243	243	243	
252	252	251	252	252	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
254	254	253	253	254	253	255	253	255	255	255	255	255	255	255	255	255	255	255	255	255
250	251	251	251	251	251	253	251	251	251	251	251	251	251	251	251	251	251	251	251	251
251	252	251	253	253	253	251	251	251	251	251	251	251	251	251	251	251	251	251	251	251
252	253	253	254	254	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
252	253	253	254	254	254	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
252	253	253	254	254	254	254	253	253	253	253	253	253	253	253	253	253	253	253	253	253
253	253	253	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254
253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253
253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253	253

Lane detector



Radar-based object detector

Lidar

Inertial measurement unit

Examples of automated driving sensor data

Camera (640 x 480 x 3)

```
239 239 237 238 241 241 241 242 243 243
252 252 251 252 252 253 253 253 253 253
```

Vision Detector

```
SensorID      = 1;
Timestamp     = 1461634696379742;
NumDetections = 6;
```

Lane Detector

Left

```
IsValid:       1
Confidence:    3
BoundaryType: 3
Offset:        1.68
HeadingAngle:  0.002
Curvature:    0.0000228
```

Right

```
IsValid:       1
Confidence:    3
```



Radar-based object detector

```
255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255
```

Lidar

```
255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255
```

Inertial measurement unit

```
255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255
```

Examples of automated driving sensor data

Camera

(640 x 480 x 3)

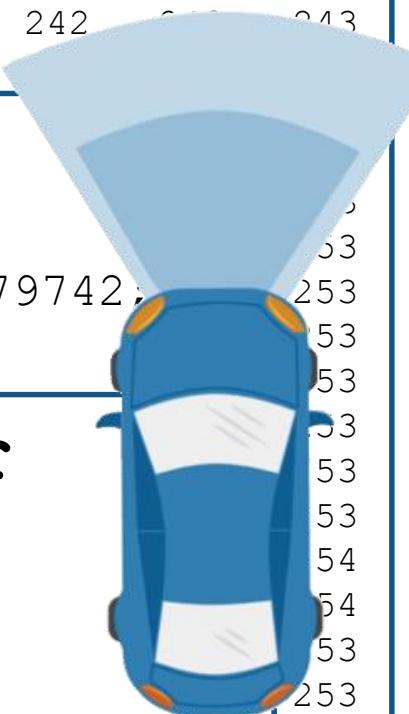
```
239 239 237 238 241 241 241 242 243
252 252 251 252 252 253 253 253 253
```

Vision Detector

```
SensorID      = 1;
Timestamp     = 1461634696379742;
NumDetections = 6;
```

Lane Detector

```
Left
  IsValid:      1
  Confidence:   3
  BoundaryType: 3
  Offset:        1.68
  HeadingAngle: 0.002
  Curvature:    0.0000228
Right
  IsValid:      1
  Confidence:   3
```



Radar Detector

```
SensorID      = 2;
Timestamp     = 1461634696407521;
NumDetections = 23;
```

Detections (1)

TrackID: 0

TrackStatus:

Position:

Velocity:

Amplitude:

Detections (2)

TrackID: 1

TrackStatus: 6

Lidar

Position: [19.59 0.34]

Velocity: [4.92 0]

Inertial measurement unit

```
TrackID: 12
TrackStatus: 5
```

Examples of automated driving sensor data

Camera

(640 x 480 x 3)

```
239 239 237 238 241 241 241 242 243
252 252 251 252 252 253 253 253
```

Vision Detector

```
SensorID = 1;
```

```
Timestamp = 1461634696379742;
```

```
NumDetections = 6;
```

```
Dete
```

Lane Detector

```
Tr
```

```
C1
```

```
Po
```

```
Ve
```

```
Si
```

```
Detec
```

```
Tr
```

```
C1
```

```
Po
```

```
Ve
```

```
Si
```

```
Left
```

```
IsValid: 1
```

```
Confidence: 3
```

```
BoundaryType: 3
```

```
Offset: 1.68
```

```
HeadingAngle: 0.002
```

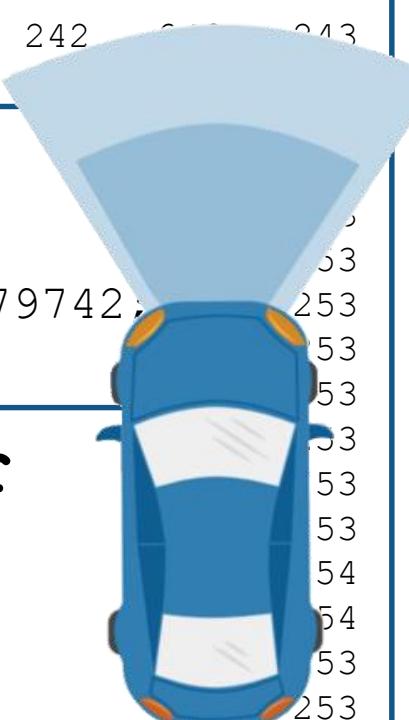
```
Curvature: 0.0000228
```

```
Right
```

```
IsValid: 1
```

```
Confidence: 3
```

```
BoundaryType: 3
```



Radar Detector

```
SensorID = 2;
```

```
Timestamp = 1461634696407521;
```

```
NumDetections = 23;
```

```
Detection
```

```
TrackID
```

```
TrackSt
```

```
Positio
```

```
Velocit
```

```
Amplitu
```

```
Detection
```

```
TrackID
```

```
TrackSt
```

```
Positio
```

```
Velocit
```

```
Amplitu
```

```
Detection
```

```
TrackID
```

```
TrackSt
```

Lidar

(47197 x 3)

	-12.2911	1.4790	-0.59
TrackID	-14.8852	1.7755	-0.64
TrackSt	-18.8020	2.2231	-0.73
Positio	-25.7033	3.0119	-0.92
Velocit	-0.0632	0.0815	1.25
Amplitu	-0.0978	0.0855	1.25
Detection	-0.2814	0.1064	1.25
TrackID	0.1129	0.1129	1.26
TrackSt	0.1270	0.1270	1.25
Positio	0.1450	0.1450	1.24
Velocit	0.1699	0.1699	1.23
Amplitu	-14.8815	1.8245	-0.64
Detection	-18.8008	2.2849	-0.74
TrackID	TrackSt	Positio	Velocit

Inertial measurement unit

Examples of automated driving sensor data

Camera

(640 x 480 x 3)

```
239 239 237 238 241 241 241 242 243  
252 252 251 252 252 253 253 253 253
```

Vision Detector

```
SensorID = 1;
```

```
Timestamp = 1461634696379742;
```

```
NumDetections = 6;
```

```
Dete
```

Lane Detector

```
Tr
```

```
C1
```

```
Po
```

```
Ve
```

```
Si
```

```
Detec
```

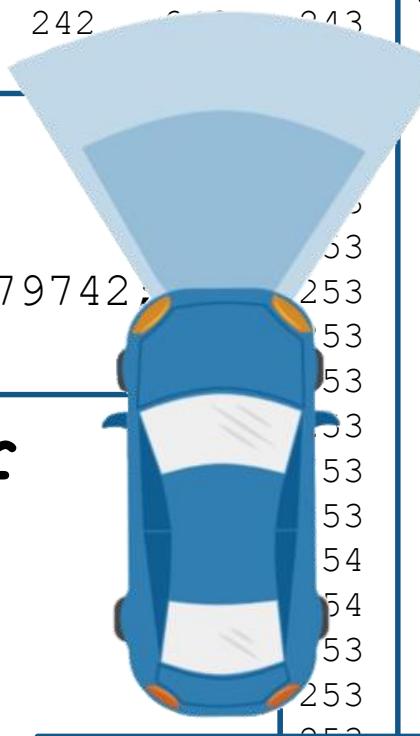
```
Tr
```

```
C1
```

```
Po
```

```
Ve
```

```
Si
```



Radar Detector

```
SensorID = 2;
```

```
Timestamp = 1461634696407521;
```

```
NumDetections = 23;
```

```
Detection
```

```
TrackID
```

```
TrackSt
```

```
Positio
```

```
Velocit
```

```
Amplitu
```

```
Detection
```

```
TrackID
```

```
TrackSt
```

Lidar

(47197 x 3)

-12.2911	1.4790	-0.59
-14.8852	1.7755	-0.64
-18.8020	2.2231	-0.73
-25.7033	3.0119	-0.92
-0.0632	0.0815	1.25
-0.0978	0.0855	1.25
-0.2814	0.1064	1.25
1.26	1.25	1.24
1.23	0.64	0.74

Inertial Measurement Unit

```
Timestamp: 1461634696379742
```

```
Velocity: 9.2795
```

```
YawRate: 0.0040
```

Visualize sensor data

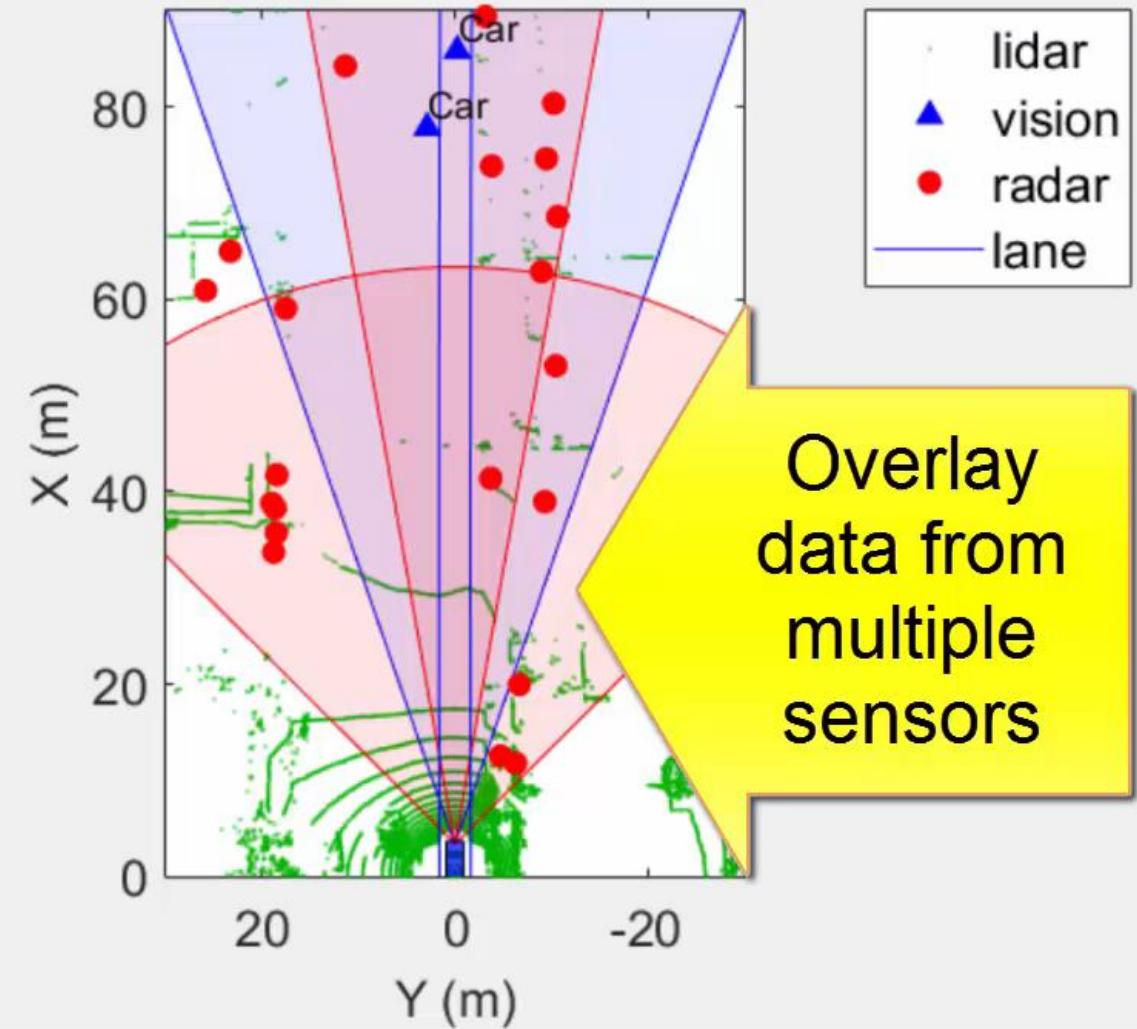


Visualize differences in sensor detections

Image Coordinates



Vehicle Coordinates



Explore logged vehicle data

- Load **video data** and corresponding **mono-camera parameters**

```
>> video = VideoReader('01_city_c2s_fcw_10s.mp4')  
>> load('FCWDemoMonoCameraSensor.mat', 'sensor')
```

- Load **detection sensor data** and corresponding **parameters**

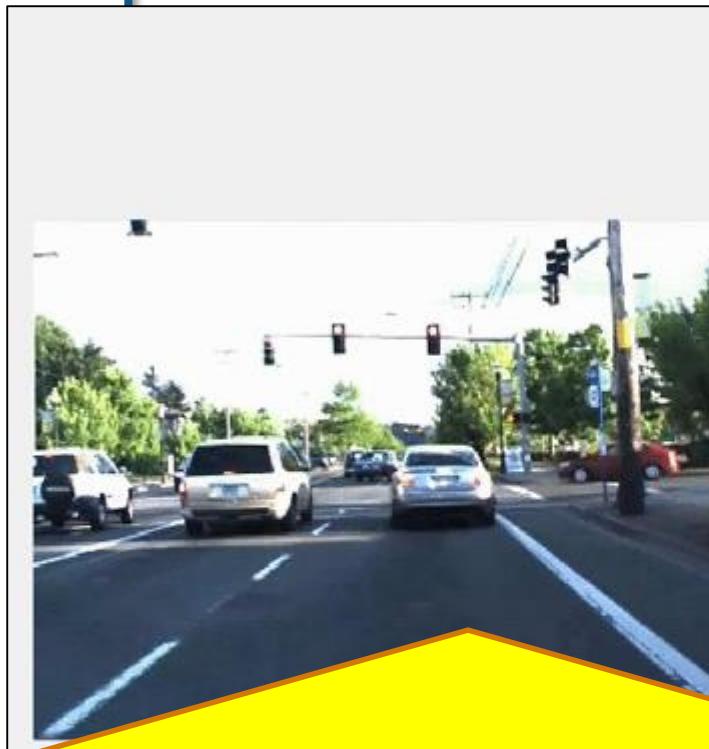
```
>> load('01_city_c2s_fcw_10s_sensor.mat', 'vision', 'lane', 'radar')  
>> load('SensorConfigurationData.mat', 'sensorParams')
```

- Load **lidar point cloud data**

```
>> load('01_city_c2s_fcw_10s_Lidar.mat', 'LidarPointCloud')
```

Visualize in image coordinates

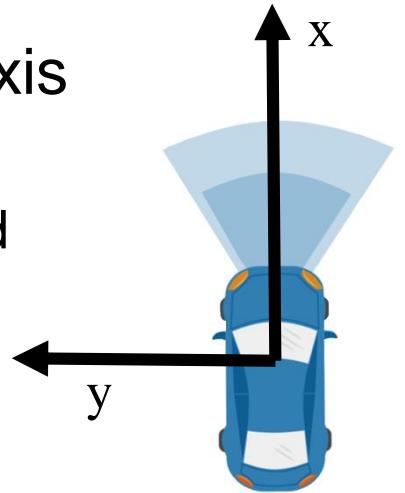
```
%% Specify time to inspect  
currentTime = 6.55;  
video.CurrentTime = currentTime;  
  
%% Extract video frame  
frame = video.readFrame;  
  
%% Plot image coordinates  
ax1 = axes(...  
    'Position',[0.02 0 0.55 1]);  
im = imshow(frame,...  
    'Parent',ax1);
```



Plot in image coordinates using
“classic” video and image functions like
imshow

Visualize in vehicle coordinates

- ISO 8855 vehicle axis coordinate system
 - Positive x is forward
 - Positive y is left



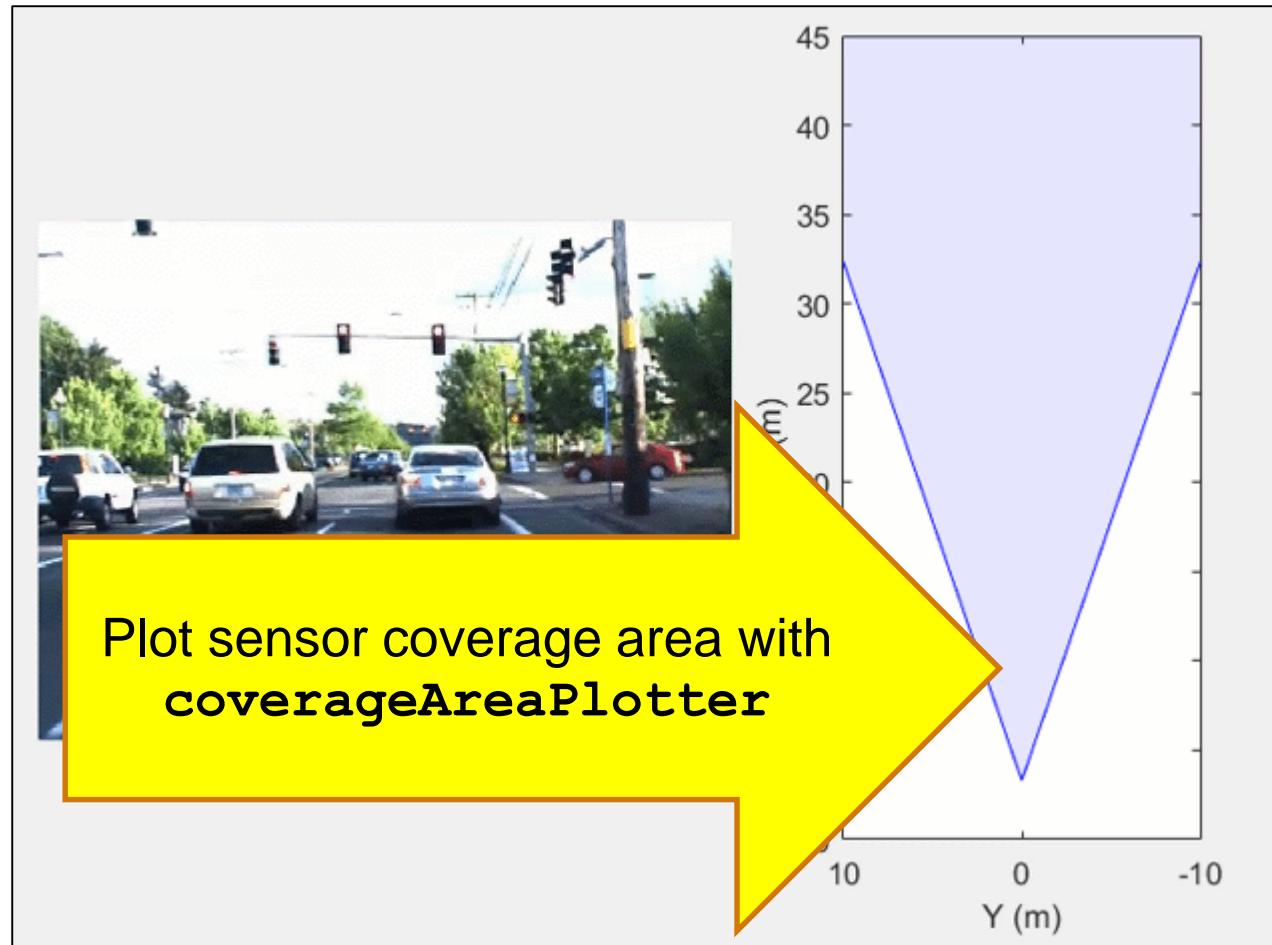
```
%% Plot in vehicle coordinates
ax2 = axes(...  
    'Position',[0.6 0.12 0.4 0.85]);  
  
bep = birdsEyePlot(...  
    'Parent',ax2,...  
    'Xlimits',[0 45],...  
    'Ylimits',[-10 10]);  
  
legend('off');
```



Visualize expected coverage area (vehicle coordinates)

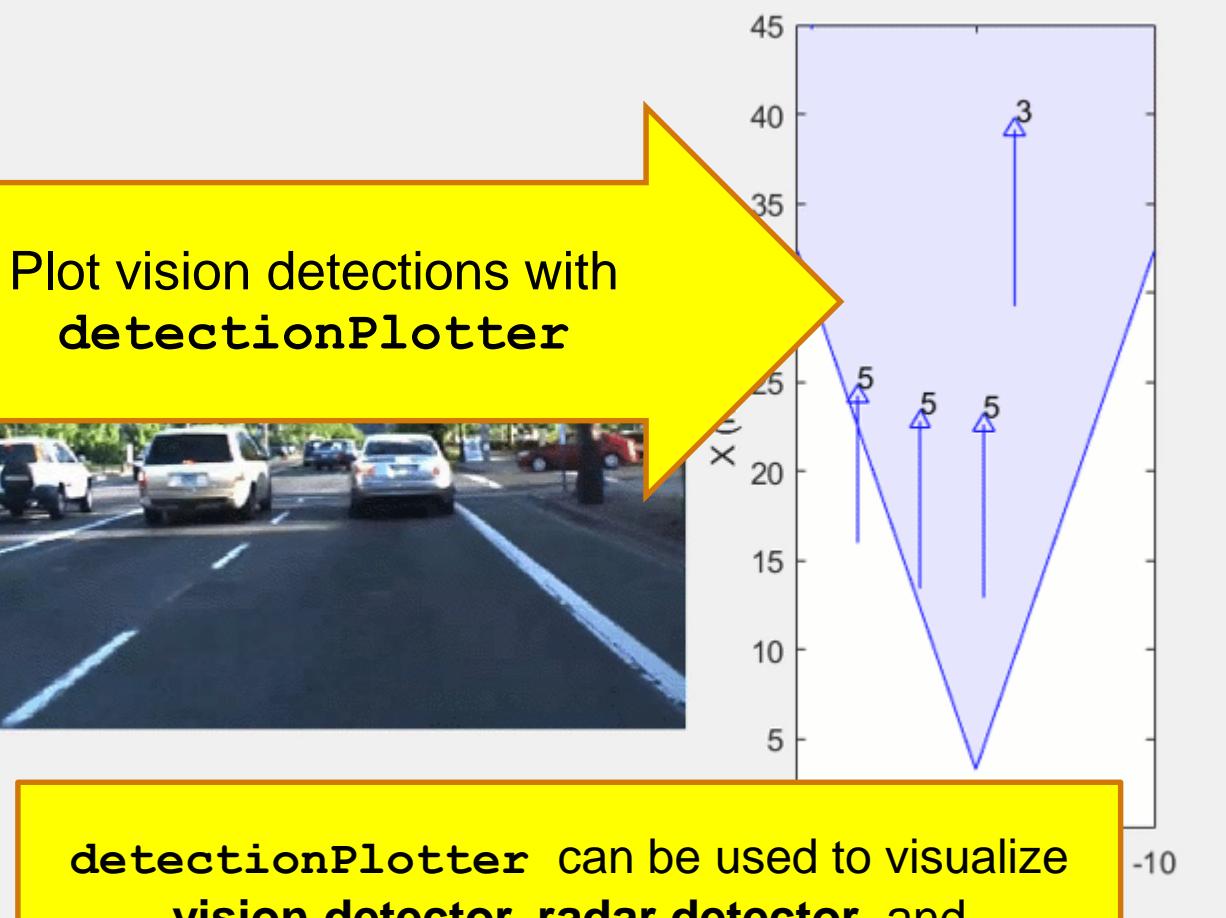
```
%% Create coverage area plotter
covPlot = coverageAreaPlotter(bep, ...
    'FaceColor','blue',...
    'EdgeColor','blue');

%% Update coverage area plotter
plotCoverageArea(covPlot, ...
    [sensorParams(1).X ... % Position x
    sensorParams(1).Y], ... % Position y
    sensorParams(1).Range, ...
    sensorParams(1).YawAngle, ...
    sensorParams(1).FoV(1)) % Field of view
```



Visualize detections (vehicle coordinates)

```
%% Create detection plotter  
detPlot = detectionPlotter(bep, ...  
    'MarkerEdgeColor', 'blue', ...  
    'Marker', '^');  
  
%% Update detection plotter  
n = round(currentTime/0.05);  
numDets = vision(n).numObjects;  
pos = zeros(numDets, 3);  
vel = zeros(numDets, 3);  
labels = repmat({''}, numDets, 1);  
for k = 1:numDets  
    pos(k, :) = vision(n).object(k).position;  
    vel(k, :) = vision(n).object(k).velocity;  
    labels{k} = num2str(...  
        vision(n).object(k).classification);  
end  
  
plotDetection(detPlot, pos, vel, labels);
```



detectionPlotter can be used to visualize **vision detector**, **radar detector**, and **lidar point cloud**

