

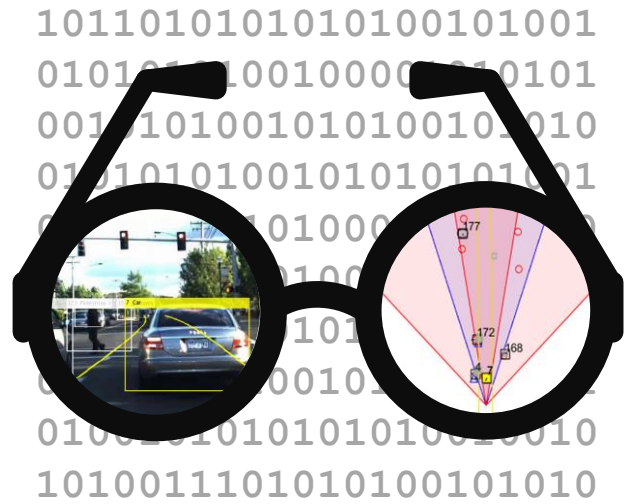


MATLAB EXPO 2017

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

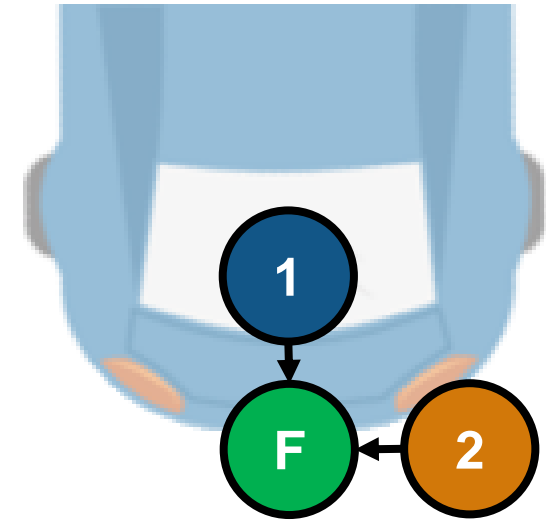
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？

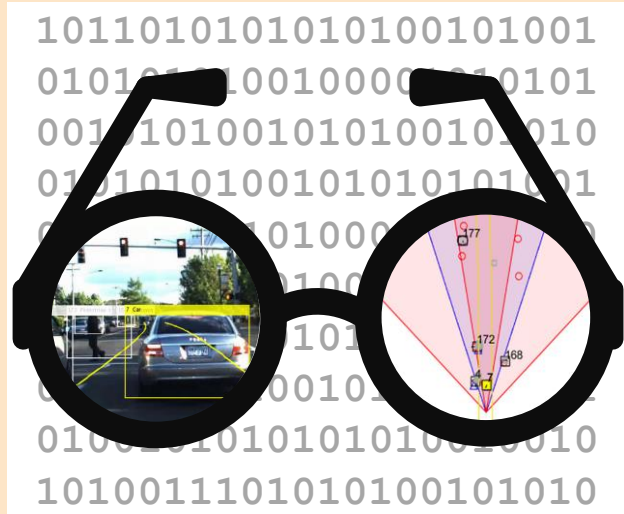


我怎样检测图
像中的目标？

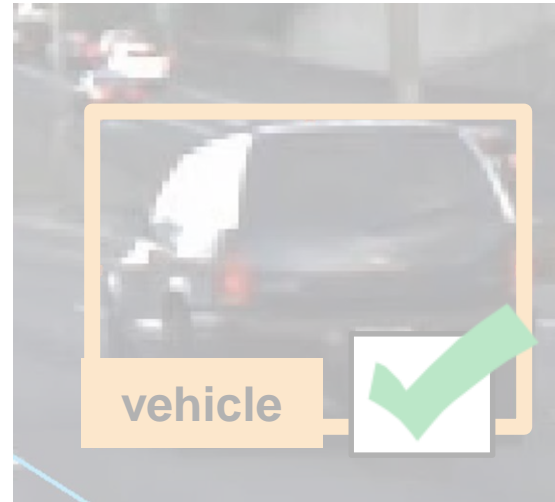


我怎样融合
多个检测结果？

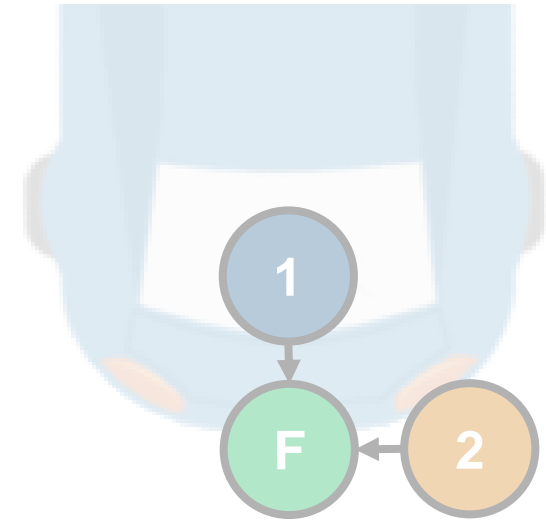
自动驾驶工程师经常遇到的问题：



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

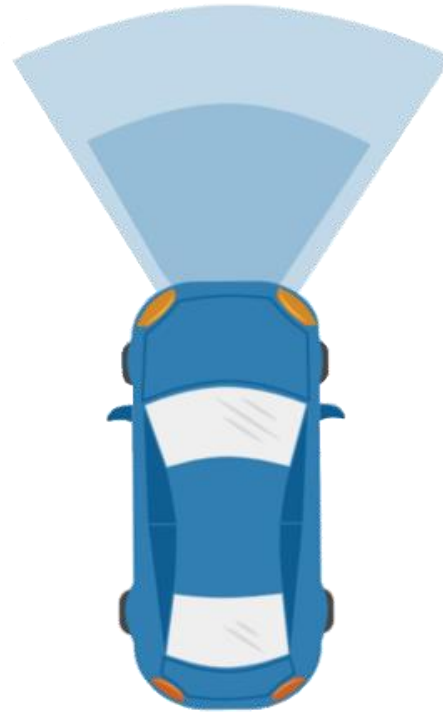
基于雷达的
目标检测

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)

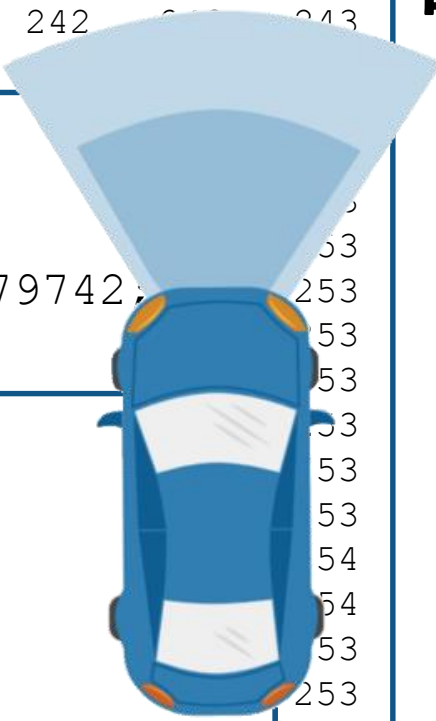
```
239 239 237 238 241 241 241 242 243
252 252 251 252 252 253 253
```

视觉检测

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

车道检测

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



雷达检测

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

激光雷达

```
(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
```

惯性测量单元

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

传感器数据可视化

Image Coordinates

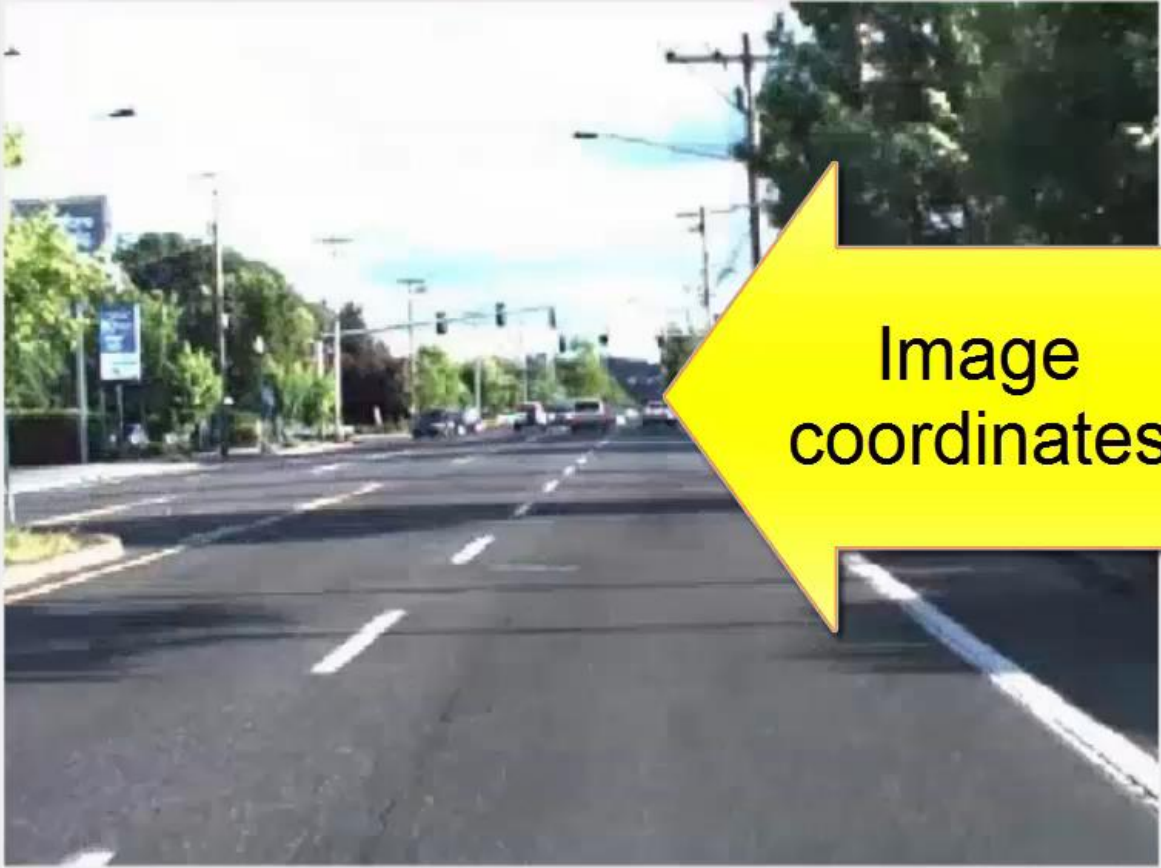
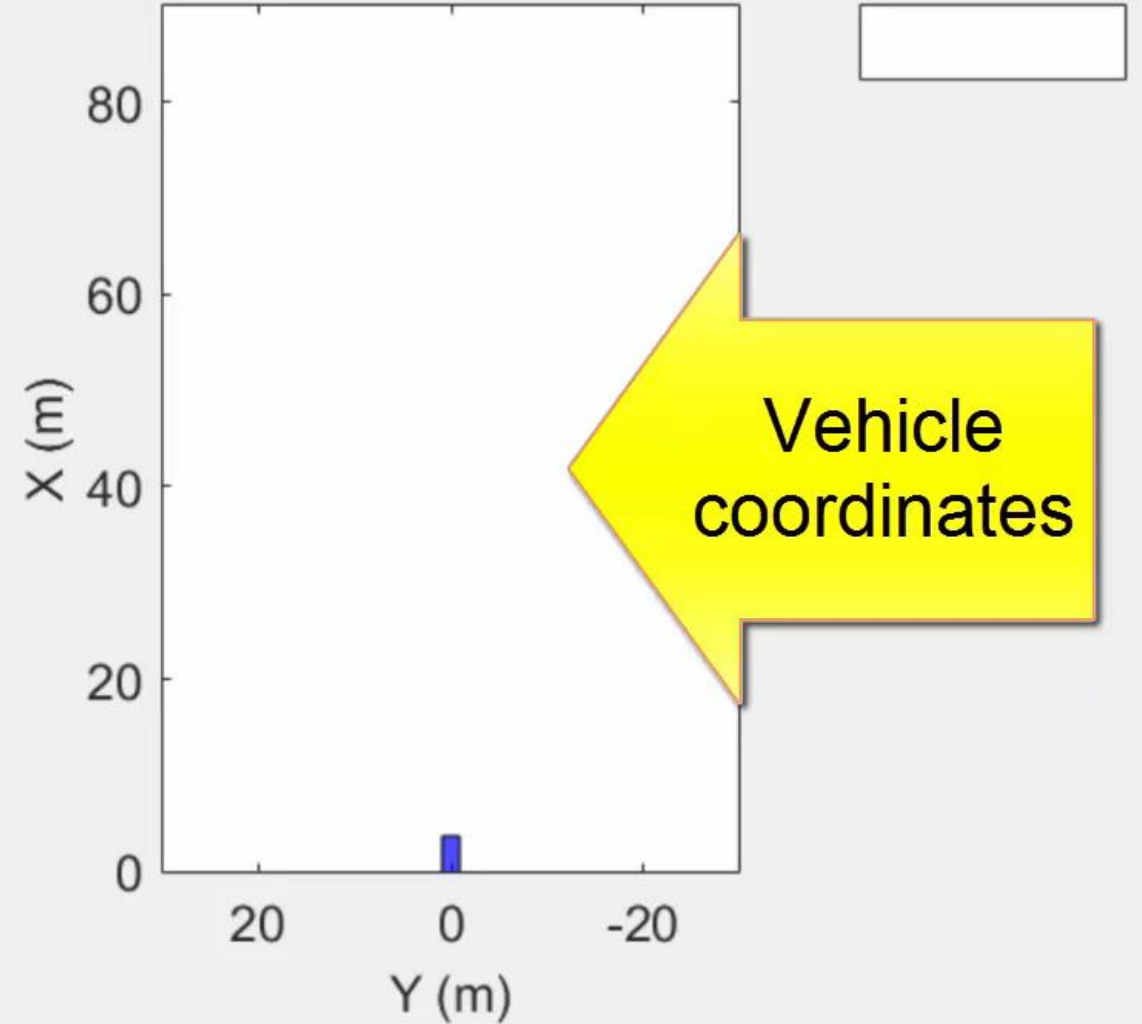


Image
coordinates

Vehicle Coordinates



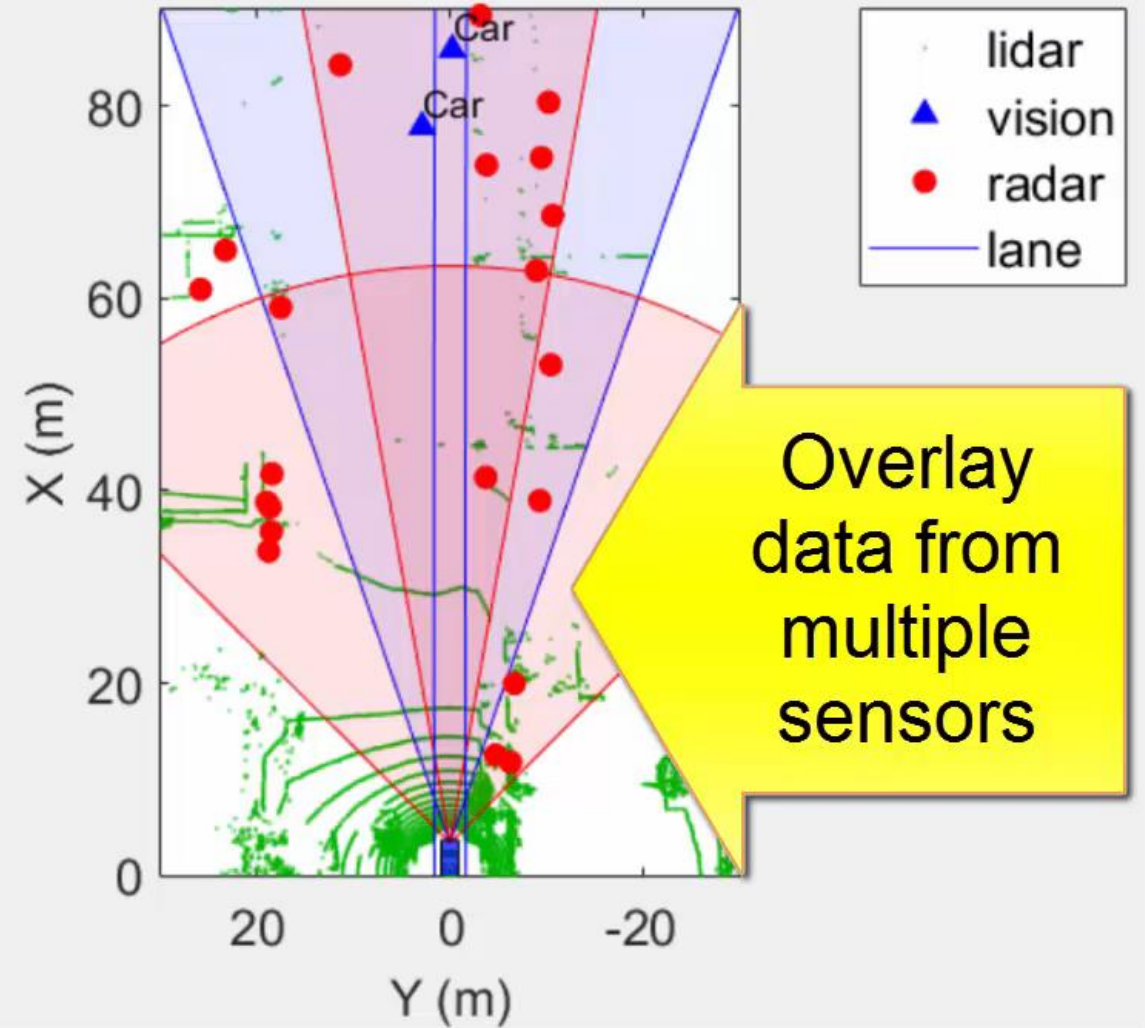
Vehicle
coordinates

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;

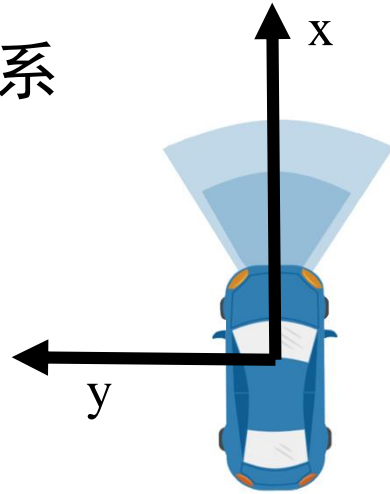
%% Plot image coordinates
ax1 = axes(...
    'Position', [0.02 0 0.55 1]);
im = imshow(frame, ...
    'Parent', ax1);
```



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



