

Automated Driving System Toolbox™

～ ADAS/自動運転 の開発・検証プラットフォーム ～

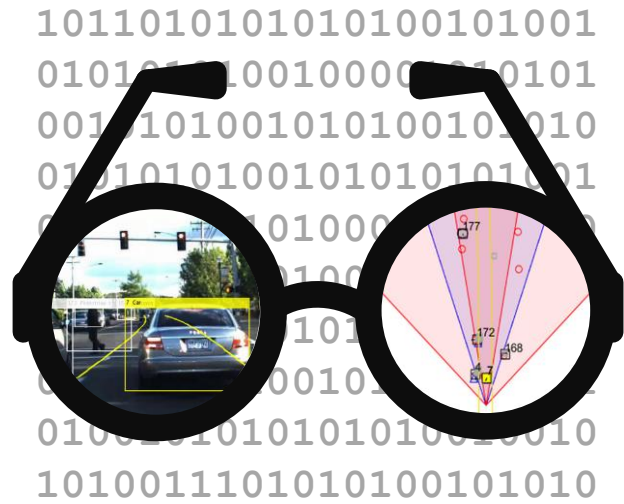
MathWorks Japan

アプリケーションエンジニアリング部

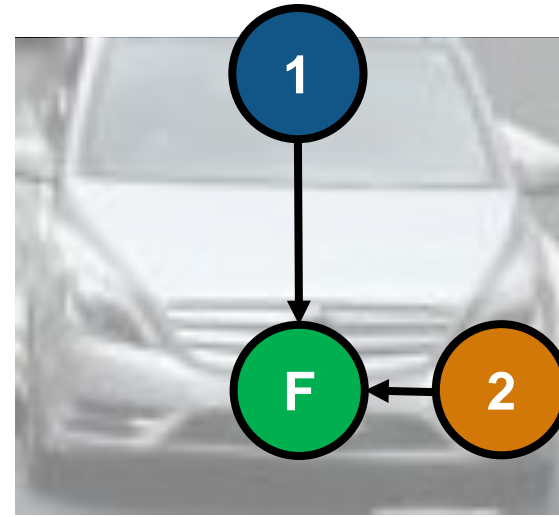
乙部 雅則

R2017a

ADAS/自動運転 開発に関して良くある悩み



センサーデータの
可視化を
どう行うか？



センサーフュージョン・
判断ロジック
をいかに開発し
検証するか？

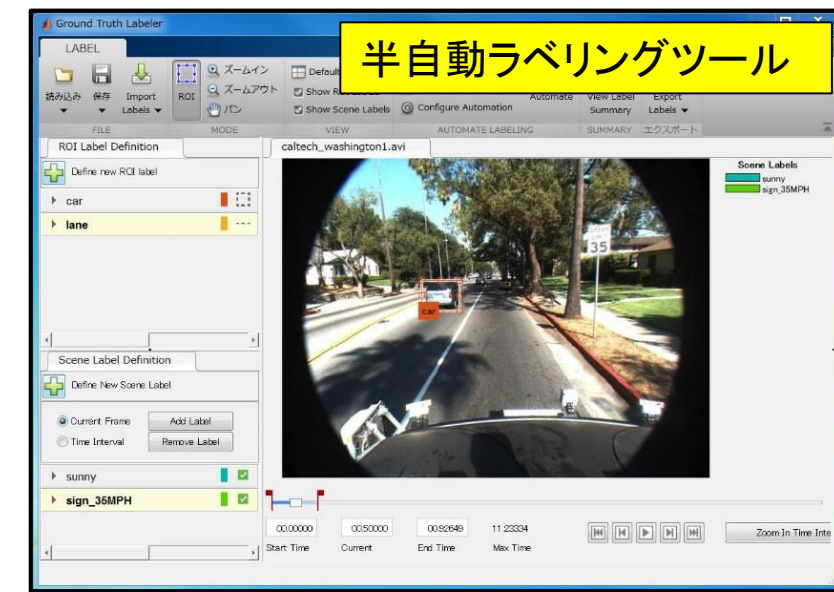
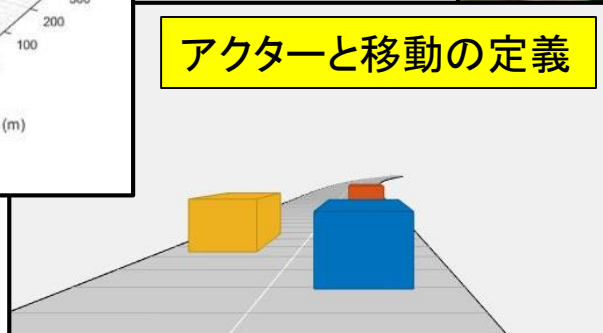
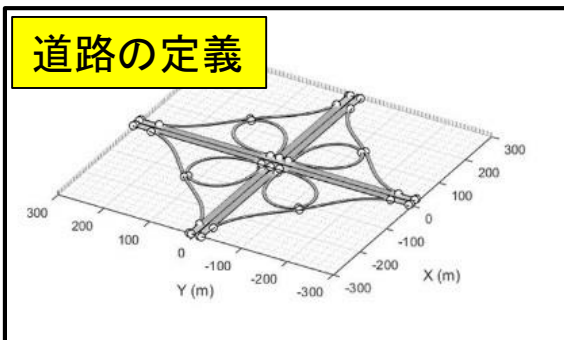
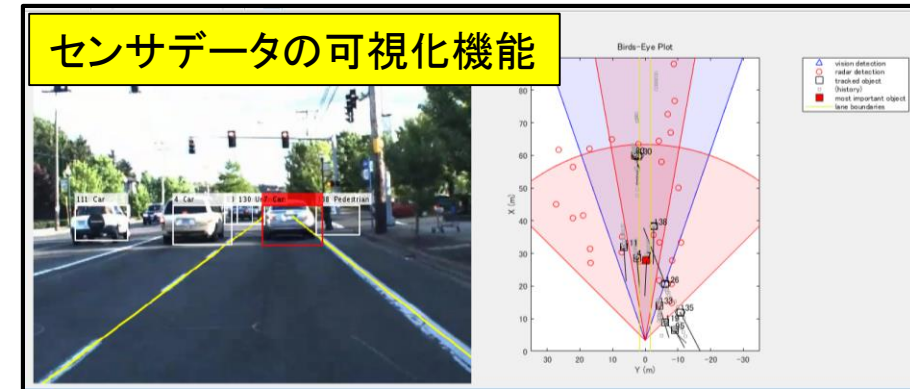


