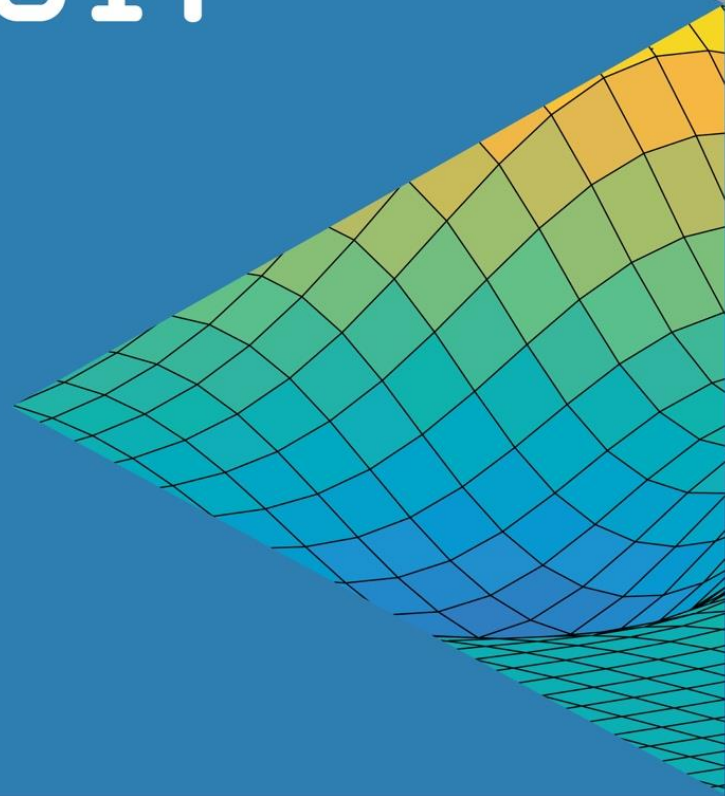


MATLAB EXPO 2017



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



Mark Corless
Principal Application Engineer



Dr. Amod Anandkumar
Senior Application Engineer

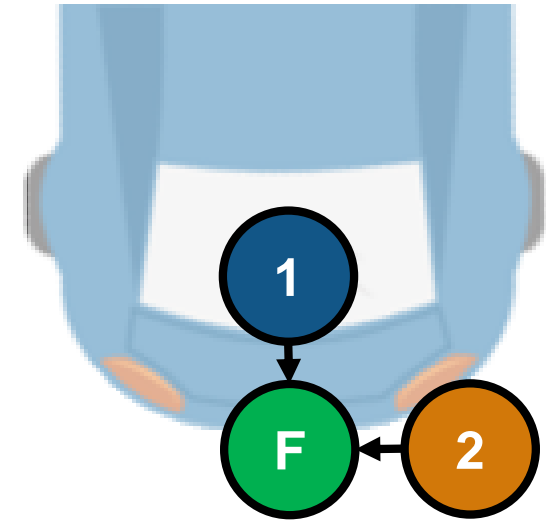
Some common questions from automated driving engineers



**How can I
visualize vehicle
data?**

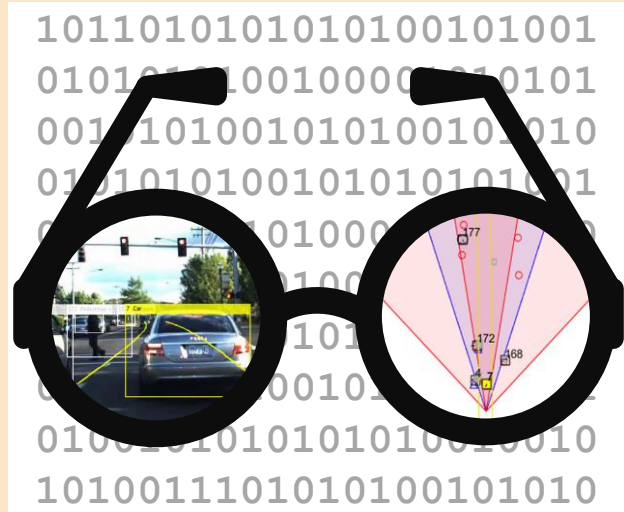


**How can I
detect objects in
images?**

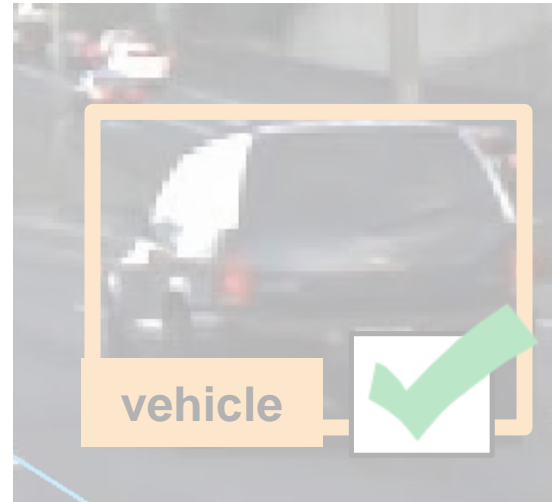


**How can I
fuse multiple
detections?**

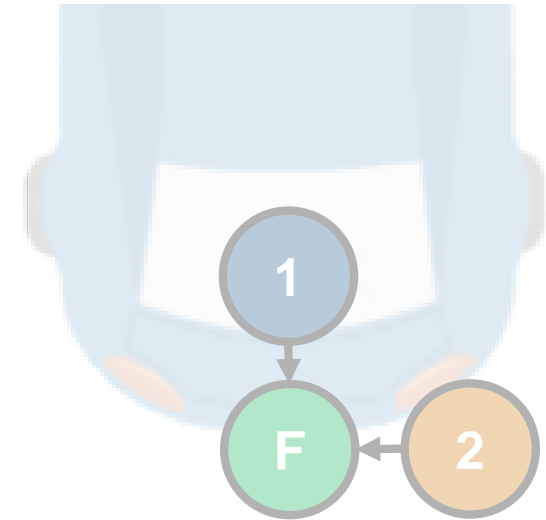
Some common questions from automated driving engineers



**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

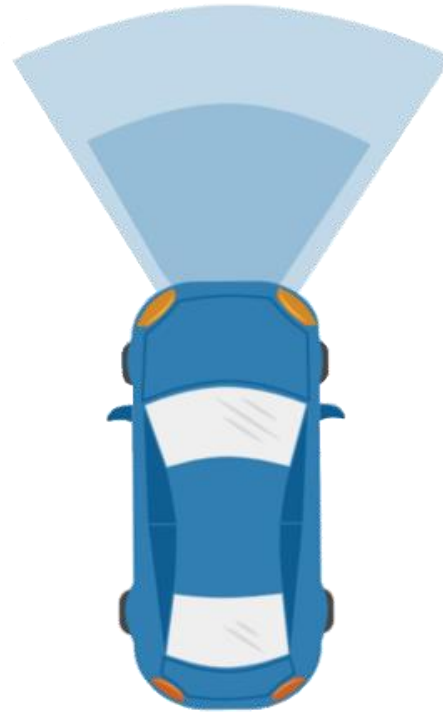
**Radar-based
object detector**

**Vision-based
object detector**

Lidar

Lane detector

**Inertial
measurement
unit**



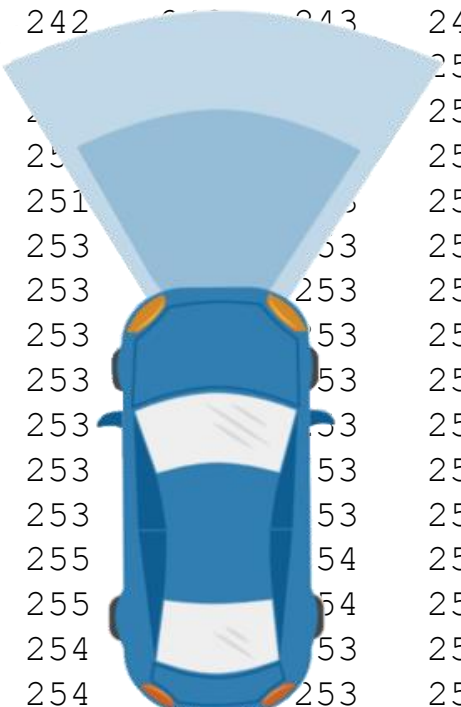
Examples of automated driving sensor data

Camera (640 x 480 x 3)

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

Vision-based object detector

Lane detector



Radar-based object detector

Lidar

Inertial measurement unit

Examples of automated driving sensor data

Camera (640 x 480 x 3)

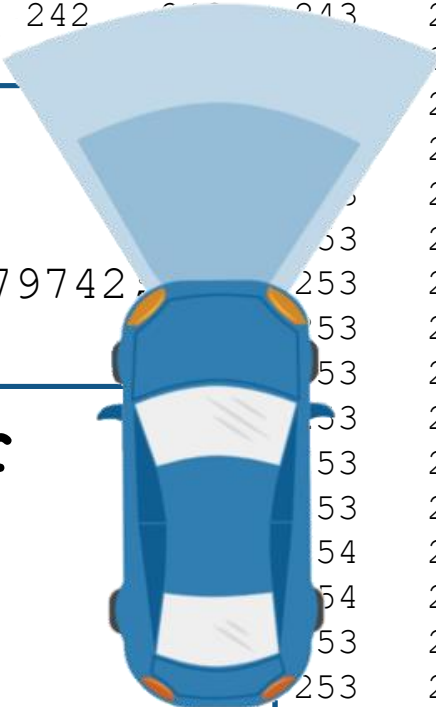
```
239 239 237 238 241 241 241 242 243 243
252 252 251 252 252 253 253
```

Vision Detector

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

Lane Detector

```
Tr Left
Cl
Po IsValid: 1
Ve Confidence: 3
Si BoundaryType: 3
Dete Offset: 1.68
Tr HeadingAngle: 0.002
Cl Curvature: 0.0000228
Po Right
Ve IsValid: 1
Si Confidence: 3
```



Radar-based object detector

Inertial measurement unit

Lidar

Examples of automated driving sensor data

Camera (640 x 480 x 3)

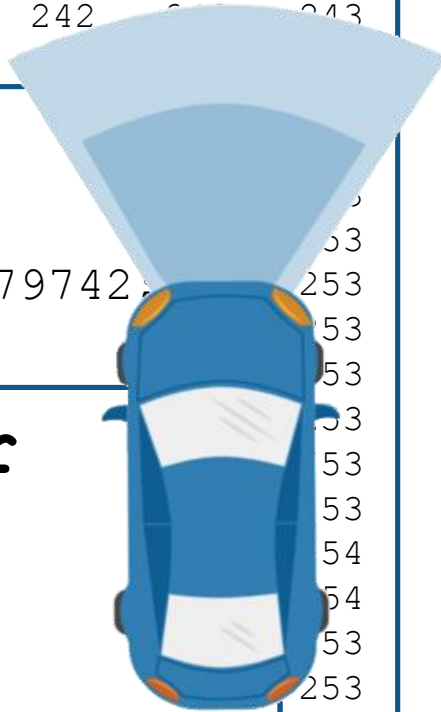
```
239 239 237 238 241 241 241 242 243
252 252 251 252 252 253 253
```

Vision Detector

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

Lane Detector

```
25 Tr Left
25 Cl
25 Po IsValid: 1
25 Ve Confidence: 3
25 Si BoundaryType: 3
25 Detec Offset: 1.68
25 Tr HeadingAngle: 0.002
25 Cl Curvature: 0.0000228
25 Po Right
25 Ve IsValid: 1
25 Si Confidence: 3
```



Radar Detector

```
SensorID = 2;
Timestamp = 1461634696407521;
NumDetections = 23;
Detections(1)
  TrackID: 0
  TrackStatus:
  Position:
  Velocity:
  Amplitude:
Detections(2)
  TrackID: 1
  TrackStatus: 6
  Position: [19.59 0.34]
  Velocity: [4.92 0]
  Amplitude:
Detections(3)
  TrackID: 12
  TrackStatus: 5
```

Lidar

Inertial measurement unit

Examples of automated driving sensor data

Camera (640 x 480 x 3)

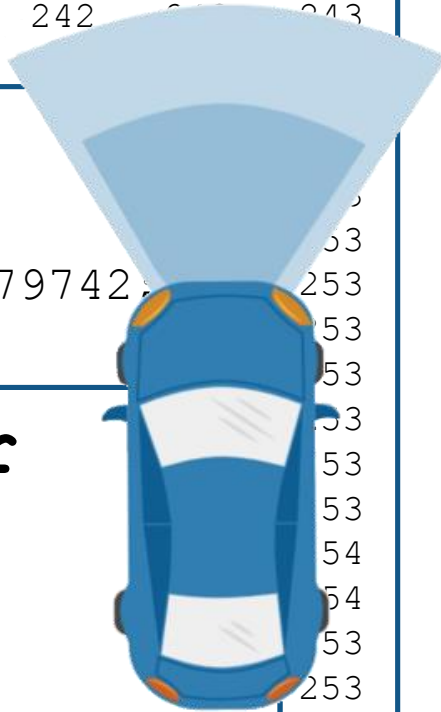
```
239 239 237 238 241 241 241 242 243
252 252 251 252 252 253 253
```

Vision Detector

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

Lane Detector

```
25 Tr Left
25 Cl
25 Po IsValid: 1
25 Ve Confidence: 3
25 Si BoundaryType: 3
25 Detec Offset: 1.68
25 Tr HeadingAngle: 0.002
25 Cl Curvature: 0.0000228
25 Po Right
25 Ve IsValid: 1
25 Si Confidence: 3
```



Radar Detector

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

Lidar (47197 x 3)

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

**Inertial
measurement
unit**

Examples of automated driving sensor data

Camera (640 x 480 x 3)

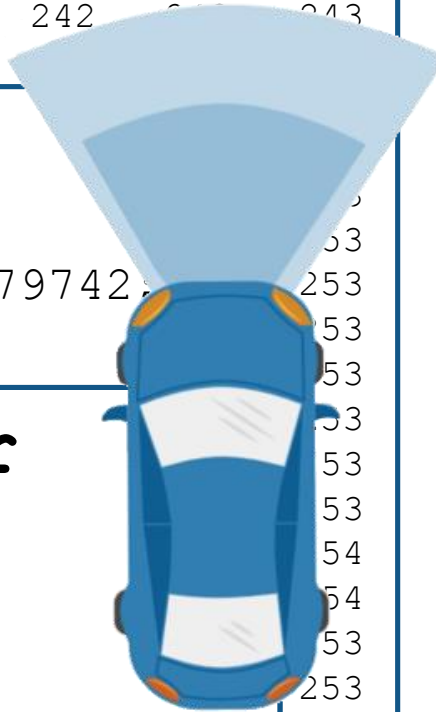
```
239 239 237 238 241 241 241 242 243
252 252 251 252 252 253 253
```

Vision Detector

```
25
25
25 SensorID = 1;
25 Timestamp = 1461634696379742;
25 NumDetections = 6;
```

Lane Detector

```
25
25 Tr Left
25 Cl
25 Po IsValid: 1
25 Ve Confidence: 3
25 Si BoundaryType: 3
25 Detec Offset: 1.68
25 Tr HeadingAngle: 0.002
25 Cl Curvature: 0.000
25 Po Right
25 Ve IsValid: 1
25 Si Confidence: 3
```



Radar Detector

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

Detection

```
TrackID
TrackSt -12.2911 1.4790 -0.59
Positio -14.8852 1.7755 -0.64
Velocit -18.8020 2.2231 -0.73
Amplitu -25.7033 3.0119 -0.92
Detection -0.0632 0.0815 1.25
TrackID -0.0978 0.0855 1.25
TrackSt -0.2814 0.1064 1.25
```

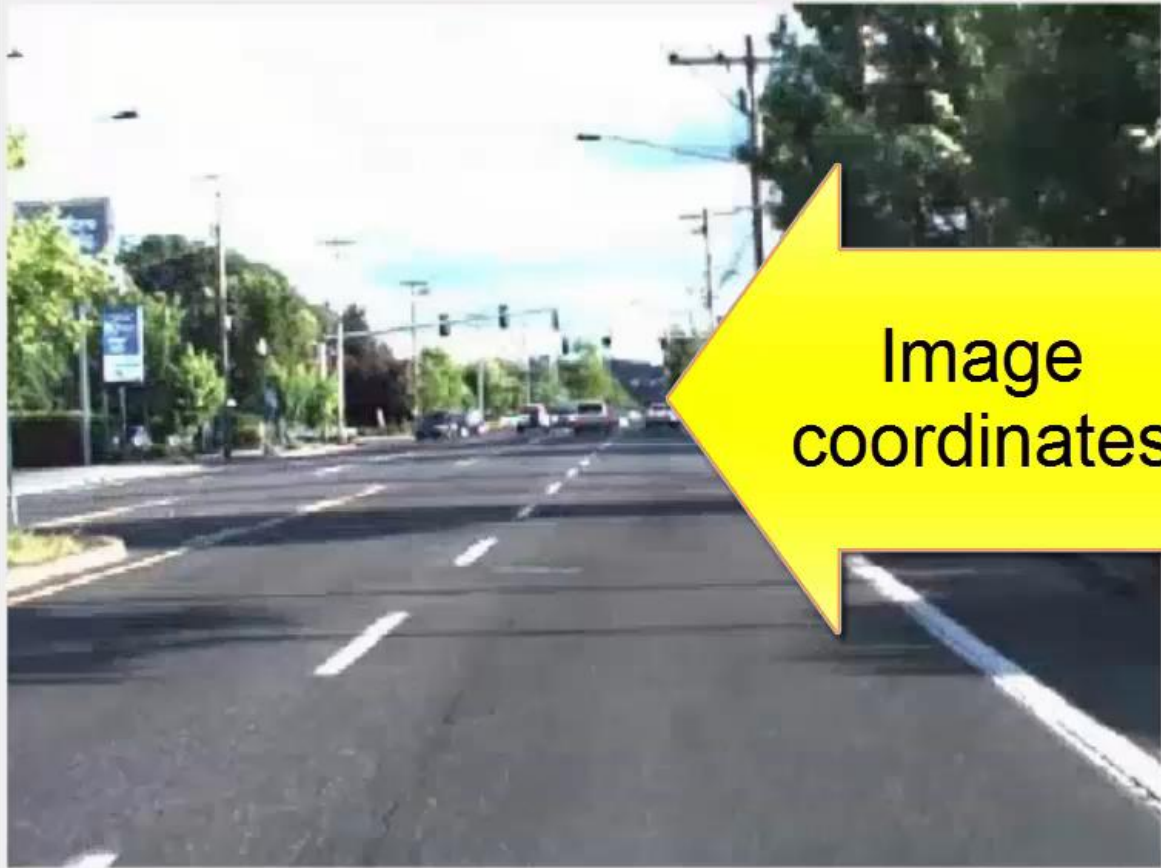
Lidar (47197 x 3)

Inertial Measurement Unit

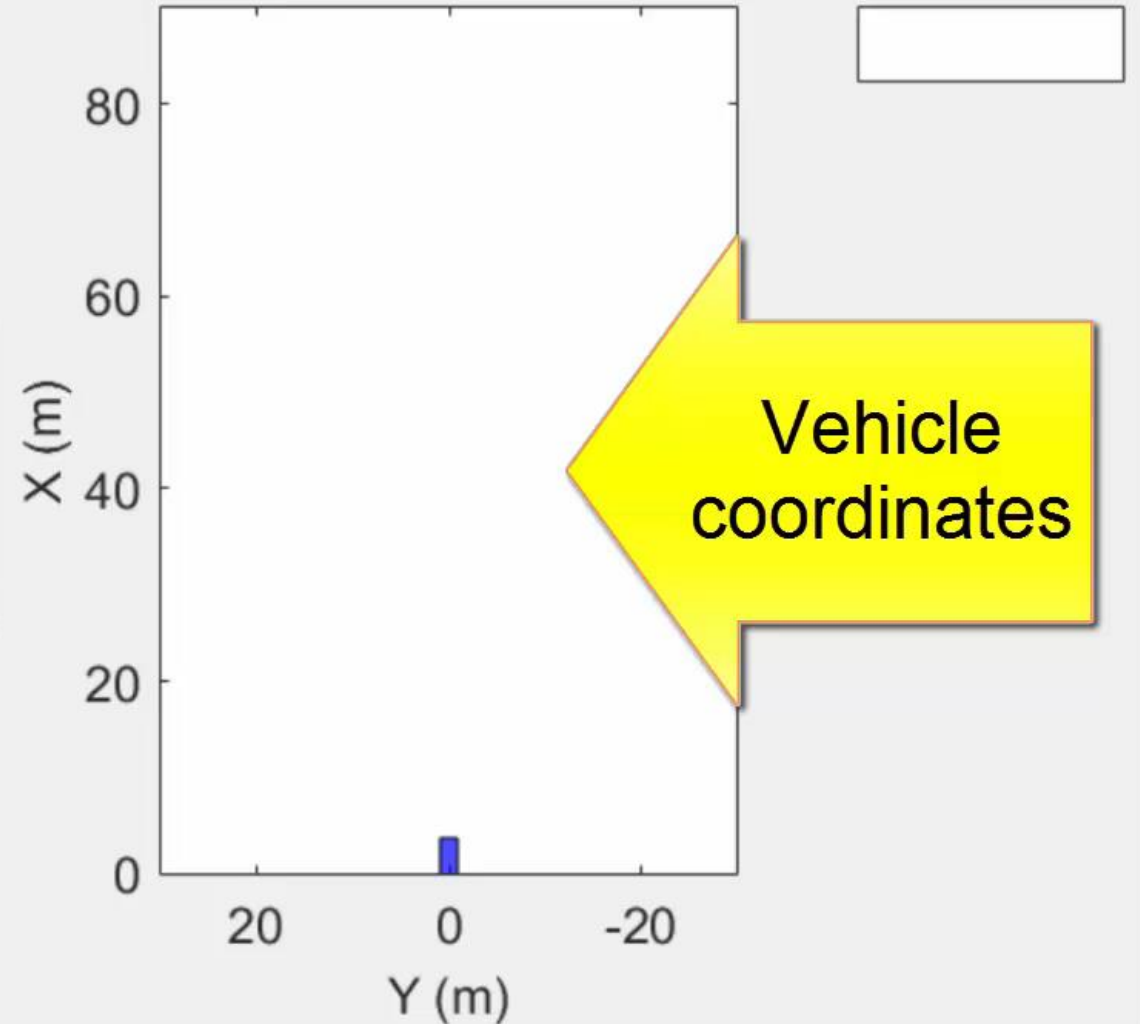
```
Timestamp: 1461634696379742
Velocity: 9.2795
YawRate: 0.0040
```

