

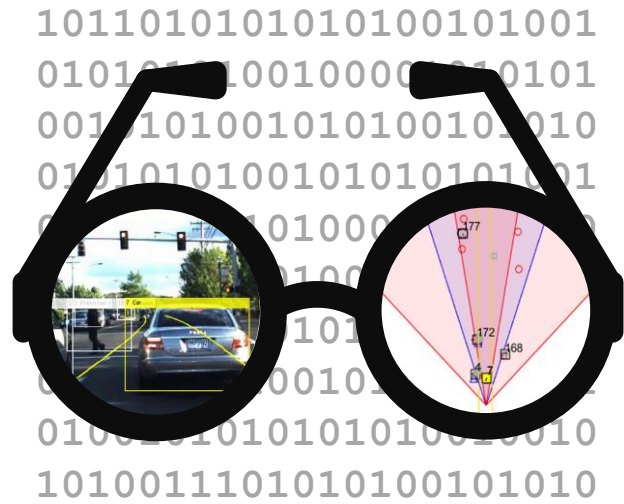
The background features a dark blue field on the left and a grey field on the right, separated by a diagonal line. In the upper right, there are white, stylized waveforms. In the lower right, there is a 3D wireframe mesh with a color gradient from yellow to blue, and a faint blue circuit board pattern.

# MATLAB EXPO 2017

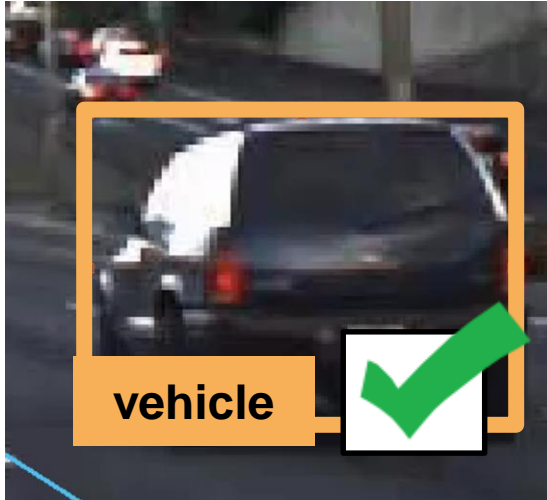
## Automated Driving: Design and Verify Perception Systems

Giuseppe Ridinò

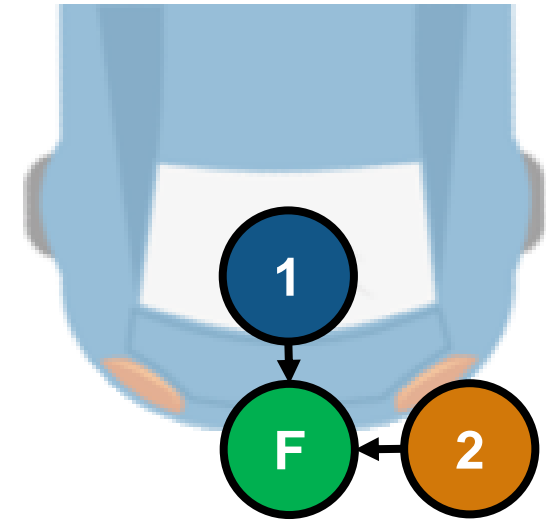
# Some common questions from automated driving engineers



**How can I  
visualize vehicle  
data?**

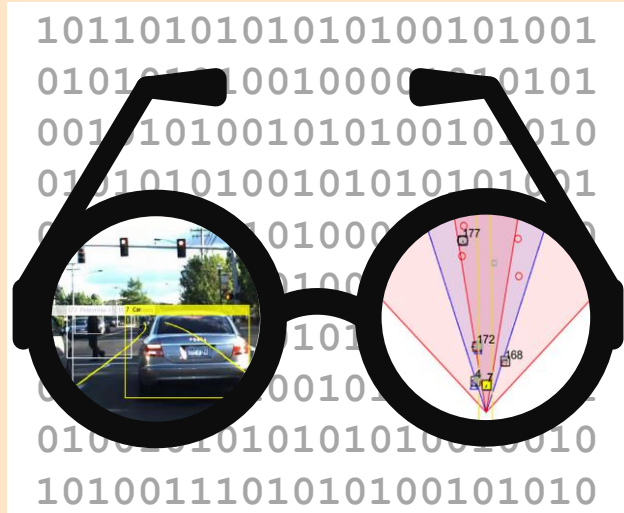


**How can I  
detect objects in  
images?**

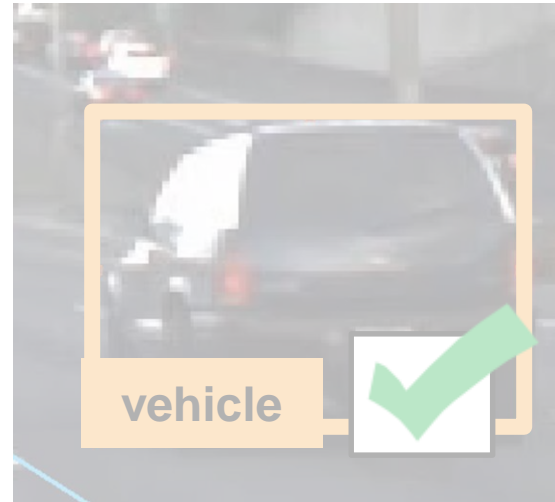


**How can I  
fuse multiple  
detections?**

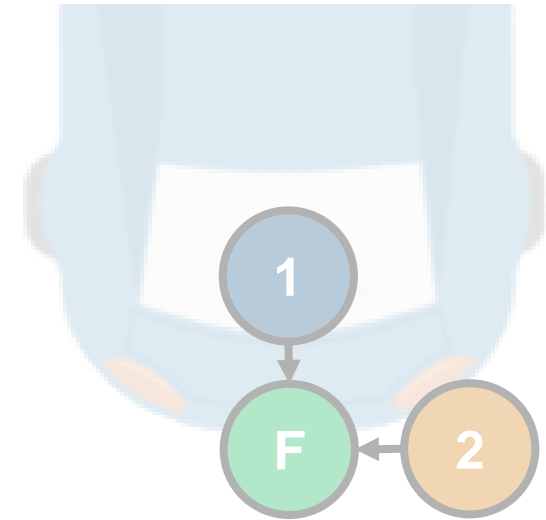
# Some common questions from automated driving engineers



**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

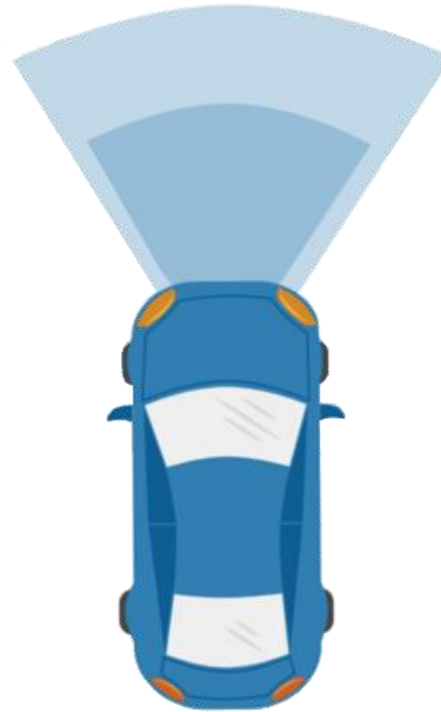
Radar-based  
object detector

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
252 252 251 252 252 253 253
```

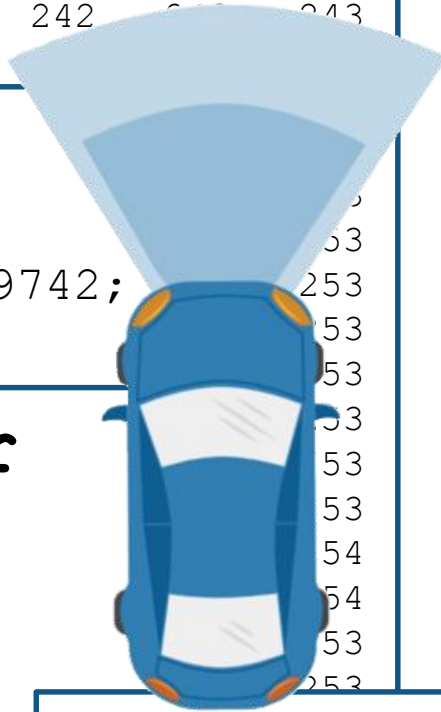
## Vision Detector

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

## Lane Detector

