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

MATLAB Expo Germany 2017

Anders Sollander
Principal Consultant
Some common questions from automated driving engineers

How can I visualize vehicle data?

How can I detect objects in images?

How can I fuse multiple detections?
Some common questions from automated driving engineers

How can I visualize vehicle data?
How can I detect objects in images?
How can I fuse multiple detections?
Examples of automated driving sensors

- Camera
- Radar-based object detector
- Vision-based object detector
- Lidar
- Lane detector
- Inertial measurement unit
Examples of automated driving sensor data

**Camera** $(640 \times 480 \times 3)$

- **Vision-based lane detector**
- **Radar-based object detector**
- **Lidar**
- **Inertial measurement unit**

**Lane detector**

**Vision-based object detector**
Examples of automated driving sensor data

Camera (640 x 480 x 3)

Vision Detector

SensorID = 1;
Timestamp = 1461634696379742;
NumDetections = 6;
Detections(1)
  TrackID:        0
  Classification: 5
  Position:      [22.61 -0.43 2.24]
  Velocity:      [-9.86 0    0]
  Size:          [0      1.75 0]

Detections(2)
  TrackID:        1
  Classification: 5
  Position:      [22.8  3.12 2.24]
  Velocity:      [-9.37 0    0]
  Size:          [0     1.8  0]

Vision based lane detector

Inertial measurement unit

Lidar

Radar-based object detector

Lane detector
Examples of automated driving sensor data

**Camera** (640 x 480 x 3)

**Vision Detector**

- SensorID = 1;
- Timestamp = 1461634696379742;
- NumDetections = 6;
- Detections:
  - TrackID: 0
    - Classification: 5
    - Position: [22.61 -0.43 2.24]
    - Velocity: [-9.86 0 0]
    - Size: [0 1.75 0]
  - TrackID: 1
    - Classification: 5
    - Position: [22.8 3.12 2.24]
    - Velocity: [-9.37 0 0]
    - Size: [0 1.8 0]

**Lane Detector**

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

**Radar-based object detector**

**Lidar**

**Inertial measurement unit**
Examples of automated driving sensor data

**Camera** (640 x 480 x 3)

**Vision Detector**
- SensorID = 1;
- Timestamp = 1461634696379742;
- NumDetections = 6;
- Detections:
  - TrackID: 0; Classification: 5; Position: [22.61 -0.43 2.24]; Velocity: [-9.86 0 0]; Size: [0 1.75 0]
  - TrackID: 1; Classification: 5; Position: [22.8 3.12 2.24]; Velocity: [-9.37 0 0]; Size: [0 1.8 0]

**Radar Detector**
- SensorID = 2;
- Timestamp = 1461634696407521;
- NumDetections = 23;
- Detections:
  - TrackID: 0; TrackStatus: 6; Position: [56.07 17.73 0.34]; Velocity: [-8.50 2.86 0]; Amplitude: 3
  - TrackID: 1; TrackStatus: 6; Position: [35.35 19.59 0.34]; Velocity: [-8.02 4.92 0]; Amplitude: 3
  - TrackID: 12; TrackStatus: 5

**Lane Detector**
- Left:
  - IsValid: 1; Confidence: 3; BoundaryType: 3; Offset: 1.68; HeadingAngle: 0.002; Curvature: 0.0000228
- Right:
  - IsValid: 1; Confidence: 3
Examples of automated driving sensor data

### Camera (640 x 480 x 3)

- **Vision Detector**
  - SensorID = 1;
  - Timestamp = 1461634696379742;
  - NumDetections = 6;
  - Detections:
    - TrackID: 0
      - Classification: 5
      - Position: [22.61 -0.43 2.24]
      - Velocity: [-9.86 0 0]
      - Size: [0 1.75 0]
    - TrackID: 1
      - Classification: 5
      - Position: [22.8 3.12 2.24]
      - Velocity: [-9.37 0 0]
      - Size: [0 1.8 0]

### Radar Detector

- **SensorID** = 2;
- **Timestamp** = 1461634696407521;
- **NumDetections** = 23;
- **Detections**:
  - TrackID: 0
    - TrackStatus: 6
    - Position: [56.07 17.73 0.34]
    - Velocity: [-8.50 2.86 0]
    - Amplitude: 3
  - TrackID: 1
    - TrackStatus: 6
    - Position: [35.35 19.59 0.34]
    - Velocity: [-8.02 4.92 0]
    - Amplitude: 3
  - TrackID: 12
    - TrackStatus: 5
    - Position: [57.69 3.13 0.34]