認知部分の開発・
検証をいかに
効率よく行うか？

Automated Driving System Toolbox

コンセプト: ADAS/自動運転 開発・検証環境の統合プラットフォーム

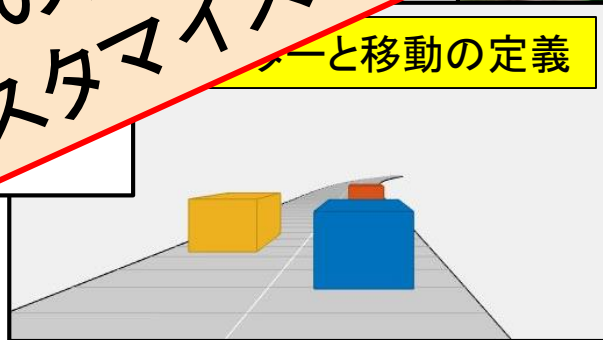
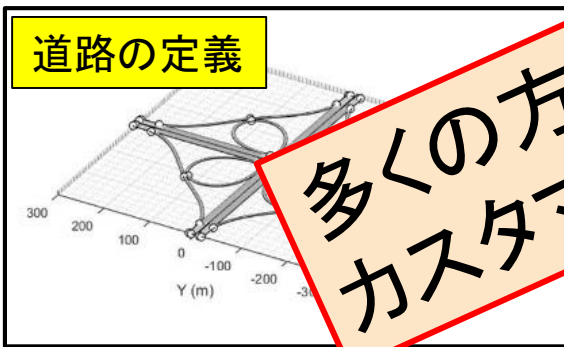
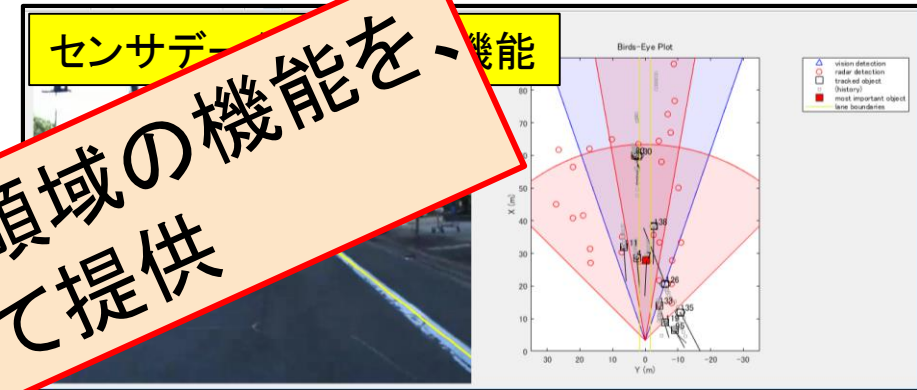
1. センサデータの可視化
2. テストシナリオ生成
3. 自動運転に関連する画像処理やトラッキング アルゴリズム
4. Ground Truth ラベリングツール



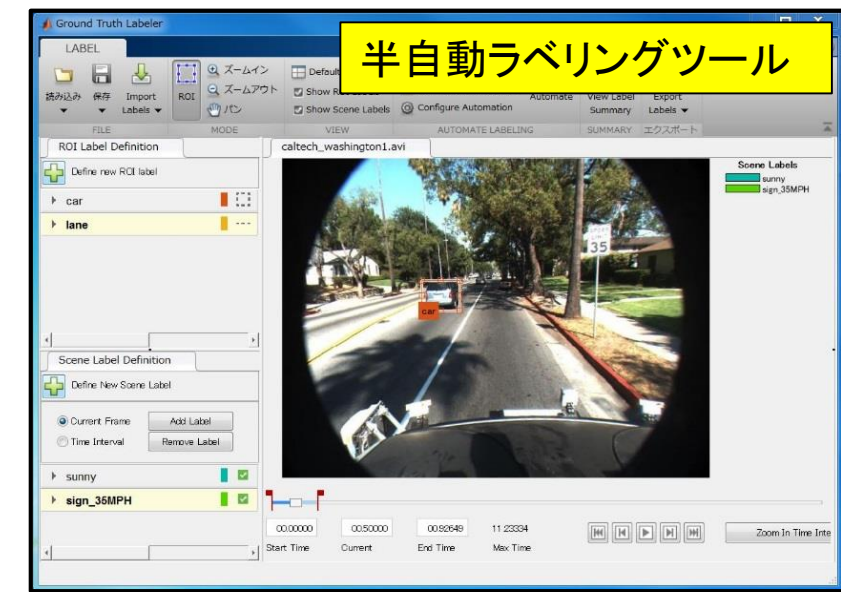
Automated Driving System Toolbox

コンセプト: ADAS/自動運転 開発・検証環境の統合プラットフォーム

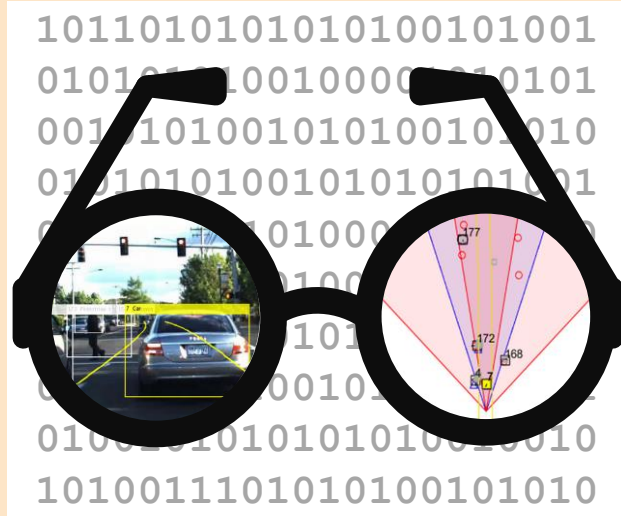
1. センサデータの可視化
2. テストシナリオ生成
3. 自動運転に関連する画像処理やトラッキング アルゴリズム
4. Ground Truth ラベリングツール