```

Trajectory
Class Left
Position IsValid: 1
Velocity Confidence: 3
Size BoundaryType: 3
Detect Offset: 1.68
Trajectory HeadingAngle: 0.002
Class Curvature: 0.000
Position Right
Velocity IsValid: 1
Size Confidence: 3
```



## Radar Detector

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

## Lidar (47197 x 3)

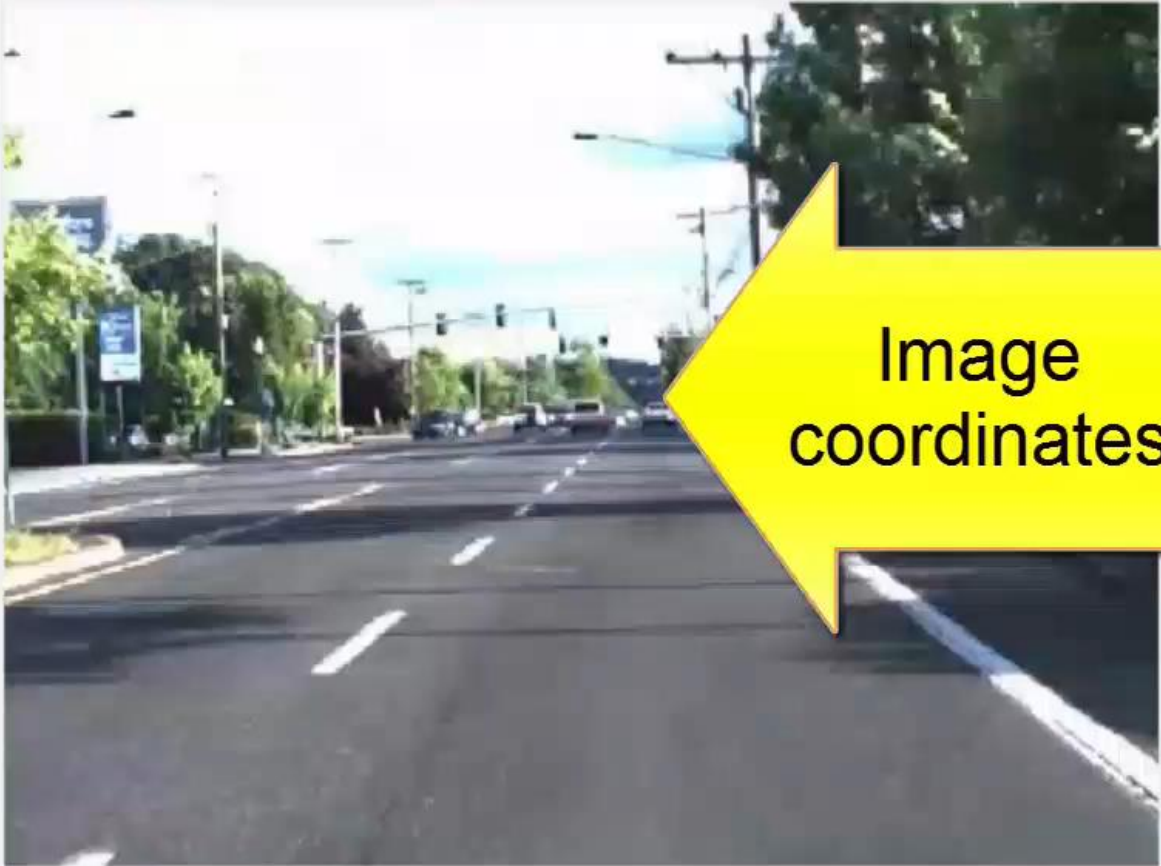
```
Detection
TrackID
TrackSt -12.2911 1.4790 -0.59
Position -14.8852 1.7755 -0.64
Velocity -18.8020 2.2231 -0.73
Amplitude -25.7033 3.0119 -0.92
Detection -0.0632 0.0815 1.25
TrackID -0.0978 0.0855 1.25
TrackSt -0.2814 0.1064 1.25
```

## Inertial Measurement Unit

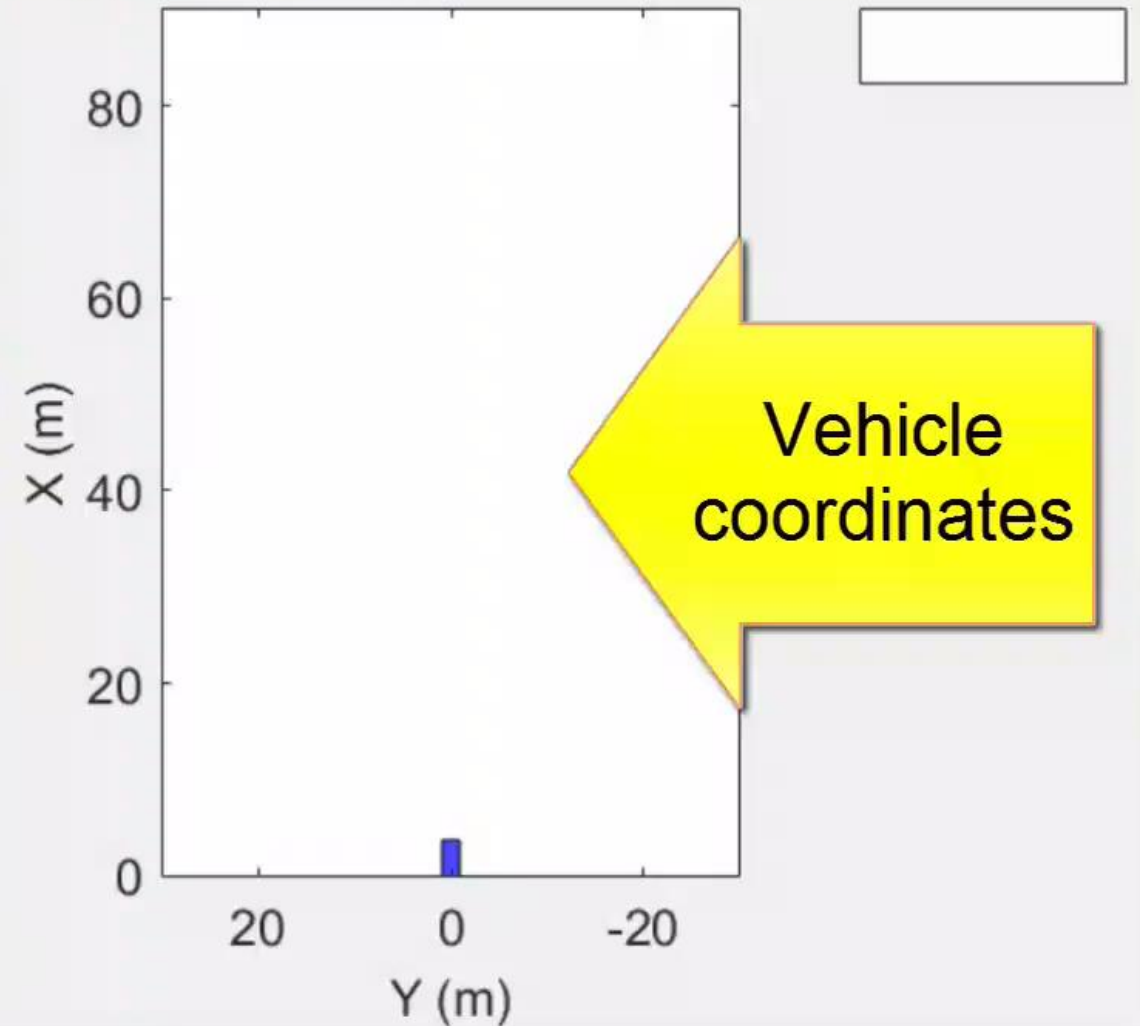
```
Timestamp: 1461634696379742
Velocity: 9.2795
YawRate: 0.0040
```

# Visualize sensor data

Image Coordinates



Vehicle Coordinates

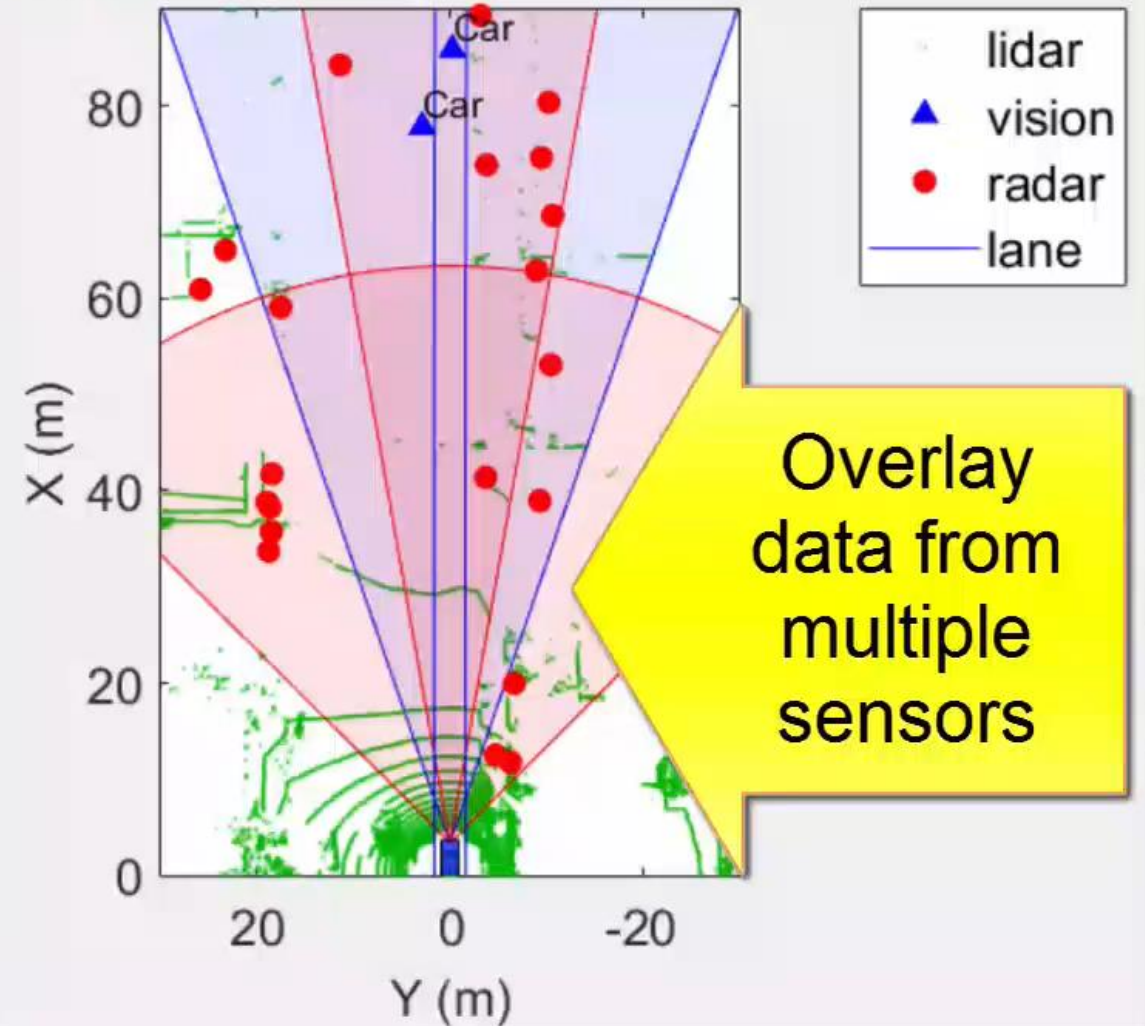


# Visualize differences in sensor detections

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')
```

