

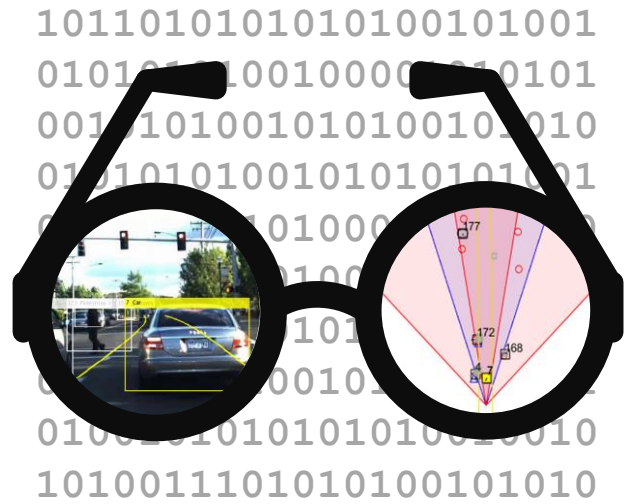


MATLAB EXPO 2017

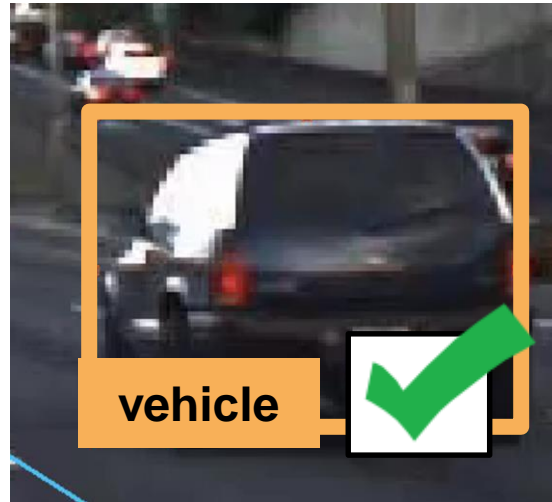
自动驾驶：设计和验证感知系统

陈小挺 高级应用工程师，MathWorks 中国

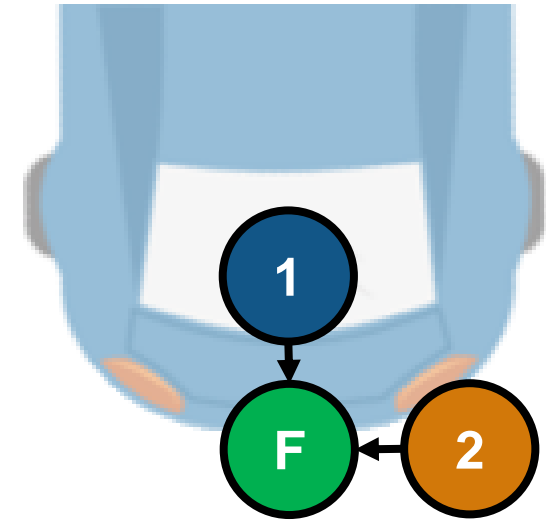
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？

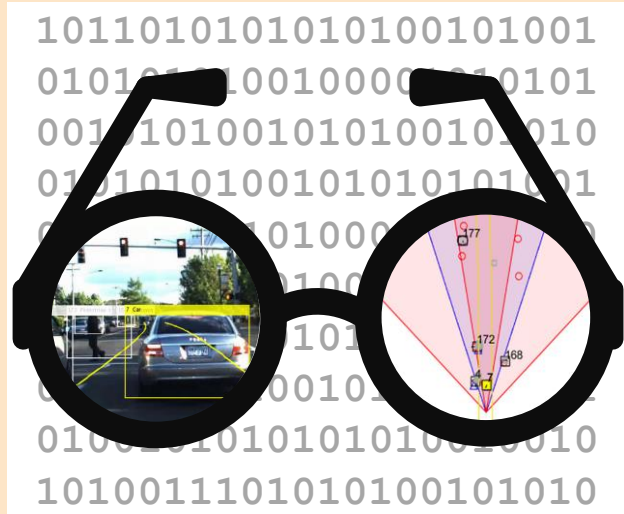


我怎样检测图
像中的目标？

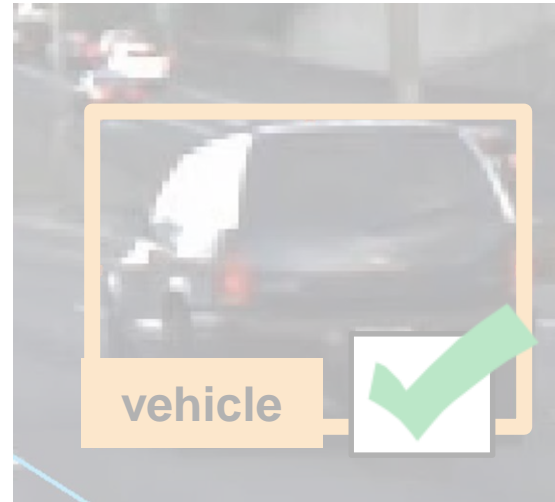


我怎样融合
多个检测结果？

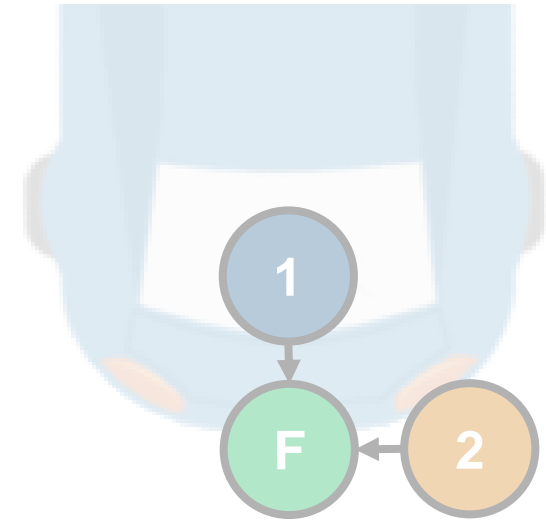
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？



我怎样检测图
像中的目标？



我怎样融合
多个检测结果？

自动驾驶中常使用的传感器

摄像头

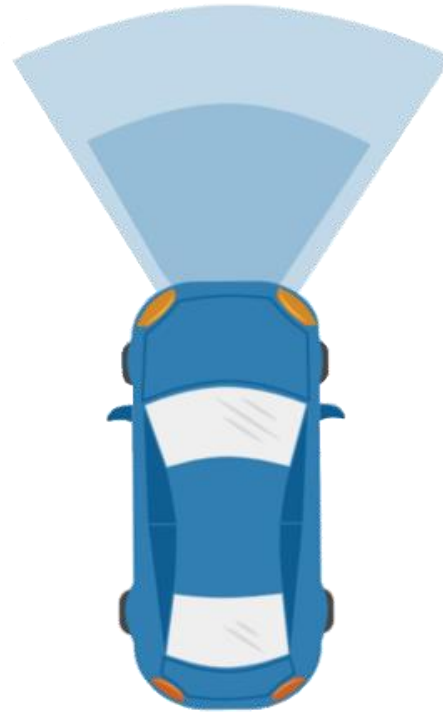
基于雷达的
目标检测

基于视觉的
目标检测

激光雷达

车道检测

惯性测量单元



自动驾驶中使用的传感器数据的例子

摄像头 (640 x 480 x 3)

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

视觉检测

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

车道检测

```
DetectionClass Left
Position IsValid: 1
Velocity Confidence: 3
Signature BoundaryType: 3
DetectionClass Right
Position IsValid: 1
Signature Confidence: 3
```

惯性测量单元

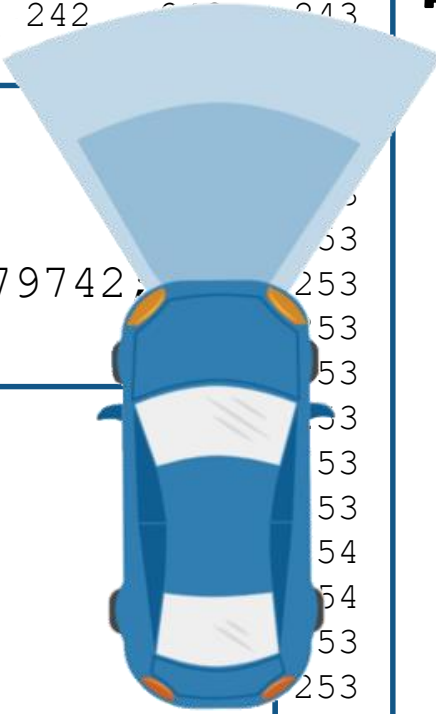
```
Timestamp: 1461634696379742
Velocity: 9.2795
YawRate: 0.0040
```

雷达检测

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

激光雷达 (47197 x 3)

```
DetectionTrackID TrackStart Position Velocity Amplitude
DetectionTrackID TrackStart Position Velocity Amplitude
-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
```



传感器数据可视化

Image Coordinates

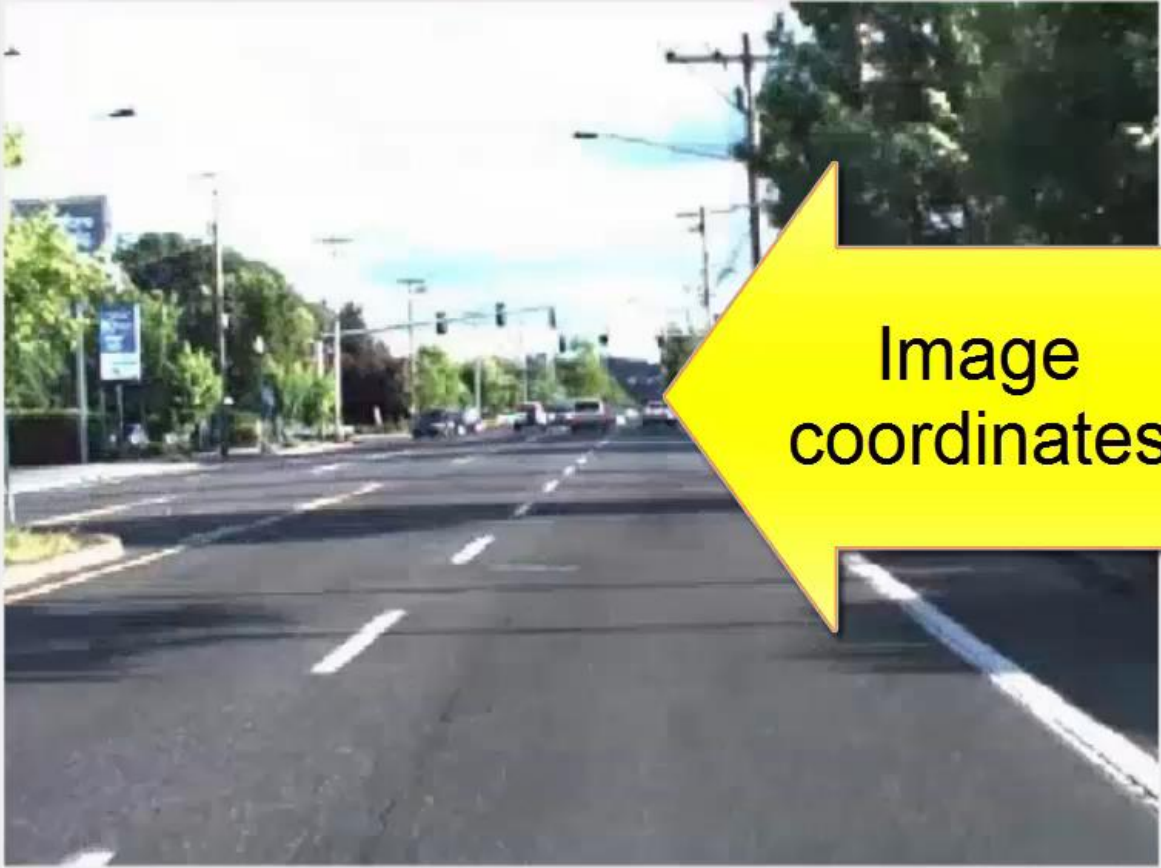
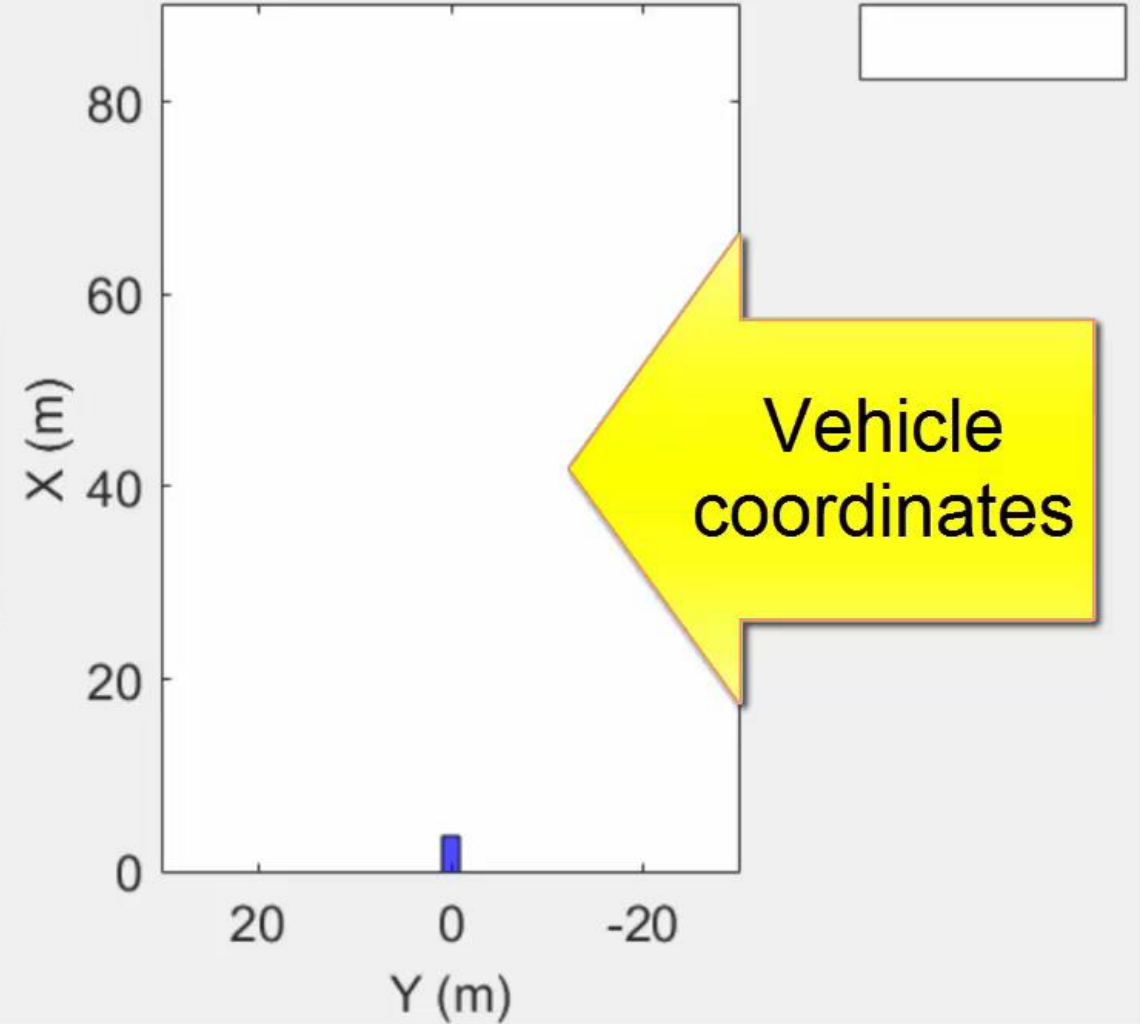