```

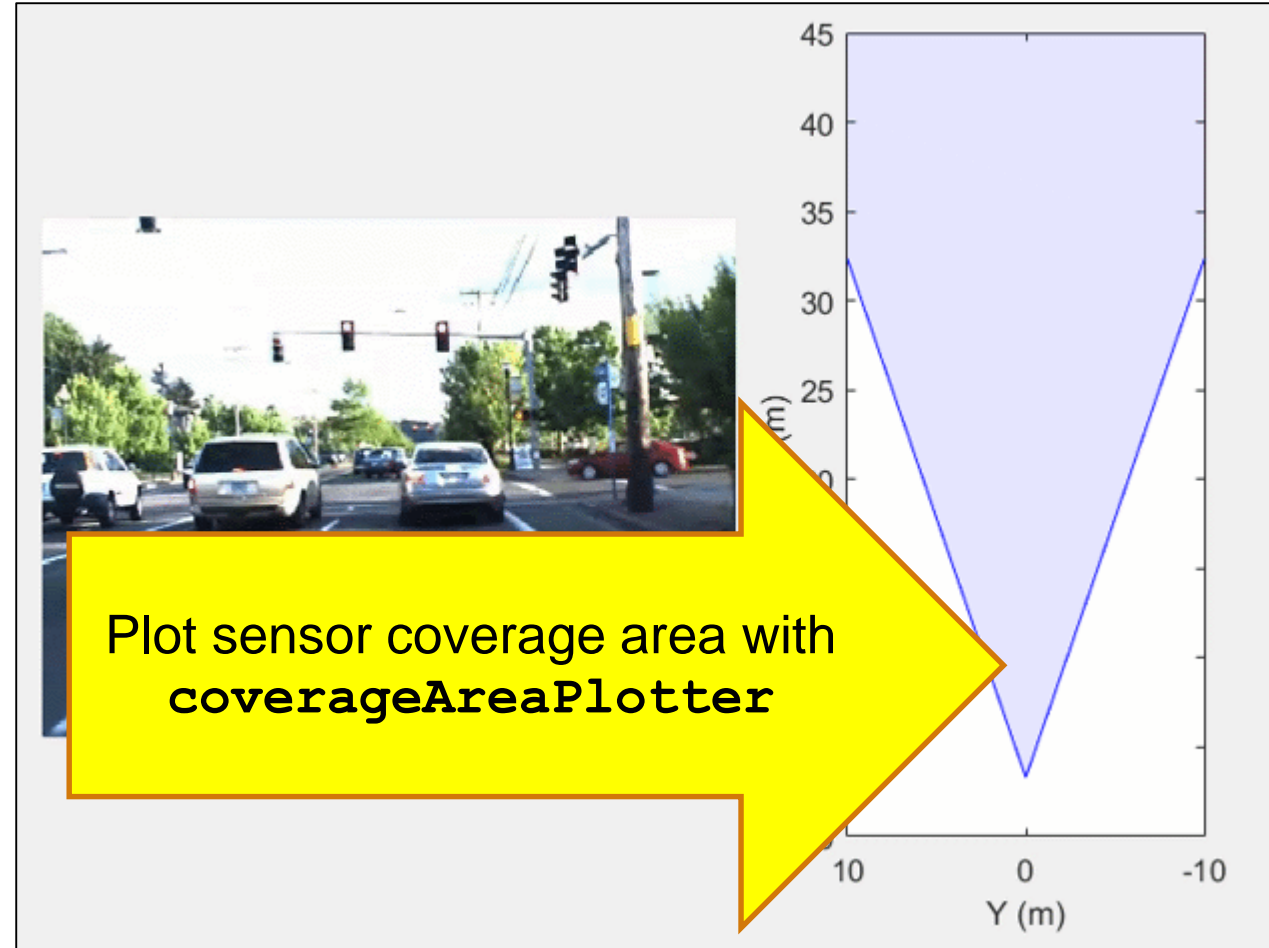
%% Plot in vehicle coordinates
ax2 = axes(...
    'Position',[0.6 0.12 0.4 0.85]);
bep = birdsEyePlot(...
    'Parent',ax2,...
    'Xlimits',[0 45],...
    'Ylimits',[-10 10]);
legend('off');
  
```



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)

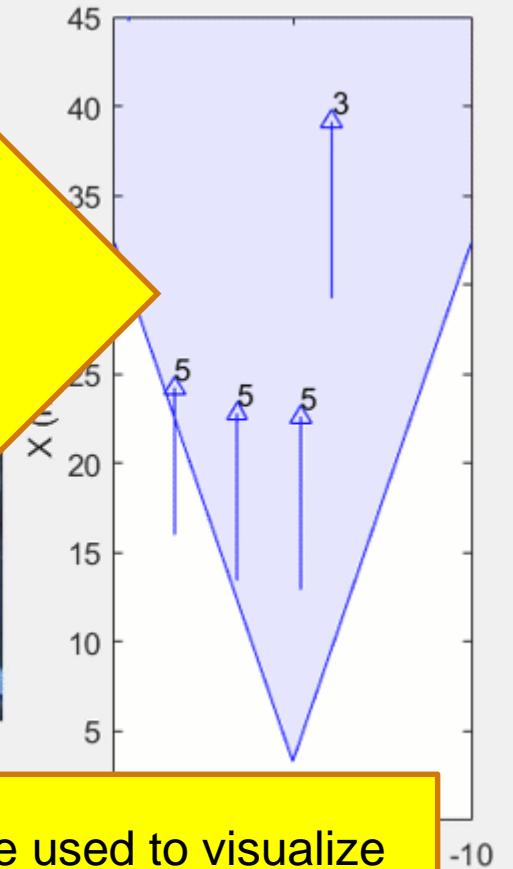
```

%% 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

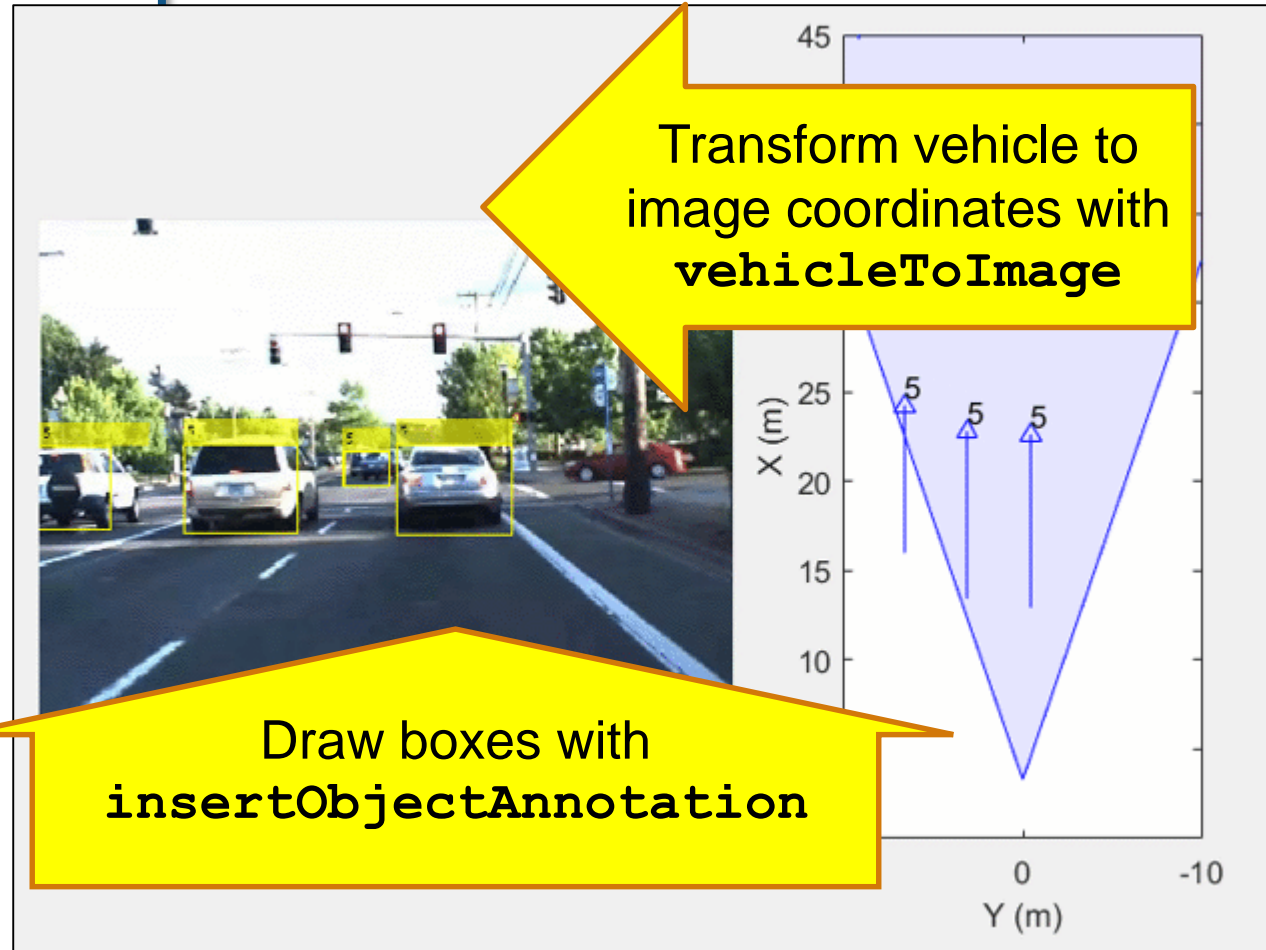
检测目标可视化 (图像坐标系)

```

%% 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)

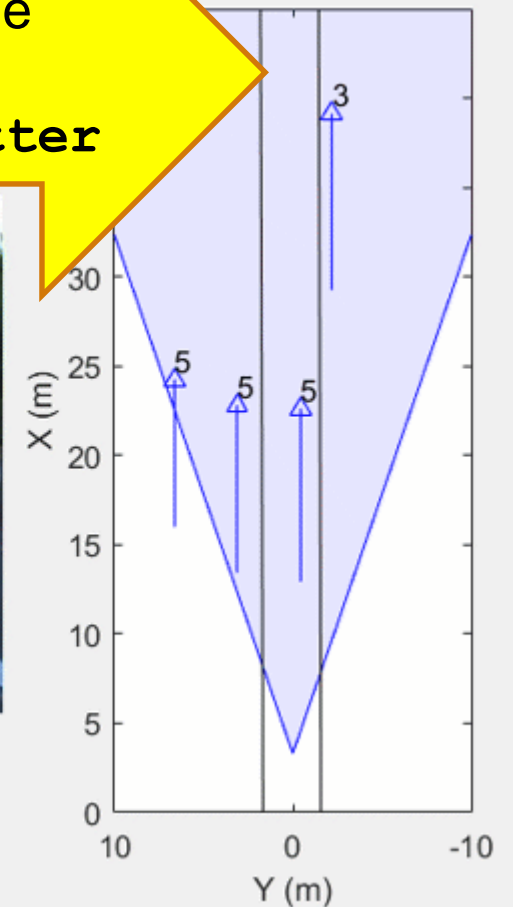
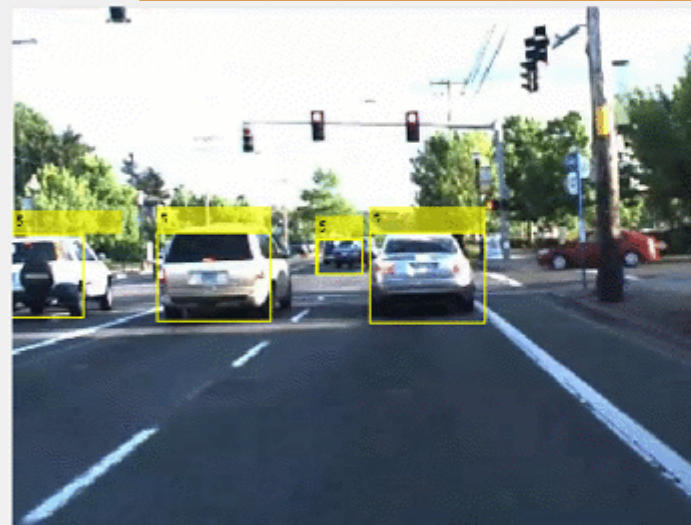
```

%% 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])

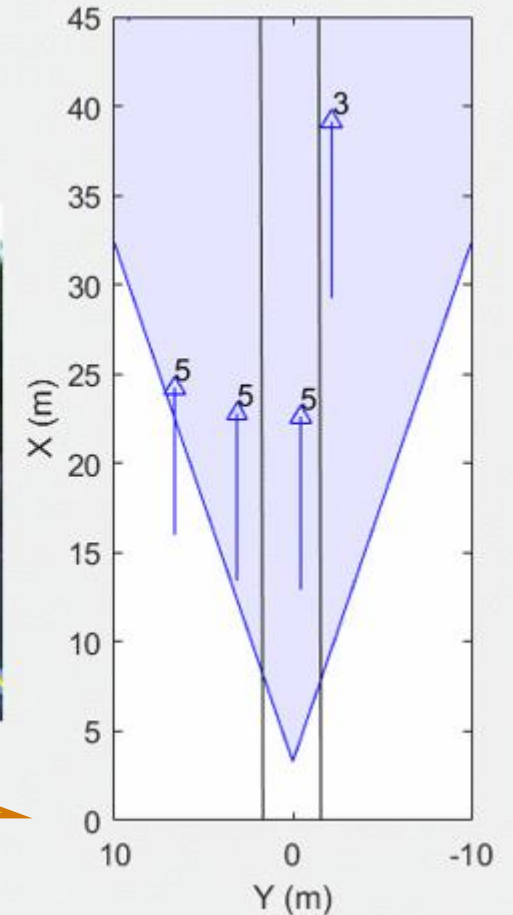
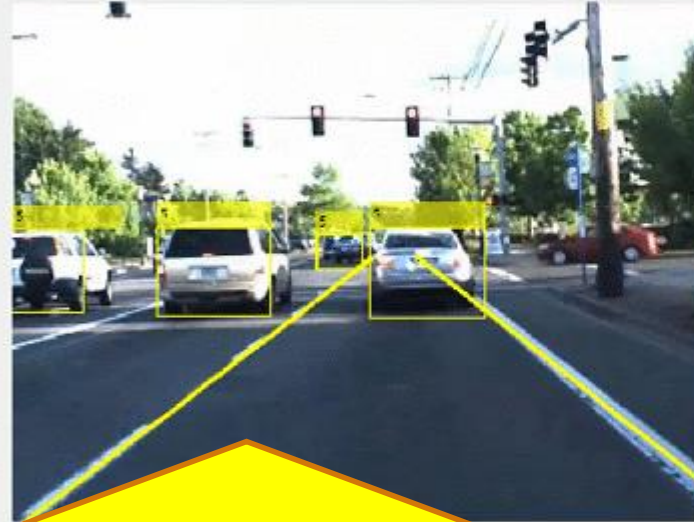
```

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`

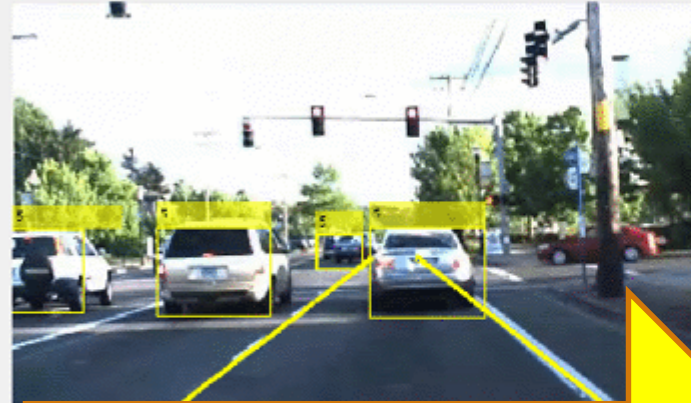
雷达检测结果可视化 (车辆坐标系)

```

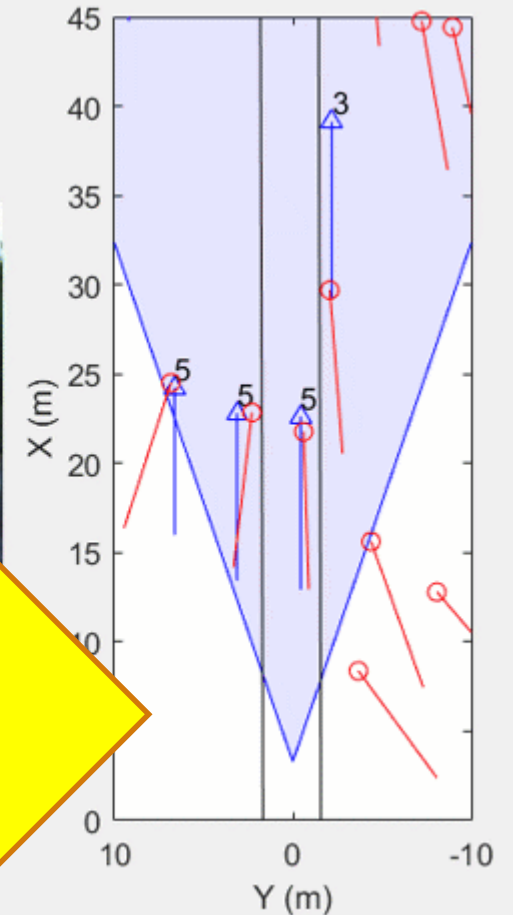
%% Create radar detection plotter
radarPlot = detectionPlotter(bep, ...
    'MarkerEdgeColor','red',...
    'Marker','o');

%% 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);

```



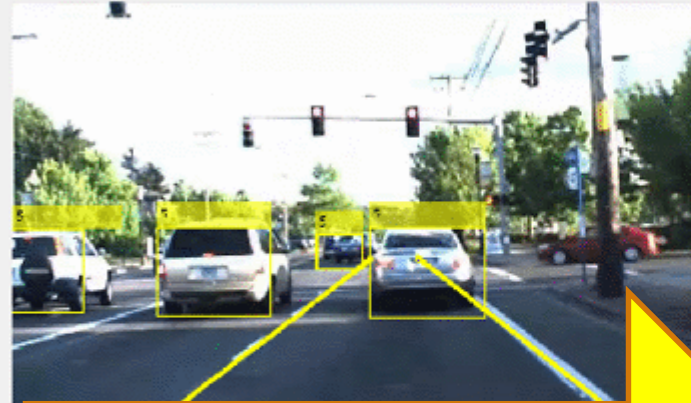
Plot radar detections just like vision detections with **detectionPlotter**



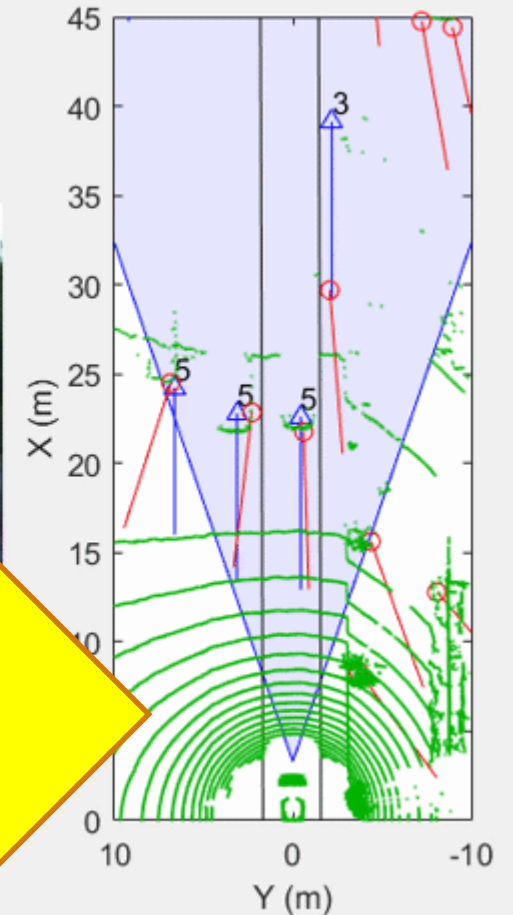
激光雷达点云可视化 (车辆坐标系)