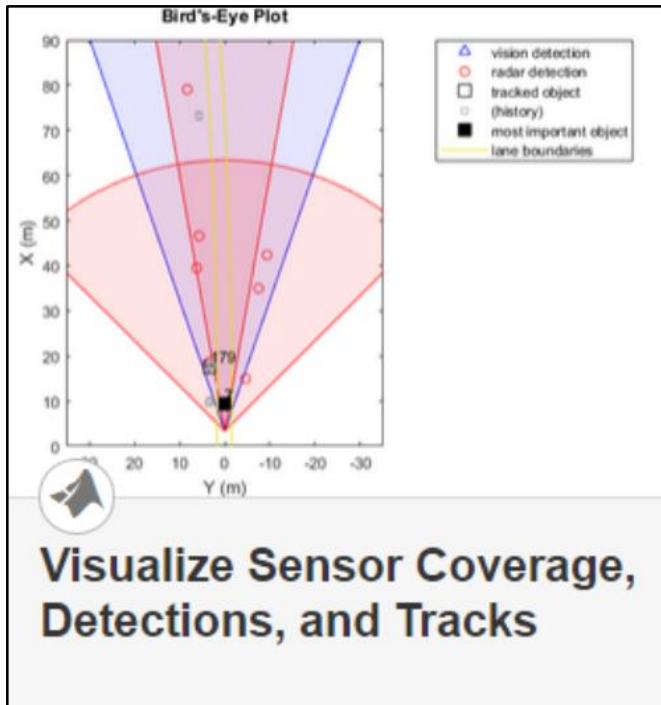
- Load **lidar point cloud data**

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

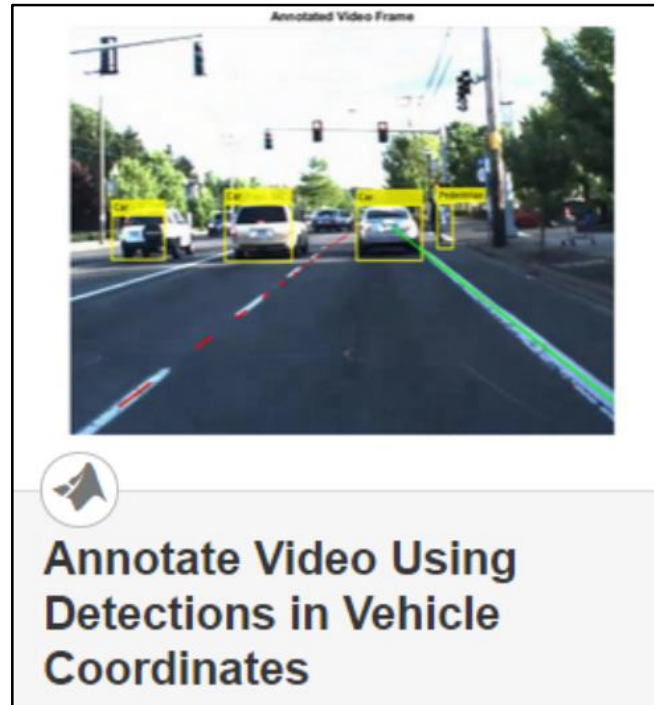


# Learn more about visualizing vehicle data

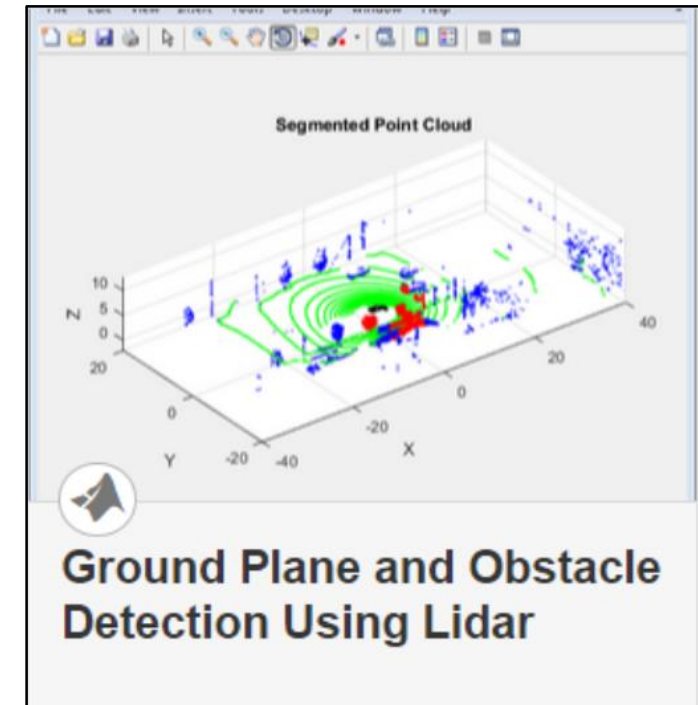
by exploring examples in the Automated Driving System Toolbox



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



- Transform between vehicle and image coordinates

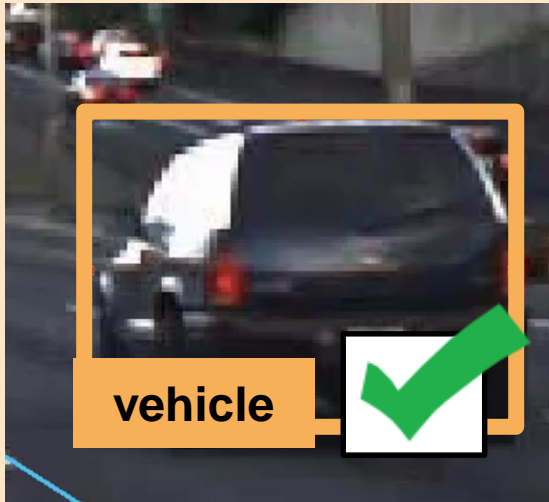


- Plot lidar point cloud

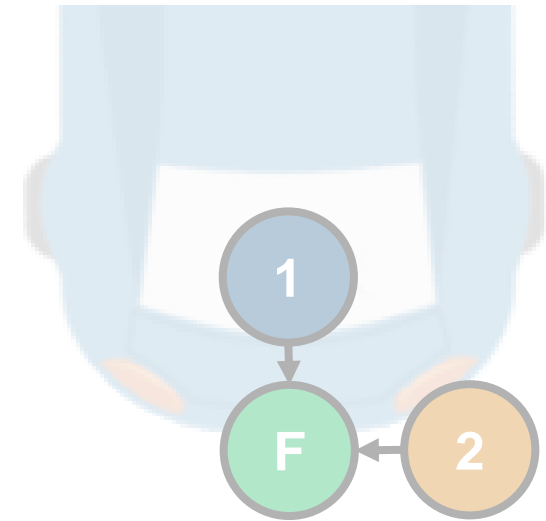
# Some common questions from automated driving engineers



How can I  
visualize vehicle  
data?

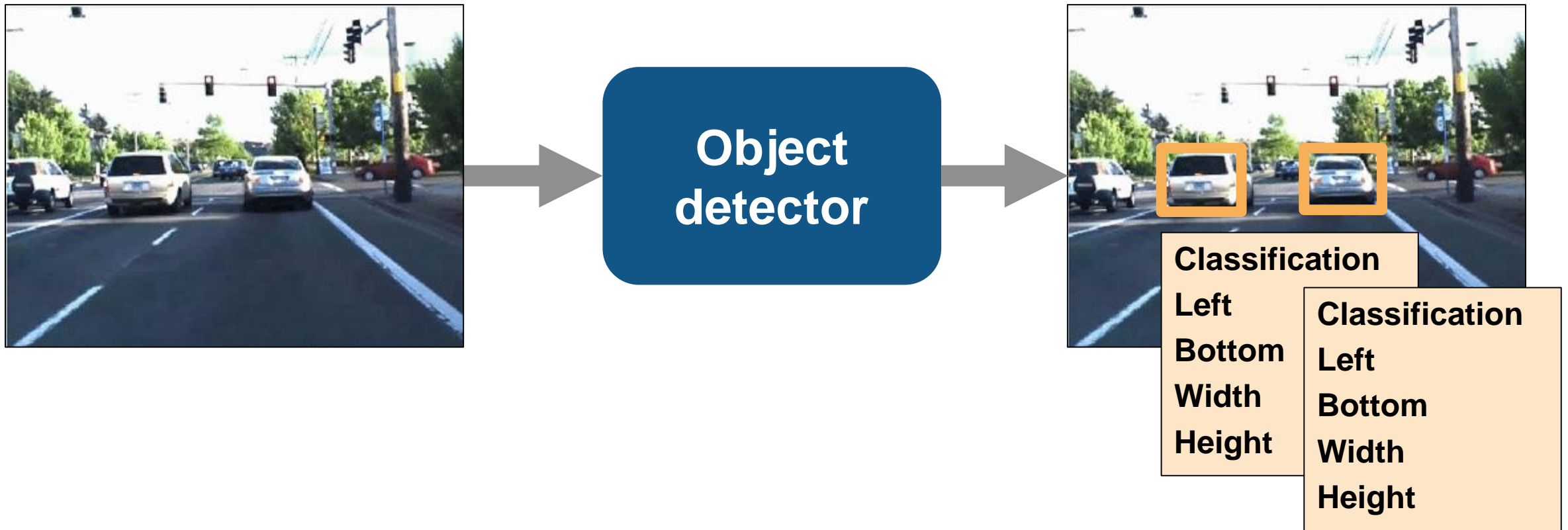


How can I  
detect objects in  
images?

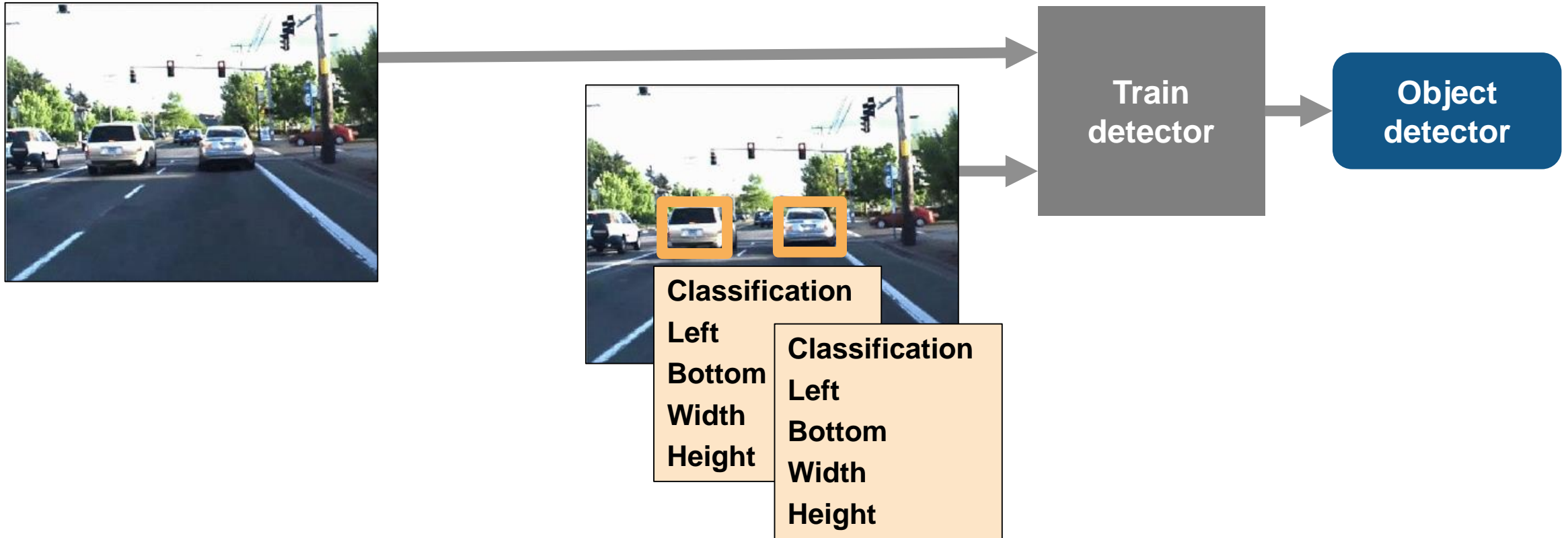


How can I  
fuse multiple  
detections?

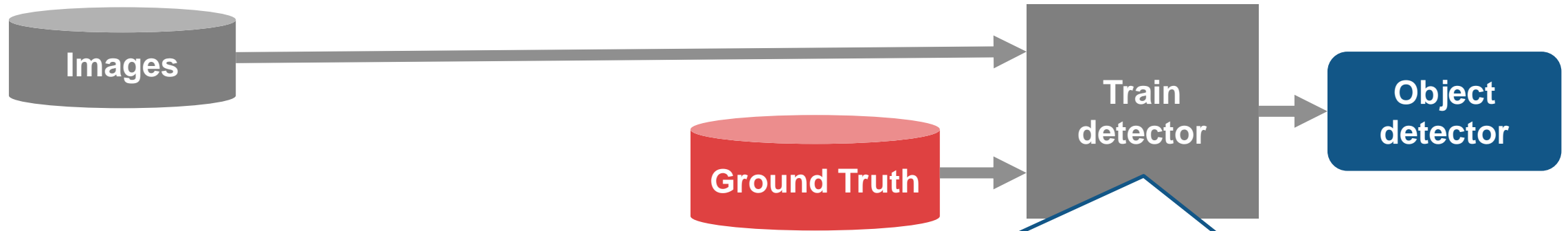
# How can I detect objects in images?