Image
coordinates

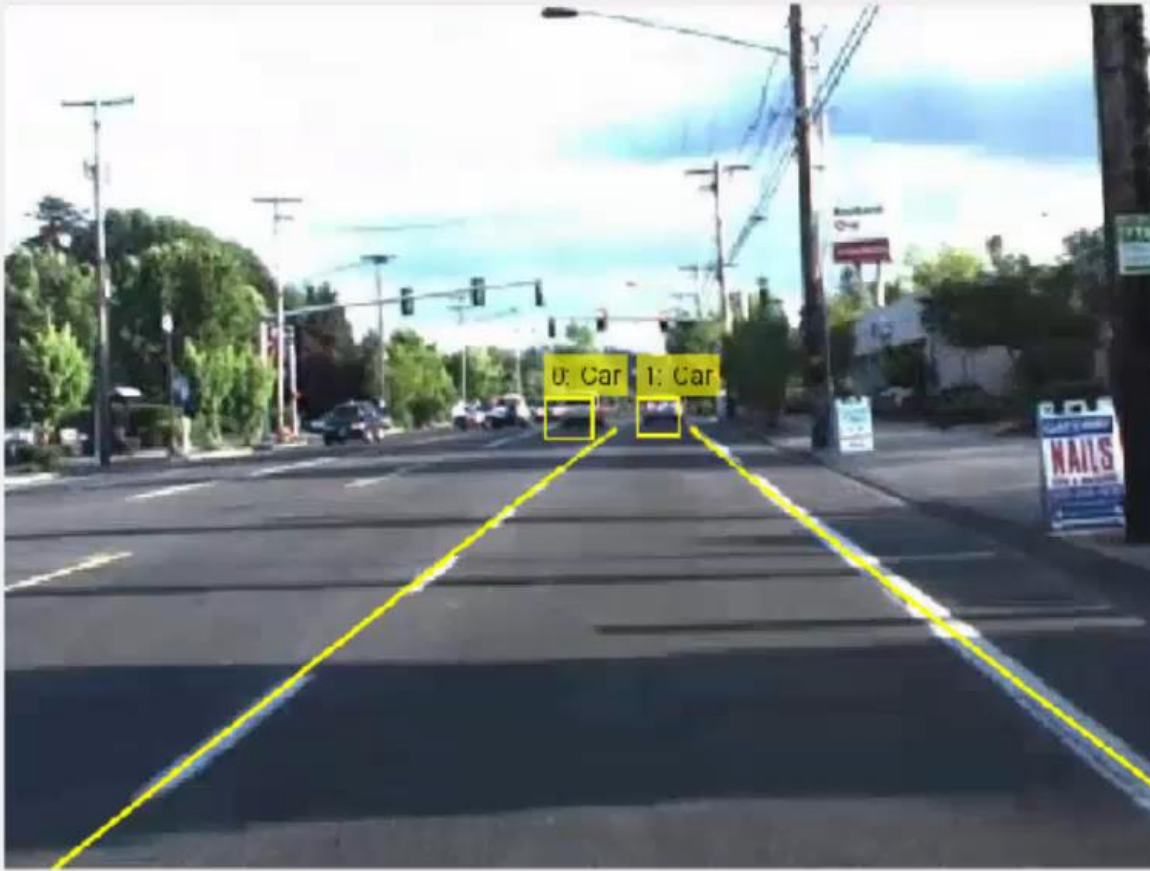
Vehicle Coordinates



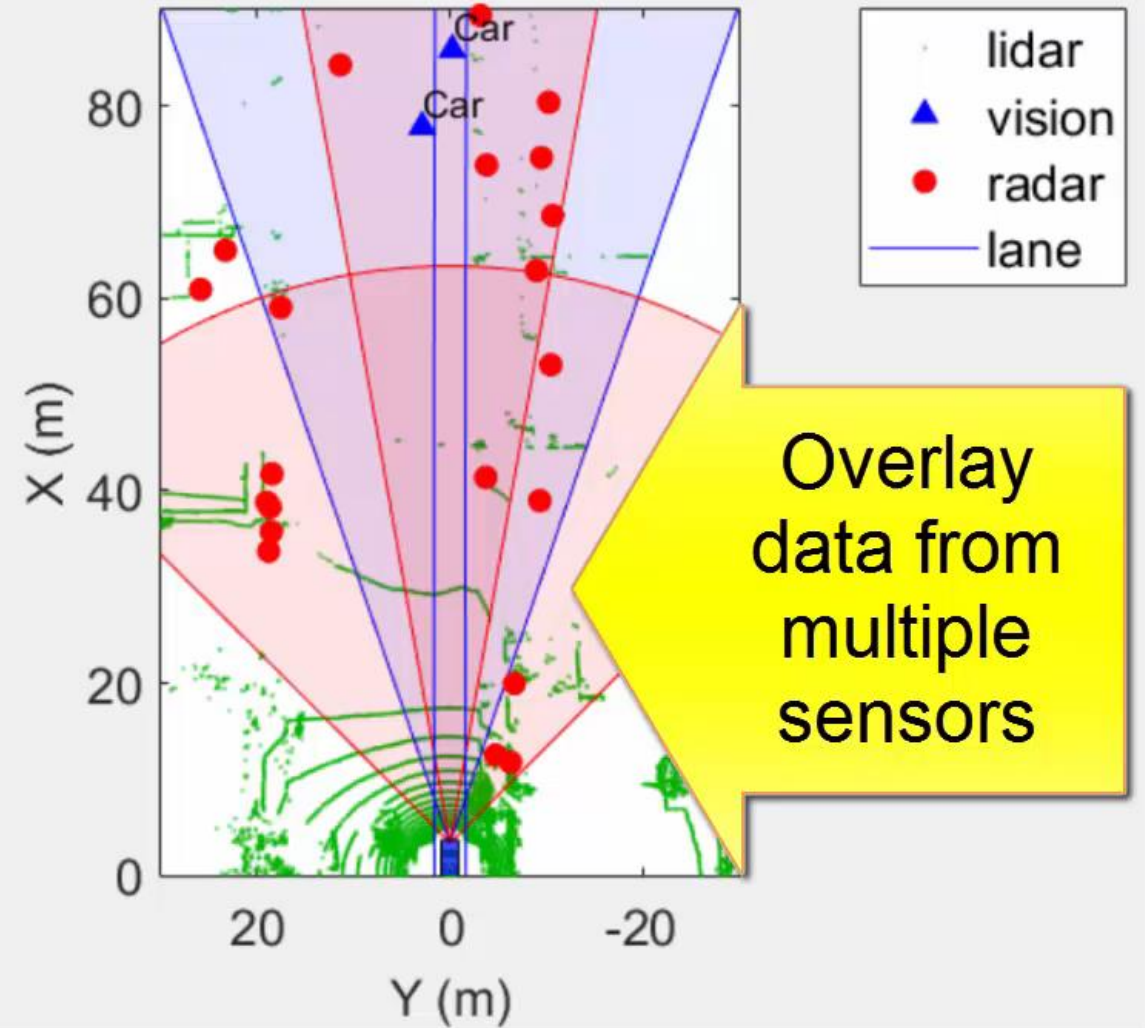
Vehicle
coordinates

传感器检测目标差异可视化

Image Coordinates



Vehicle Coordinates



探索已记录的车辆数据

- 导入 摄像头数据 和相应的 单摄像头参数

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

- 导入 传感器检测数据 和相应的 参数

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

- 导入 激光雷达点云数据

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


在图像坐标系中可视化

```
%% Specify time to inspect
currentTime = 6.55;
video.CurrentTime = currentTime;

%% Extract video frame
frame = video.readFrame;

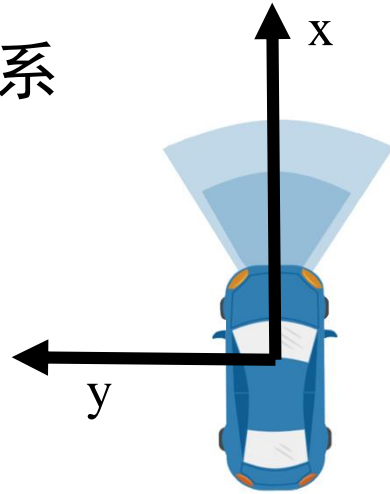
%% Plot image coordinates
ax1 = axes(...
    'Position', [0.02 0 0.55 1]);
im = imshow(frame, ...
    'Parent', ax1);
```



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

在车辆坐标系中可视化

- ISO 8855 车辆坐标系
 - 前向为正x
 - 左向为正y



```

%% 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');
  
```



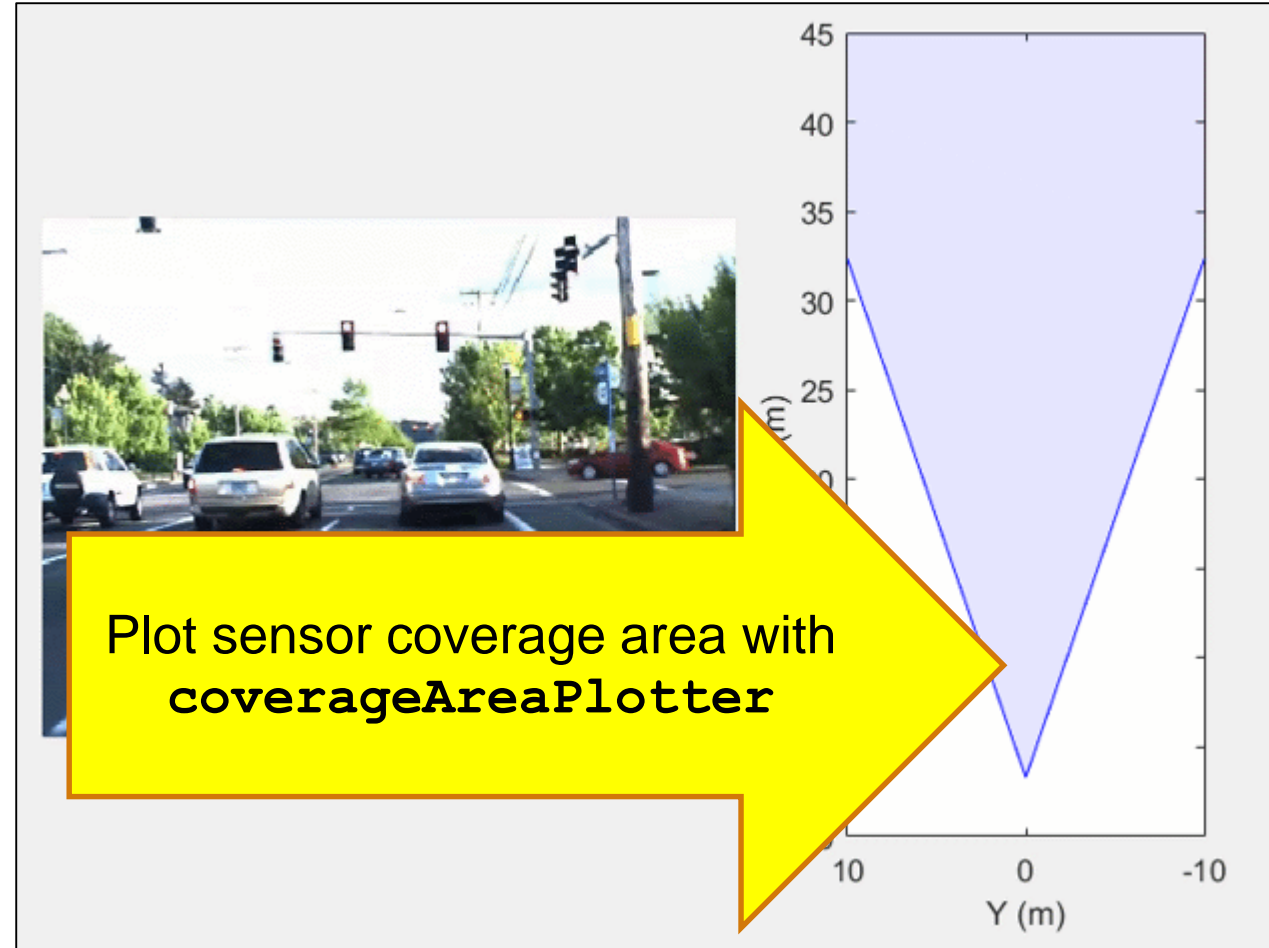
覆盖区域可视化(车辆坐标系)

```

%% 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

```



检测目标可视化 (车辆坐标轴)

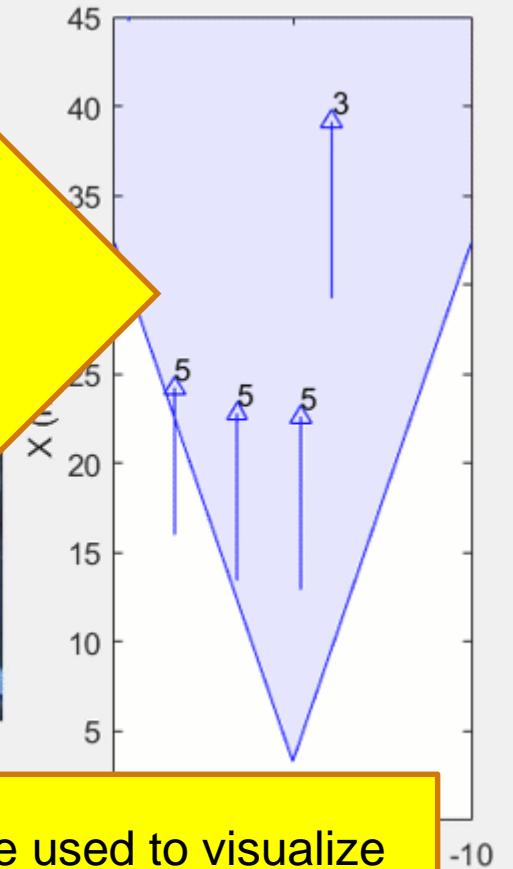
```

%% 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

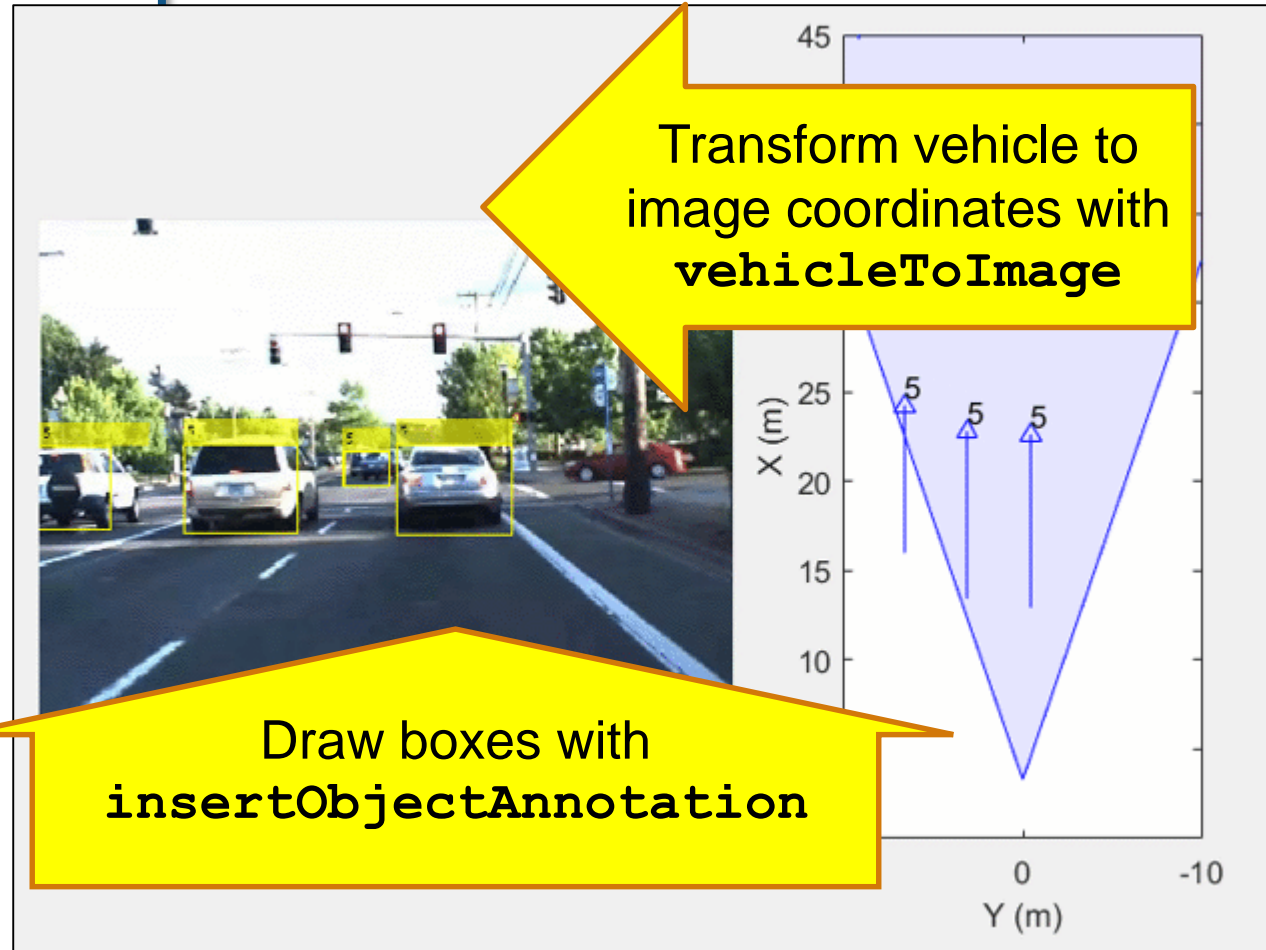
检测目标可视化 (图像坐标系)

```

%% 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;

```



车道线边界可视化 (车辆坐标)

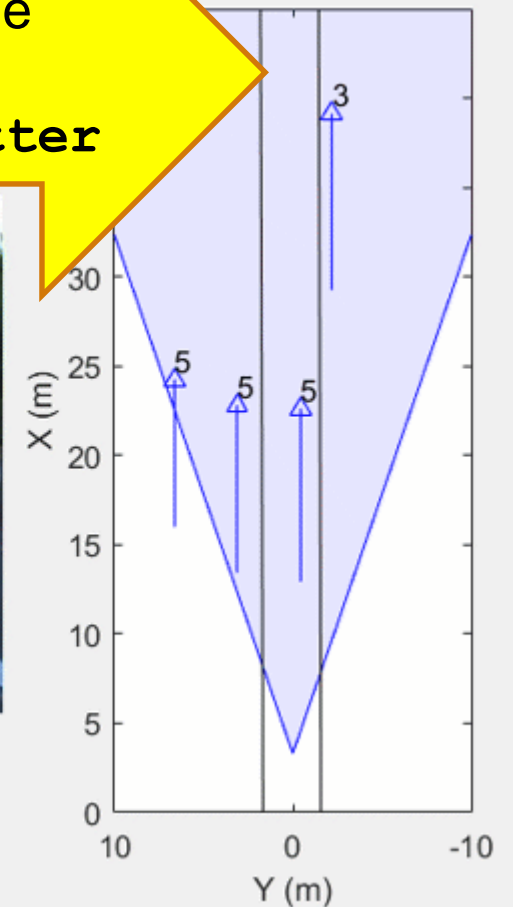
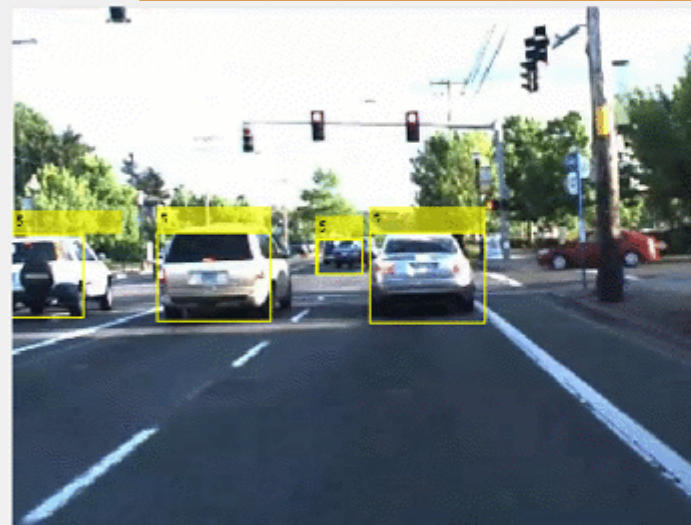
```

%% 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])

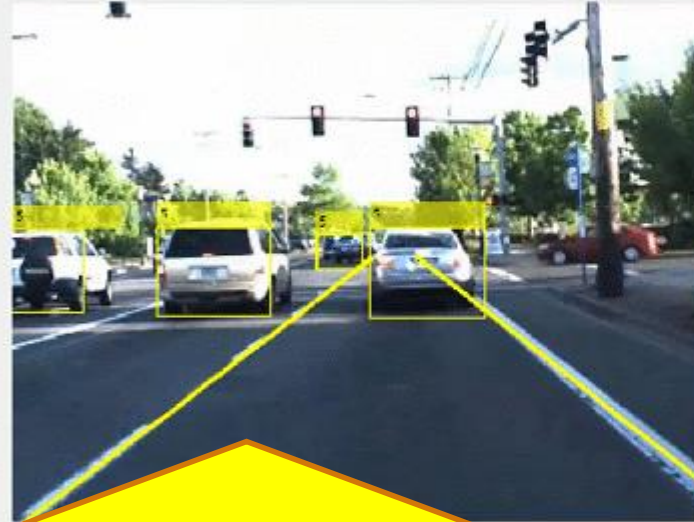
```

Plot lanes in vehicle coordinates with `laneBoundaryPlotter`

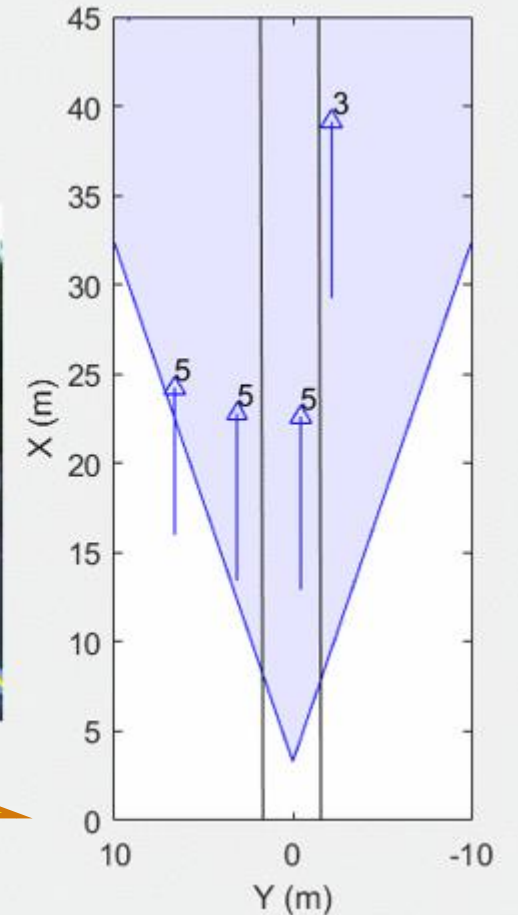


车道线边界可视化 (图像坐标系)

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



Plot lanes in image coordinates with `insertLaneBoundary`



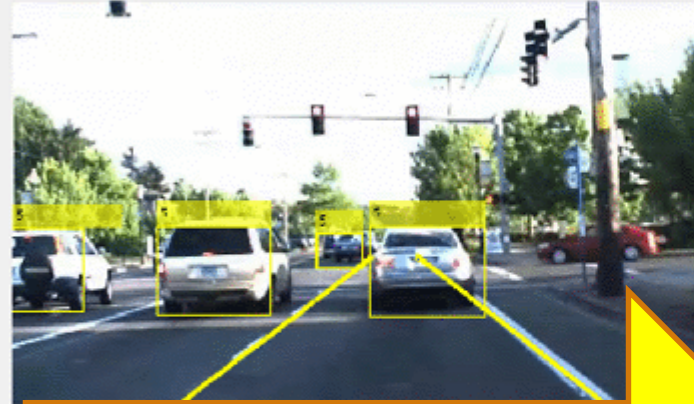
雷达检测结果可视化 (车辆坐标系)