Visualize sensor data

Image Coordinates



Vehicle Coordinates

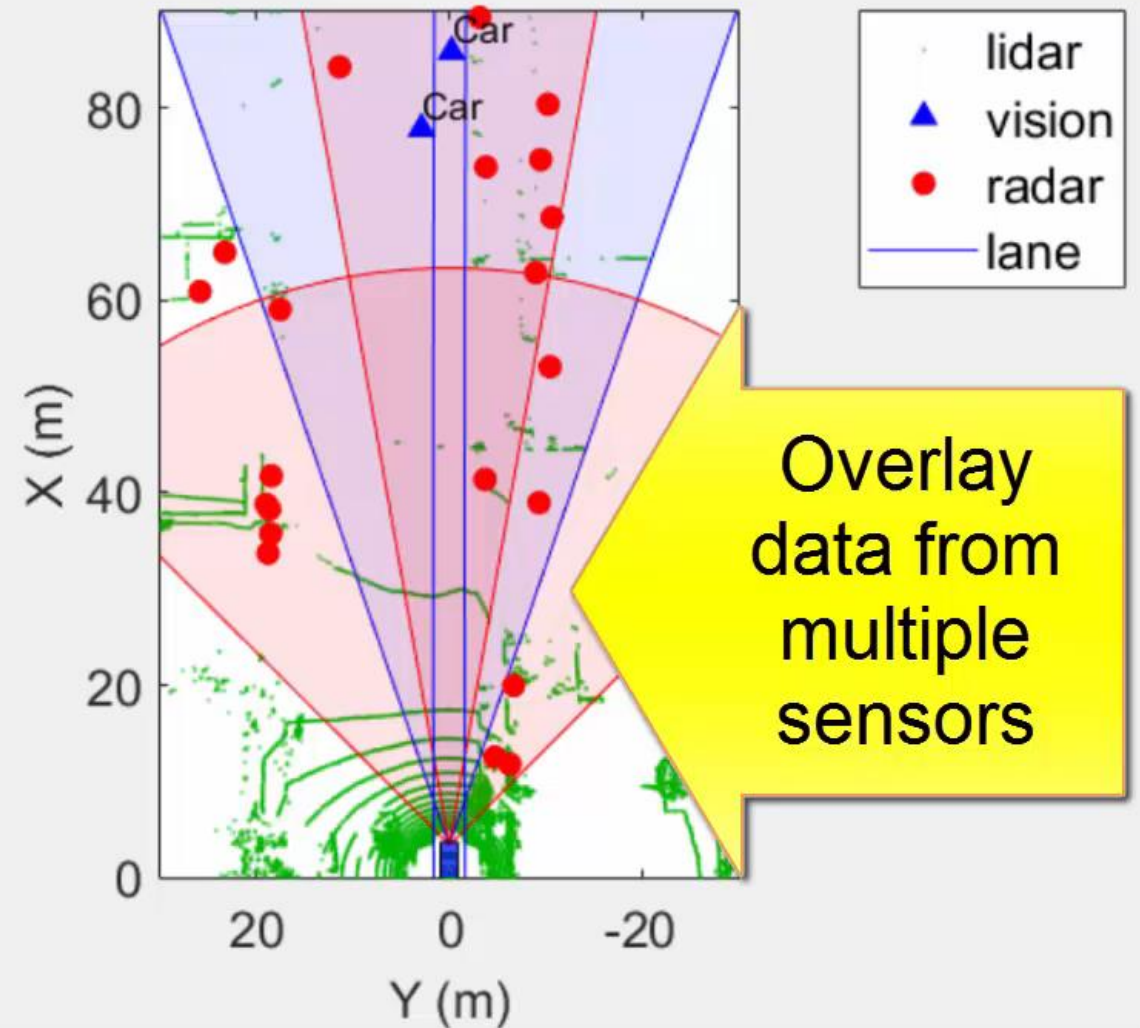


Visualize differences in sensor detections

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;

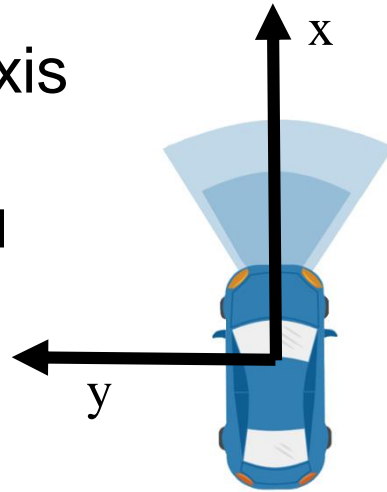
%% Plot image coordinates
ax1 = axes(...
    'Position', [0.02 0 0.55 1]);
im = imshow(frame, ...
    'Parent', ax1);
```



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');
```

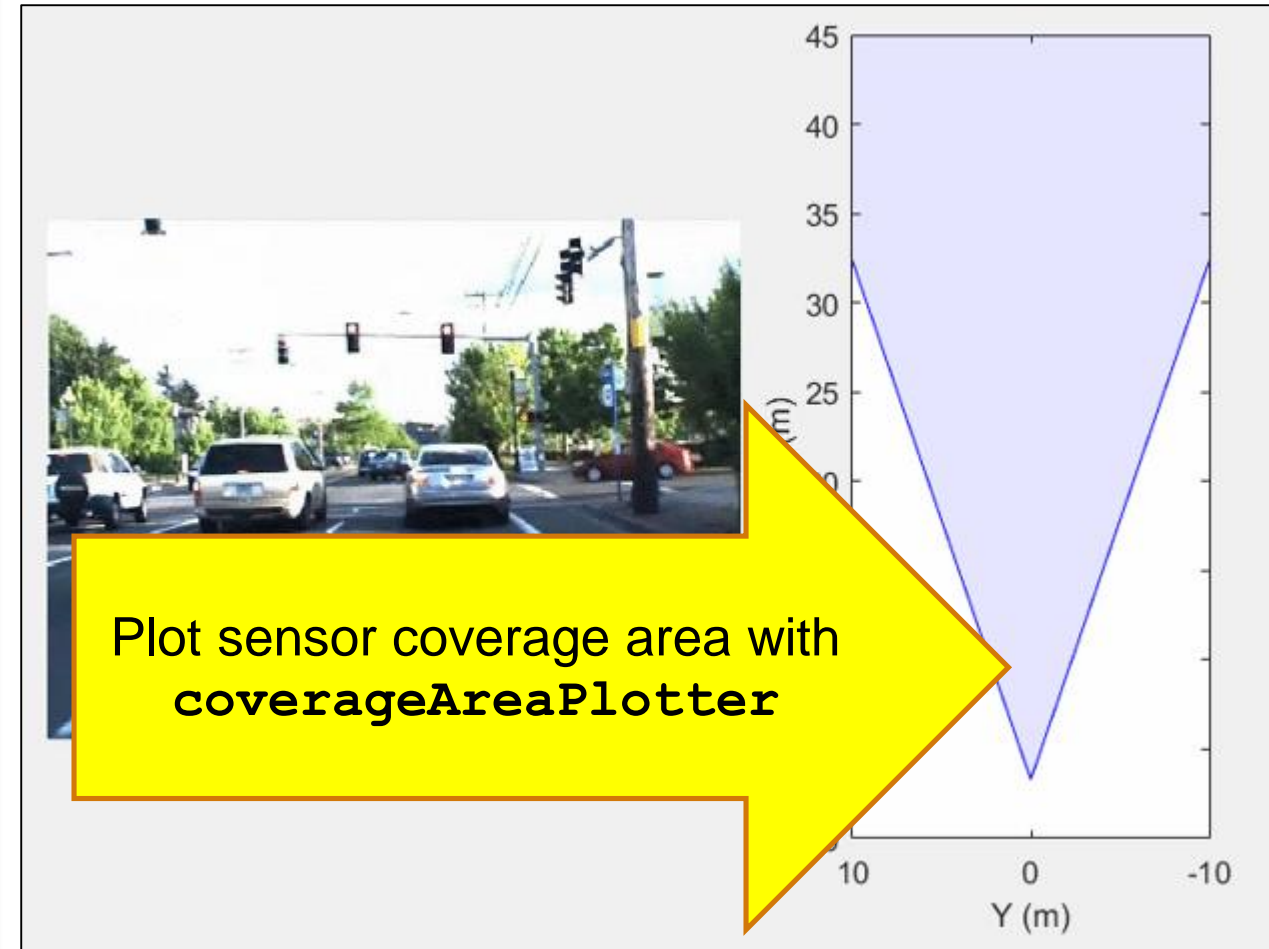


Plot in vehicle
coordinates with
birdsEyePlot

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)

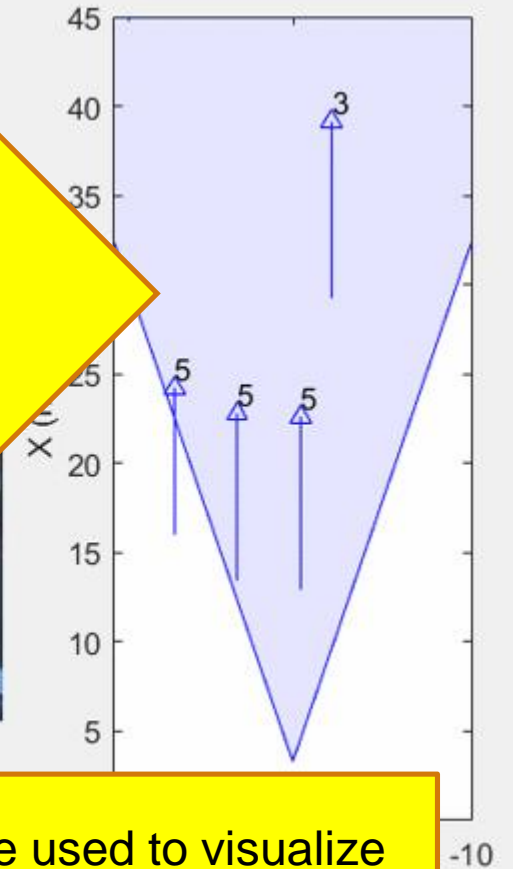
```

%% 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);

```

Plot vision detections with
detectionPlotter



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

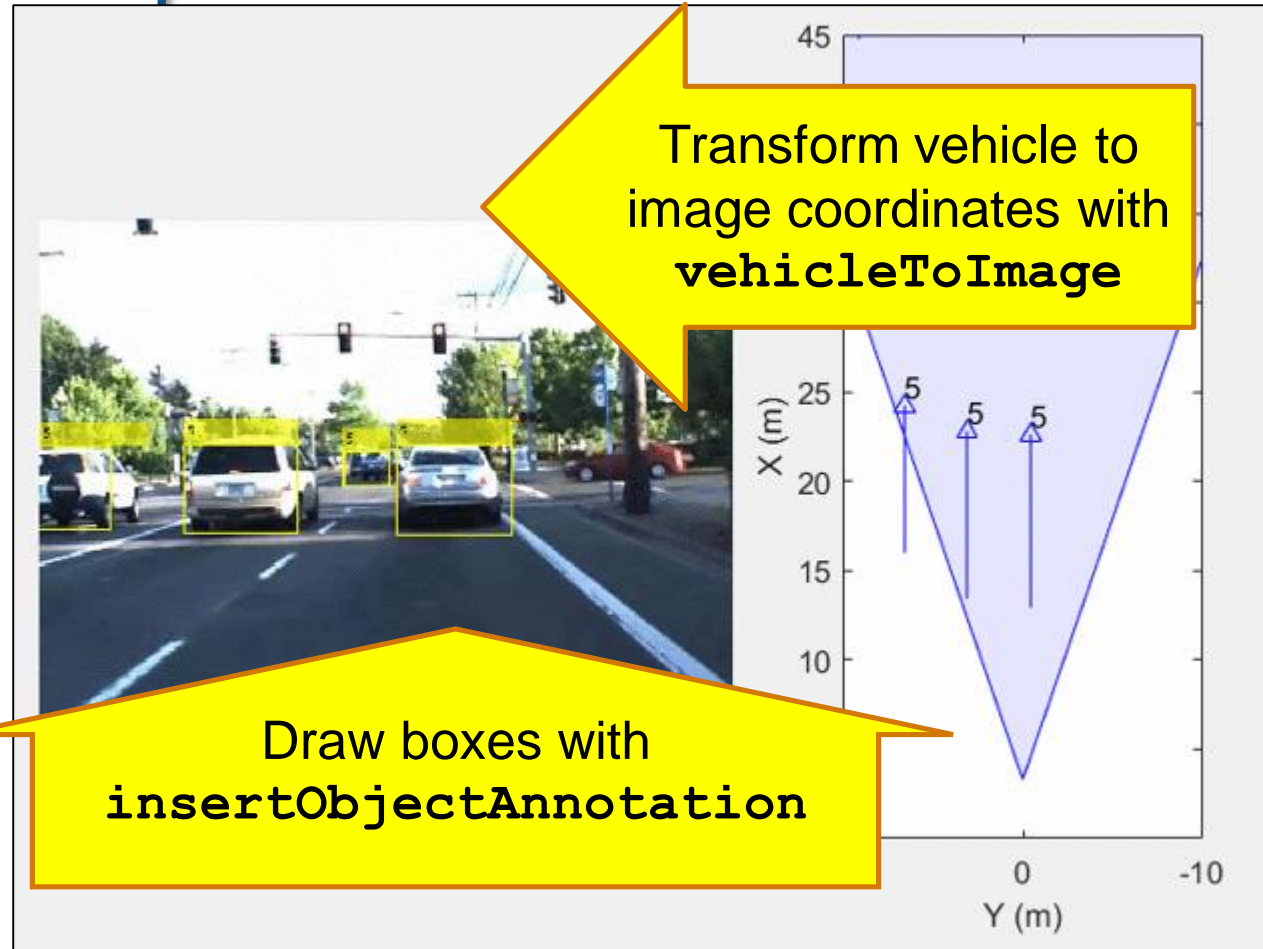
Visualize detections (image coordinates)

```
% 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)

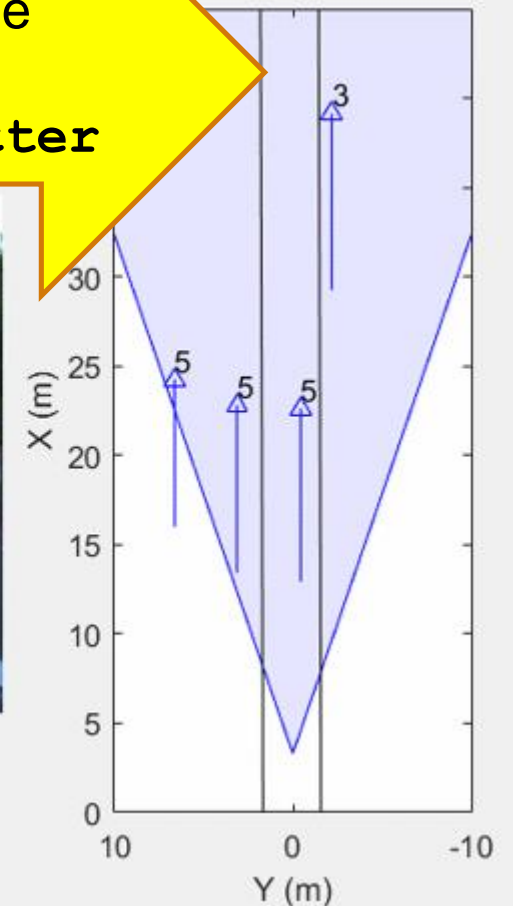
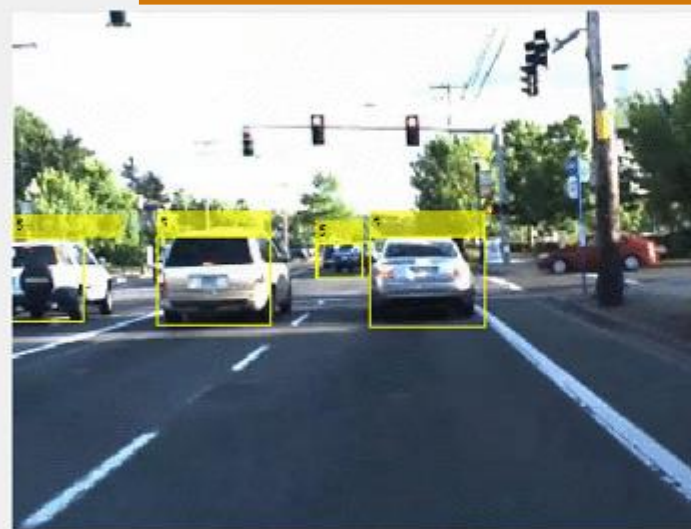
```

%% 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])

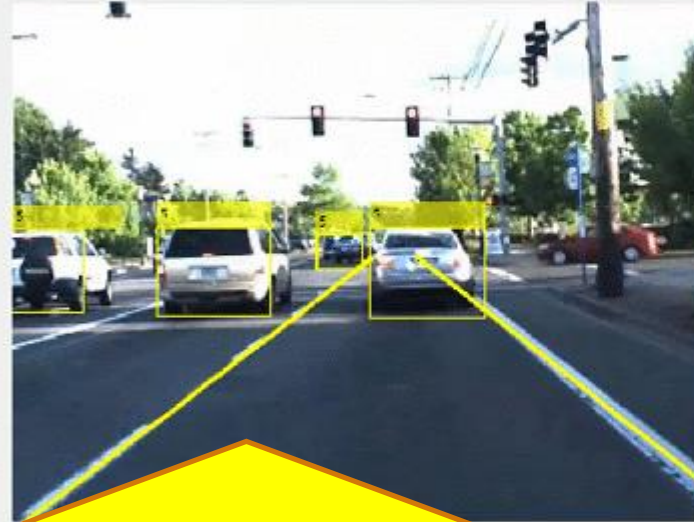
```

Plot lanes in vehicle coordinates with `laneBoundaryPlotter`

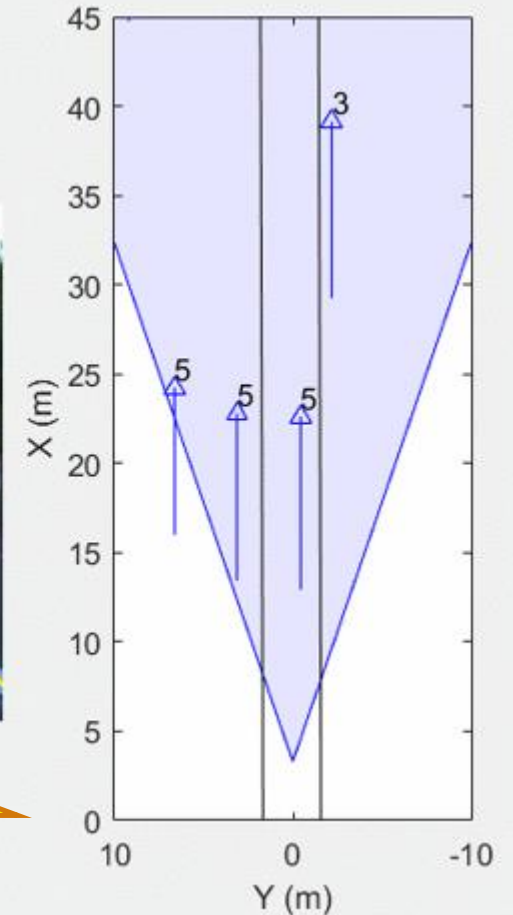


Visualize lane boundaries (image coordinates)

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



Plot lanes in image coordinates with `insertLaneBoundary`



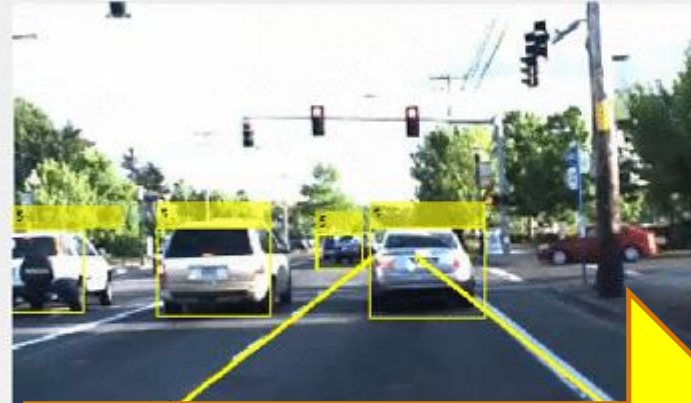
Visualize radar detections (vehicle coordinates)

```

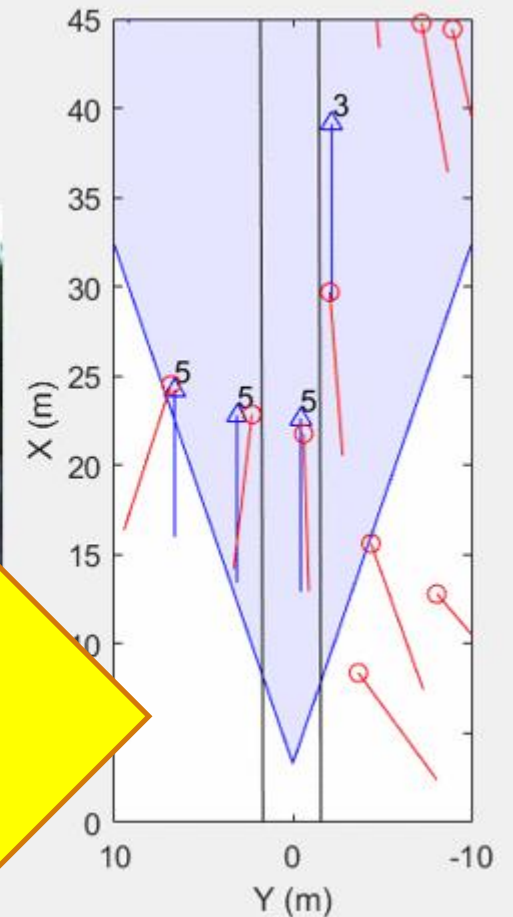
%% Create radar detection plotter
radarPlot = detectionPlotter(bep, ...
    'MarkerEdgeColor','red',...
    'Marker','o');

%% 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);

```



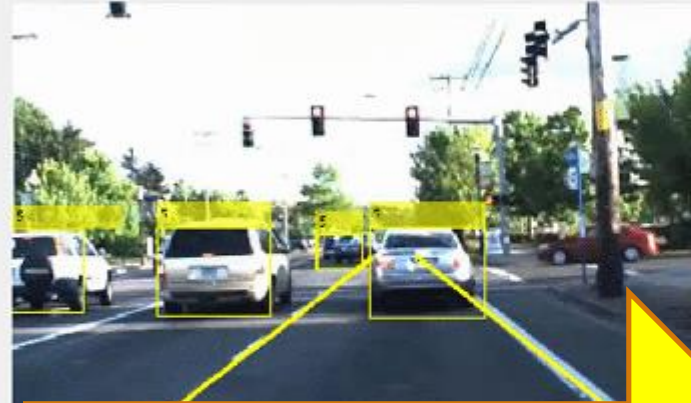
Plot radar detections just like vision detections with **detectionPlotter**



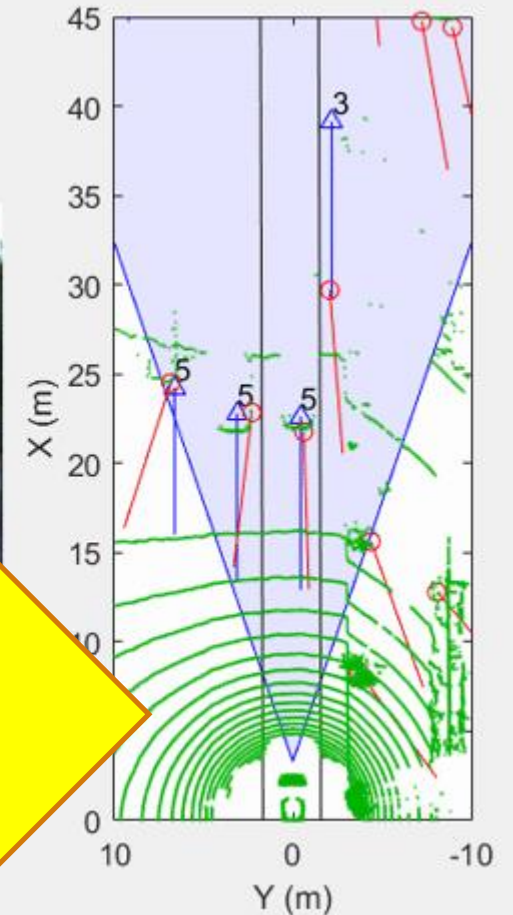
Visualize lidar point cloud (vehicle coordinates)