# Train object detectors based on ground truth



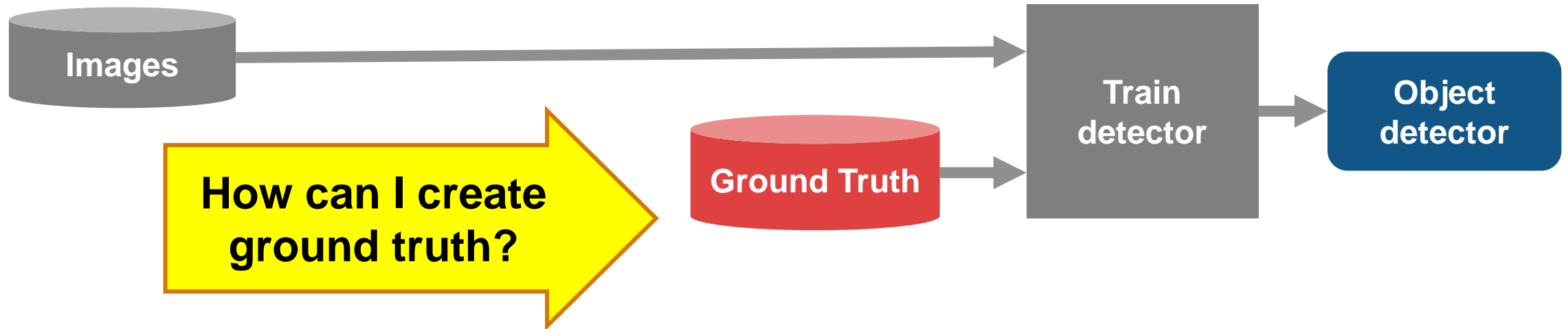
# Train object detectors based on ground truth



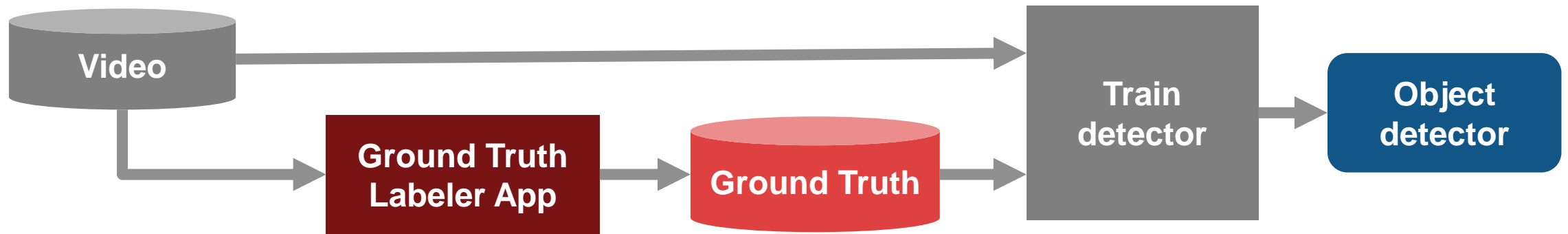
Design object detectors with the Computer Vision System Toolbox

<b>Machine Learning</b>	Aggregate Channel Feature	<code>trainACFObjectDetector</code>
	Cascade	<code>trainCascadeObjectDetector</code>
<b>Deep Learning</b>	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



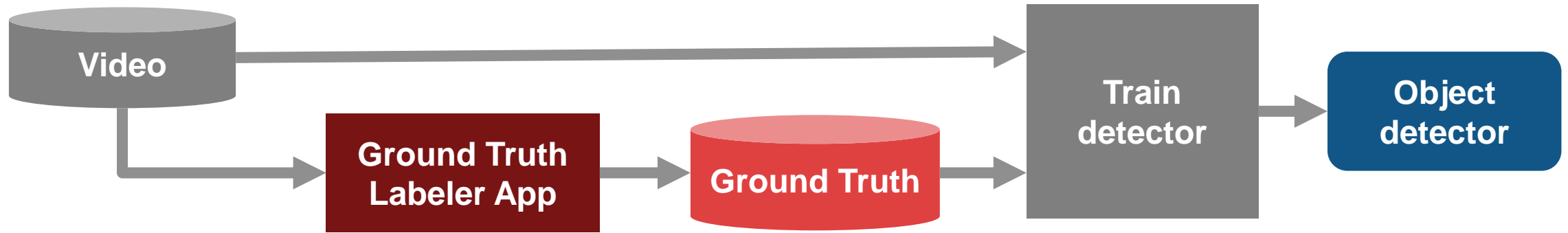
# Automate labeling based on a manually labeled frame with point tracker

The screenshot displays the Ground Truth Labeler software interface. The main window shows a video frame with a car labeled "Car". A context menu is open over the car, listing three algorithms: "ACF Vehicle Detector", "Point Tracker", and "Temporal Interpolator". A yellow arrow points to the "Point Tracker" option, which is highlighted. The interface includes a toolbar with options like "Load", "Save", "Import Labels", "ROI", "Zoom In", "Zoom Out", and "Pan". The left sidebar shows "ROI Label Definition" with a "Car" label and "Scene Label Definition" with "SunnyDay" and "LaneChange" labels. The bottom of the window features a timeline with markers for "Start Time", "Current", "End Time", and "Max Time", along with playback controls and a "Zoom In Time Interval" button.

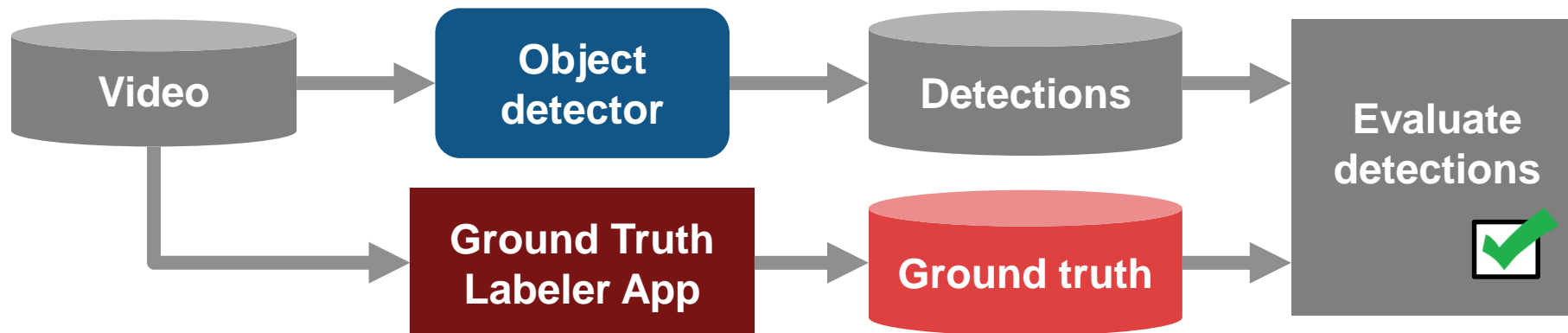
**Track region with Point Tracker**



## Ground truth labeling to train detectors



## Ground truth labeling to evaluate detectors



# Customize Ground Truth Labeler App

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

- Top Panel:** Contains a toolbar with icons for Load, Save, Import Labels, ROI, Zoom In, Zoom Out, and Pan. It also includes a 'Default Layout' button and an 'Algorithm:' dropdown menu with options for 'Select Algorithm', 'Automate', 'Configure Automation', 'View Label Summary', and 'Export Labels'.
- Left Panel:** A sidebar with sections for 'DATA SOURCE', 'LABEL DEFINITIONS', and 'SESSION'. Under 'DATA SOURCE', 'Custom Reader' is highlighted with a red box. Below this, there are options for 'Label Definitions' and 'Session'.
- Center Panel:** A large yellow arrow points to the 'Custom Reader' option with the text: "Add custom image reader with `groundTruthDataSource`". Below the arrow, a video frame shows a street scene with labels for 'Car', 'Pedestrian', and 'Lane'.
- Bottom Panel:** A 'Scene Label Definition' section with a 'Define New Scene Label' button and radio buttons for 'Current Frame' and 'Time Interval'. Below this is a timeline with 'Start Time' (00.00000), 'Current' (09.00000), 'End Time' (10.20000), and 'Max Time' (10.20000). Navigation buttons and a 'Zoom' button are also present.

# Customize Ground Truth Labeler App

The screenshot shows the 'Ground Truth Labeler - gtlCustomizations' window. On the left, there are panels for 'ROI Label Definition' and 'Scene Label Definition'. The 'ROI Label Definition' panel lists labels: Car, Pedestrian, StopLight, and Lane. The 'Scene Label Definition' panel has options for 'Current Frame' and 'Time Interval'. In the center, a video frame shows a car and a pedestrian with bounding boxes and labels. On the right, an 'Algorithm:' dropdown menu is open, showing a list of algorithms: Point Tracker, Temporal Interpolator, and ACF Vehicle Detector. Below the list are buttons for 'Add Algorithm', 'Refresh list', 'Create New Algorithm', and 'Import Algorithm'. The 'Create New Algorithm' button is highlighted with a red box. At the bottom, a timeline shows 'Start Time' (00.00000), 'Current' (09.00000), 'End Time', and 'Max Time'.

Add custom automation algorithm  
`driving.automation.AutomationAlgorithm`

# Customize Ground Truth Labeler App

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 Export

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

01\_city\_c2s\_fcw\_1

00.00000 09.00000 10.20000 10.20000

Start Time Current End Time Max Time

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

Figure 1: Point Cloud Pla...

File Edit View Insert Tools Desktop Window Help

# Learn more about detecting objects in images

by exploring examples in the Automated Driving System Toolbox

The screenshot shows the 'Ground Truth Labeler' application interface. On the left, there are panels for 'DATA SOURCE' (Video, Image Sequence, Custom Reader), 'LABEL DEFINITIONS' (Label Definitions), and 'SESSION'. The main area is divided into 'ROI Label Definition' and 'Scene Label Definition' sections. The 'ROI Label Definition' section includes a 'Define New ROI Label' button and a list of labels like 'cars' and 'streetLights'. The 'Scene Label Definition' section includes a 'Define New Scene Label' button and options for 'Current Frame' (Add Label, Remove Label) and 'Time Interval'. A video preview window shows a street scene with bounding boxes around objects. At the bottom, there are four steps: 'LOAD Video, Image Sequence, or Custom Reader', 'DEFINE ROIs and Scene Label Definitions', 'SET Interval and Controls', and 'LABEL Rectangles & Lines'.

**Define Ground Truth Data for Video or Image Sequences**

- Label detections with Ground Truth Labeler App

The screenshot shows the 'Automate' interface for lane boundary detection. It features a central video window with a road scene and yellow and blue lines representing lane boundaries. The interface includes a toolbar with 'Zoom In', 'Zoom Out', 'Default Layout', 'Settings', 'Run', 'Stop', 'Undo Run', 'Accept', and 'Cancel'. Below the video window, there are 'ROI Label Definition' and 'Scene Label Definition' panels. The 'ROI Label Definition' panel shows a 'LaneBound...' label. The 'Scene Label Definition' panel has 'Current Frame' and 'Time Interval' options. A timeline at the bottom shows 'Start Time', 'Current', 'End Time', and 'Max Time' values. A 'MATLAB' logo is visible in the bottom left corner.

**Automate Ground Truth Labeling of Lane Boundaries**

- Add automation algorithm for lane tracking

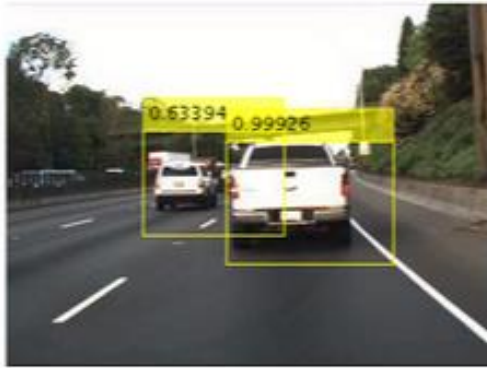
The screenshot shows the 'Ground Truth Labeler' application interface with a 3D Lidar display. The main window shows a street scene with a 3D Lidar point cloud overlay. The interface includes a toolbar with 'Load', 'Save', 'Import', 'ROI', 'Zoom In', 'Zoom Out', 'Fan', 'Show ROI Labels', 'Configure Automation', 'Automate', 'View Label', 'Export Labels', and 'Session'. Below the main window, there are 'ROI Label Definition' and 'Scene Label Definition' panels. A '3D View' window is open, showing a 3D coordinate system with 'X', 'Y', and 'Z' axes. The text 'driving.connector.Connector class' is highlighted in orange.