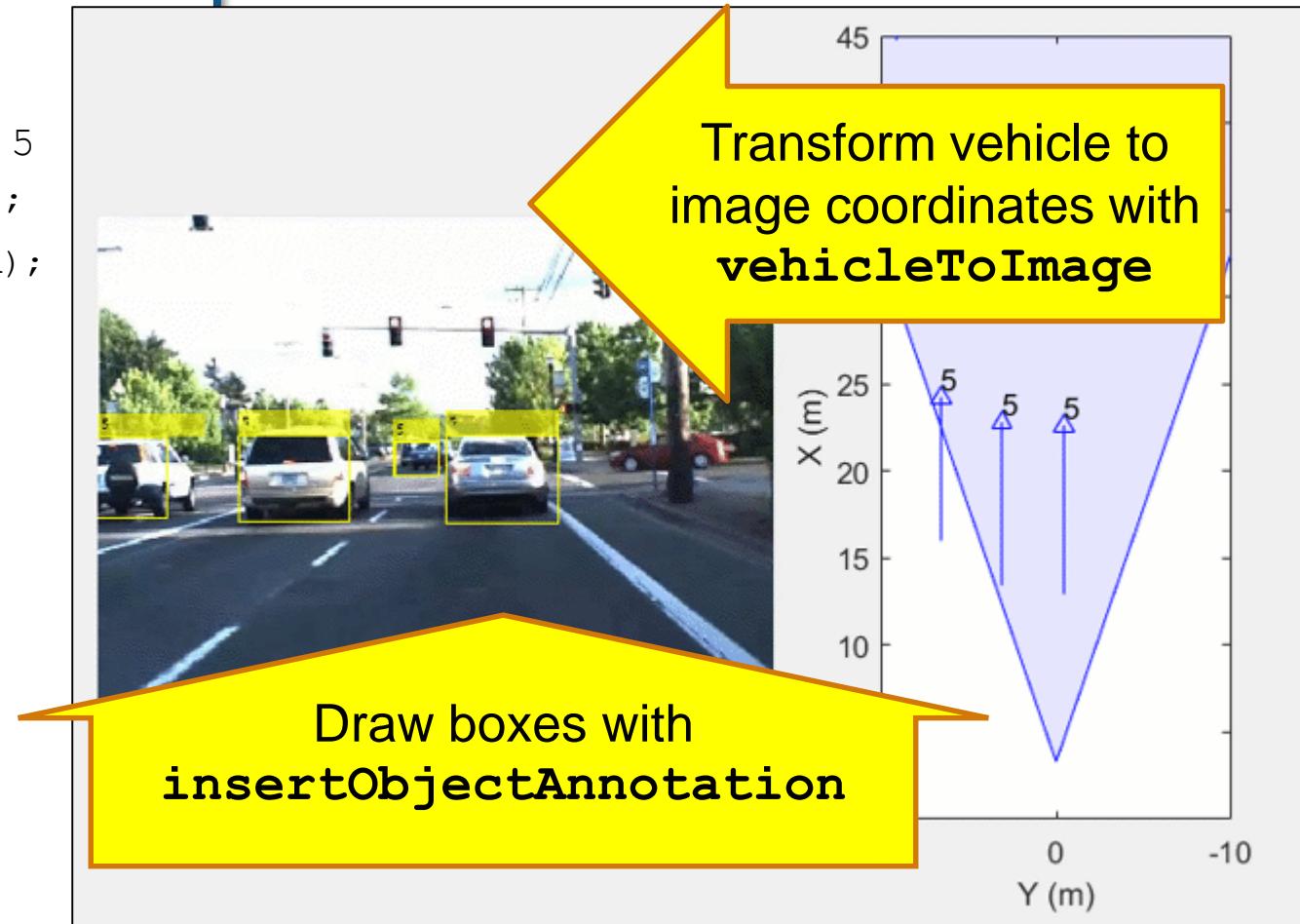
Visualize detections (image coordinates)

```

%% Bounding box positions in image coordinates
imBoxes = zeros(numDets, 4);
for k = 1:numDets
    if vision(n).object(k).classification == 5
        vehPosLR = vision(n).object(k).position(1:2)';
        imPosLR = vehicleToImage(sensor, vehPosLR);
        boxHeight = 1.4 * 1333 / vehPosLR(1);
        boxWidth = 1.8 * 1333 / vehPosLR(1);
        imBoxes(k,:)=[imPosLR(1) - boxWidth/2, ...
                      imPosLR(2) - boxHeight, ...
                      boxWidth, boxHeight];
    end
end

%% Draw bounding boxes on image frame
frame = insertObjectAnnotation(frame, ...
    'Rectangle', imBoxes, labels, ...
    'Color', 'yellow', 'LineWidth', 2);
im.CData = frame;

```

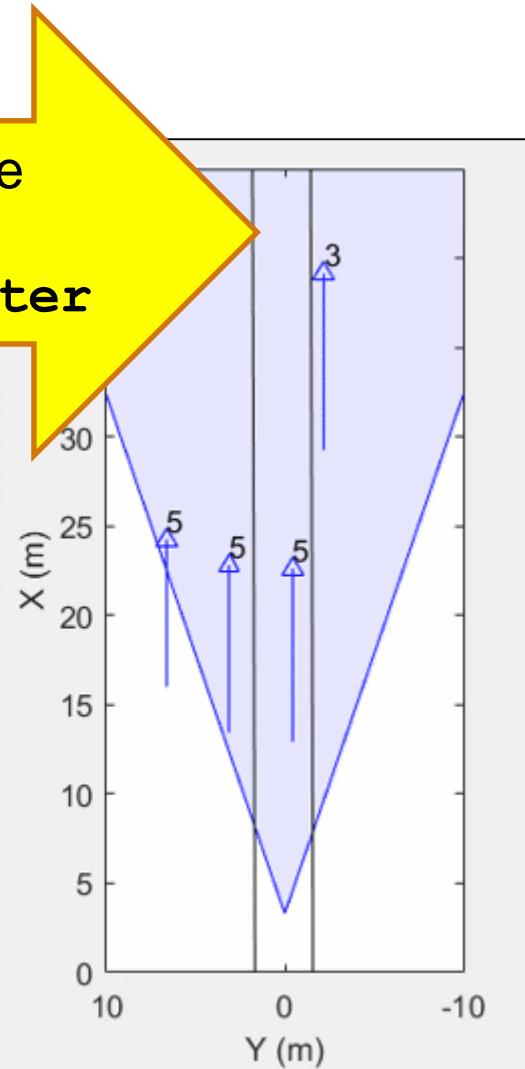
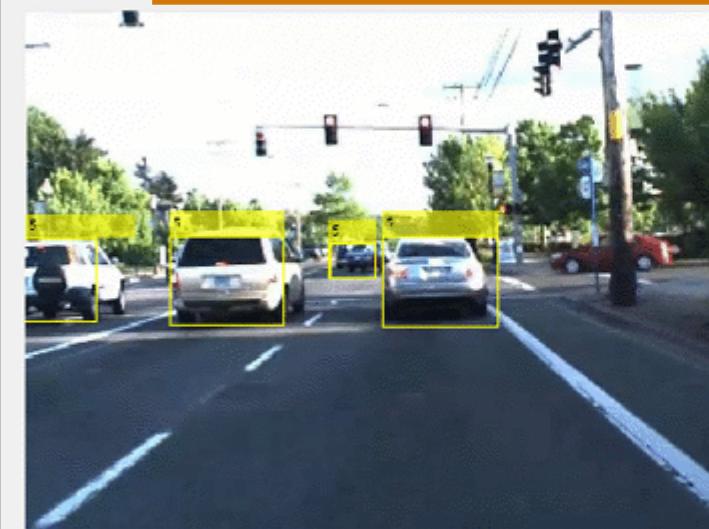


Visualize lane boundaries (vehicle coordinates)

```
%% Create lane detection plotter
lanePlot = laneBoundaryPlotter(bep, ...
    'Color','black');

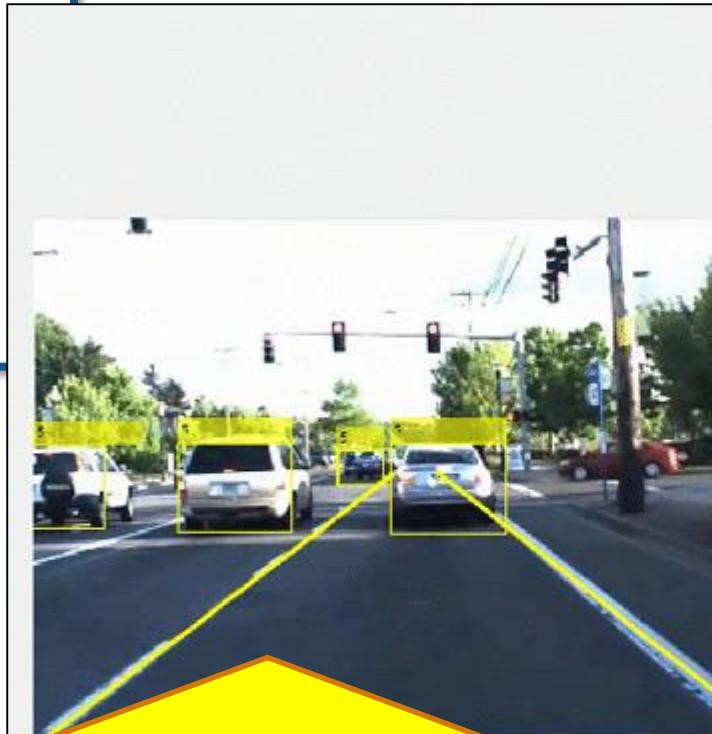
%% Update lane detection plotter
lb = parabolicLaneBoundary([...
    lane(n).left.curvature, ...
    lane(n).left.headingAngle, ...
    lane(n).left.offset]);
rb = parabolicLaneBoundary([...
    lane(n).right.curvature, ...
    lane(n).right.headingAngle, ...
    lane(n).right.offset]);
plotLaneBoundary(lanePlot, [lb rb])
```

Plot lanes in vehicle
coordinates with
laneBoundaryPlotter

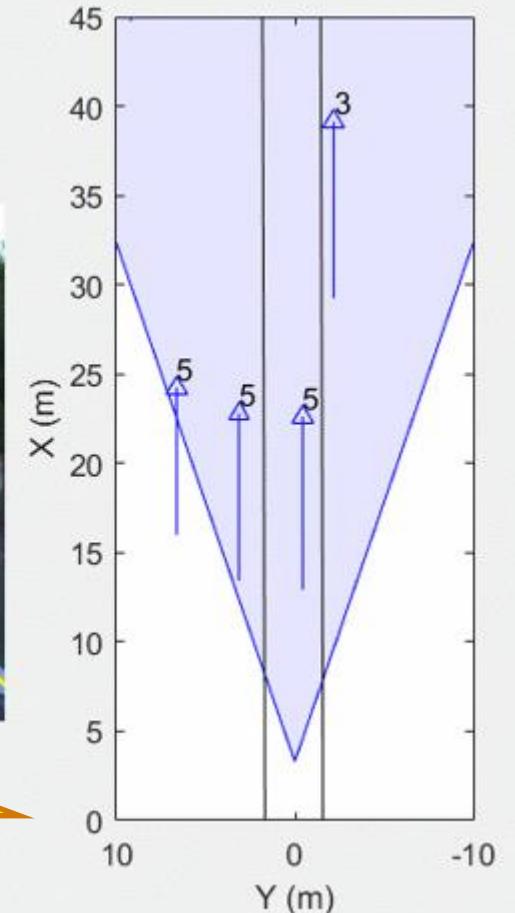


Visualize lane boundaries (image coordinates)

```
%% Draw in image coordinates  
frame = insertLaneBoundary(frame, ...  
    [lb rb], sensor, (1:100), ...  
    'LineWidth', 5);  
  
im.CData = frame;
```



Plot lanes in image coordinates with **insertLaneBoundary**



Visualize radar detections (vehicle coordinates)

```
%% Create radar detection plotter
radarPlot = detectionPlotter(bep, ...
    'MarkerEdgeColor', 'red', ...
    'Marker', 'o');

%% Update radar detection plotter
numDets = radar(n).numObjects;
pos = zeros(numDets, 3);
vel = zeros(numDets, 3);
for k = 1:numDets
    pos(k, :) = radar(n).object(k).position;
    vel(k, :) = radar(n).object(k).velocity;
end
plotDetection(radarPlot, pos, vel);
```



Plot radar detections just like vision detections with **detectionPlotter**

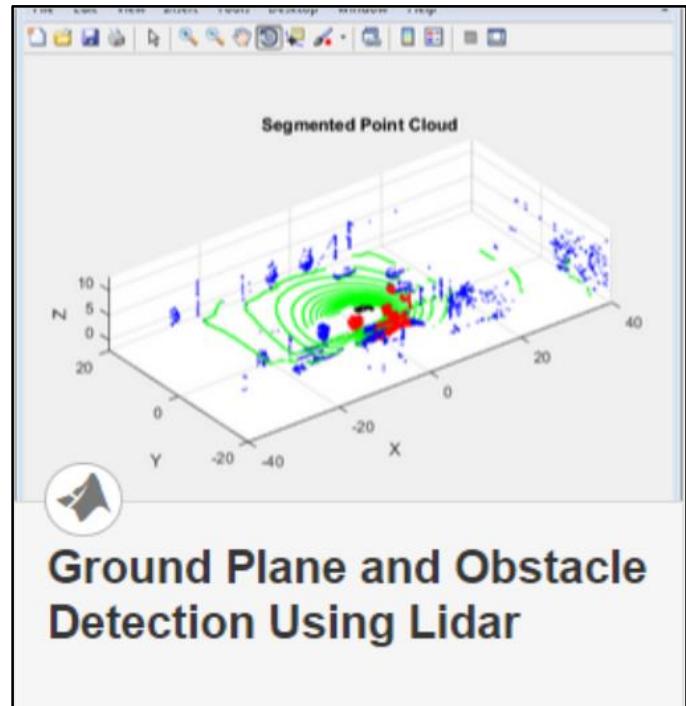
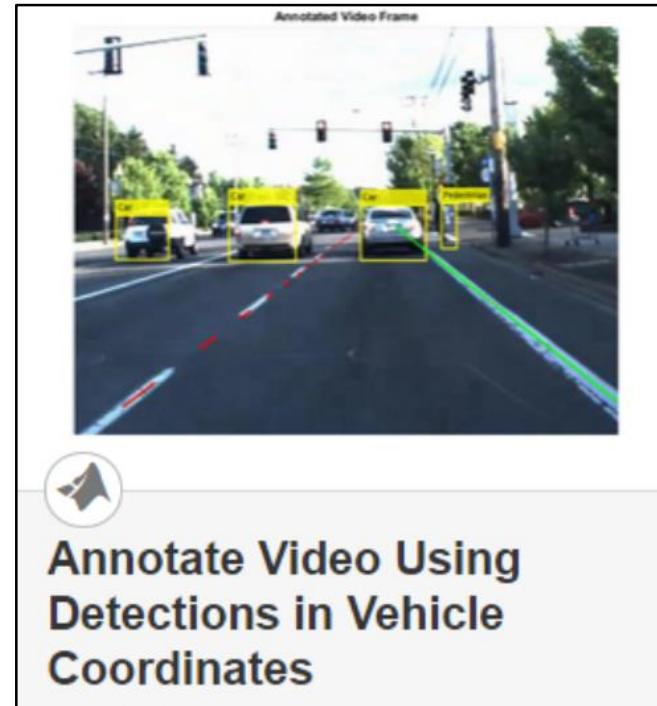
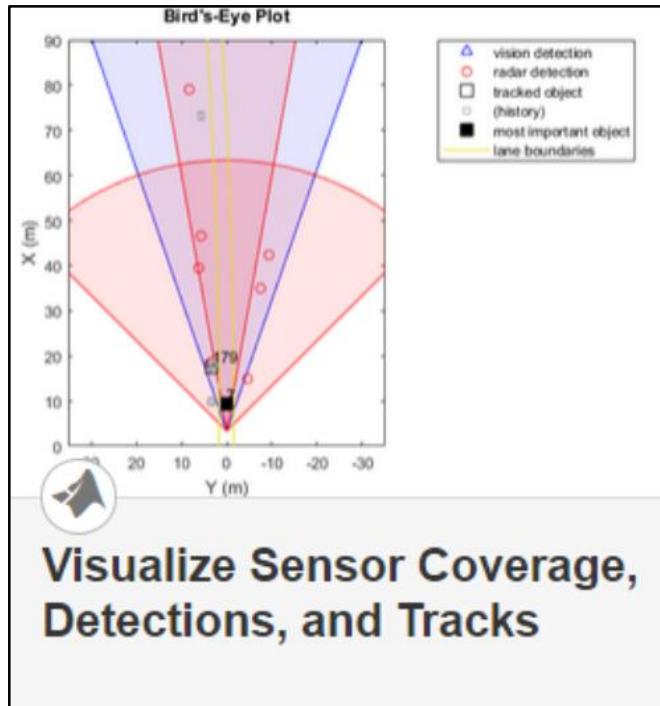
Visualize lidar point cloud (vehicle coordinates)

```
%% Create lidar detection plotter  
  
lidarPlot = detectionPlotter(bep, ...  
    'Marker', '.', ...  
    'MarkerSize', 1.5, ...  
    'MarkerEdgeColor', [0 0.7 0]); % Green  
  
%% Update lidar detection plotter  
n = round(video.CurrentTime/0.1);  
pos = ...  
LidarPointCloud(n).ptCloud.Location(:,1:2);  
  
plotDetection(lidarPlot, pos);
```



Learn more about visualizing vehicle data

by exploring examples in the Automated Driving System Toolbox



- **Plot object detectors in vehicle coordinates**
 - Vision & radar detector
 - Lane detectors
 - Detector coverage areas

- **Transform between vehicle and image coordinates**

- **Plot lidar point cloud**

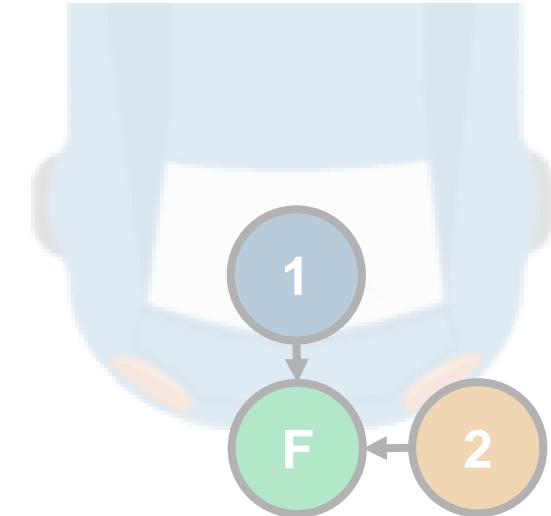
Some common questions from automated driving engineers



How can I
visualize vehicle
data?

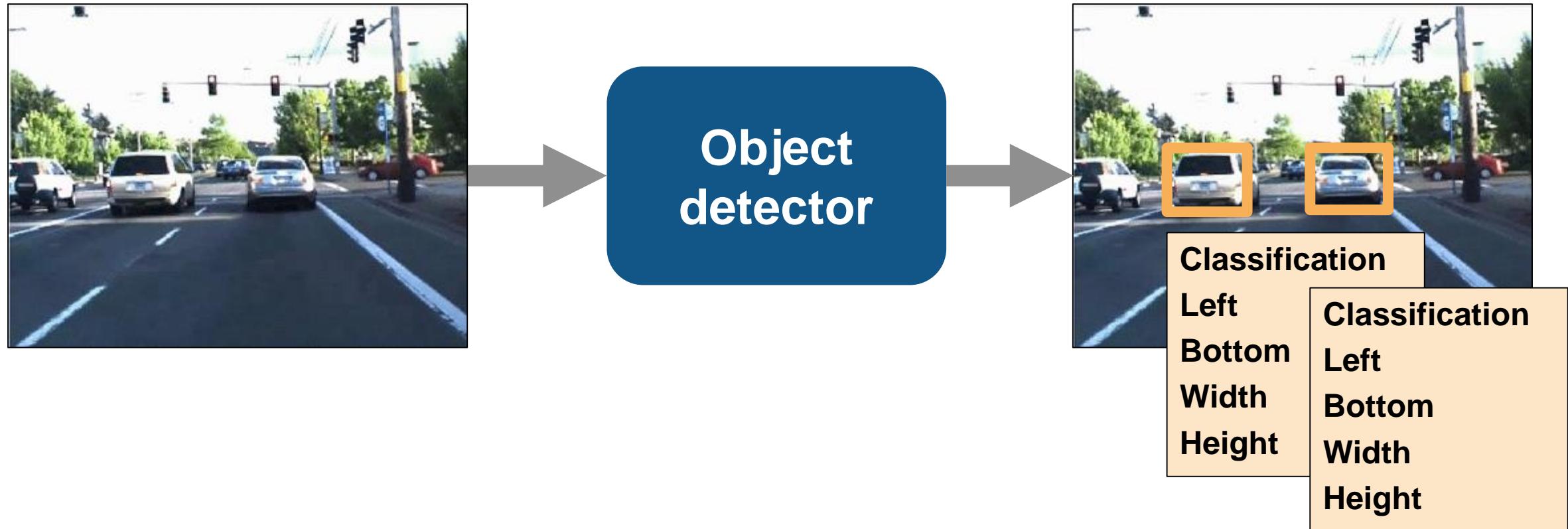


How can I
detect objects in
images?

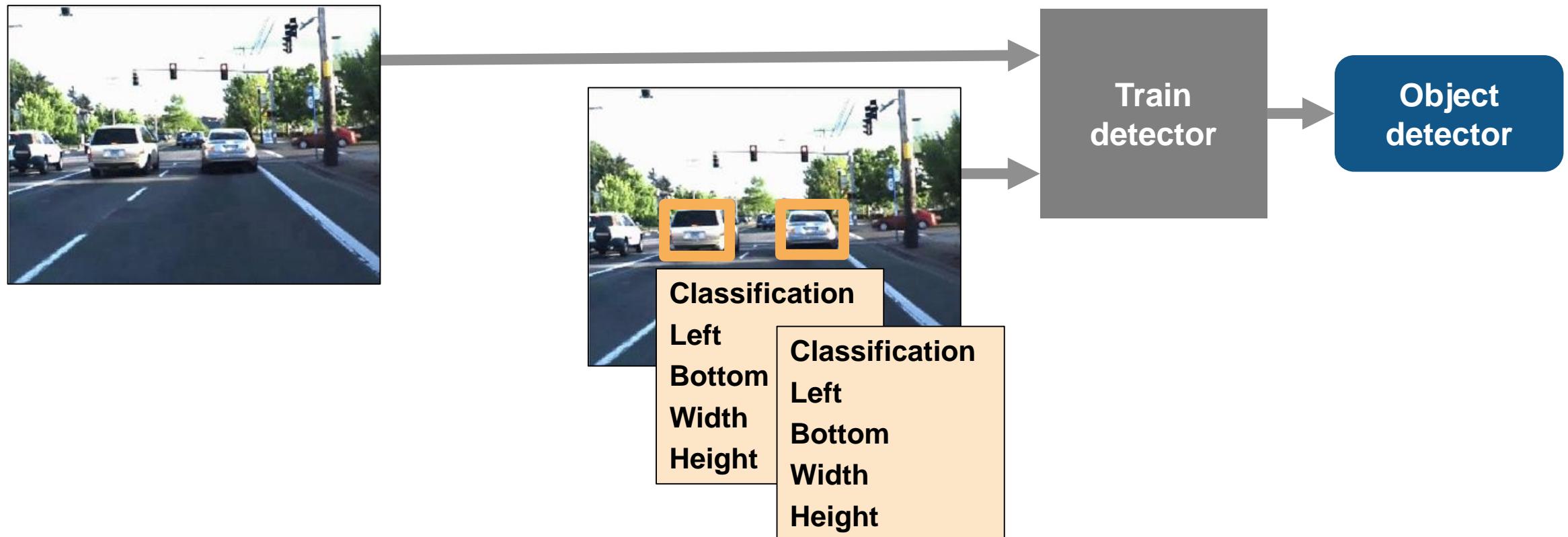


How can I
fuse multiple
detections?

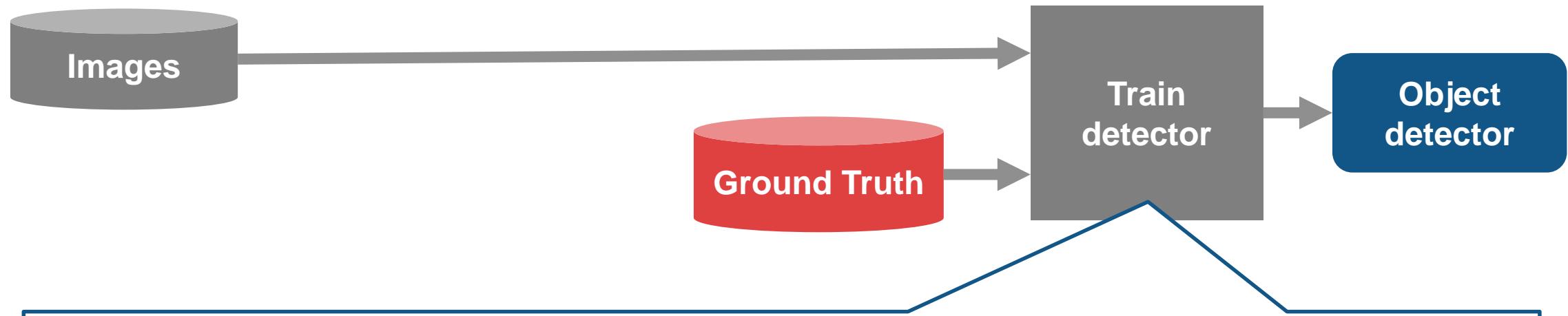
How can I detect objects in images?