### Lane Detector

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

### Lidar (47197 x 3)

- Inertial measurement unit
## Examples of automated driving sensor data

### Camera (640 x 480 x 3)

<table>
<thead>
<tr>
<th>SensorID</th>
<th>Timestamp</th>
<th>NumDetections</th>
<th>Detections(1)</th>
<th>Detections(2)</th>
<th>Detections(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TrackID: 0</td>
<td>TrackID: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Classification: 5</td>
<td>Classification: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Position: [22.61 -0.43 2.24]</td>
<td>Position: [22.8 3.12 2.24]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Velocity: [-9.86 0 0]</td>
<td>Velocity: [-9.37 0 0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Size: [0 1.75 0]</td>
<td>Size: [0 1.8 0]</td>
<td></td>
</tr>
</tbody>
</table>

### Vision Detector

<table>
<thead>
<tr>
<th>SensorID</th>
<th>Timestamp</th>
<th>NumDetections</th>
<th>Detections(1)</th>
<th>Detections(2)</th>
<th>Detections(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TrackID: 0</td>
<td>TrackID: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Classification: 5</td>
<td>Classification: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Position: [22.61 -0.43 2.24]</td>
<td>Position: [22.8 3.12 2.24]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Velocity: [-9.86 0 0]</td>
<td>Velocity: [-9.37 0 0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Size: [0 1.75 0]</td>
<td>Size: [0 1.8 0]</td>
<td></td>
</tr>
</tbody>
</table>

### Lane Detector

- **Left**
  - IsValid: 1
  - Confidence: 3
  - BoundaryType: 3
  - Offset: 1.68
  - HeadingAngle: 0.002
  - Curvature: 0.000

- **Right**
  - IsValid: 1
  - Confidence: 3

### Radar Detector

<table>
<thead>
<tr>
<th>SensorID</th>
<th>Timestamp</th>
<th>NumDetections</th>
<th>Detections(1)</th>
<th>Detections(2)</th>
<th>Detections(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TrackID: 0</td>
<td>TrackID: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Classification: 5</td>
<td>Classification: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Position: [56.07 17.73 0.34]</td>
<td>Position: [35.35 19.59 0.34]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Velocity: [-8.50 2.86 0]</td>
<td>Velocity: [-8.02 4.92 0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Amplitude: 3</td>
<td>Amplitude: 3</td>
<td></td>
</tr>
</tbody>
</table>

### Lidar (47197 x 3)

<table>
<thead>
<tr>
<th>TrackSt</th>
<th>Positio</th>
<th>Velocit</th>
<th>Amplitu</th>
<th>Detection</th>
<th>TrackID</th>
<th>TrackSt</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12.2911</td>
<td>1.4790</td>
<td>-0.5900</td>
<td>-14.8852</td>
<td>1.7755</td>
<td>-0.6411</td>
<td>-0.7346</td>
</tr>
<tr>
<td>-18.8020</td>
<td>2.2231</td>
<td>-0.6475</td>
<td>-0.0632</td>
<td>0.0815</td>
<td>1.2501</td>
<td>1.2530</td>
</tr>
<tr>
<td>-0.0978</td>
<td>0.0855</td>
<td>1.2561</td>
<td>-0.2814</td>
<td>0.1064</td>
<td>1.2598</td>
<td>1.2630</td>
</tr>
<tr>
<td>-0.4611</td>
<td>0.1270</td>
<td>1.2572</td>
<td>-0.6184</td>
<td>0.1450</td>
<td>1.2475</td>
<td>1.2511</td>
</tr>
<tr>
<td>-0.8369</td>
<td>0.1699</td>
<td>1.2319</td>
<td>-14.8815</td>
<td>1.8245</td>
<td>-0.6478</td>
<td>-0.7403</td>
</tr>
<tr>
<td>-25.7033</td>
<td>3.0119</td>
<td>-0.9236</td>
<td>-25.7134</td>
<td>3.0970</td>
<td>-0.9265</td>
<td>-0.7346</td>
</tr>
</tbody>
</table>

### Inertial Measurement Unit

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Velocity</th>
<th>YawRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1461634696379742</td>
<td>9.2795</td>
<td>0.0040</td>
</tr>
</tbody>
</table>
Visualize sensor data
Visualize differences in sensor detections
Explore logged vehicle data