**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**



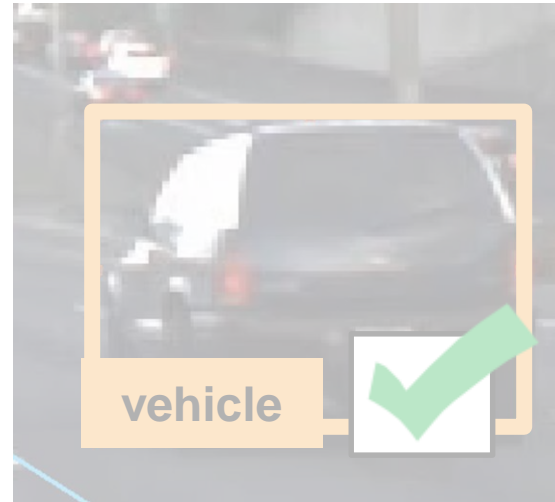
**Visual Perception Using Monocular Camera**

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

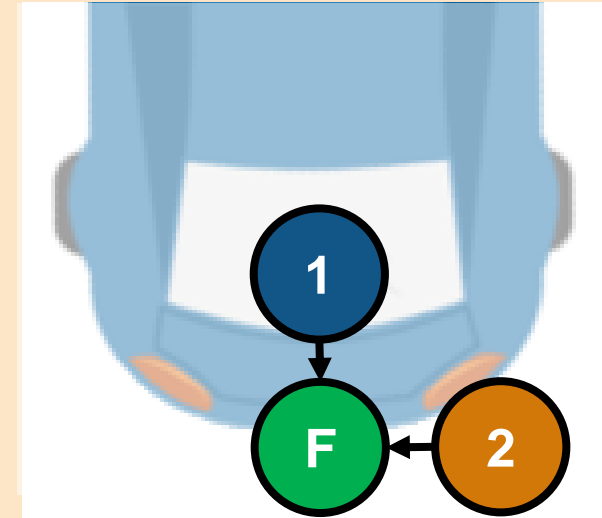
# Some common questions from automated driving engineers



How can I  
visualize vehicle  
data?

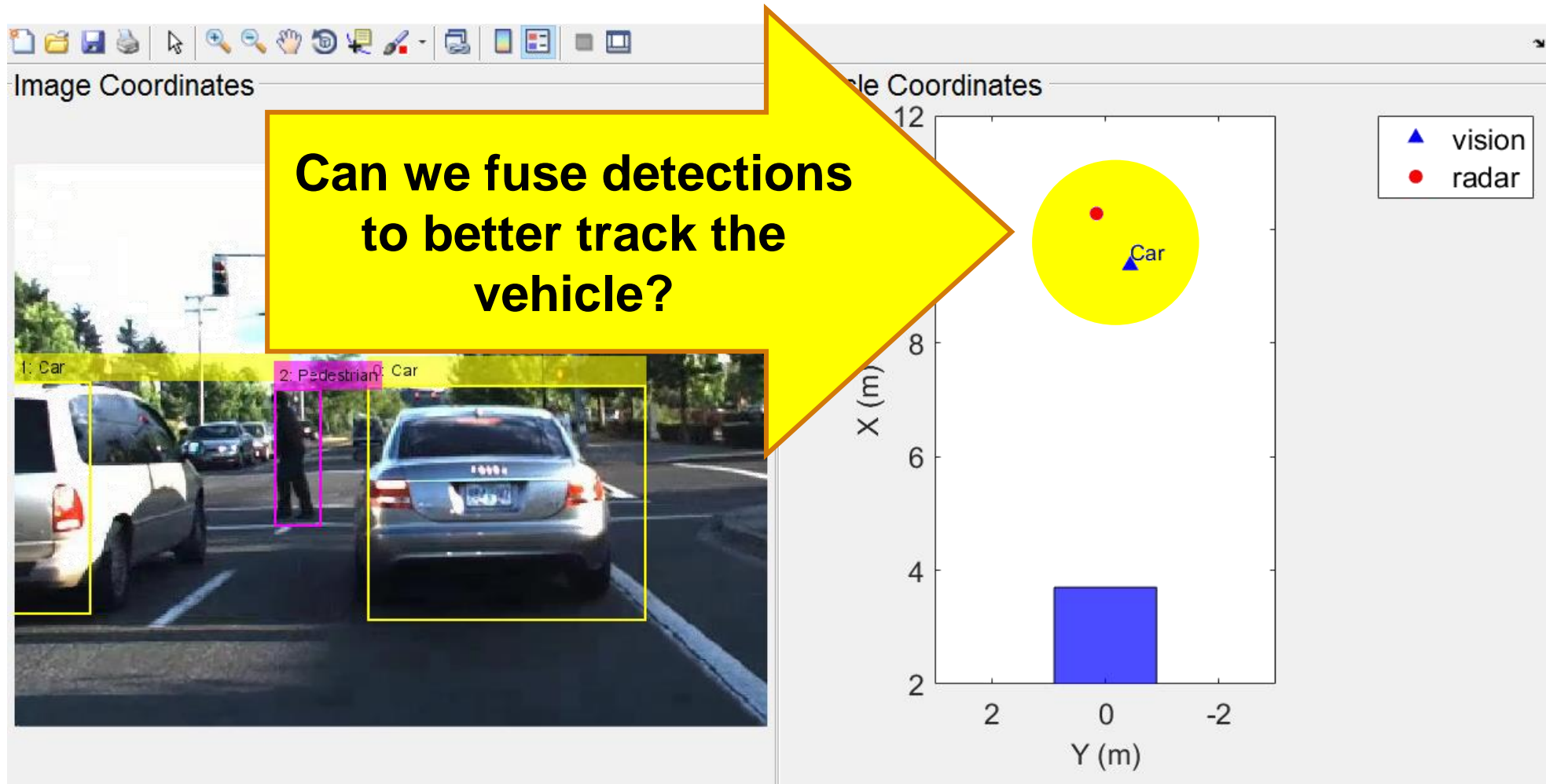


How can I  
detect objects in  
images?



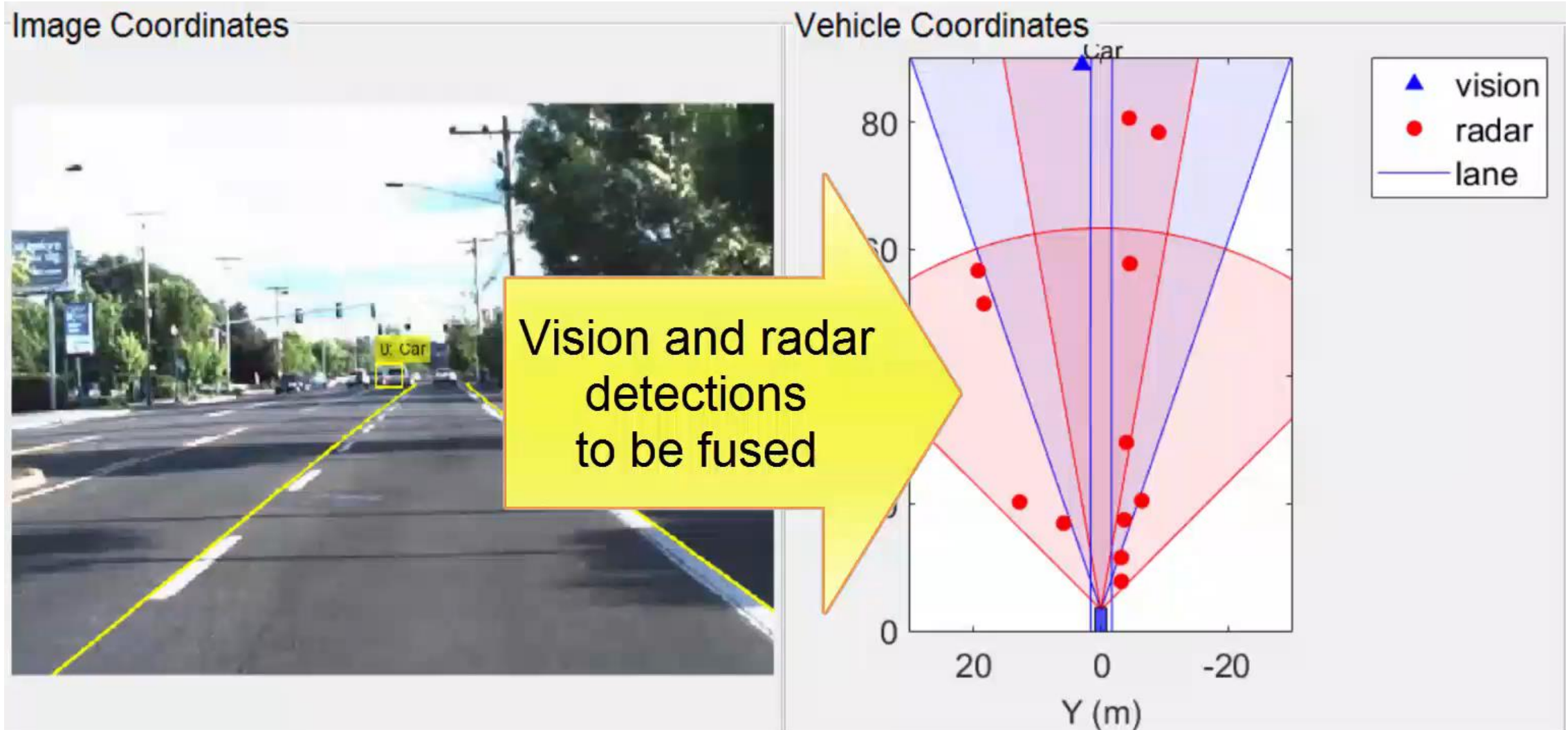
How can I  
fuse multiple  
detections?