```
% 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);
```

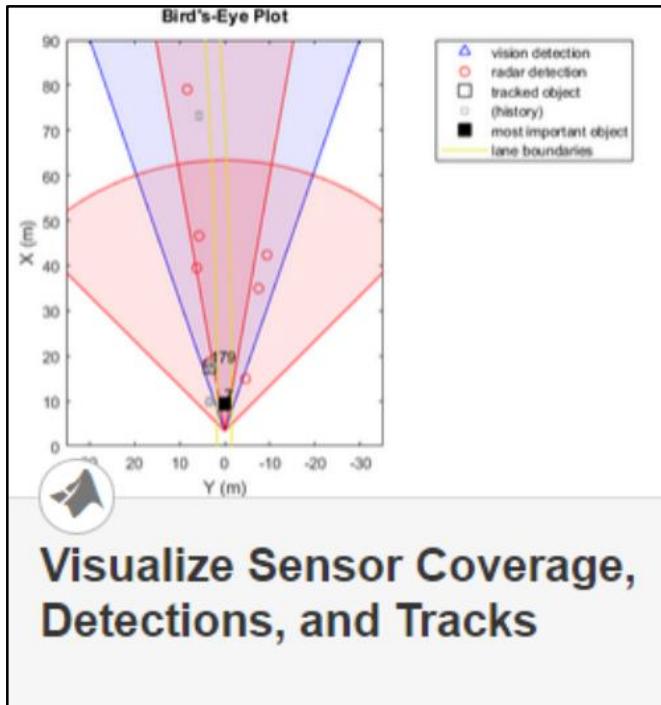


Plot lidar points just like
vision detections with
detectionPlotter

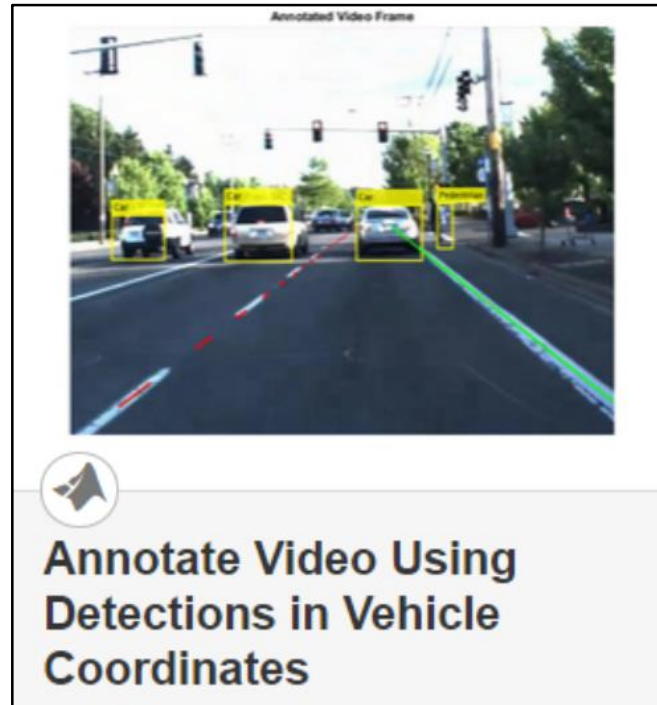


了解更多车辆数据可视化

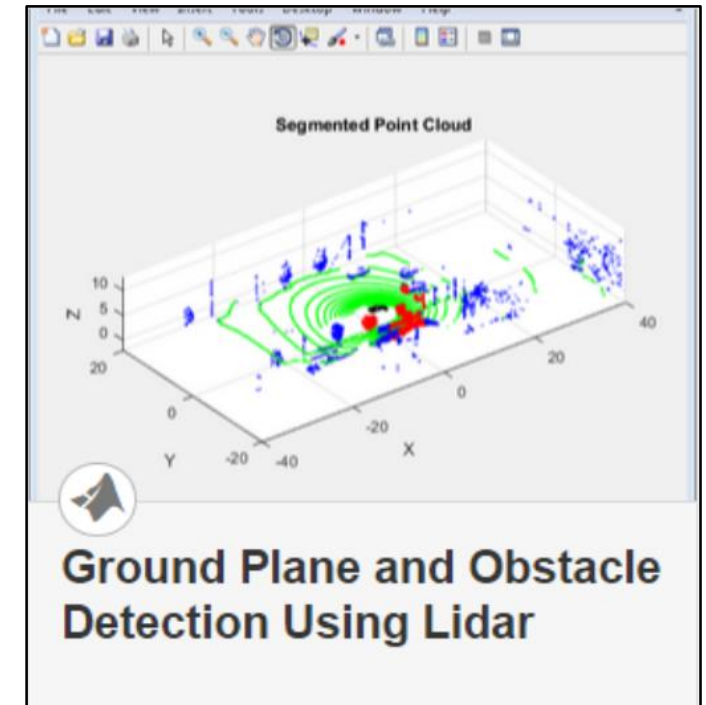
查看Automated Driving System Toolbox中的例子



- 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

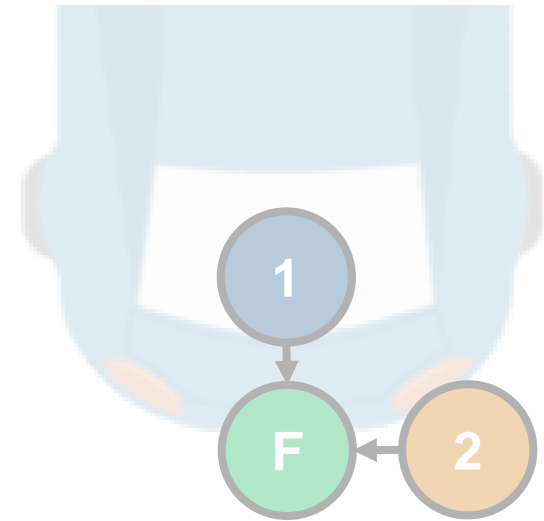
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？

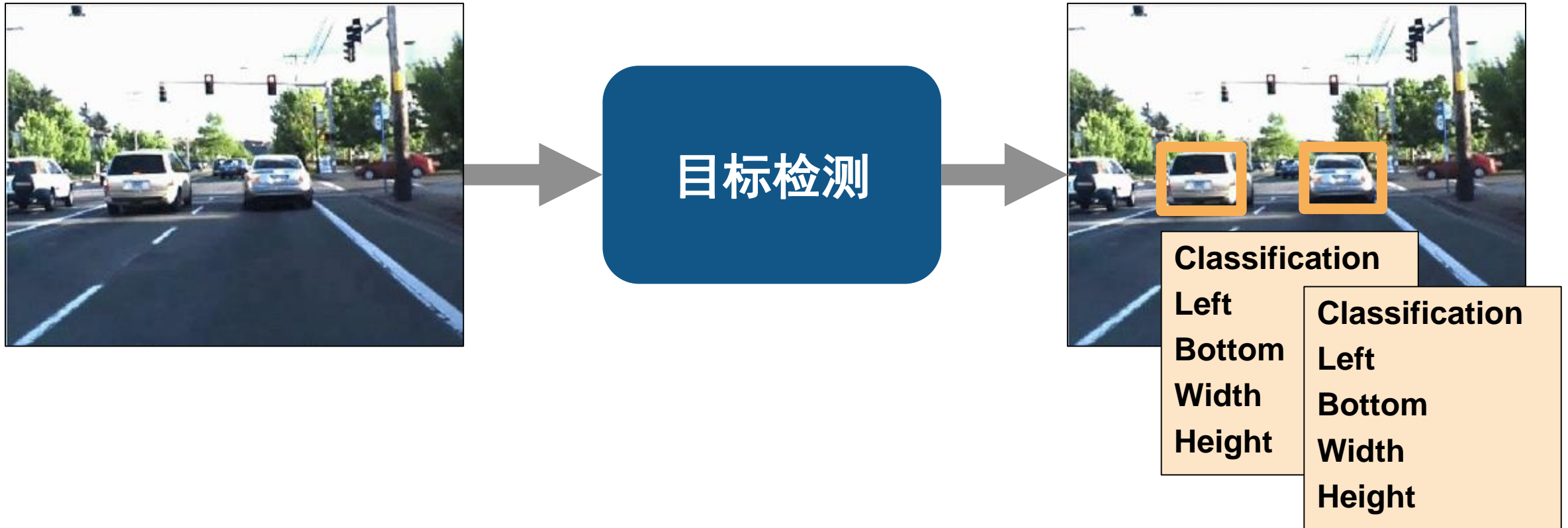


我怎样检测图
像中的目标？

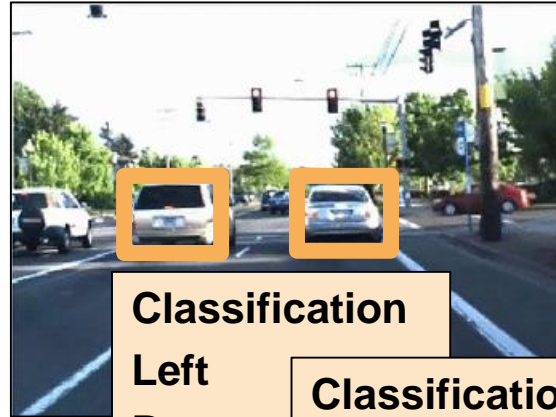


我怎样融合
多个检测结果？

我怎样检测图像中的目标？



基于真实值训练目标检测算法



Classification
Left
Bottom
Width
Height

Classification
Left
Bottom
Width
Height



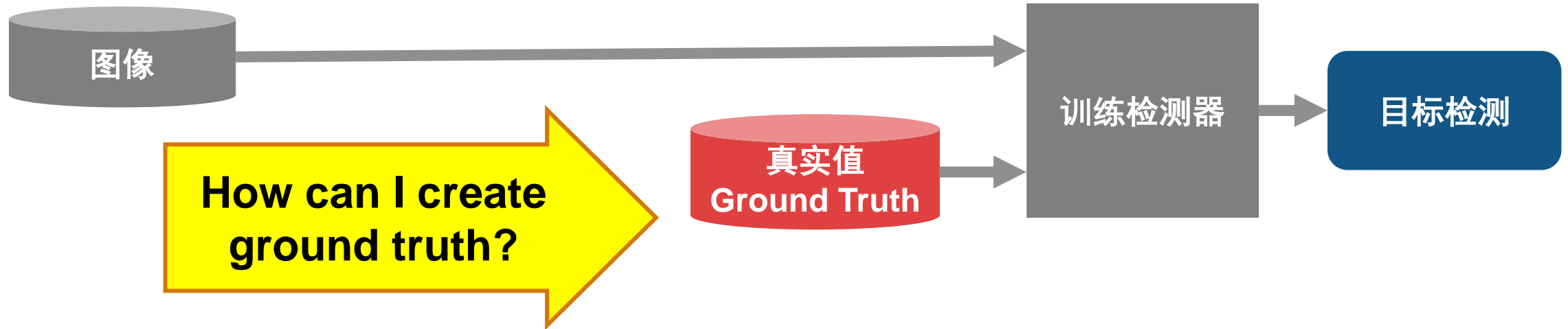
基于真实值训练目标检测算法



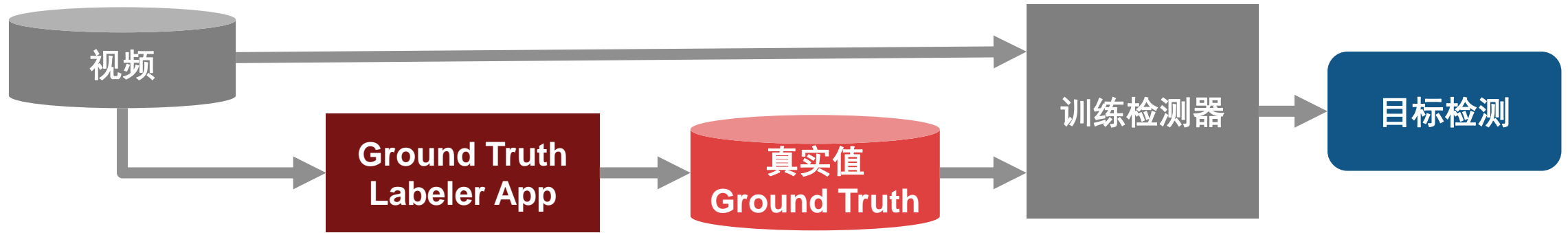
借助Computer Vision System Toolbox设计目标检测器

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>

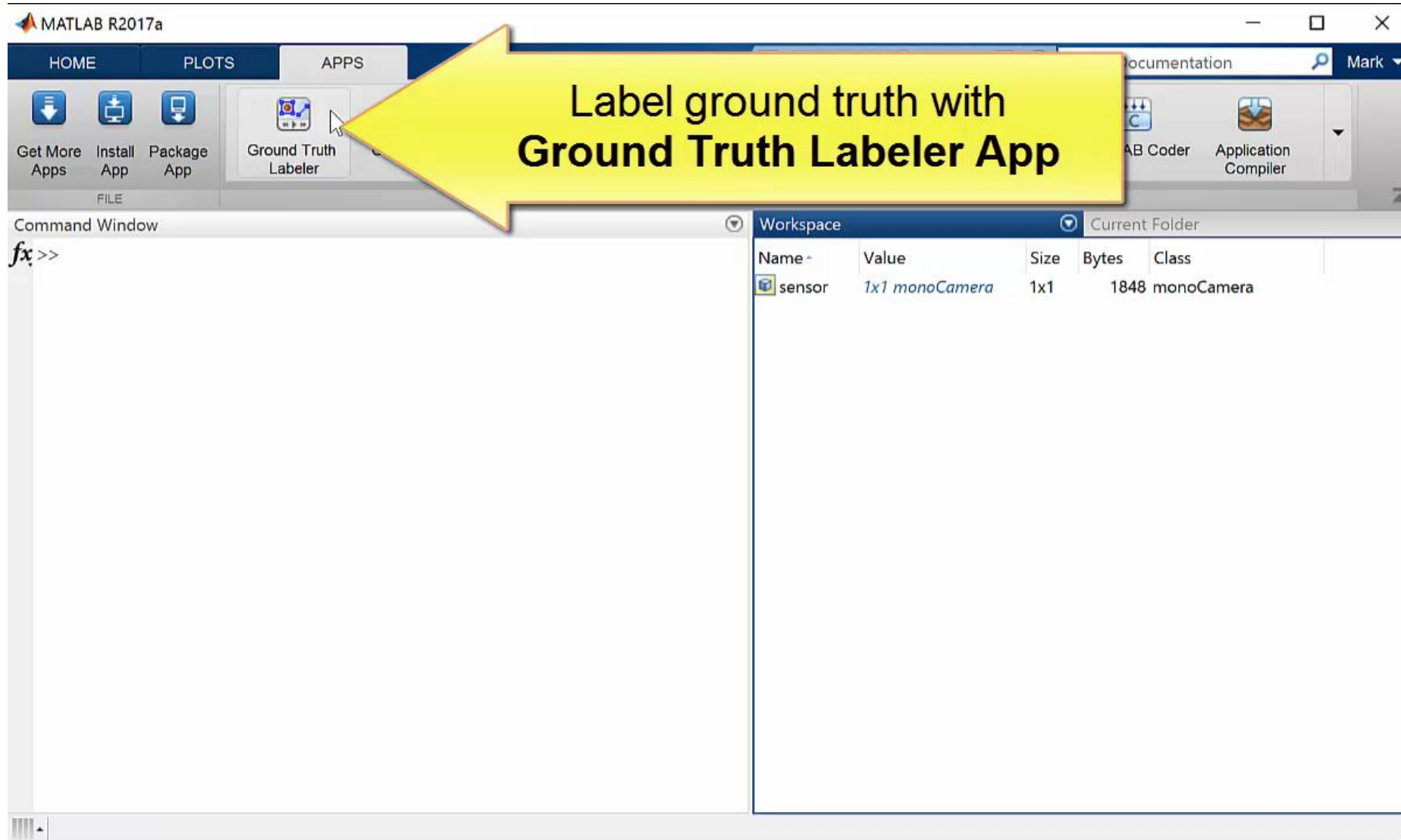
为训练检测器提供真实值



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

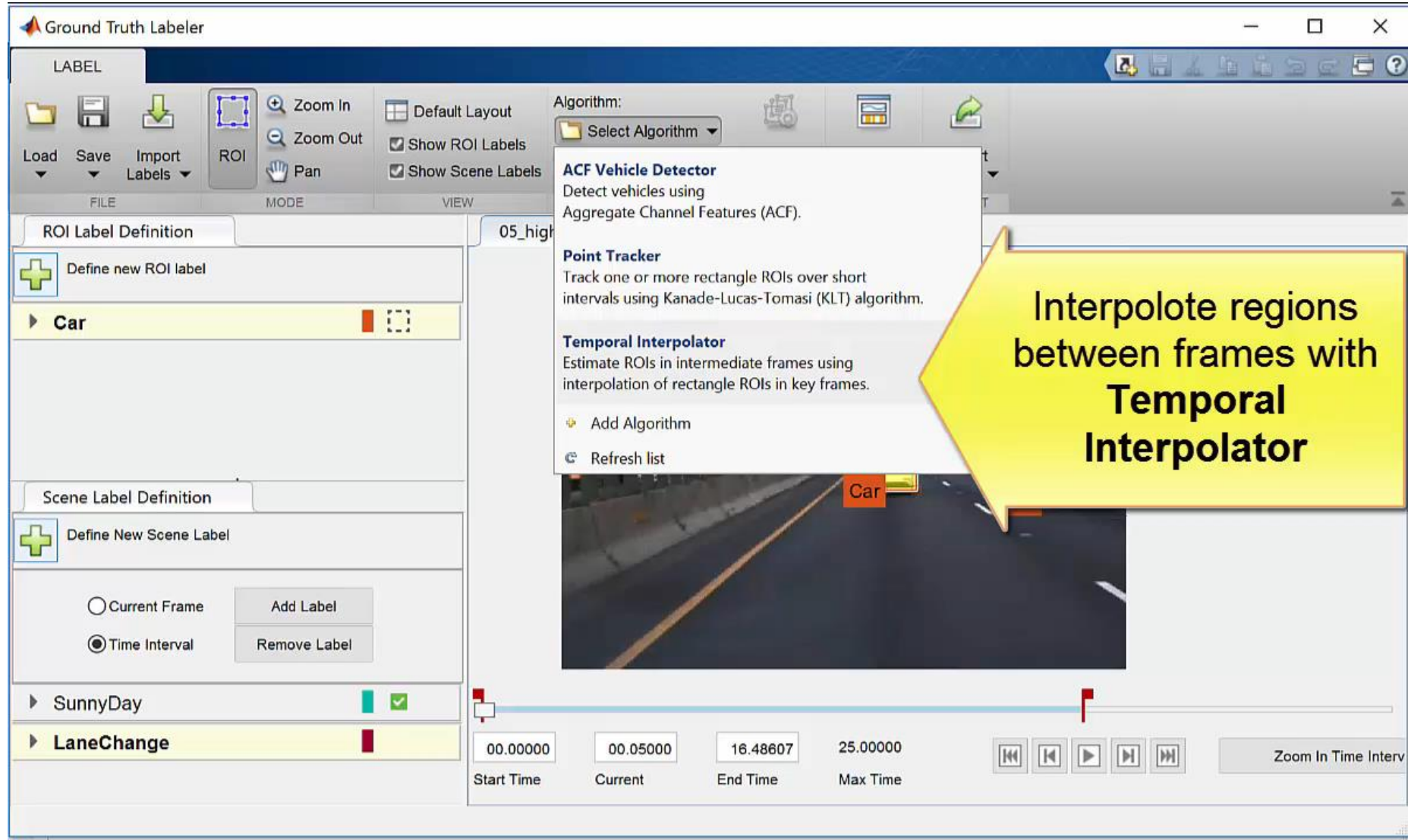


Label ground truth with
Ground Truth Labeler App

Command Window
fx >>

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 main window is titled "Ground Truth Labeler" and features a toolbar with options like "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, including "Car".
- Scene Label Definition:** Contains a "Define New Scene Label" button, radio buttons for "Current Frame" and "Time Interval", and "Add Label" and "Remove Label" buttons.
- Algorithm List:** A dropdown menu is open, showing three 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.
- Video Player:** A video frame is shown with a "Car" label overlaid on a road scene.
- Timeline:** A horizontal timeline at the bottom shows the current frame at 16.48607, with markers for Start Time (00.00000), Current (00.05000), End Time (16.48607), and Max Time (25.00000). Navigation buttons and a "Zoom In Time Interv" button are also present.

A yellow callout box with a black border points to the "Temporal Interpolator" algorithm, containing the text: "Interpolate regions between frames with Temporal Interpolator".

在手工标注帧的基础上自动标注 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 yellow arrow points to the "Point Tracker" algorithm in the "Algorithm:" dropdown menu. The interface includes a toolbar with "Load", "Save", "Import Labels", and "ROI" buttons. The "Algorithm:" dropdown menu is open, showing the following options:

- Temporal Interpolator**
 - 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

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 timeline at the bottom with a "Zoom In Time Interval" button.

Track region with
Point Tracker

Start Time	Current	End Time	Max Time
06.75310	06.75310	14.54546	25.00000

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:" dropdown menu, with the text "Detect initial regions with Vehicle Detector".

The interface includes a top toolbar with icons for Load, Save, Import Labels, ROI, Zoom In, Zoom Out, Pan, Default Layout, Show ROI Labels, and Show Scene Labels. The "Algorithm:" dropdown menu is open, showing the following options:

- 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.