```
% 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);
```

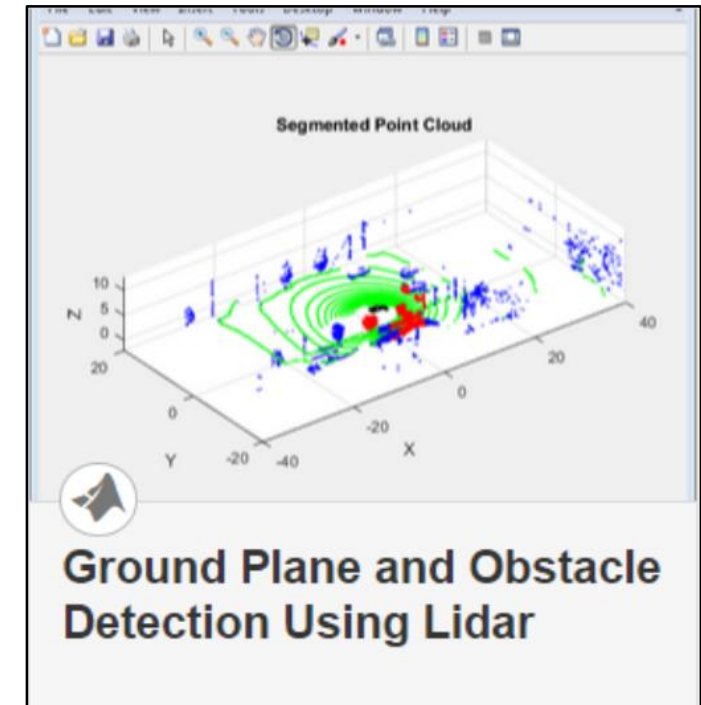
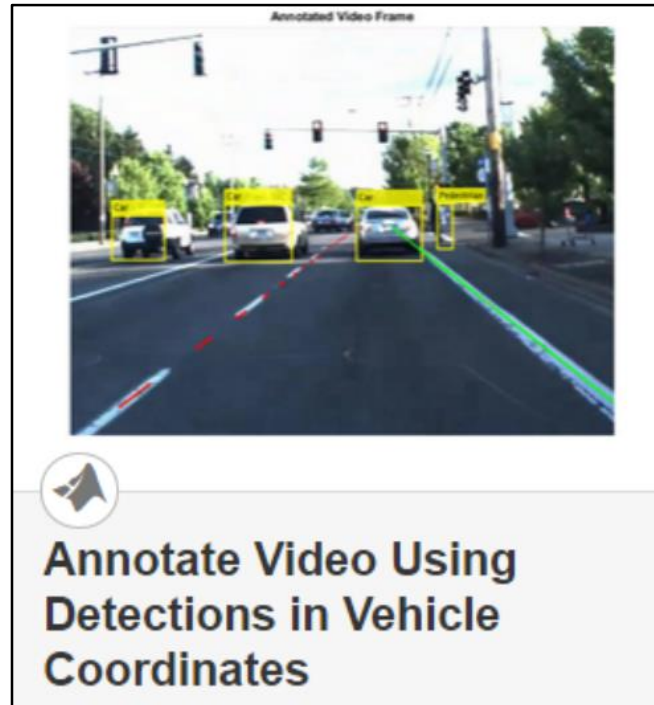
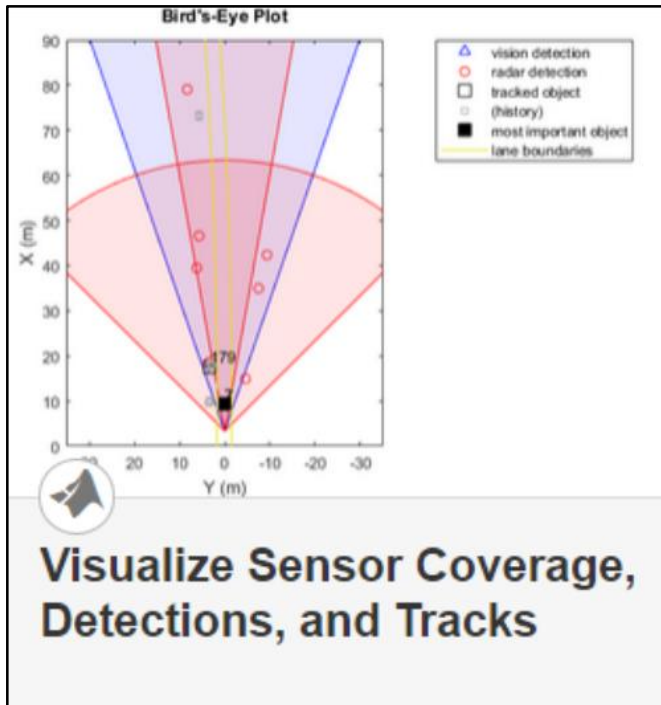


Plot lidar points just like
vision detections with
detectionPlotter



Learn more about visualizing vehicle data

by exploring examples in the Automated Driving System Toolbox

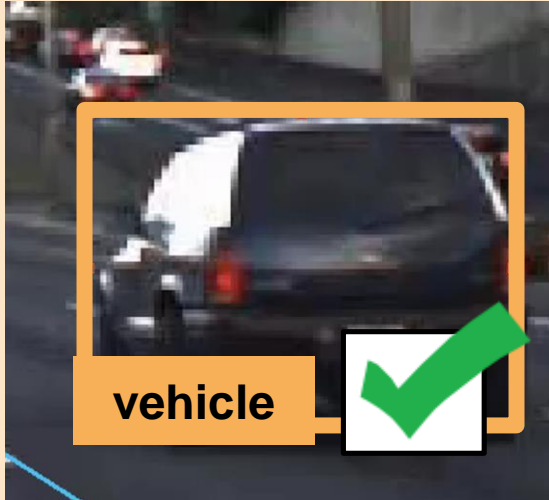


- **Plot object detectors in vehicle coordinates**
 - Vision & radar detector
 - Lane detectors
 - Detector coverage areas
- **Transform between vehicle and image coordinates**
- **Plot lidar point cloud**

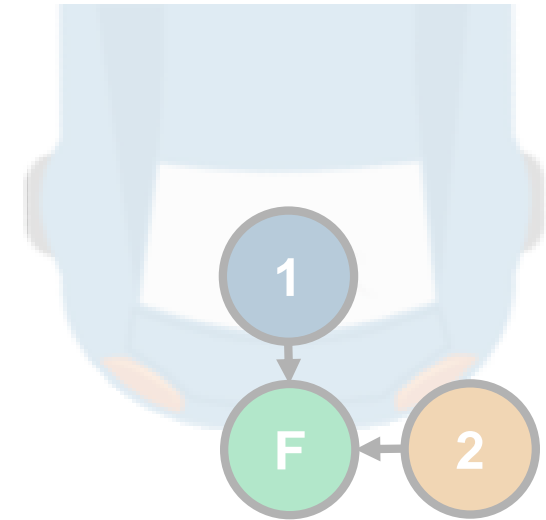
Some common questions from automated driving engineers



How can I
visualize vehicle
data?

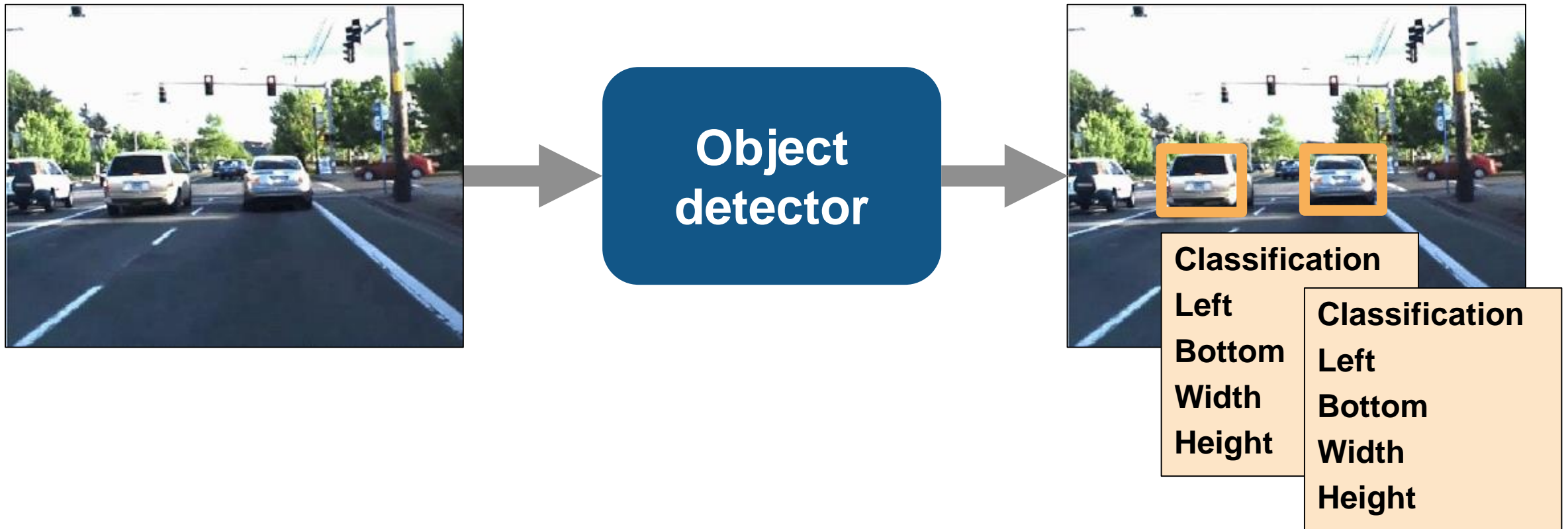


How can I
detect objects in
images?

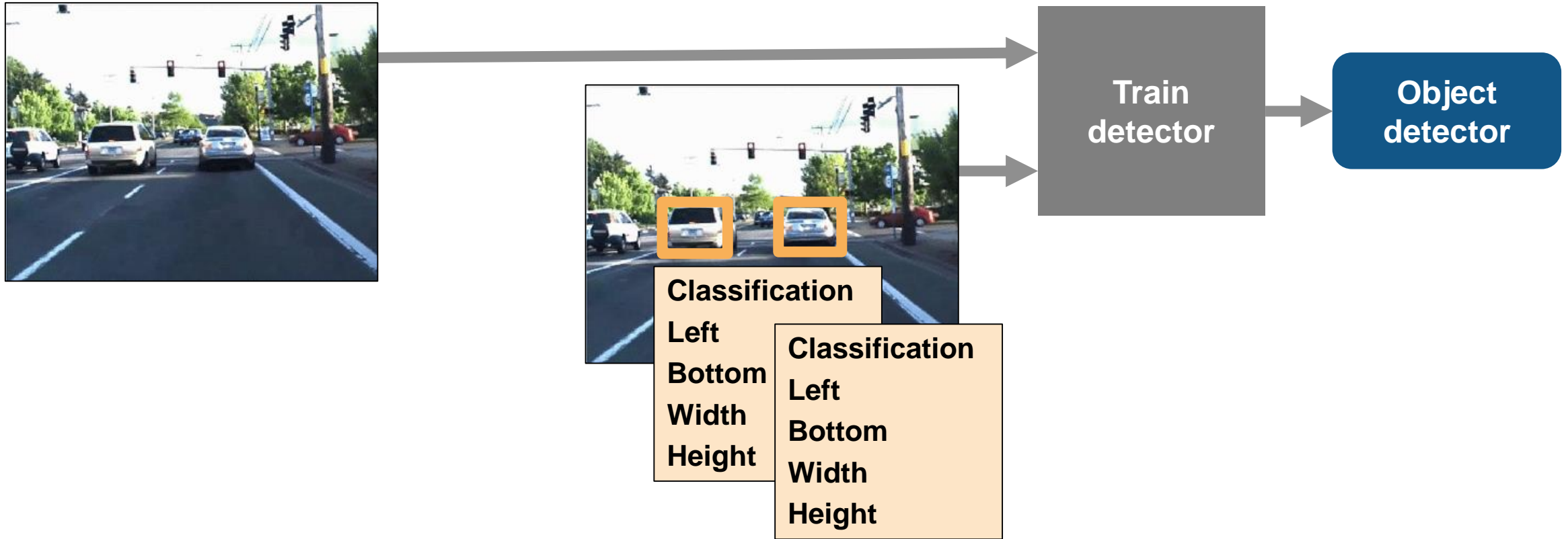


How can I
fuse multiple
detections?

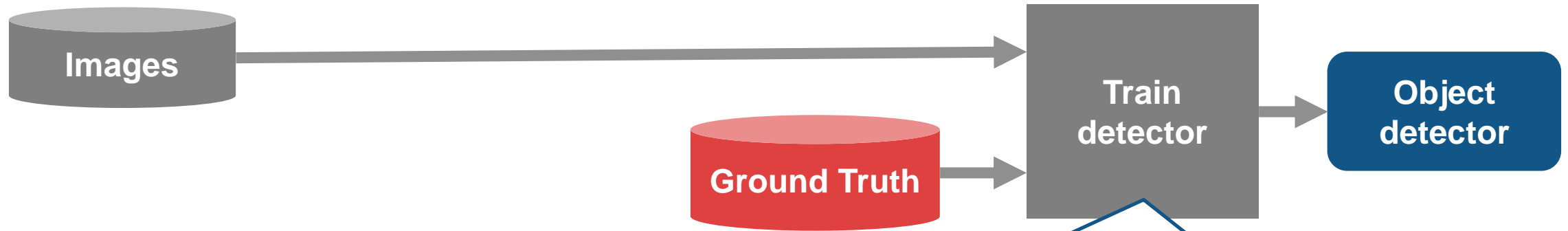
How can I detect objects in images?



Train object detectors based on ground truth



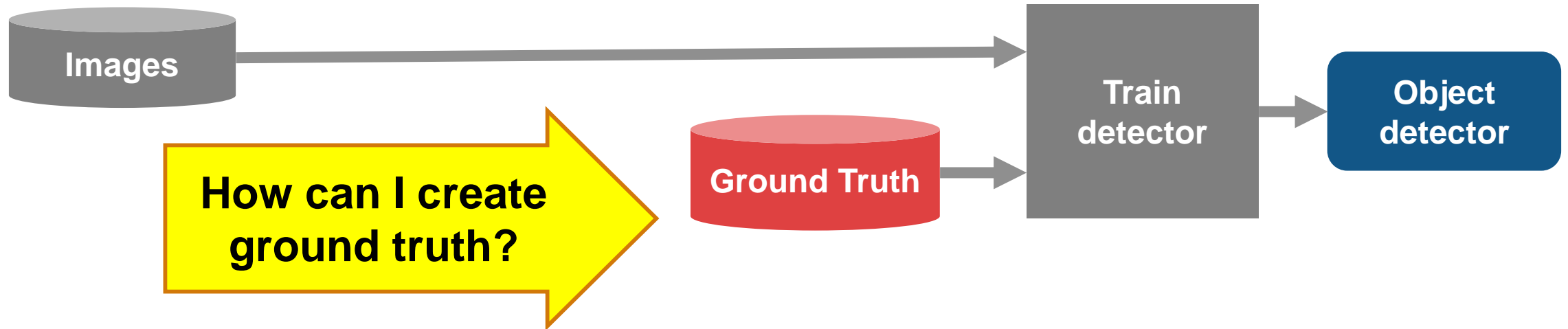
Train object detectors based on ground truth



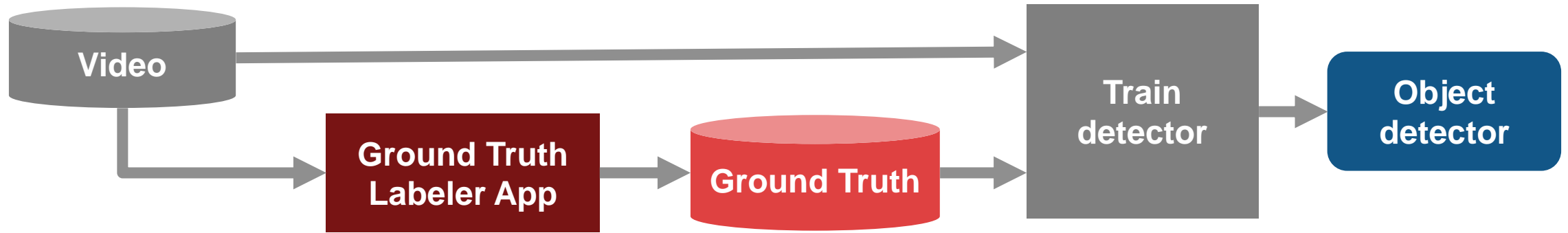
Design object detectors with the Computer Vision System Toolbox

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>

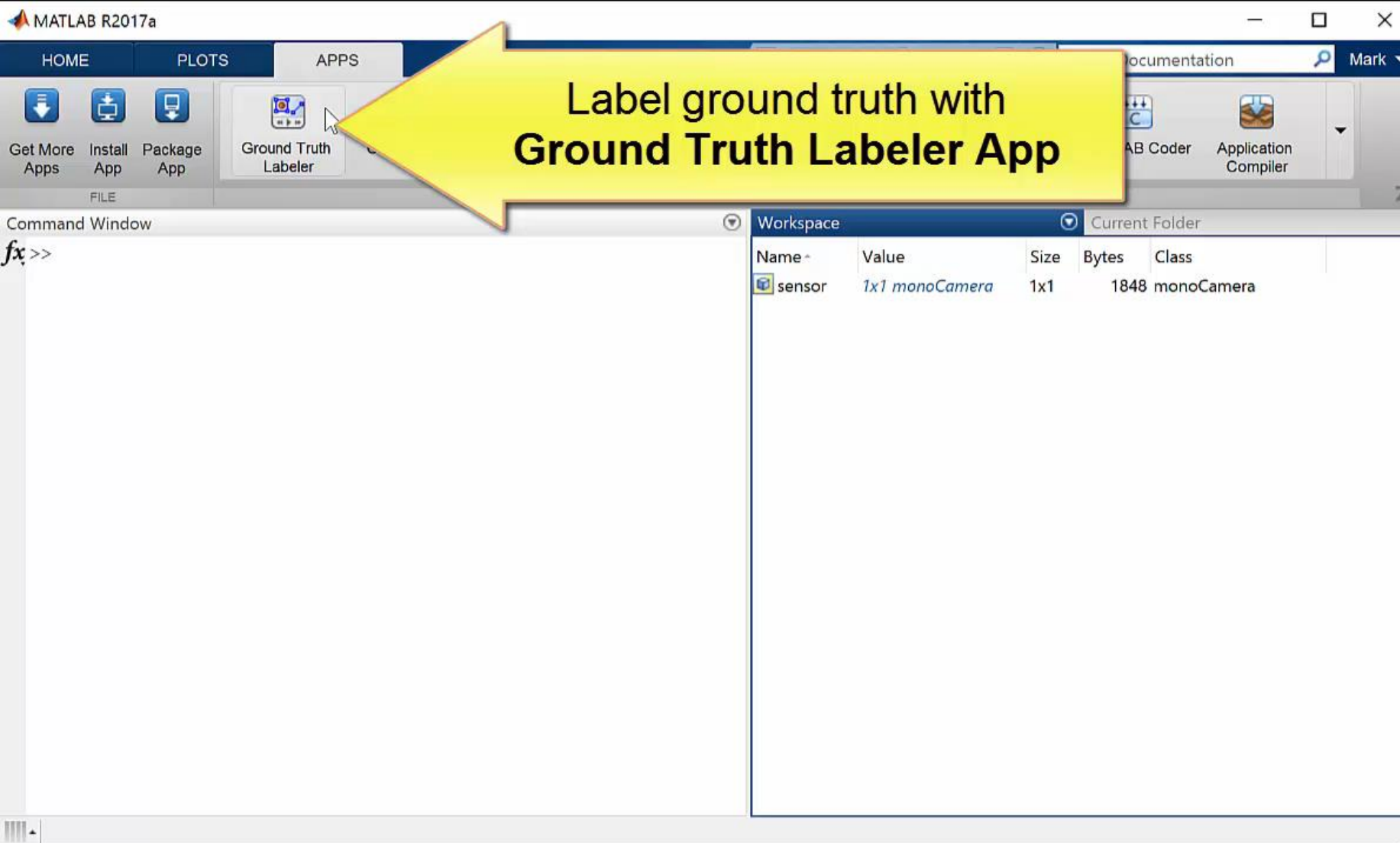
Specify ground truth to train detector



Specify ground truth to train detector



Manually label ground truth objects with Ground Truth Labeling App

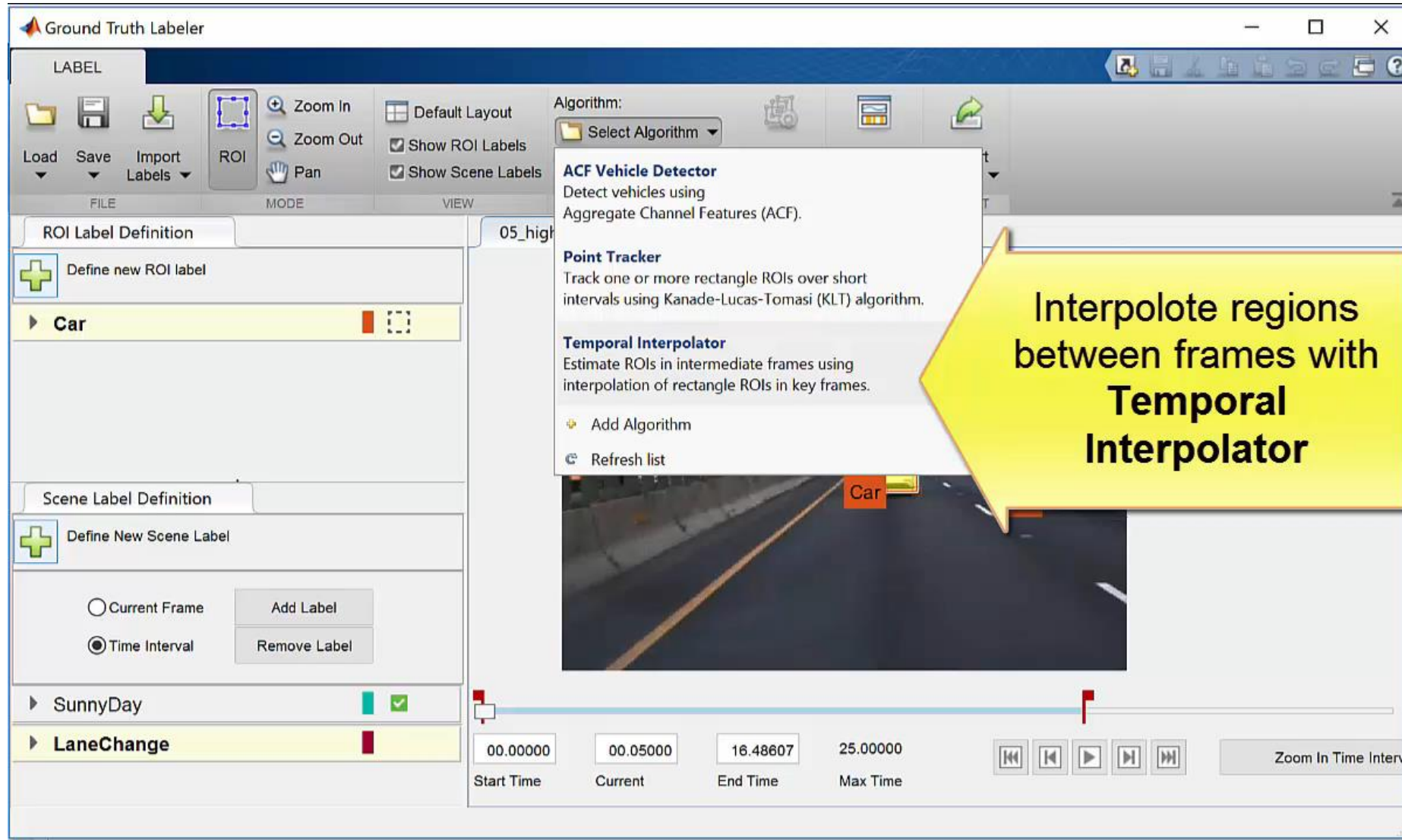


Label ground truth with
Ground Truth Labeler App

Command Window
fx >>

Name	Value	Size	Bytes	Class
sensor	1x1 monoCamera	1x1	1848	monoCamera

Automate labeling between manually labeled frames with temporal interpolator



The screenshot displays the Ground Truth Labeler software interface. The main window is titled "Ground Truth Labeler" and features a toolbar with icons for Load, Save, Import Labels, ROI, Zoom In, Zoom Out, Pan, Default Layout, Show ROI Labels, and Show Scene Labels. The interface is divided into several sections:

- ROI Label Definition:** Contains a "Define new ROI label" button and a list of labels. The "Car" label is currently selected and highlighted in yellow.
- Scene Label Definition:** Contains a "Define New Scene Label" button and radio buttons for "Current Frame" and "Time Interval". The "Time Interval" option is selected. Below this, there are "Add Label" and "Remove Label" buttons.
- Algorithm List:** A dropdown menu is open, showing a list of algorithms:
 - ACF Vehicle Detector:** Detect vehicles using Aggregate Channel Features (ACF).
 - Point Tracker:** Track one or more rectangle ROIs over short intervals using Kanade-Lucas-Tomasi (KLT) algorithm.
 - Temporal Interpolator:** Estimate ROIs in intermediate frames using interpolation of rectangle ROIs in key frames.Buttons for "Add Algorithm" and "Refresh list" are also visible.
- Video Player:** A video frame is shown with a red bounding box labeled "Car" around a vehicle. A yellow callout box with a black border points to the "Temporal Interpolator" algorithm, containing the text: "Interpolate regions between frames with **Temporal Interpolator**".
- Timeline:** A horizontal timeline at the bottom shows the current frame position. The "Current" frame is at 16.48607, with "Start Time" at 00.00000 and "Max Time" at 25.00000. Playback controls and a "Zoom In Time Interv" button are also present.

Automate labeling based on a manually labeled frame with point tracker

The screenshot displays the 'Ground Truth Labeler' application window. The interface is divided into several sections:

- Top Panel:** Contains a 'LABEL' menu with options like 'Load', 'Save', and 'Import Labels'. It also includes 'Zoom In', 'Zoom Out', and 'Pan' controls. A 'Default Layout' section has checkboxes for 'Show ROI Labels' and 'Show Scene Labels'.
- Left Panel:**
 - ROI Label Definition:** Shows a list of labels, with 'Car' selected and highlighted in yellow.
 - Scene Label Definition:** Shows a list of scene labels, with 'SunnyDay' and 'LaneChange' visible.
- Center Panel:** Displays a video frame of a highway with a red bounding box around a car labeled 'Car'. A context menu is open over this ROI, listing algorithms:
 - Temporal Interpolator:** Estimate ROIs in intermediate frames using interpolation of rectangle ROIs in key frames.
 - ACF Vehicle Detector:** Detect vehicles using Aggregate Channel Features (ACF).
 - Point Tracker:** Track one or more rectangle ROIs over short intervals using Kanade-Lucas-Tomasi (KLT) algorithm. This option is currently selected by the mouse cursor.
- Bottom Panel:** Features a timeline with markers and numerical values: 'Start Time' (06.75310), 'Current' (06.75310), 'End Time' (14.54546), and 'Max Time' (25.00000). Playback controls and a 'Zoom In Time Interval' button are also present.

A large yellow arrow on the right side of the interface points to the 'Point Tracker' option in the context menu, with the text 'Track region with Point Tracker' written inside it.

Automate initial ground truth of vehicles with ACF ground truth detector

The screenshot displays the Ground Truth Labeler software interface. The main window shows a video frame of a highway with a car labeled "Car". The interface includes a menu bar with "FILE", "MODE", and "VIEW". The "MODE" menu is open, showing options like "Zoom In", "Zoom Out", and "Pan". The "VIEW" menu is also open, showing options like "Default Layout", "Show ROI Labels", and "Show Scene Labels".

The "Algorithm:" dropdown menu is set to "Point Tracker". The "ACF Vehicle Detector" algorithm is selected, with a description: "Detect vehicles using Aggregate Channel Features (ACF)". Other algorithms listed include "Point Tracker" (Track one or more rectangle ROIs over short intervals using Kanade-Lucas-Tomasi (KLT) algorithm.) and "Temporal Interpolator" (Estimate ROIs in intermediate frames using interpolation of rectangle ROIs in key frames.).

A yellow callout box with a black border and arrow points to the "ACF Vehicle Detector" algorithm, containing the text: "Detect initial regions with Vehicle Detector".

The "ROI Label Definition" panel on the left shows a list of labels: "Car" (selected), "SunnyDay", and "LaneChange". The "Scene Label Definition" panel shows "SunnyDay" and "LaneChange" as scene labels.

The bottom of the interface features a timeline with a play button and a "Zoom In Time Interval" button. The timeline shows the following values:

Start Time	Current	End Time	Max Time
14.54546	14.54546	21.12013	25.00000

Export labeled regions as MATLAB time table

The screenshot displays the Ground Truth Labeler application window. A yellow arrow points to the 'View Label Summary' button in the top toolbar, with the text 'Explore labels by viewing label summary' overlaid on it. The interface includes a 'FILE' menu with 'Load', 'Save', and 'Import Labels' options. The main workspace shows a video frame titled '05_highway_lanechange_25s.mp4' with three cars labeled 'Car'. A 'Scene Labels' legend on the right identifies 'SunnyDay' (cyan) and 'LaneChange' (red). The bottom panel features a timeline with 'Start Time' (00.00000), 'Current' (14.55000), 'End Time' (25.00000), and 'Max Time' (25.00000) fields, along with playback controls and a 'Zoom In Time Interval' button.

Ground Truth Labeler

LABEL

Load Save Import Labels

FILE

ROI Label Definition

05_highway_lanechange_25s.mp4

Define new ROI label

Car

Scene Label Definition

Define New Scene Label

Current Frame Add Label

Time Interval Remove Label

SunnyDay

LaneChange

Scene Labels

SunnyDay

LaneChange

00.00000 14.55000 25.00000 25.00000

Start Time Current End Time Max Time

Zoom In Time Interval

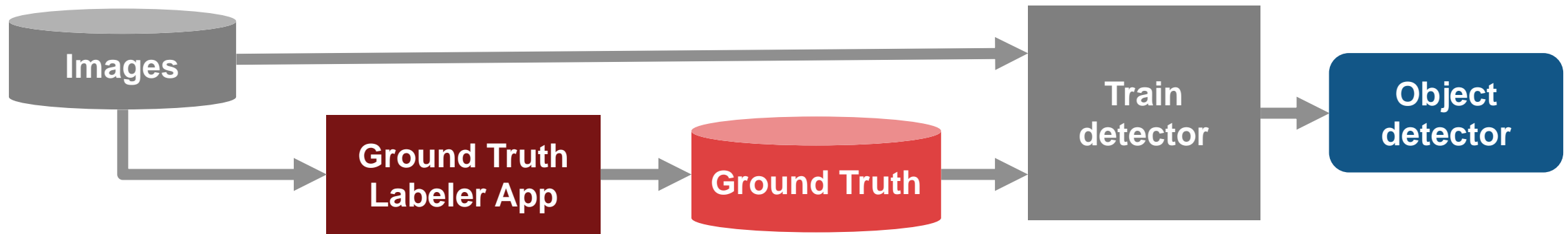
Explore labels by viewing label summary

View Label Summary

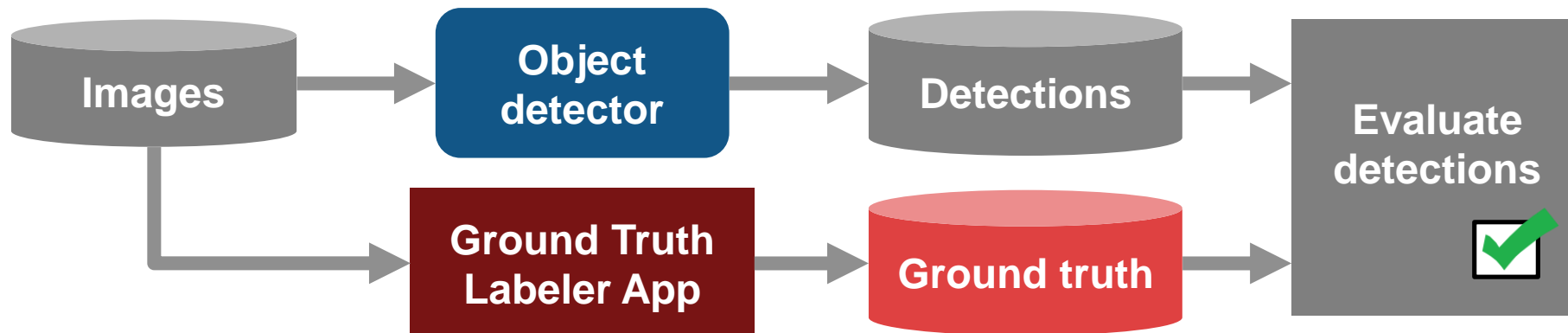
Export Labels

SUMMARY EXPORT

Ground truth labeling to train detectors



Ground truth labeling to evaluate detectors



Customize Ground Truth Labeler App

Ground Truth Labeler - gtlCustomizations

LABEL

Load Save Import Labels ROI Zoom In Zoom Out Pan

FILE MODE VIEW AUTOMATE LABELING SUMMARY EXPORT

Algorithm: Select Algorithm Automate View Label Summary Export Labels

Show ROI Labels Show Scene Labels

ROI Label Definition

Define new ROI label

- Car
- Pedestrian
- StopLight
- Lane**

Scene Label Definition

Define New Scene Label

Current Frame Add Label
 Time Interval Remove Label

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

01_city_c2s_fcw_10s.mp4

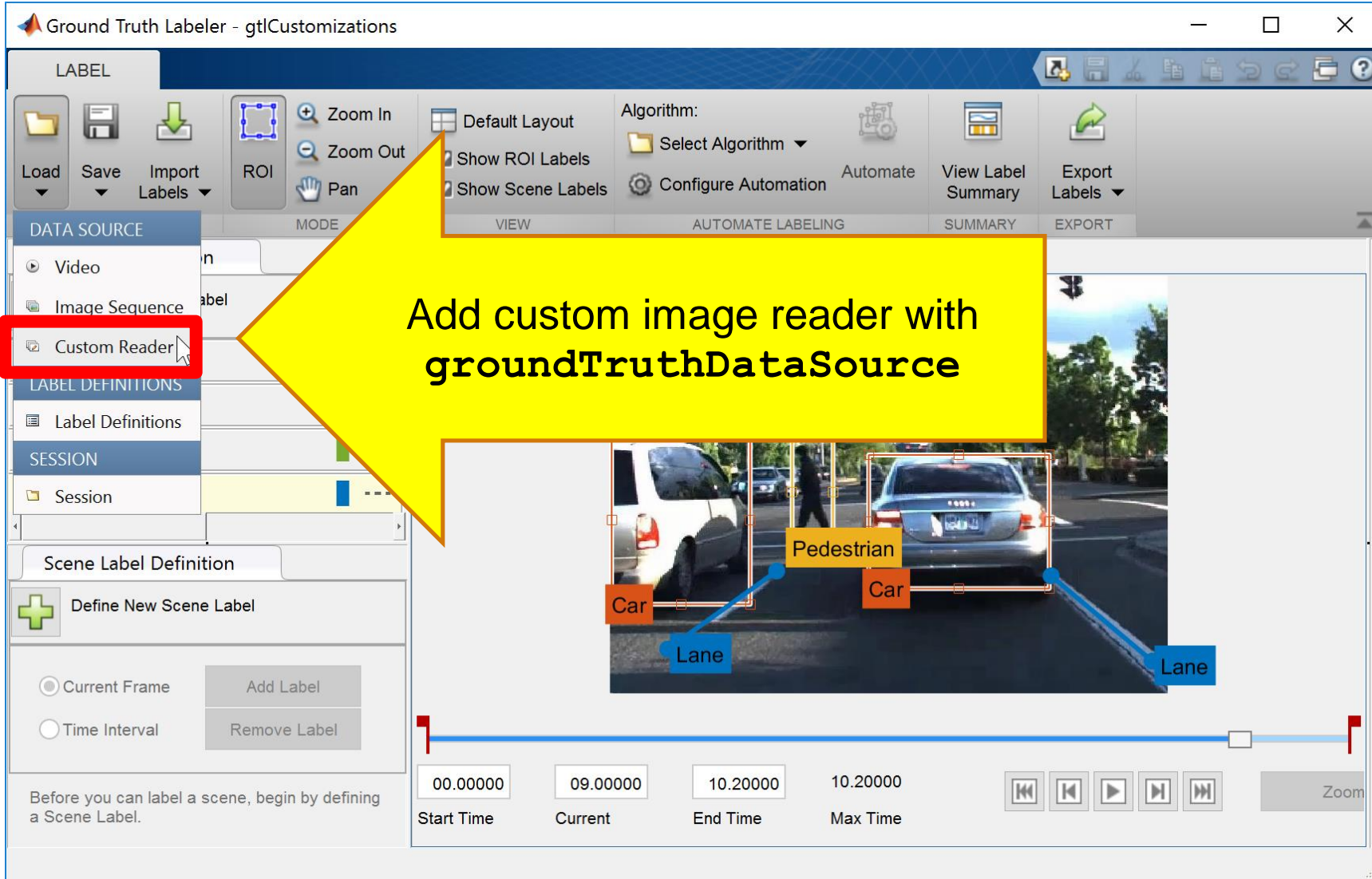
Car Pedestrian StopLight StopLight Lane Lane

00.00000 09.00000 10.20000 10.20000

Start Time Current End Time Max Time

Zoom

Customize Ground Truth Labeler App



The screenshot displays the Ground Truth Labeler application window titled "Ground Truth Labeler - gtlCustomizations". The interface includes a top toolbar with icons for Load, Save, Import Labels, ROI, Zoom In, Zoom Out, Pan, Default Layout, Show ROI Labels, Show Scene Labels, Algorithm selection, Automate, View Label Summary, and Export Labels. Below the toolbar are tabs for DATA SOURCE, LABEL DEFINITIONS, and SESSION. The DATA SOURCE tab is active, and the "Custom Reader" option is highlighted with a red box. A large yellow arrow points from the "Custom Reader" option to a central text box that reads "Add custom image reader with `groundTruthDataSource`". The main workspace shows a video frame of a street scene with bounding boxes for "Car", "Pedestrian", and "Lane". A timeline at the bottom indicates the current frame is at 09.00000, with start and end times at 00.00000 and 10.20000 respectively. A "Zoom" button is visible on the right side of the timeline.

Ground Truth Labeler - gtlCustomizations

LABEL

Load Save Import Labels ROI Zoom In Zoom Out Pan Default Layout Show ROI Labels Show Scene Labels Algorithm: Select Algorithm Automate View Label Summary Export Labels

DATA SOURCE

Video Image Sequence **Custom Reader**

LABEL DEFINITIONS

Label Definitions

SESSION

Session

Scene Label Definition

Define New Scene Label

Current Frame Add Label

Time Interval Remove Label

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

00.00000 09.00000 10.20000 10.20000

Start Time Current End Time Max Time

Zoom

Add custom image reader with `groundTruthDataSource`

Customize Ground Truth Labeler App