Train object detectors based on ground truth



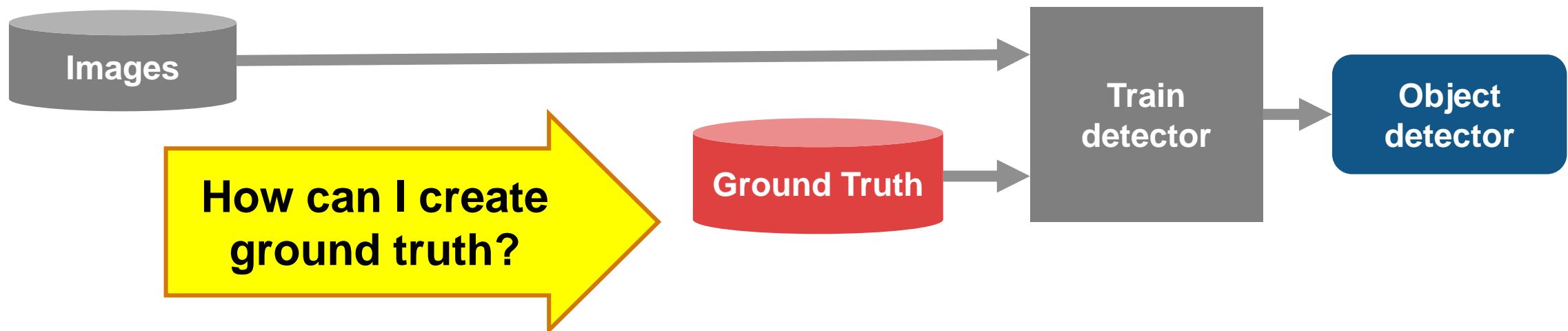
Train object detectors based on ground truth



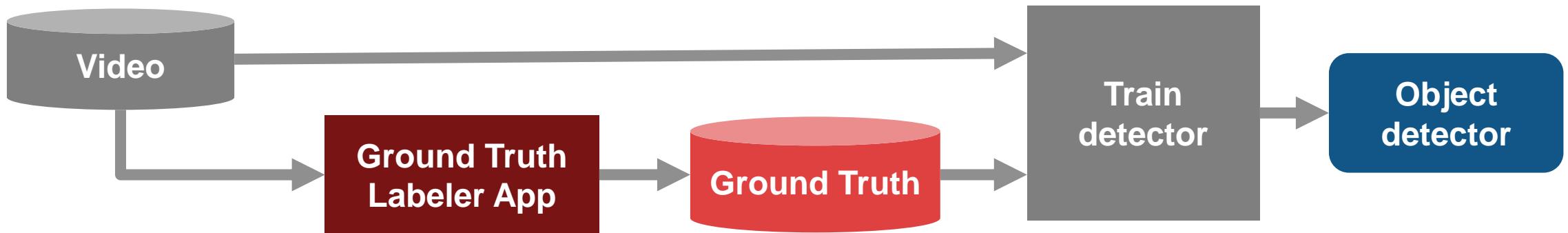
Design object detectors with the Computer Vision System Toolbox

Machine Learning	Aggregate Channel Feature	<code>trainACFObjectDetector</code>
	Cascade	<code>trainCascadeObjectDetector</code>
Deep Learning	R-CNN (Regions with Convolutional Neural Networks)	<code>trainRCNNObjectDetector</code>
	Fast R-CNN	<code>trainFastRCNNObjectDetector</code>
	Faster R-CNN	<code>trainFasterRCNNObjectDetector</code>

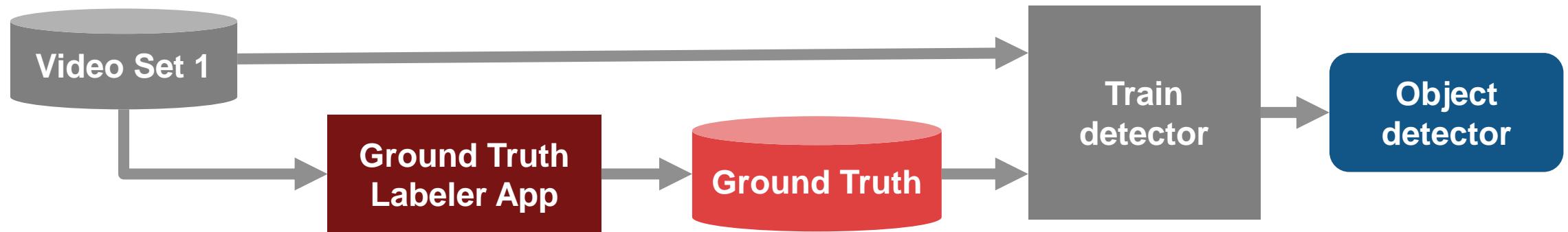
Specify ground truth to train detector



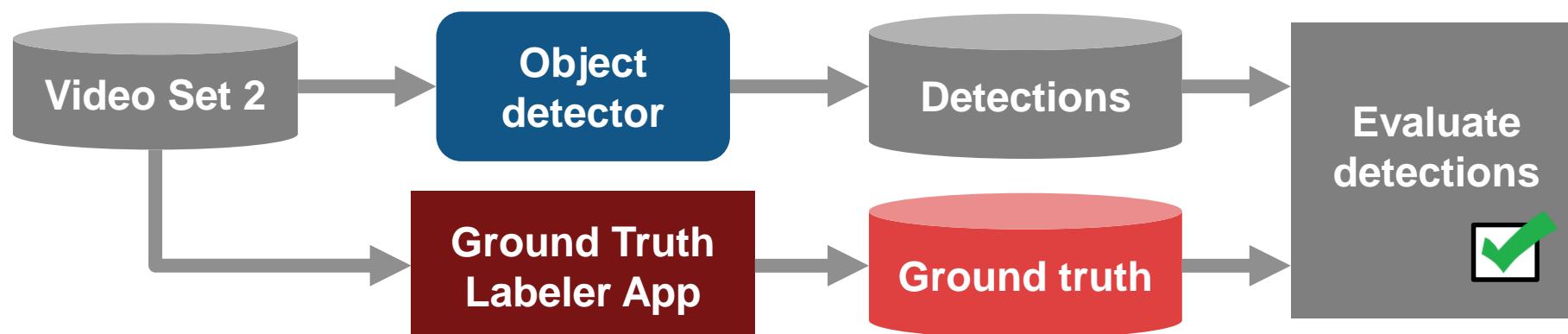
Specify ground truth to train detector



Specify ground truth to train detectors

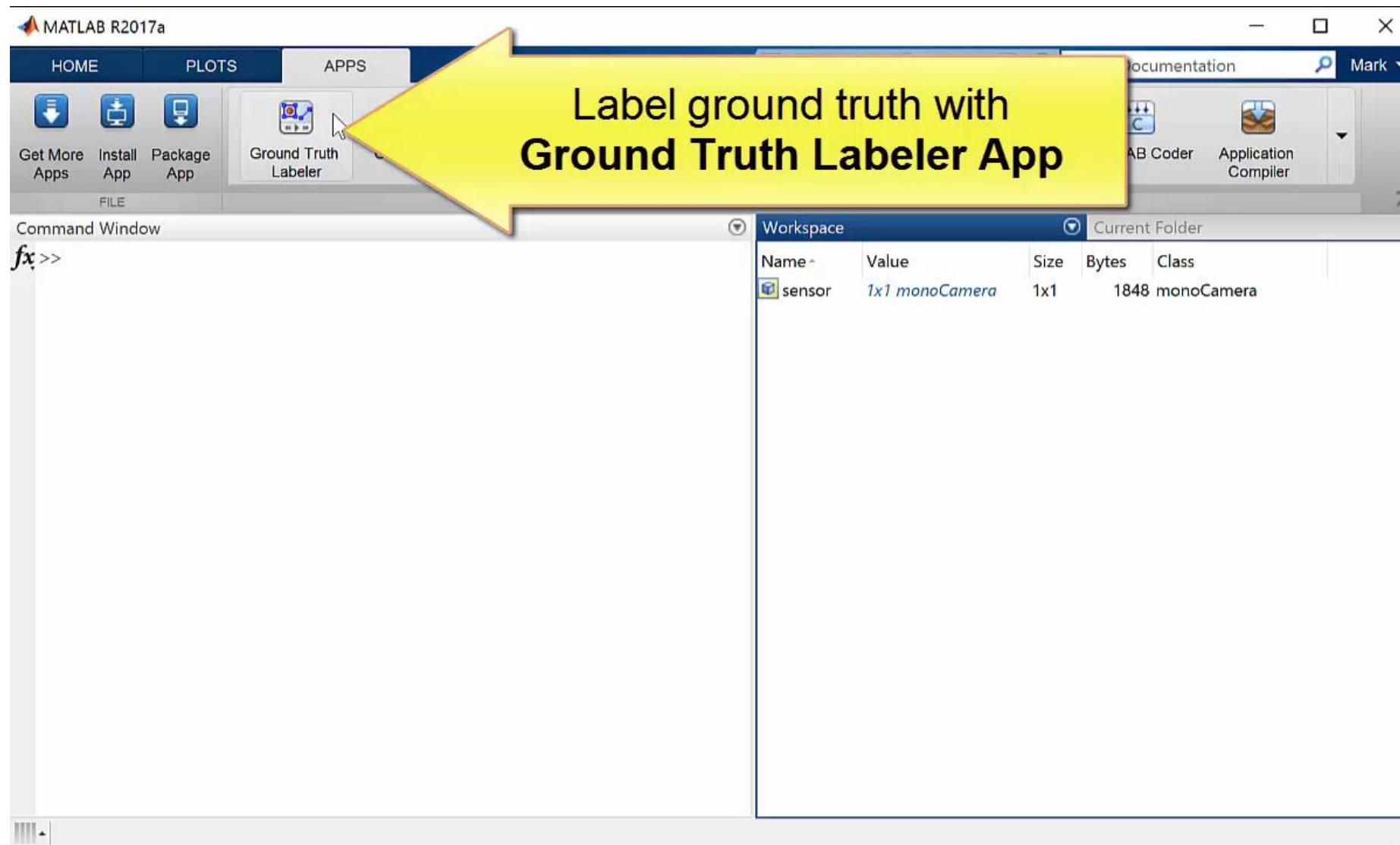


Specify ground truth to evaluate detectors

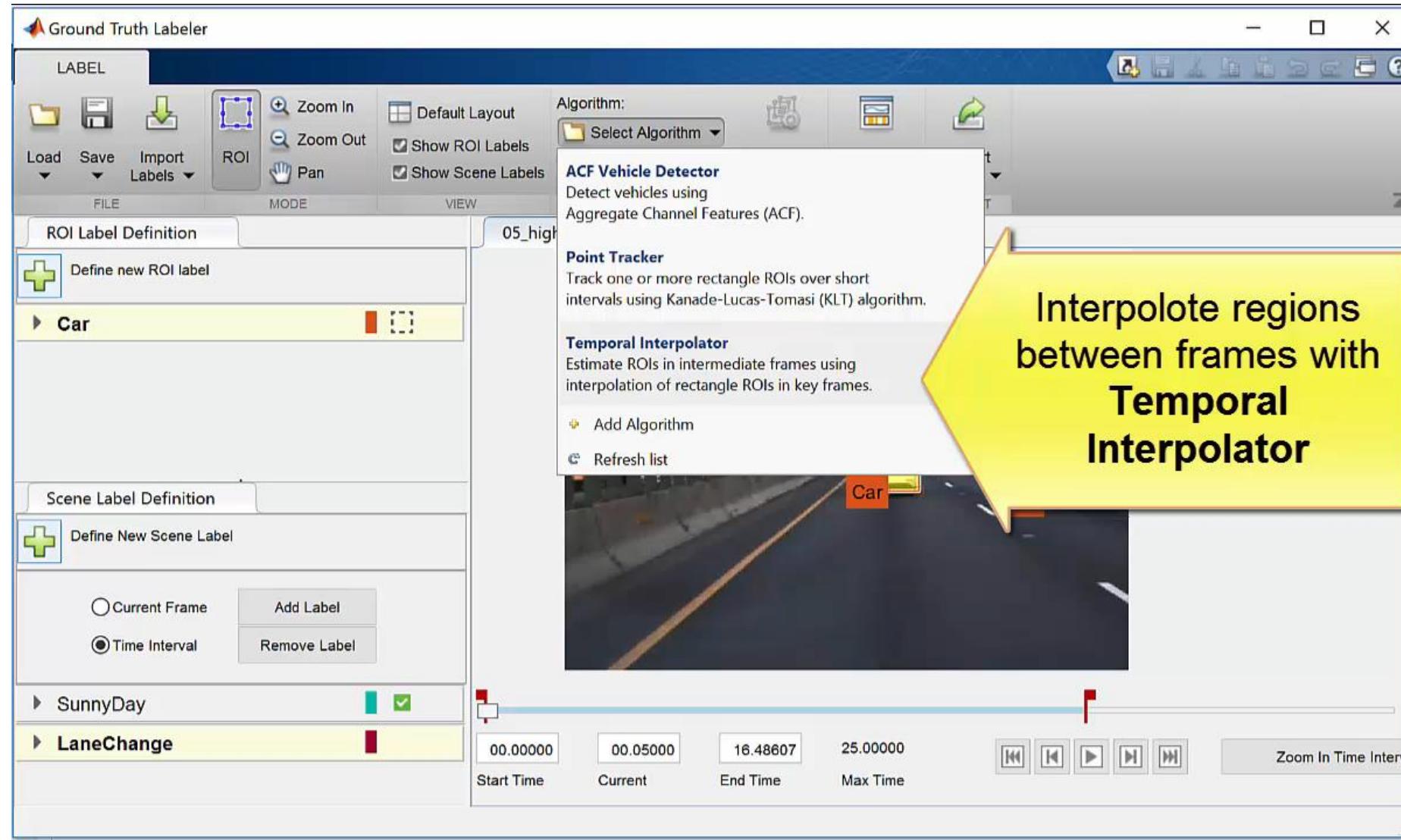


Manually label ground truth objects

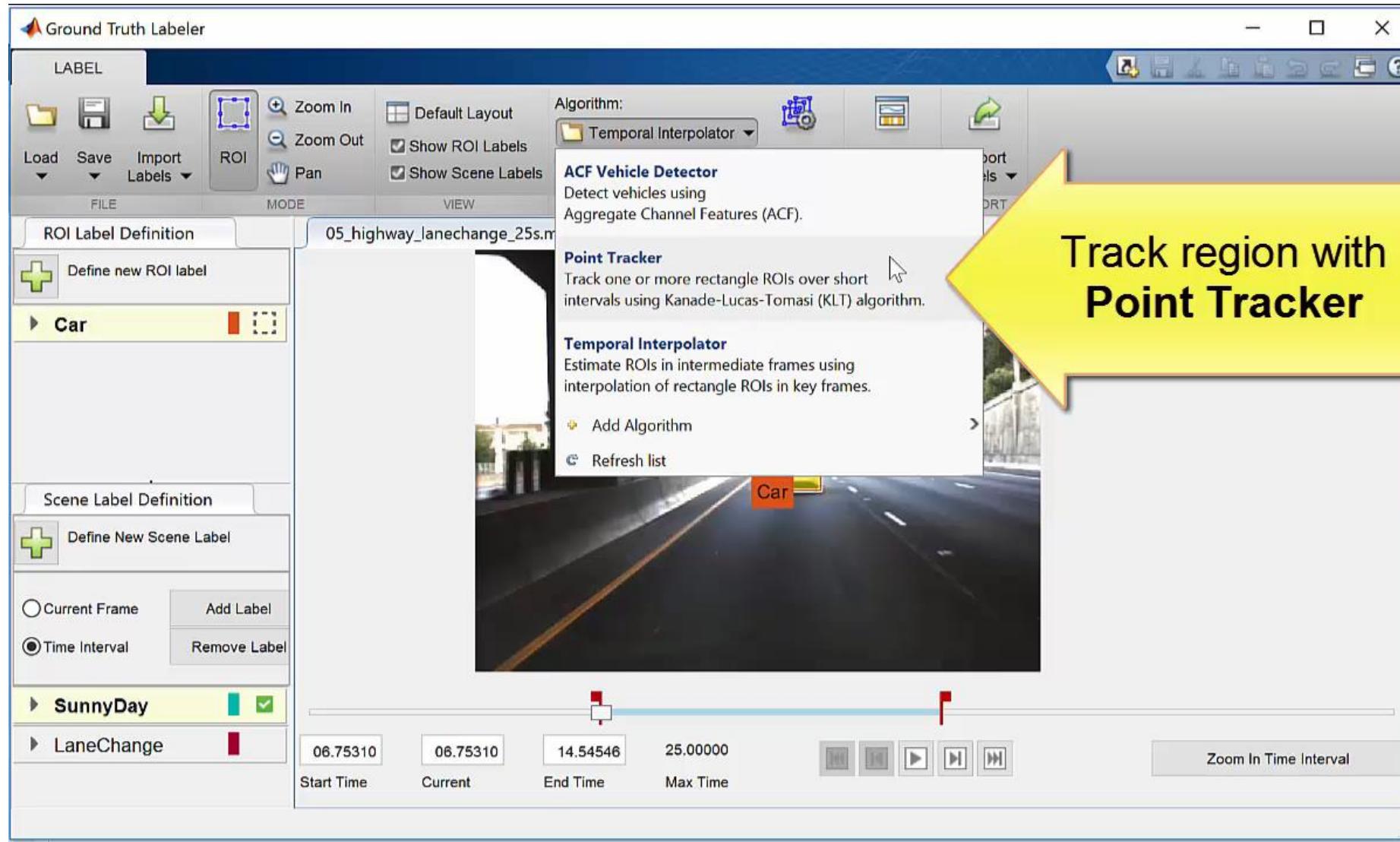
with Ground Truth Labeling App



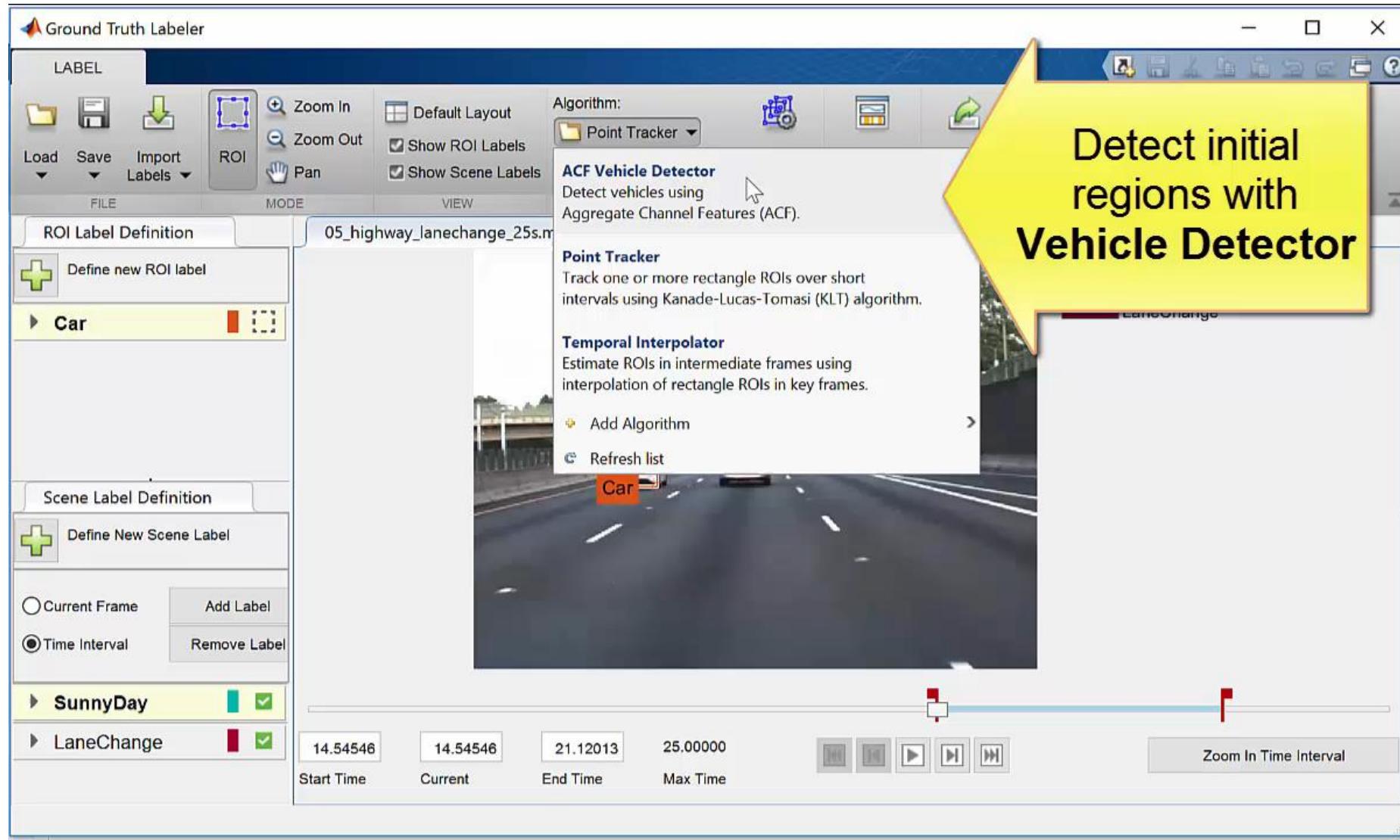
Automate labeling between manually labeled frames with temporal interpolator



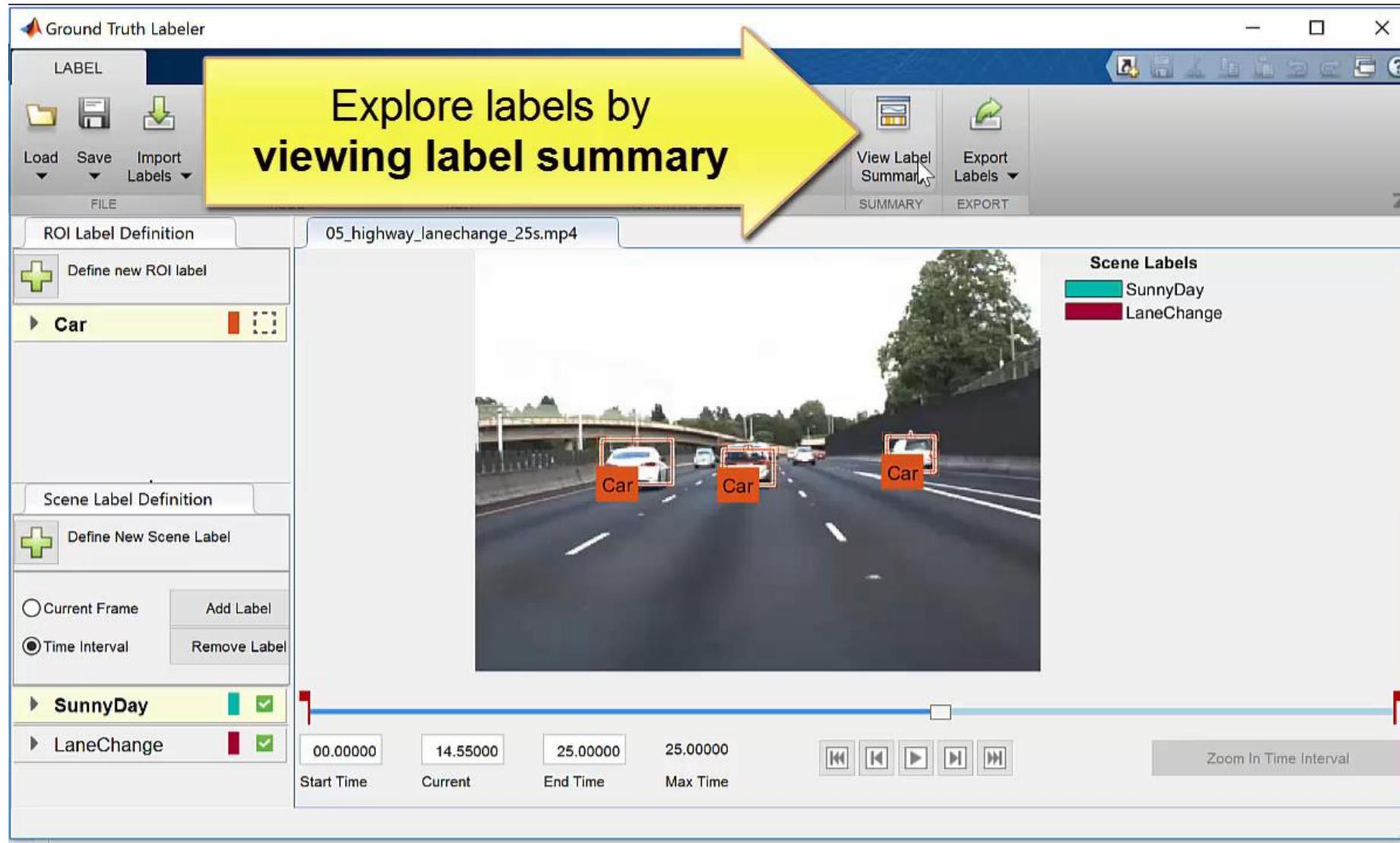
Automate labeling based on a manually labeled frame with point tracker