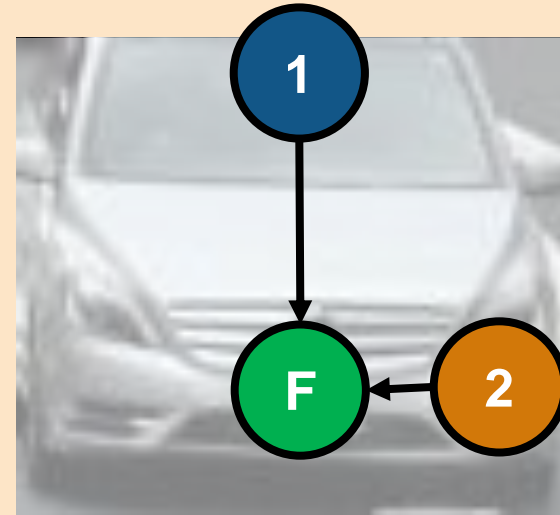
多くの方が共通して使用する非競争領域の機能を、カスタマイズ容易な部品・ツールとして提供



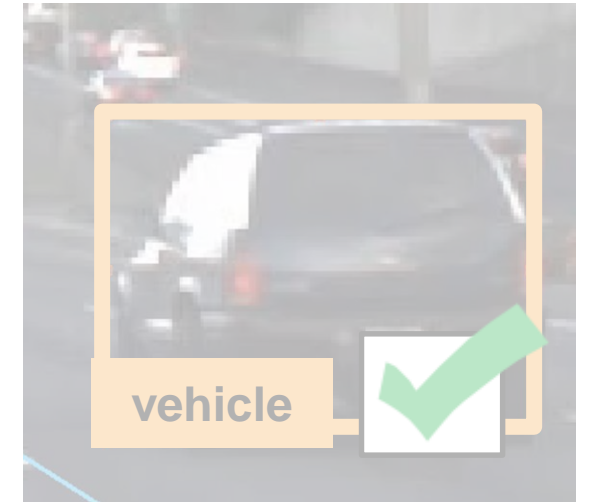
ADAS/自動運転 開発に関して良くある悩み



センサーデータの
可視化を
どう行うか？



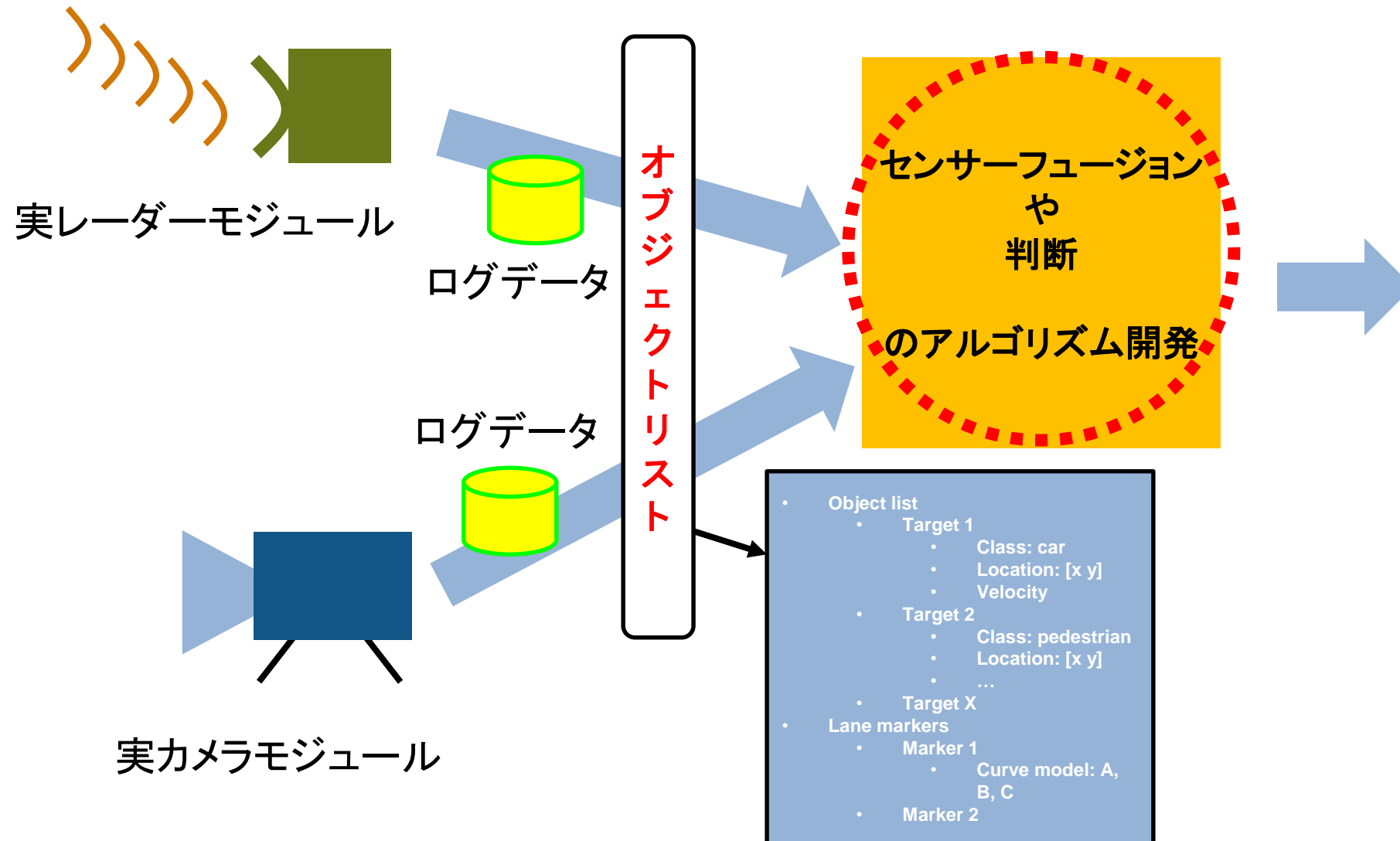
センサーフュージョン・
判断ロジック
をいかに開発し
検証するか？



認知部分の開発・
検証をいかに
効率よく行うか？

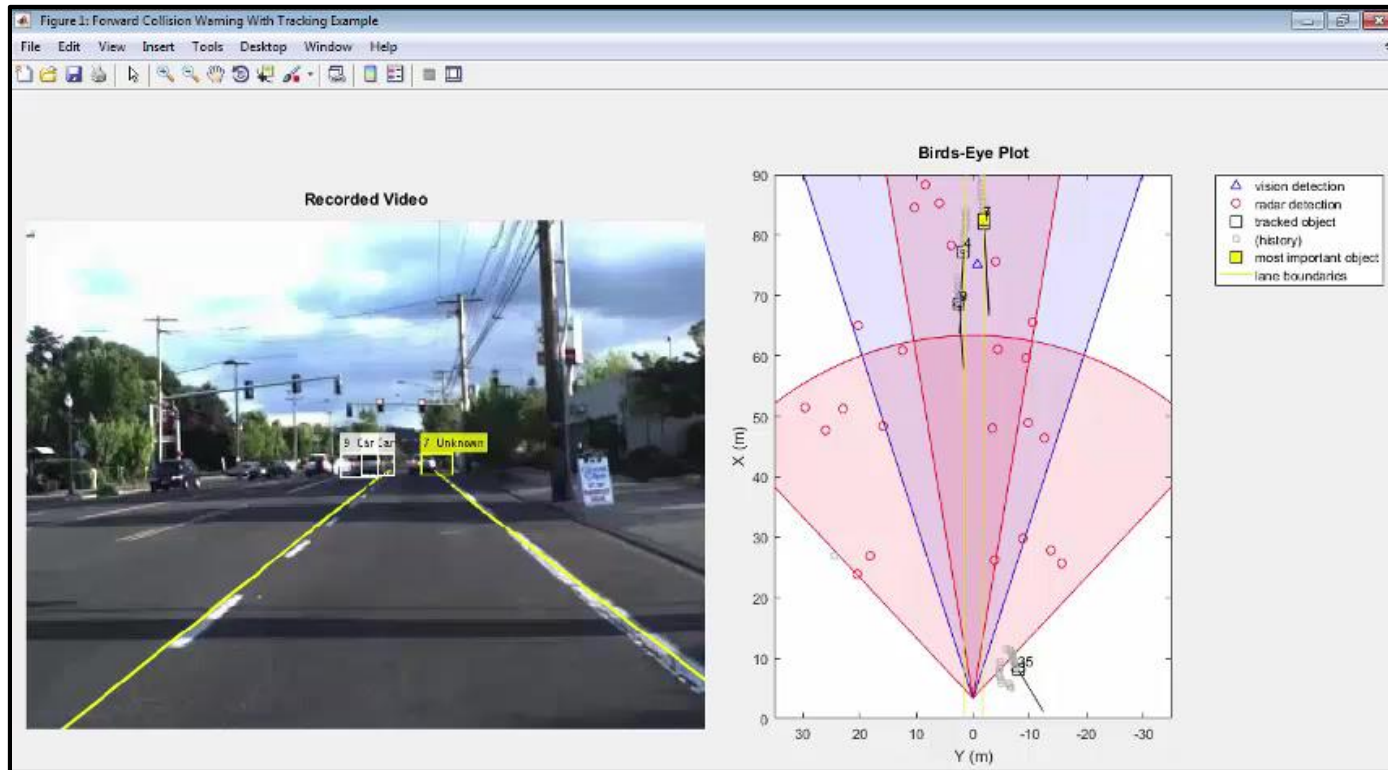
ユースケース 1 センサーフュージョン アルゴリズム 開発

～ 多様な入力データを用いた、センサーフュージョン開発環境の提供 (1)～



ユースケース 1

センサーフュージョン アルゴリズム 開発



実カメラ・レーダーモジュールの出力(オブジェクトリスト)をプロット。
ビデオへのアノテーション。センサーフュージョン。

提供機能