Below the algorithm list are buttons for "Add Algorithm" and "Refresh list". The "Car" label is highlighted in the "ROI Label Definition" section. The "Scene Label Definition" section 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.

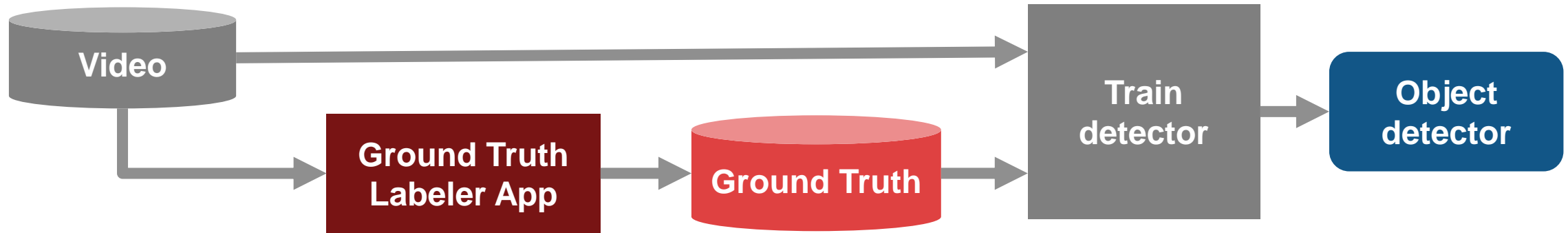
Start Time	Current	End Time	Max Time
14.54546	14.54546	21.12013	25.00000

以MATLAB时间表方式导出标注区域

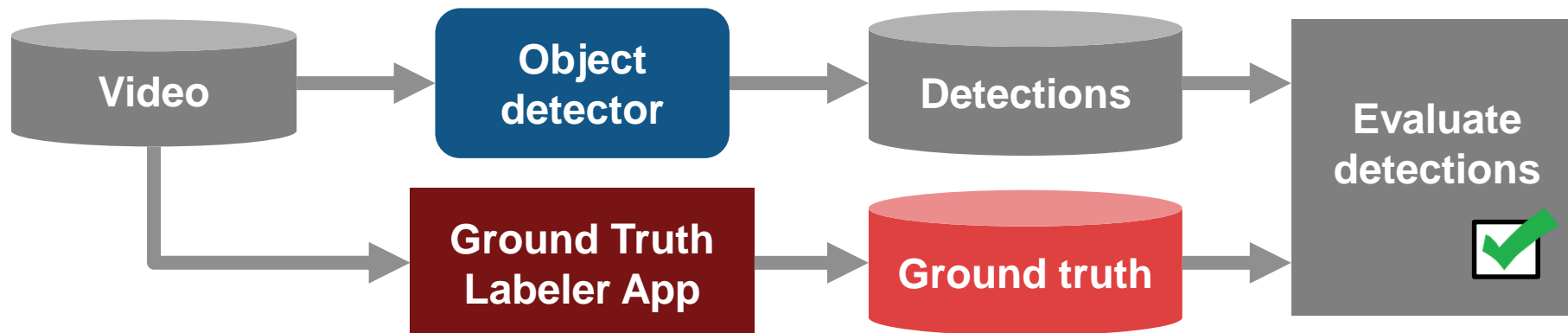
The screenshot displays the Ground Truth Labeler application window. A large yellow arrow points to the 'View Label Summary' button in the top toolbar, with the text 'Explore labels by viewing label summary' overlaid on it. The interface is divided into several sections:

- Top Toolbar:** Includes 'Load', 'Save', 'Import Labels', 'View Label Summary', and 'Export Labels' buttons.
- Left Panel:** Contains 'ROI Label Definition' and 'Scene Label Definition' sections. Under 'ROI Label Definition', 'Car' is selected. Under 'Scene Label Definition', 'SunnyDay' and 'LaneChange' are listed with checkboxes.
- Center Video:** Shows a frame from a video labeled '05_highway_lanechange_25s.mp4' with three cars highlighted by orange bounding boxes and labeled 'Car'.
- Right Panel:** Displays 'Scene Labels' with a legend for 'SunnyDay' (teal) and 'LaneChange' (red).
- Bottom Timeline:** Features a horizontal timeline with markers for 'Start Time' (00.00000), 'Current' (14.55000), 'End Time' (25.00000), and 'Max Time' (25.00000). It includes playback controls and a 'Zoom In Time Interval' button.

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 blue), 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 software interface. The main window is titled "Ground Truth Labeler - gtlCustomizations". The interface is divided into several sections:

- Top Bar:** Contains icons for Load, Save, Import Labels, ROI, Zoom In, Zoom Out, Pan, Default Layout, Show ROI Labels, Show Scene Labels, Algorithm (Select Algorithm, Configure Automation), Automate, View Label Summary, and Export Labels.
- Left Panel:** Contains sections for DATA SOURCE, LABEL DEFINITIONS, and SESSION. The DATA SOURCE section is expanded, showing Video, Image Sequence, and Custom Reader (highlighted with a red box).
- Center Panel:** Displays a video frame with ground truth labels for "Car", "Pedestrian", and "Lane". A yellow callout box with a large arrow points to the Custom Reader option in the DATA SOURCE section, containing the text: "Add custom image reader with `groundTruthDataSource`".
- Bottom Panel:** Contains a timeline with Start Time (00.00000), Current (09.00000), End Time (10.20000), and Max Time (10.20000). It also includes playback controls (Play, Stop, Previous, Next) and a Zoom button.

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:** Contains 'LABEL' and various icons for file operations (Load, Save, Import Labels) and viewing (Zoom In, Zoom Out, Pan).
- Left Panel:** Features 'ROI Label Definition' and 'Scene Label Definition' sections. Under 'ROI Label Definition', there are predefined labels: Car (orange), Pedestrian (yellow), StopLight (green), and Lane (blue). A 'Define new ROI label' button is also present.
- 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:** An 'Algorithm:' dropdown menu is open, listing 'Point Tracker', 'Temporal Interpolator', and 'ACF Vehicle Detector'. Below the list, there are buttons for 'Add Algorithm', 'Refresh list', 'Create New Algorithm' (highlighted with a red box), and 'Import Algorithm'.