The screenshot shows the 'Ground Truth Labeler - gtlCustomizations' application window. The interface includes a top toolbar with 'FILE', 'MODE', and 'VIEW' sections. The 'MODE' section contains 'Zoom In', 'Zoom Out', and 'Pan' options. The 'VIEW' section has 'Default Layout', 'Show ROI Labels', and 'Show Scene Labels' checkboxes. On the left, there are two panels: 'ROI Label Definition' with a list of labels (Car, Pedestrian, StopLight, Lane) and 'Scene Label Definition' with 'Current Frame' and 'Time Interval' radio buttons. The main workspace displays a video frame with bounding boxes for 'Car' and 'Pedestrian'. A context menu is open over the 'Algorithm:' dropdown, listing 'Point Tracker', 'Temporal Interpolator', and 'ACF Vehicle Detector'. At the bottom of this menu, 'Create New Algorithm' and 'Import Algorithm' are highlighted with a red box. A large yellow callout box at the bottom contains the text: 'Add custom automation algorithm driving.automation.AutomationAlgorithm'. The bottom status bar shows 'Start Time' (00.00000), 'Current' (09.00000), 'End Time', and 'Max Time'.

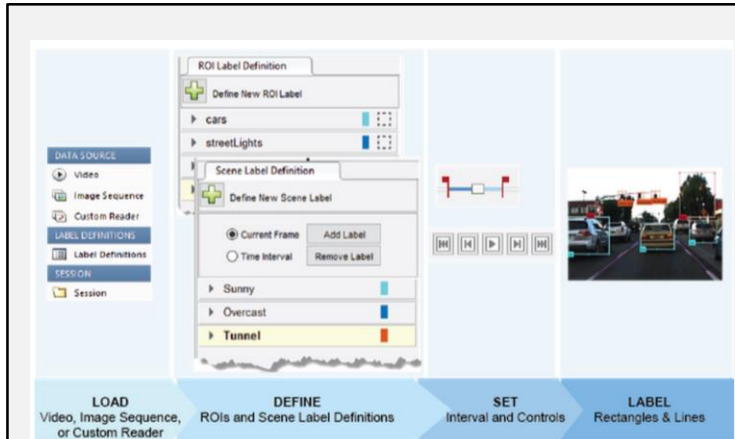
Customize Ground Truth Labeler App

The screenshot displays the Ground Truth Labeler application interface. On the left, the 'LABEL' menu includes options for Load, Save, Import Labels, and ROI. Below this are 'ROI Label Definition' and 'Scene Label Definition' sections, each with a 'Define new ROI label' or 'Define New Scene Label' button. A list of labels includes Car, Pedestrian, StopLight, and Lane. The central video frame shows a street scene with bounding boxes for a white car, a pedestrian, and a blue car, with labels 'Car', 'Pedestrian', and 'Lane' connected to the boxes. A timeline at the bottom shows 'Start Time' (00.00000), 'Current' (09.00000), 'End Time' (10.20000), and 'Max Time' (10.20000). On the right, a window titled 'Figure 1: Point Cloud Pla...' shows a 3D point cloud visualization of the scene with blue lines representing lane markings.

Add connection to other tools with **driving.connector.Connector**

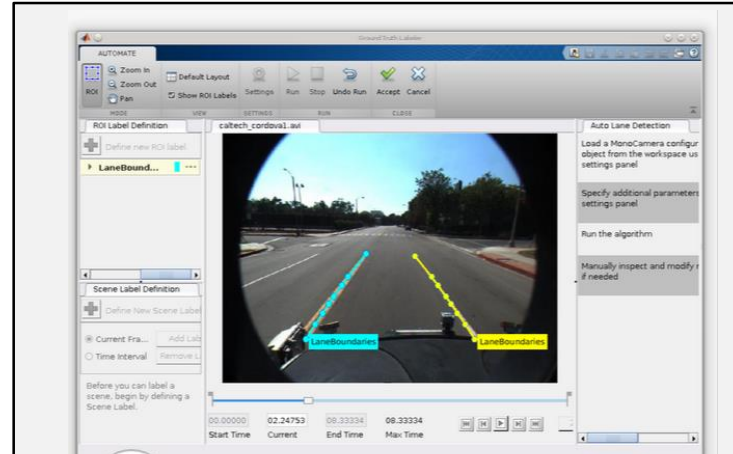
Learn more about detecting objects in images

by exploring examples in the Automated Driving System Toolbox



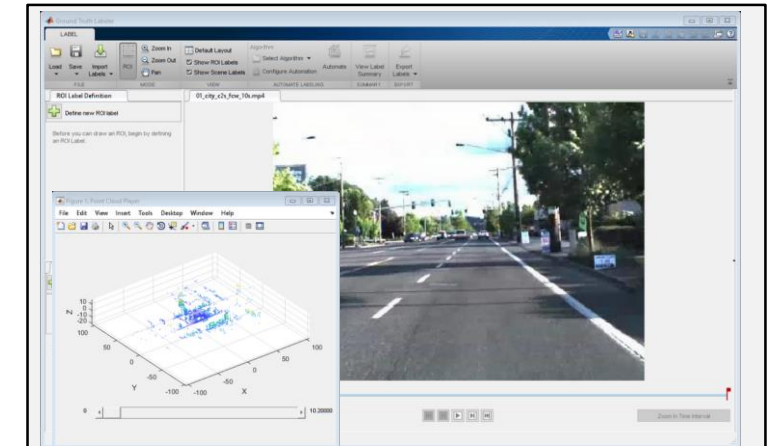
Define Ground Truth Data for Video or Image Sequences

- Label detections with Ground Truth Labeler App



Automate Ground Truth Labeling of Lane Boundaries

- Add automation algorithm for lane tracking

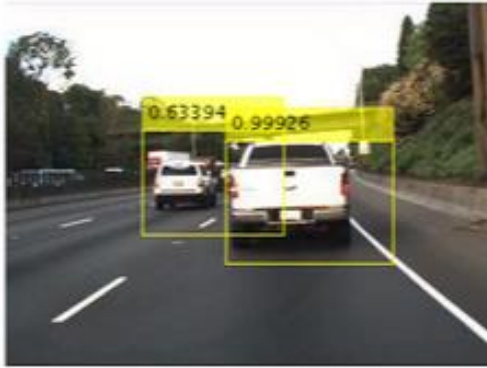


driving.connector.Connector class
Connect Lidar Display to Ground Truth Labeler

- 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

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



Track Pedestrians from a Moving Car

- **Explore pre-trained pedestrian detector**



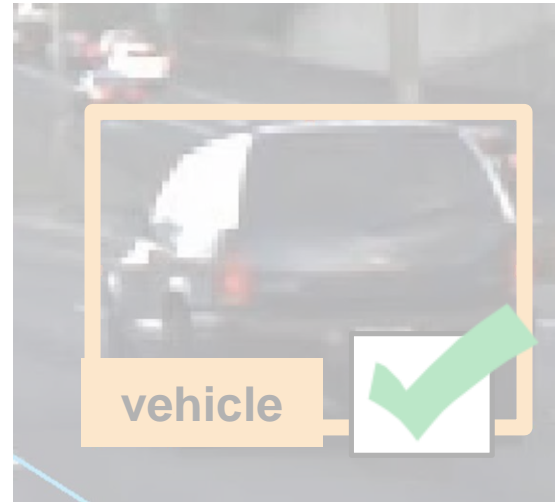
Visual Perception Using Monocular Camera

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

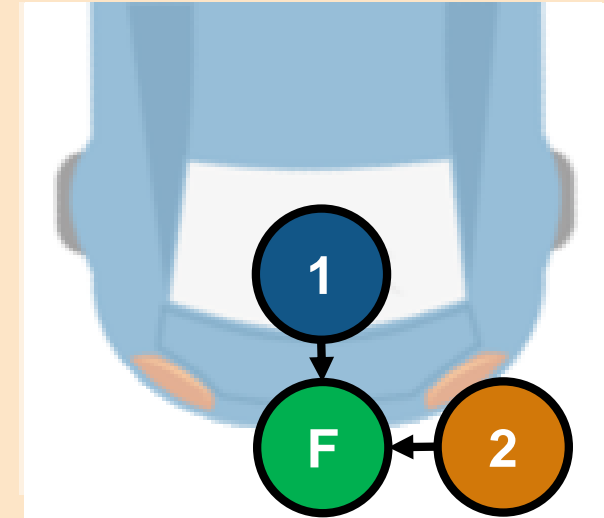
Some common questions from automated driving engineers



How can I
visualize vehicle
data?

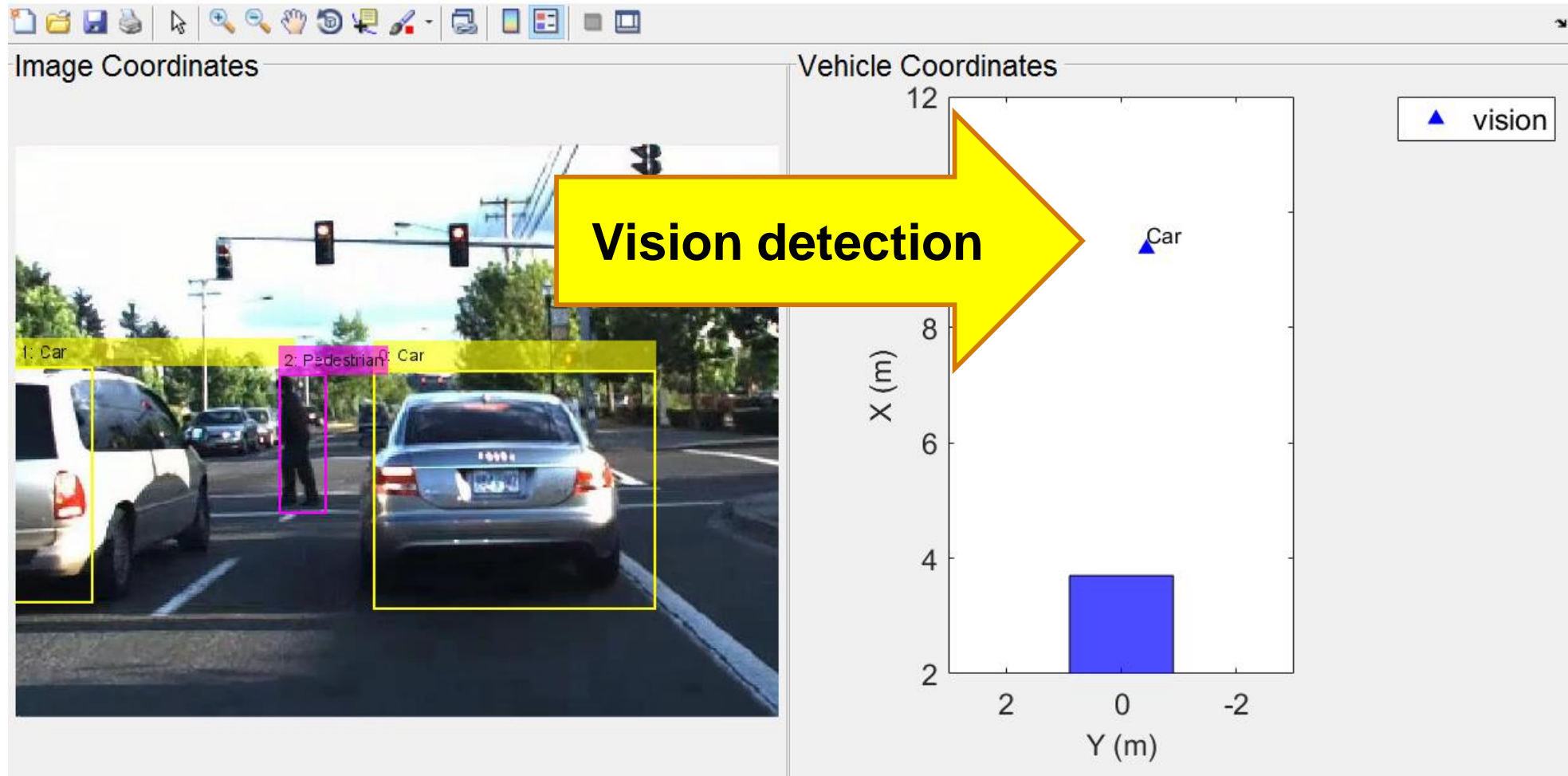


How can I
detect objects in
images?

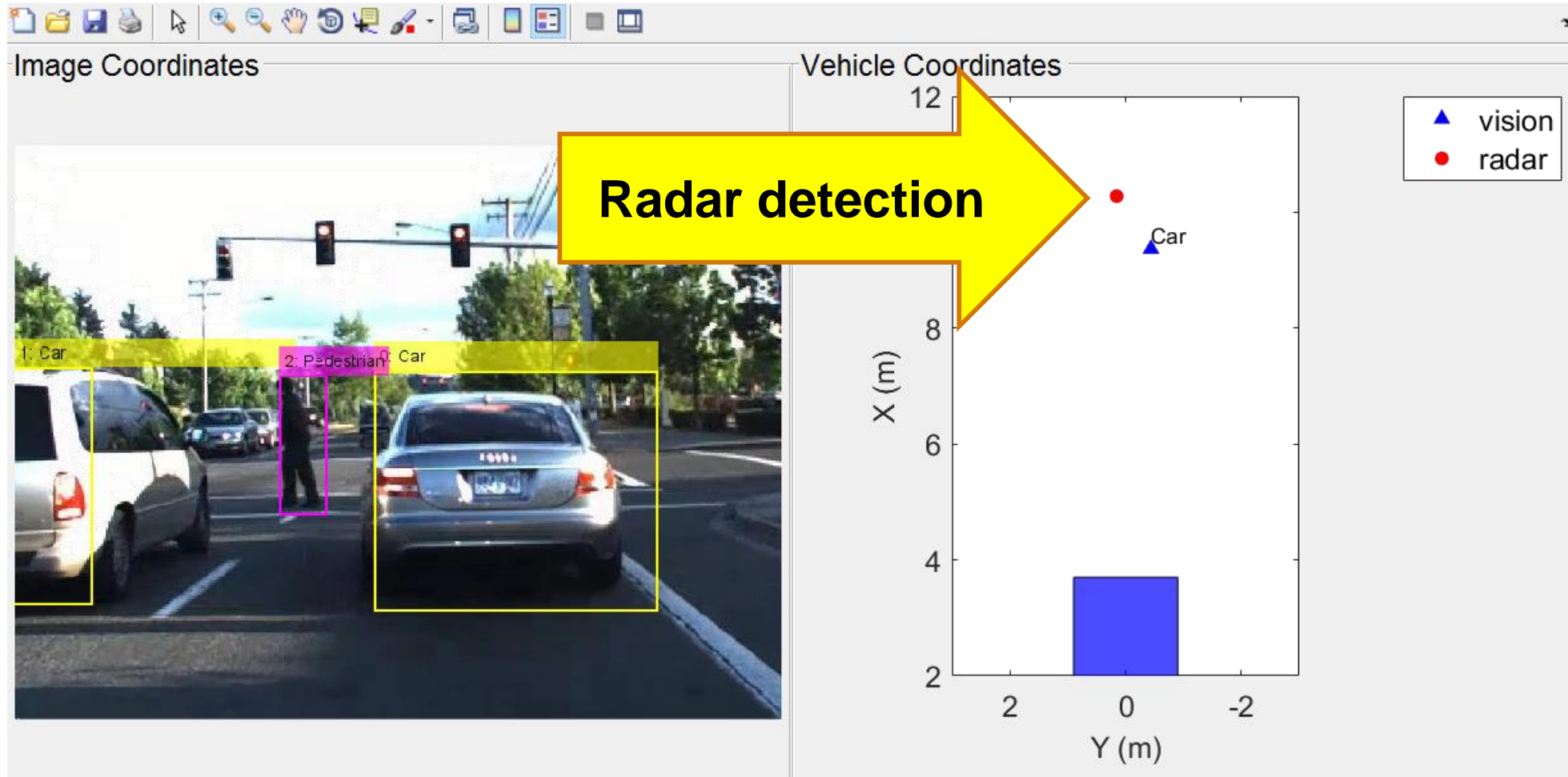


How can I
fuse multiple
detections?

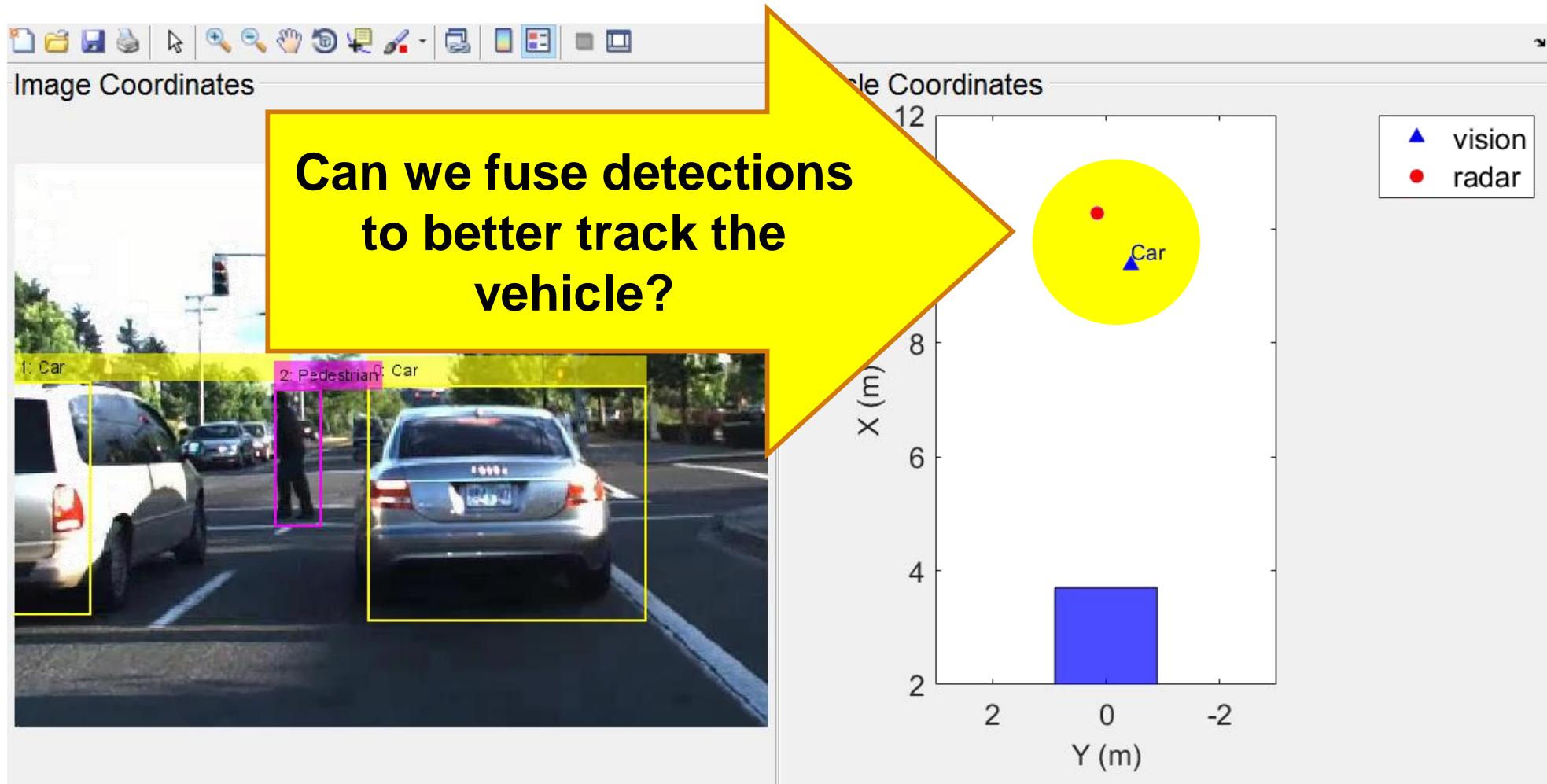
Example of radar and vision detections of a vehicle



Example of radar and vision detections of a vehicle

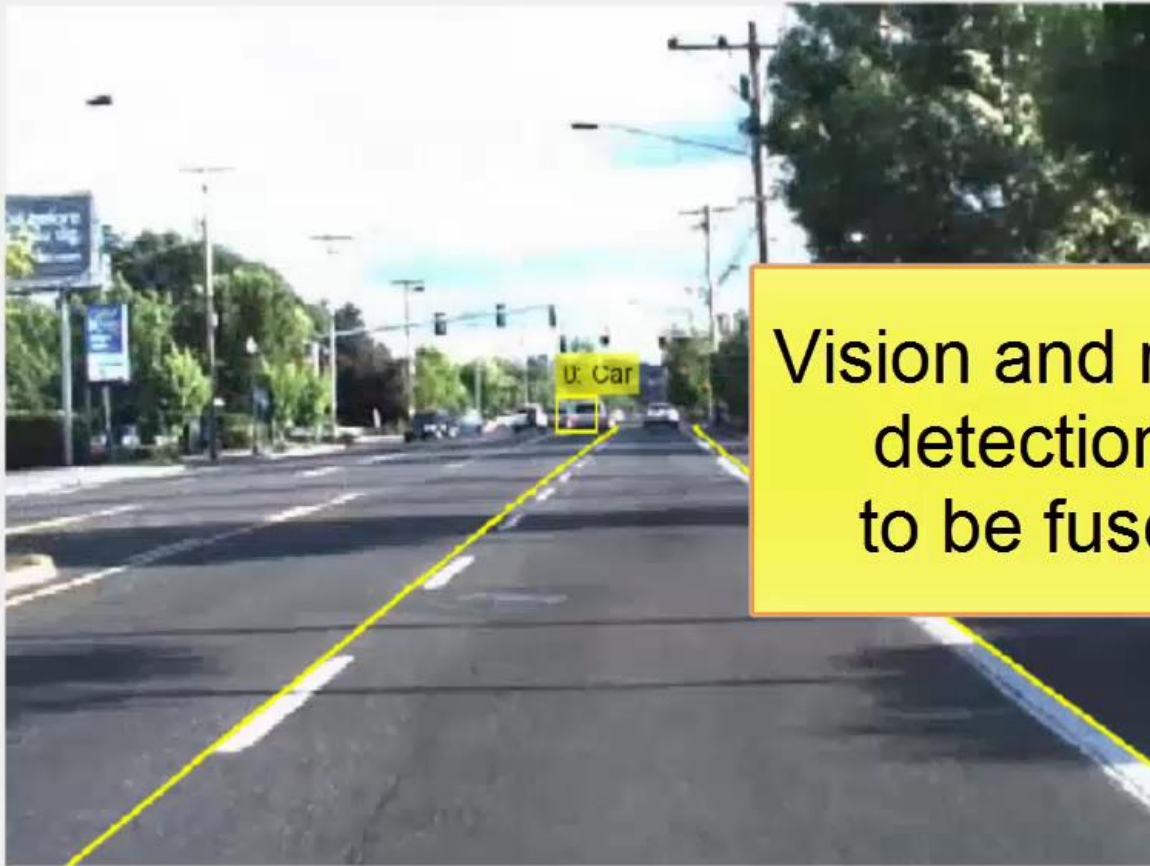


Example of radar and vision detections of a vehicle

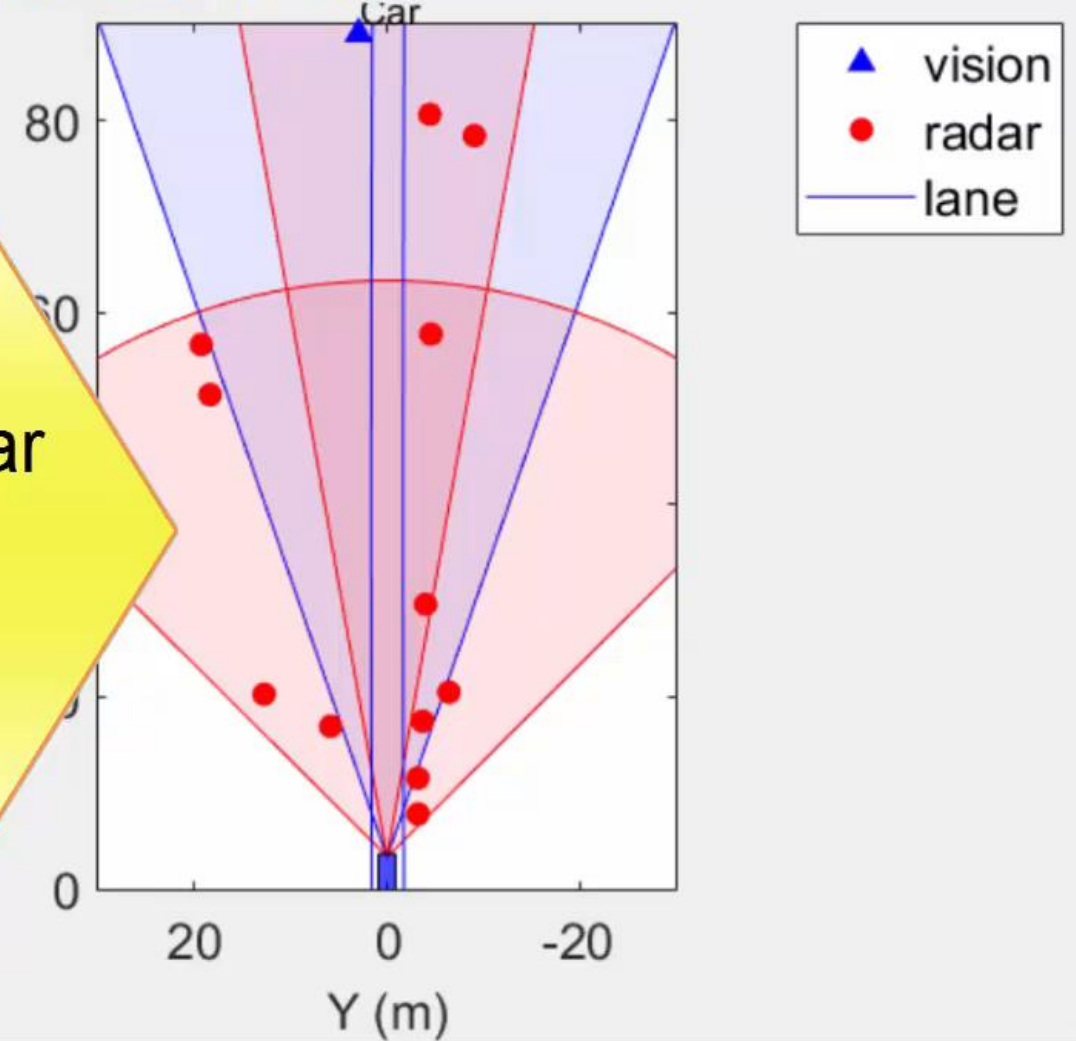


Fuse detections with multi-object tracker

Image Coordinates



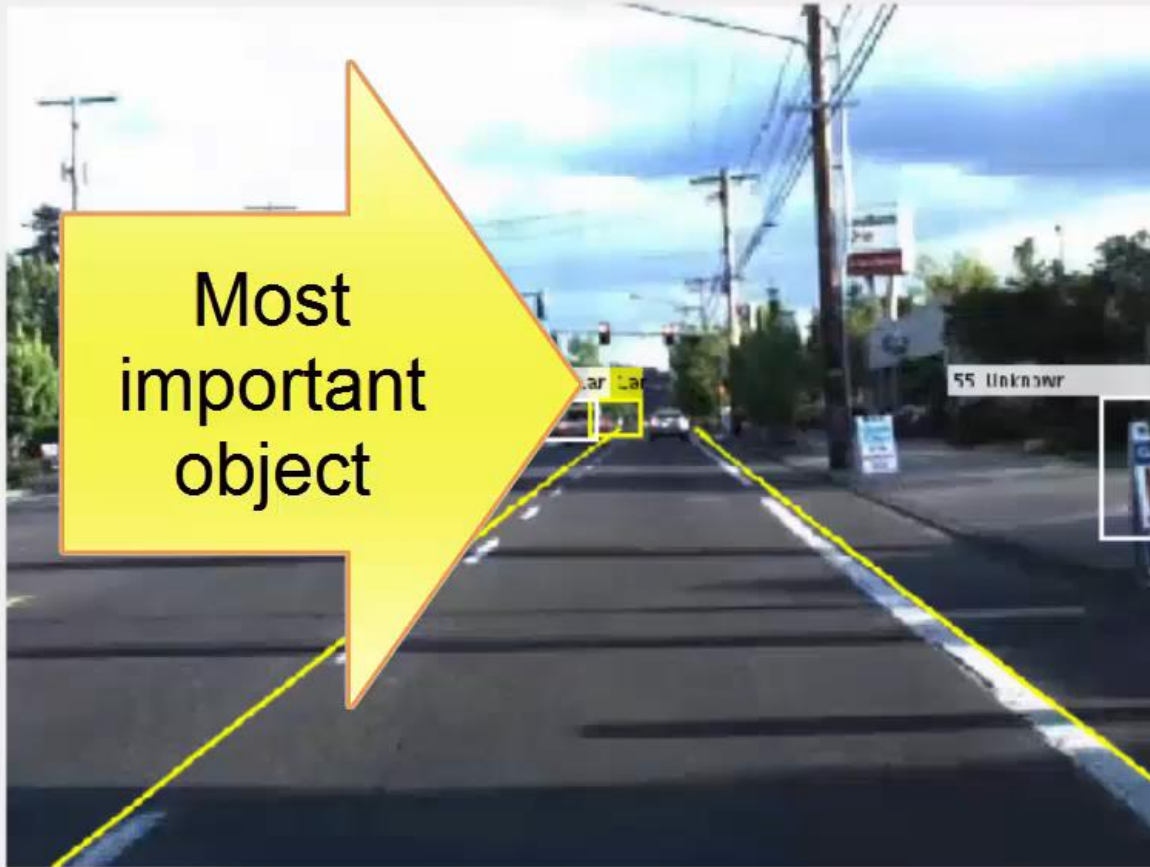
Vehicle Coordinates



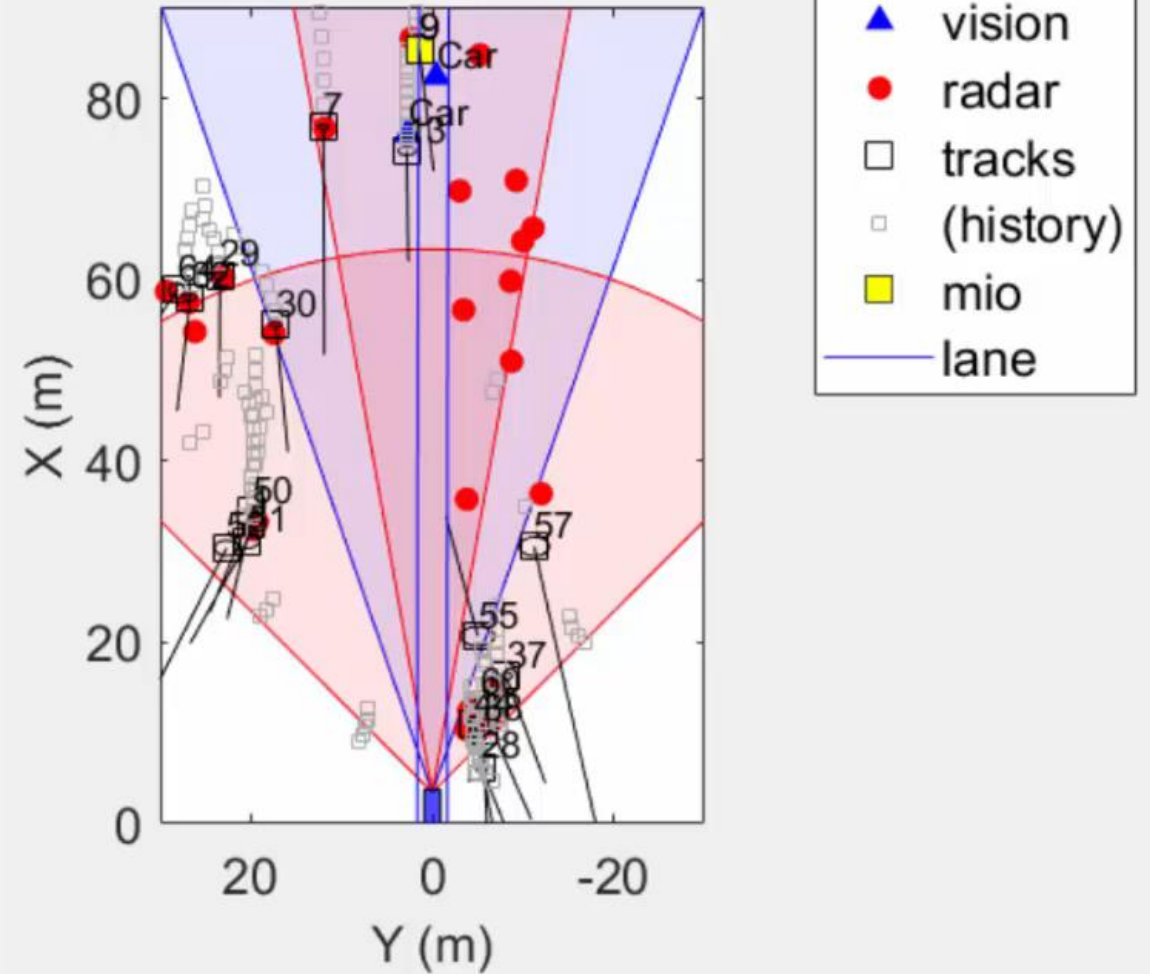
Vision and radar
detections
to be fused

Integrate tracker into higher level algorithm

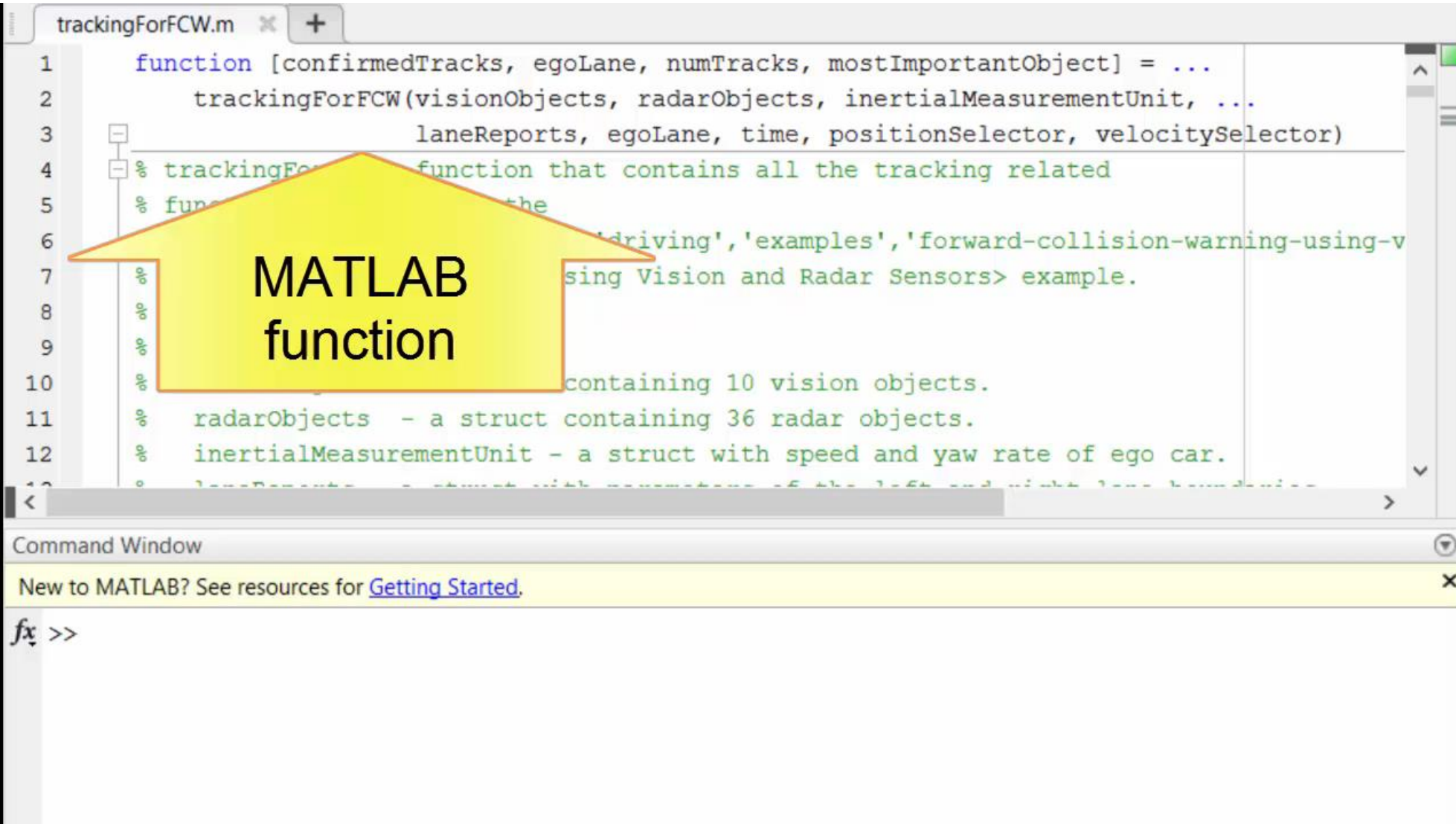
Image Coordinates



Vehicle Coordinates



Generate C code for algorithm



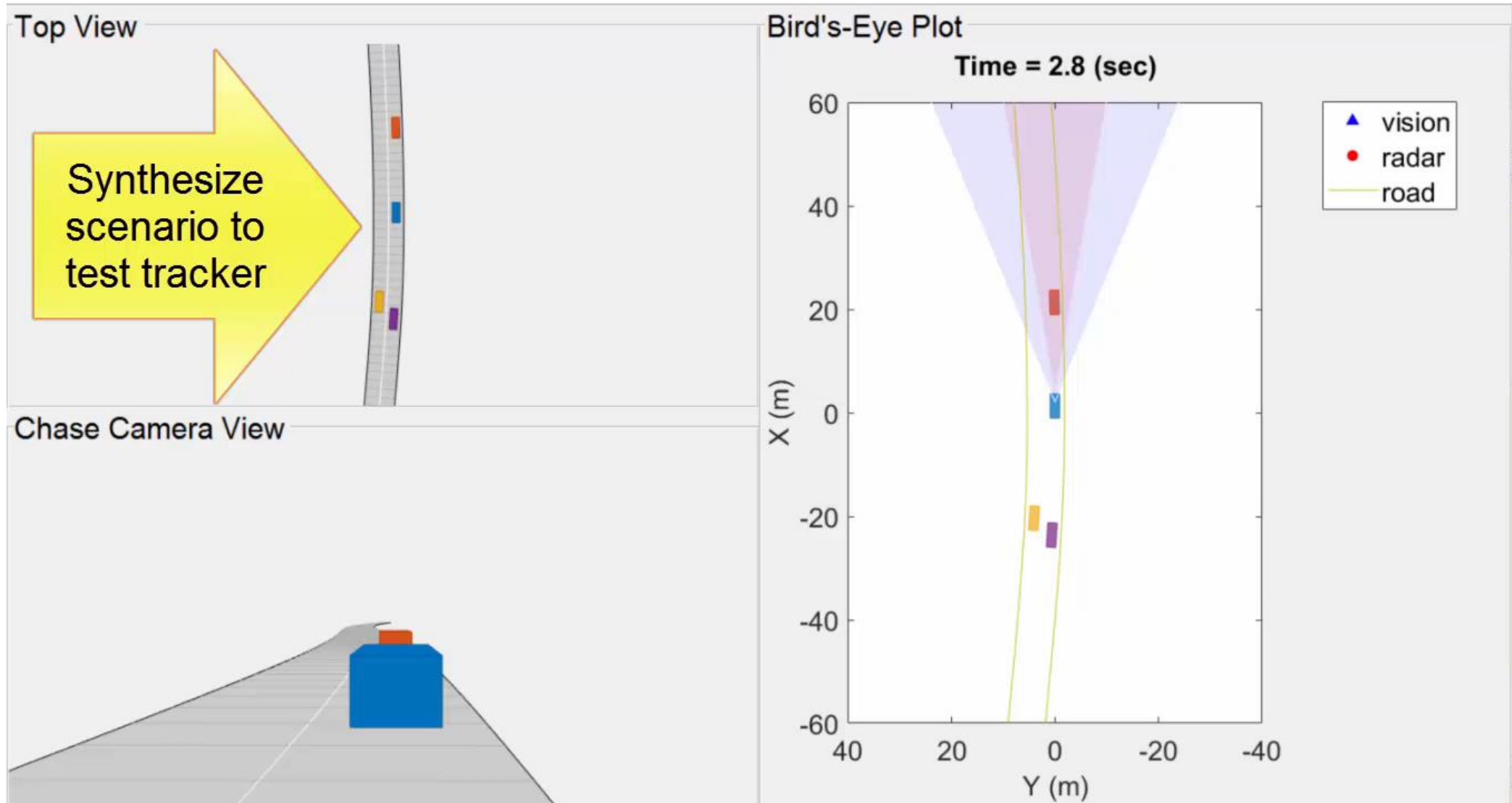
```
trackingForFCW.m x +
1 function [confirmedTracks, egoLane, numTracks, mostImportantObject] = ...
2 trackingForFCW(visionObjects, radarObjects, inertialMeasurementUnit, ...
3 laneReports, egoLane, time, positionSelector, velocitySelector)
4 % trackingForFCW function that contains all the tracking related
5 % function the
6 % 'driving', 'examples', 'forward-collision-warning-using-v
7 % sing Vision and Radar Sensors> example.
8 %
9 %
10 % containing 10 vision objects.
11 % radarObjects - a struct containing 36 radar objects.
12 % inertialMeasurementUnit - a struct with speed and yaw rate of ego car.
13 % laneReports - a struct with parameters of the left and right lane boundaries
14 %
15 %
16 %
17 %
18 %
19 %
20 %
21 %
22 %
23 %
24 %
25 %
26 %
27 %
28 %
29 %
30 %
31 %
32 %
33 %
34 %
35 %
36 %
37 %
38 %
39 %
40 %
41 %
42 %
43 %
44 %
45 %
46 %
47 %
48 %
49 %
50 %
51 %
52 %
53 %
54 %
55 %
56 %
57 %
58 %
59 %
60 %
61 %
62 %
63 %
64 %
65 %
66 %
67 %
68 %
69 %
70 %
71 %
72 %
73 %
74 %
75 %
76 %
77 %
78 %
79 %
80 %
81 %
82 %
83 %
84 %
85 %
86 %
87 %
88 %
89 %
90 %
91 %
92 %
93 %
94 %
95 %
96 %
97 %
98 %
99 %
100 %
101 %
102 %
103 %
104 %
105 %
106 %
107 %
108 %
109 %
110 %
111 %
112 %
113 %
114 %
115 %
116 %
117 %
118 %
119 %
120 %
121 %
122 %
123 %
124 %
125 %
126 %
127 %
128 %
129 %
130 %
131 %
132 %
133 %
134 %
135 %
136 %
137 %
138 %
139 %
140 %
141 %
142 %
143 %
144 %
145 %
146 %
147 %
148 %
149 %
150 %
151 %
152 %
153 %
154 %
155 %
156 %
157 %
158 %
159 %
160 %
161 %
162 %
163 %
164 %
165 %
166 %
167 %
168 %
169 %
170 %
171 %
172 %
173 %
174 %
175 %
176 %
177 %
178 %
179 %
180 %
181 %
182 %
183 %
184 %
185 %
186 %
187 %
188 %
189 %
190 %
191 %
192 %
193 %
194 %
195 %
196 %
197 %
198 %
199 %
200 %
201 %
202 %
203 %
204 %
205 %
206 %
207 %
208 %
209 %
210 %
211 %
212 %
213 %
214 %
215 %
216 %
217 %
218 %
219 %
220 %
221 %
222 %
223 %
224 %
225 %
226 %
227 %
228 %
229 %
230 %
231 %
232 %
233 %
234 %
235 %
236 %
237 %
238 %
239 %
240 %
241 %
242 %
243 %
244 %
245 %
246 %
247 %
248 %
249 %
250 %
251 %
252 %
253 %
254 %
255 %
256 %
257 %
258 %
259 %
260 %
261 %
262 %
263 %
264 %
265 %
266 %
267 %
268 %
269 %
270 %
271 %
272 %
273 %
274 %
275 %
276 %
277 %
278 %
279 %
280 %
281 %
282 %
283 %
284 %
285 %
286 %
287 %
288 %
289 %
290 %
291 %
292 %
293 %
294 %
295 %
296 %
297 %
298 %
299 %
300 %
301 %
302 %
303 %
304 %
305 %
306 %
307 %
308 %
309 %
310 %
311 %
312 %
313 %
314 %
315 %
316 %
317 %
318 %
319 %
320 %
321 %
322 %
323 %
324 %
325 %
326 %
327 %
328 %
329 %
330 %
331 %
332 %
333 %
334 %
335 %
336 %
337 %
338 %
339 %
340 %
341 %
342 %
343 %
344 %
345 %
346 %
347 %
348 %
349 %
350 %
351 %
352 %
353 %
354 %
355 %
356 %
357 %
358 %
359 %
360 %
361 %
362 %
363 %
364 %
365 %
366 %
367 %
368 %
369 %
370 %
371 %
372 %
373 %
374 %
375 %
376 %
377 %
378 %
379 %
380 %
381 %
382 %
383 %
384 %
385 %
386 %
387 %
388 %
389 %
390 %
391 %
392 %
393 %
394 %
395 %
396 %
397 %
398 %
399 %
400 %
401 %
402 %
403 %
404 %
405 %
406 %
407 %
408 %
409 %
410 %
411 %
412 %
413 %
414 %
415 %
416 %
417 %
418 %
419 %
420 %
421 %
422 %
423 %
424 %
425 %
426 %
427 %
428 %
429 %
430 %
431 %
432 %
433 %
434 %
435 %
436 %
437 %
438 %
439 %
440 %
441 %
442 %
443 %
444 %
445 %
446 %
447 %
448 %
449 %
450 %
451 %
452 %
453 %
454 %
455 %
456 %
457 %
458 %
459 %
460 %
461 %
462 %
463 %
464 %
465 %
466 %
467 %
468 %
469 %
470 %
471 %
472 %
473 %
474 %
475 %
476 %
477 %
478 %
479 %
480 %
481 %
482 %
483 %
484 %
485 %
486 %
487 %
488 %
489 %
490 %
491 %
492 %
493 %
494 %
495 %
496 %
497 %
498 %
499 %
500 %
501 %
502 %
503 %
504 %
505 %
506 %
507 %
508 %
509 %
510 %
511 %
512 %
513 %
514 %
515 %
516 %
517 %
518 %
519 %
520 %
521 %
522 %
523 %
524 %
525 %
526 %
527 %
528 %
529 %
530 %
531 %
532 %
533 %
534 %
535 %
536 %
537 %
538 %
539 %
540 %
541 %
542 %
543 %
544 %
545 %
546 %
547 %
548 %
549 %
550 %
551 %
552 %
553 %
554 %
555 %
556 %
557 %
558 %
559 %
560 %
561 %
562 %
563 %
564 %
565 %
566 %
567 %
568 %
569 %
570 %
571 %
572 %
573 %
574 %
575 %
576 %
577 %
578 %
579 %
580 %
581 %
582 %
583 %
584 %
585 %
586 %
587 %
588 %
589 %
590 %
591 %
592 %
593 %
594 %
595 %
596 %
597 %
598 %
599 %
600 %
601 %
602 %
603 %
604 %
605 %
606 %
607 %
608 %
609 %
610 %
611 %
612 %
613 %
614 %
615 %
616 %
617 %
618 %
619 %
620 %
621 %
622 %
623 %
624 %
625 %
626 %
627 %
628 %
629 %
630 %
631 %
632 %
633 %
634 %
635 %
636 %
637 %
638 %
639 %
640 %
641 %
642 %
643 %
644 %
645 %
646 %
647 %
648 %
649 %
650 %
651 %
652 %
653 %
654 %
655 %
656 %
657 %
658 %
659 %
660 %
661 %
662 %
663 %
664 %
665 %
666 %
667 %
668 %
669 %
670 %
671 %
672 %
673 %
674 %
675 %
676 %
677 %
678 %
679 %
680 %
681 %
682 %
683 %
684 %
685 %
686 %
687 %
688 %
689 %
690 %
691 %
692 %
693 %
694 %
695 %
696 %
697 %
698 %
699 %
700 %
701 %
702 %
703 %
704 %
705 %
706 %
707 %
708 %
709 %
710 %
711 %
712 %
713 %
714 %
715 %
716 %
717 %
718 %
719 %
720 %
721 %
722 %
723 %
724 %
725 %
726 %
727 %
728 %
729 %
730 %
731 %
732 %
733 %
734 %
735 %
736 %
737 %
738 %
739 %
740 %
741 %
742 %
743 %
744 %
745 %
746 %
747 %
748 %
749 %
750 %
751 %
752 %
753 %
754 %
755 %
756 %
757 %
758 %
759 %
760 %
761 %
762 %
763 %
764 %
765 %
766 %
767 %
768 %
769 %
770 %
771 %
772 %
773 %
774 %
775 %
776 %
777 %
778 %
779 %
780 %
781 %
782 %
783 %
784 %
785 %
786 %
787 %
788 %
789 %
790 %
791 %
792 %
793 %
794 %
795 %
796 %
797 %
798 %
799 %
800 %
801 %
802 %
803 %
804 %
805 %
806 %
807 %
808 %
809 %
810 %
811 %
812 %
813 %
814 %
815 %
816 %
817 %
818 %
819 %
820 %
821 %
822 %
823 %
824 %
825 %
826 %
827 %
828 %
829 %
830 %
831 %
832 %
833 %
834 %
835 %
836 %
837 %
838 %
839 %
840 %
841 %
842 %
843 %
844 %
845 %
846 %
847 %
848 %
849 %
850 %
851 %
852 %
853 %
854 %
855 %
856 %
857 %
858 %
859 %
860 %
861 %
862 %
863 %
864 %
865 %
866 %
867 %
868 %
869 %
870 %
871 %
872 %
873 %
874 %
875 %
876 %
877 %
878 %
879 %
880 %
881 %
882 %
883 %
884 %
885 %
886 %
887 %
888 %
889 %
890 %
891 %
892 %
893 %
894 %
895 %
896 %
897 %
898 %
899 %
900 %
901 %
902 %
903 %
904 %
905 %
906 %
907 %
908 %
909 %
910 %
911 %
912 %
913 %
914 %
915 %
916 %
917 %
918 %
919 %
920 %
921 %
922 %
923 %
924 %
925 %
926 %
927 %
928 %
929 %
930 %
931 %
932 %
933 %
934 %
935 %
936 %
937 %
938 %
939 %
940 %
941 %
942 %
943 %
944 %
945 %
946 %
947 %
948 %
949 %
950 %
951 %
952 %
953 %
954 %
955 %
956 %
957 %
958 %
959 %
960 %
961 %
962 %
963 %
964 %
965 %
966 %
967 %
968 %
969 %
970 %
971 %
972 %
973 %
974 %
975 %
976 %
977 %
978 %
979 %
980 %
981 %
982 %
983 %
984 %
985 %
986 %
987 %
988 %
989 %
990 %
991 %
992 %
993 %
994 %
995 %
996 %
997 %
998 %
999 %
1000 %
```

Command Window

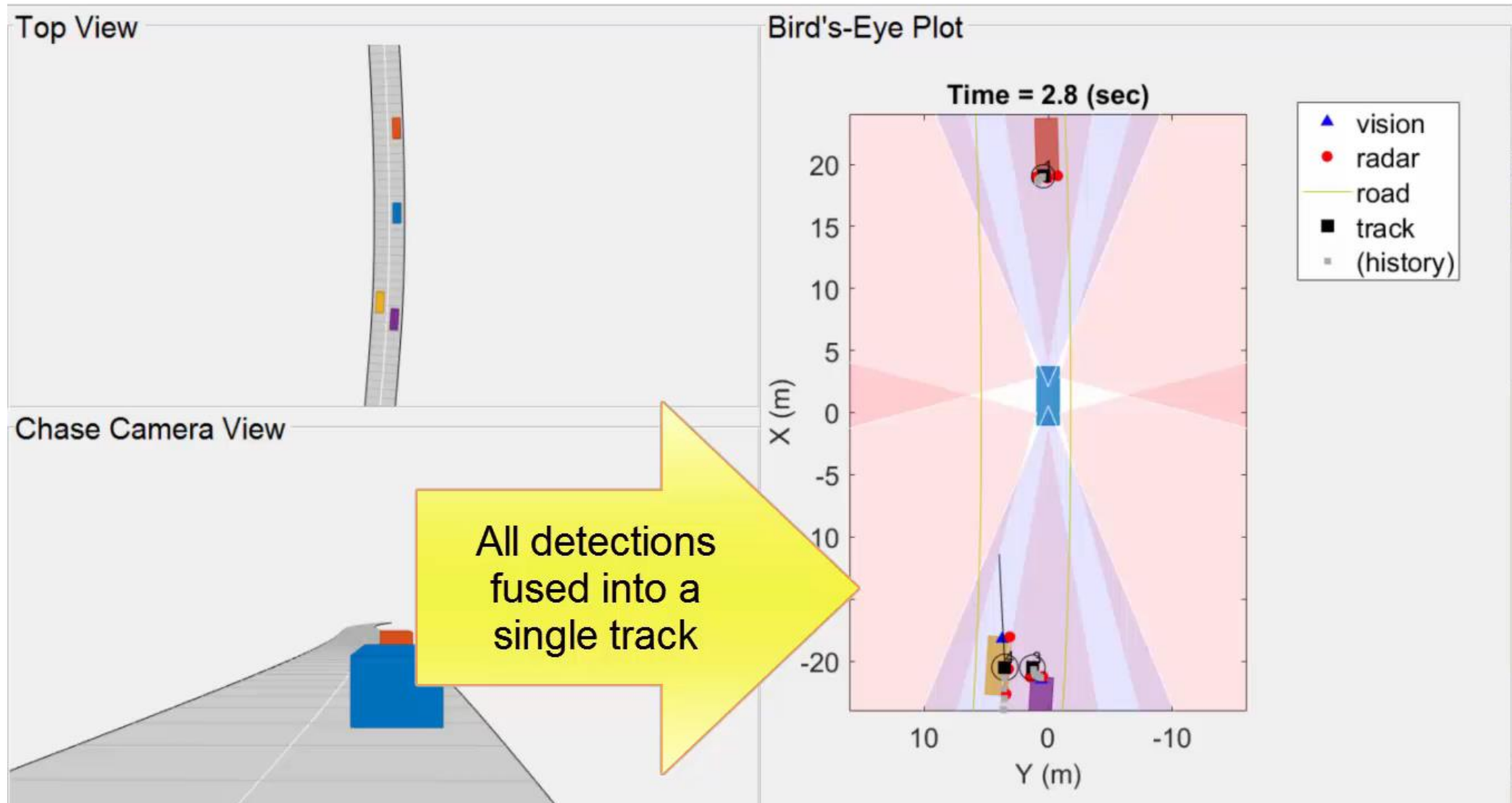
New to MATLAB? See resources for [Getting Started](#).

fx >>

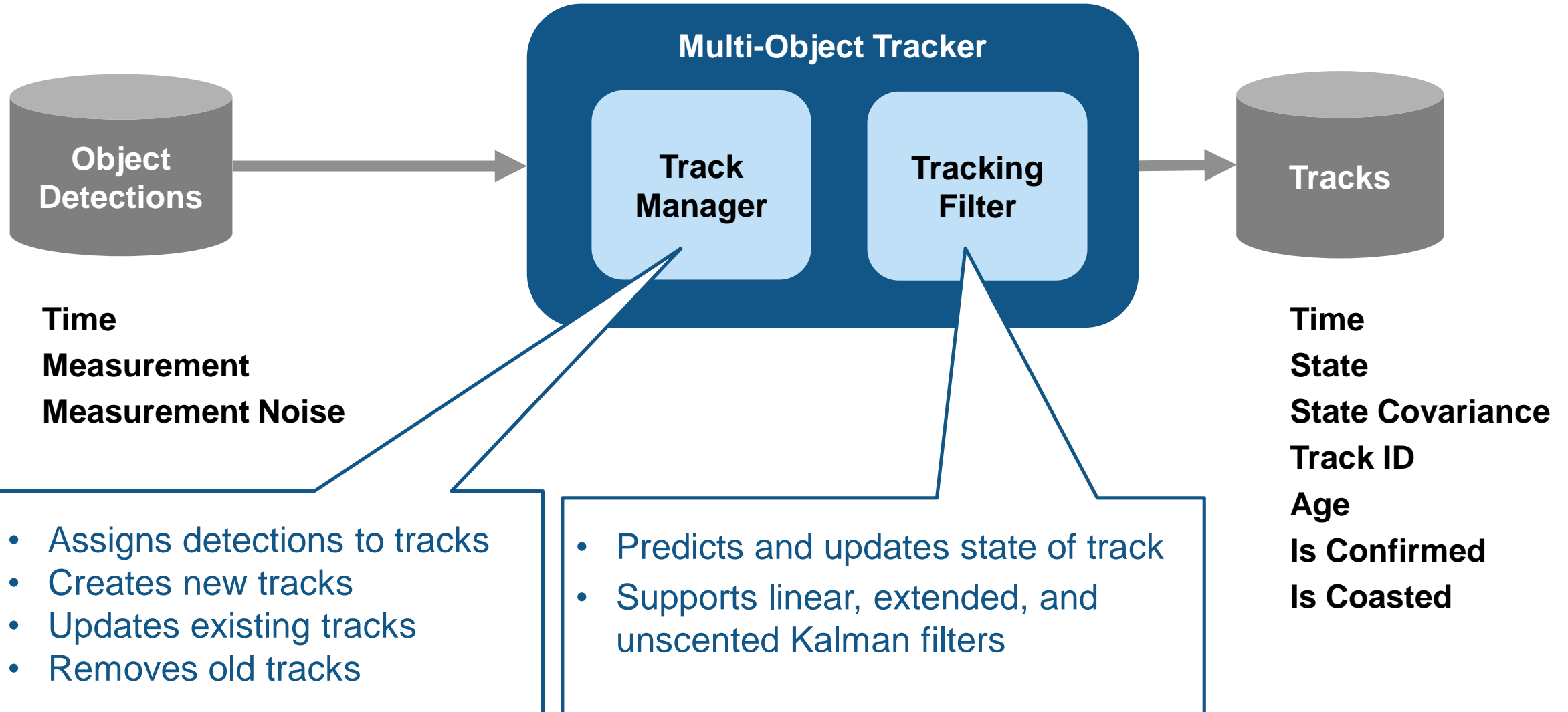
Synthesize scenario to test tracker



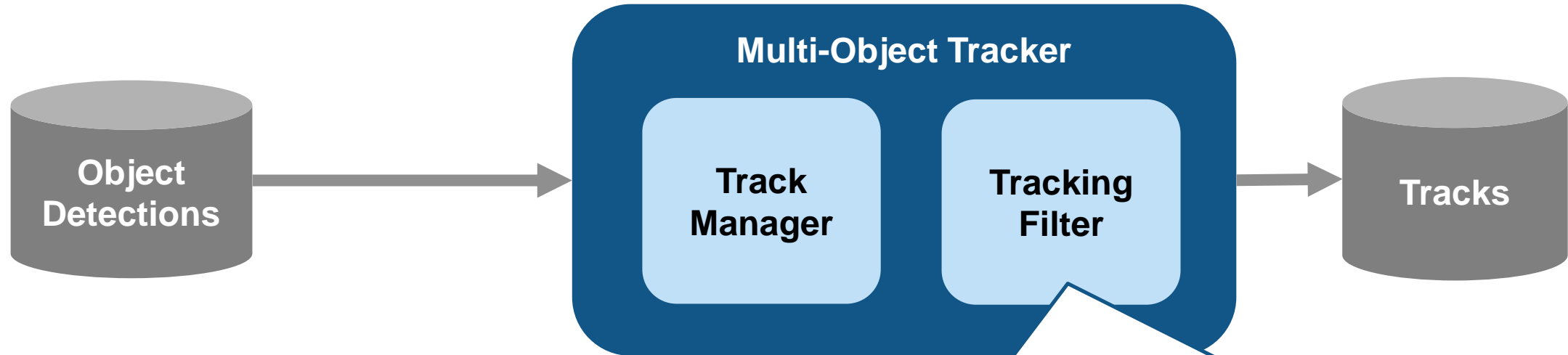
Test tracker against synthesized data



Track multiple object detections

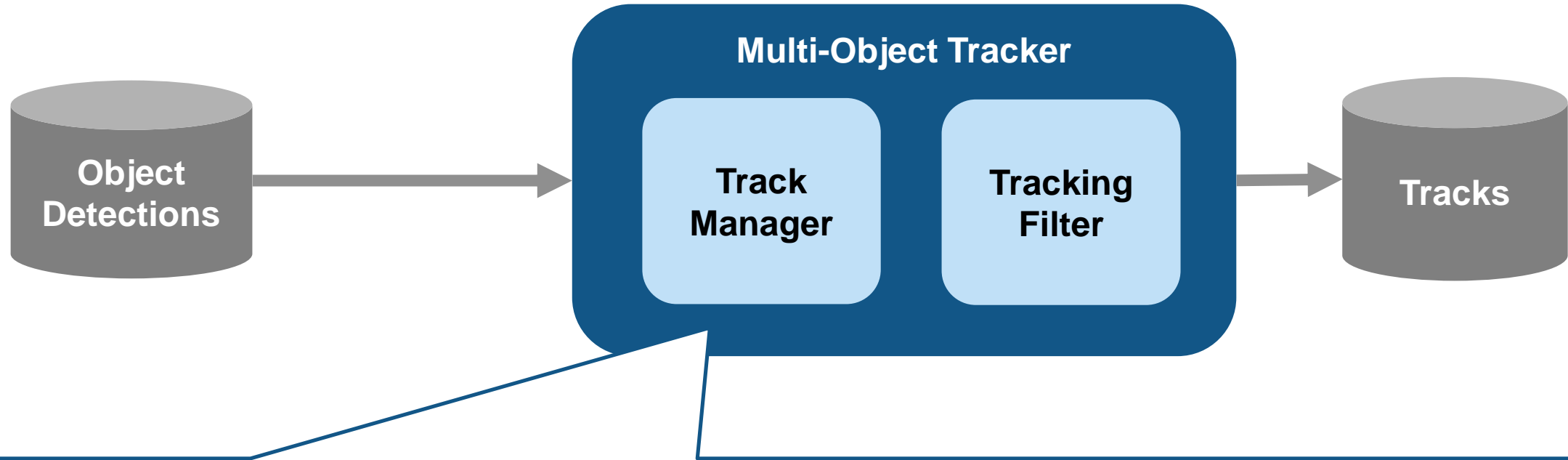


Examples of Kalman Filter (KF) initialization functions



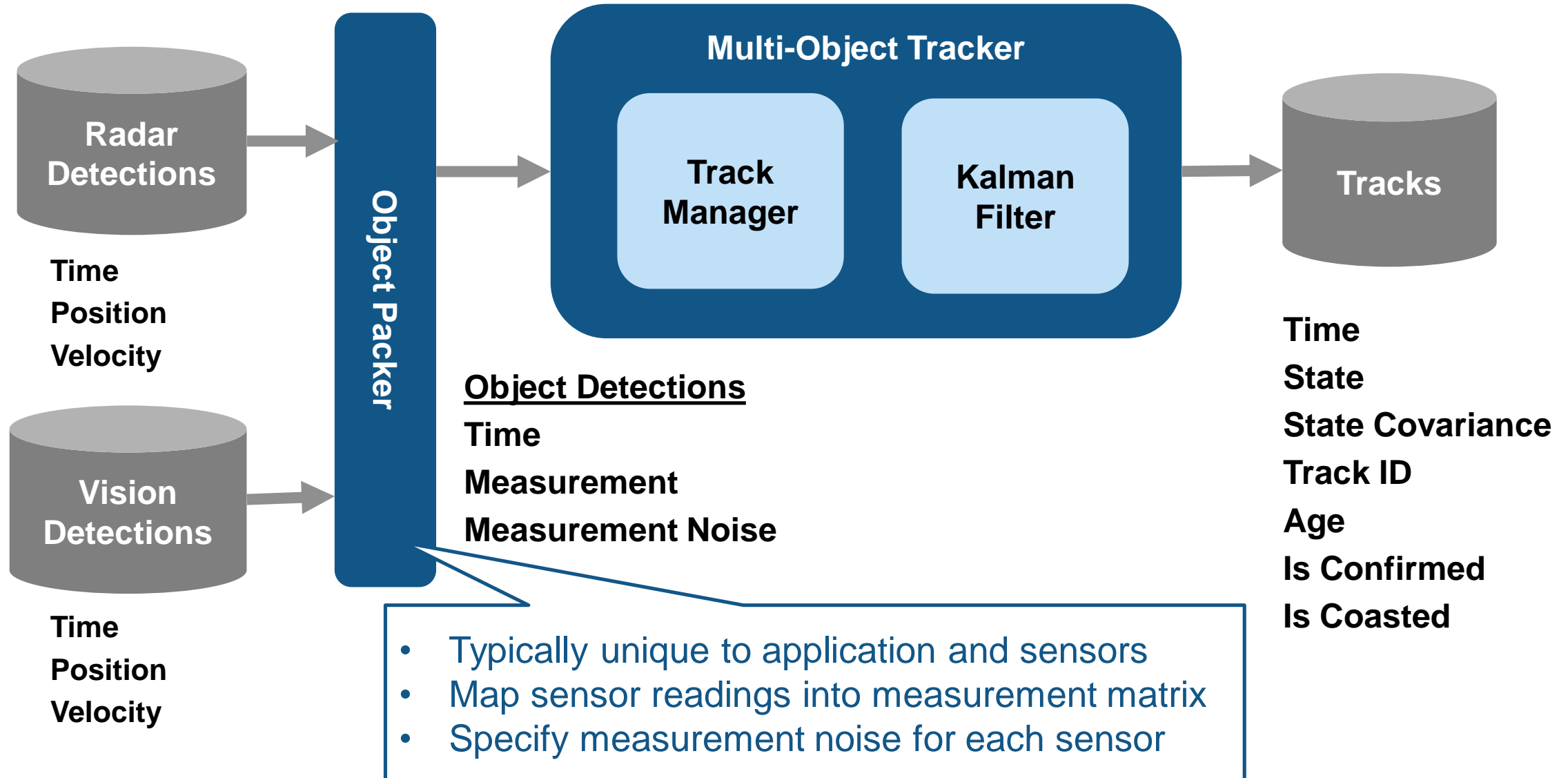
	Linear KF (trackingKF)	Extended KF (trackingEKF)	Unscented KF (trackingUKF)
Constant velocity	<code>initcvkf</code>	<code>initcvekf</code>	<code>initcvukf</code>
Constant acceleration	<code>initcakf</code>	<code>initcaekf</code>	<code>initcaukf</code>
Constant turn	Not applicable	<code>initctekf</code>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

Example of configuring multi-object tracker

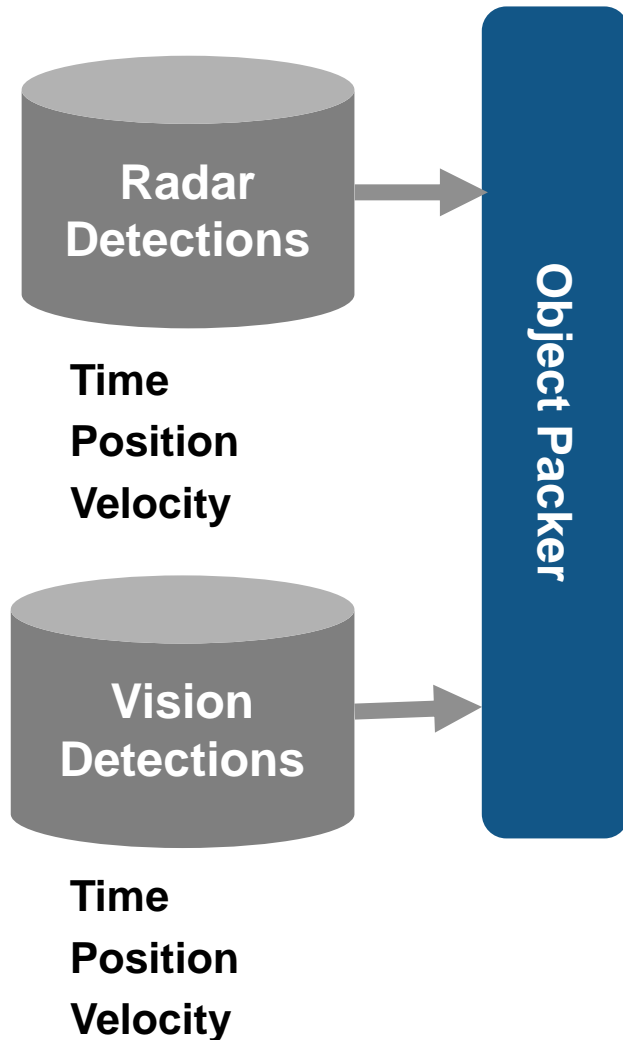