センサーフュージョンとトラッキングフレームワーク

- トラックの生成・消滅の管理

各種トラッキングのフィルタと運動モデル

- 線形, 拡張 & unscented カルマンフィルタ
- 速度一定, 加速度一定, 回転速度一定

可視化機能

- 検出結果表示とトラッキング向け、2次元鳥瞰図表示
- 前方撮影ビデオへの、カメラ・レーダーモジュール出力の合成

サンプルプログラム

- カメラとレーダーを用いた、前方衝突警告
- 単眼カメラによる、複数の車両トラッキング

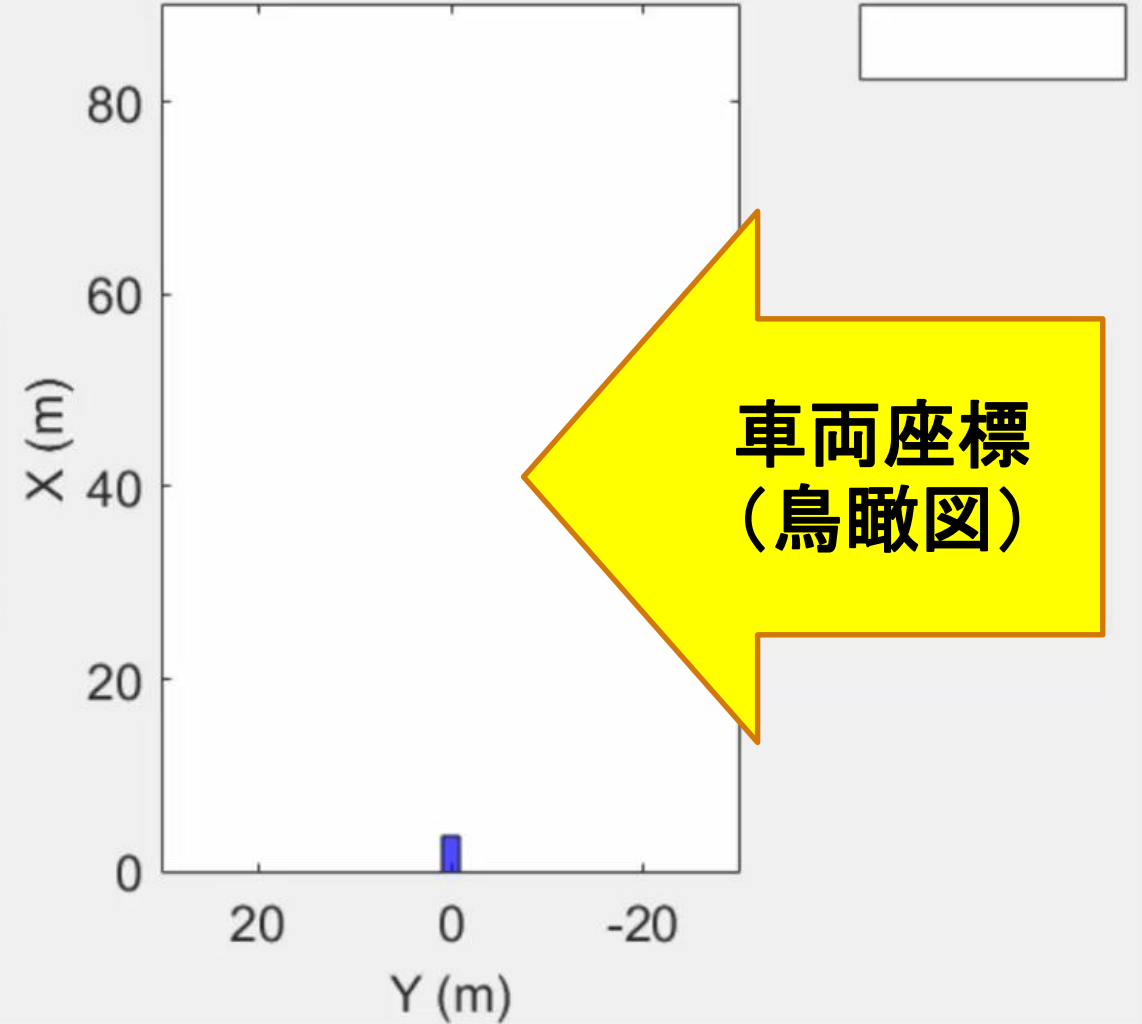
センサーデータの可視化

Image Coordinates



画像座標

Vehicle Coordinates



車両座標
(鳥瞰図)

画像座標への表示

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

%% Extract video frame
frame = video.readFrame;

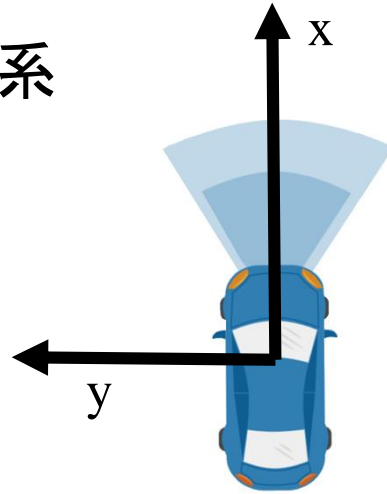