# Example of radar and vision detections of a vehicle

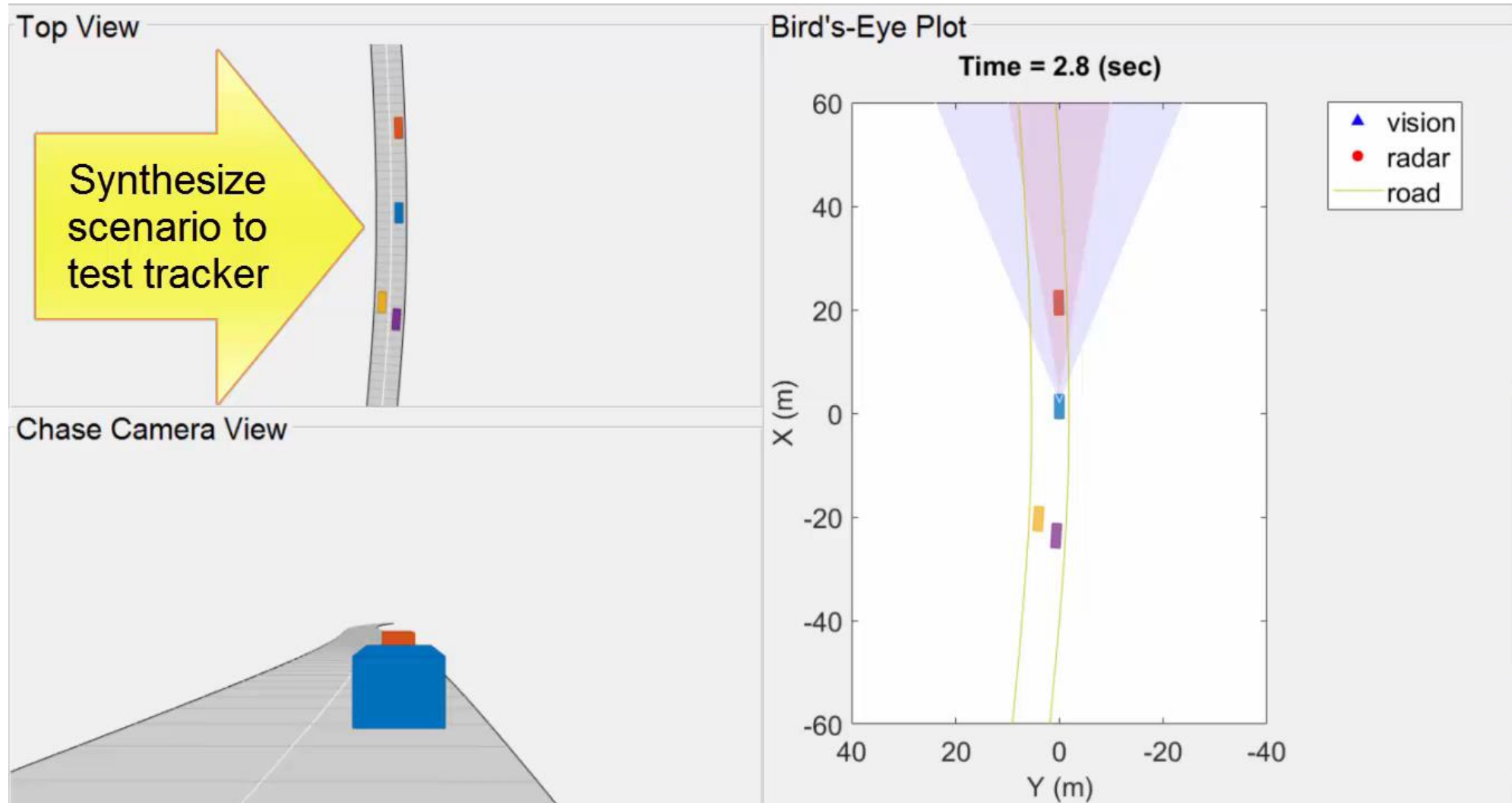




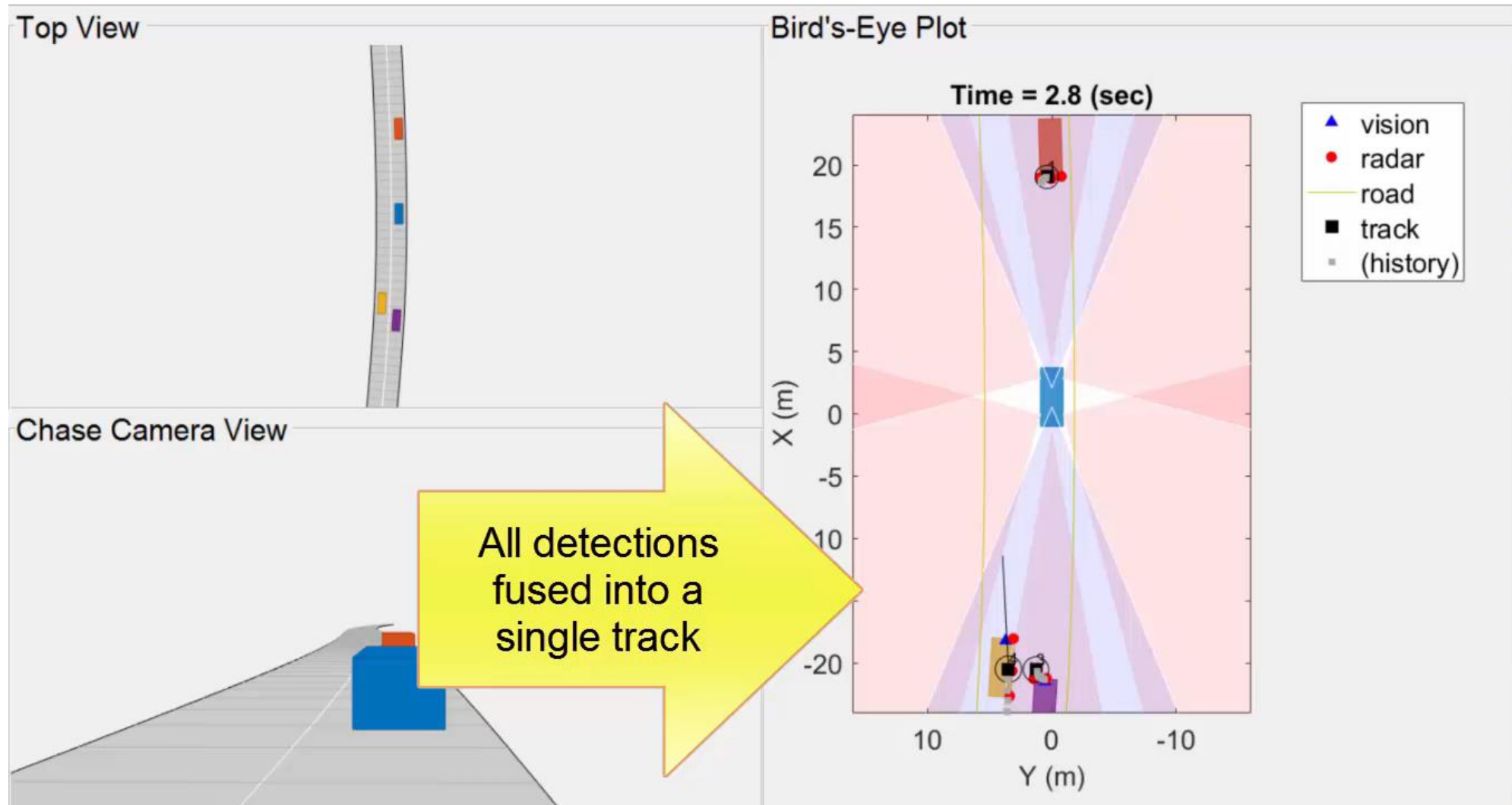
# Fuse detections with multi-object tracker