```
tracker = multiObjectTracker(...
    'FilterInitializationFcn', @initcaekf, ... % Handle to tracking Kalman filter
    'AssignmentThreshold',    35,... % Normalized distance from track for assignment
    'ConfirmationParameters', [2 ... % Min number of assignments for confirmation
                               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 object detection format



```
% Example sensor data
```

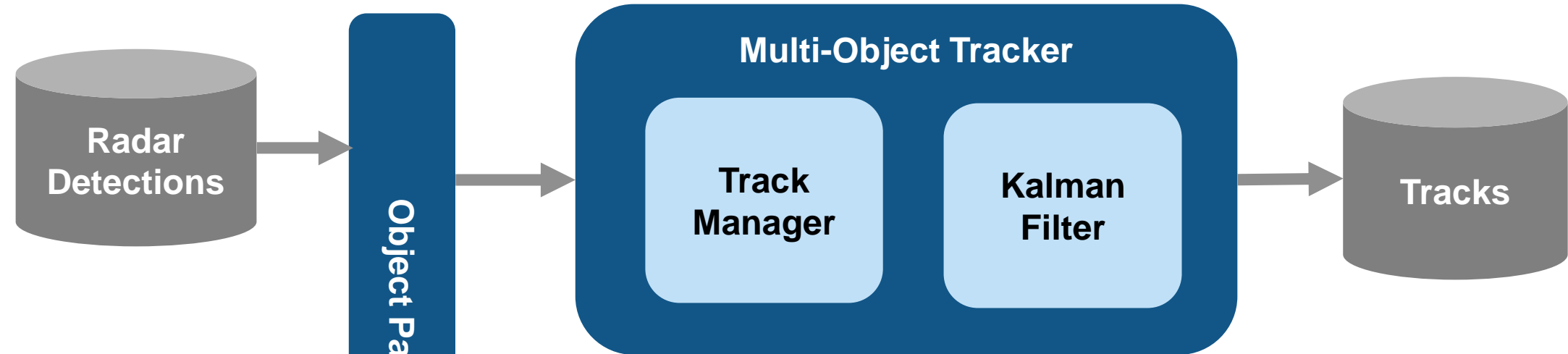
```
radar.Time      = 0.049;      % (sec)
radar.Position  = [10 0.5];  % [x y] (m)
radar.Velocity  = [4.4 0.1]; % [x y] (m/sec)
vision.Time     = 0.051;     % (sec)
vision.Position = [11 0.1];  % [x y] (m)
vision.Velocity = 4.1;       % [x]   (m/sec)
```

```
% 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]));
```