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 image displays the Ground Truth Labeler software interface. The main window, titled "Ground Truth Labeler - gtlCustomizations", features a menu bar with "FILE", "MODE", and "VIEW". The "MODE" section includes "ROI" and "Pan" tools. The "VIEW" section includes "Default Layout", "Show ROI Labels", and "Show Scene Labels". The "Algorithm:" section includes "Select Algorithm", "Automate", "View Label", and "Export".

The interface is divided into several panels:

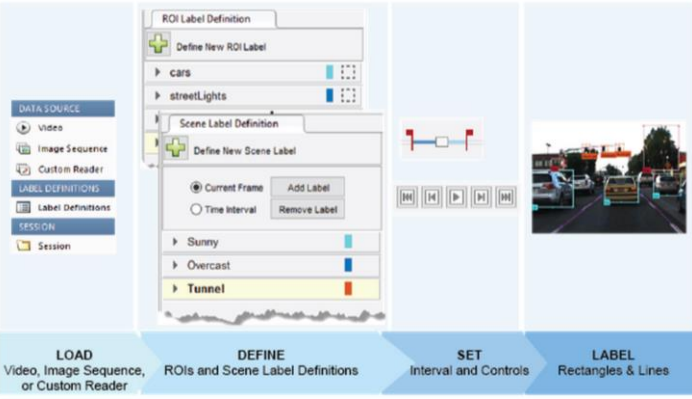
- ROI Label Definition:** Contains a "Define new ROI label" button and a list of labels: Car, Pedestrian, StopLight, and Lane. The "Lane" label is currently selected.
- Scene Label Definition:** Contains a "Define New Scene Label" button and options for "Current Frame" and "Time Interval".
- Main View:** Displays a video frame with a car, a pedestrian, and a lane. Labels are applied to these objects, and blue lines connect them to a point cloud visualization.
- Timeline:** Shows a progress bar with "Start Time" (00.00000), "Current" (09.00000), "End Time" (10.20000), and "Max Time" (10.20000). Playback controls and a "Zoom" button are also present.

A yellow callout box with a large arrow points to the "driving.connector.Connector" tool, with the text: "Add connection to other tools with **driving.connector.Connector**".

The point cloud visualization, titled "Figure 1: Point Cloud Pla...", shows a 3D view of the scene with blue lines representing the lane and other objects. The point cloud is rendered in blue and green, with the lane highlighted in blue.

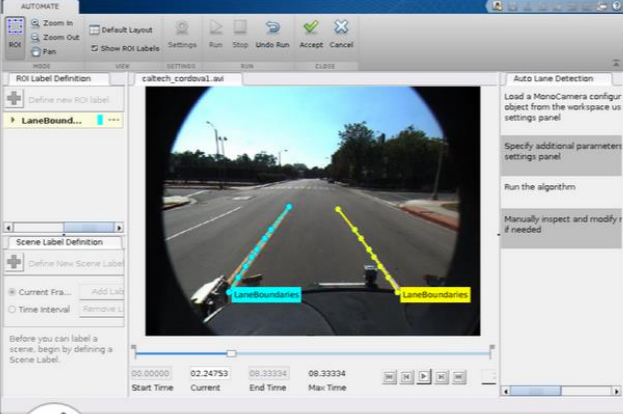
了解更多在图像中检测目标

查看Automated Driving System Toolbox中的例子



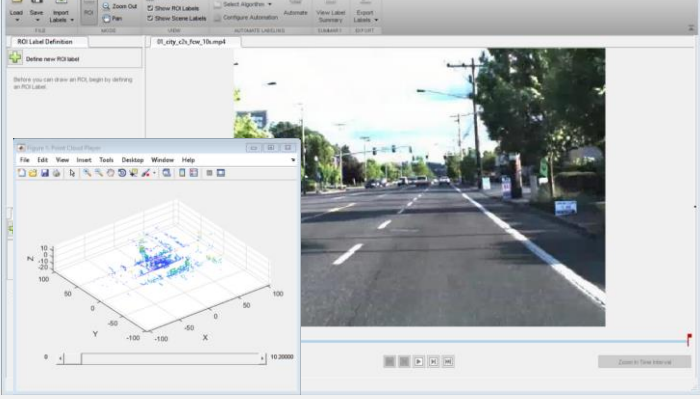
Define Ground Truth Data for Video or Image Sequences

- 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

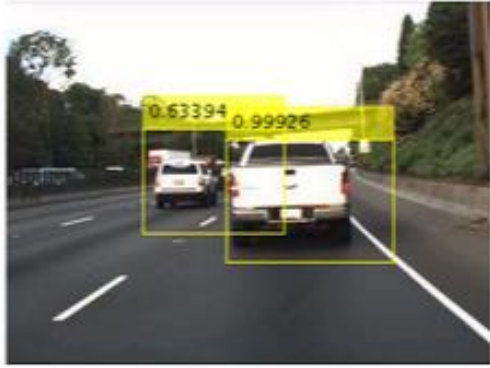


driving.connector.Connector class
Connect Lidar Display to Ground Truth Labeler

- 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**



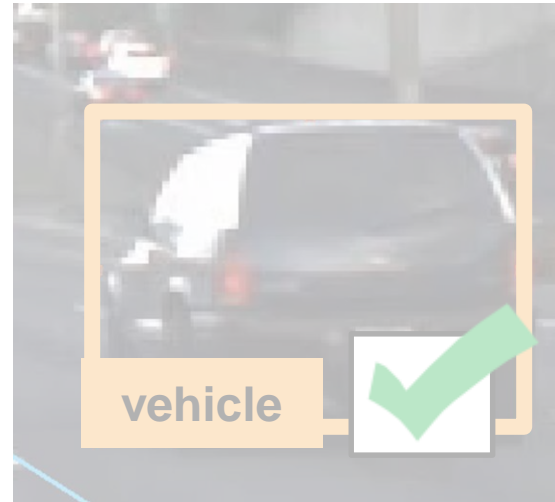
Visual Perception Using Monocular Camera

- **考察车道检测器**
根据摄像头传感器模型转换坐标系

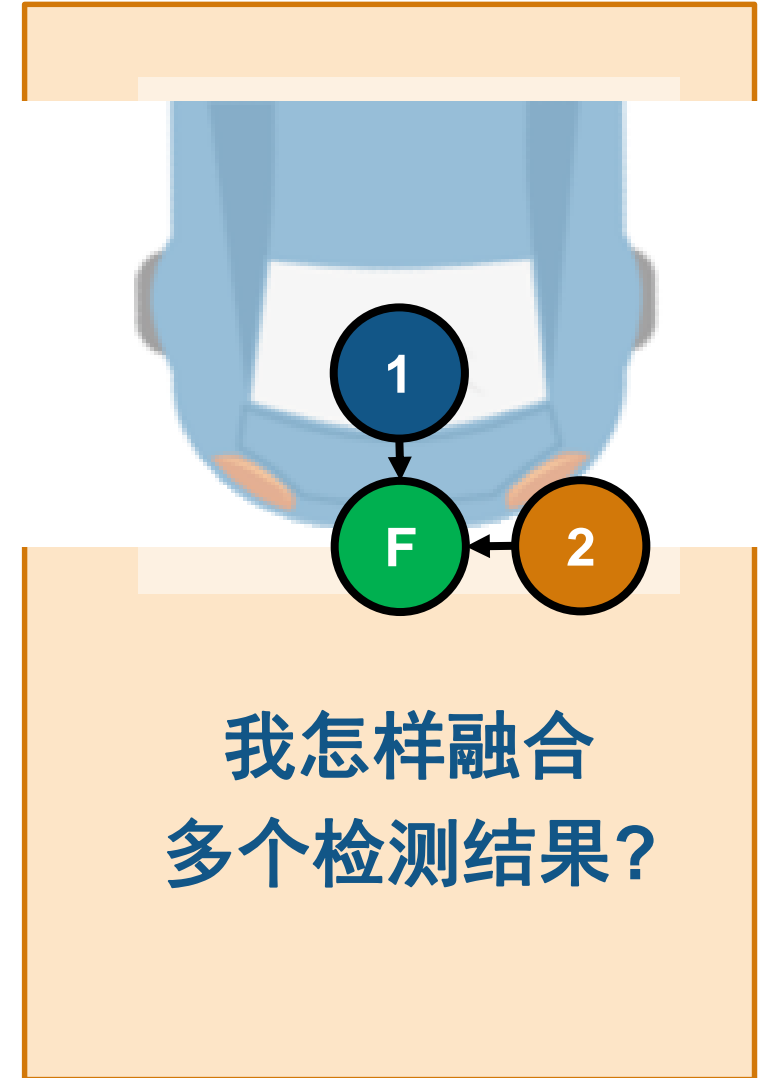
自动驾驶工程师经常遇到的问题：



我怎样可视化
车辆的数据？

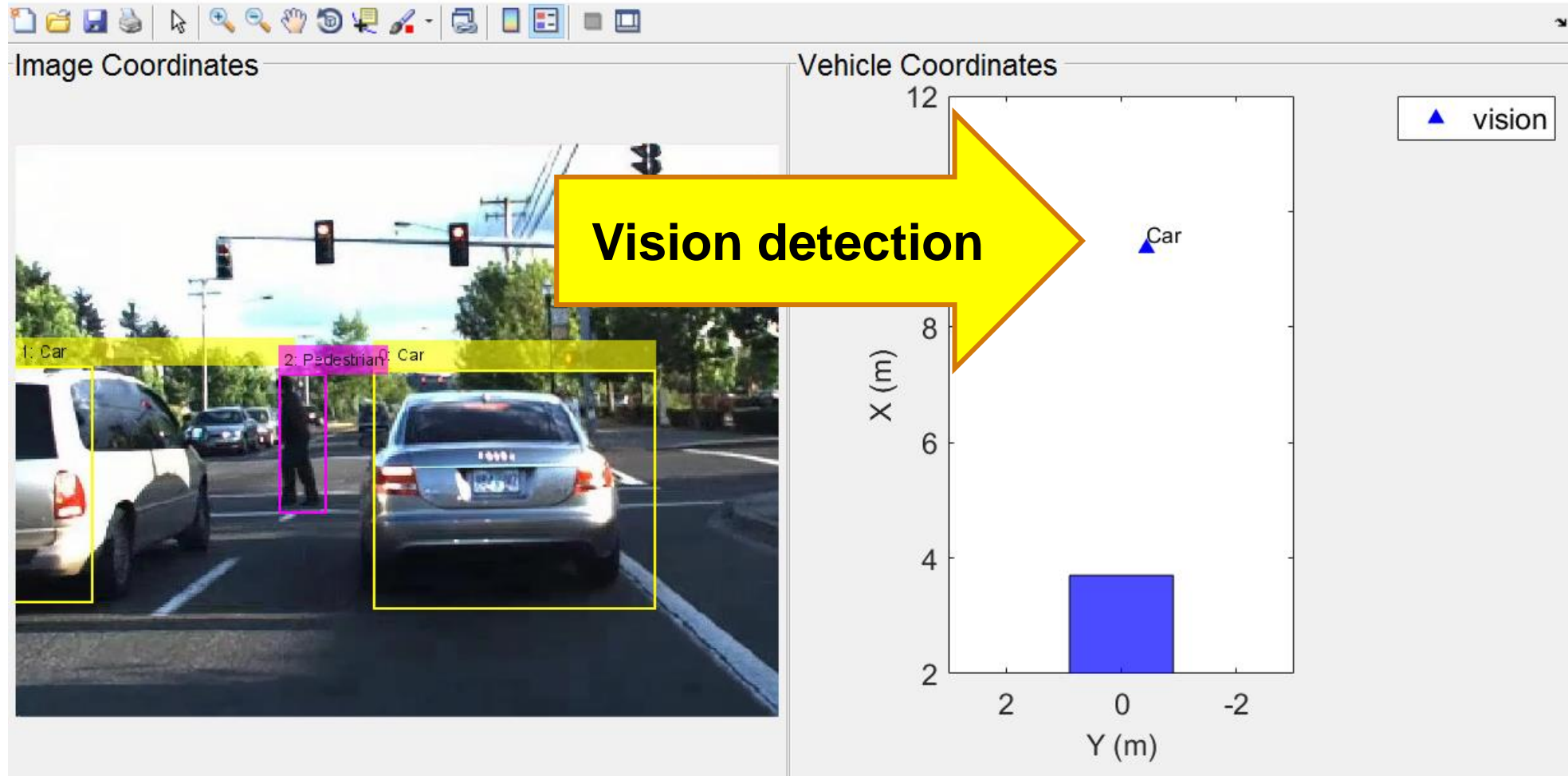


我怎样检测图
像中的目标？

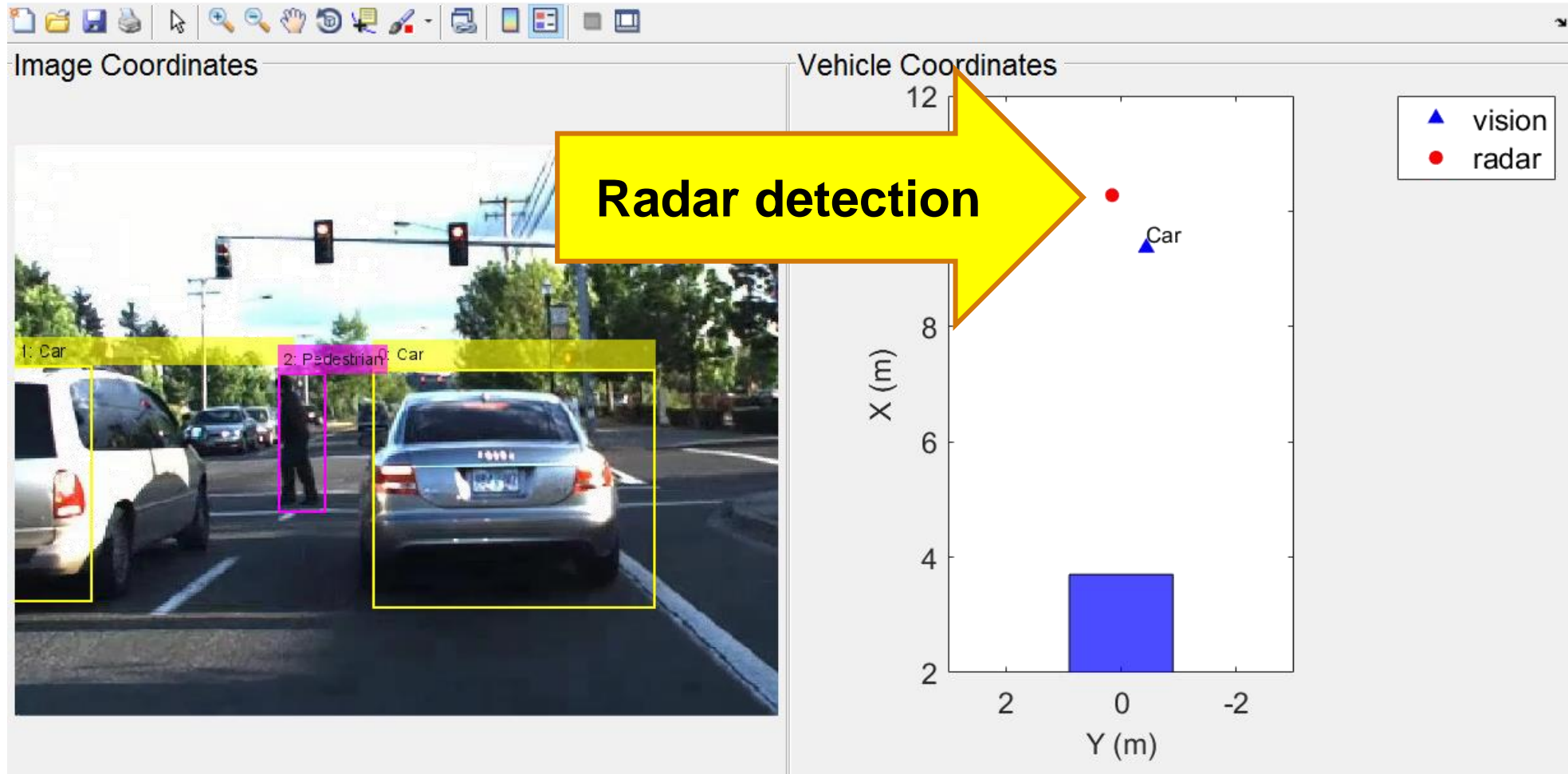


我怎样融合
多个检测结果？

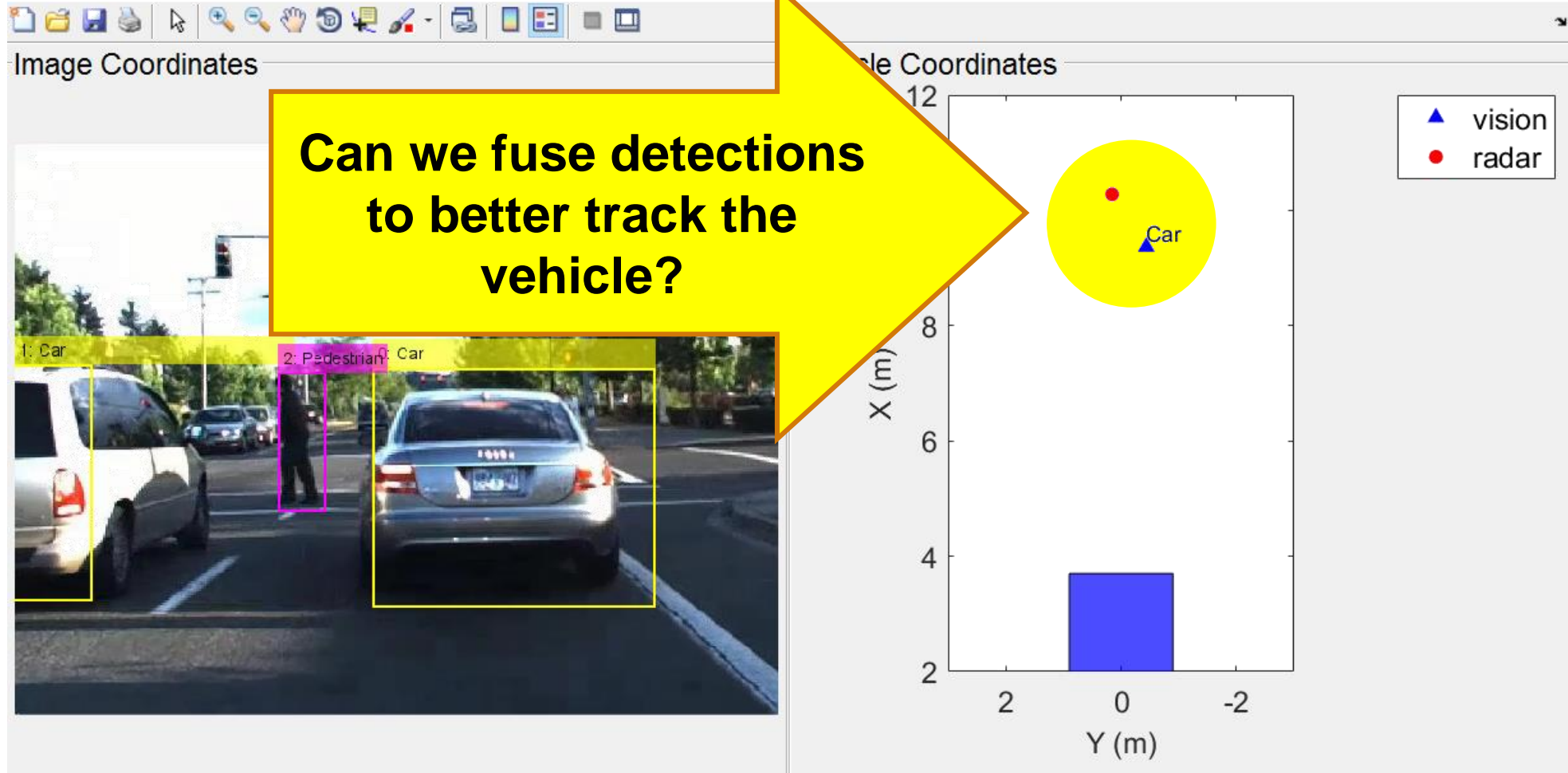
雷达和视觉检测车辆的例子



雷达和视觉检测车辆的例子

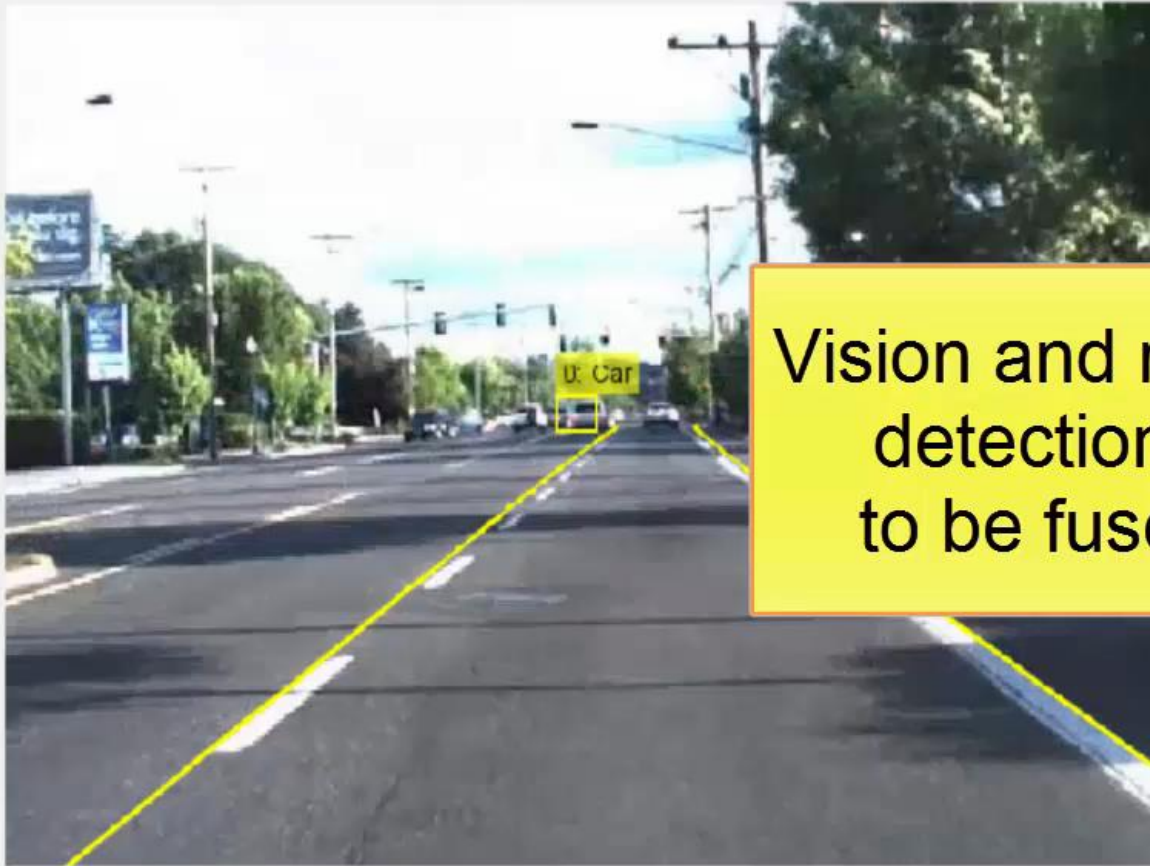


雷达和视觉检测车辆的例子

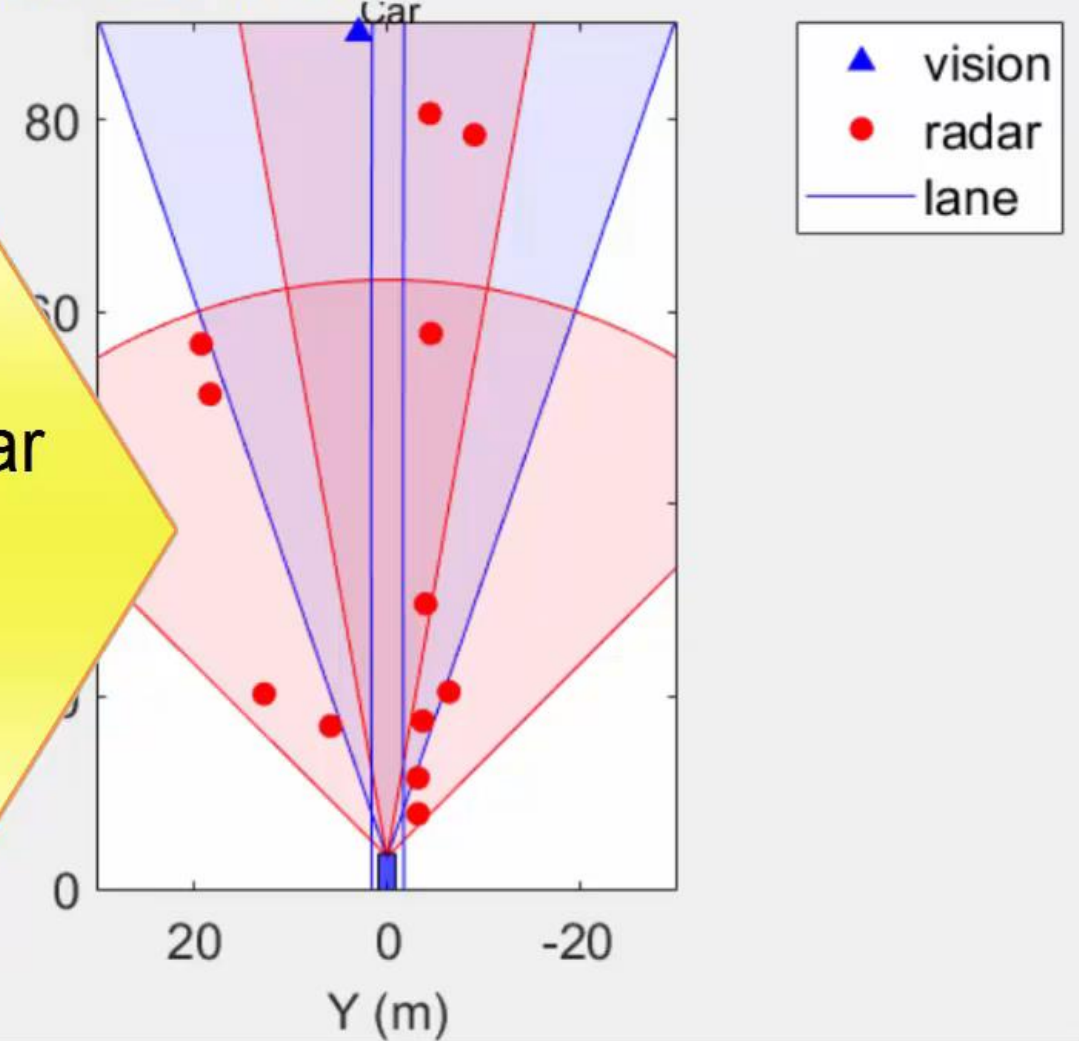


用多目标跟踪器融合检测目标

Image Coordinates



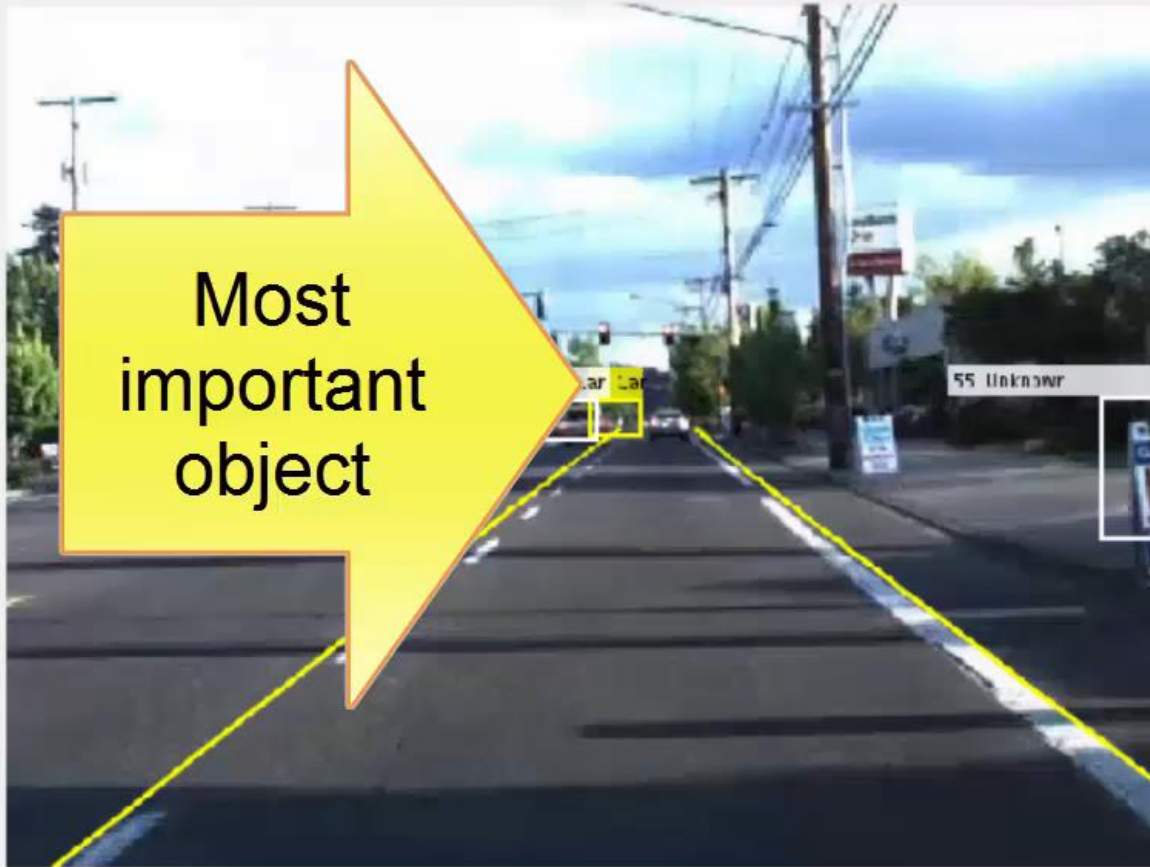
Vehicle Coordinates



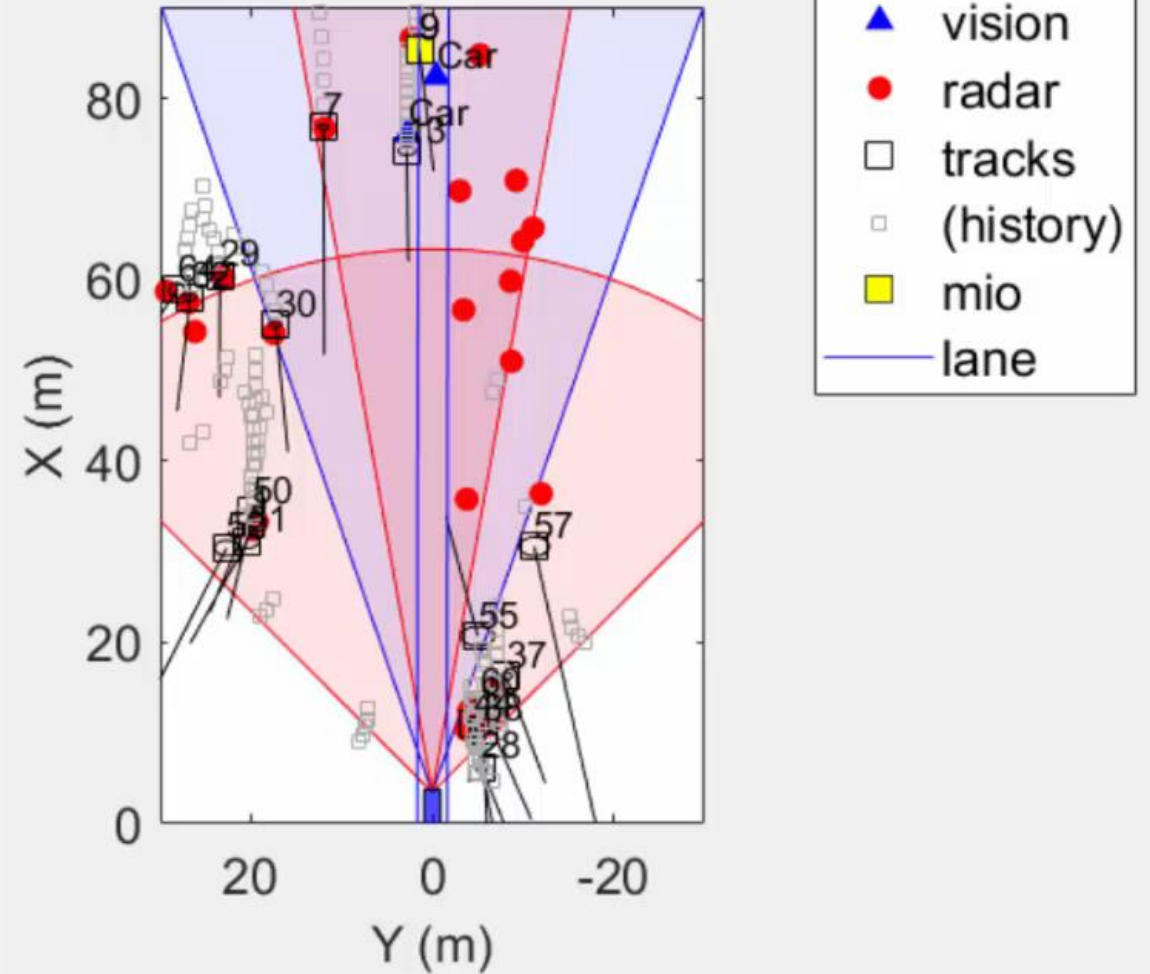
Vision and radar
detections
to be fused

将跟踪器集成到更上层的算法中

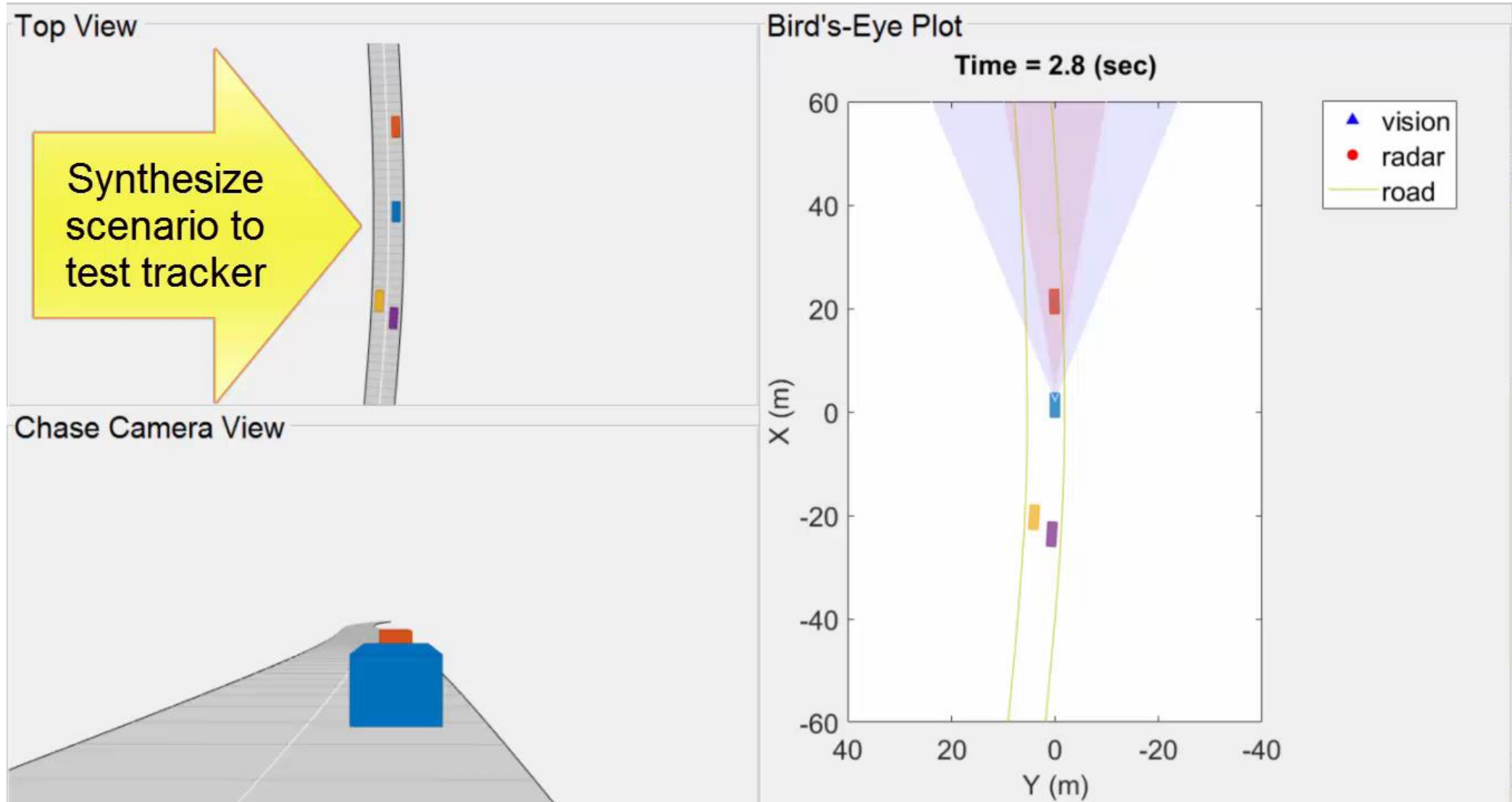
Image Coordinates



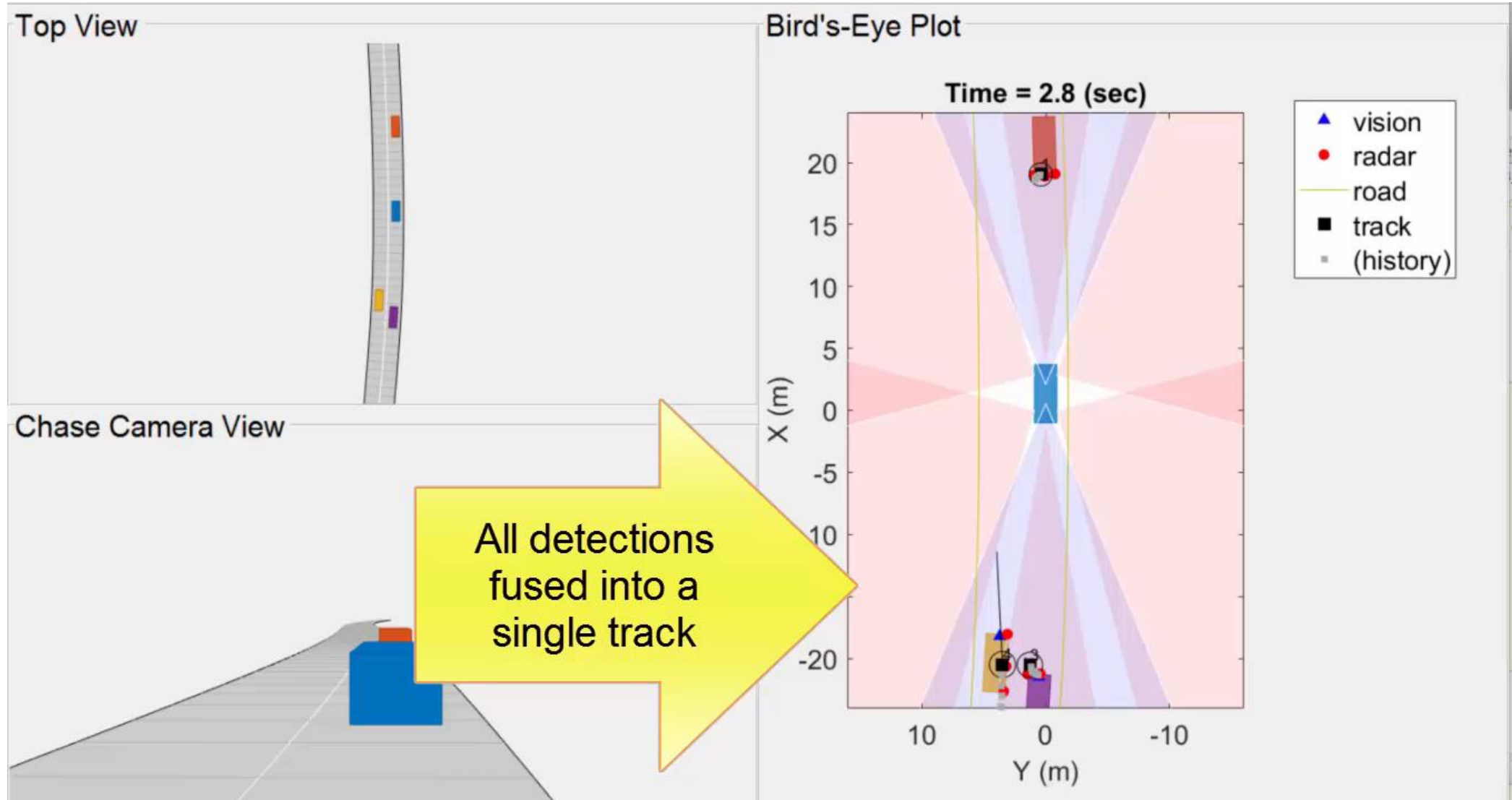
Vehicle Coordinates



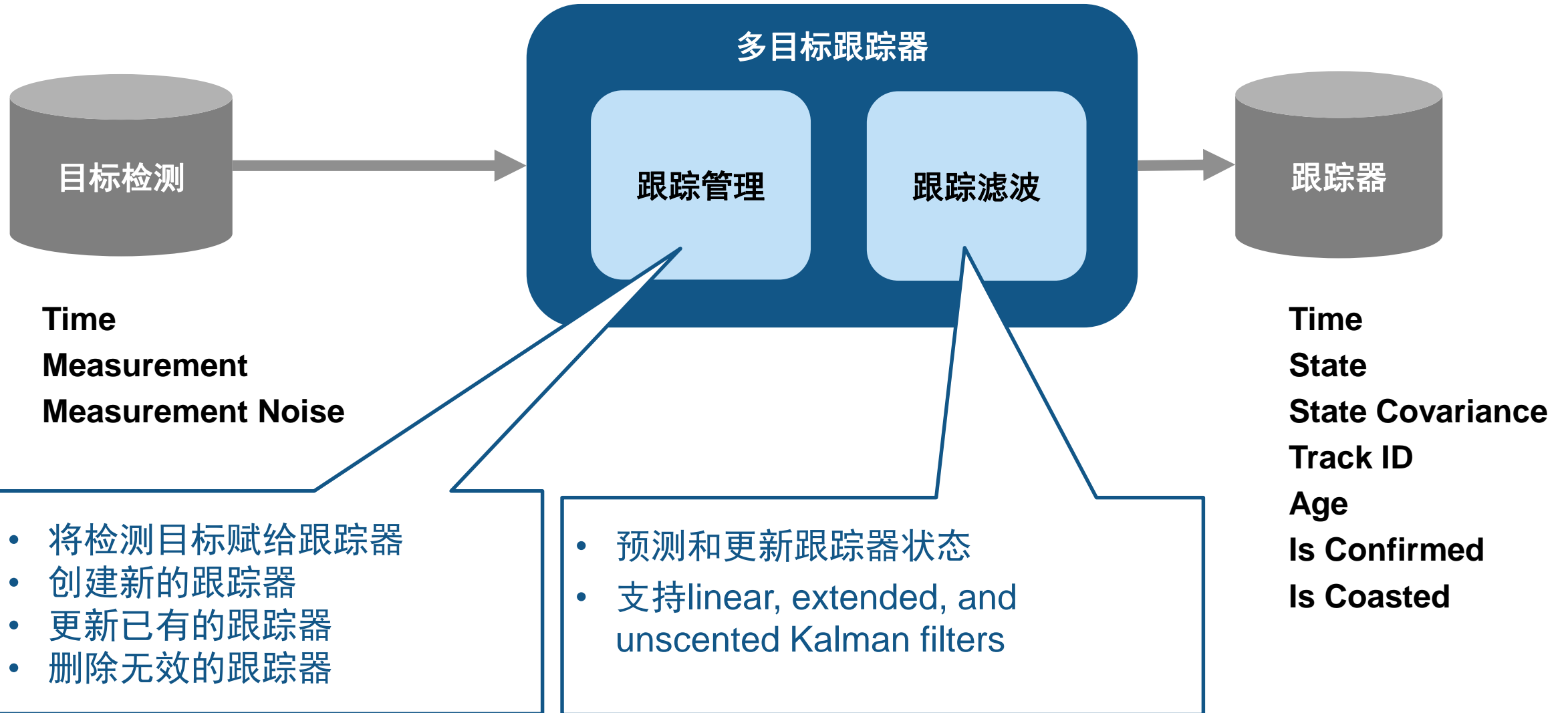
生成交通场景测试跟踪器



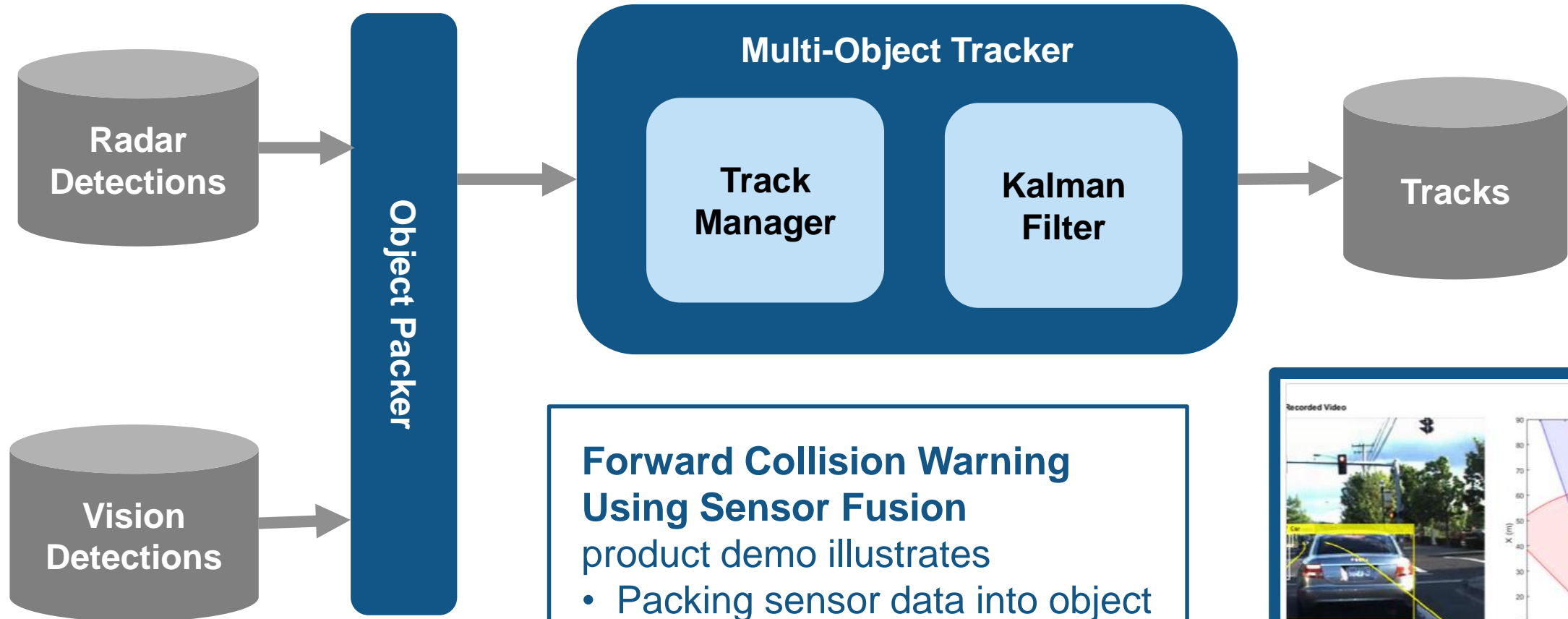
用合成的数据测试跟踪器



跟踪多目标检测



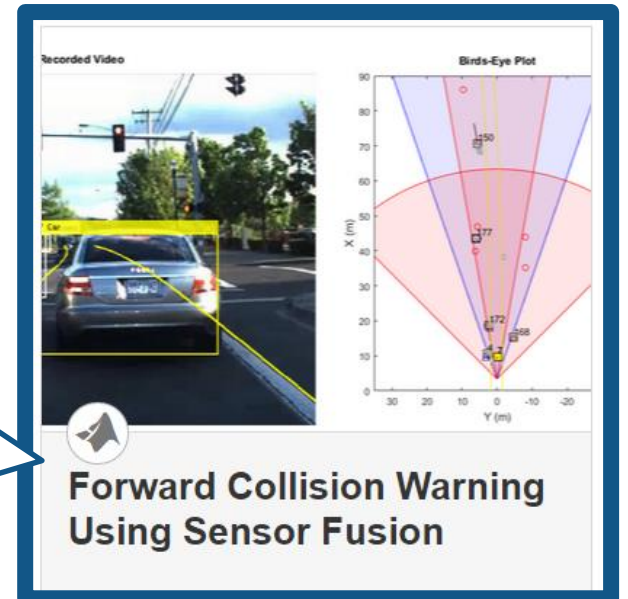
通过例子了解更多传感器融合



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)
```