%% Plot image coordinates
ax1 = axes(...
    'Position', [0.02 0 0.55 1]);
im = imshow(frame, ...
    'Parent', ax1);
```



画像座標への表示
(各種画像処理用関数):
imshow 等

鳥瞰図への表示：車両座標

- ISO 8855 車両座標系
 - 前方：x 正方向
 - 左：y 正方向



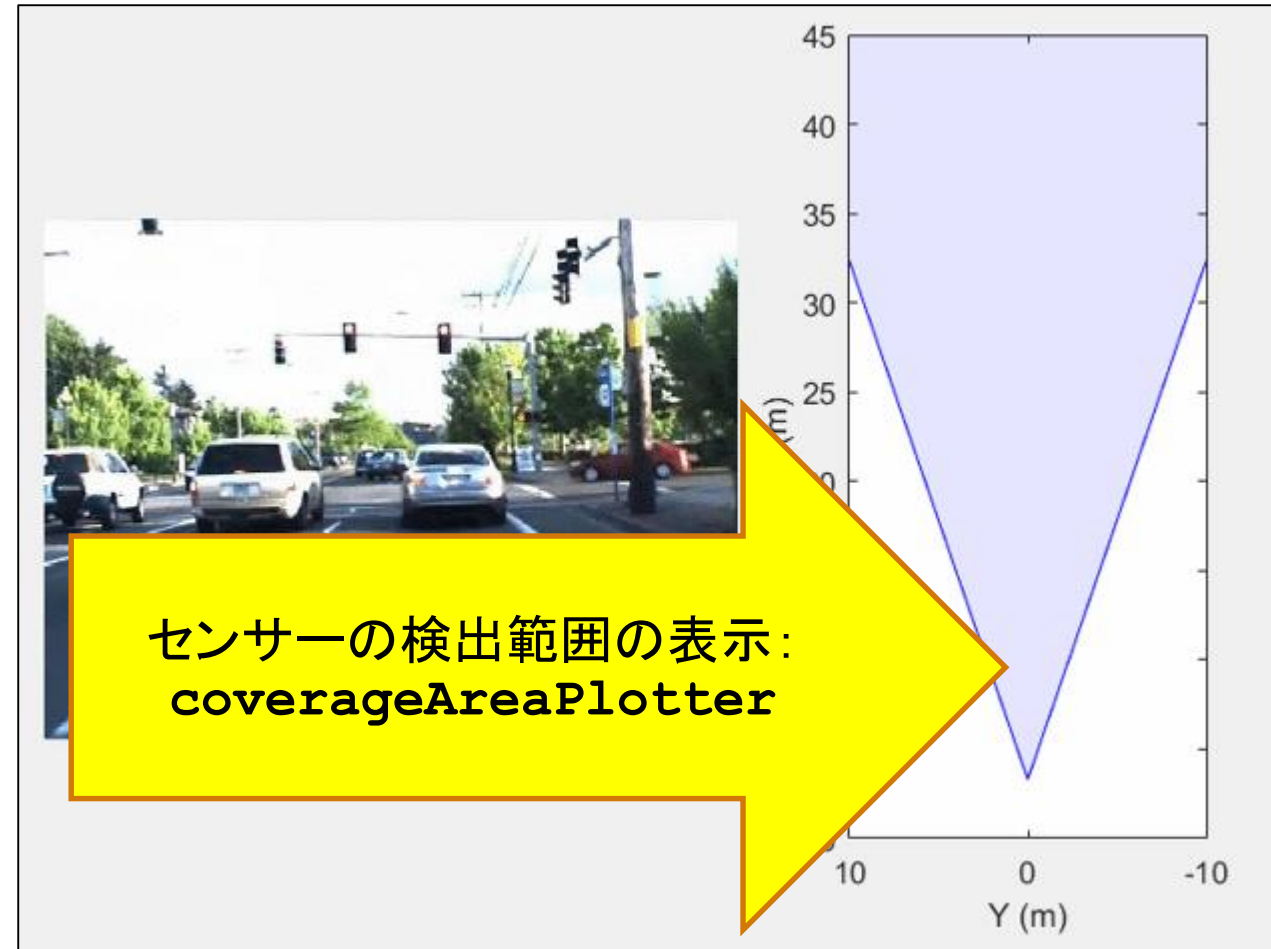
```
%% Plot in vehicle coordinates
ax2 = axes(...
    'Position',[0.6 0.12 0.4 0.85]);
bep = birdsEyePlot(...
    'Parent',ax2,...
    'Xlimits',[0 45],...
    'Ylimits',[-10 10]);
legend('off');
```



センサーの検出範囲の可視化 (鳥瞰図)