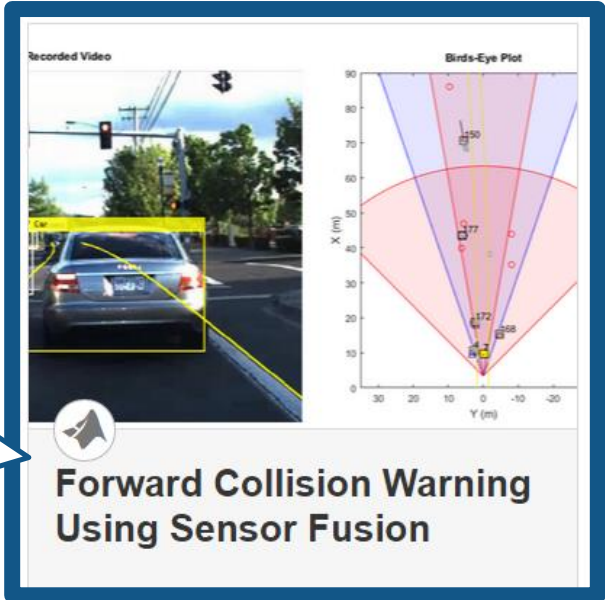
```
packedDetections(2) = objectDetection(vision.Time, ...
    [vision.Position(1); vision.Velocity(1); ...
    vision.Position(2); 0], ...
    'MeasurementNoise', diag([1,1,2,10]));
```