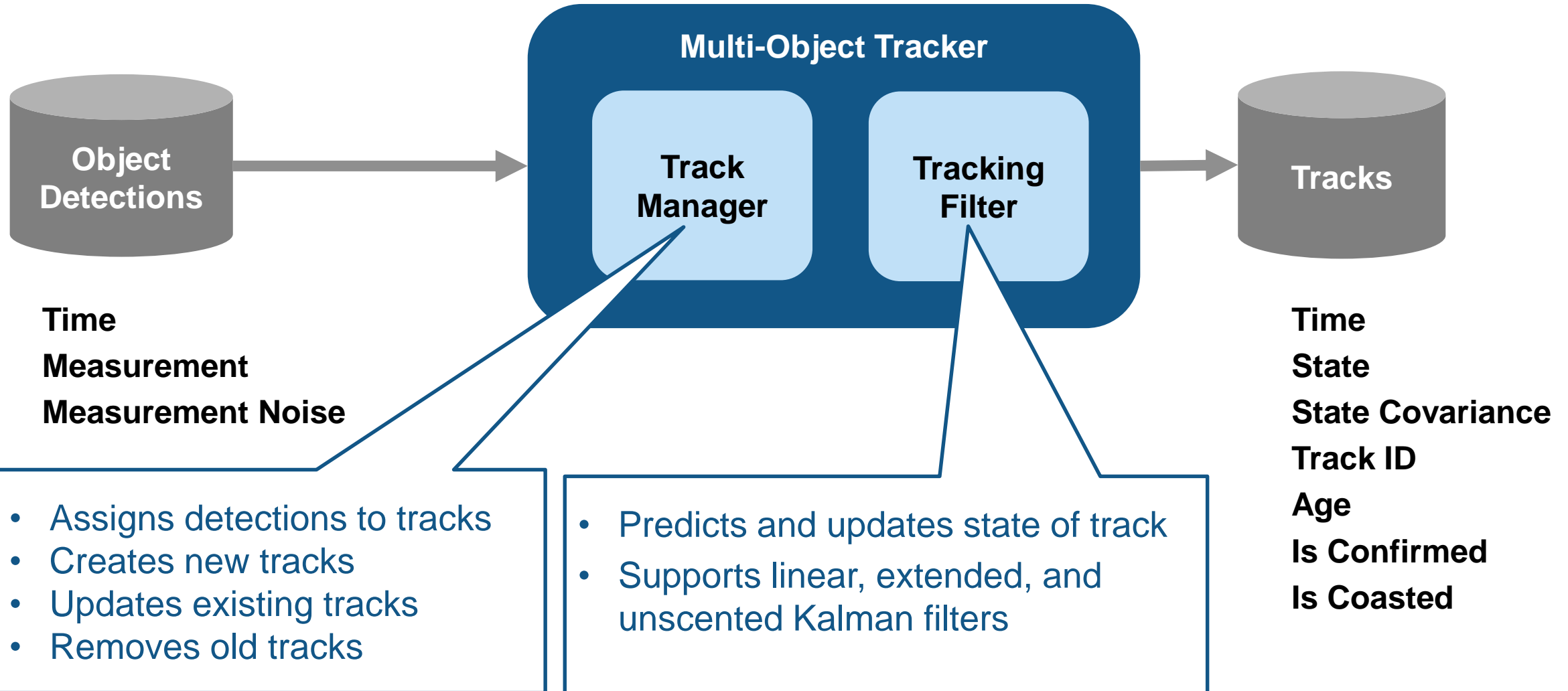
# Synthesize scenario to test tracker



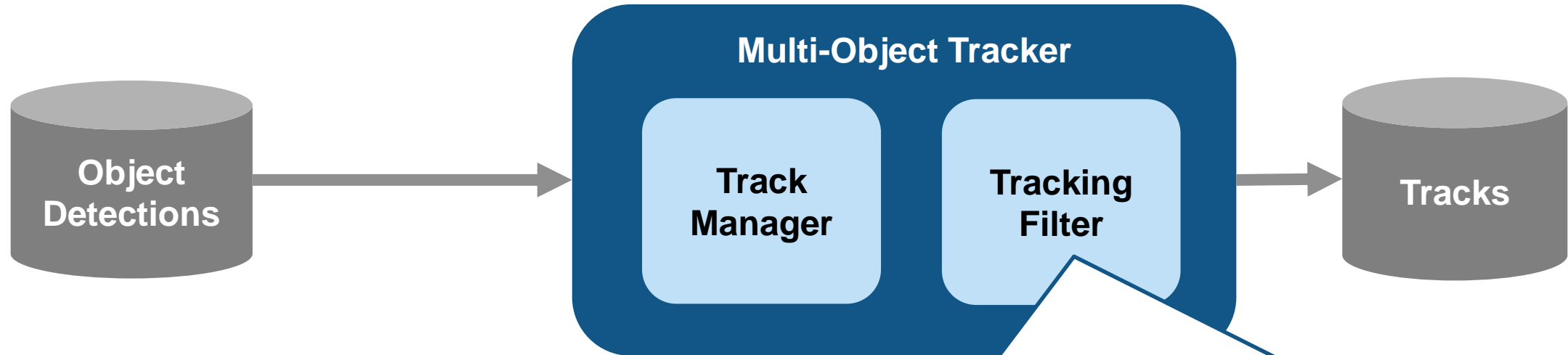
# Test tracker against synthesized data



# Track multiple object detections

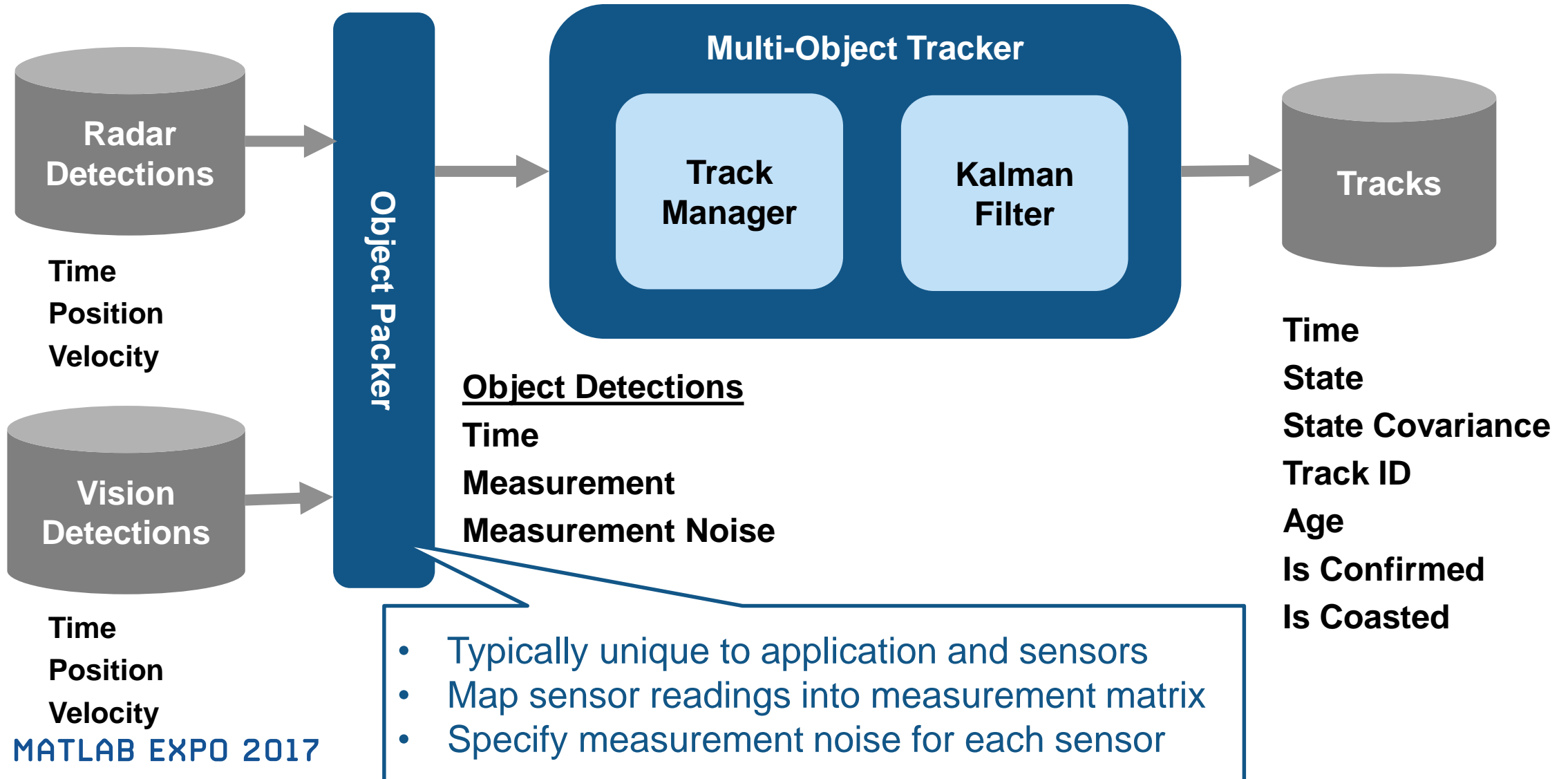


# Examples of Kalman Filter (KF) initialization functions

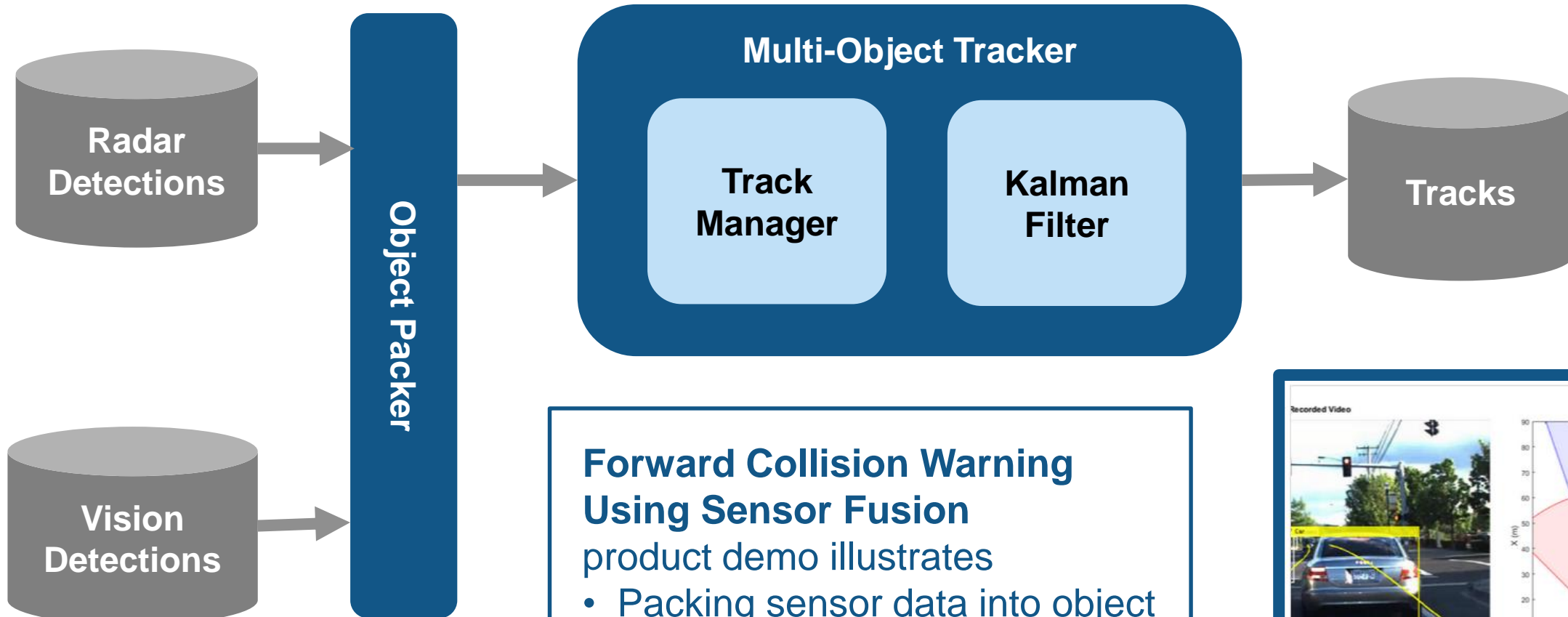


	Linear KF (trackingKF)	Extended KF (trackingEKF)	Unscented KF (trackingUKF)
<b>Constant velocity</b>	initcvkf	initcvekf	initcvukf
<b>Constant acceleration</b>	initcakf	initcaekf	initcaukf
<b>Constant turn</b>	Not applicable	initctekf	initctukf

# Fuse and track multiple detections from different sensors

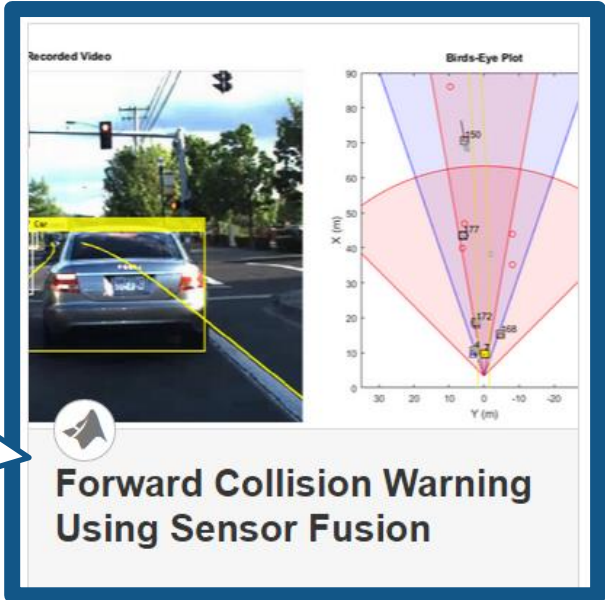


# Explore demo to learn more about fusing detections



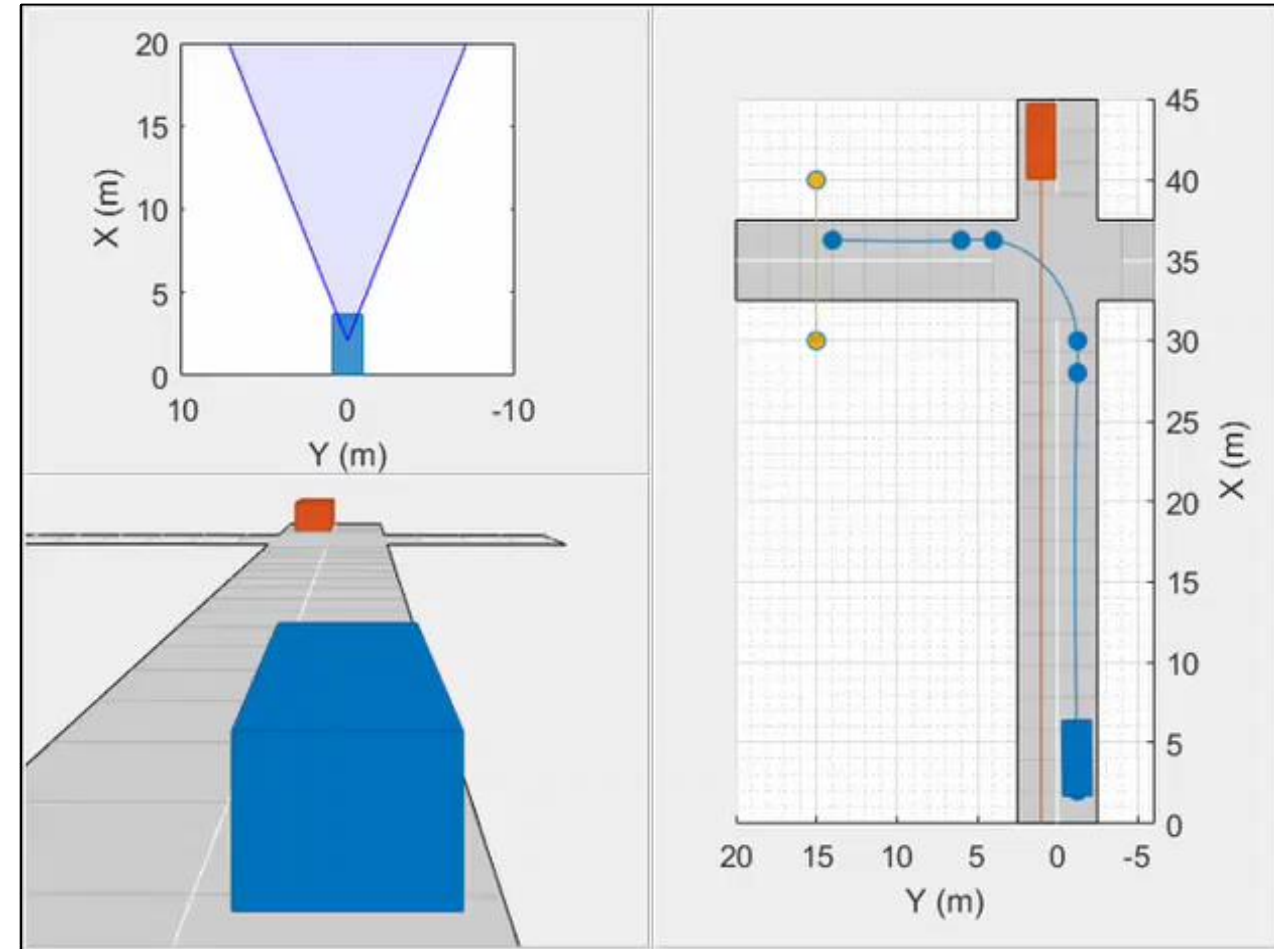
**Forward Collision Warning Using Sensor Fusion**  
 product demo illustrates