```

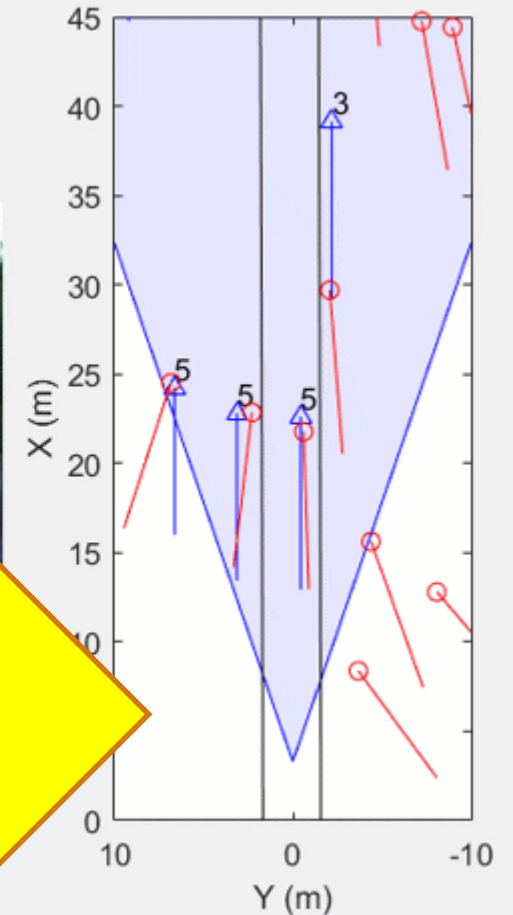
%% Create radar detection plotter
radarPlot = detectionPlotter(bep, ...
    'MarkerEdgeColor','red',...
    'Marker','o');

%% 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);

```



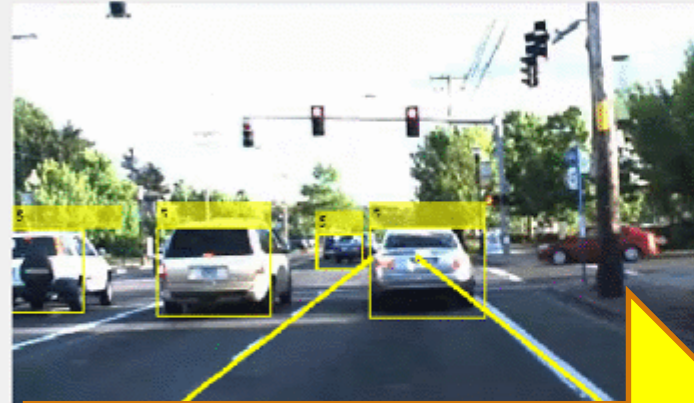
Plot radar detections just like vision detections with **detectionPlotter**



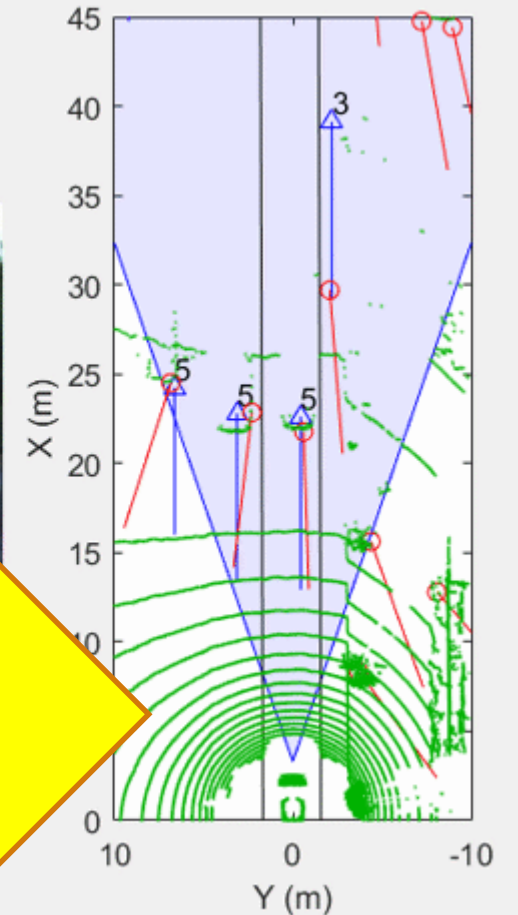
激光雷达点云可视化 (车辆坐标系)

```
% 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);
```

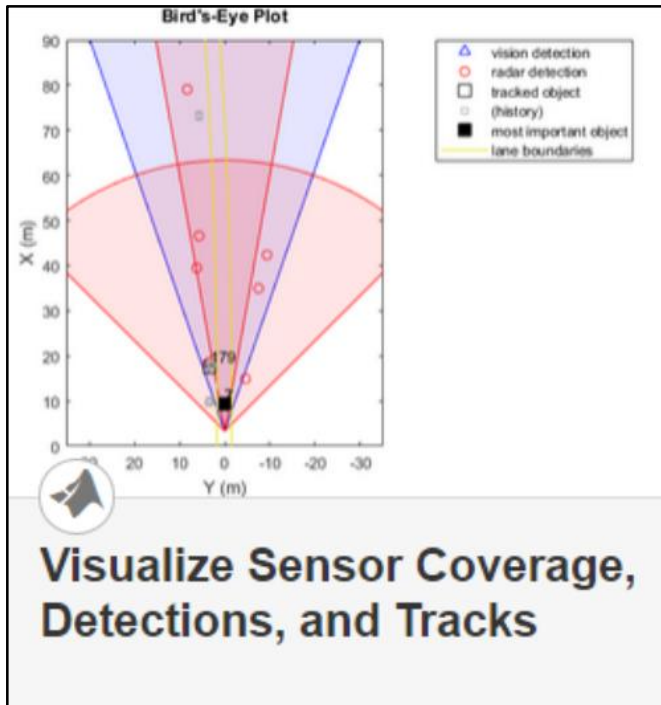


Plot lidar points just like
vision detections with
detectionPlotter

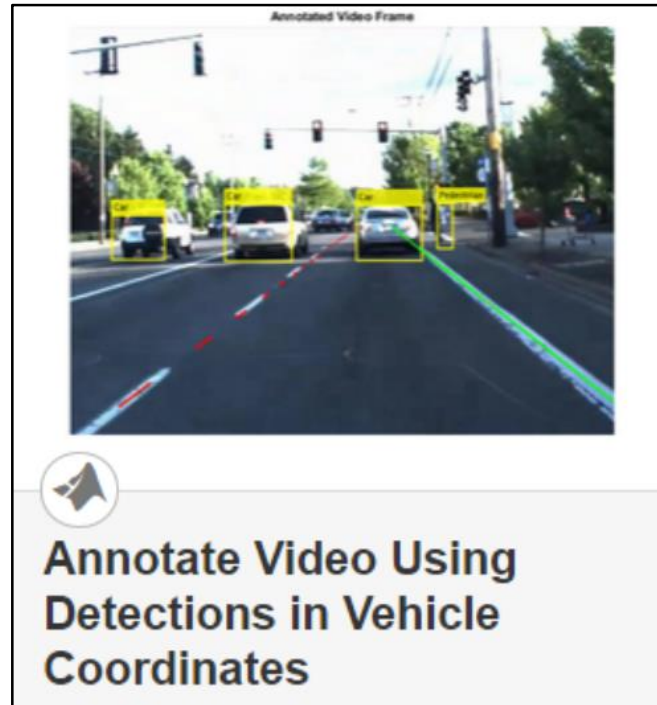


了解更多车辆数据可视化

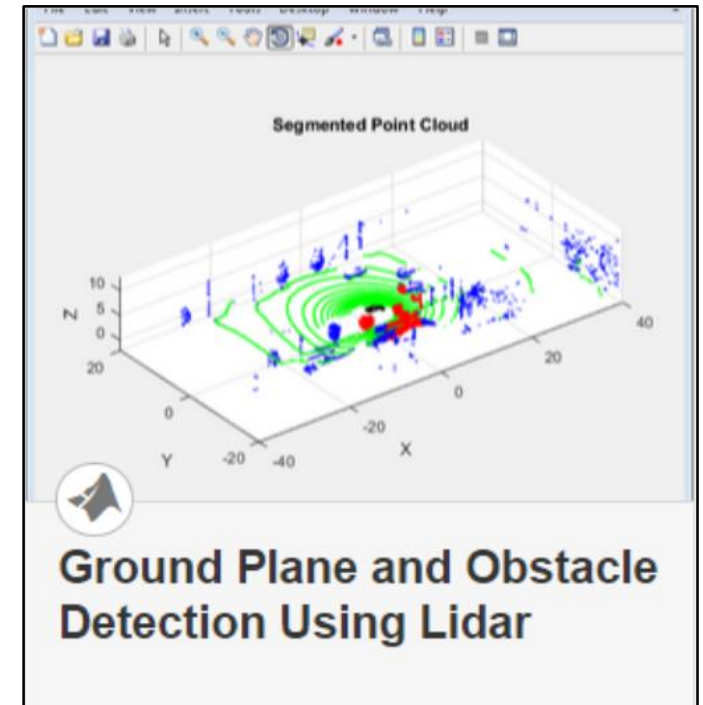
查看Automated Driving System Toolbox中的例子



- 在车辆坐标系中呈现检测目标
 - Vision & radar detector
 - Lane detectors
 - Detector coverage areas



- 车辆坐标系和图像坐标系转换

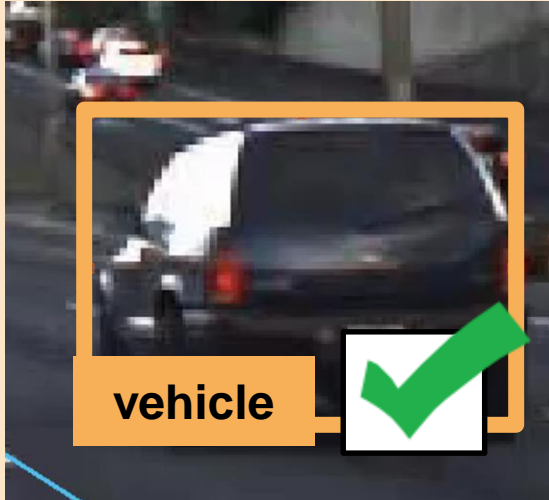


- 绘制点云数据

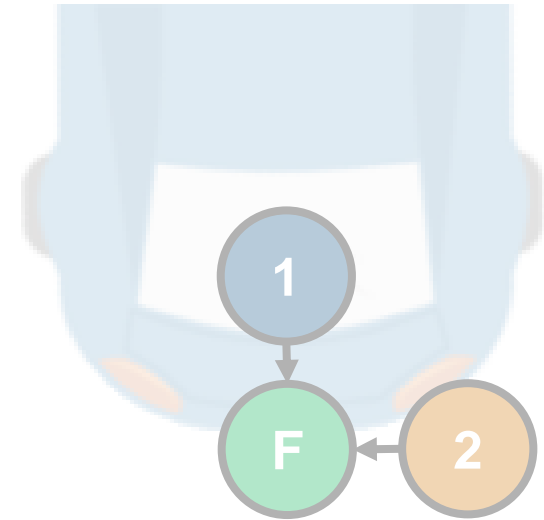
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？

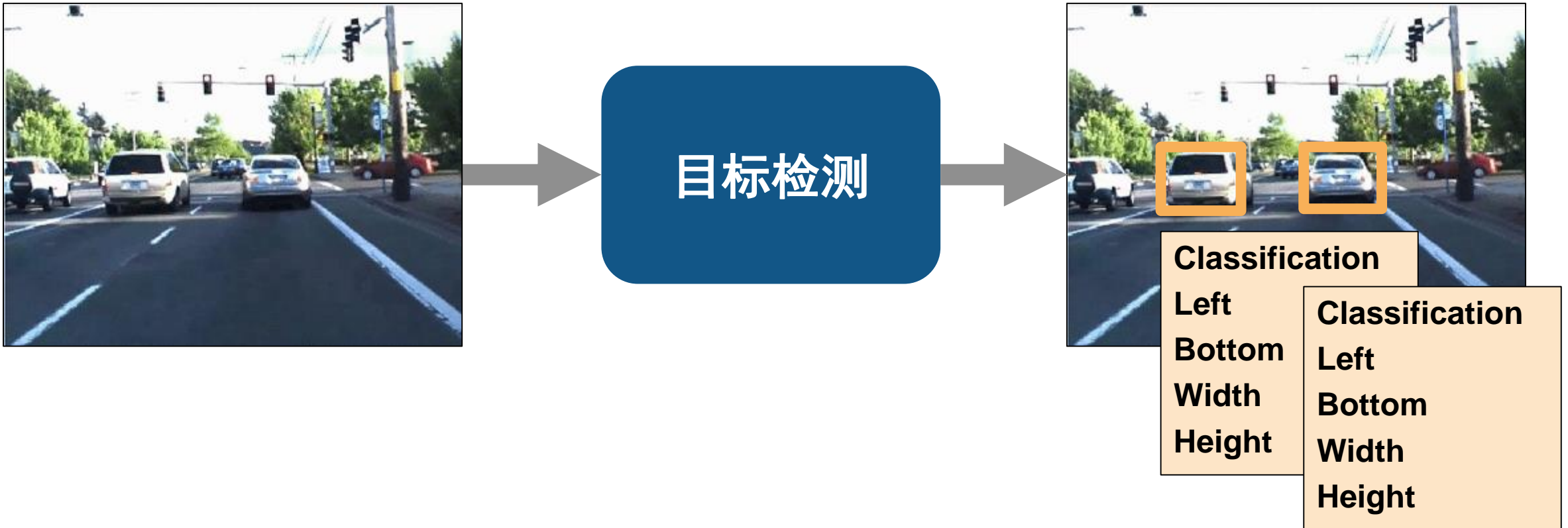


我怎样检测图
像中的目标？

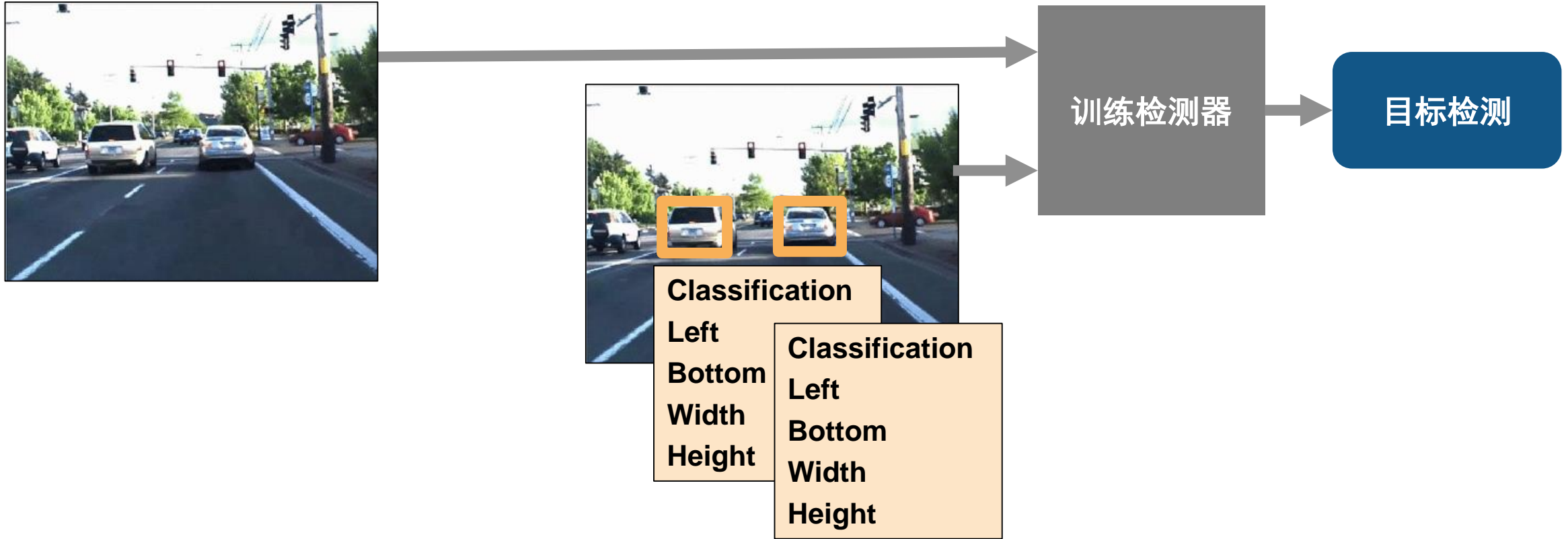


我怎样融合
多个检测结果？

我怎样检测图像中的目标？



基于真实值训练目标检测算法



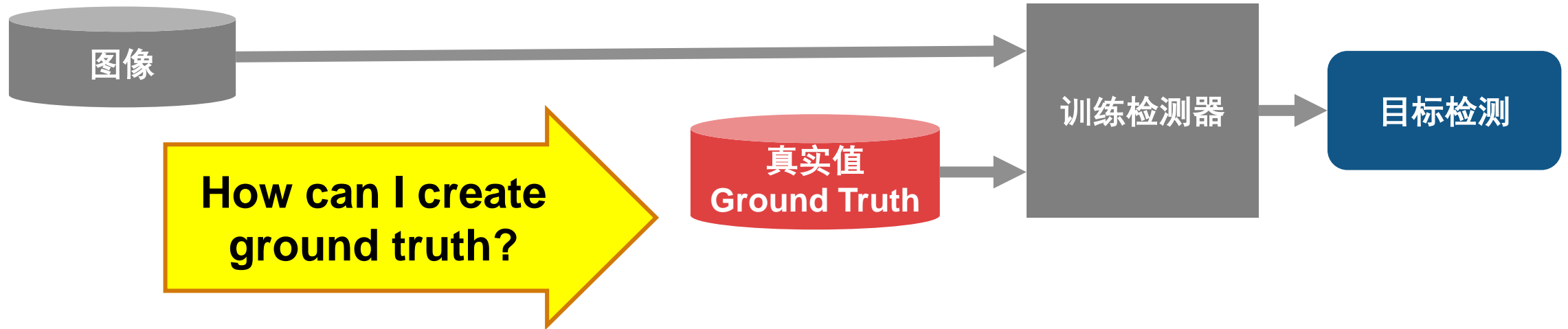
基于真实值训练目标检测算法



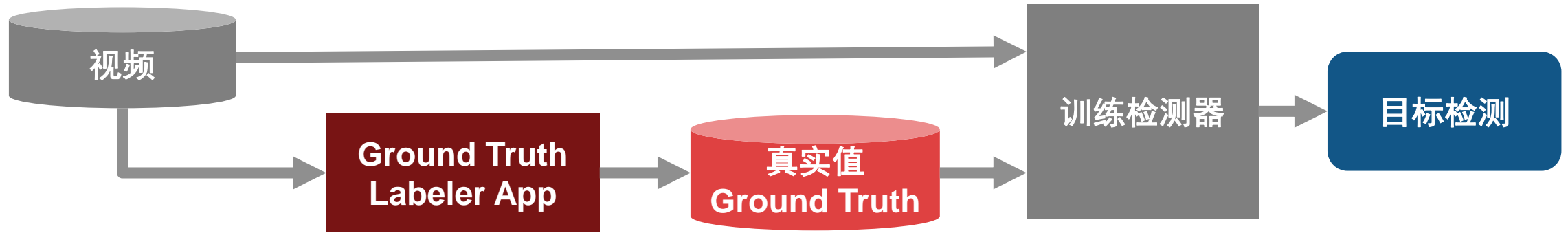
借助Computer Vision System Toolbox设计目标检测器

机器学习	Aggregate Channel Feature	<code>trainACFObjectDetector</code>
	Cascade	<code>trainCascadeObjectDetector</code>
深度学习	R-CNN (Regions with Convolutional Neural Networks)	<code>trainRCNNObjectDetector</code>
	Fast R-CNN	<code>trainFastRCNNObjectDetector</code>
	Faster R-CNN	<code>trainFasterRCNNObjectDetector</code>

为训练检测器提供真实值

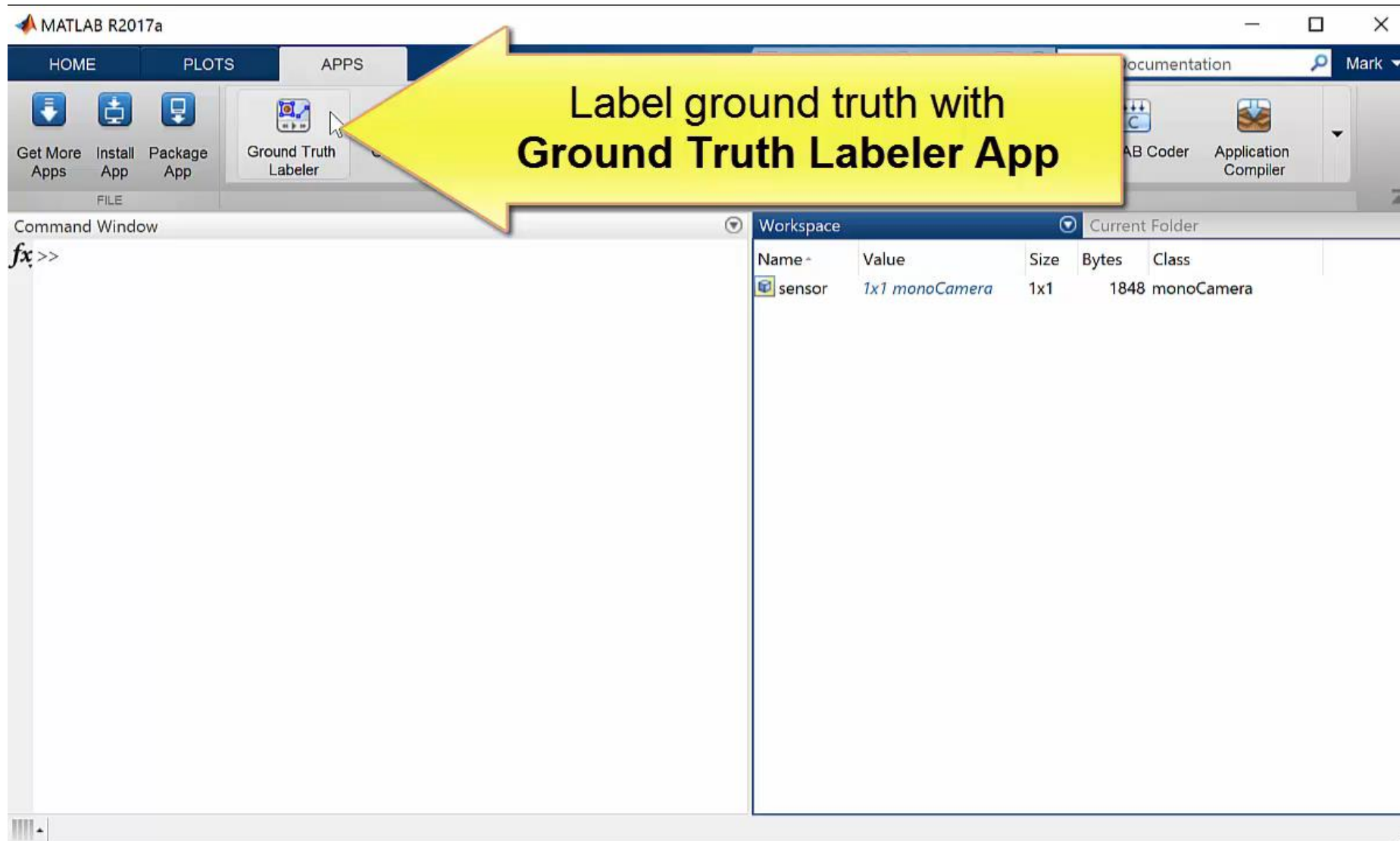


为训练检测器提供真实值



手工标注目标的真实值

with Ground Truth Labeling App



MATLAB R2017a

HOME PLOTS APPS

Get More Apps Install App Package App

Ground Truth Labeler

Label ground truth with
Ground Truth Labeler App

Documentation Mark

AB Coder Application Compiler

Command Window