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



Generate C code for algorithm with MATLAB Coder

```
trackingForFCW_kernel.m x +
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], mxArray_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

```

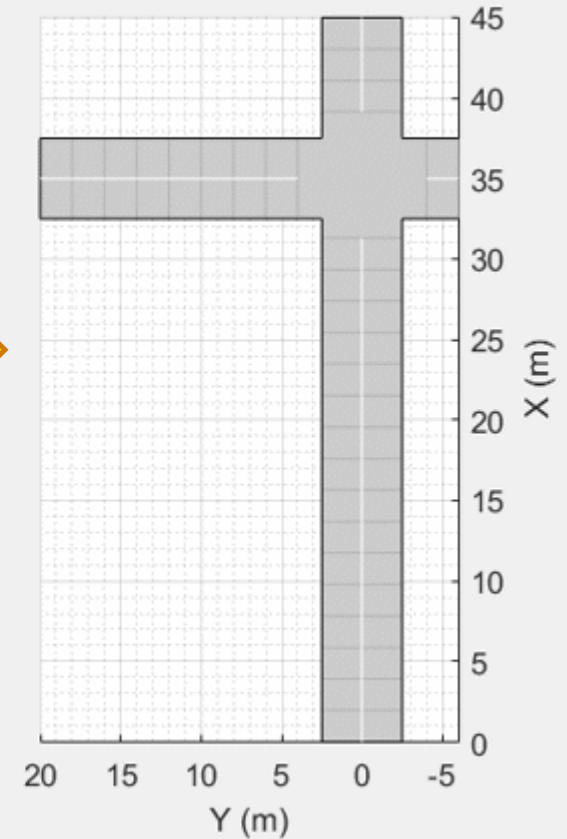
%% Create a new scenario
s = drivingScenario('SampleTime', 0.05);

%% 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];

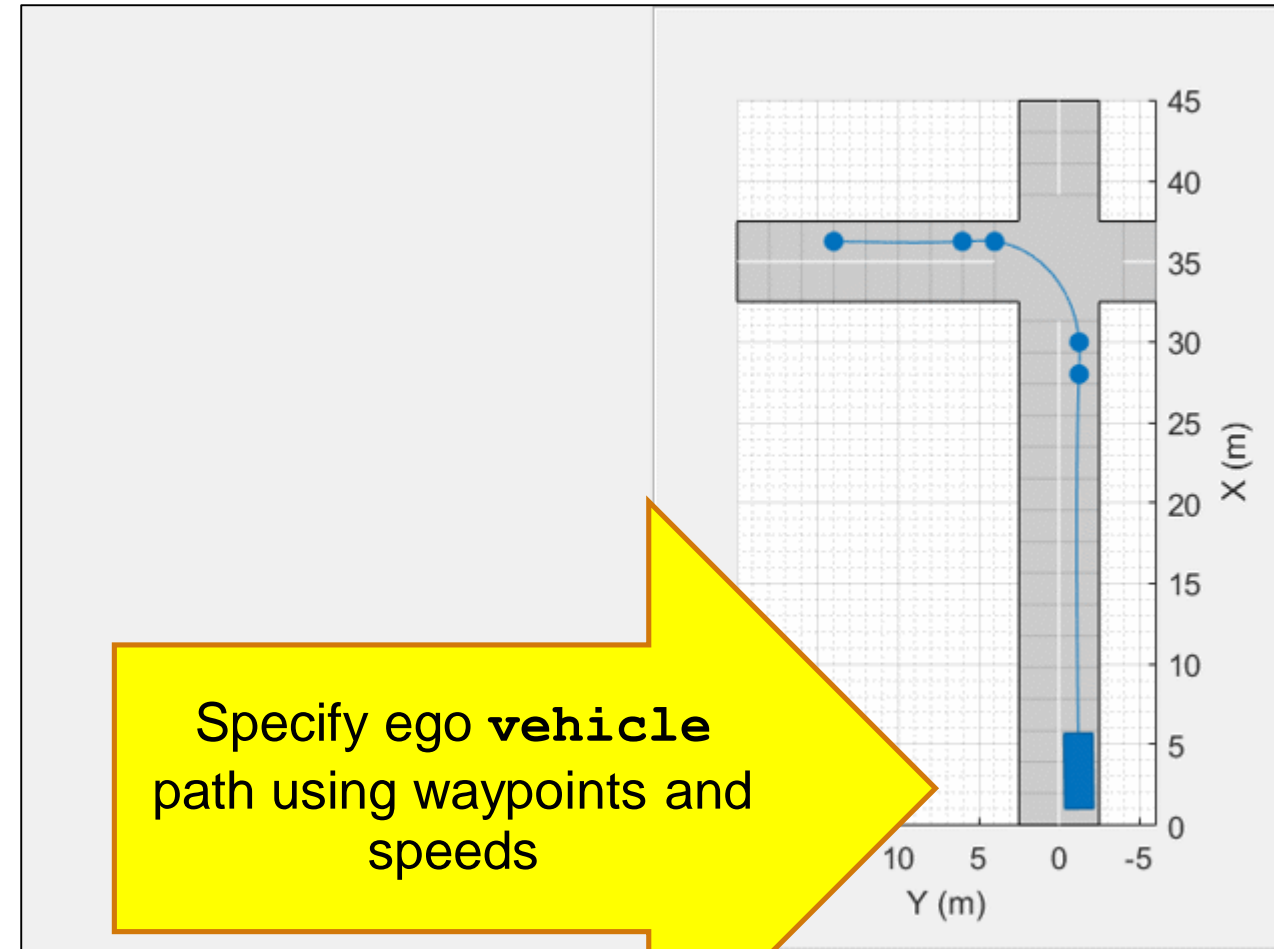
```

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);
```

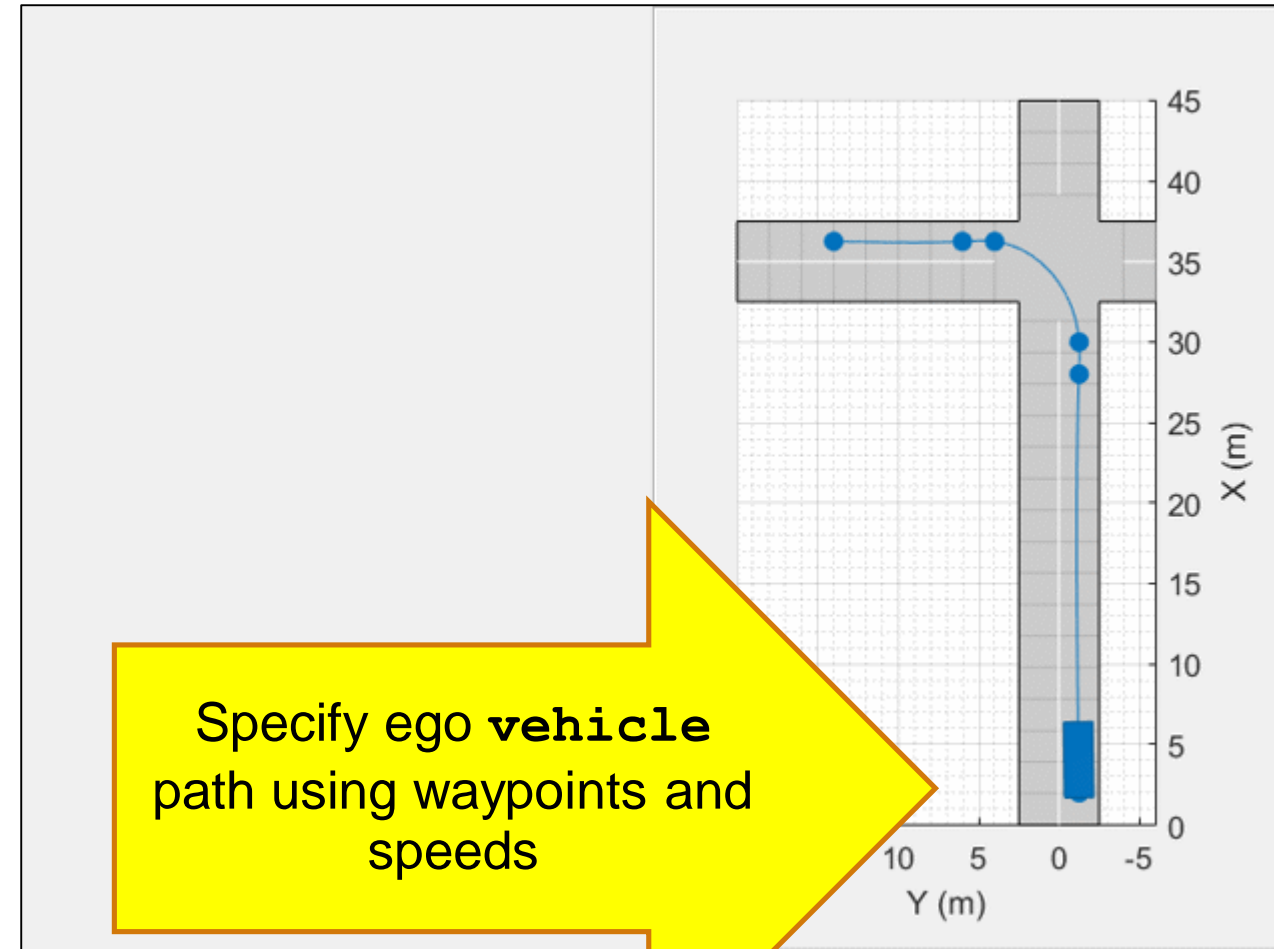


Add ego vehicle

```
%% 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
```