```
%% 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
```



認識結果の可視化 (車両座標)

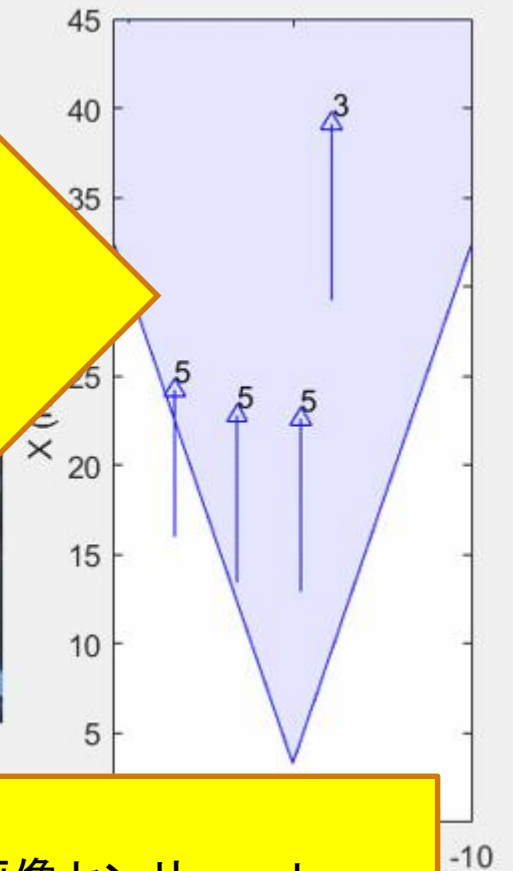
```

%% 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);

```

画像認識結果のプロット:
detectionPlotter



detectionPlotter は、画像センサー・レーダー・LiDAR等の結果のプロットに使用可

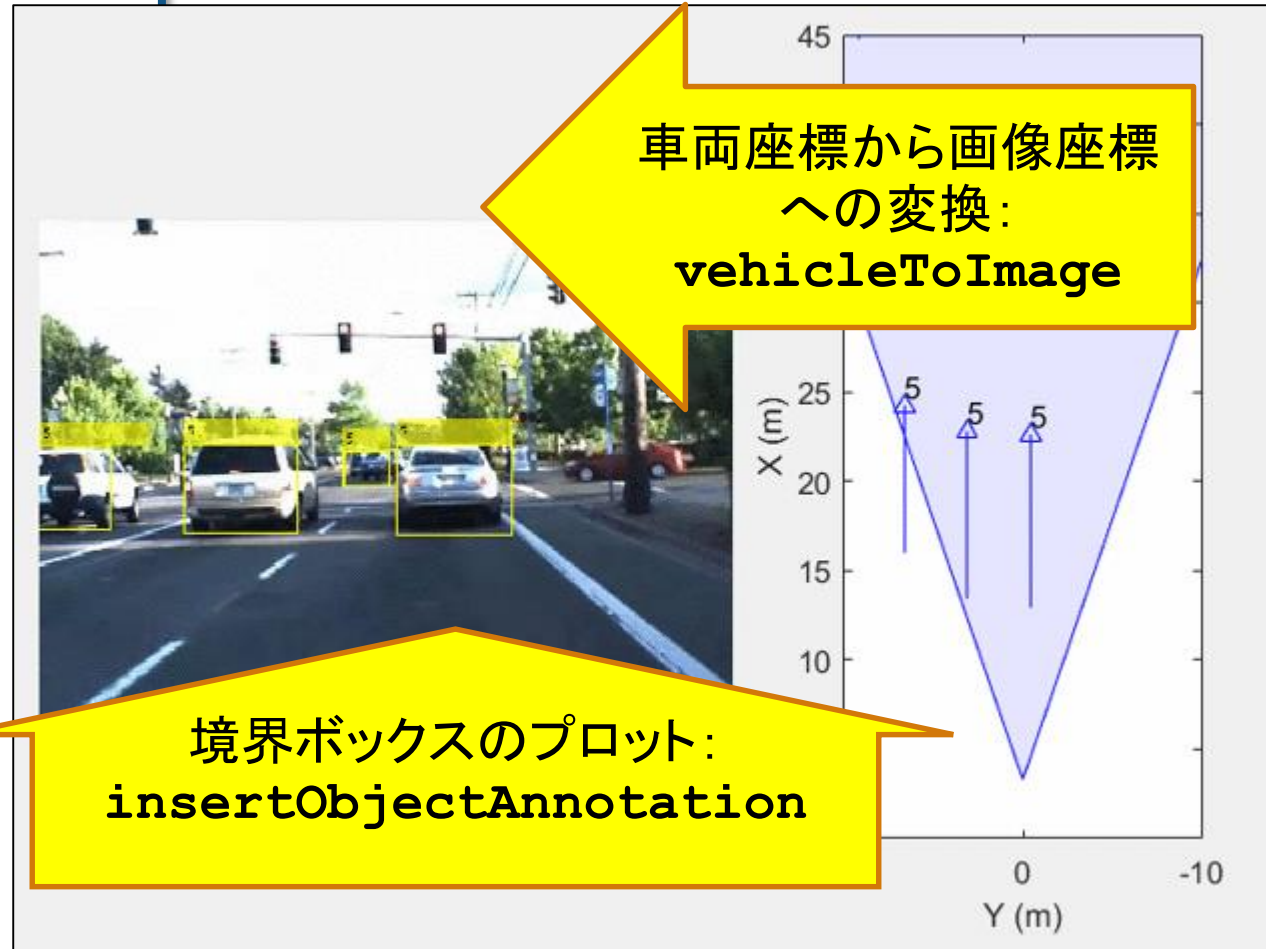
検出結果のプロット (画像座標)