- **Load video data** and corresponding **mono-camera parameters**
  
  ```
  >> video = VideoReader('01_city_c2s_fcw_10s.mp4')
  >> load('FCWDemoMonoCameraSensor.mat', 'sensor')
  ```

- **Load detection sensor data** and corresponding **parameters**
  
  ```
  >> load('01_city_c2s_fcw_10s_sensor.mat', 'vision','lane','radar')
  >> load('SensorConfigurationData.mat', 'sensorParams')
  ```

- **Load lidar point cloud data**
  
  ```
  >> load('01_city_c2s_fcw_10s_Lidar.mat', 'LidarPointCloud')
  ```
Visualize in image coordinates

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

%% Extract video frame
frame = video.readFrame;

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

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

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

```matlab
%% 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
```

Plot sensor coverage area with `coverageAreaPlotter`
Visualize detections (vehicle coordinates)

%% Create detection plotter

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

%% Update detection plotter

n = round(currentTime/0.05);
numDets = vision(n).numObjects;
pos = zeros(numDets,3);
vel = zeros(numDets,3);
labels = repmat({''},numDets,1);
for k = 1:numDets
    pos(k,:) = vision(n).object(k).position;
    vel(k,:) = vision(n).object(k).velocity;
    labels{k} = num2str(...
        vision(n).object(k).classification);
end
plotDetection(detPlot,pos,vel,labels);

Plot vision detections with detectionPlotter
detectionPlotter can be used to visualize vision detector, radar detector, and lidar point cloud
%% 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;
%% 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])
Visualize lane boundaries (image coordinates)

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

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

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

Plot radar detections just like vision detections with `detectionPlotter`
Visualize lidar point cloud (vehicle coordinates)

%%% Create lidar detection plotter
lidarPlot = detectionPlotter(bep, ...
    'Marker', '.', ...
    'MarkerSize', 1.5, ...
    'MarkerEdgeColor', [0 0.7 0]); % Green

%%% Update lidar detection plotter
n = round(video.CurrentTime/0.1);
pos = ...
    LidarPointCloud(n).ptCloud.Location(:,1:2);
plotDetection(lidarPlot,pos);

Plot lidar points just like vision detections with detectionPlotter
Learn more about visualizing vehicle data by exploring examples in the Automated Driving System Toolbox

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

- Transform between vehicle and image coordinates

- Plot lidar point cloud
Some common questions from automated driving engineers

How can I visualize vehicle data?

How can I detect objects in images?

How can I fuse multiple detections?
How can I detect objects in images?
Train object detectors based on ground truth

Classification
Left
Bottom
Width
Height

Train detector

Object detector
Train object detectors based on ground truth

Design object detectors with the Computer Vision System Toolbox

<table>
<thead>
<tr>
<th>Machine Learning</th>
<th>Aggregate Channel Feature</th>
<th>trainACFObjectDetector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascade</td>
<td></td>
<td>trainCascadeObjectDetector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deep Learning</th>
<th>R-CNN (Regions with Convolutional Neural Networks)</th>
<th>trainRCNNOBJECTDETECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast R-CNN</td>
<td></td>
<td>trainFastRCNNOBJECTDETECTOR</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td></td>
<td>trainFasterRCNNOBJECTDETECTOR</td>
</tr>
</tbody>
</table>
How can I create ground truth?
Specify ground truth to train detector
Specify ground truth to **train** detectors

- Video Set 1
  - Ground Truth Labeler App
  - Ground Truth
  - Train detector
  - Object detector

Specify ground truth to **evaluate** detectors

- Video Set 2
  - Object detector
  - Detections
  - Evaluate detections
  - Ground truth
  - Ground Truth Labeler App
Manually label ground truth objects with Ground Truth Labeling App

Label ground truth with Ground Truth Labeler App
Automate labeling between manually labeled frames with temporal interpolator
Automate labeling based on a manually labeled frame with point tracker
Automate initial ground truth of vehicles with ACF ground truth detector

Detect initial regions with Vehicle Detector
Export labeled regions as MATLAB time table
Customize Ground Truth Labeler App
Customize Ground Truth Labeler App

Add custom image reader with `groundTruthDataSource`
Customize Ground Truth Labeler App

Add custom automation algorithm driving.automation.AutomationAlgorithm
Customize Ground Truth Labeler App

Add connection to other tools with `driving.connector.Connector`
Learn more about detecting objects in images by exploring examples in the Automated Driving System Toolbox

- Label detections with Ground Truth Labeler App
- Add automation algorithm for lane tracking
- Extend connectivity of Ground Truth Labeler App
Learn more about detecting objects in images
by exploring examples in the Automated Driving System Toolbox

- Train object detector using deep learning and machine learning techniques
- Explore pre-trained pedestrian detector
- Explore lane detector using coordinate transforms for mono-camera sensor model
Some common questions from automated driving engineers

How can I detect objects in images?