Add target vehicle and pedestrian actor

```

%% Add child pedestrian actor
child = actor(s, 'Length',0.24,...
               'Width',0.45,...
               'Height',1.7,...
               'Position',[40 -5 0],...
               'Yaw',180);

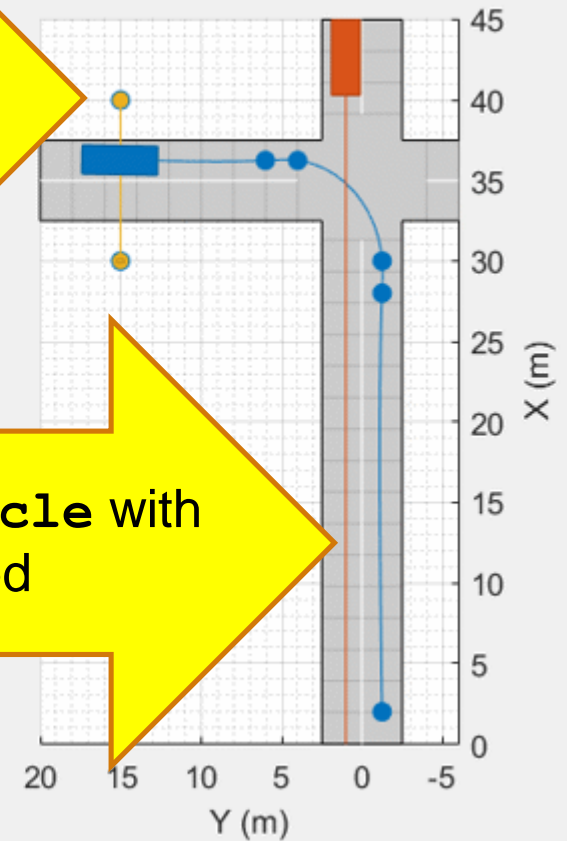
path(child,...
      [30 15; 40 15],... % Waypoints (m)
      1.39); % Speed (m/s) = 5 km/hr

%% Add Target vehicle
targetVehicle = vehicle(s);
path(targetVehicle,...
      [44 1; -4 1],... % Waypoints (m)
      [5 ; 14]); % Speeds (m/s)

```

Specify pedestrian actor size and path

Specify target vehicle with varying speed

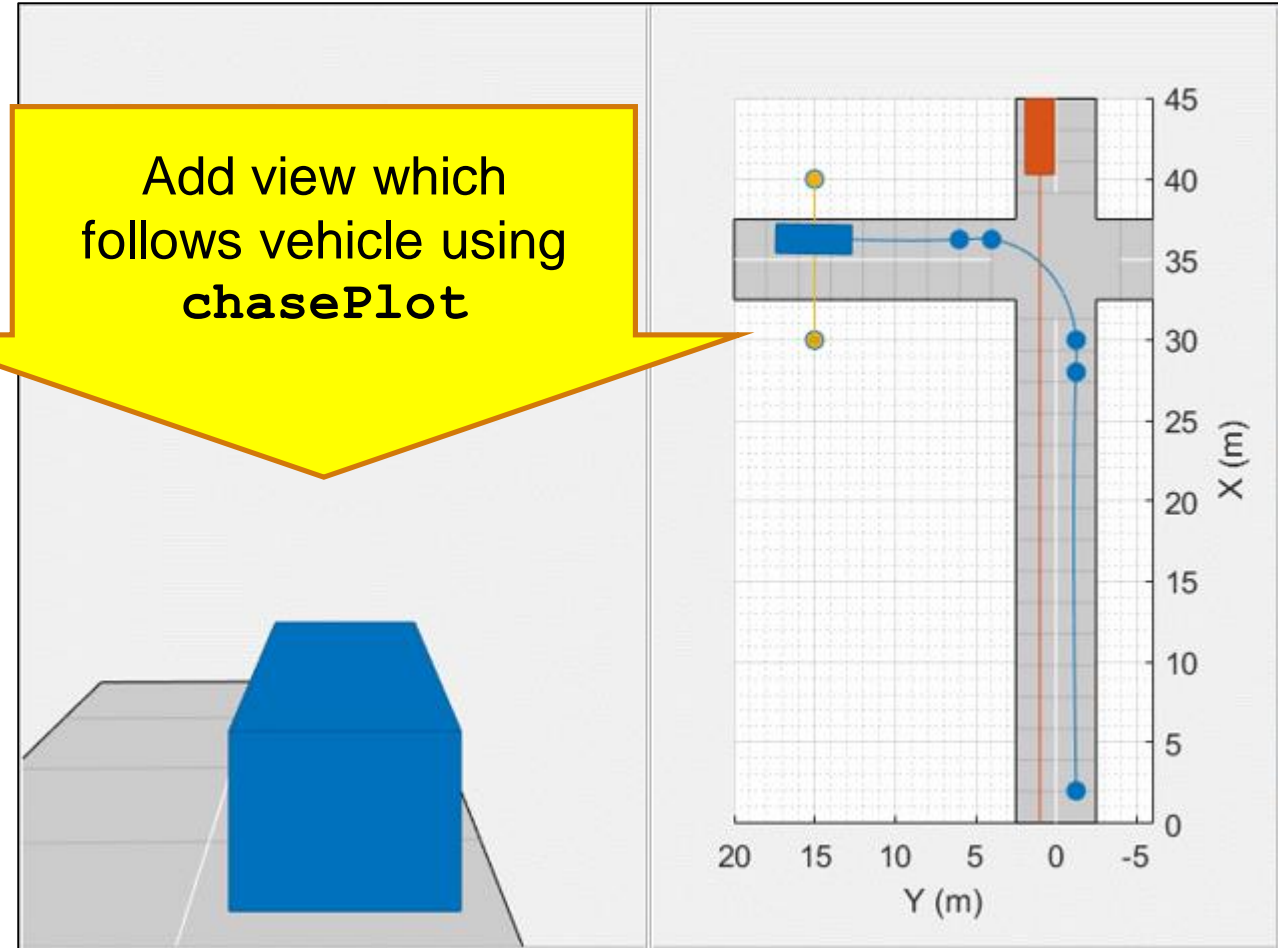


View scenario from behind ego vehicle

```

%% 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)

```



View scenario from behind ego vehicle

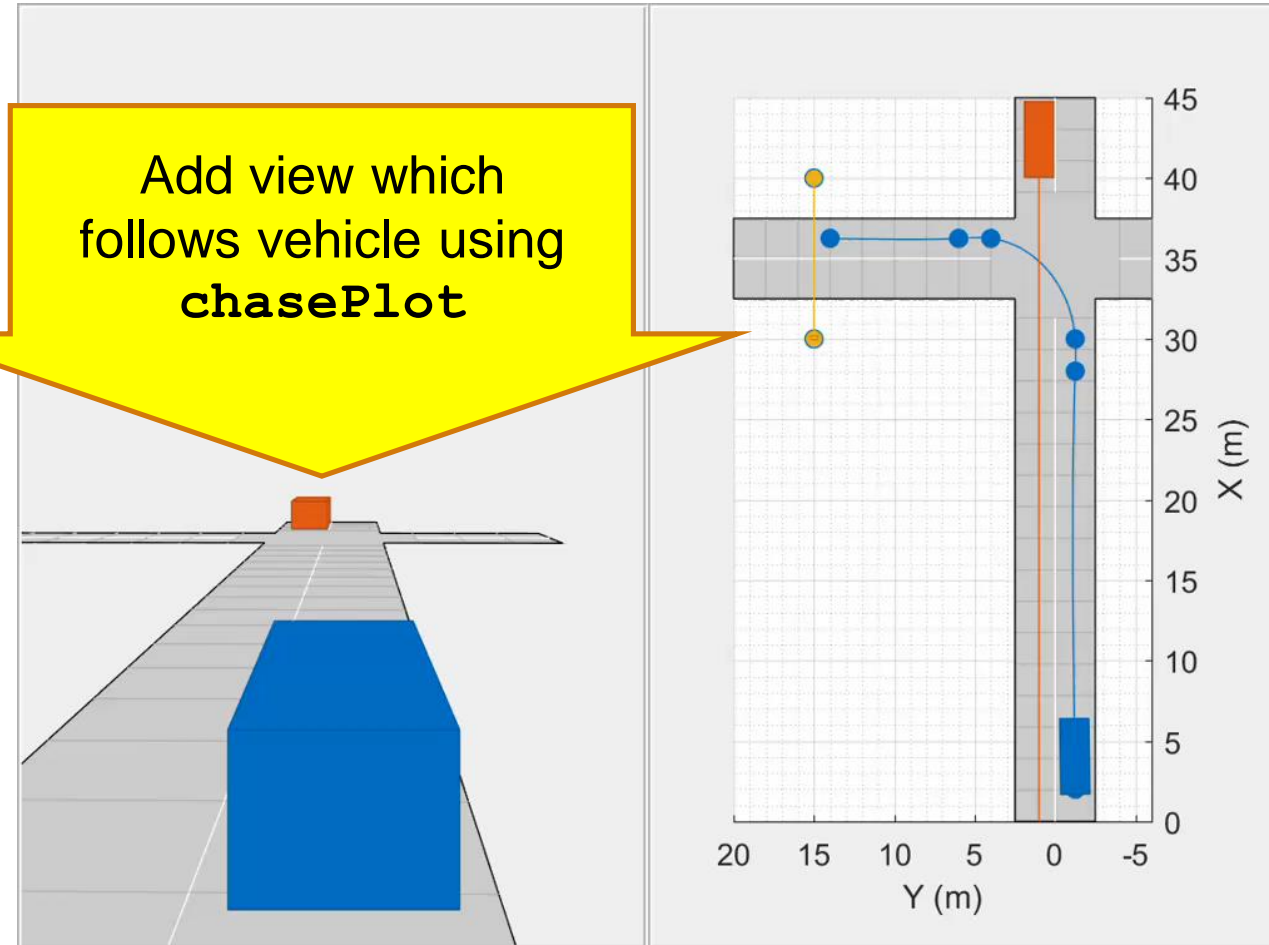
```

%% 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**



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
    ...
    'UpdateInterval', s.SampleTime, ...
    'BoundingBoxAccuracy', 5, ...
    'MaxRange', 150, ...
    'ActorProfiles', actorProfiles(s));
```

Model radar detection
sensor using
radarDetectionGenerator

Create birds eye plot to view sensor detections

```

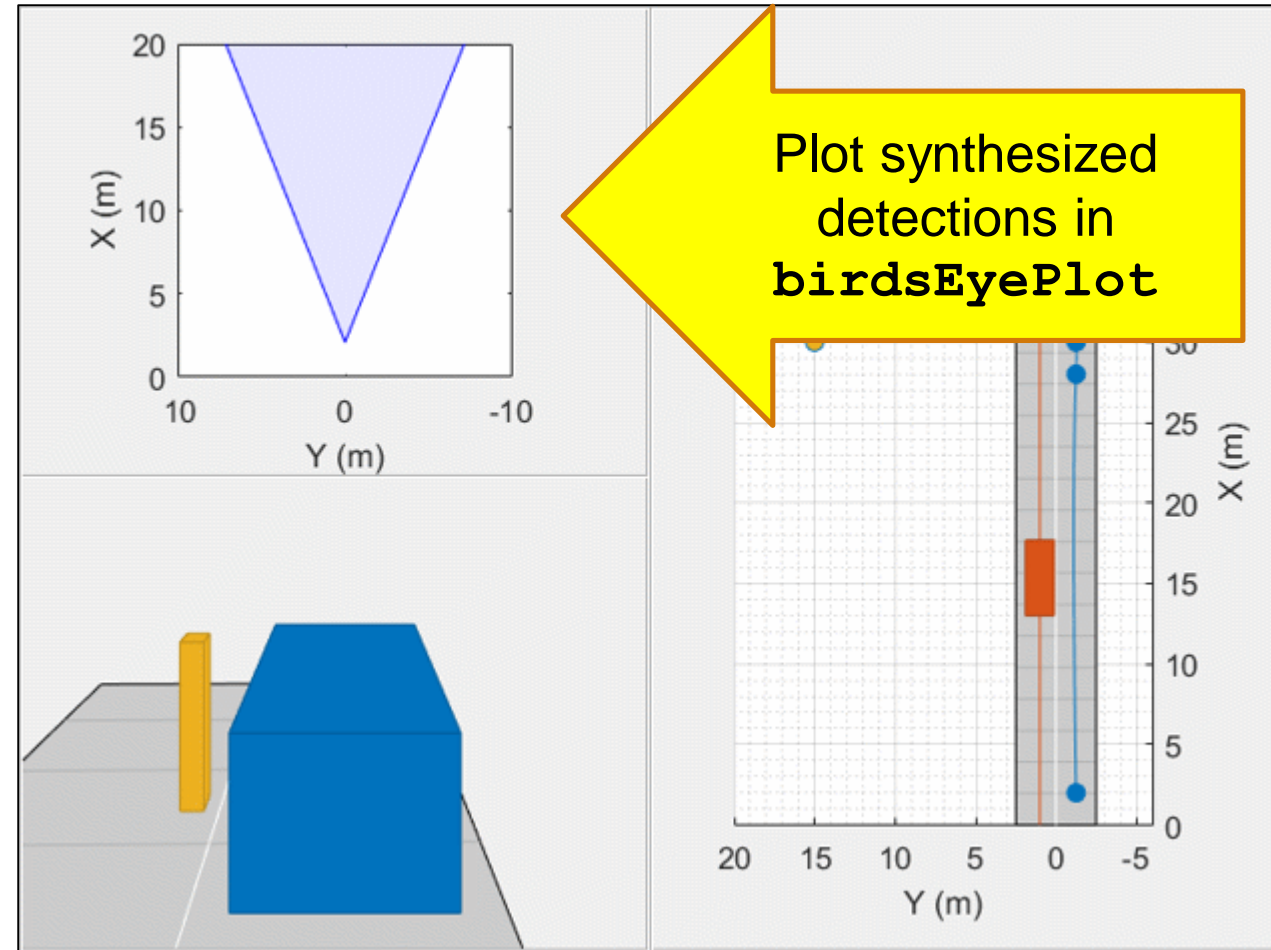
%% Add sensor birds eye plot (top left)
p3 = uipanel('Position',[0 0.5 0.5 0.5]);
a3 = axes('Parent',p3);
bep = birdsEyePlot('Parent',a3,...
    'Xlimits', [0 20],...
    'Ylimits', [-10 10]);
legend(a3,'off');

% 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

```

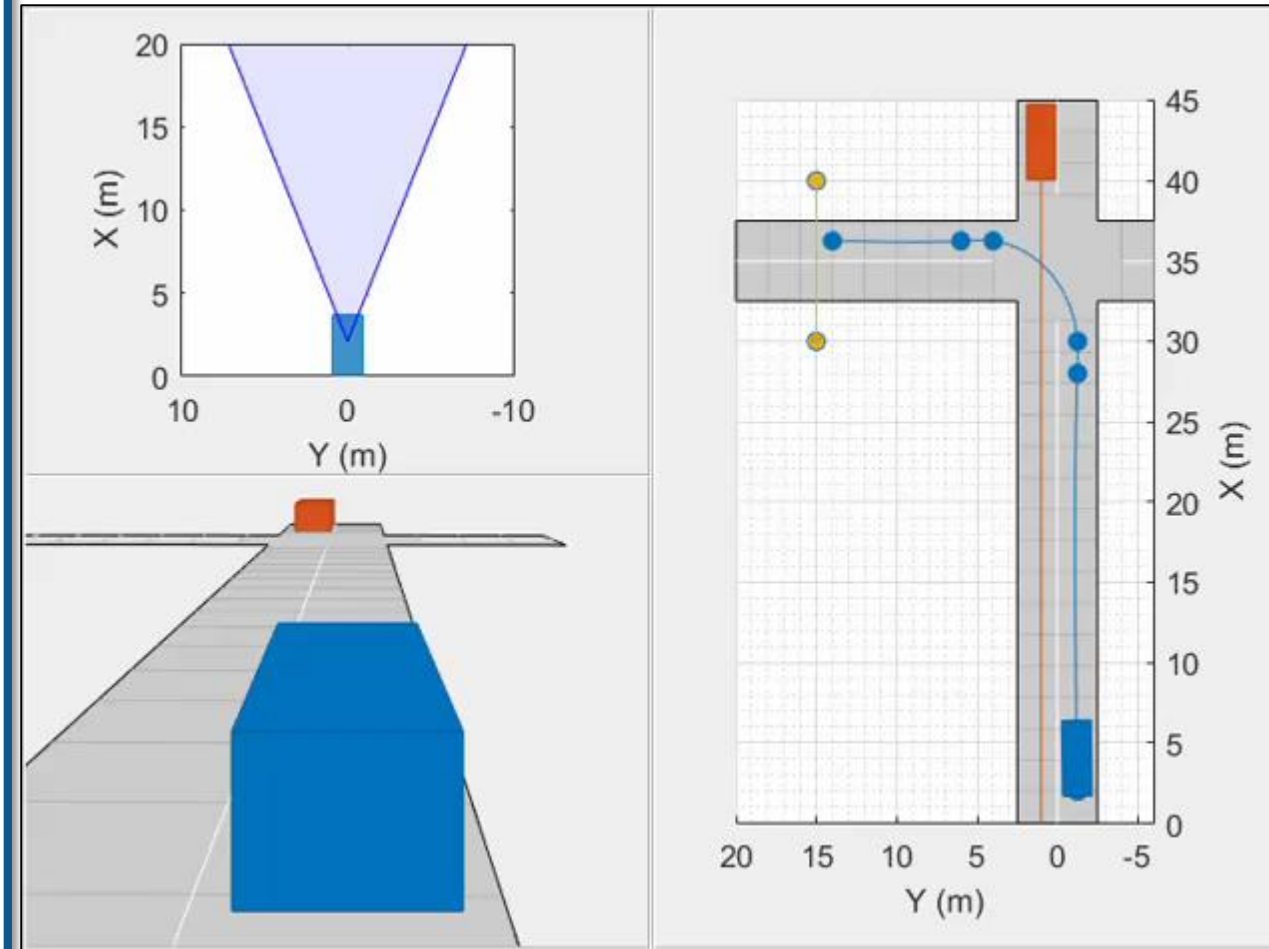
restart(s)
while advance(s)
    % Get detections in ego vehicle coordinates
    det = sensor(targetPoses(egoCar), ...
                s.SimulationTime);

    % 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
end

```

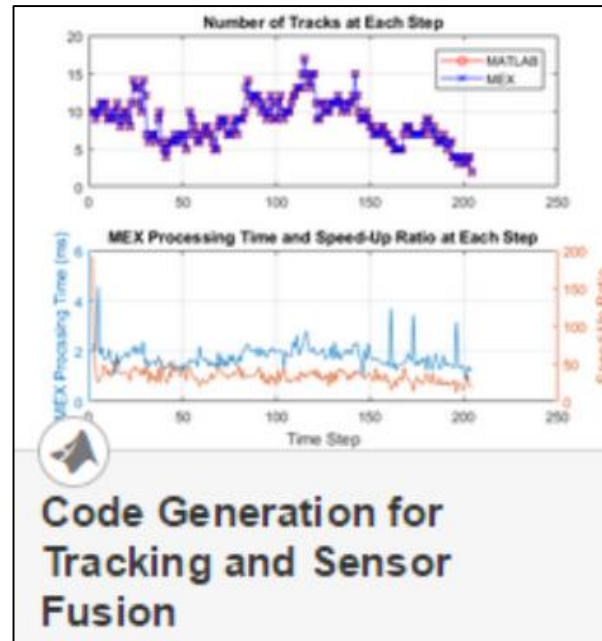


Learn more about sensor fusion

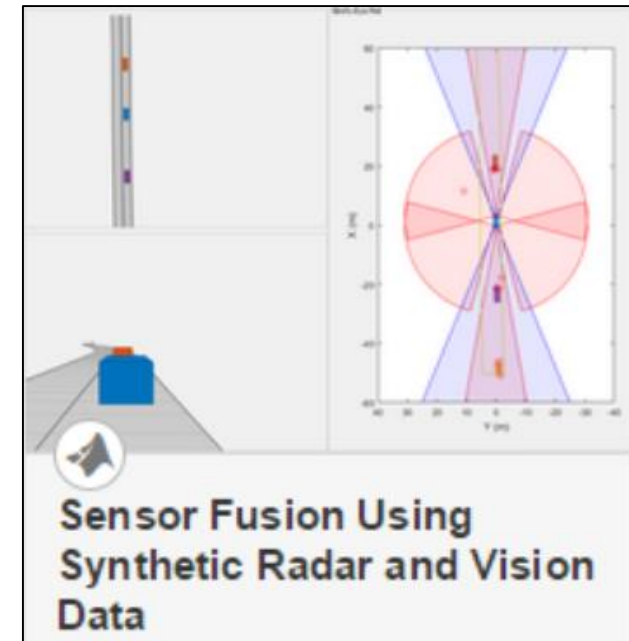
by exploring examples in the Automated Driving System Toolbox



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



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



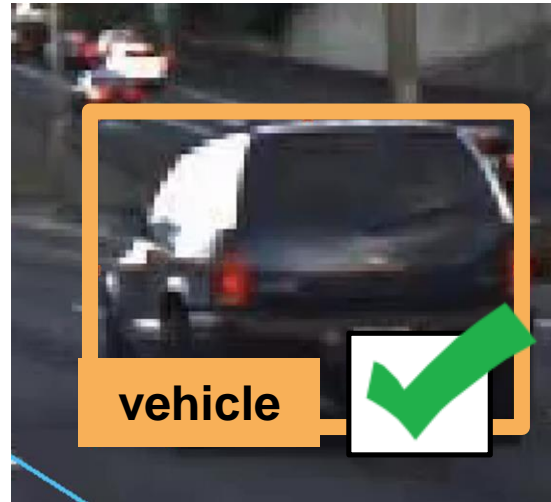
- **Synthesize driving scenario** to test multi-object tracker

The Automated Driving System Toolbox helps you...



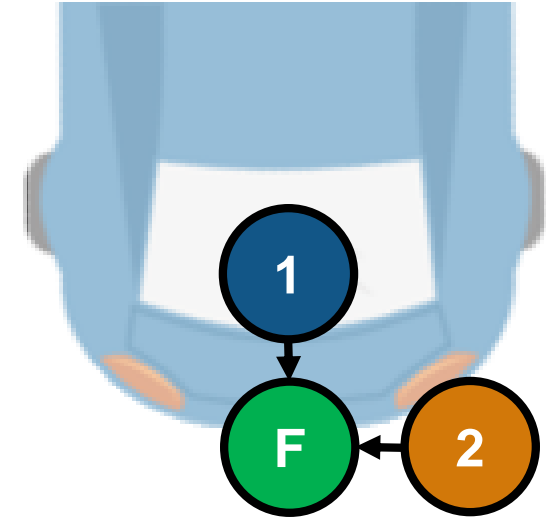
Visualize vehicle data

- 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

Stop by the demo booths to learn more about...

- Automated Driving System Toolbox
- Testing ADAS Algorithms: From Desktop to Real Time
- Automated Driving: Traffic Sign Recognition and Control
- Deep Learning for Computer Vision
- Developing Autonomous Systems

Attend related sessions to learn about...

- Developing Autonomous Systems with MATLAB and Simulink
14:45 – 15:30

- Simplifying Image Processing and Computer Vision Application Development
16:45 – 17:30



Vivek Raju, Application Engineer, MathWorks India



Elza John, Training Engineer, MathWorks India



Image Processing with MATLAB

This two-day course provides hands-on experience with performing image analysis. Examples and exercises demonstrate the use of appropriate MATLAB® and Image Processing Toolbox™ functionality throughout the analysis process.

Computer Vision with MATLAB

This one-day course provides hands-on experience with performing computer vision tasks. Examples and exercises demonstrate the use of appropriate MATLAB® and Computer Vision System Toolbox™ functionality



MathWorks®

Accelerating the pace of engineering and science

Speaker Details

Email: Mark.Corless@mathworks.com

Email: Amod.Anandkumar@mathworks.in

Contact MathWorks India

Products/Training Enquiry Booth

Call: 080-6632-6000

Email: info@mathworks.in

Your feedback is valued.

Please complete the feedback form provided to you.