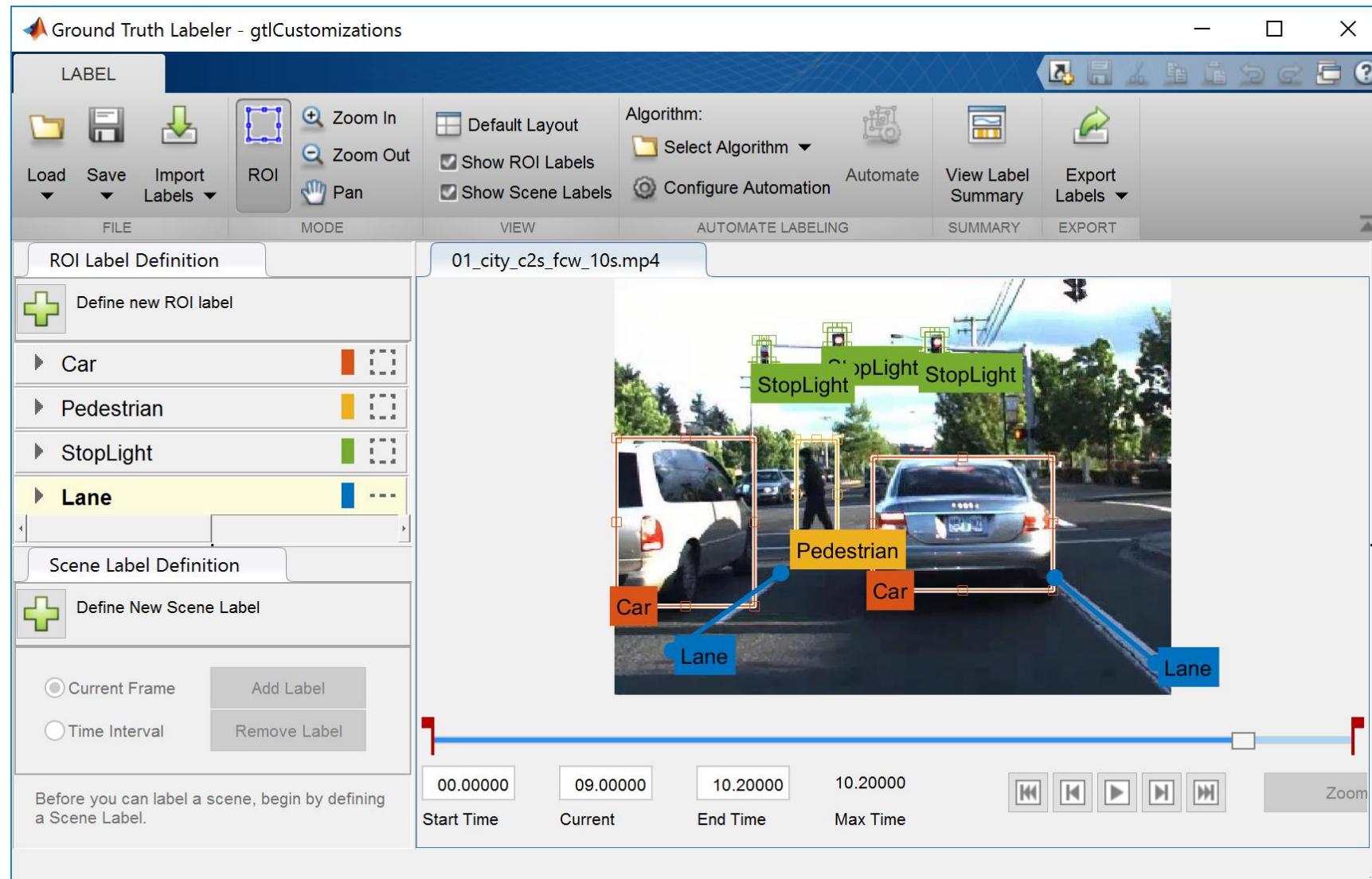
Automate initial ground truth of vehicles with ACF ground truth detector



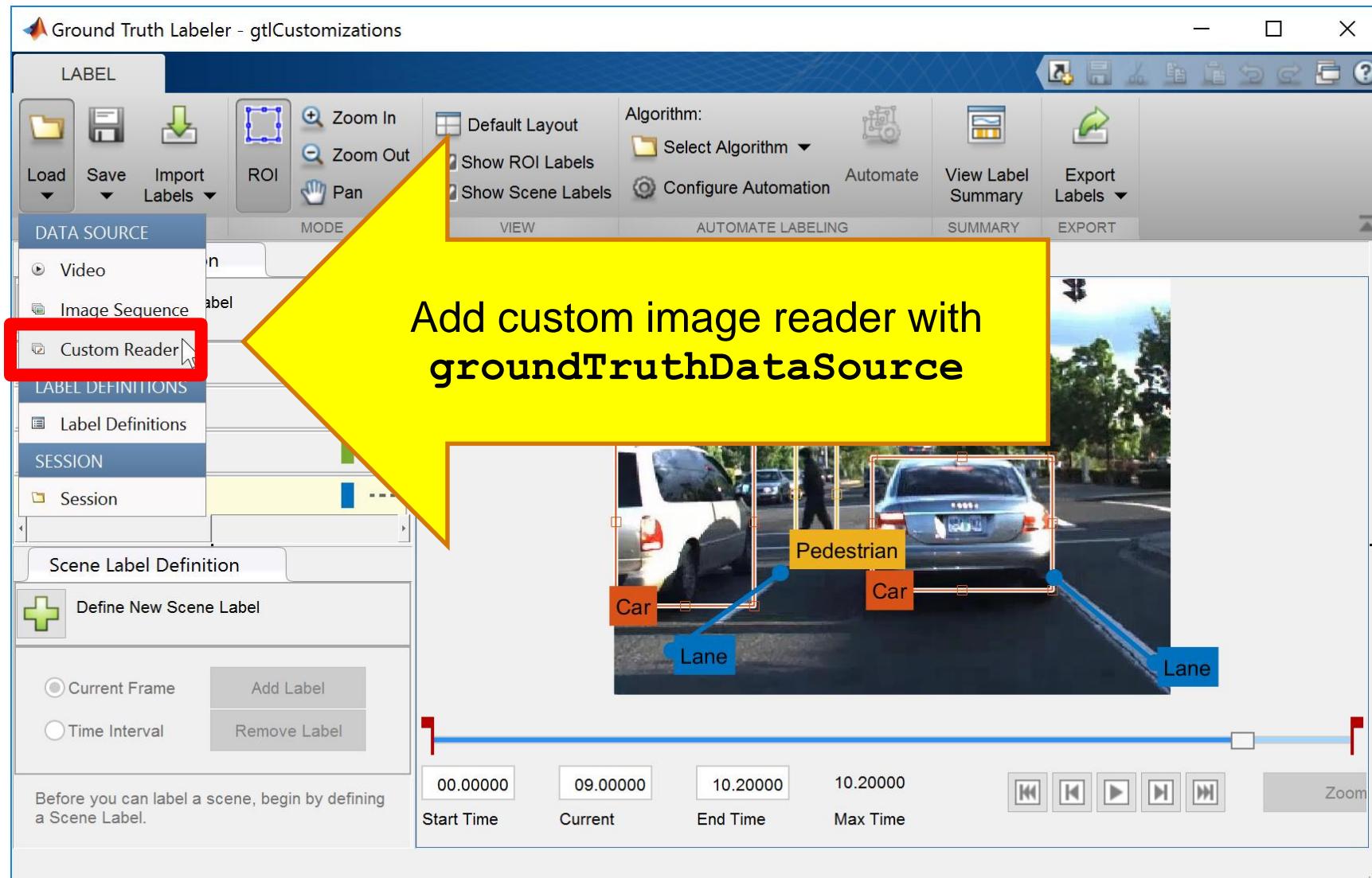
Export labeled regions as MATLAB time table



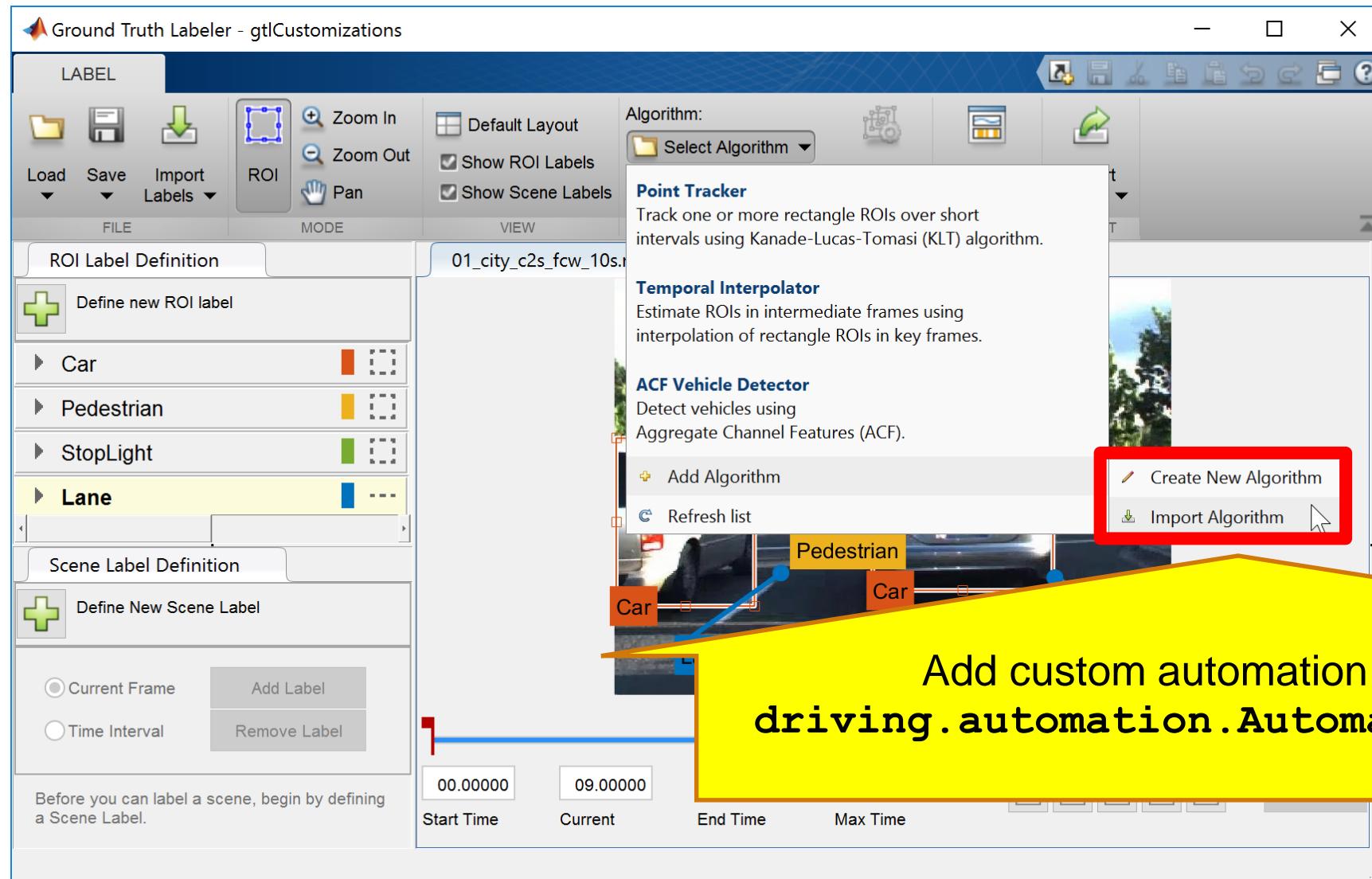
Customize Ground Truth Labeler App



Customize Ground Truth Labeler App



Customize Ground Truth Labeler App



Customize Ground Truth Labeler App

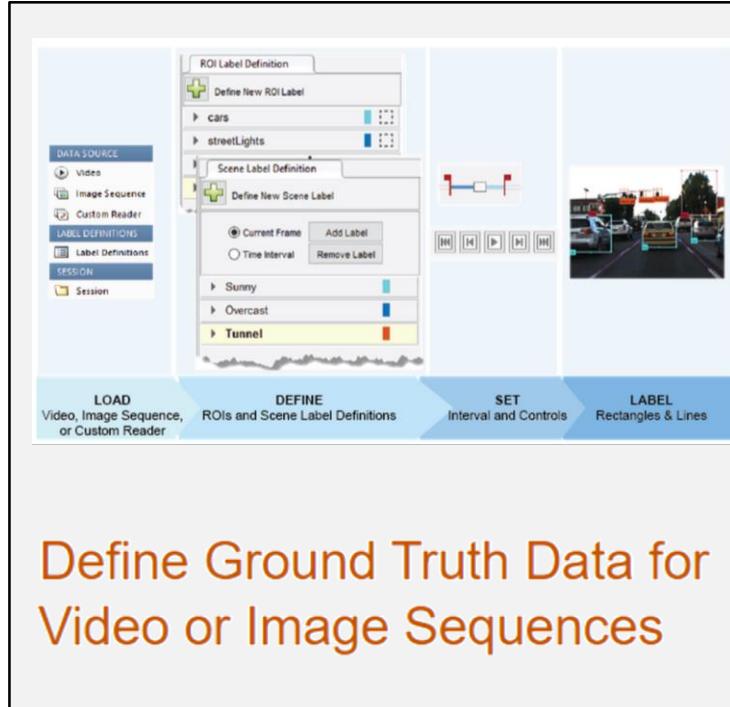
The image shows two MATLAB windows. The main window is the 'Ground Truth Labeler' app, titled 'gtlCustomizations'. It has a toolbar with 'FILE' and 'MODE' buttons, and a central view showing a video frame with three labeled objects: 'Pedestrian' (yellow), 'Car' (orange), and 'Lane' (blue). Below the video frame is a timeline with time points 00.00000, 09.00000, 10.20000, and 10.20000. A large yellow callout box points from the text in the center to this timeline area. The text in the callout box reads: 'Add connection to other tools with **driving.connector.Connector**'. To the right of the main window is another MATLAB figure window titled 'Figure 1: Point Cloud Pla...'. This window displays a 3D point cloud visualization of a scene with blue lines representing lane markings.

Add connection to other tools with
driving.connector.Connector

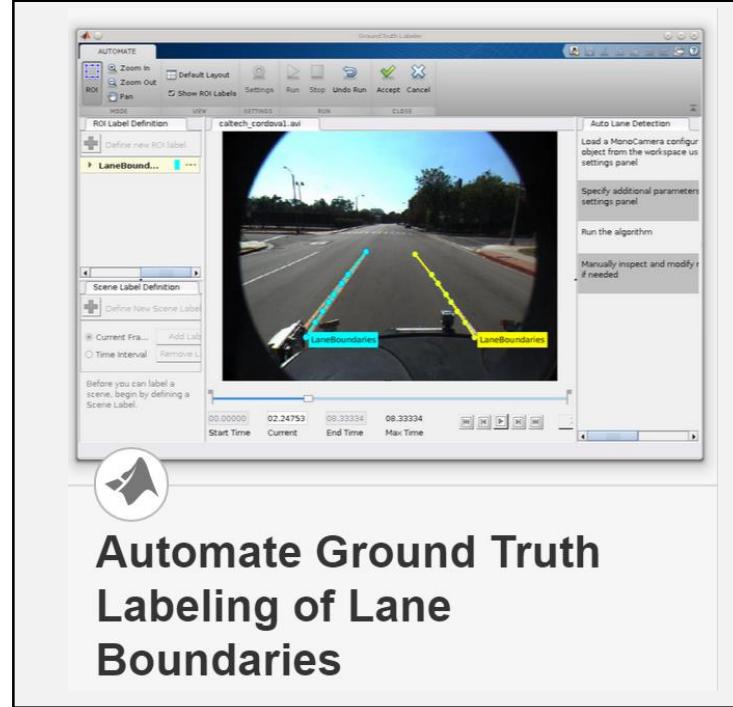
Before you can label a scene, begin by defining a Scene Label.

Learn more about detecting objects in images

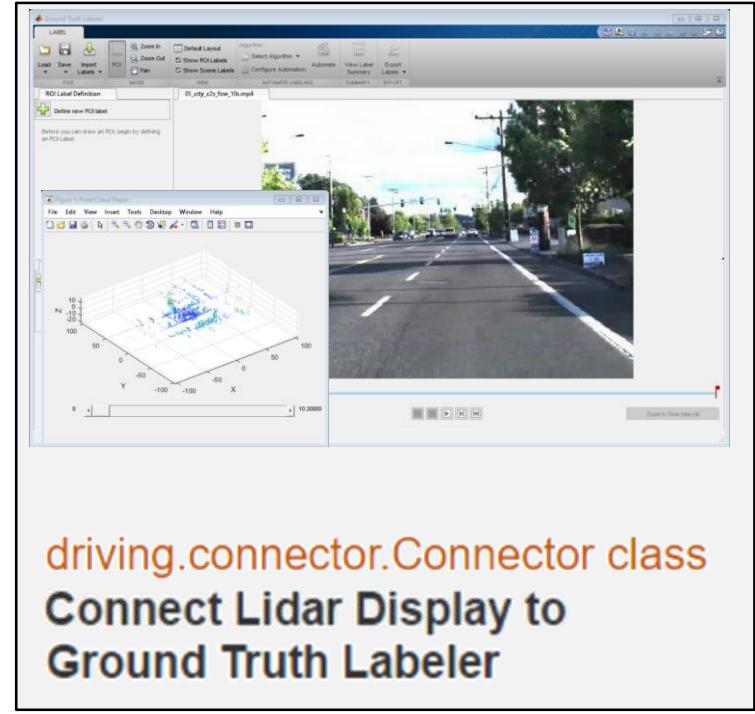
by exploring examples in the Automated Driving System Toolbox



- **Label detections with Ground Truth Labeler App**



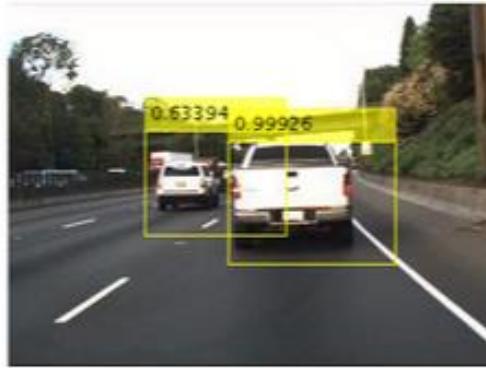
- **Add automation algorithm for lane tracking**



- **Extend connectivity of Ground Truth Labeler App**

Learn more about detecting objects in images

by exploring examples in the Automated Driving System Toolbox



Train a Deep Learning Vehicle Detector



Track Pedestrians from a Moving Car



Visual Perception Using Monocular Camera

- **Train object detector** using deep learning and machine learning techniques

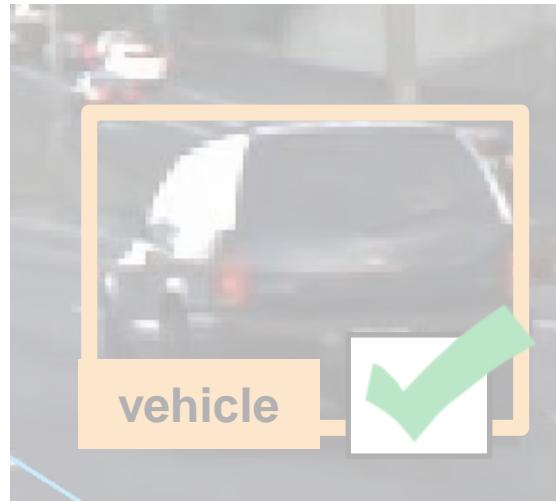
- **Explore pre-trained pedestrian detector**

- **Explore lane detector** using coordinate transforms for mono-camera sensor model

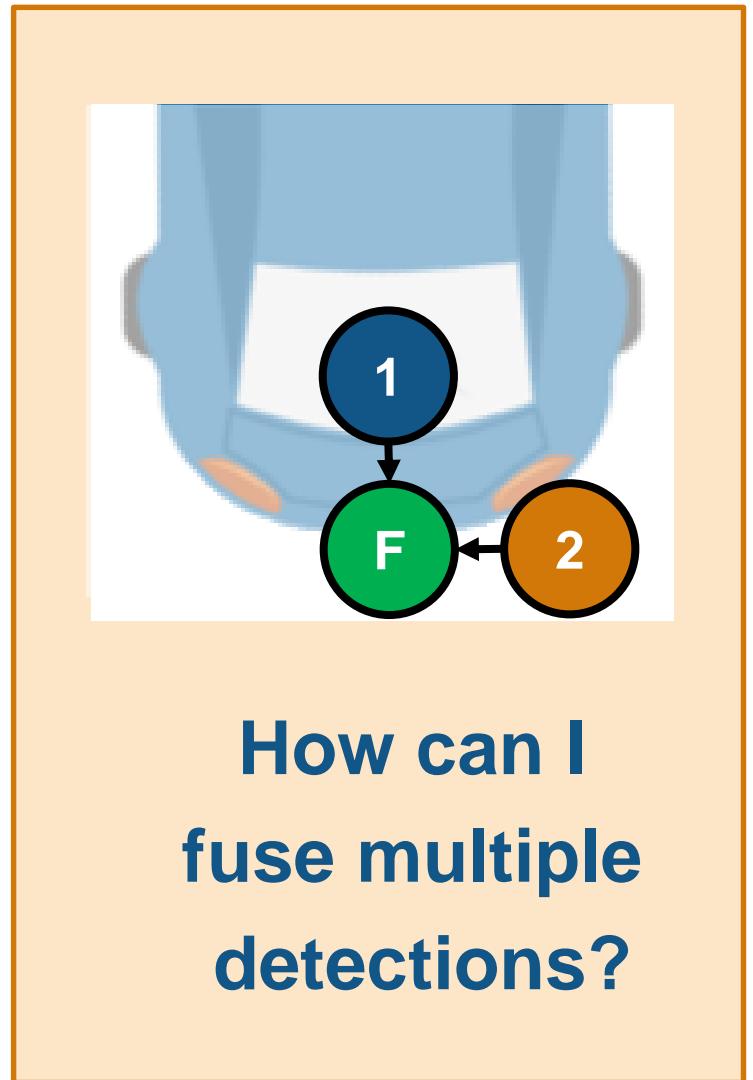
Some common questions from automated driving engineers



How can I
visualize vehicle
data?