```
fx >>
```

Workspace

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

在手工标注的帧之间自动标注 with temporal interpolator

The screenshot displays the 'Ground Truth Labeler' software interface. The 'Algorithm' dropdown menu is open, showing three options: 'ACF Vehicle Detector', 'Point Tracker', and 'Temporal Interpolator'. The 'Temporal Interpolator' option is highlighted, and a yellow callout box points to it with the text: 'Interpolate regions between frames with Temporal Interpolator'. The interface also shows a 'ROI Label Definition' panel with a 'Car' label, a 'Scene Label Definition' panel with 'SunnyDay' and 'LaneChange' labels, and a video player at the bottom with a timeline showing 'Start Time', 'Current', 'End Time', and 'Max Time'.

Ground Truth Labeler

LABEL

Load Save Import Labels ROI Zoom In Zoom Out Pan Default Layout Show ROI Labels Show Scene Labels

Algorithm:

Select Algorithm

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.

Add Algorithm Refresh list

ROI Label Definition

05_high

Define new ROI label

Car

Scene Label Definition

Define New Scene Label

Current Frame Add Label

Time Interval Remove Label

SunnyDay

LaneChange

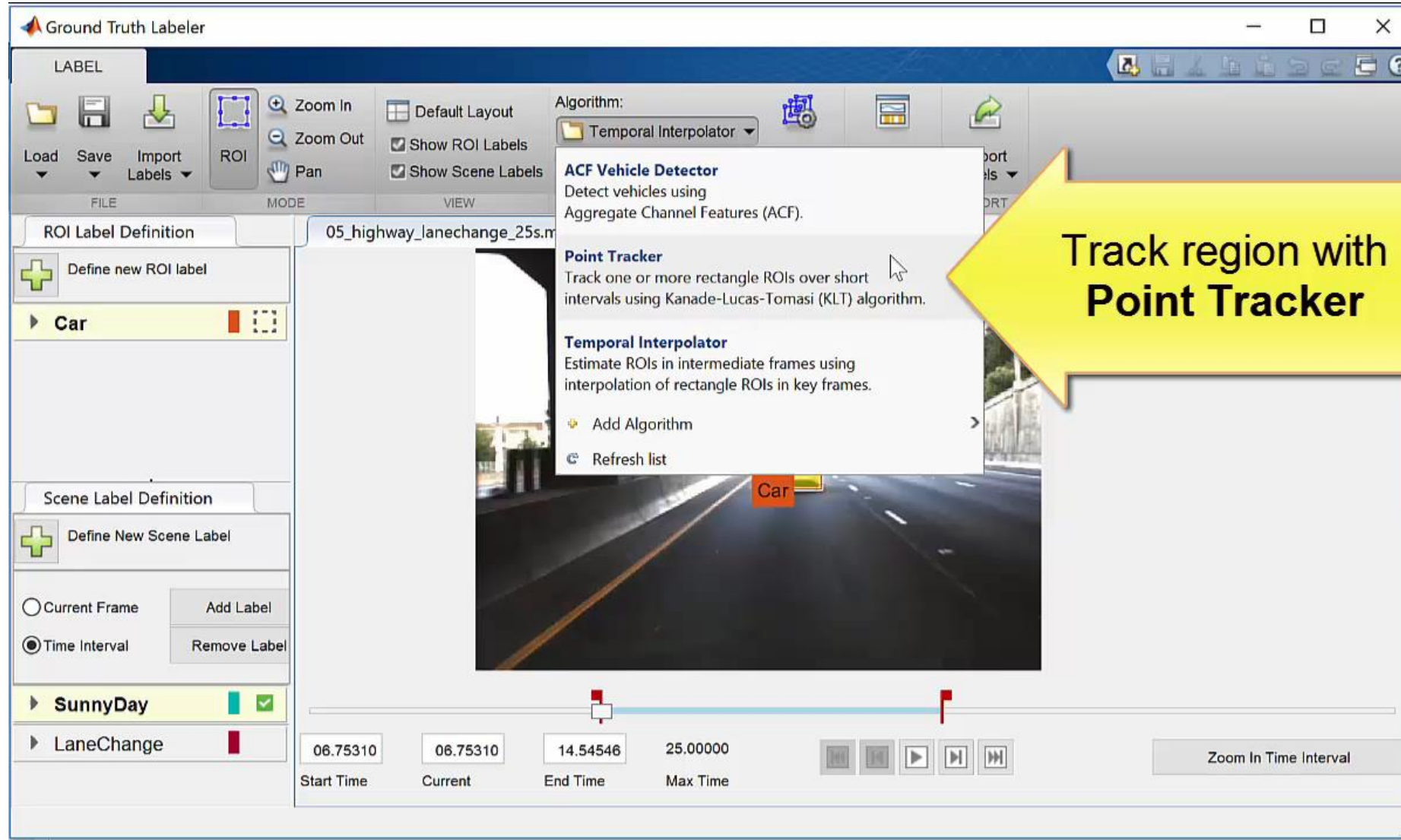
00.00000 00.05000 16.48607 25.00000

Start Time Current End Time Max Time

Zoom In Time Interv

Interpolate regions between frames with Temporal Interpolator

在手工标注帧的基础上自动标注 with point tracker



自动化检测车辆目标的真实值 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 red bounding box around a car, labeled "Car". A yellow callout box points to the "ACF Vehicle Detector" algorithm in the "Algorithm:" list, with the text "Detect initial regions with Vehicle Detector".

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: Add Algorithm, Refresh list

ROI Label Definition

- Define new ROI label
- Car

Scene Label Definition

- Define New Scene Label
- Current Frame (Add Label)
- Time Interval (Remove Label)
- SunnyDay
- LaneChange

Timeline

Start Time	Current	End Time	Max Time
14.54546	14.54546	21.12013	25.00000

Buttons: Zoom In Time Interval

以MATLAB时间表方式导出标注区域

The screenshot displays the Ground Truth Labeler application window. A large 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 left sidebar contains 'ROI Label Definition' and 'Scene Label Definition' sections. The bottom timeline shows a current time of 14.55000, with start, end, and max time markers at 00.00000, 25.00000, and 25.00000 respectively. Playback controls and a 'Zoom In Time Interval' button are also visible.

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

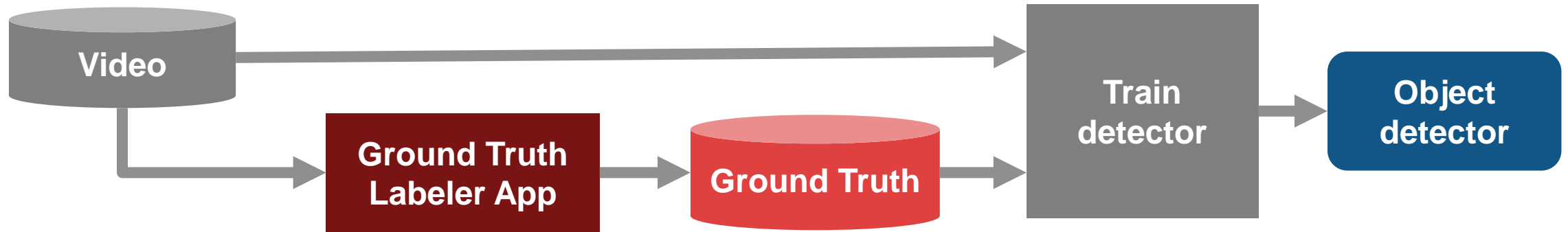
Explore labels by viewing label summary

View Label Summary

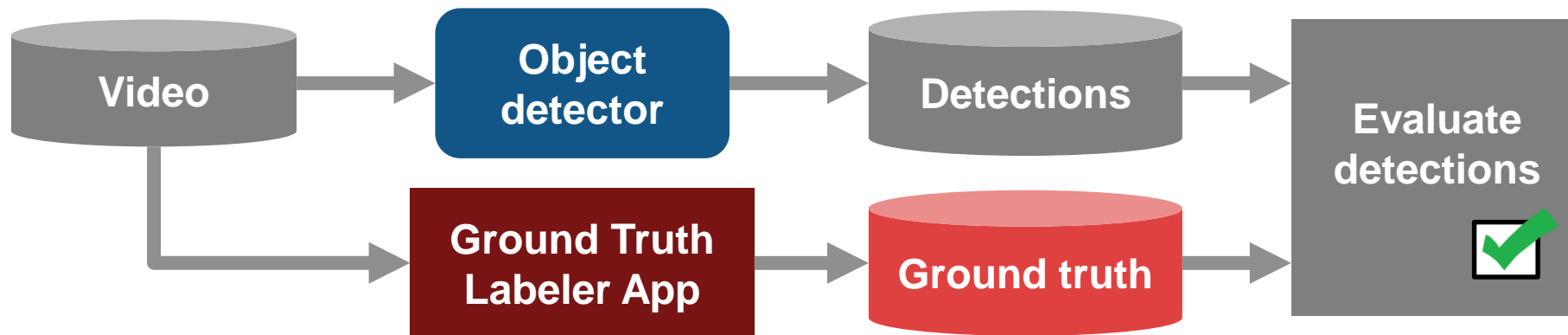
Export Labels

SUMMARY EXPORT

Ground truth labeling to train detectors



Ground truth labeling to evaluate detectors



定制化真实值标注应用程序

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

FILE MODE VIEW AUTOMATE LABELING SUMMARY EXPORT

ROI Label Definition

01_city_c2s_fcw_10s.mp4

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.

00.00000 09.00000 10.20000 10.20000
 Start Time Current End Time Max Time

The screenshot displays the Ground Truth Labeler software interface. The main window shows a video frame titled '01_city_c2s_fcw_10s.mp4' with several objects labeled: two cars (one white, one silver), a pedestrian, and three stoplights. The labels are color-coded: Car (orange), Pedestrian (yellow), StopLight (green), and Lane (blue). The interface includes a menu bar with options like Load, Save, Import Labels, ROI, Zoom In, Zoom Out, Pan, Default Layout, Show ROI Labels, Show Scene Labels, Algorithm, Automate, View Label Summary, and Export Labels. On the left, there are panels for 'ROI Label Definition' and 'Scene Label Definition'. The 'ROI Label Definition' panel lists 'Car', 'Pedestrian', 'StopLight', and 'Lane' with corresponding color swatches and icons. The 'Scene Label Definition' panel has options for 'Current Frame' and 'Time Interval', with 'Add Label' and 'Remove Label' buttons. At the bottom, a timeline shows the video's duration from 00.00000 to 10.20000, with a current time of 09.00000. Playback controls and a 'Zoom' button are also visible.

定制化真实值标注应用程序

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, Select Algorithm, Configure Automation, 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 a red box highlights the "Custom Reader" option in the list. A large yellow arrow points from the text "Add custom image reader with `groundTruthDataSource`" to the "Custom Reader" option. The main workspace shows a video frame with ground truth labels for "Car", "Pedestrian", and "Lane". A timeline at the bottom indicates the current time is 09.00000, with start and end times at 00.00000 and 10.20000 respectively. The interface also includes a "Scene Label Definition" section with a "Define New Scene Label" button and "Add Label" and "Remove Label" buttons.