How can I fuse multiple detections?

How can I visualize vehicle data?
Example of radar and vision detections of a vehicle
Example of radar and vision detections of a vehicle

Radar detection
Example of radar and vision detections of a vehicle

Can we fuse detections to better track the vehicle?
Fuse detections with multi-object tracker

Vision and radar detections to be fused
Integrate tracker into higher level algorithm
Generate C code for algorithm

MATLAB function
Synthesize scenario to test tracker
Test tracker against synthesized data

All detections fused into a single track
Track multiple object detections

**Multi-Object Tracker**

- Assigns detections to tracks
- Creates new tracks
- Updates existing tracks
- Removes old tracks

- Predicts and updates state of track
- Supports linear, extended, and unscented Kalman filters

**Time**
**Measurement**
**Measurement Noise**

**Track Manager**

**Tracking Filter**

**Detections** → **Tracks**

**Time**
**State**
**State Covariance**
**Track ID**
**Age**
**Is Confirmed**
**Is Coasted**
Examples of Kalman Filter (KF) initialization functions

<table>
<thead>
<tr>
<th></th>
<th>Linear KF</th>
<th>Extended KF</th>
<th>Unscented KF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(trackingKF)</td>
<td>(trackingEKF)</td>
<td>(trackingUKF)</td>
</tr>
<tr>
<td>Constant velocity</td>
<td>initcvkf</td>
<td>initcvekf</td>
<td>initcvukf</td>
</tr>
<tr>
<td>Constant acceleration</td>
<td>initcakf</td>
<td>initcaekf</td>
<td>initcaukf</td>
</tr>
<tr>
<td>Constant turn</td>
<td>Not applicable</td>
<td>initctekf</td>
<td>initctukf</td>
</tr>
</tbody>
</table>
Example of configuring multi-object tracker

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

- Radar Detections
  - Time
  - Position
  - Velocity

- Vision Detections
  - Time
  - Position
  - Velocity

Detection Packer

- Detections
  - Time
  - Measurement
  - Measurement Noise

Multi-Object Tracker

- Track Manager
- Kalman Filter

Tracks

- Time
- State
- State Covariance
- Track ID
- Age
- Is Confirmed
- Is Coasted

- Typically unique to application and sensors
- Map sensor readings into measurement matrix
- Specify measurement noise for each sensor
% Example sensor data
radar.Time = 0.049; % (sec)
radar.Position = [10 0.5]; % [x y] (m)
radar.Velocity = [4.4 0.1]; % [x y] (m/sec)
vision.Time = 0.051; % (sec)
vision.Position = [11 0.1]; % [x y] (m)
vision.Velocity = 4.1; % [x] (m/sec)

% Pack to constant acceleration measurement format:
% [positionX; velocityX; positionY; velocityY]
packedDetections(1) = objectDetection(radar.Time,...
    [radar.Position(1); radar.Velocity(1);...
    radar.Position(2); radar.Velocity(2)],...
        'MeasurementNoise', diag([1,1,2,10]));

packedDetections(2) = objectDetection(vision.Time,...
    [vision.Position(1); vision.Velocity(1);...
    vision.Position(2); 0],...
        'MeasurementNoise', diag([2,1,1,10]));
Explore demo to learn more about fusing detections

- Radar Detections
- Vision Detections
- Detection Packer
- Multi-Object Tracker
  - Track Manager
  - Kalman Filter
- Tracks

**Forward Collision Warning Using Sensor Fusion**

- Packing sensor data into object detections
- Initializing Kalman filter
- Configuring multi-object tracker
Generate C code for algorithm
with MATLAB Coder

```matlab
function [confirmedTracks, egoLane, numTracks, mostImportantObject] = ...
trackingForFCW_kernel(visionObjects, radarObjects, inertialMeasurementUnit, ...
laneReports, egoLane, time, positionSelector, velocitySelector)
```