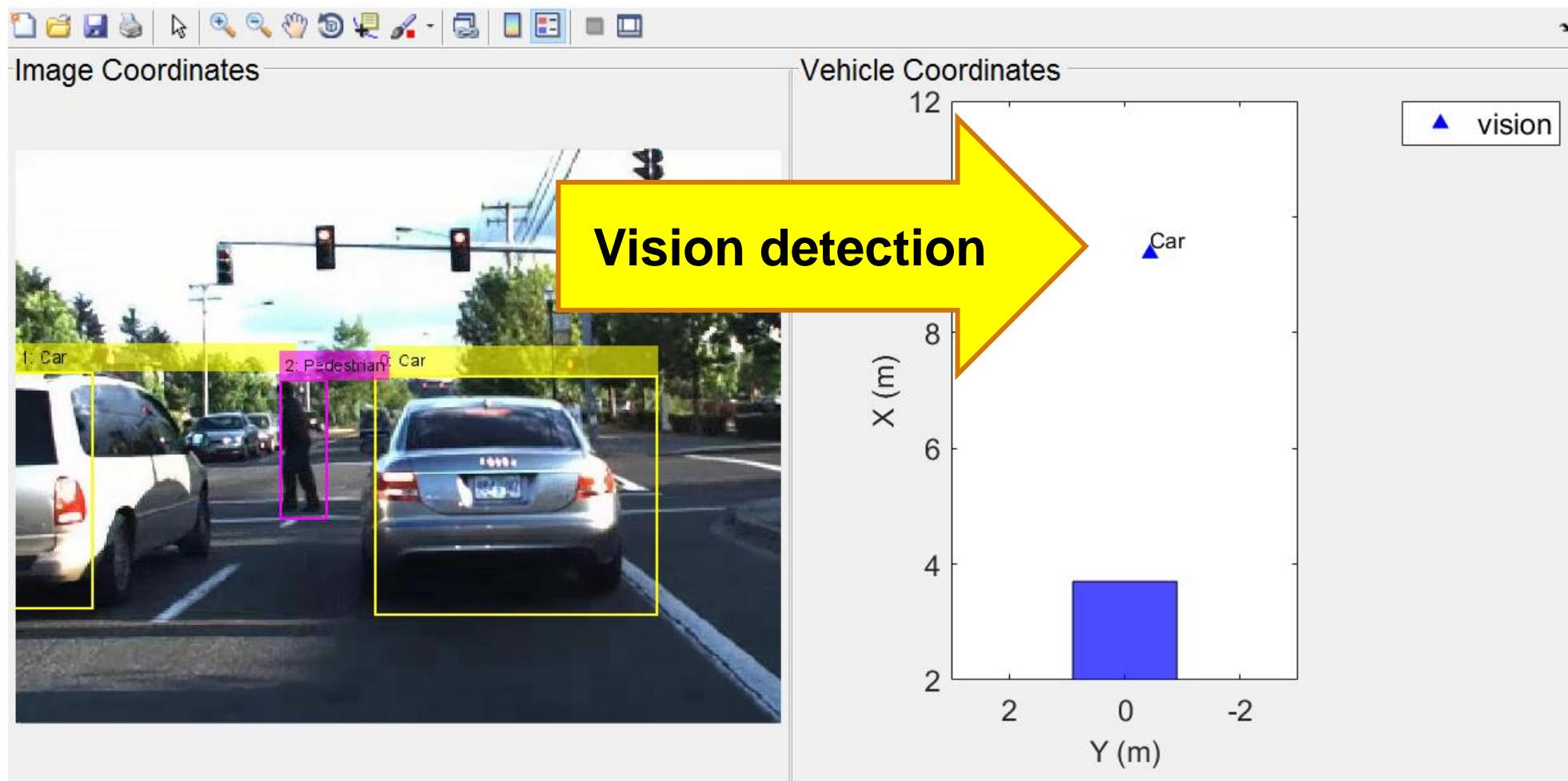


How can I
detect objects in
images?

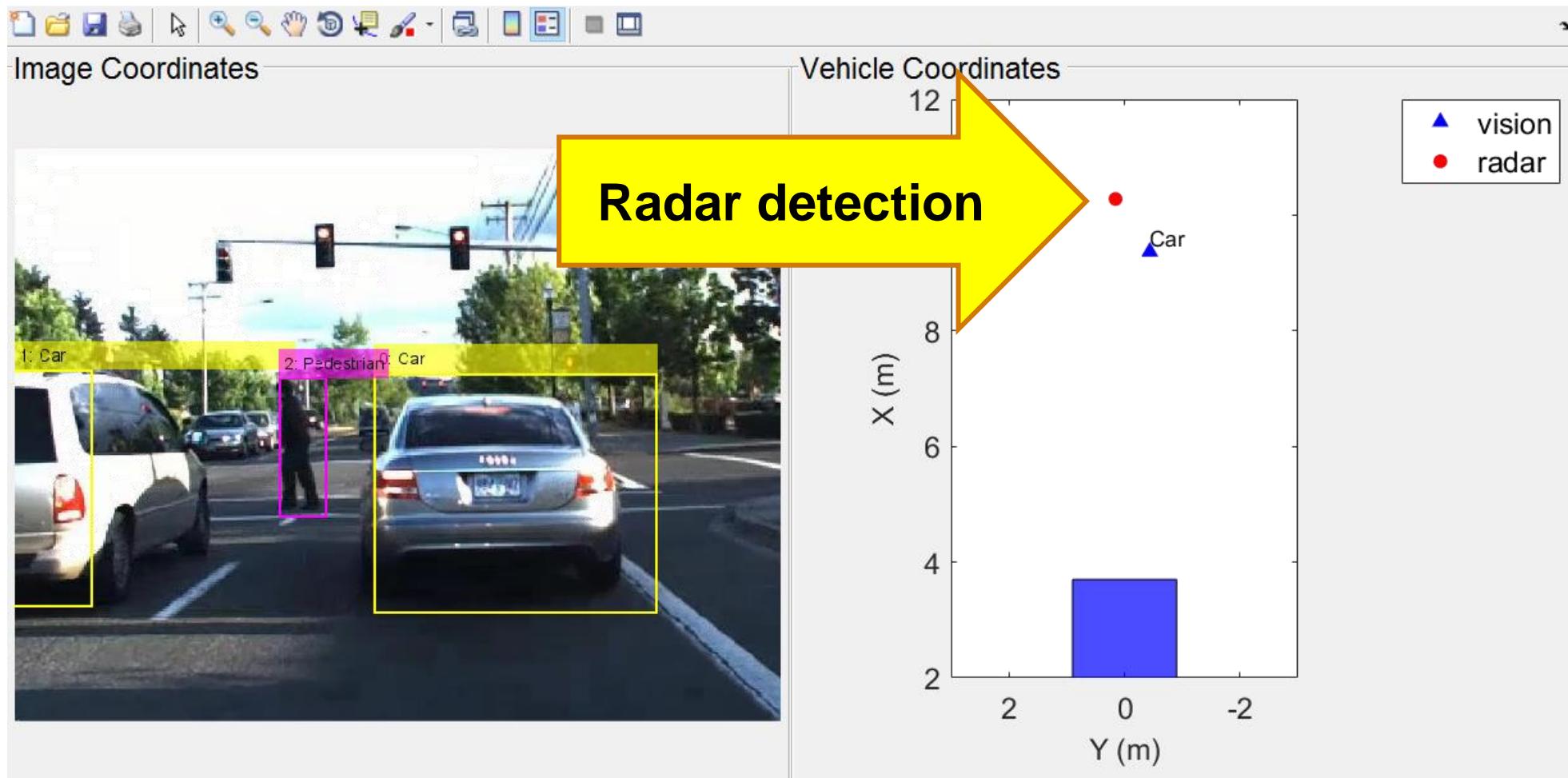


How can I
fuse multiple
detections?

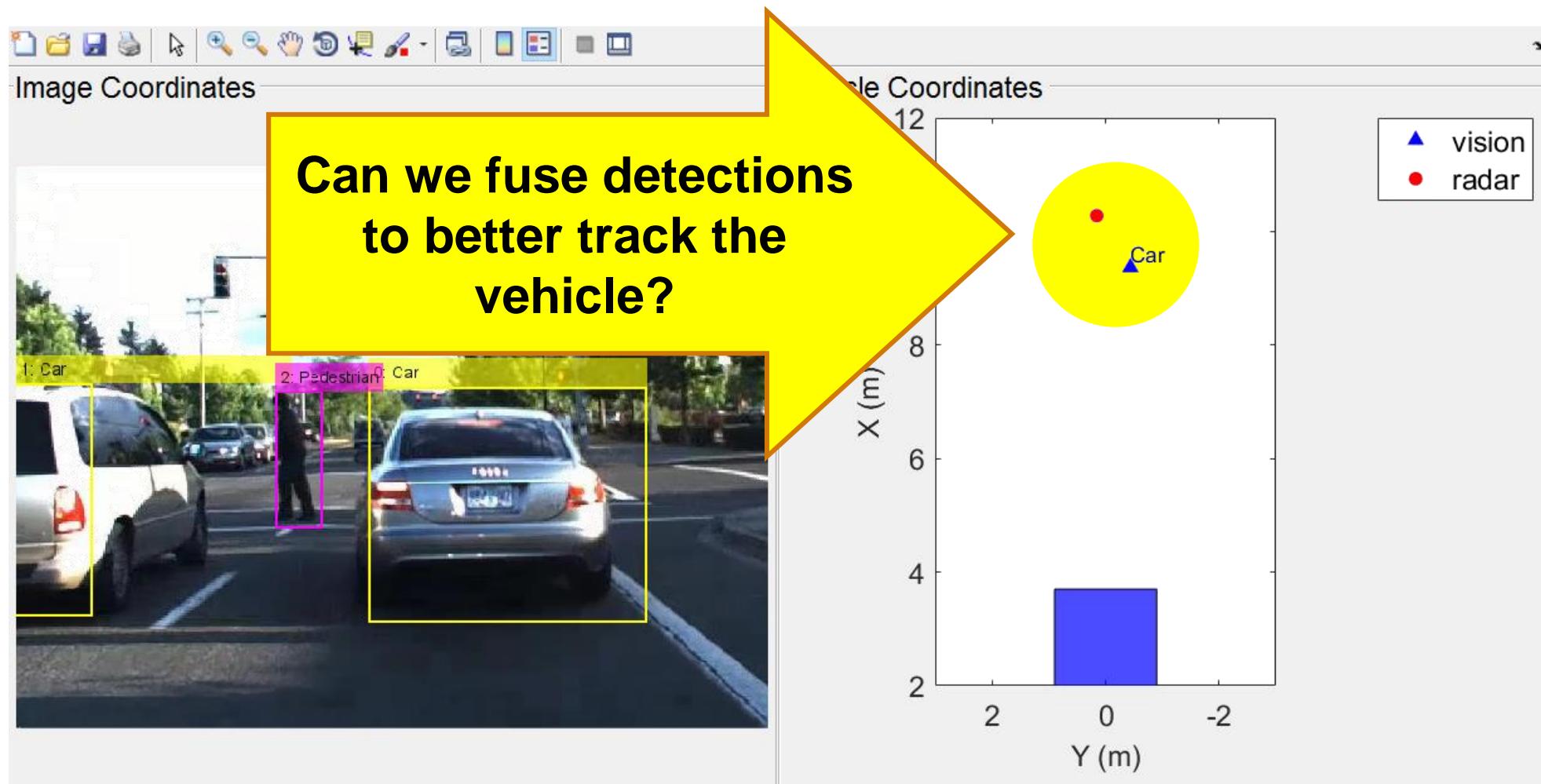
Example of radar and vision detections of a vehicle



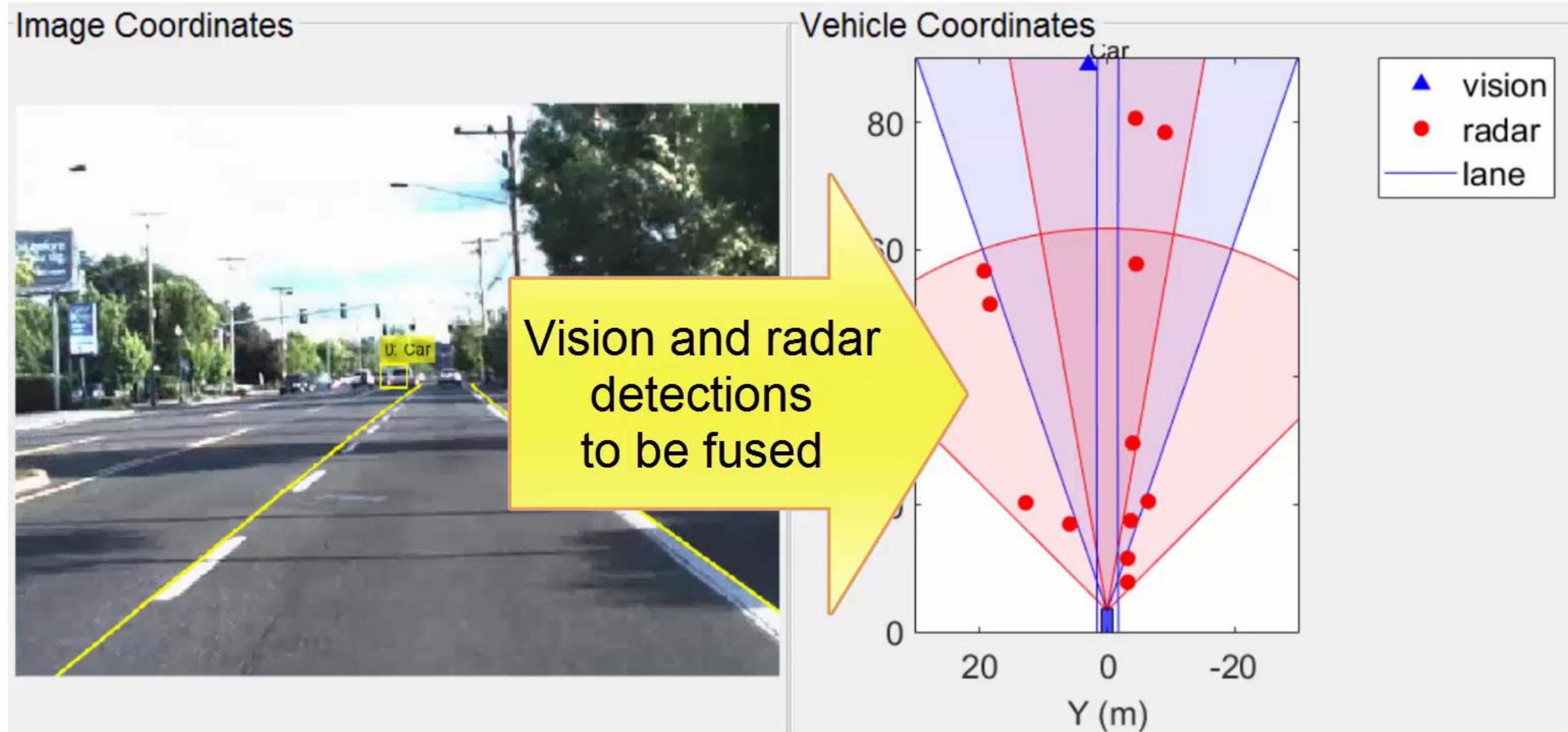
Example of radar and vision detections of a vehicle



Example of radar and vision detections of a vehicle

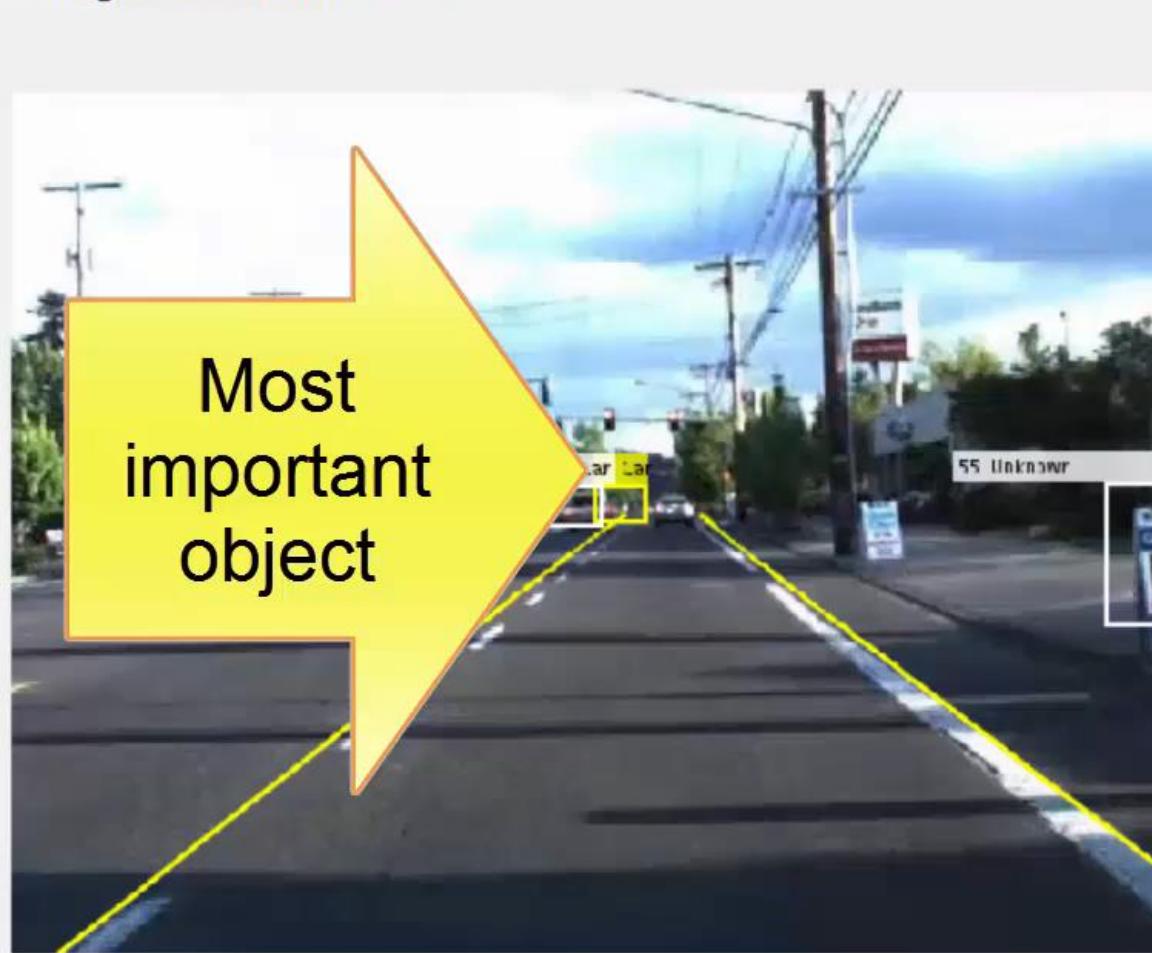


Fuse detections with multi-object tracker

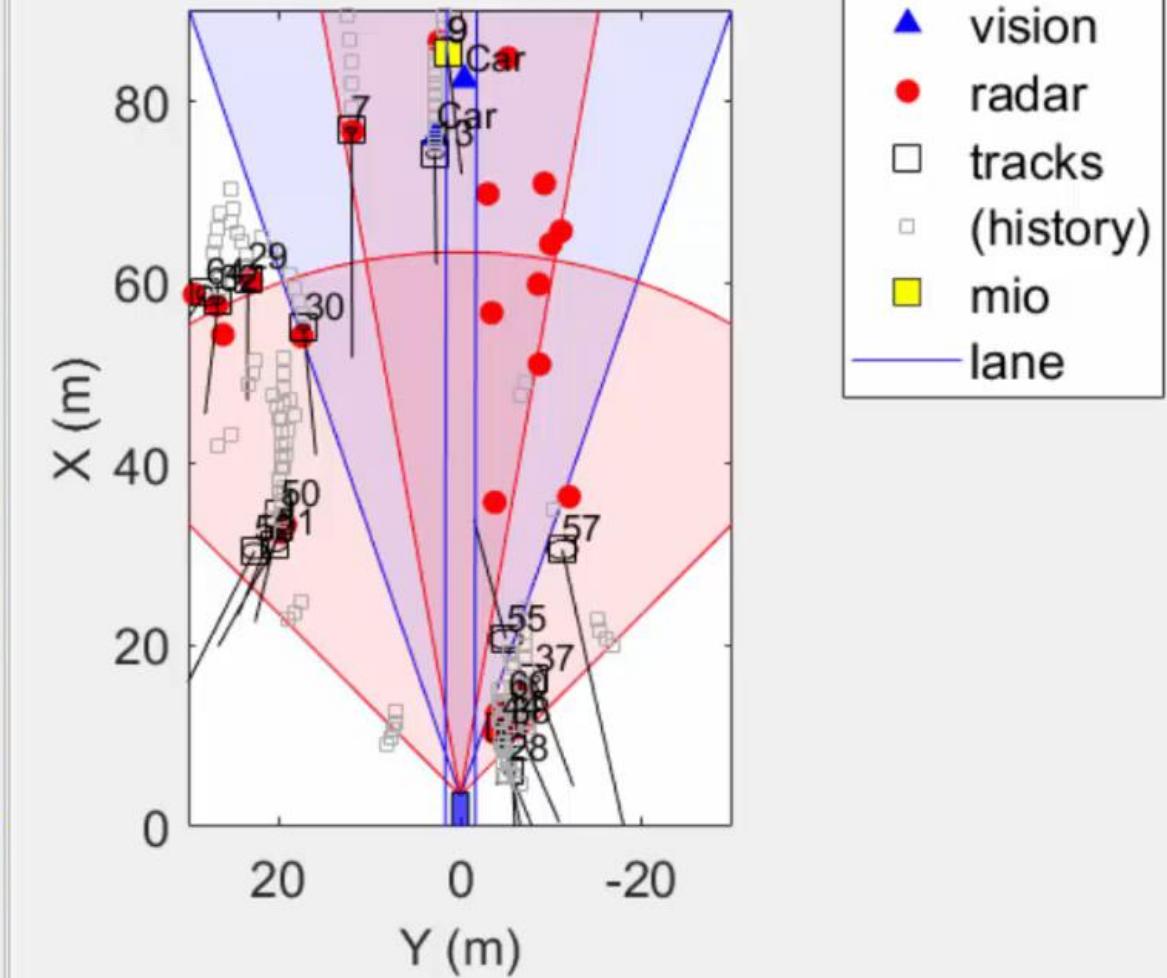


Integrate tracker into higher level algorithm

Image Coordinates



Vehicle Coordinates



Generate C code for algorithm

```
trackingForFCW.m
```

```
1 function [confirmedTracks, egoLane, numTracks, mostImportantObject] = ...
2     trackingForFCW(visionObjects, radarObjects, inertialMeasurementUnit, ...
3         laneReports, egoLane, time, positionSelector, velocitySelector)
4 % trackingForFCW - a function that contains all the tracking related
5 % functions for forward collision warning. It takes in visionObjects
6 % containing 10 vision objects.
7 % radarObjects - a struct containing 36 radar objects.
8 % inertialMeasurementUnit - a struct with speed and yaw rate of ego car.
```