Ground Truth Labeler - gtlCustomizations

LABEL

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

Algorithm: Configure Automation

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`

定制化真实值标注应用程序

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

- Top Bar:** Includes 'LABEL' and 'Algorithm:' tabs, along with icons for file operations and a help icon.
- Left Panel:** Contains 'ROI Label Definition' and 'Scene Label Definition' sections. Under 'ROI Label Definition', there are entries for 'Car', 'Pedestrian', 'StopLight', and 'Lane'. Under 'Scene Label Definition', there are options for 'Current Frame' and 'Time Interval'.
- Center Panel:** Shows a video frame with bounding boxes for 'Car' and 'Pedestrian'. A timeline at the bottom indicates 'Start Time' (00.00000), 'Current' (09.00000), 'End Time', and 'Max Time'.
- Right Panel:** Features an 'Algorithm:' dropdown menu. A list of algorithms is displayed, including 'Point Tracker', 'Temporal Interpolator', and 'ACF Vehicle Detector'. A red box highlights the 'Create New Algorithm' and 'Import Algorithm' options.

A large yellow callout box with a black border is overlaid on the bottom right of the interface, containing the text:

Add custom automation algorithm
`driving.automation.AutomationAlgorithm`

定制化真实值标注应用程序

The screenshot displays the **Ground Truth Labeler - gtlCustomizations** application window. The interface includes a menu bar with options like **FILE**, **MODE**, and **VIEW**. A toolbar contains icons for **Load**, **Save**, **Import Labels**, **ROI**, **Zoom In**, **Zoom Out**, **Pan**, **Default Layout**, **Show ROI Labels**, **Show Scene Labels**, **Algorithm**, **Automate**, **View Label**, and **Export**.

On the left, the **ROI Label Definition** panel lists labels: **Car**, **Pedestrian**, **StopLight**, and **Lane**. The **Scene Label Definition** panel includes options for **Current Frame** and **Time Interval**, with **Add Label** and **Remove Label** buttons.

The main workspace shows a video frame with bounding boxes for **Car**, **Pedestrian**, and **Lane**. A yellow callout box with the text **Add connection to other tools with driving.connector.Connector** points to the **ROI** tool icon in the toolbar.

At the bottom, a timeline shows **Start Time** (00.00000), **Current** (09.00000), **End Time** (10.20000), and **Max Time** (10.20000). Playback controls and a **Zoom** button are also visible.

On the right, a separate window titled **Figure 1: Point Cloud Pla...** displays a 3D point cloud visualization of the scene, with a red border around it.

了解更多在图像中检测目标

查看Automated Driving System Toolbox中的例子

Define Ground Truth Data for Video or Image Sequences

- 用Ground Truth Labeler App 标注检测结果

Automate Ground Truth Labeling of Lane Boundaries

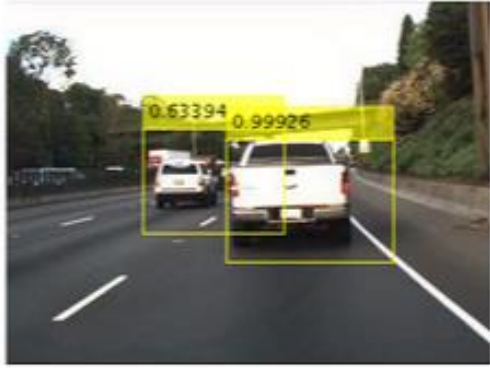
- 为车道线检测 加入自动化算法

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

- 为Ground Truth Labeler App 连接扩展功能

了解更多在图像中检测目标

查看Automated Driving System Toolbox中的例子



Train a Deep Learning Vehicle Detector

- **训练目标检测器**
应用深度学习和机器学习技术



Track Pedestrians from a Moving Car

- **探索预先训练好的行人检测器**



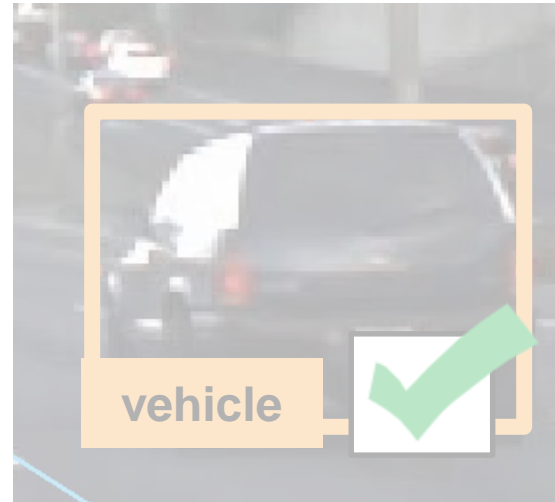
Visual Perception Using Monocular Camera

- **考察车道检测器**
根据摄像头传感器模型转换坐标系

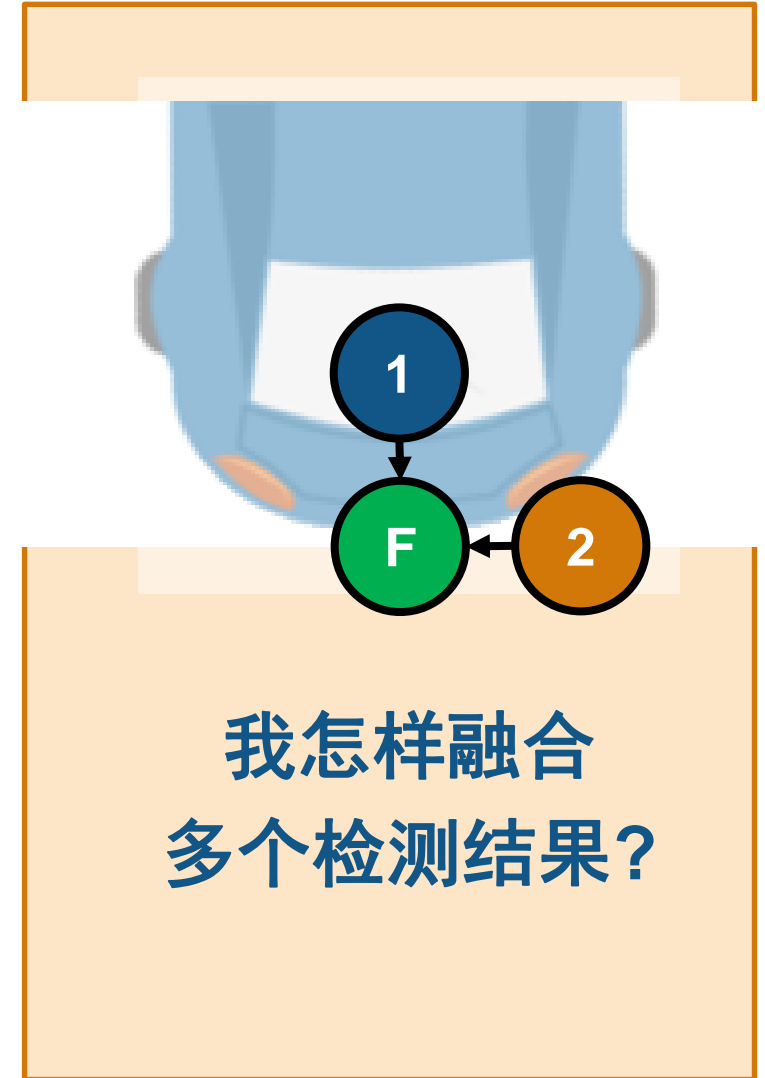
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？

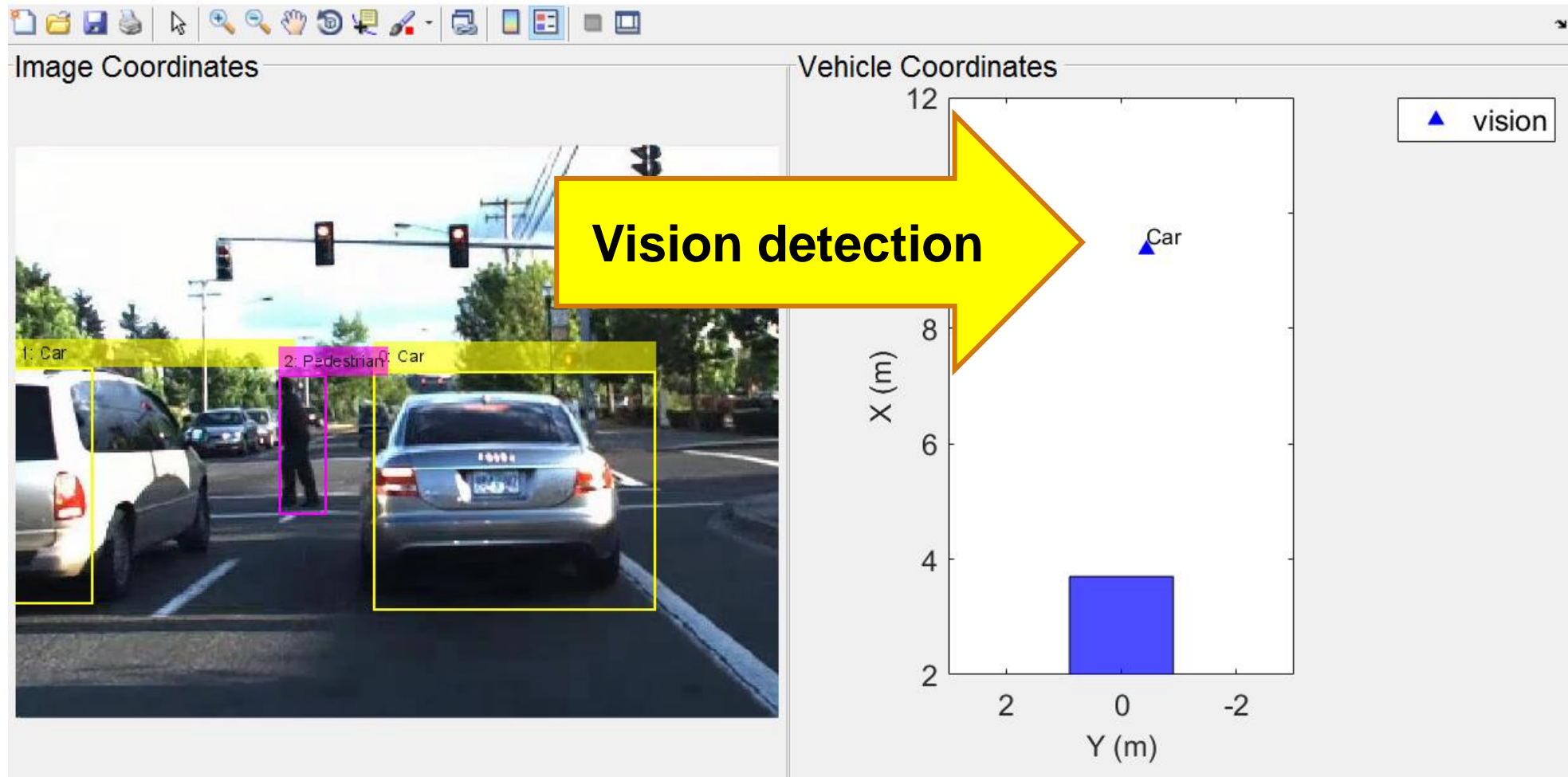


我怎样检测图
像中的目标？

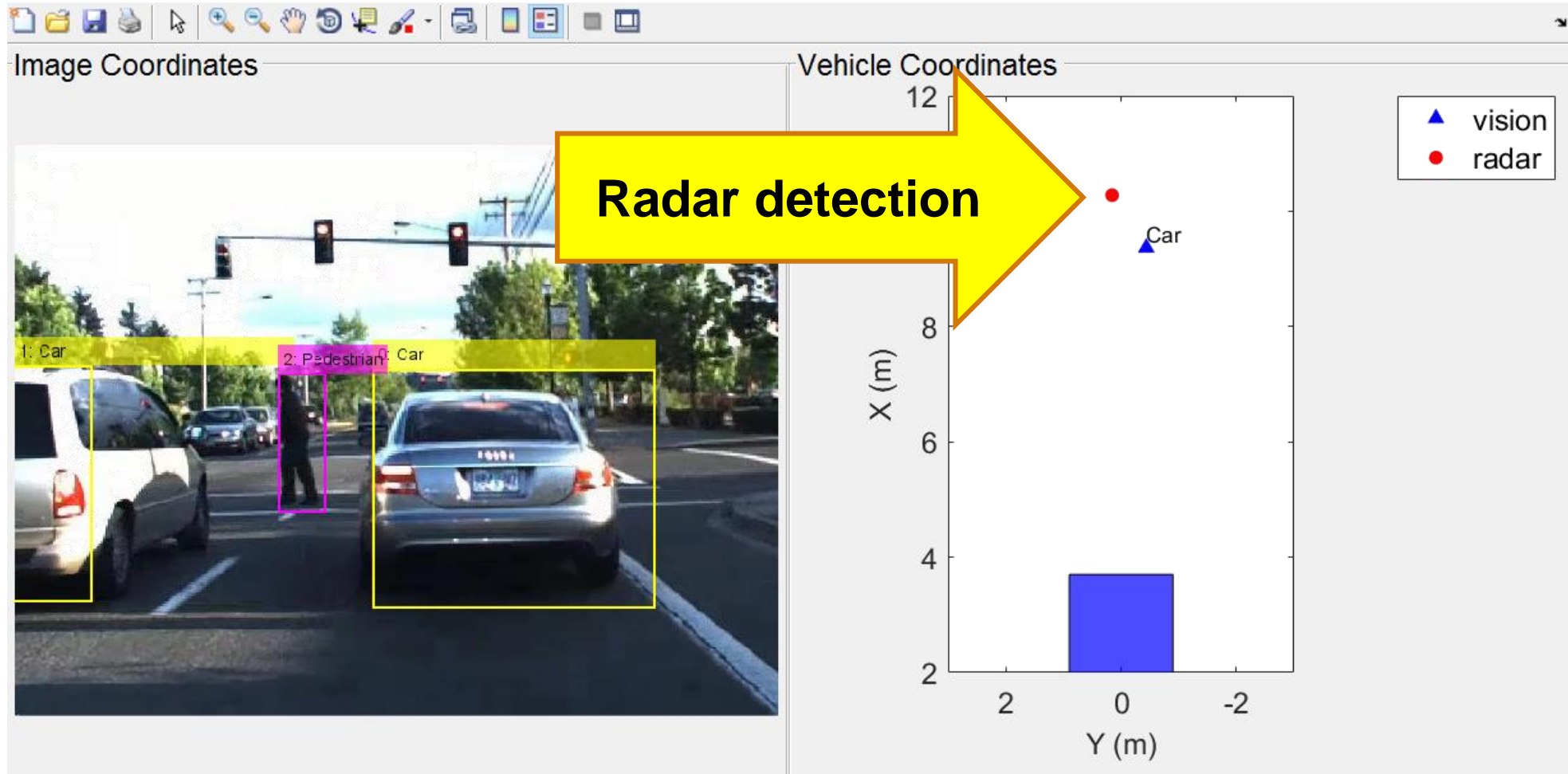


我怎样融合
多个检测结果？

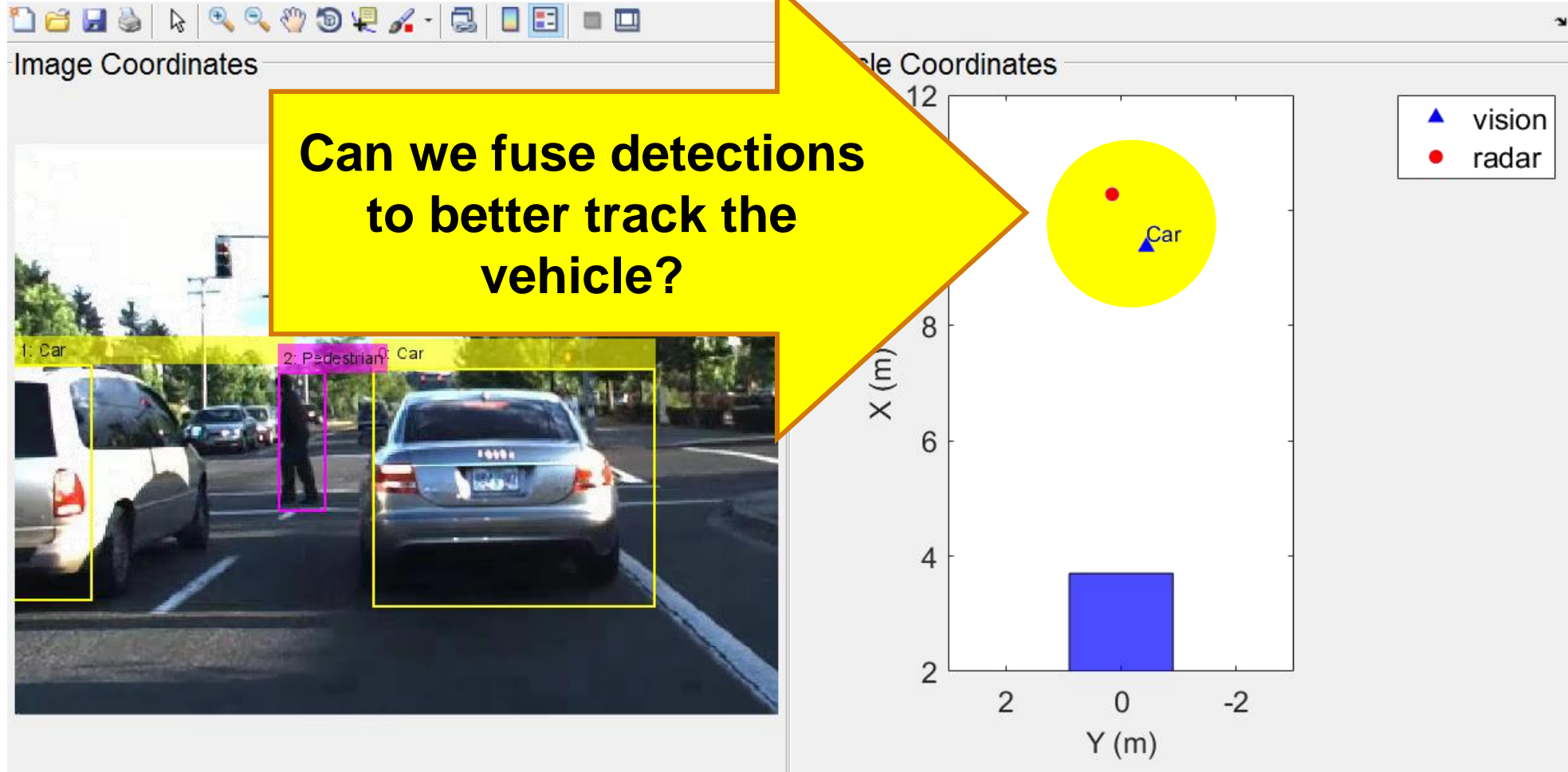
雷达和视觉检测车辆的例子



雷达和视觉检测车辆的例子

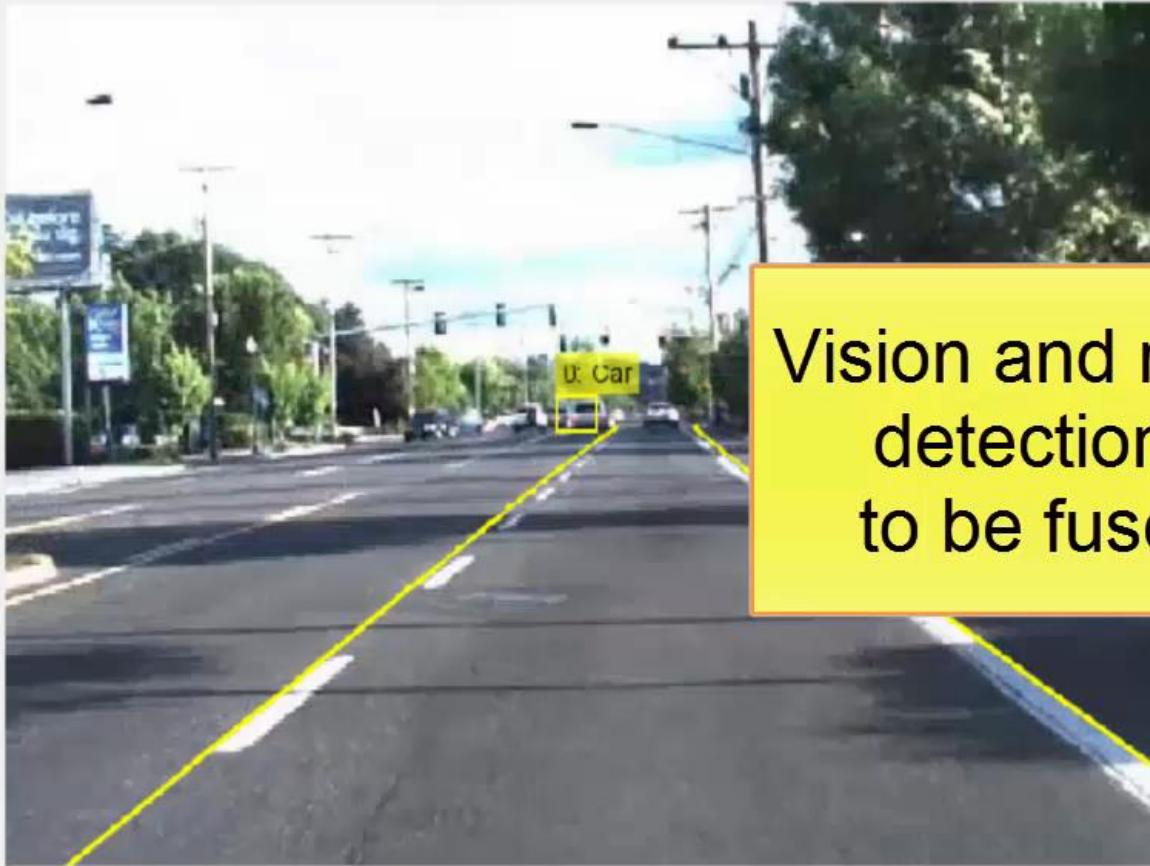


雷达和视觉检测车辆的例子

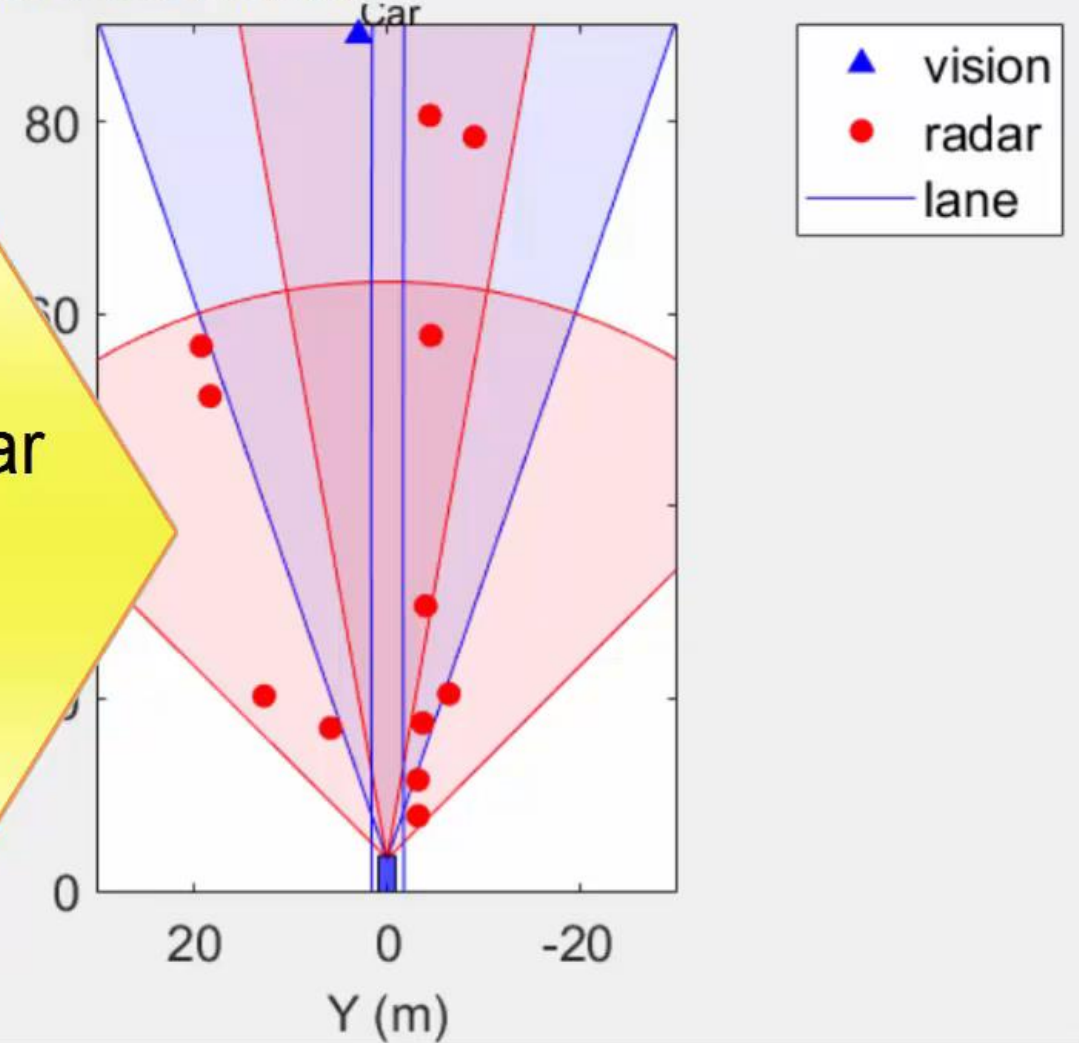


用多目标跟踪器融合检测目标

Image Coordinates



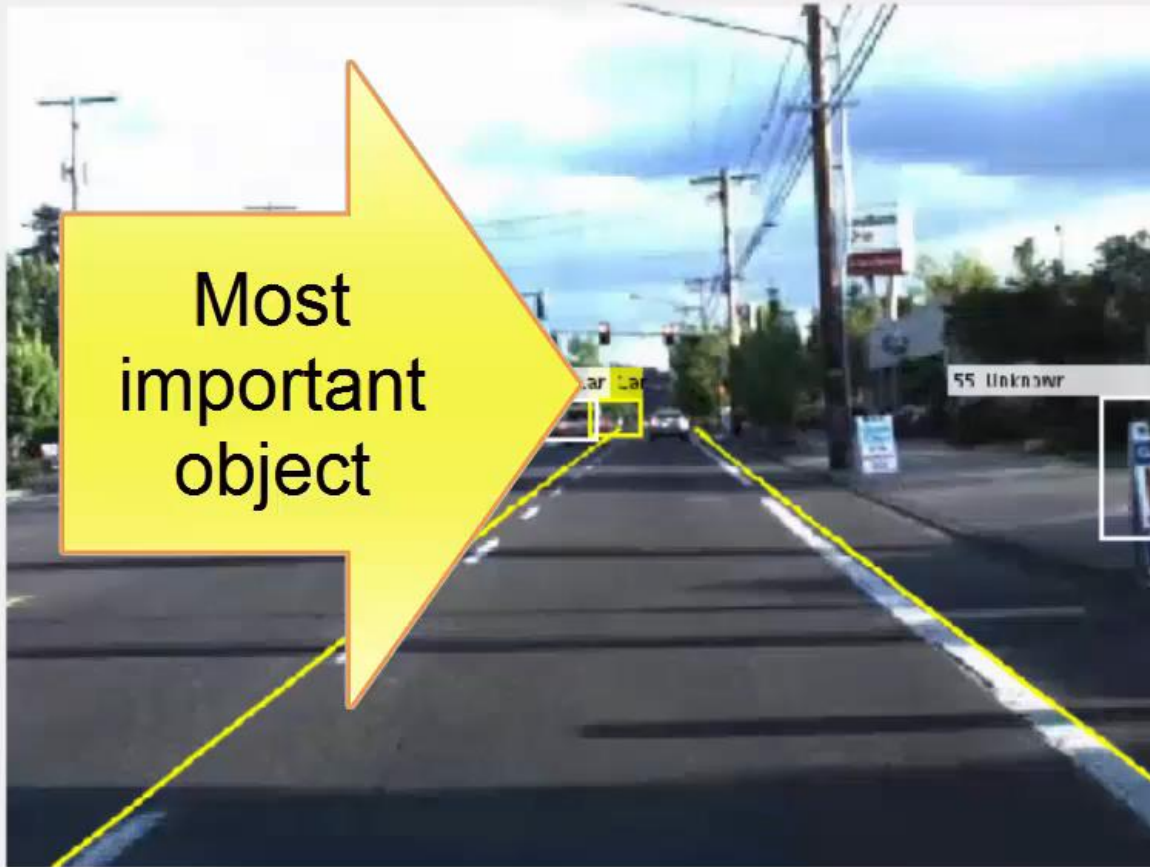
Vehicle Coordinates



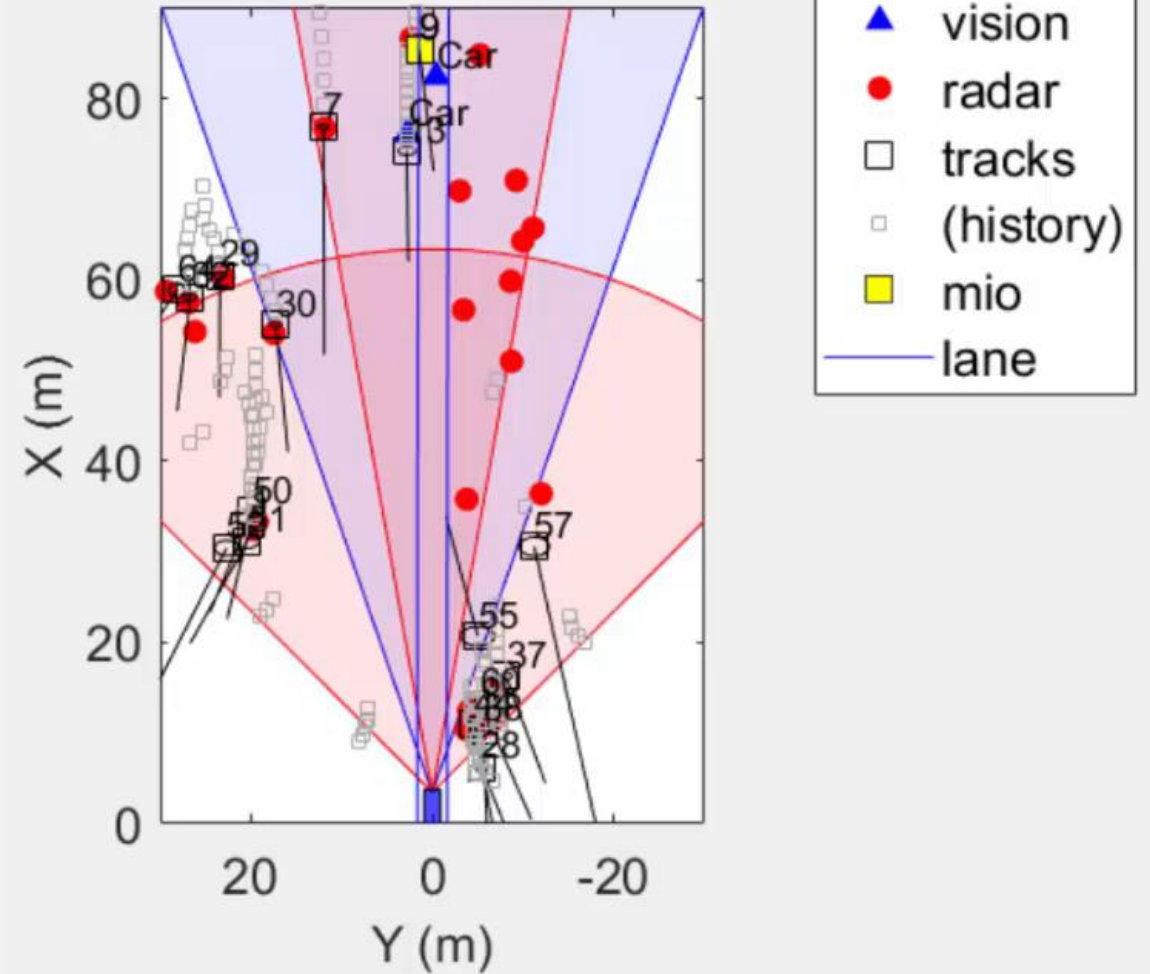
Vision and radar
detections
to be fused

将跟踪器集成到更上层的算法中

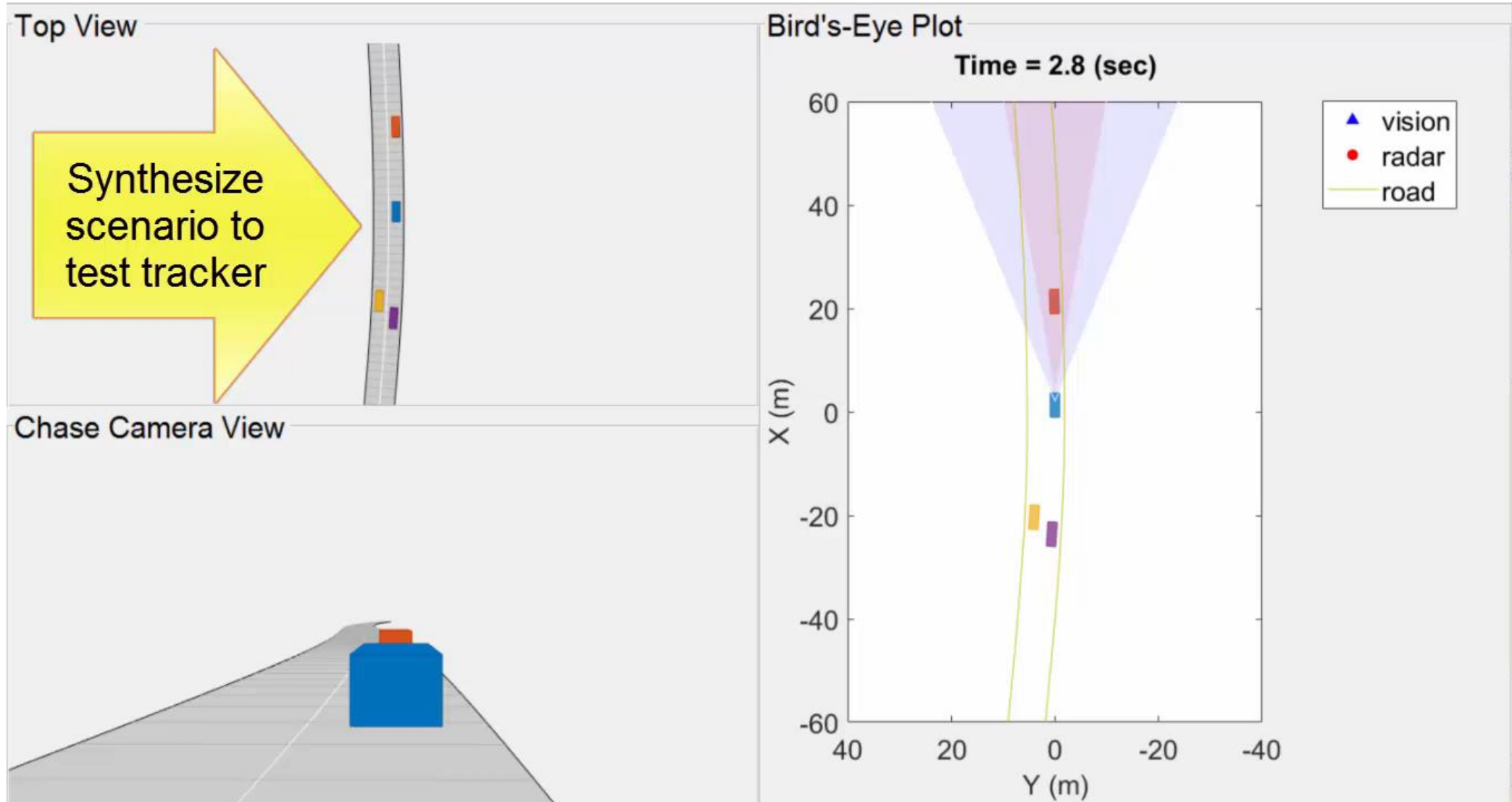
Image Coordinates



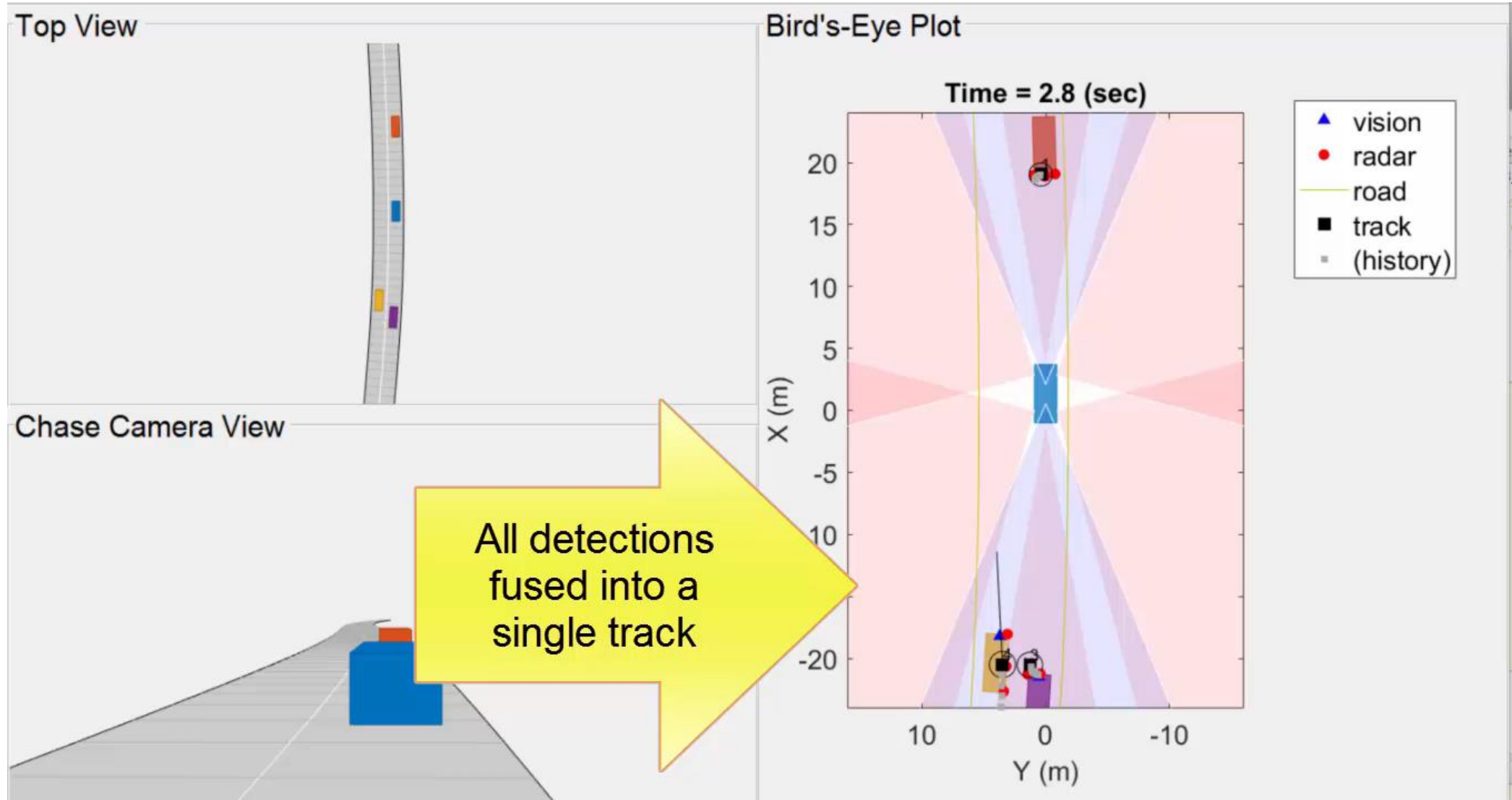
Vehicle Coordinates



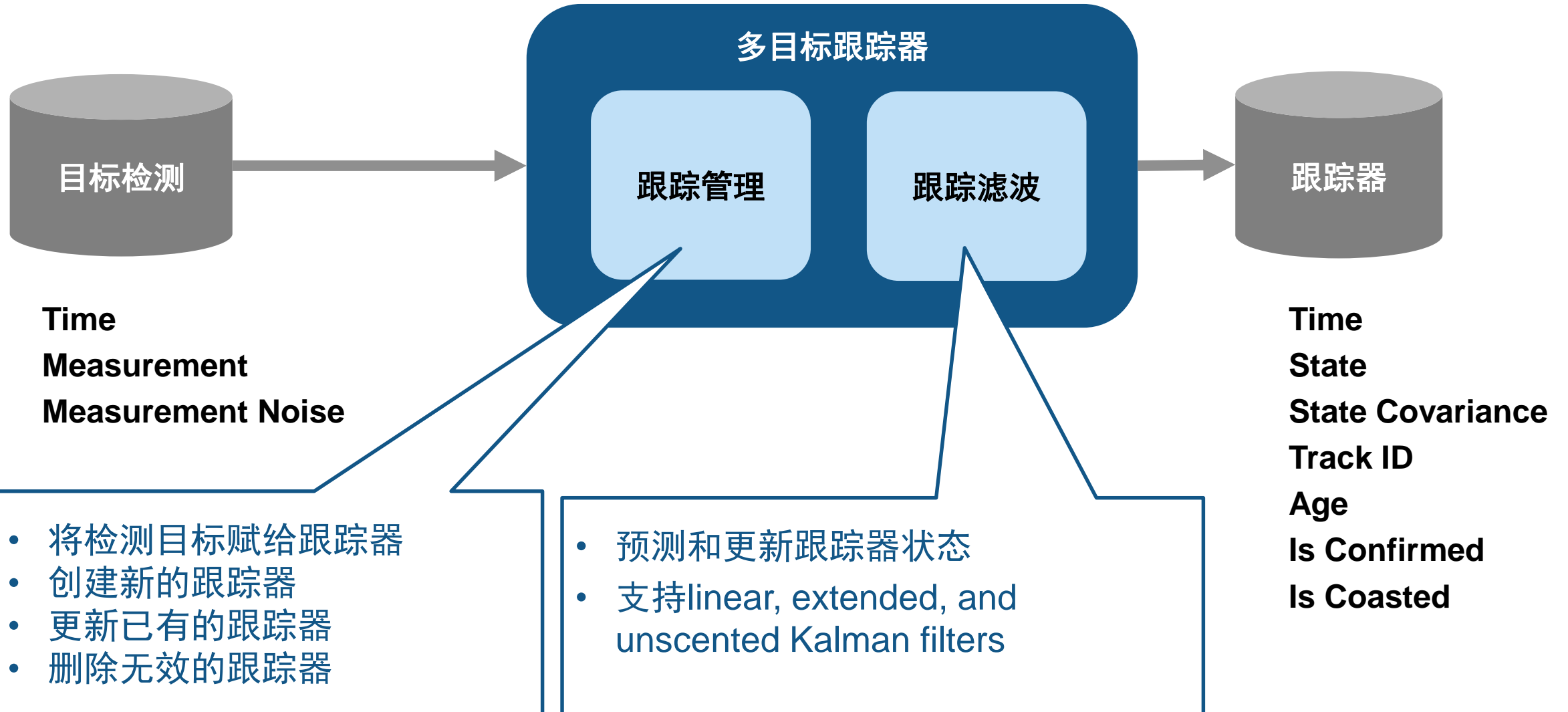
生成交通场景测试跟踪器



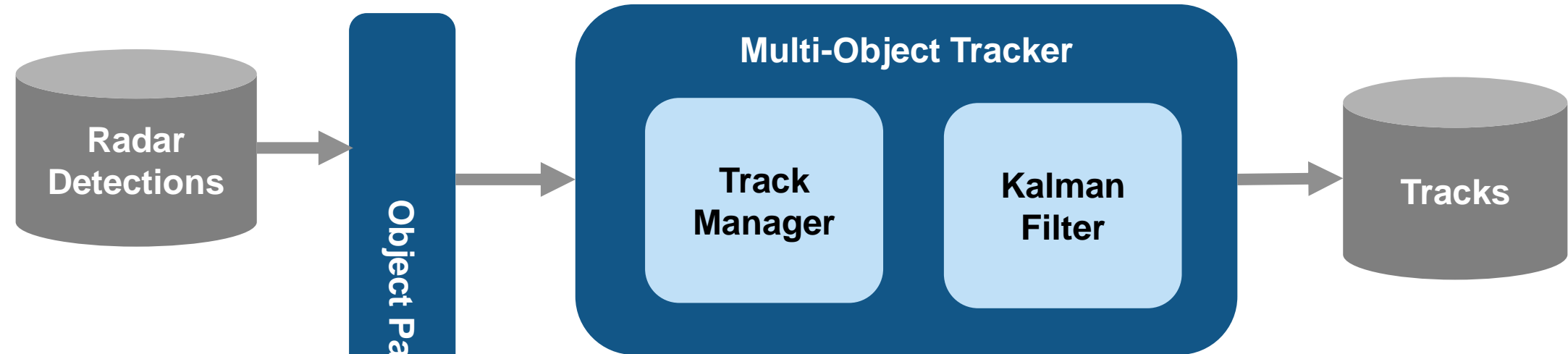
用合成的数据测试跟踪器



跟踪多目标检测



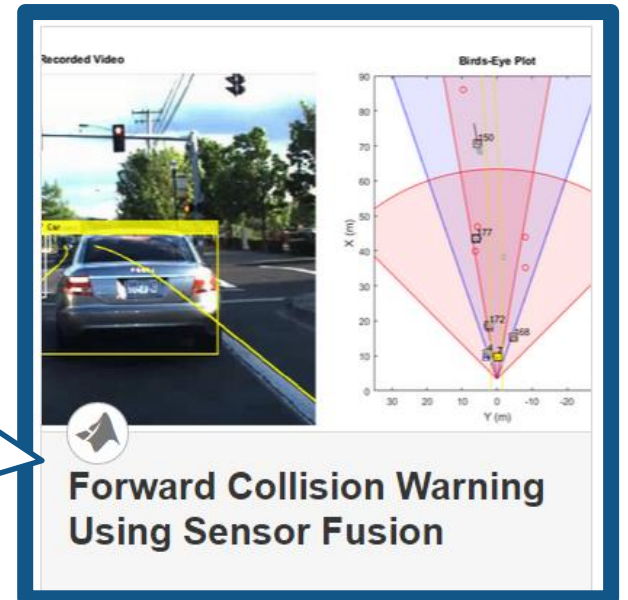
通过例子了解更多传感器融合



Forward Collision Warning Using Sensor Fusion

product demo illustrates

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



将算法自动生成C代码

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)
```