Generate C code with `codegen`
Generate C-code for algorithm with MATLAB Coder

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

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

%% Generate code for sensor fusion algorithm: trackingForFCW_kernel
codegen trackingForFCW_kernel -args exampleInputs -config:dll -launchreport
Install patch to generate C code from multiObjectTracker

- [https://www.mathworks.com/support/bugreports/1546972](https://www.mathworks.com/support/bugreports/1546972)

```markdown
### 1546972

#### Summary

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

#### Description

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

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

#### Workaround

Installation instructions
Specify driving scenario and roads

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

%% Create road
road(s, [ 0 0; ... % Centers [x,y] (m)
    45 0], ... % Width (m)
road(s, [35 20; ...
    35 -10], ... % Width (m)
road(s, [ ...

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

Specify road centers and width as part of a drivingScenario
%% Add ego vehicle

egoCar = vehicle(s);

waypoints = [ 2  -1.25; ... % [x y] (m)
              28  -1.25; ...
              30  -1.25; ...
             36.25  4; ...
             36.25  6; ...
             36.25  14];

speed = 13.89; % (m/s) = 50 km/hr

path(egoCar, waypoints, speed);

Specify ego vehicle path using waypoints and speeds
%% Add ego vehicle

type = 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(type, waypoints, speed);

%% Play scenario

while advance(s)
    pause(s.SampleTime);
end

Specify ego vehicle path using waypoints and speeds
%% 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

Specify target vehicle with varying speed

Specify pedestrian actor size and path
%% Add chase view (left)
p2 = uipanel('Position',[0 0 0.5 1]);
a2 = axes('Parent',p2);
chasePlot(egoCar,...
    'Parent',a2,...
    'Centerline','on',...
    'ViewHeight',3.5,...  \% (m)
    'ViewLocation',[-8 0]); \% [x y] (m)
View scenario from behind ego vehicle

%% Add chase view (left)
p2 = uipanel('Position',[0 0 0.5 1]);
a2 = axes('Parent',p2);
chasePlot(egoCar,...
    'Parent',a2,...
    'Centerline','on',...
    'ViewHeight',3.5,... % (m)
    'ViewLocation',[-8 0]); % [x y] (m)

%% Play scenario
restart(s)
while advance(s)
    pause(s.SampleTime);
end
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 may not be detected if they appear above the horizon line
- Large range measurement errors can be introduced for detected objects at different road elevations

Model Vision Sensor Detections product demo illustrates these effects
Model vision detection sensor

```matlab
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]), ...  % Image size
    'UpdateInterval', s.SampleTime, ...
    'BoundingBoxAccuracy', 5, ...
    'MaxRange', 150, ...
    'ActorProfiles', actorProfiles(s));
```

Extrinsic mounting parameters

Coverage area is determined based on `cameraIntrinsics`

Model radar detection sensor using `radarDetectionGenerator`
Create birds eye plot to view sensor detections

```matlab
%% Add sensor birds eye plot (top left)
P3 = uipanel('Position', [0 0.5 0.5 0.5]);
a3 = axes('Parent', p3);
bep = birdsEyePlot('Parent', a3, ...  
    'Xlimits', [0 20], ...  
    'Ylimits', [-10 10]);
legend(a3, 'off');

% Create plotters
covPlot = coverageAreaPlotter(bep, ...  
    'FaceColor', 'blue', ...  
    'EdgeColor', 'blue');
plotCoverageArea(covPlot, ...  
    sensor.SensorLocation, sensor.MaxRange, ...  
    sensor.Yaw, sensor.FieldOfView(1));
detPlot = detectionPlotter(bep, ...  
    'MarkerEdgeColor', 'blue', ...  
    'Marker', '^');
truthPlot = outlinePlotter(bep);
```

Plot synthesized detections in `birdsEyePlot`
Play scenario with sensor models

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

- Design multi-object tracker based on logged vehicle data
- Generate C/C++ code from algorithm which includes a multi-object tracker
- Synthesize driving scenario to test multi-object tracker
The Automated Driving System Toolbox helps you...

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

**Detect objects in images**
- Train deep learning networks
- Label ground truth
- Connect to other tools

**Fuse multiple detections**
- Design multi-object tracker
- Generate C/C++
- Synthesize driving scenarios