```

%% 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;

```



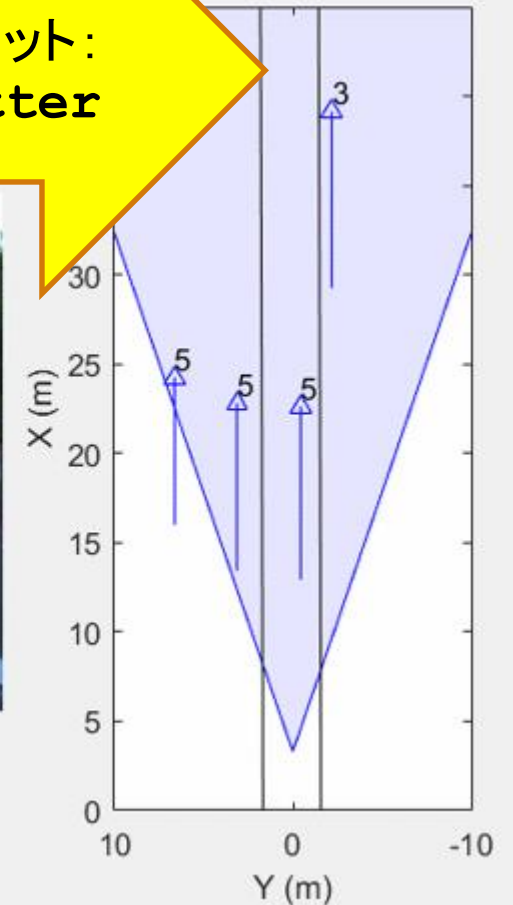
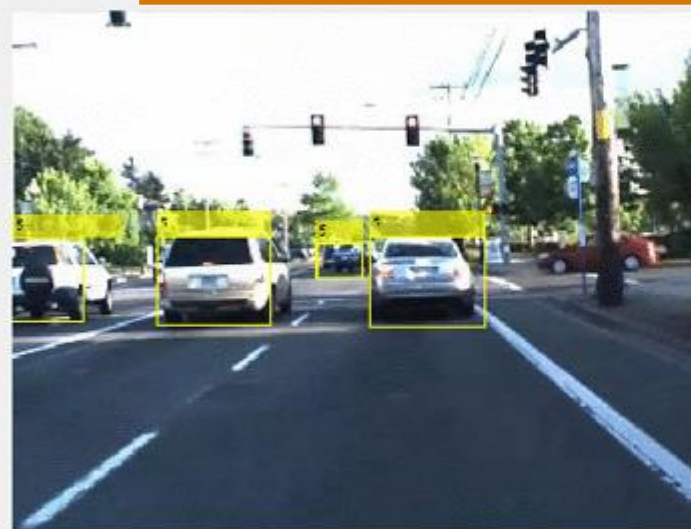
区画線の可視化 (鳥瞰図)

```

%% Create lane detection plotter
lanePlot = laneBoundaryPlotter (bep, ...
    'Color', 'black');