指定驾驶场景和道路

```

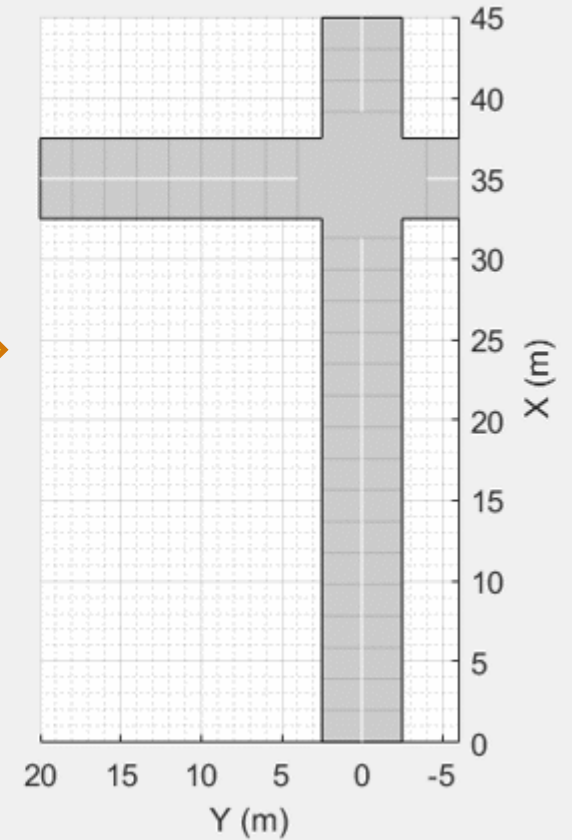
%% Create a new scenario
s = drivingScenario('SampleTime', 0.05);

%% 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];

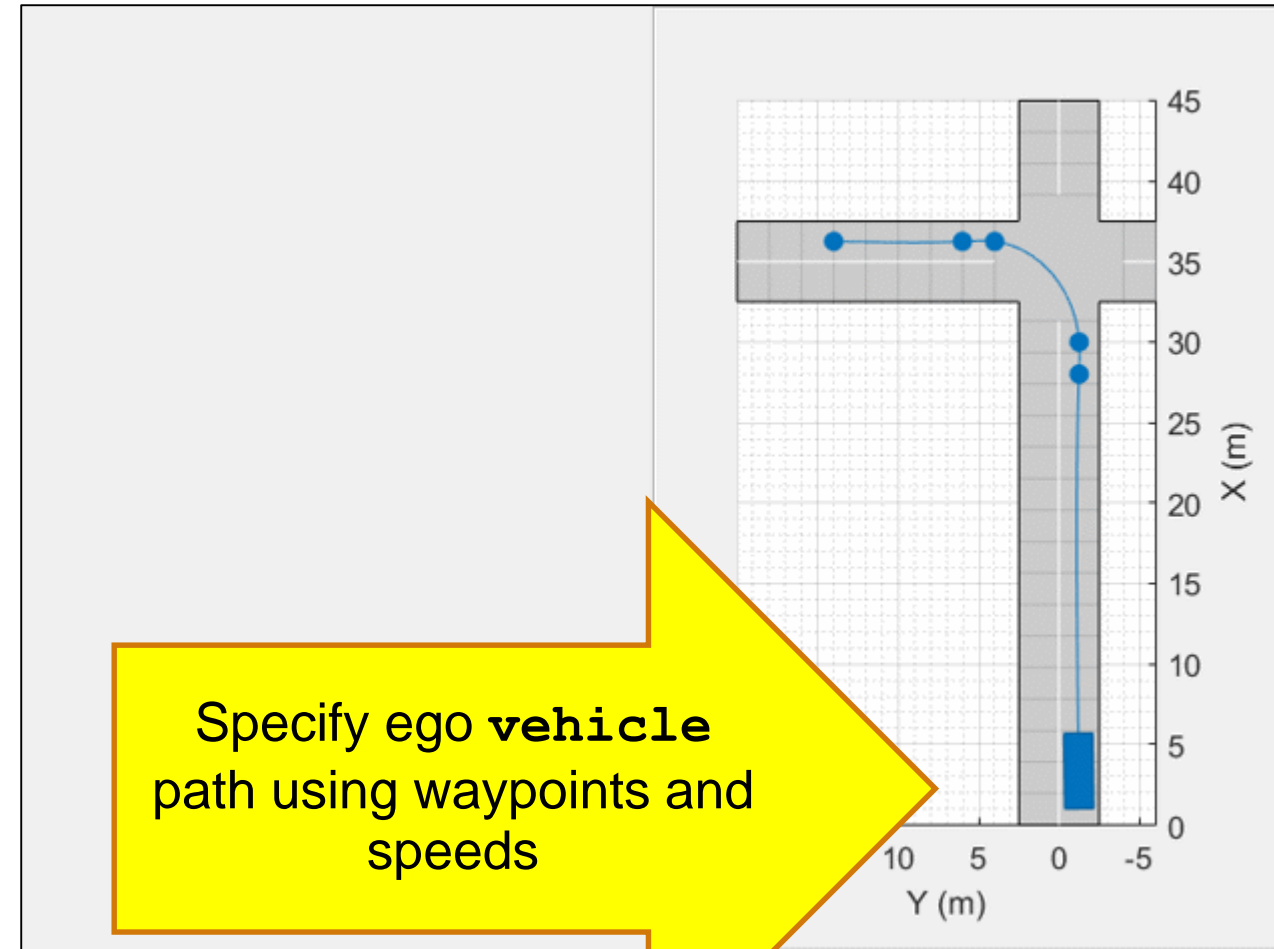
```