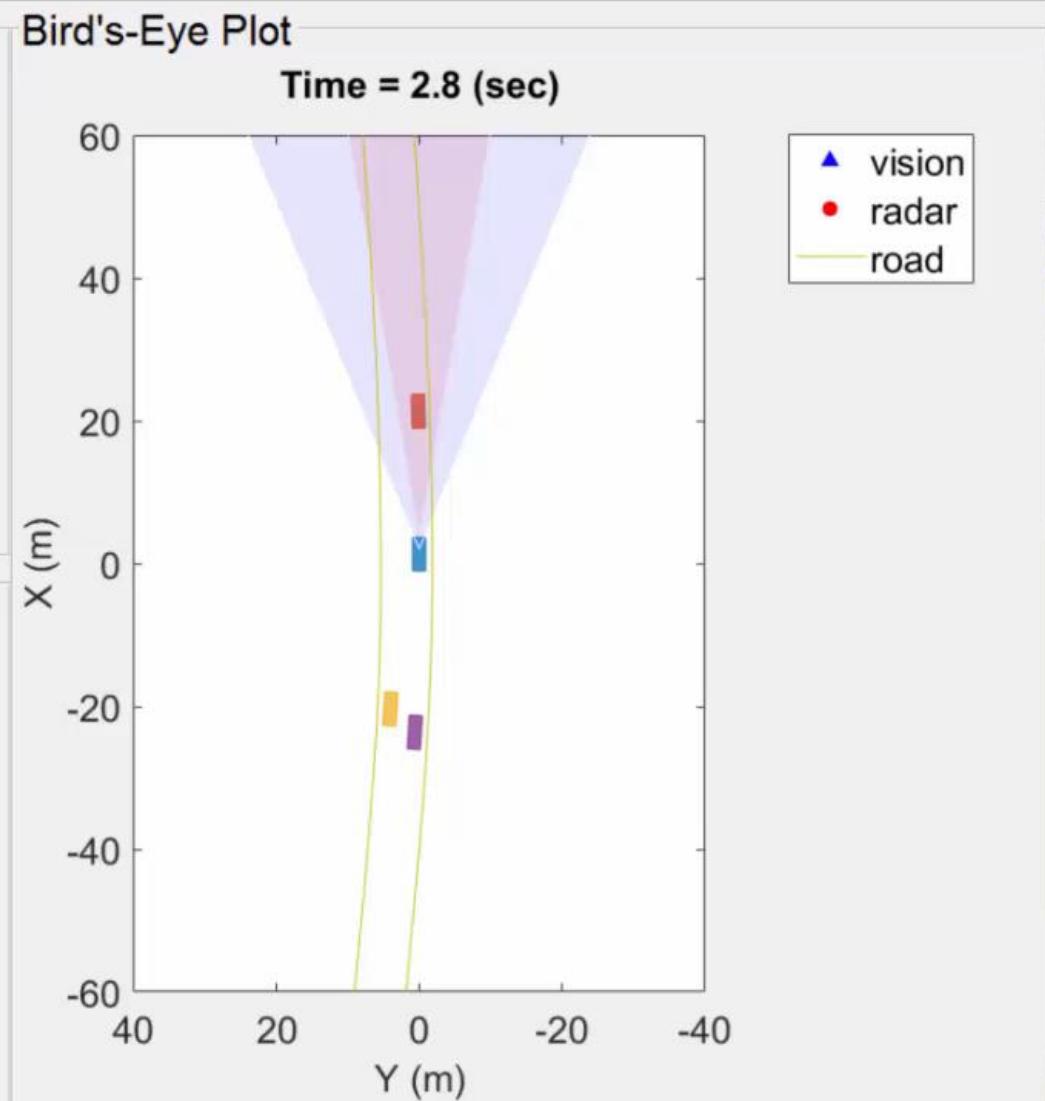
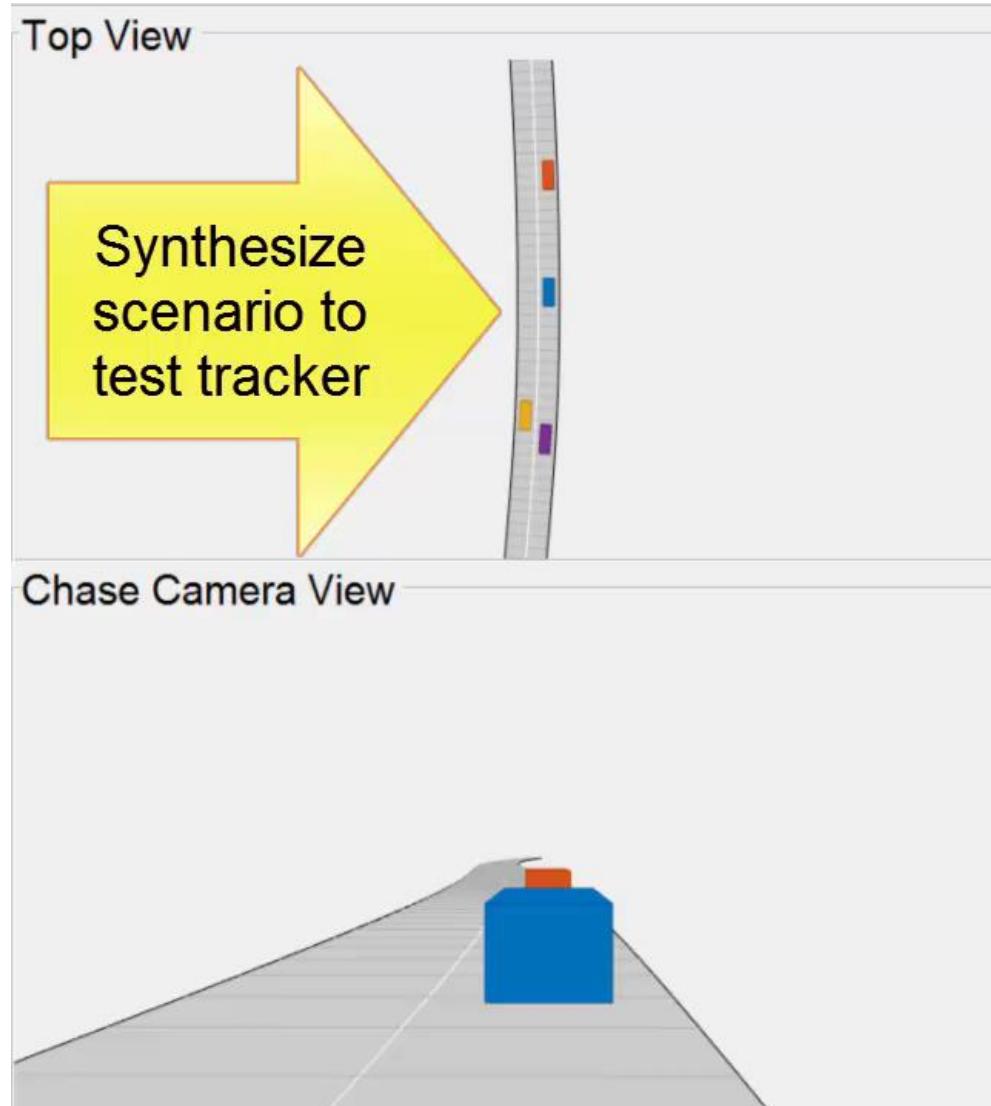
MATLAB function

function [confirmedTracks, egoLane, numTracks, mostImportantObject] = ...
trackingForFCW(visionObjects, radarObjects, inertialMeasurementUnit, ...
laneReports, egoLane, time, positionSelector, velocitySelector)
% trackingForFCW - a function that contains all the tracking related
% functions for forward collision warning. It takes in visionObjects
% containing 10 vision objects.
% radarObjects - a struct containing 36 radar objects.
% inertialMeasurementUnit - a struct with speed and yaw rate of ego car.

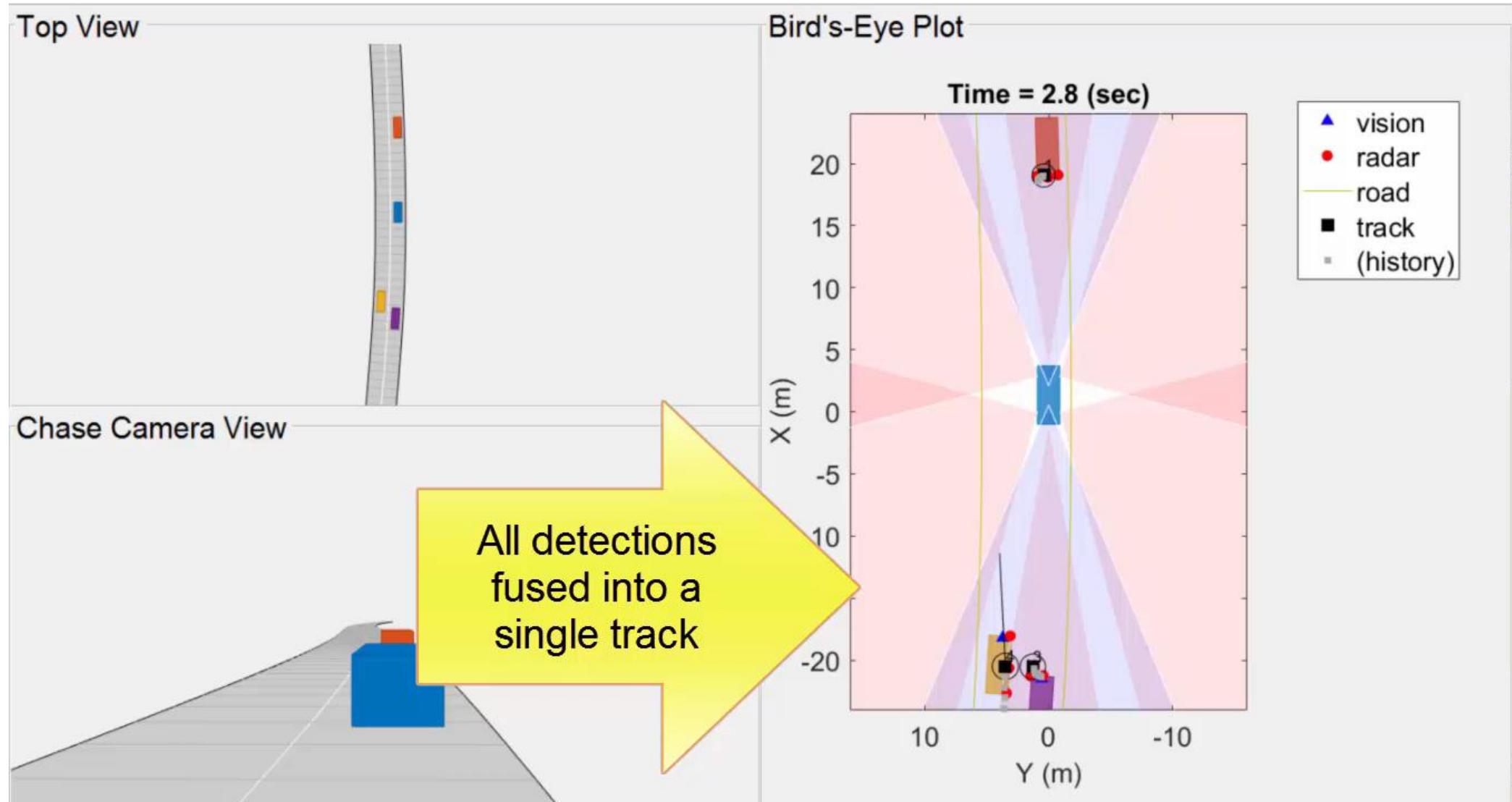
Command Window

New to MATLAB? See resources for [Getting Started](#).

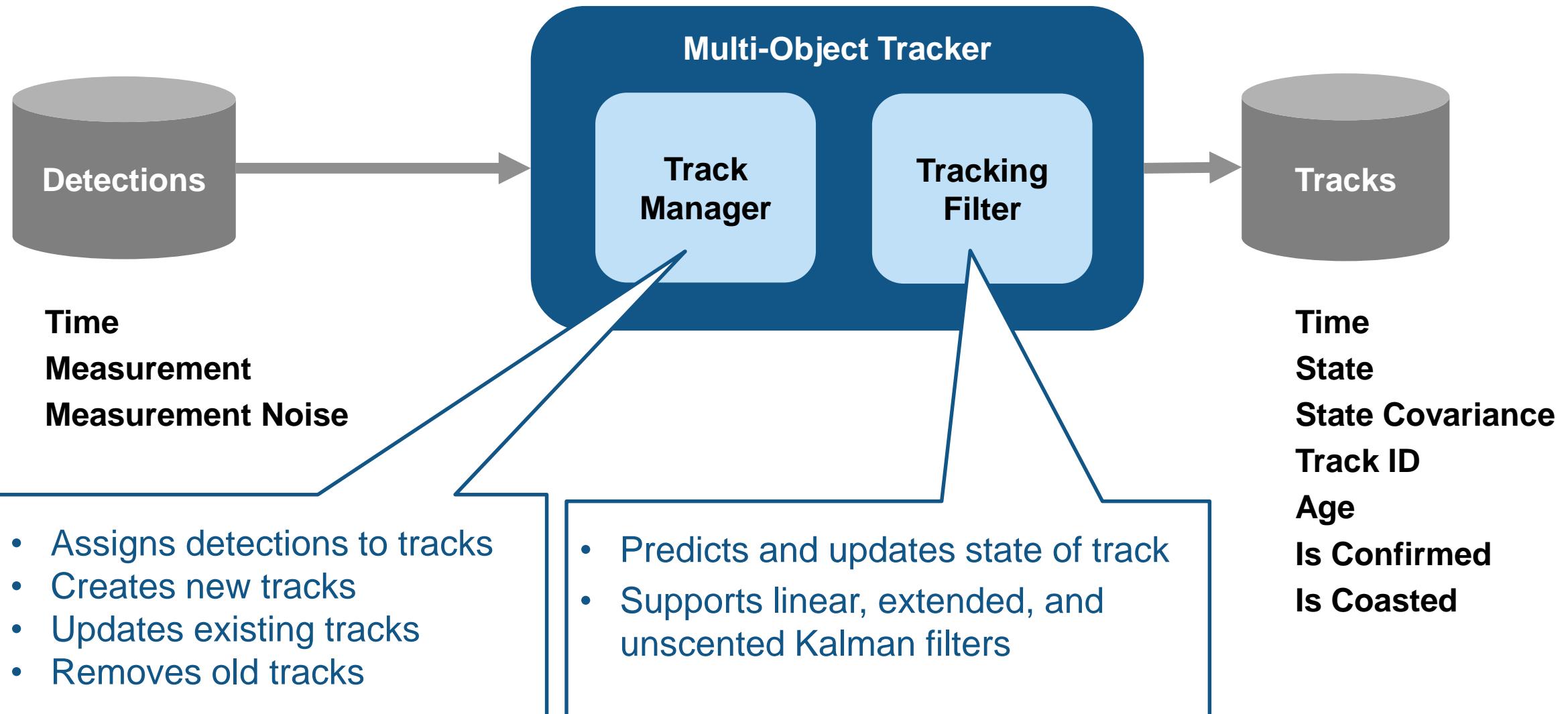
Synthesize scenario to test tracker



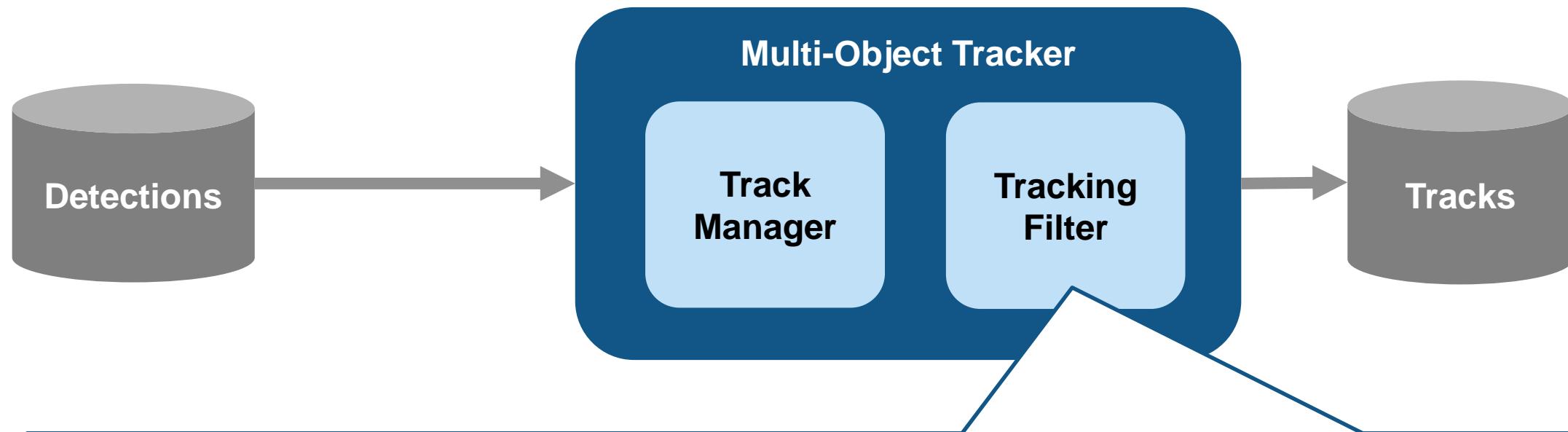
Test tracker against synthesized data



Track multiple object detections

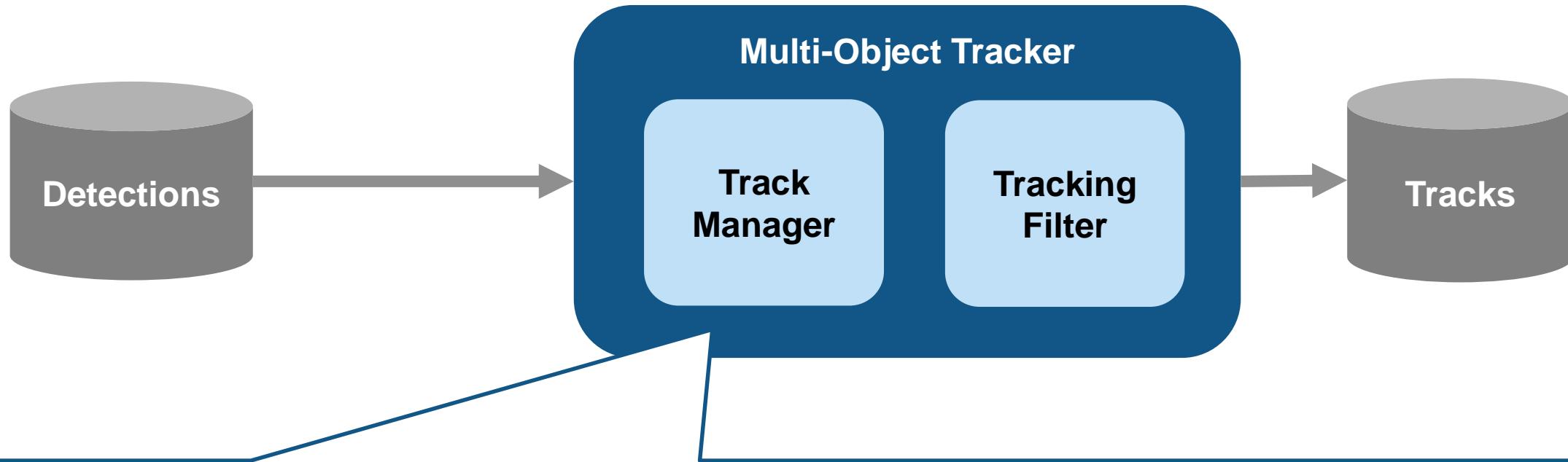


Examples of Kalman Filter (KF) initialization functions



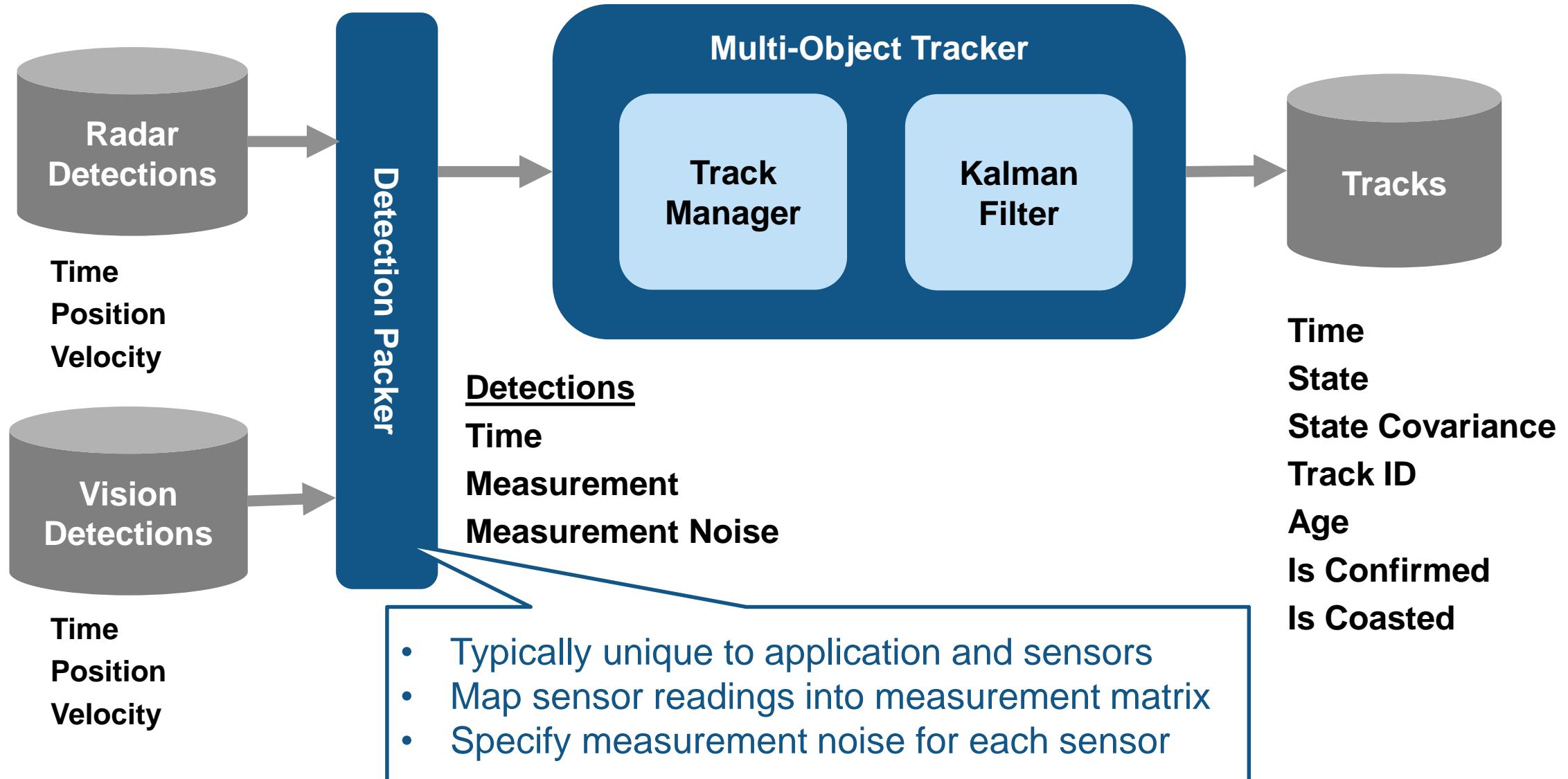
	Linear KF (trackingKF)	Extended KF (trackingEKF)	Unscented KF (trackingUKF)
Constant velocity	initcvkf	initcvekf	initcvukf
Constant acceleration	initcakf	initcaekf	initcaukf
Constant turn	Not applicable	initctekf	initctukf

Example of configuring multi-object tracker

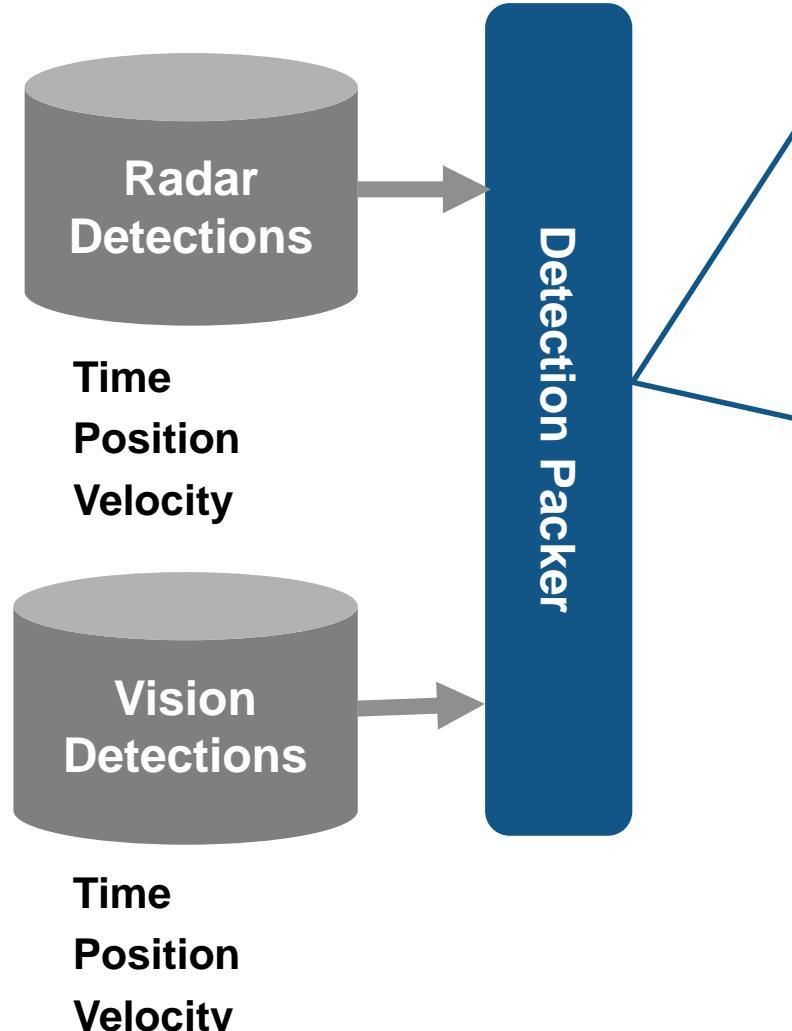


```
tracker = multiObjectTracker(...  
    'FilterInitializationFcn', @initcaekf, ... % Handle to tracking Kalman filter  
    'AssignmentThreshold', 35, ... % Normalized distance from track for assignment  
    'ConfirmationParameters', [2 ... % Min number of assignments for confirmation  
                             3], ... % Min number of updates for confirmation  
    'NumCoastingUpdates', 5); % Threshold for track deletion
```

Fuse and track multiple detections from different sensors



Example of packing sensor data into detection measurements



```
% Example sensor data
```