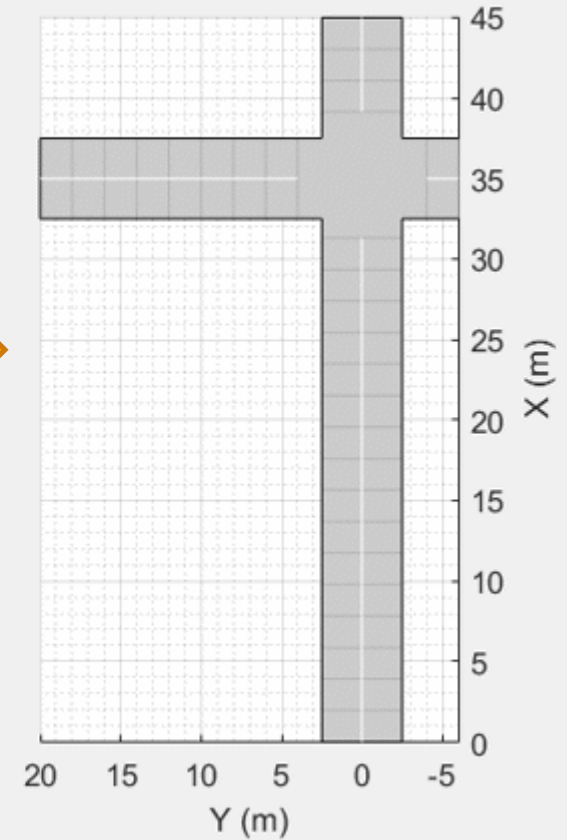
指定驾驶场景和道路

```
%% 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);

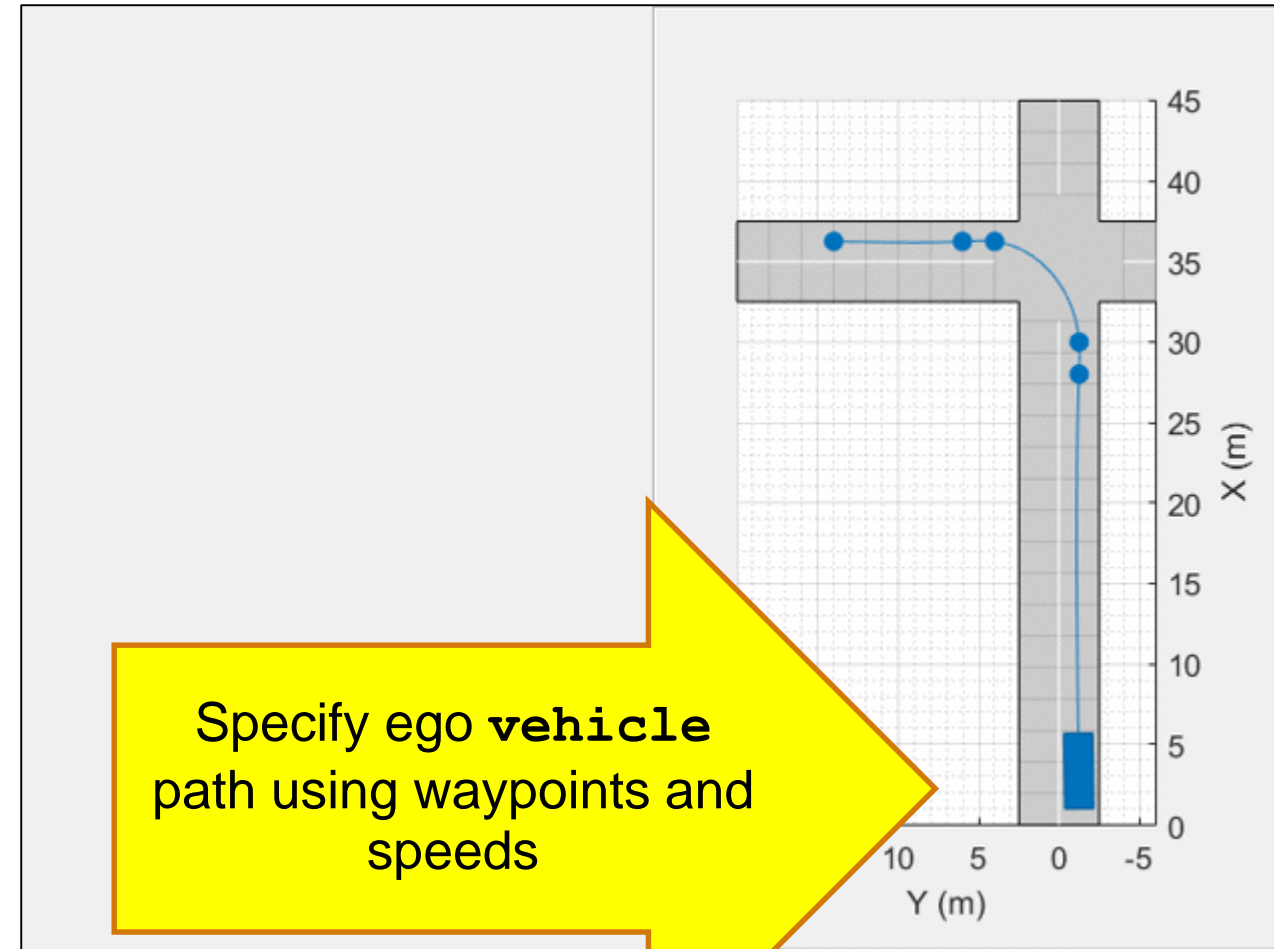
%% Plot scenario
p1 = uipanel('Position', [0.5 0 0.5 1]);
a1 = axes('Parent', p1);
plot(s, 'Parent', a1, ...
      'Centerline', 'on', 'Waypoints', 'on')
a1.XLim = [0 45];
a1.YLim = [-6 20];
```