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



增加车辆（本车）

```
%% 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);
```

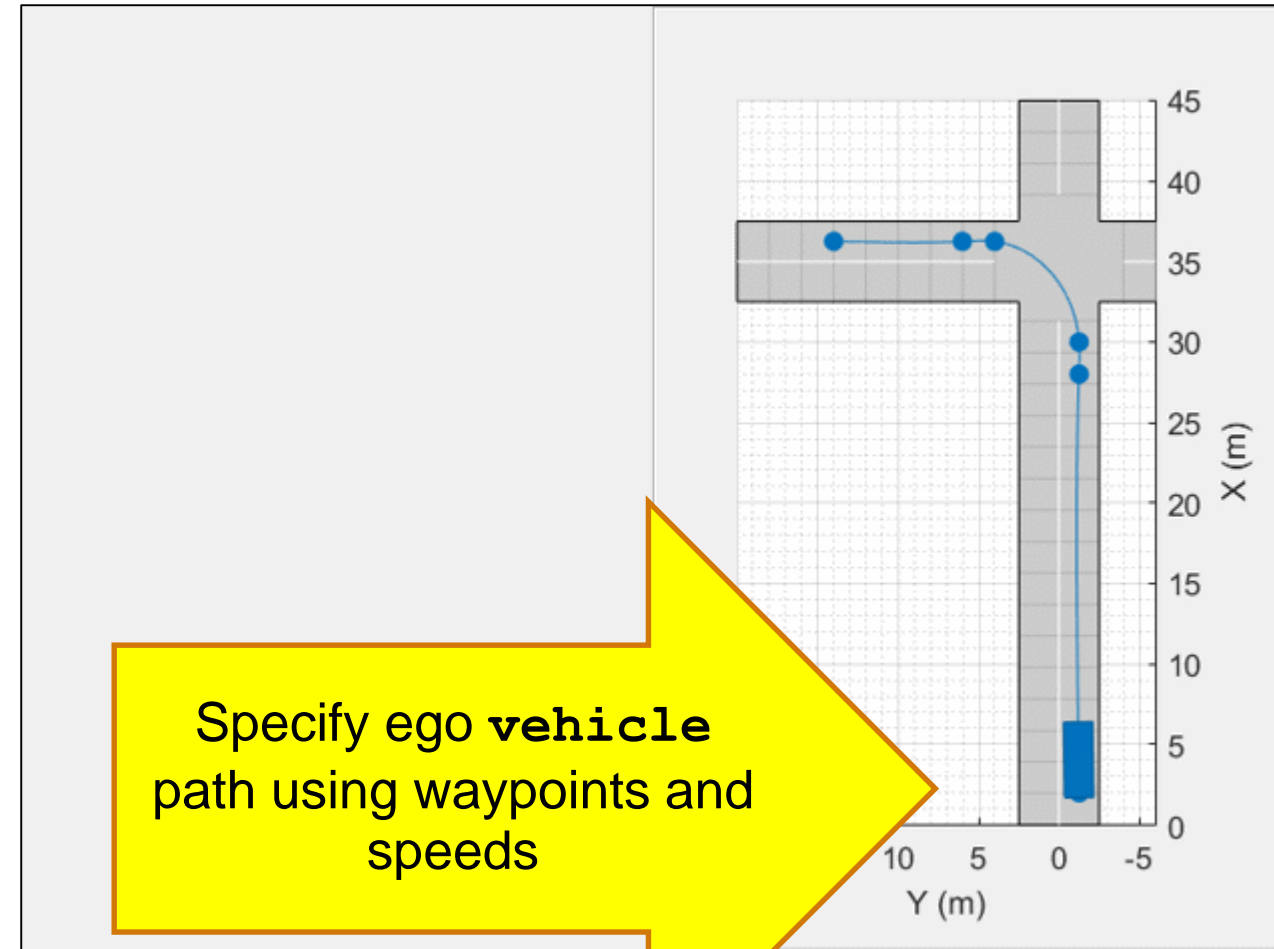


增加车辆（本车）

```
%% 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
```



增加目标车辆和行人参与者

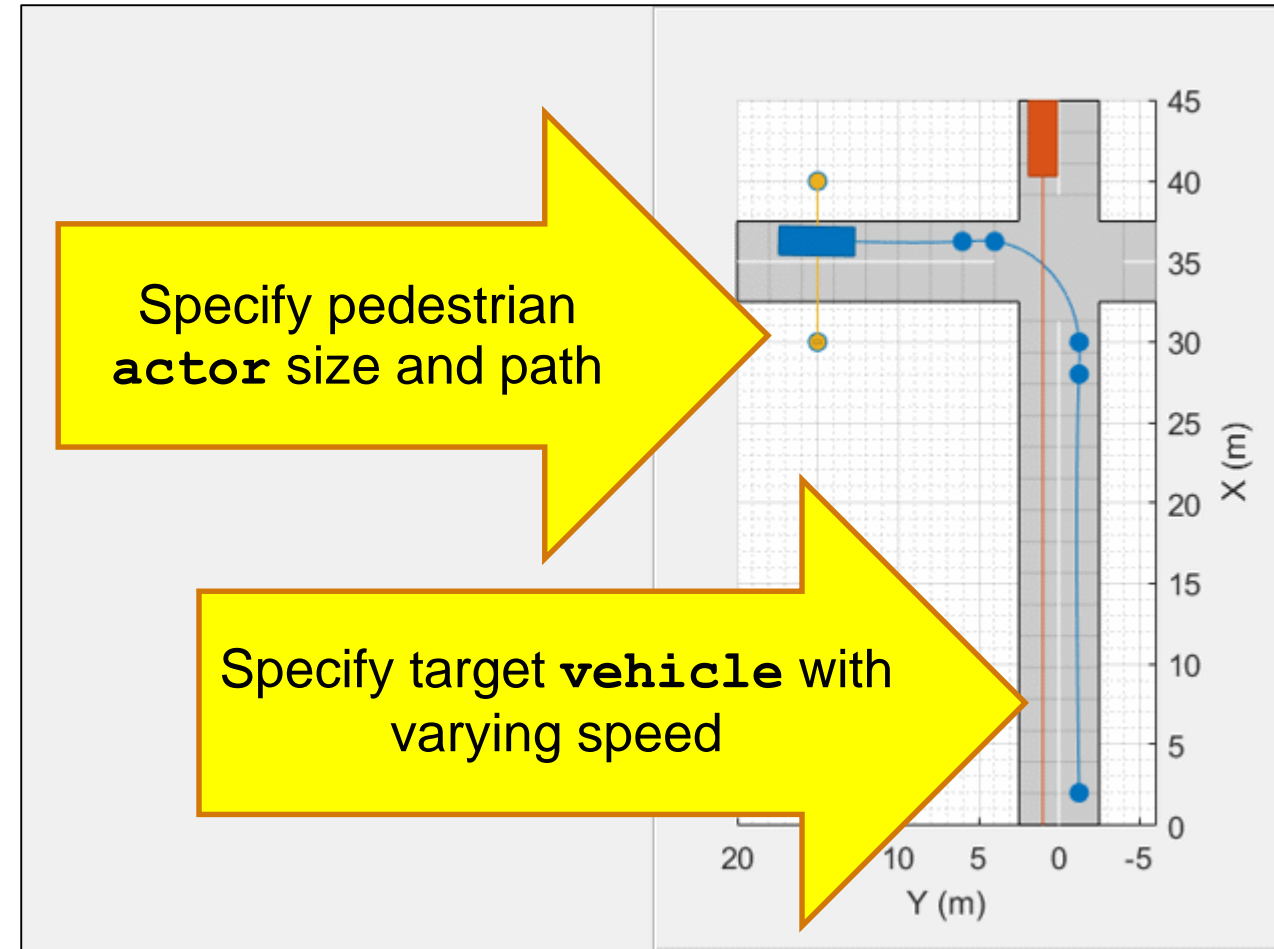
```

%% 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);

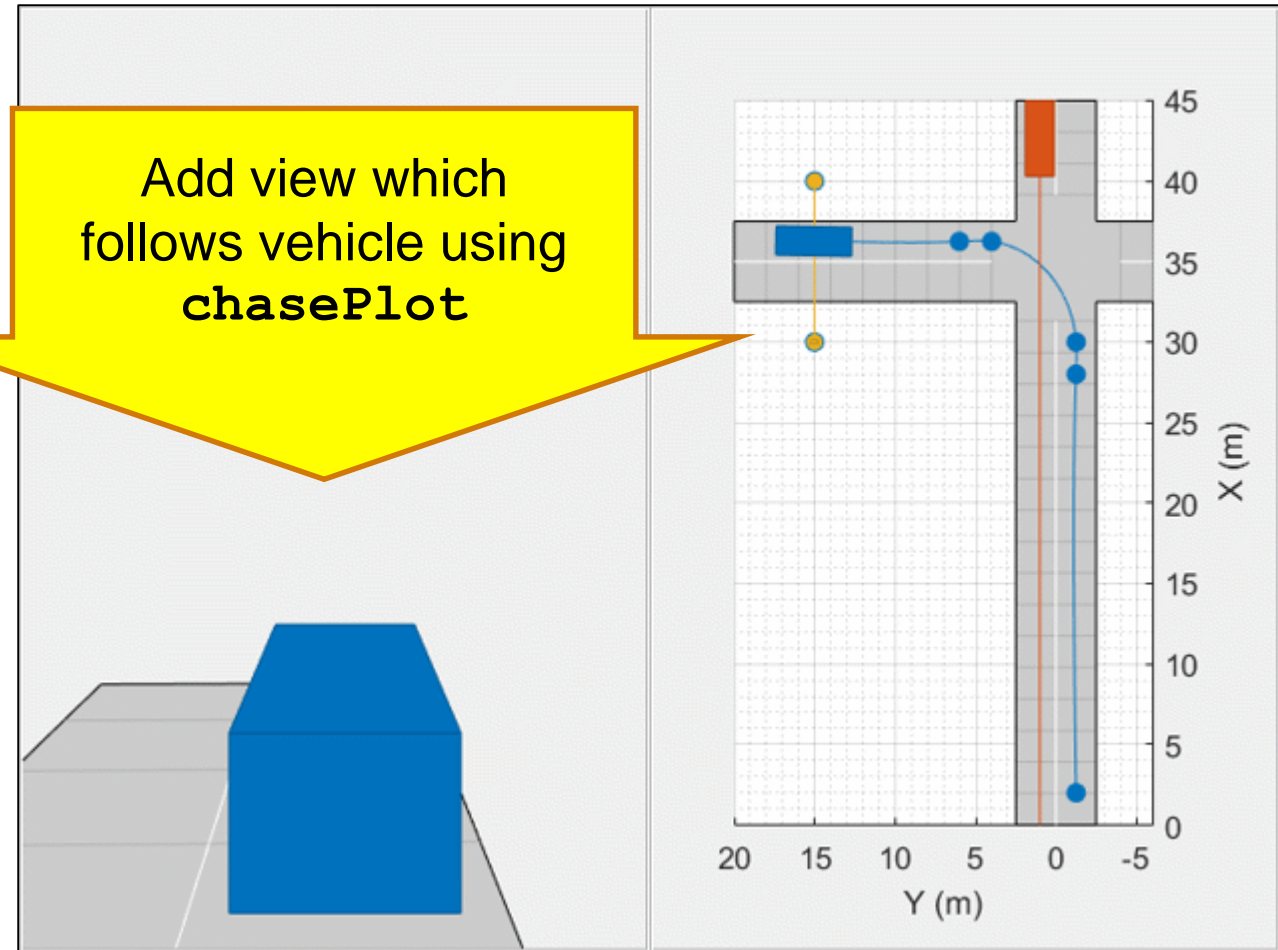
path(child,...
    [30 15; 40 15],... % Waypoints (m)
    1.39); % Speed (m/s) = 5 km/hr