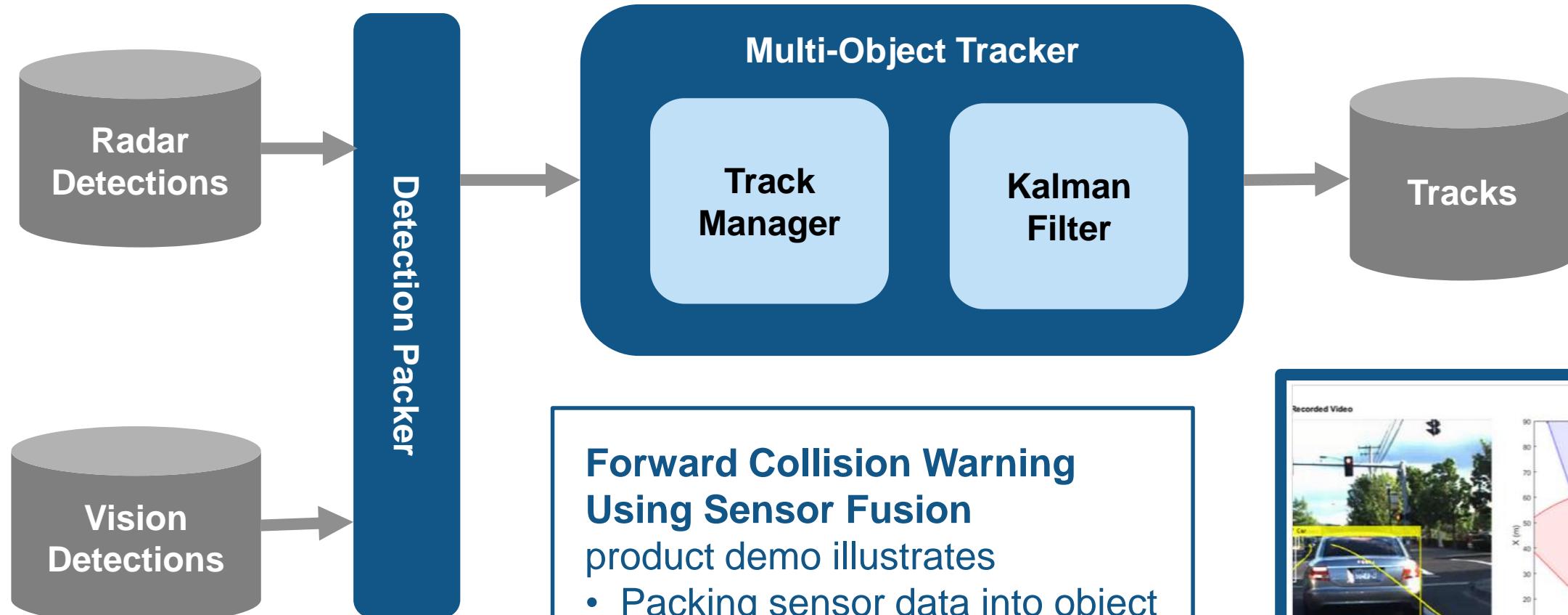
```
radar.Time      = 0.049;          % (sec)
radar.Position  = [10 0.5];       % [x y] (m)
radar.Velocity  = [4.4 0.1];      % [x y] (m/sec)
vision.Time     = 0.051;          % (sec)
vision.Position = [11 0.1];       % [x y] (m)
vision.Velocity = 4.1;           % [x] (m/sec)
```

```
% Pack to constant acceleration measurement format:
% [positionX; velocityX; positionY; velocityY]
```

```
packedDetections(1) = objectDetection(radar.Time, ...
    [radar.Position(1); radar.Velocity(1); ...
    radar.Position(2); radar.Velocity(2)], ...
    'MeasurementNoise', diag([1,1,2,10]));
```

```
packedDetections(2) = objectDetection(vision.Time, ...
    [vision.Position(1); vision.Velocity(1); ...
    vision.Position(2); 0], ...
    'MeasurementNoise', diag([2,1,1,10]));
```

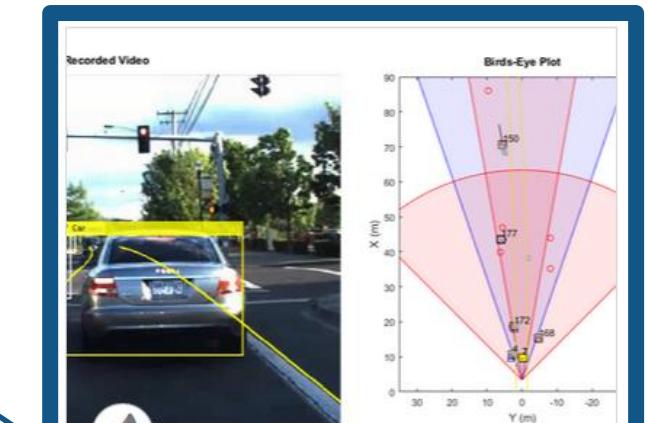
Explore demo to learn more about fusing detections



Forward Collision Warning Using Sensor Fusion

product demo illustrates

- Packing sensor data into object detections
- Initializing Kalman filter
- Configuring multi-object tracker



Forward Collision Warning Using Sensor Fusion

Generate C code for algorithm

with MATLAB Coder

```
trackingForFCW_kernel.m × +  
1 function [confirmedTracks, egoLane, numTracks, mostImportantObject] = ...  
2     trackingForFCW_kernel(visionObjects, radarObjects, inertialMeasurementUnit, ...  
3     laneReports, egoLane, time, positionSelector, velocitySelector)
```

Generate C code with
codegen

File: [trackingForFCW_kernel.c](#)

```
1629 */  
1630 void trackingForFCW_kernel(const struct0_T *visionObjects, const struct2_T  
1631 *radarObjects, const struct4_T *inertialMeasurementUnit, const struct5_T  
1632 *laneReports, struct7_T *egoLane, double time, const double positionSelector  
1633 [12], const double velocitySelector[12], emxArray_struct8_T *confirmedTracks,  
1634 double *numTracks, struct10_T *mostImportantObject)
```

Generate C-code for algorithm with MATLAB Coder

```
%% Create variables that will be used to specify example input arguments
[visionObjects, radarObjects, imu, lanes] = ...
    helperReadSensorRecordingsFile('01_city_c2s_fcw_10s_sensor.mat');
egoLane = struct('left', [0 0 0], 'right', [0 0 0]);
time = 0;
positionSelector = [1 0 0 0 0 0; 0 0 0 1 0 0];
velocitySelector = [0 1 0 0 0 0; 0 0 0 0 1 0];

exampleInputs = {visionObjects(1), radarObjects(1), imu(1), ...
    lanes(1), egoLane, time, ...
    positionSelector, velocitySelector};

%% Generate code for sensor fusion algorithm: trackingForFCW_kernel
codegen trackingForFCW_kernel -args exampleInputs -config:dll -launchreport
```

Install patch to generate C code from multiObjectTracker

- <https://www.mathworks.com/support/bugreports/1546972>

 **1546972**

 **Summary**

Code generation fails for multiObjectTracker in 'lib' or 'dll' configuration

Description

Code generation of a function that uses multiObjectTracker fails under some conditions with the following error message:

```
??? Property 'pSampleDetection.Measurement' is undefined on  
some execution paths but is used inside the called  
function.
```

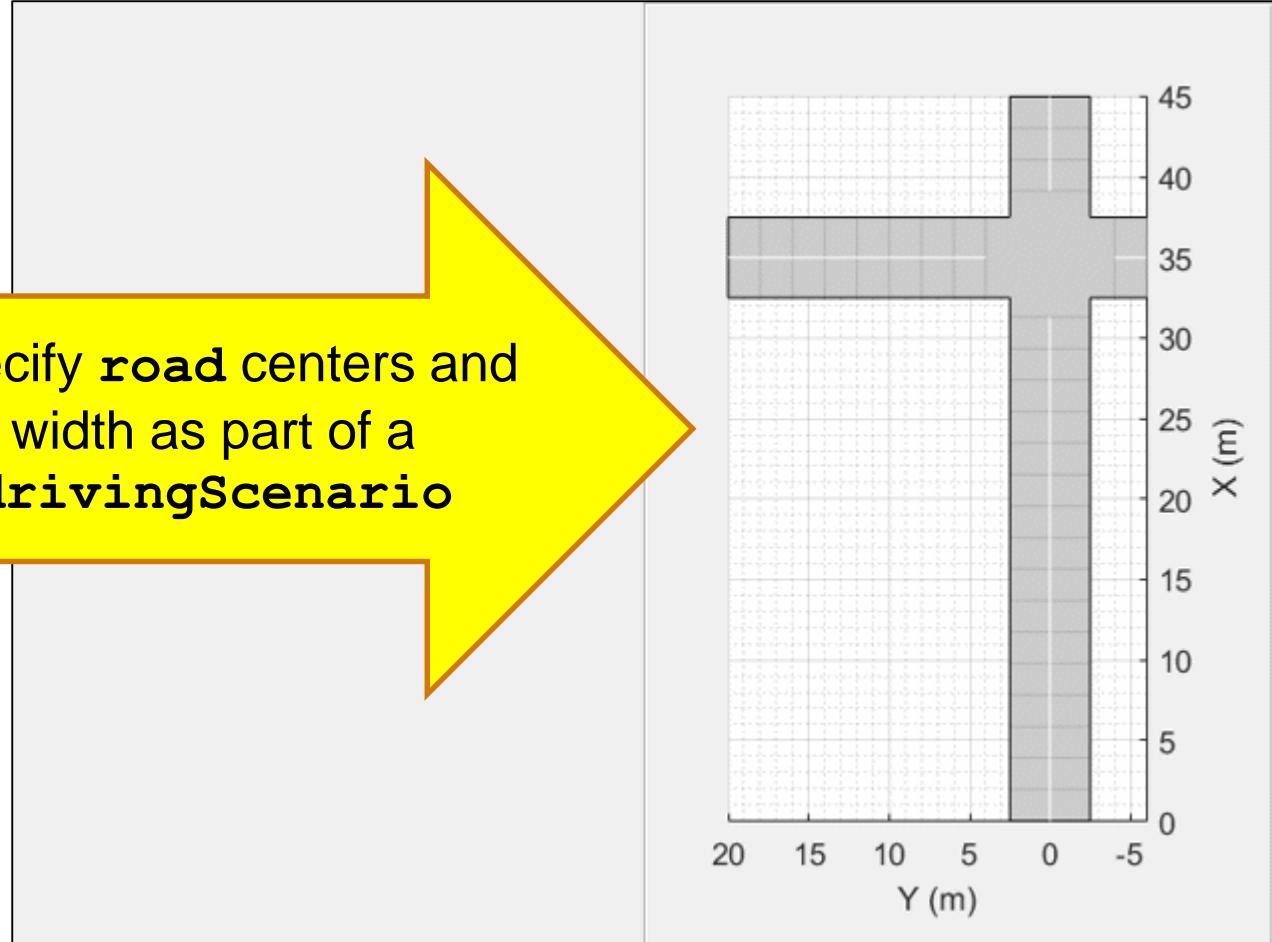
Workaround

Installation instructions

Specify driving scenario and roads

```
%% Create a new scenario  
  
s = drivingScenario('SampleTime', 0.05);  
  
%% Create road  
  
road(s, [ 0 0; ... % Centers [x,y] (m)  
          45 0], ...  
          5); % Width (m)  
  
road(s, [35 20; ...  
          35 -10], ...  
          5);  
  
%% Plot scenario  
  
p1 = uipanel('Position', [0.5 0 0.5 1]);  
a1 = axes('Parent', p1);  
  
plot(s, 'Parent', a1, ...  
      'Centerline', 'on', 'Waypoints', 'on')  
a1.XLim = [0 45];  
a1.YLim = [-6 20];
```

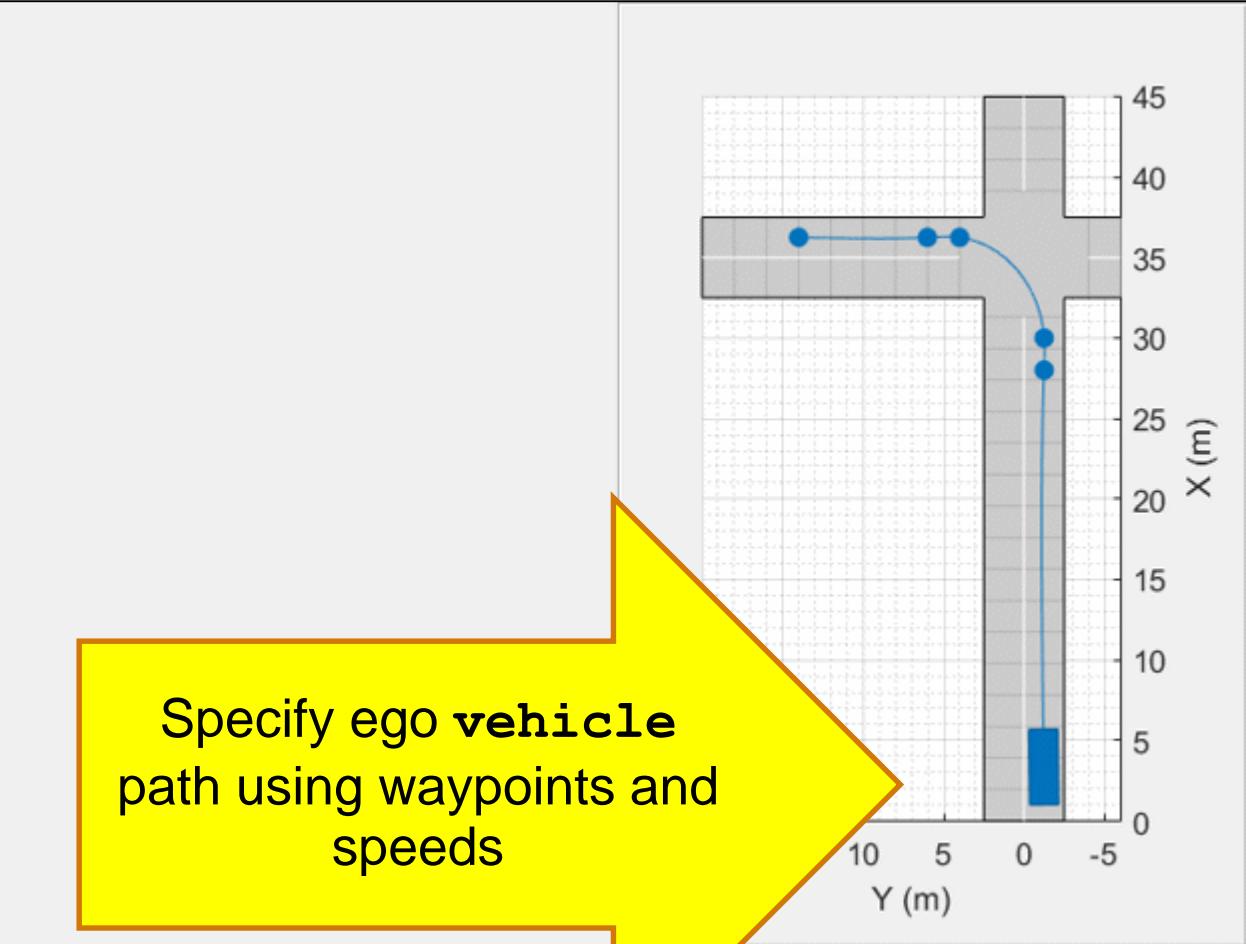
Specify **road** centers and width as part of a **drivingScenario**



Add ego vehicle

```
%% Add ego vehicle  
egoCar = vehicle(s);  
waypoints = [ 2 -1.25; ... % [x y] (m)  
            28 -1.25; ...  
            30 -1.25; ...  
            36.25 4; ...  
            36.25 6; ...  
            36.25 14];  
speed = 13.89; % (m/s) = 50 km/hr  
path(egoCar, waypoints, speed);
```

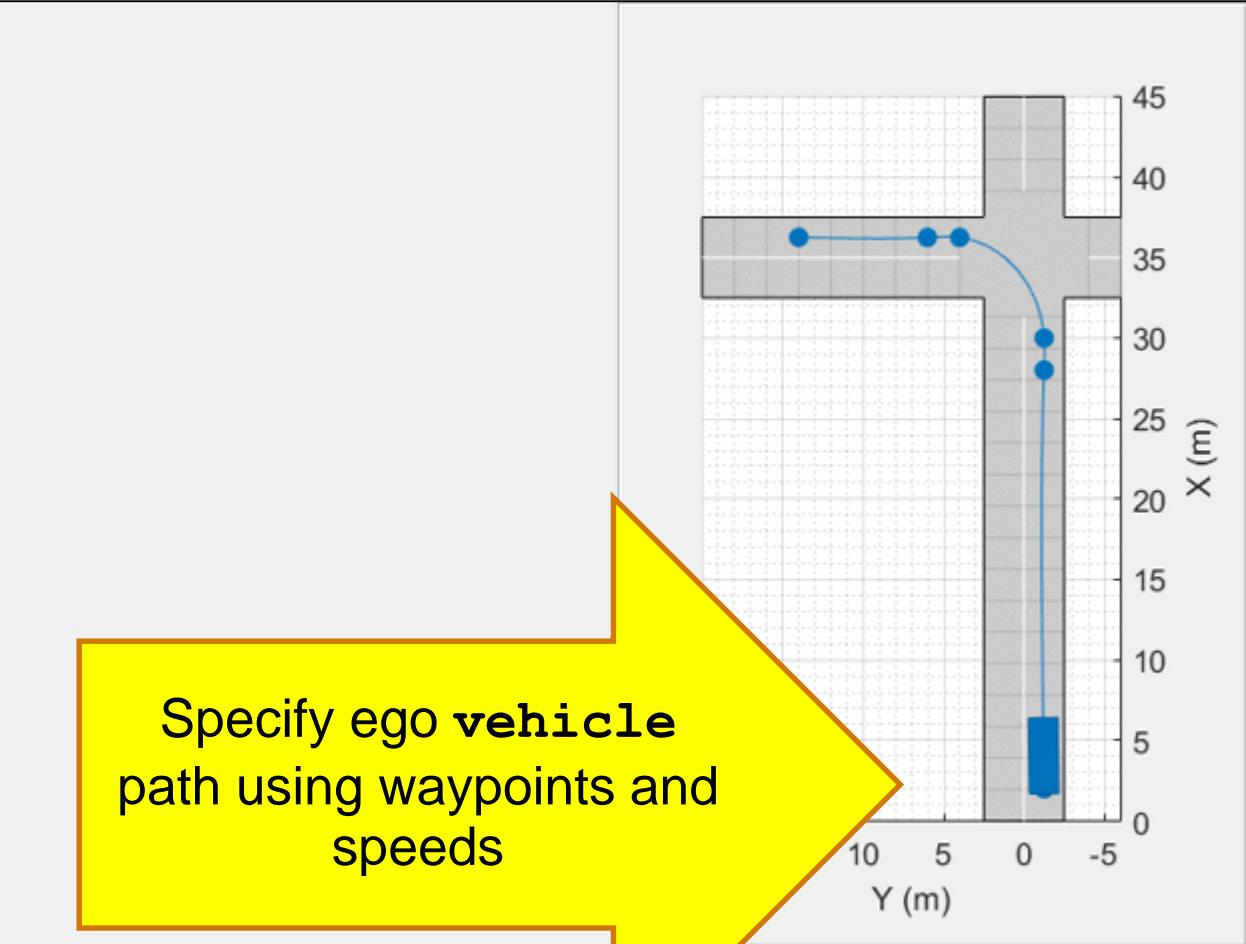
Specify ego **vehicle**
path using waypoints and
speeds



Add ego vehicle

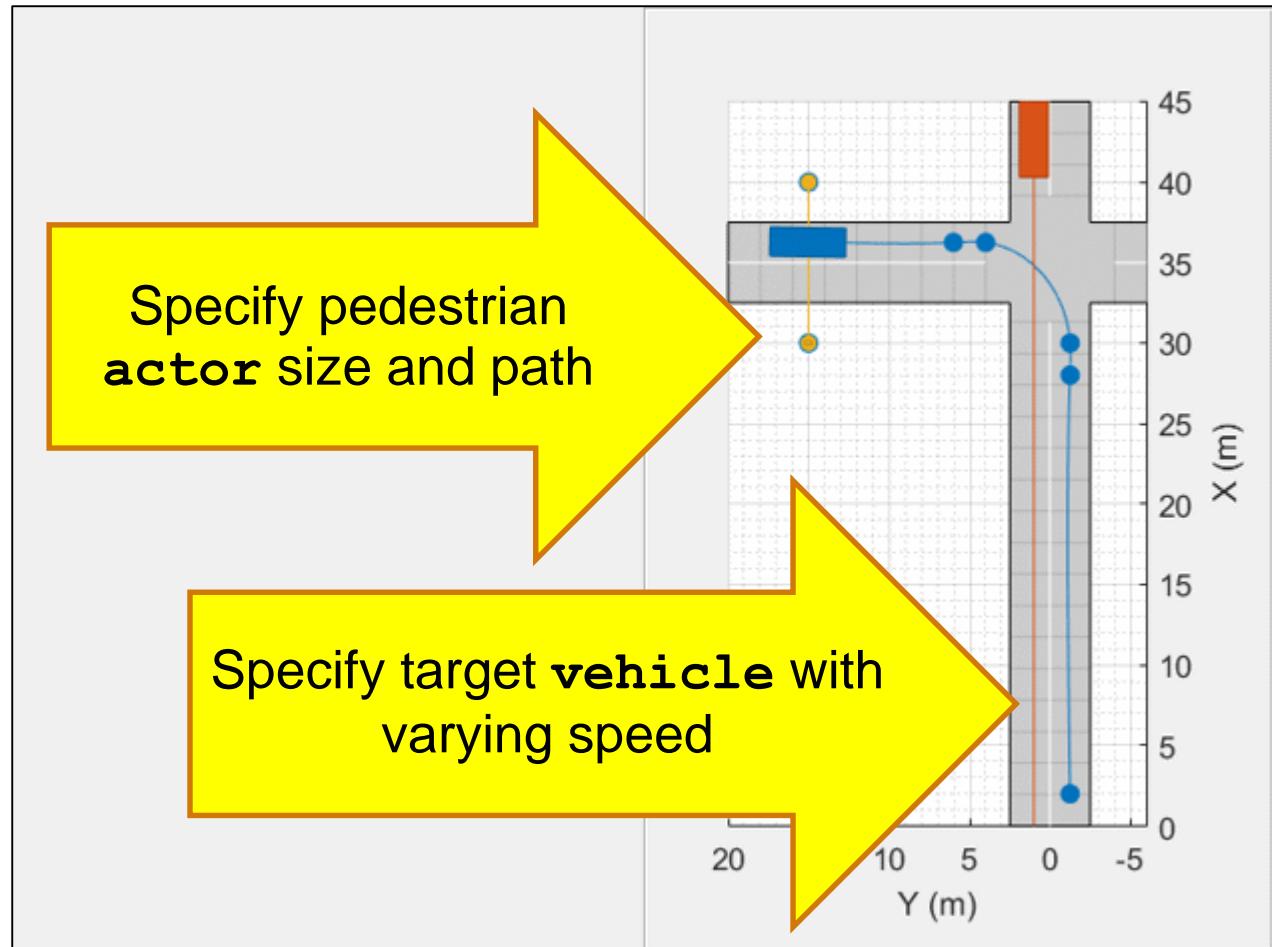
```
%% Add ego vehicle  
  
egoCar = vehicle(s);  
  
waypoints = [ 2 -1.25; ... % [x y] (m)  
             28 -1.25; ...  
             30 -1.25; ...  
             36.25 4; ...  
             36.25 6; ...  
             36.25 14];  
  
speed = 13.89; % (m/s) = 50 km/hr  
  
path(egoCar, waypoints, speed);  
  
%% Play scenario  
  
while advance(s)  
    pause(s.SampleTime);  
end
```

Specify ego **vehicle**
path using waypoints and
speeds



Add target vehicle and pedestrian actor

```
%% Add Target vehicle  
targetVehicle = vehicle(s);  
  
path(targetVehicle, ...  
    [44 1; -4 1], ... % Waypoints (m)  
    [5 ; 14]); % Speeds (m/s)  
  
%% Add child pedestrian actor  
child = actor(s, 'Length', 0.24, ...  
    'Width', 0.45, ...  
    'Height', 1.7, ...  
    'Position', [40 -5 0], ...  
    'Yaw', 180);  
  
path(child, ...  
    [30 15; 40 15], ... % Waypoints (m)  
    1.39); % Speed (m/s) = 5 km/hr
```