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）

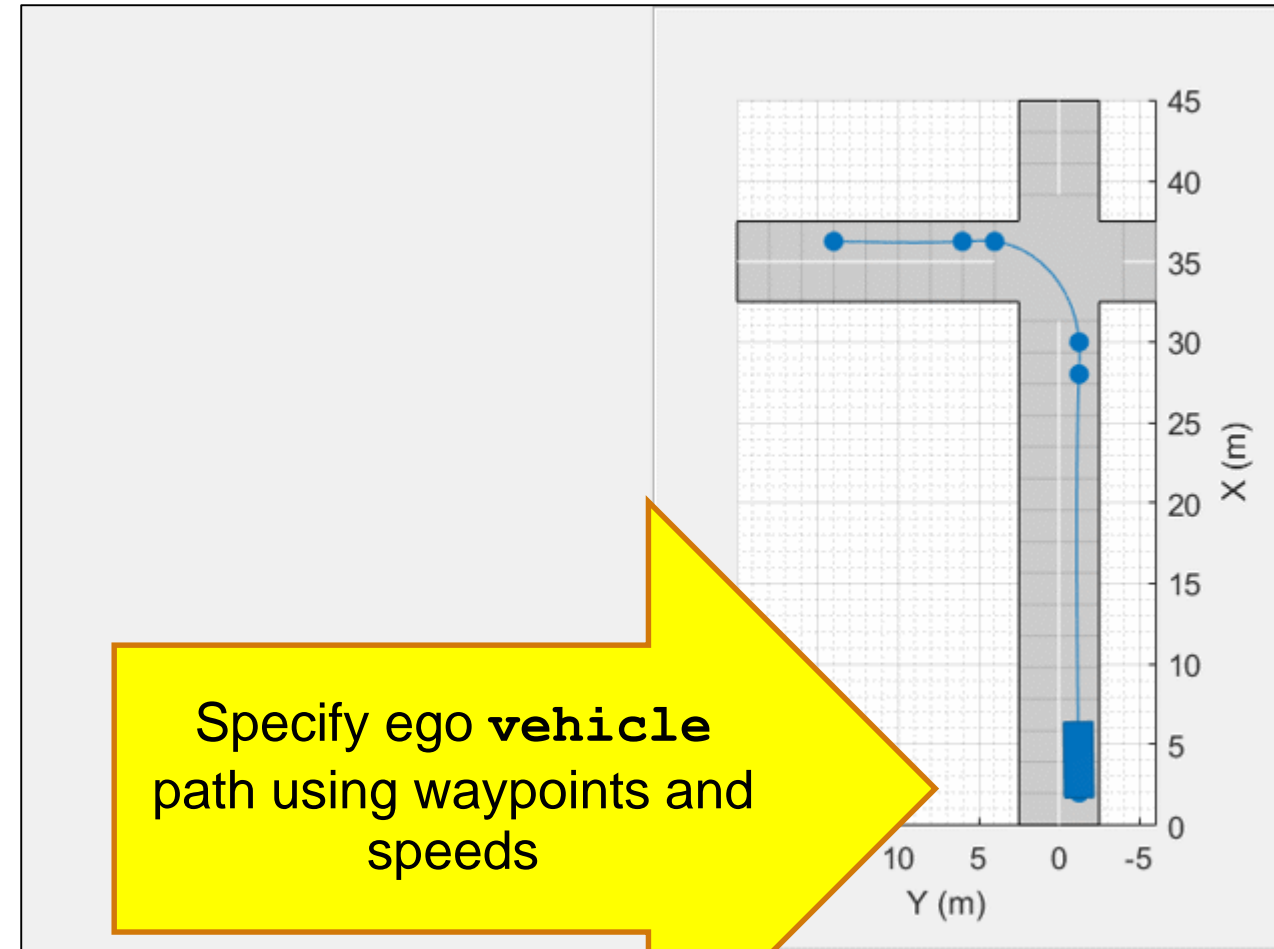
```

%% 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

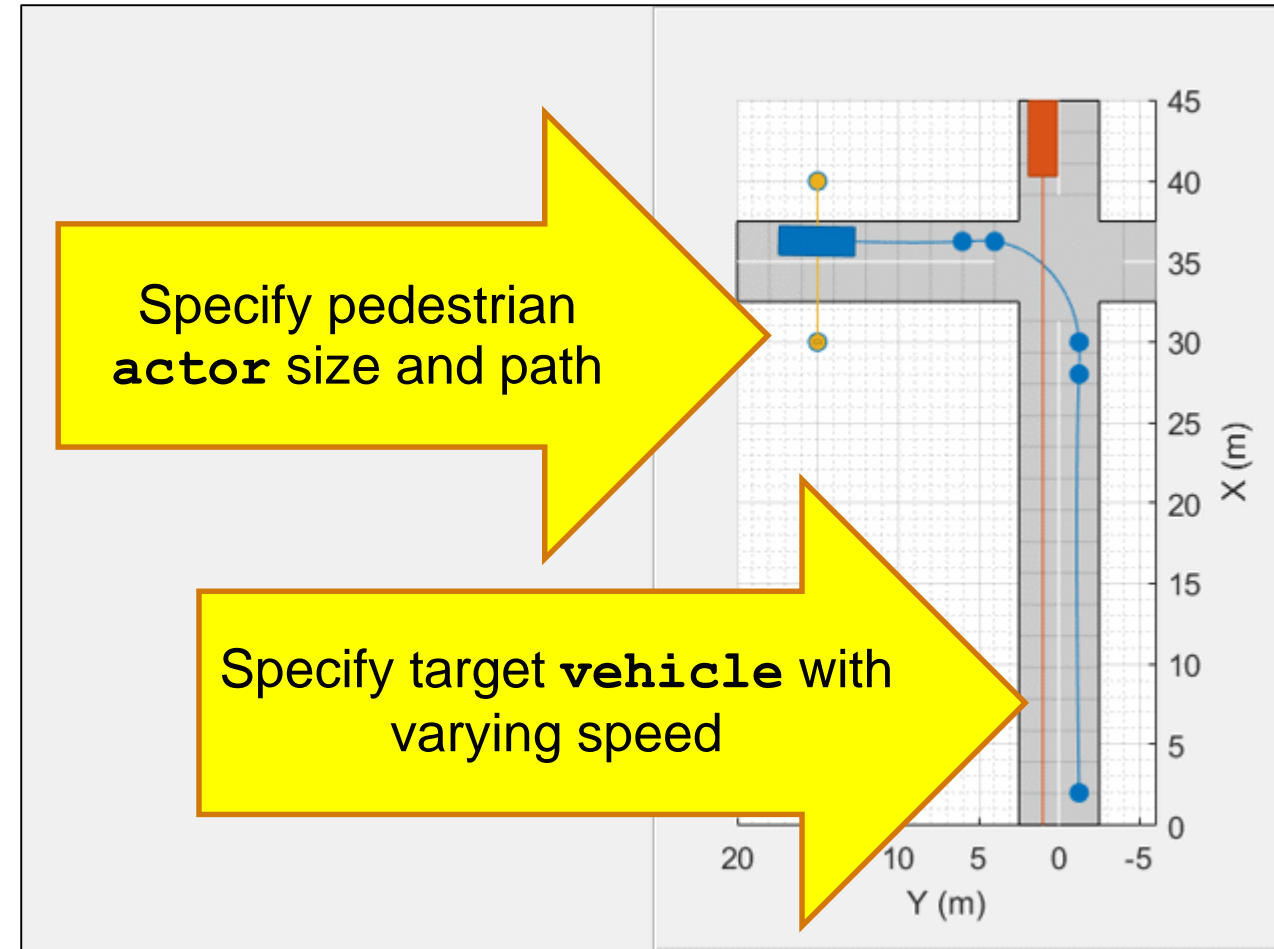
```

%% 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);

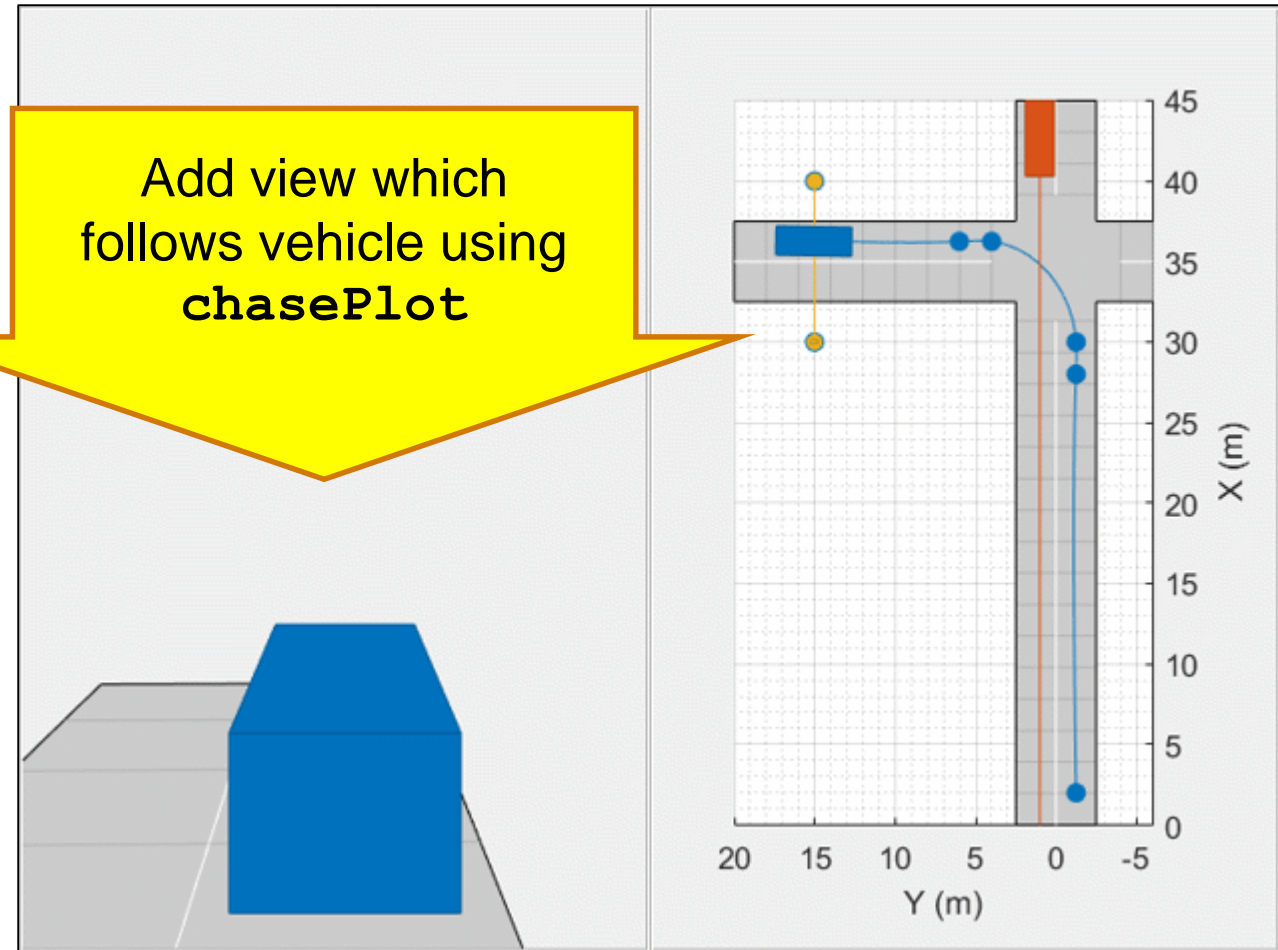
path(child,...
    [30 15; 40 15],... % Waypoints (m)
    1.39); % Speed (m/s) = 5 km/hr

```



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)
```