```



以本车后方的视角观察场景

```
%% 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)
```



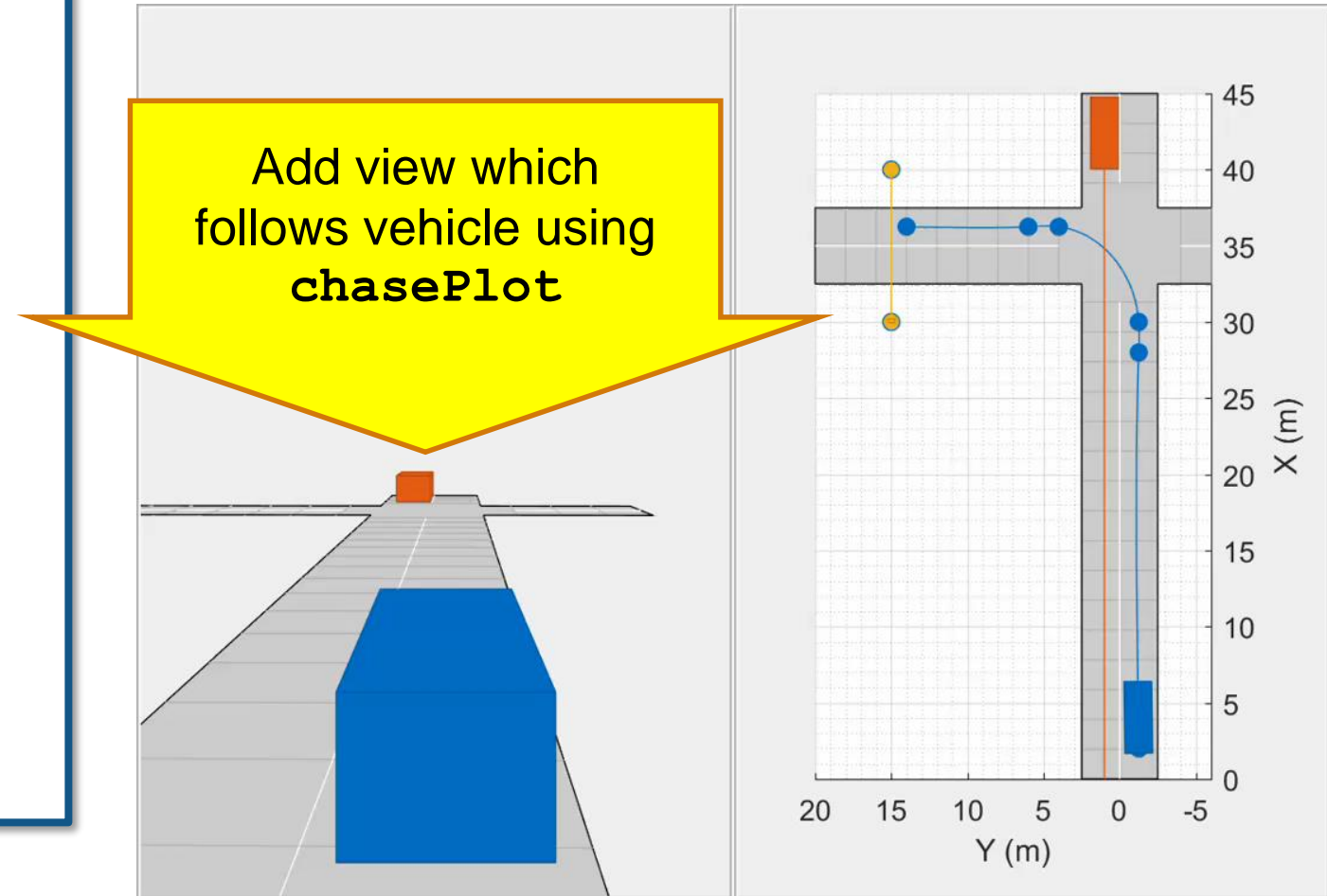
以本体车辆的后方视角观察场景 vehicle

```

%% 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)

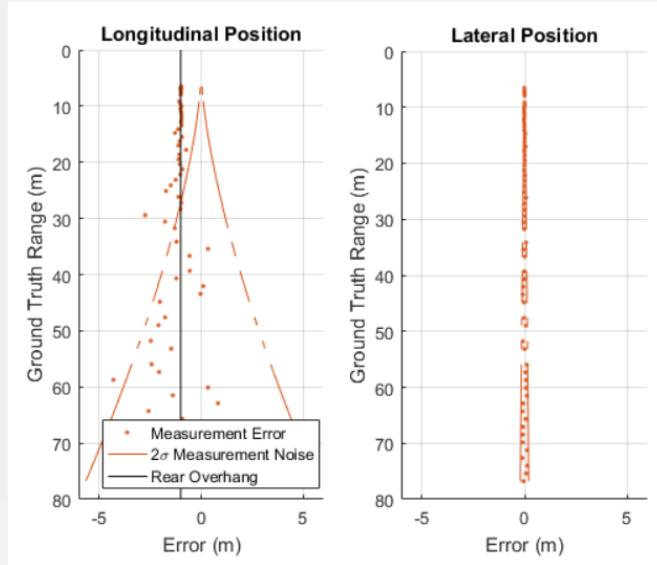
%% Play scenario
restart(s)
while advance(s)
    pause(s.SampleTime);
end

```



仿真视觉传感器目标检测的效应

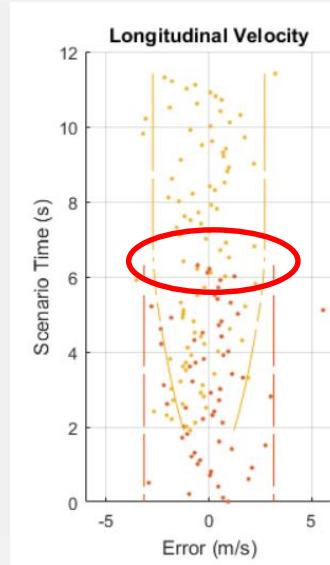
距离效应



距离测量精度
随着目标距离
增加而降低

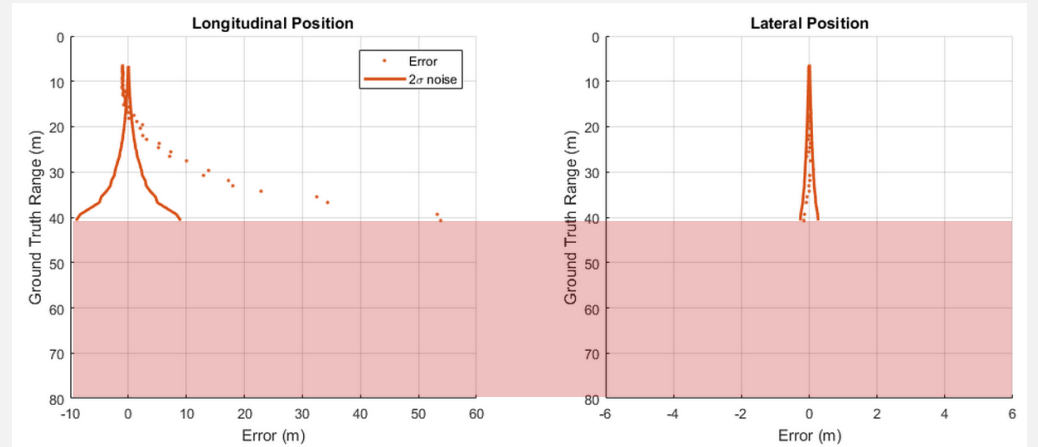
角度测量精度
在覆盖范围内
保持一致

阻挡效应



部分或完全
被阻挡的目标
无法被检测到

路面抬升效应



在覆盖区域内的目标可能没有被检测到，
因为他们出现在地平线上方

检测到的目标可能也有比较大的距离测量误差

建模视觉传感器

```
%% 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**

带着传感器模型运行场景

```

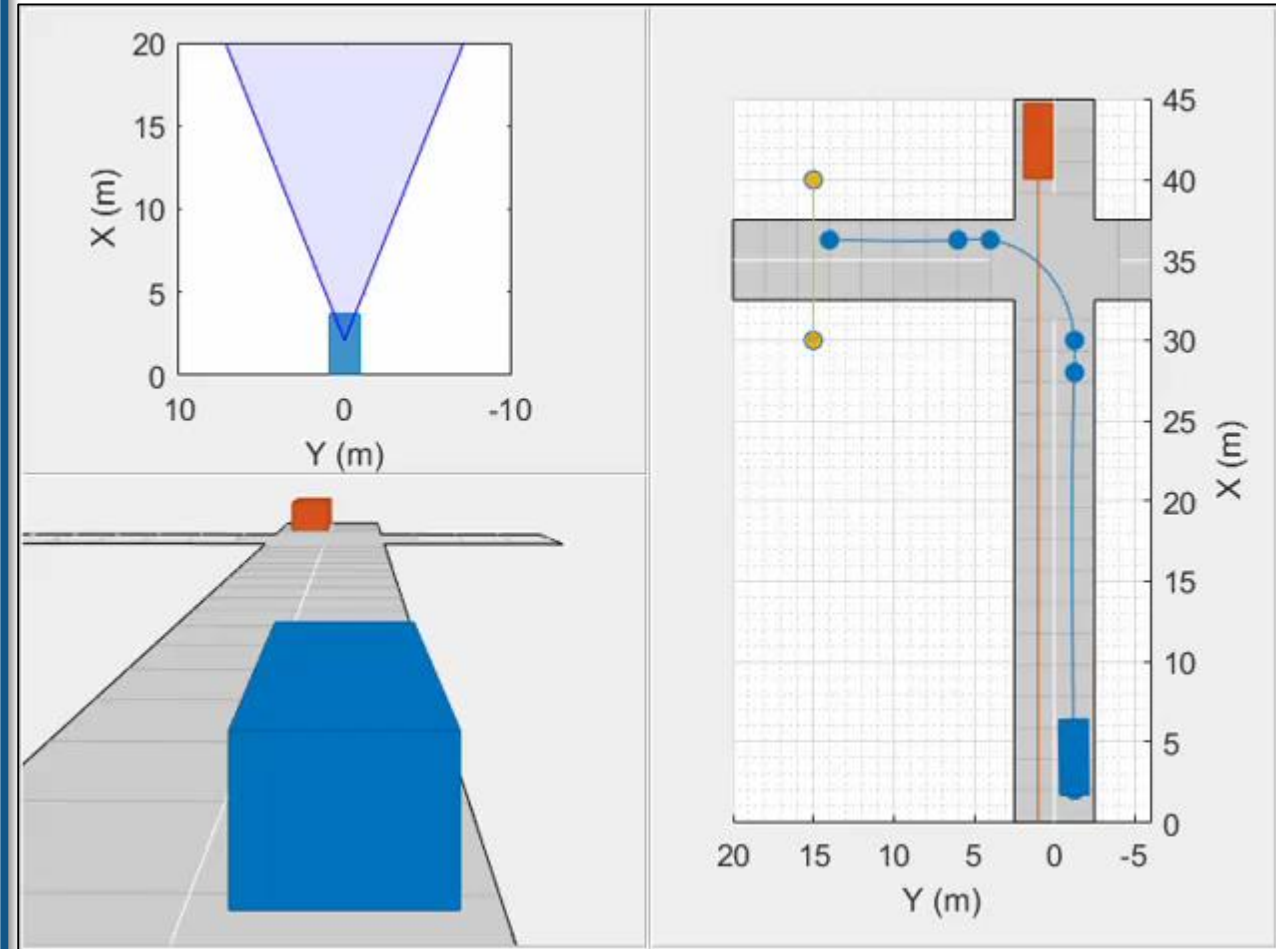
restart(s)
while advance(s)
    % Get detections in ego vehicle coordinates
    det = sensor(targetPoses(egoCar), ...
                s.SimulationTime);

    % 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

```

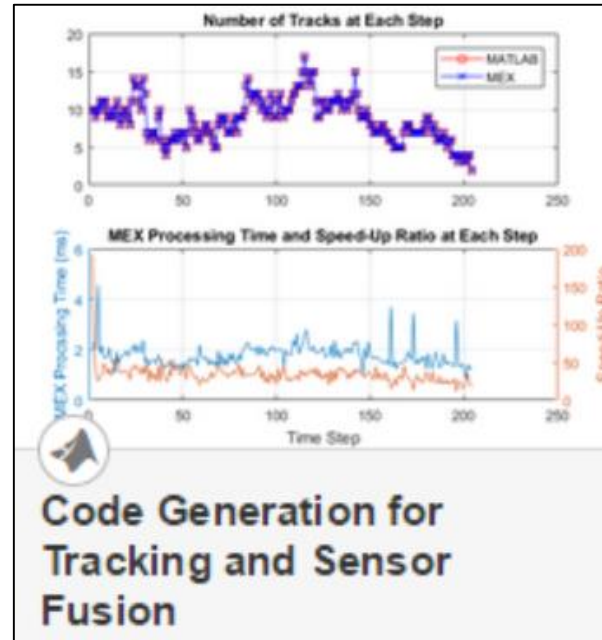


了解更多传感器融合

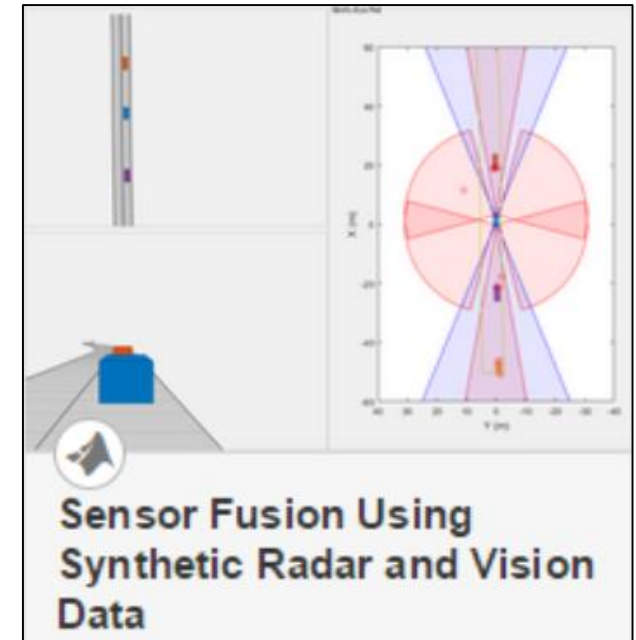
查看Automated Driving System Toolbox中的例子



- **设计**
基于记录的车辆数据
设计目标跟踪器

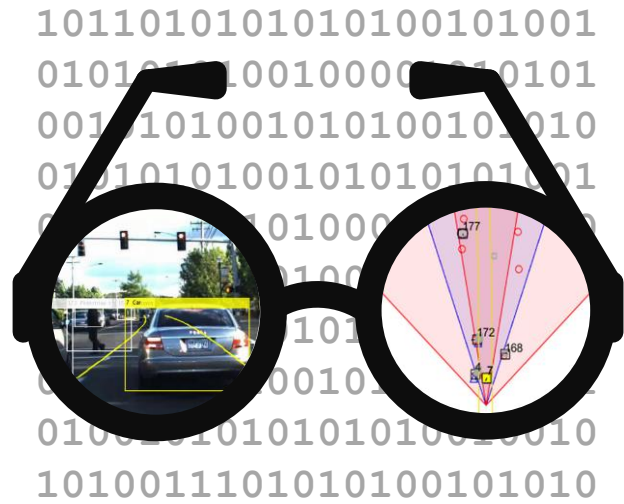


- **生成 C/C++代码**
将多目标跟踪器
生成代码



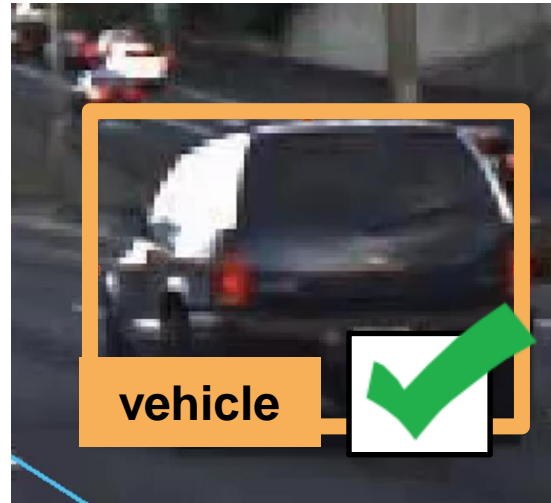
- **合成驾驶场景**
测试多目标跟踪器

自动驾驶工具箱(Automated Driving System Toolbox)能帮您...



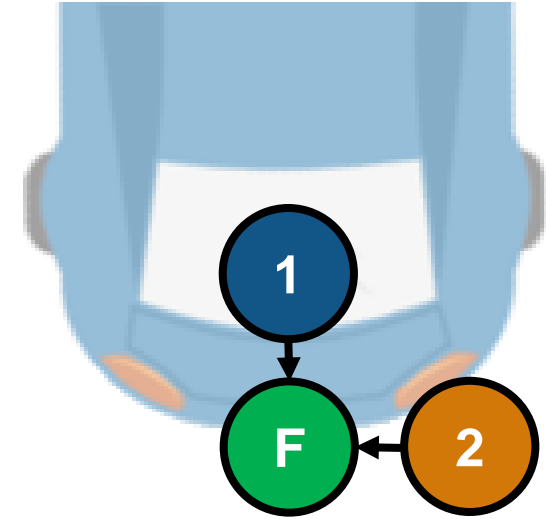
可视化车辆数据

- 绘制传感器检测结果
- 绘制覆盖范围
- 图像坐标系和车辆坐标系转换



在图像中检测目标

- 训练深度学习网络
- 标记真实值
- 连接到其他工具



融合多个检测结果

- 设计多目标跟踪器
- 生成 C/C++
- 合成驾驶场景