View scenario from behind ego vehicle

```
%% Add chase view (left)
p2 = uipanel('Position',[0 0 0.5 1]);
a2 = axes('Parent',p2);
chasePlot(egoCar,...  

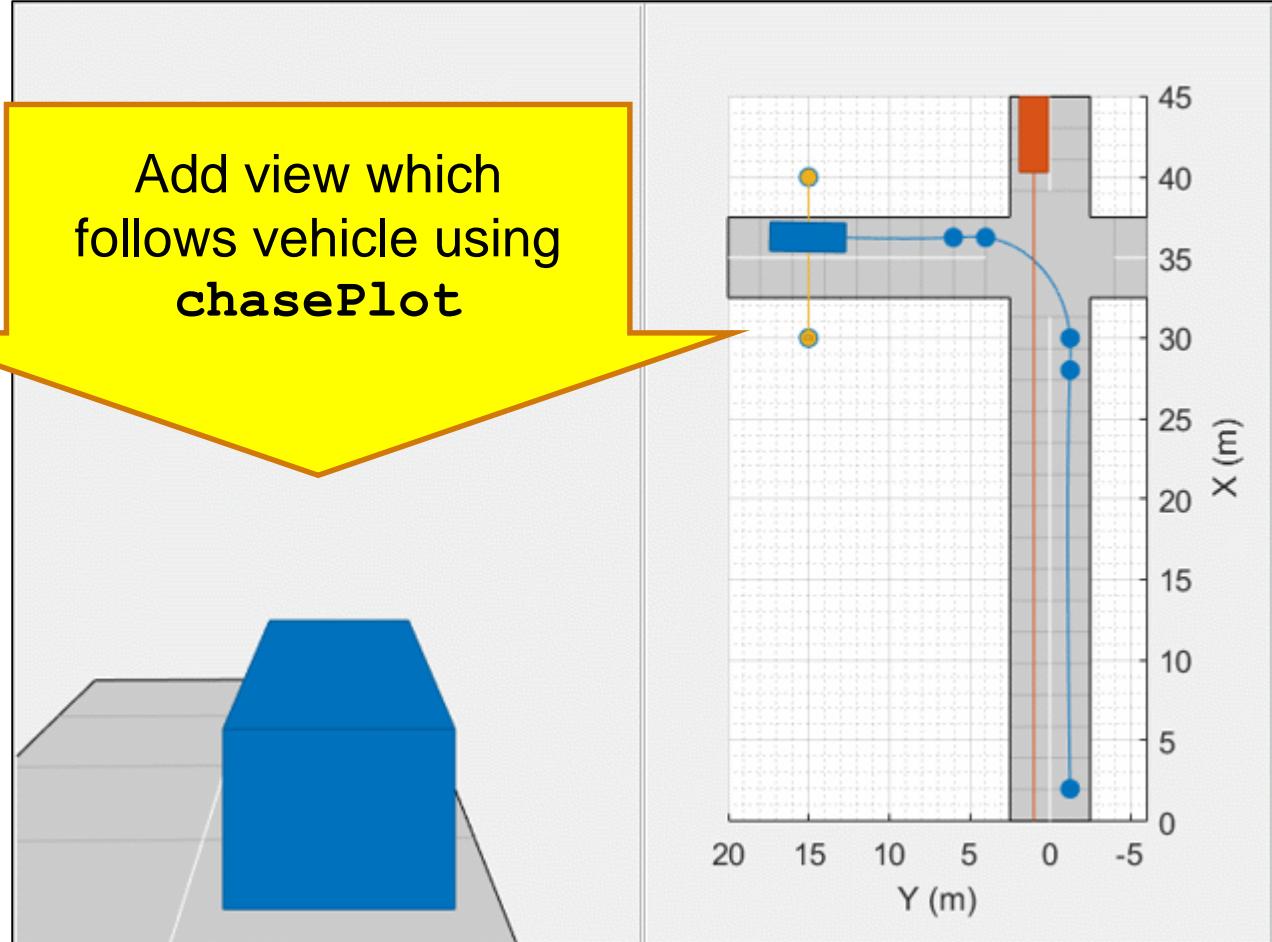
    'Parent',a2,...  

    'Centerline','on',...  

    'ViewHeight',3.5,... % (m)  

    'ViewLocation',[-8 0]); % [x y] (m)
```

Add view which
follows vehicle using
chasePlot



View scenario from behind ego vehicle

```
%% Add chase view (left)
p2 = uipanel('Position',[0 0 0.5 1]);
a2 = axes('Parent',p2);
chasePlot(egoCar,...  

    'Parent',a2,...  

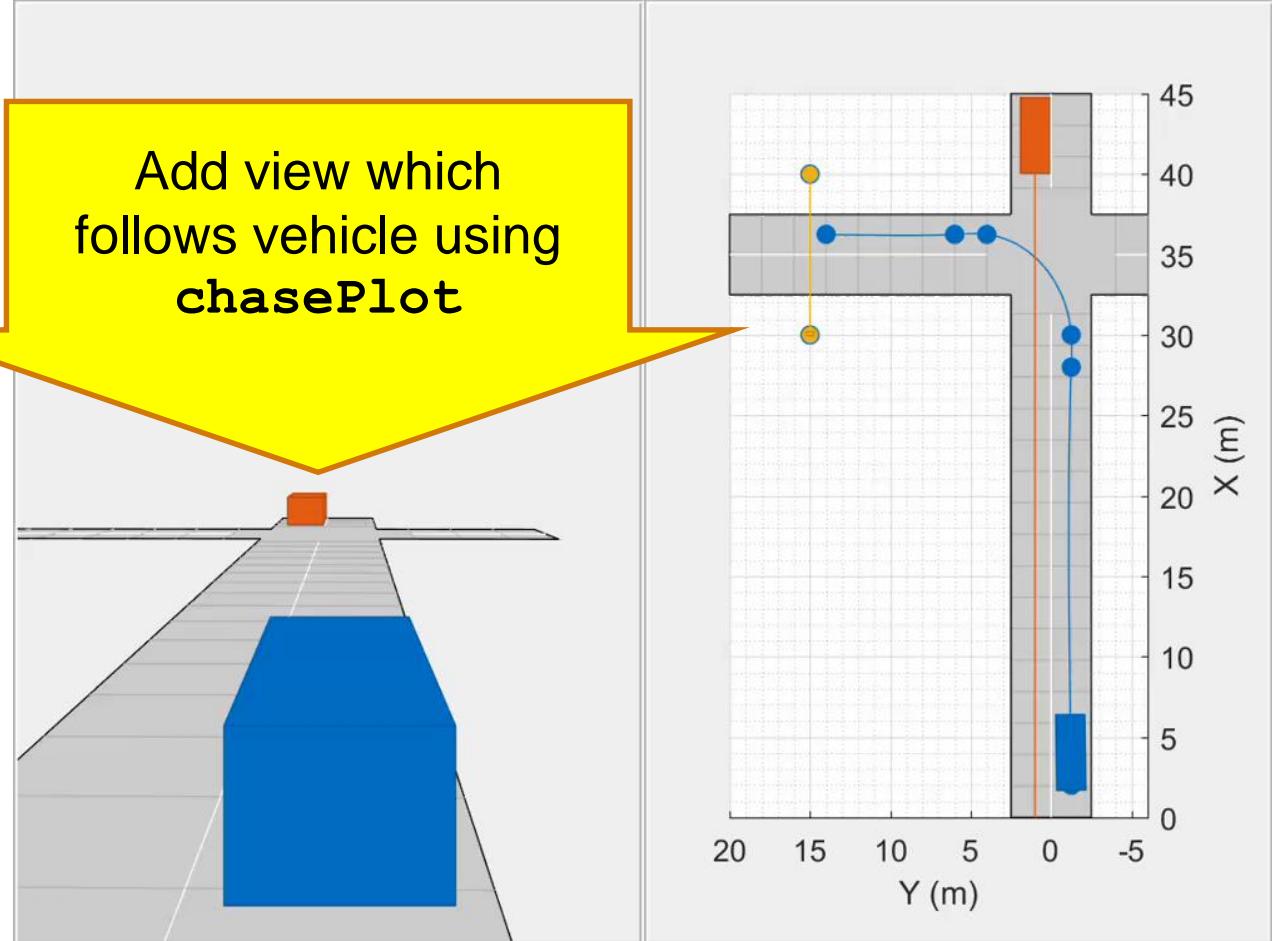
    'Centerline','on',...  

    'ViewHeight',3.5,... % (m)  

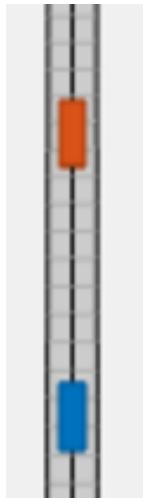
    'ViewLocation',[-8 0]); % [x y] (m)

%% Play scenario
restart(s)
while advance(s)
    pause(s.SampleTime);
end
```

Add view which
follows vehicle using
chasePlot

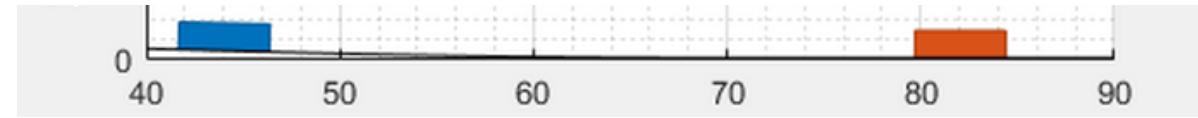


Simulate effects of vision detection sensor



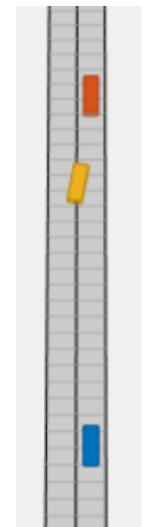
Range effects

- Range measurement accuracy degrades with distance to object
- Angle measurement accuracy consistent throughout coverage area



Road elevation effects

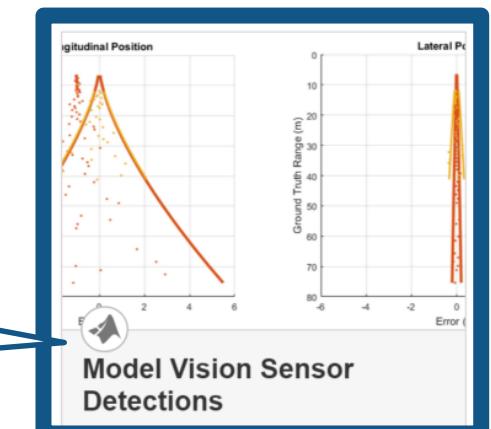
- Objects may not be detected if they appear above the horizon line
- Large range measurement errors can be introduced for detected objects at different road elevations



Occlusion effects

- Partially or completely occluded objects are not detected

Model Vision Sensor Detections
product demo illustrates these effects



Model vision detection sensor

```
%% Create vision detection generator  
sensor = visionDetectionGenerator(...  
    'SensorLocation', [0.75*egoCar.Wheelbase 0], ...  
    'Height', 1.1, ...  
    'Pitch', 1, ...  
    'Intrinsics', cameraIntrinsics(...  
        800, ... % Focal length  
        [320 240], ... % Principal point  
        [480 640]), ... % Image size  
    'RadialDistortion', [0 0], ...  
    'TangentialDistortion', [0 0]), ...  
    'UpdateInterval', s.SampleTime, ...  
    'BoundingBoxAccuracy', 5, ...  
    'MaxRange', 150, ...  
    'ActorProfiles', actorProfiles(s)));
```

Extrinsic mounting parameters

Coverage area is determined based
on **cameraIntrinsics**

Model radar detection
sensor using
radarDetectionGenerator

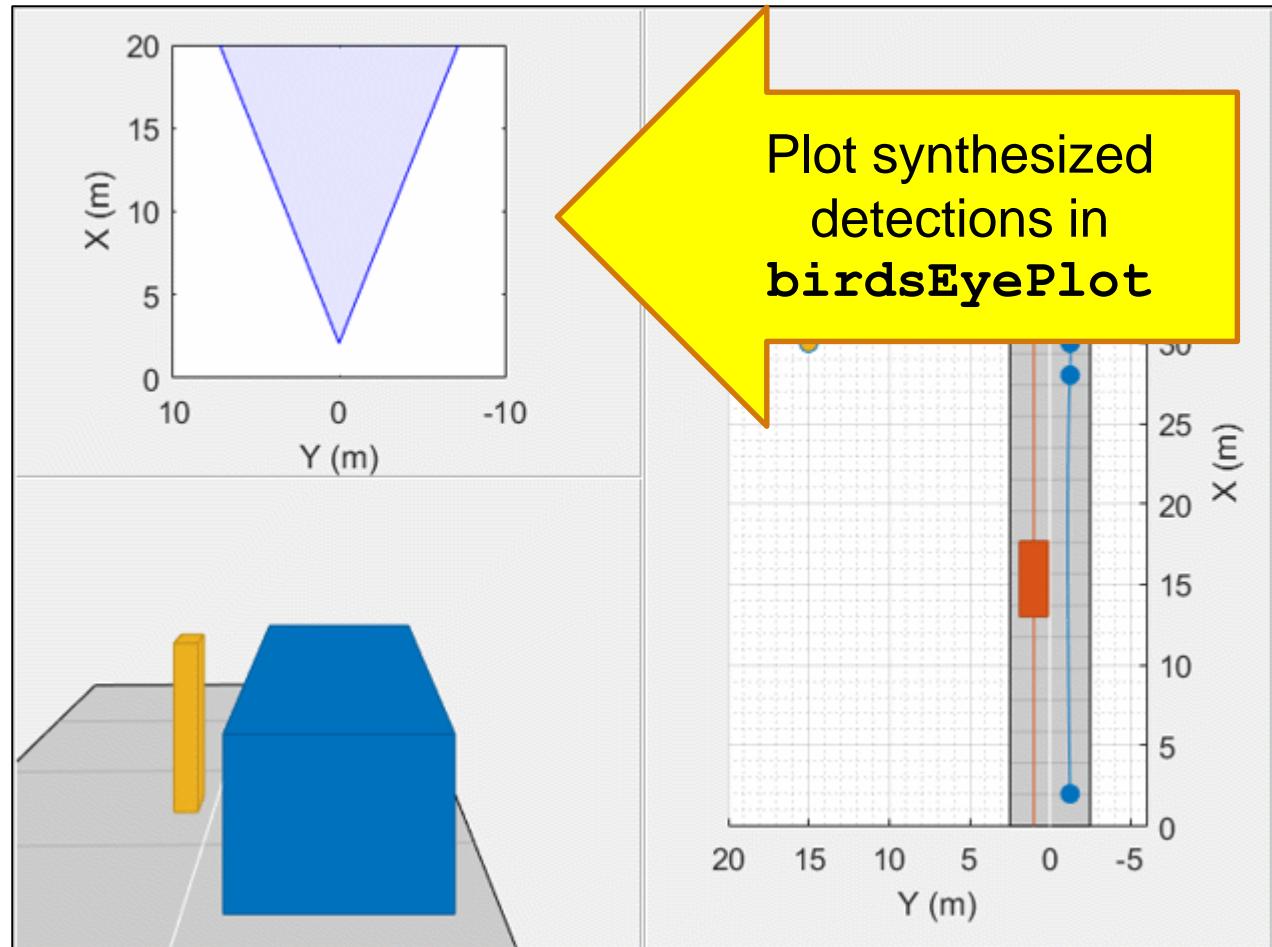
Create birds eye plot to view sensor detections

```
%% Add sensor birds eye plot (top left)
p3 = uipanel('Position',[0 0.5 0.5 0.5]);
a3 = axes('Parent',p3);
bep = birdsEyePlot('Parent',a3, ...
    'Xlimits', [0 20], ...
    'Ylimits', [-10 10]);
legend(a3, 'off');

% Create plotters
covPlot = coverageAreaPlotter(bep, ...
    'FaceColor','blue',...
    'EdgeColor','blue');
plotCoverageArea(covPlot, ...
    sensor.SensorLocation,sensor.MaxRange, ...
    sensor.Yaw,sensor.FieldOfView(1))

detPlot = detectionPlotter(bep, ...
    'MarkerEdgeColor','blue',...
    'Marker','^');

truthPlot = outlinePlotter(bep);
```

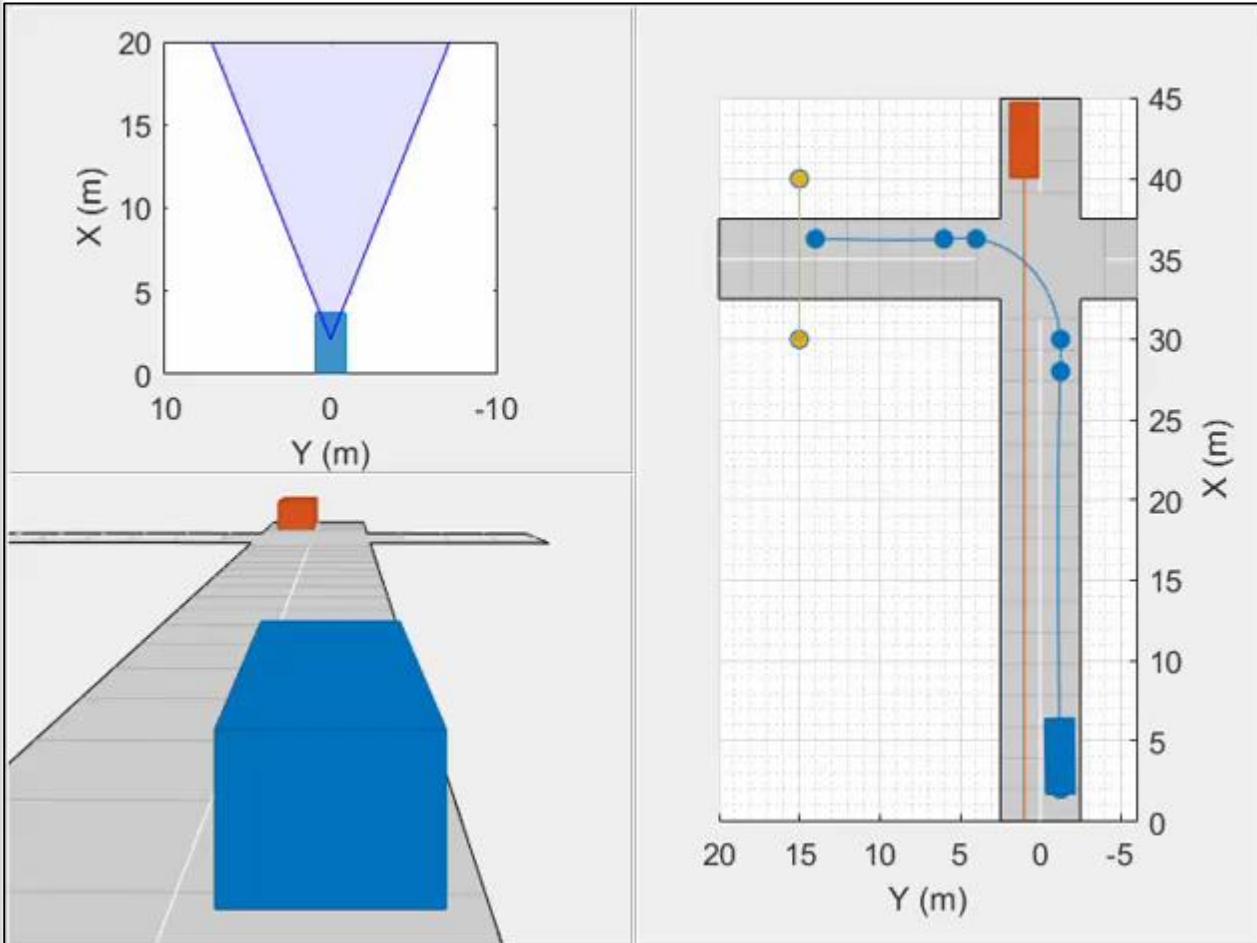


Play scenario with sensor models

```

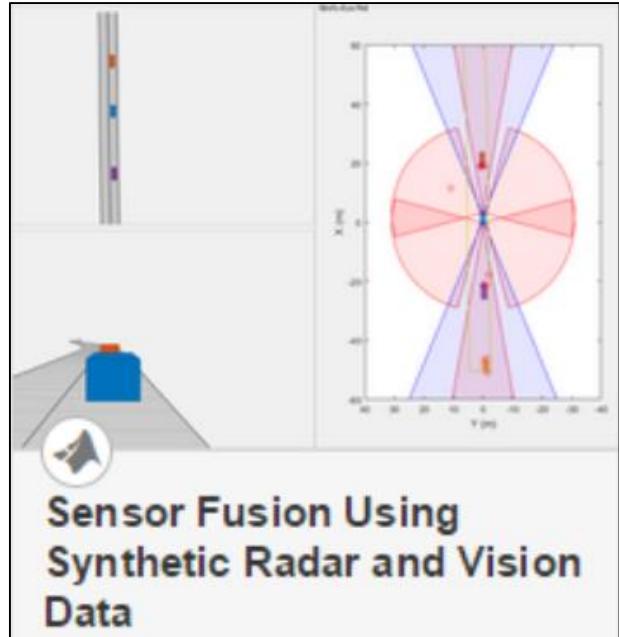
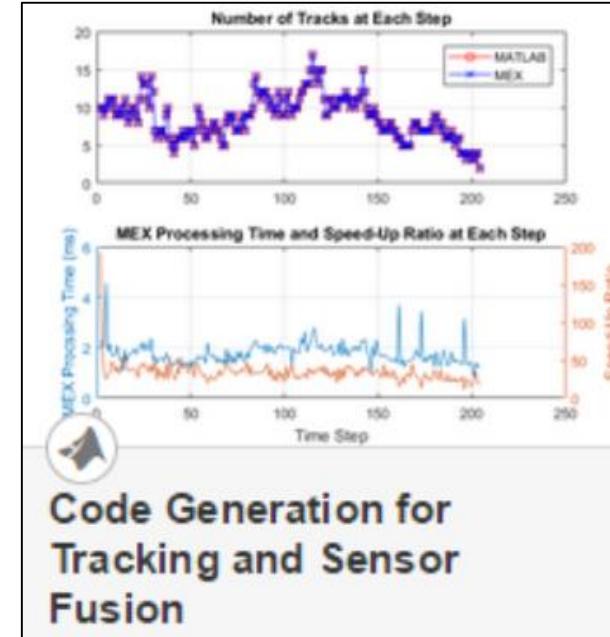
restart(s)
while advance(s)
    % Get detections in ego vehicle coordinates
    det = sensor(targetPoses(egoCar), ...
                 s.SimulationTime);
    % Update plotters
    if isempty(det)
        clearData(detPlot)
    else % Unpack measurements to position/velocity
        pos = cellfun(@(d)d.Measurement(1:2), ...
                      det, 'UniformOutput',false);
        vel = cellfun(@(d)d.Measurement(4:5), ...
                      det, 'UniformOutput',false);
        plotDetection(detPlot, ...
                        cell2mat(pos)'), cell2mat(vel)');
    end
    [p, y, l, w, oo, c] = targetOutlines(egoCar);
    plotOutline(truthPlot,p,y,l,w, ...
                 'OriginOffset', oo, 'Color', c);
end

```



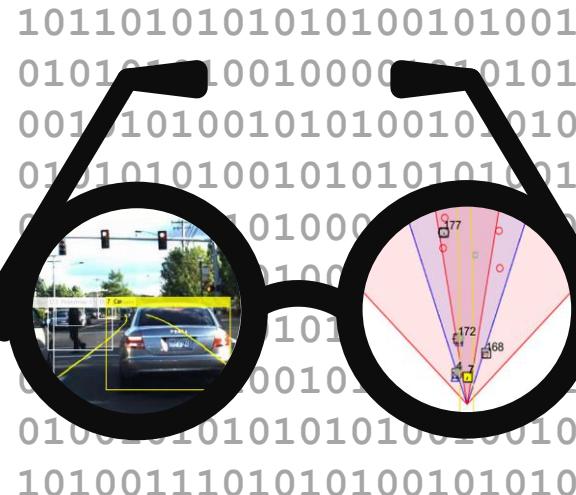
Learn more about sensor fusion

by exploring examples in the Automated Driving System Toolbox



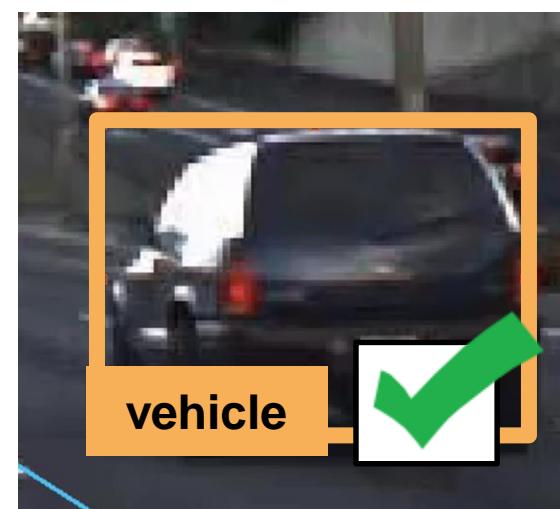
- **Design** multi-object tracker based on logged vehicle data
- **Generate C/C++** code from algorithm which includes a multi-object tracker
- **Synthesize driving scenario** to test multi-object tracker

The Automated Driving System Toolbox helps you...



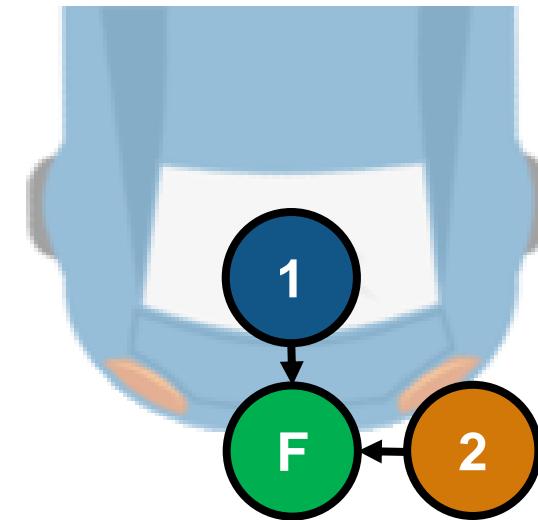
Visualize vehicle data

- Plot sensor detections
- Plot coverage areas
- Transform between image and vehicle coordinates



Detect objects in images

- Train deep learning networks
- Label ground truth
- Connect to other tools



Fuse multiple detections

- Design multi-object tracker
- Generate C/C++
- Synthesize driving scenarios