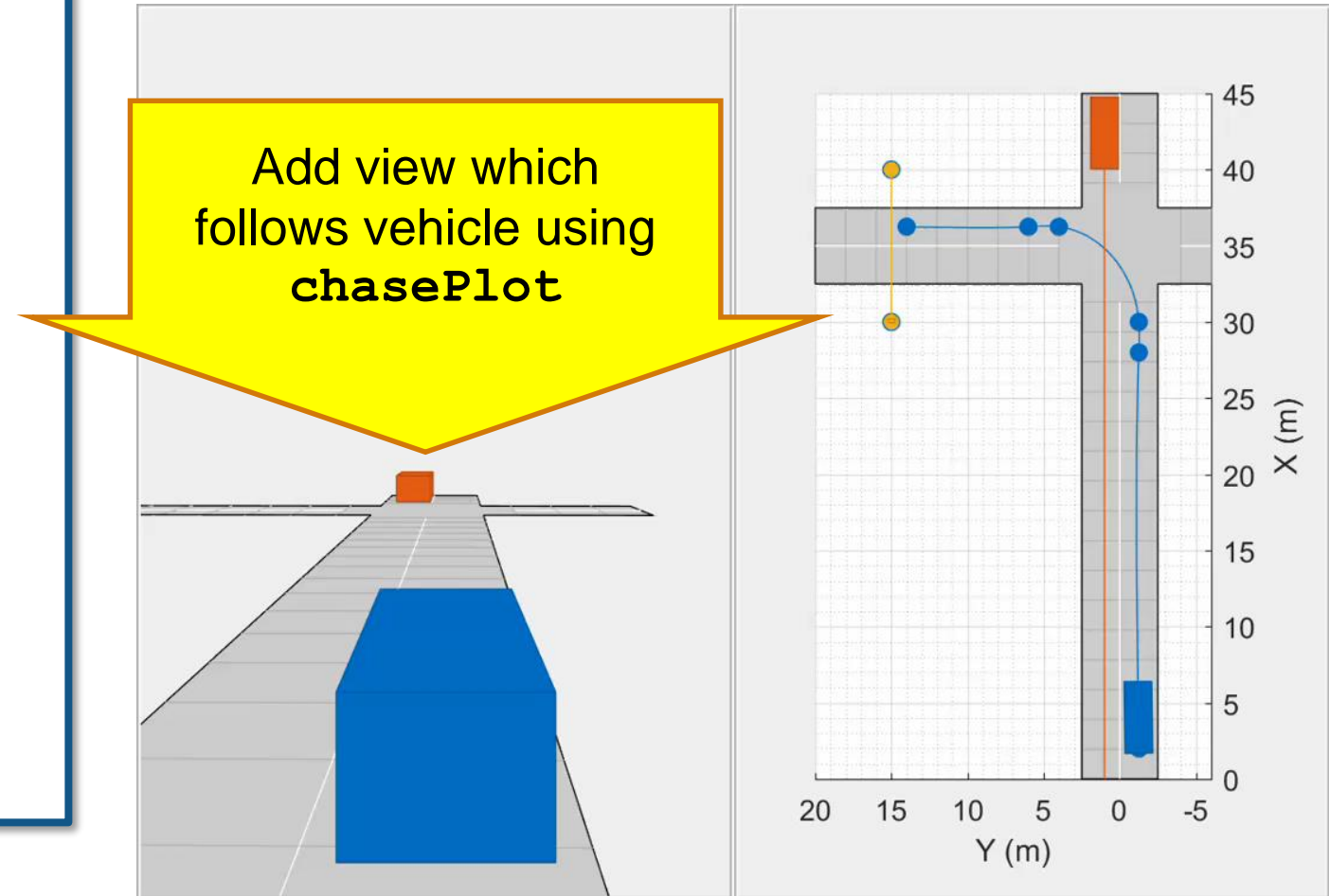
以本体车辆的后方视角观察场景 vehicle

```

%% 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)

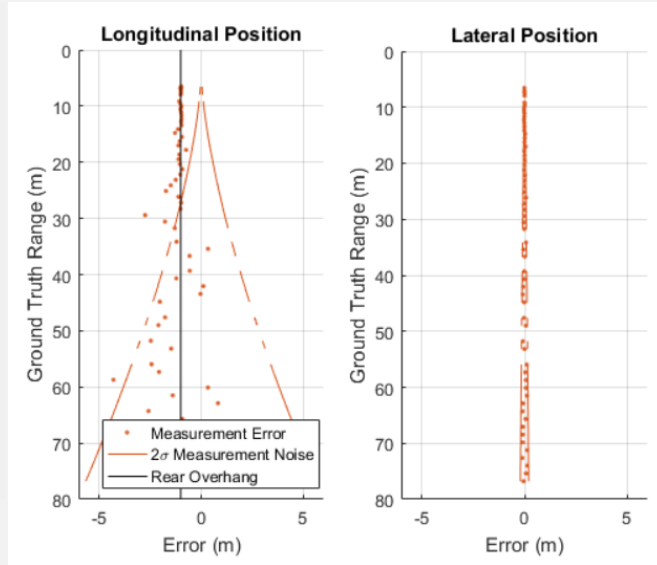
%% Play scenario
restart(s)
while advance(s)
    pause(s.SampleTime);
end

```



仿真视觉传感器目标检测的效应

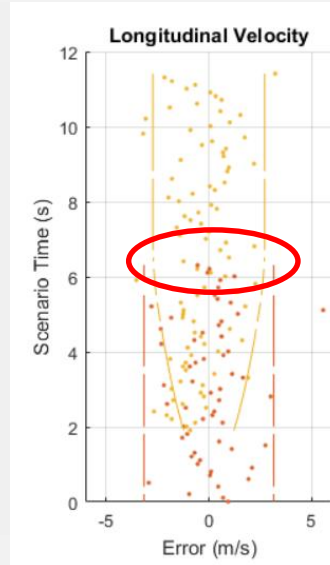
距离效应



距离测量精度
随着目标距离
增加而降低

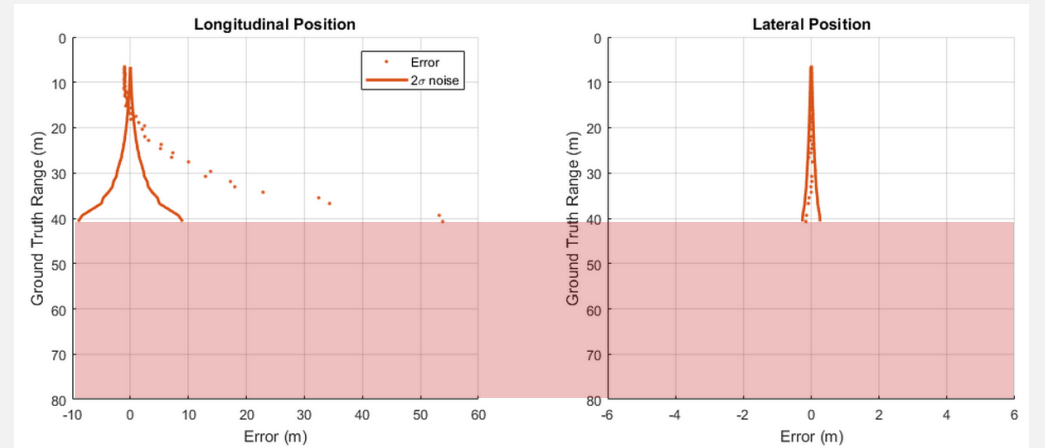
角度测量精度
在覆盖范围内
保持一致

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

```

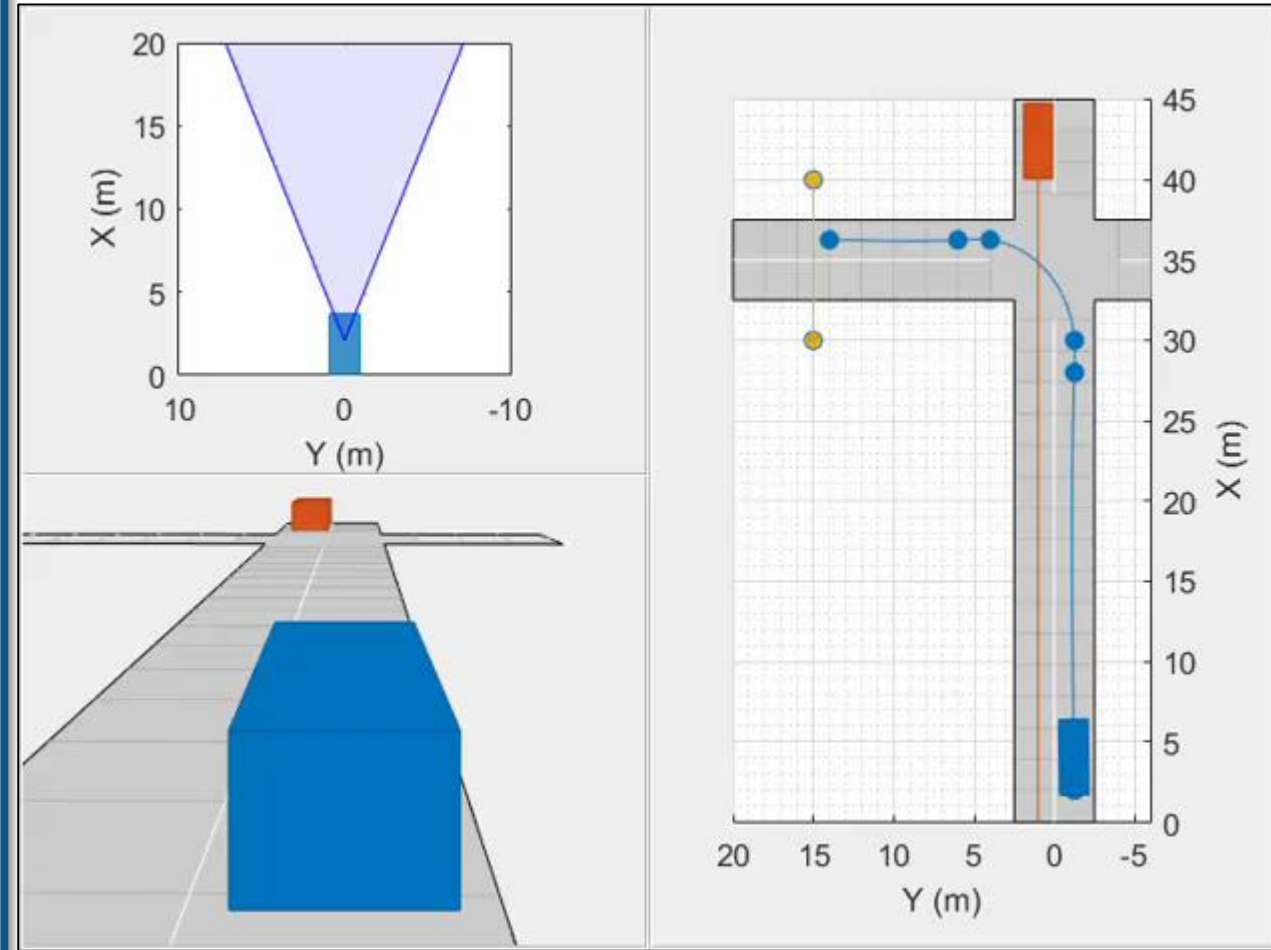
restart(s)
while advance(s)
    % Get detections in ego vehicle coordinates
    det = sensor(targetPoses(egoCar),...
                s.SimulationTime);

    % 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

```

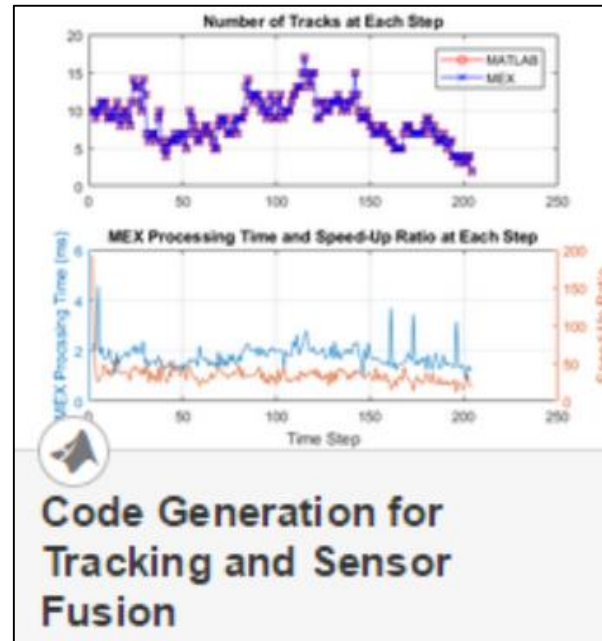


了解更多传感器融合

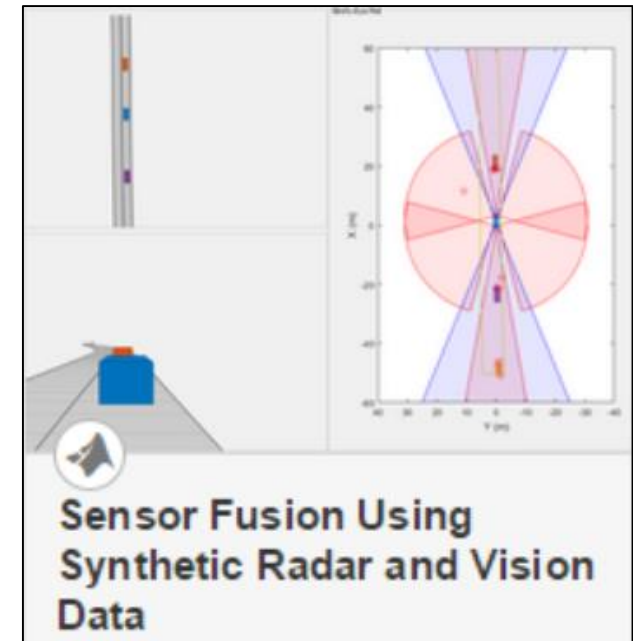
查看Automated Driving System Toolbox中的例子



- 设计
基于记录的车辆数据
设计目标跟踪器

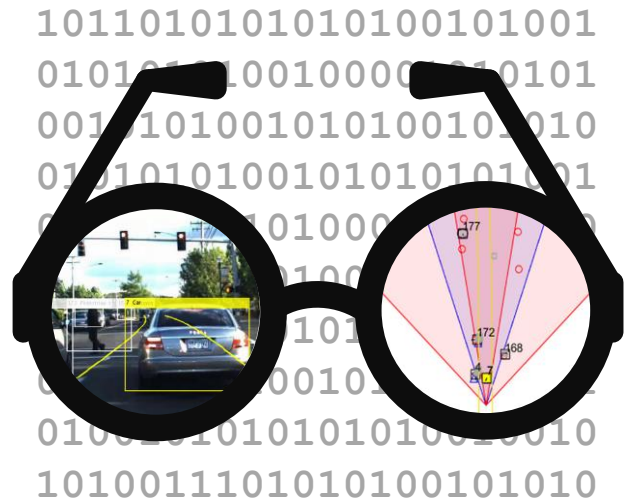


- 生成 C/C++代码
将多目标跟踪器
生成代码



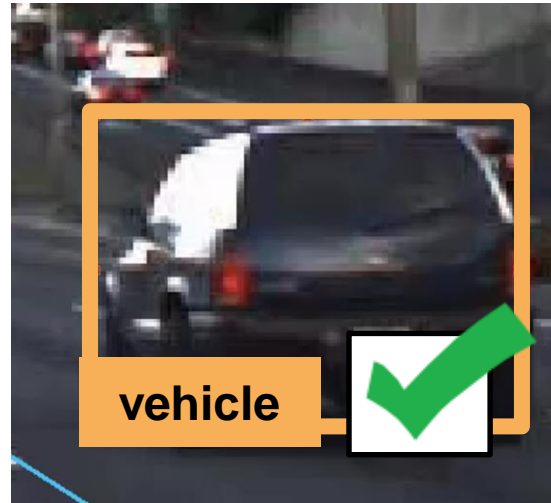
- 合成驾驶场景
测试多目标跟踪器

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



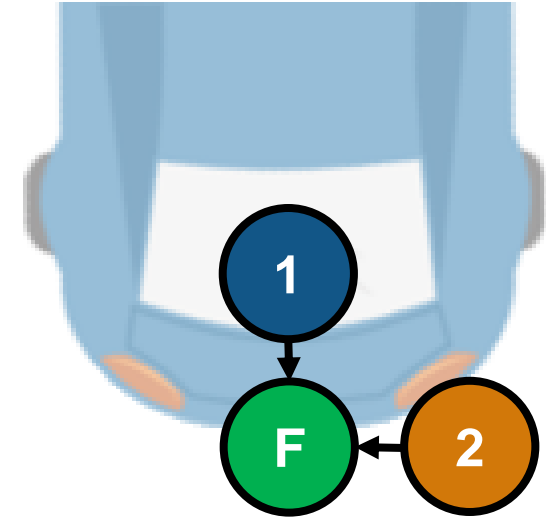
可视化车辆数据

- 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