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



# Virtual scenario generation

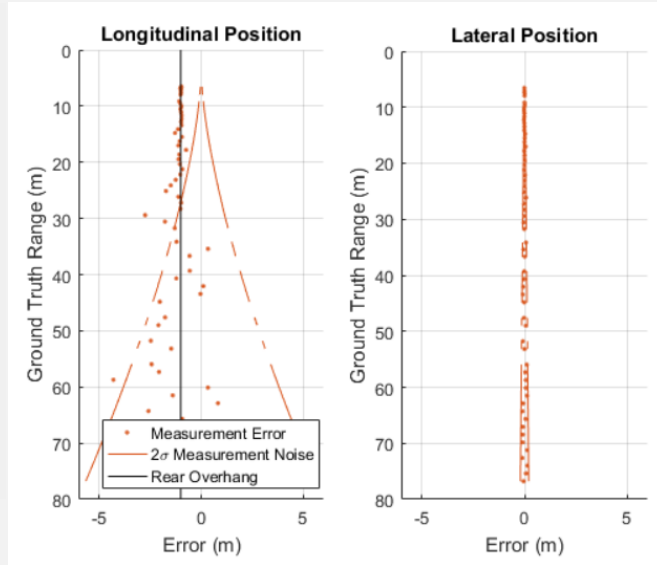
- Specify driving scenario and roads
- Add ego vehicle
- Add target vehicle and pedestrian actor
- Play scenario with chase plot
- Create birds eye plot to view sensor detections
- Play scenario with sensor models





# Simulate effects of vision detection sensor

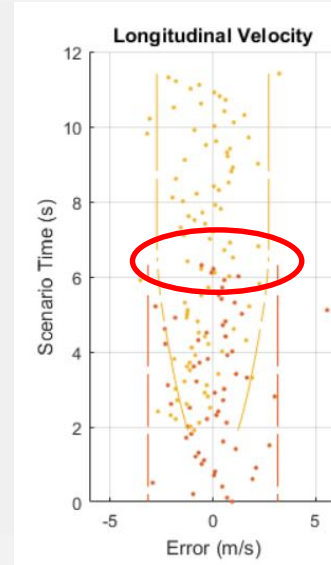
## Range Effects



Range measurement accuracy degrades with distance to object

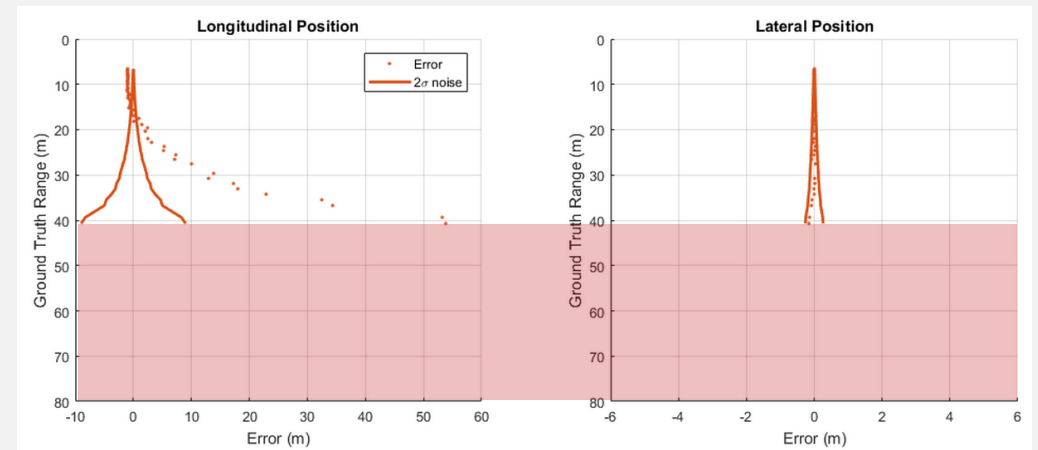
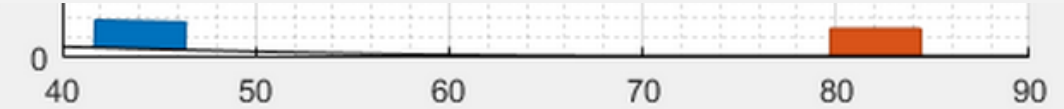
Angle measurement accuracy consistent throughout coverage area

## Occlusion Effects



Partially or completely occluded objects are not detected

## Road Elevation Effects



Objects in coverage area may not be detected because they appear above the horizon line

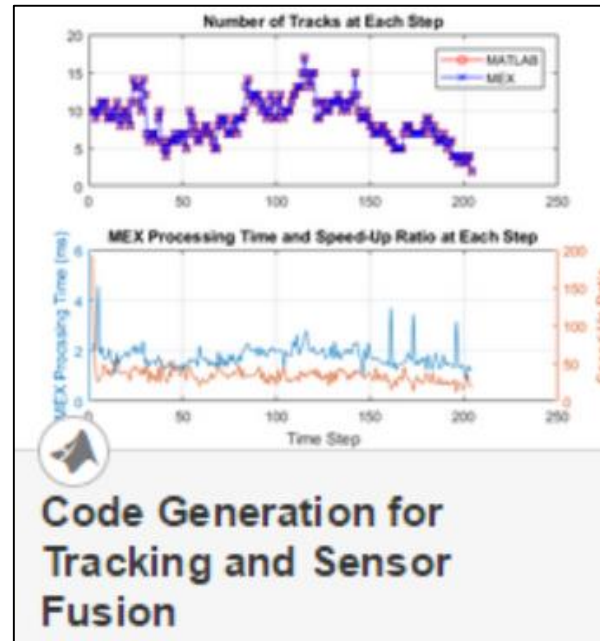
Large range measurement errors may be introduced for detected objects

# Learn more about sensor fusion

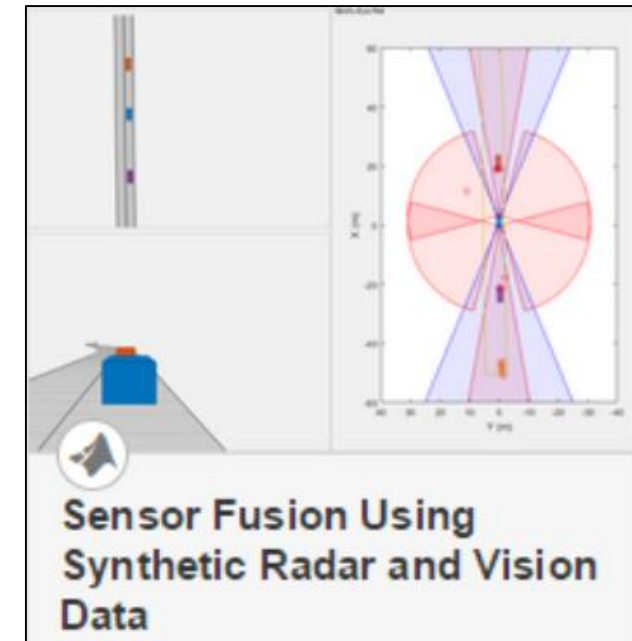
by exploring examples in the Automated Driving System Toolbox



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

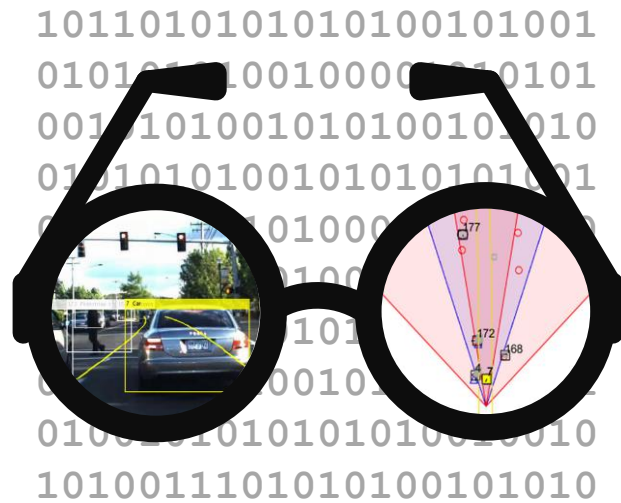


- **Generate C/C++** code from algorithm which includes a multi-object tracker



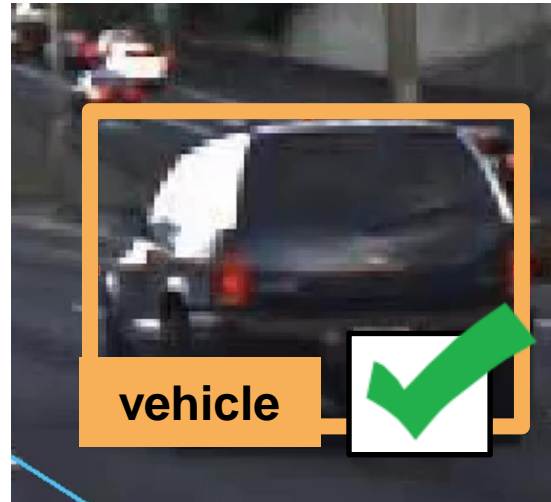
- **Synthesize driving scenario** to test multi-object tracker

# The Automated Driving System Toolbox helps you...



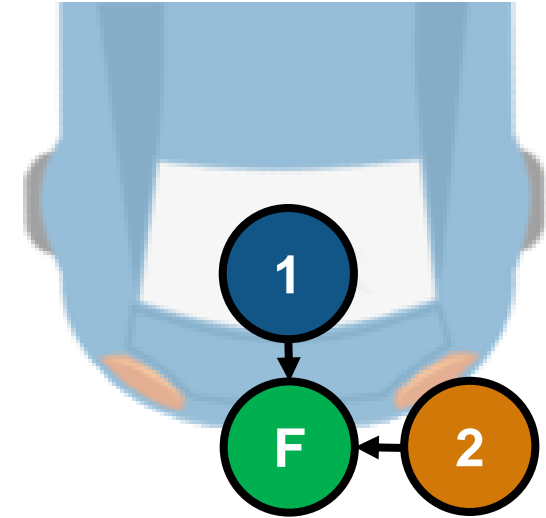
## Visualize vehicle data

- 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