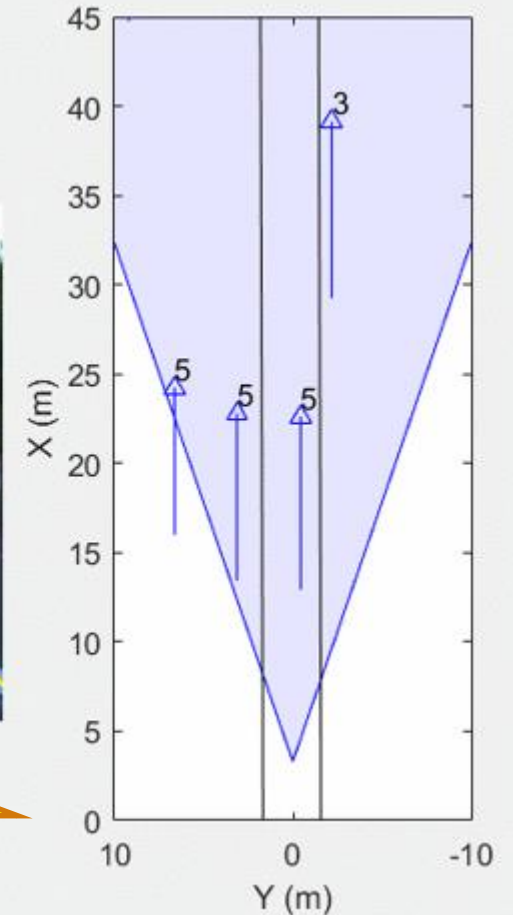
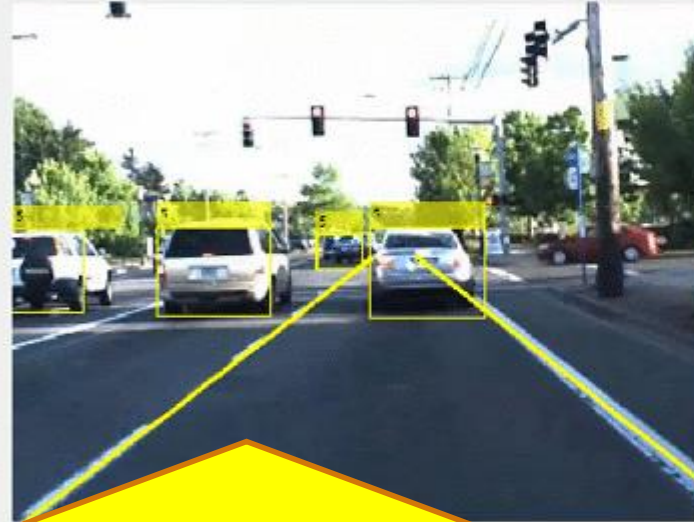
%% Update lane detection plotter
lb = parabolicLaneBoundary ([...
    lane(n).left.curvature, ...
    lane(n).left.headingAngle, ...
    lane(n).left.offset]);
rb = parabolicLaneBoundary ([...
    lane(n).right.curvature, ...
    lane(n).right.headingAngle, ...
    lane(n).right.offset]);
plotLaneBoundary (lanePlot, [lb rb])
  
```

区画線を鳥瞰図へプロット:
laneBoundaryPlotter



区画線の可視化 (画像座標)

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



区画線を画像へプロット:
`insertLaneBoundary`

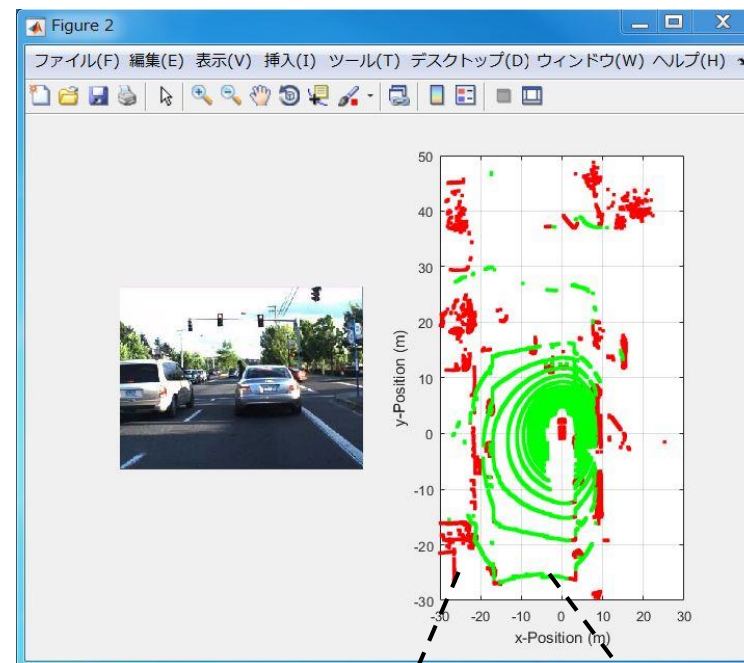
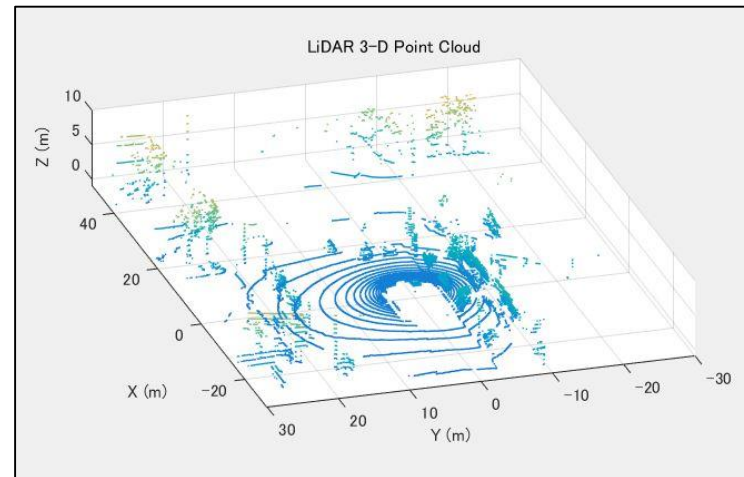
LiDAR信号処理

- LiDAR: **L**ight **D**etection **A**nd **R**anging
- レーザーによる高精度な距離測定

[3次元点群処理用の各種関数]

- 3次元表示機能
- ノイズ除去
- 点群データの間引き
- 幾何学形状(面等)へのフィッティング
- 垂線の計算
- 複数点群の位置あわせ
- 複数点群のマージ

MATLABを用いることで
データの可視化や解析・
アルゴリズム開発の効率化

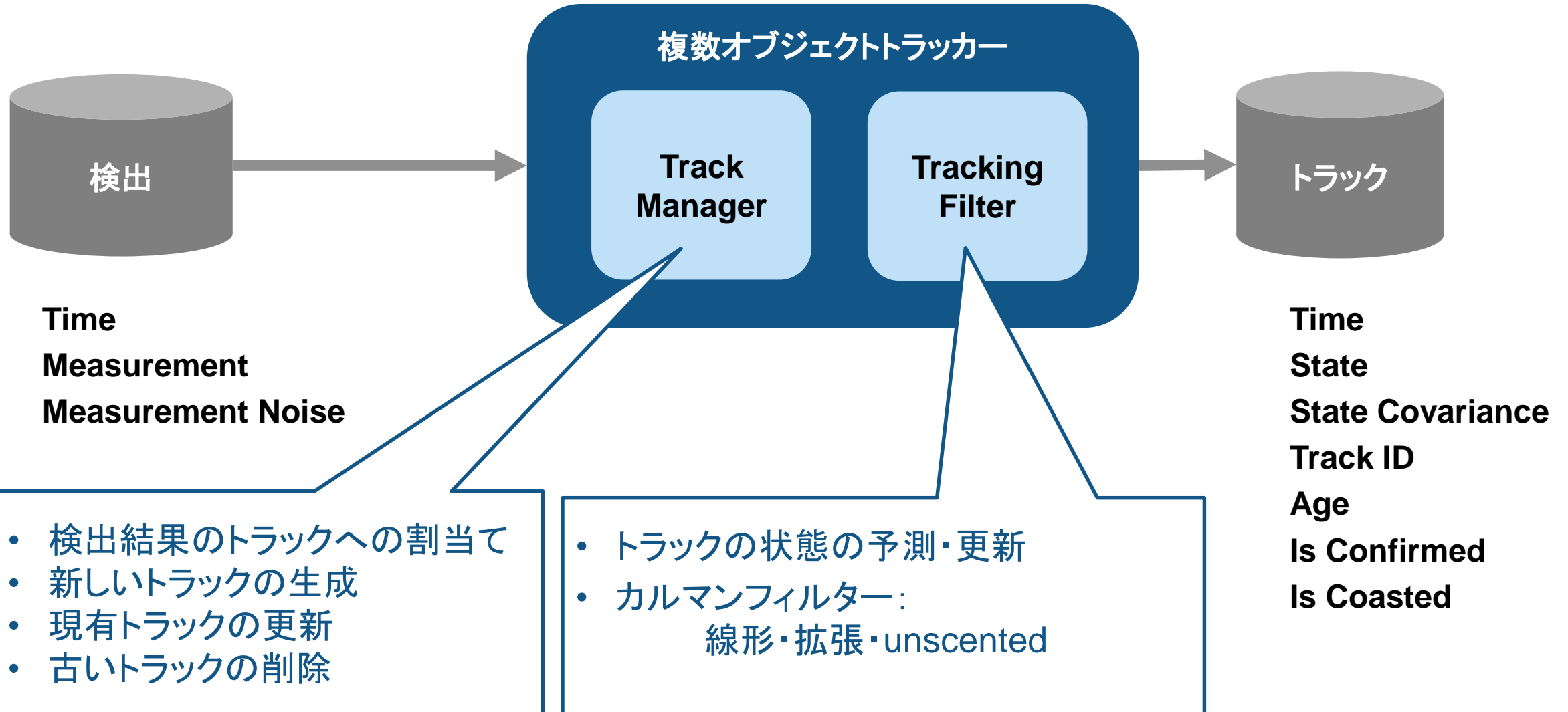


赤: 障害物

緑: 路面

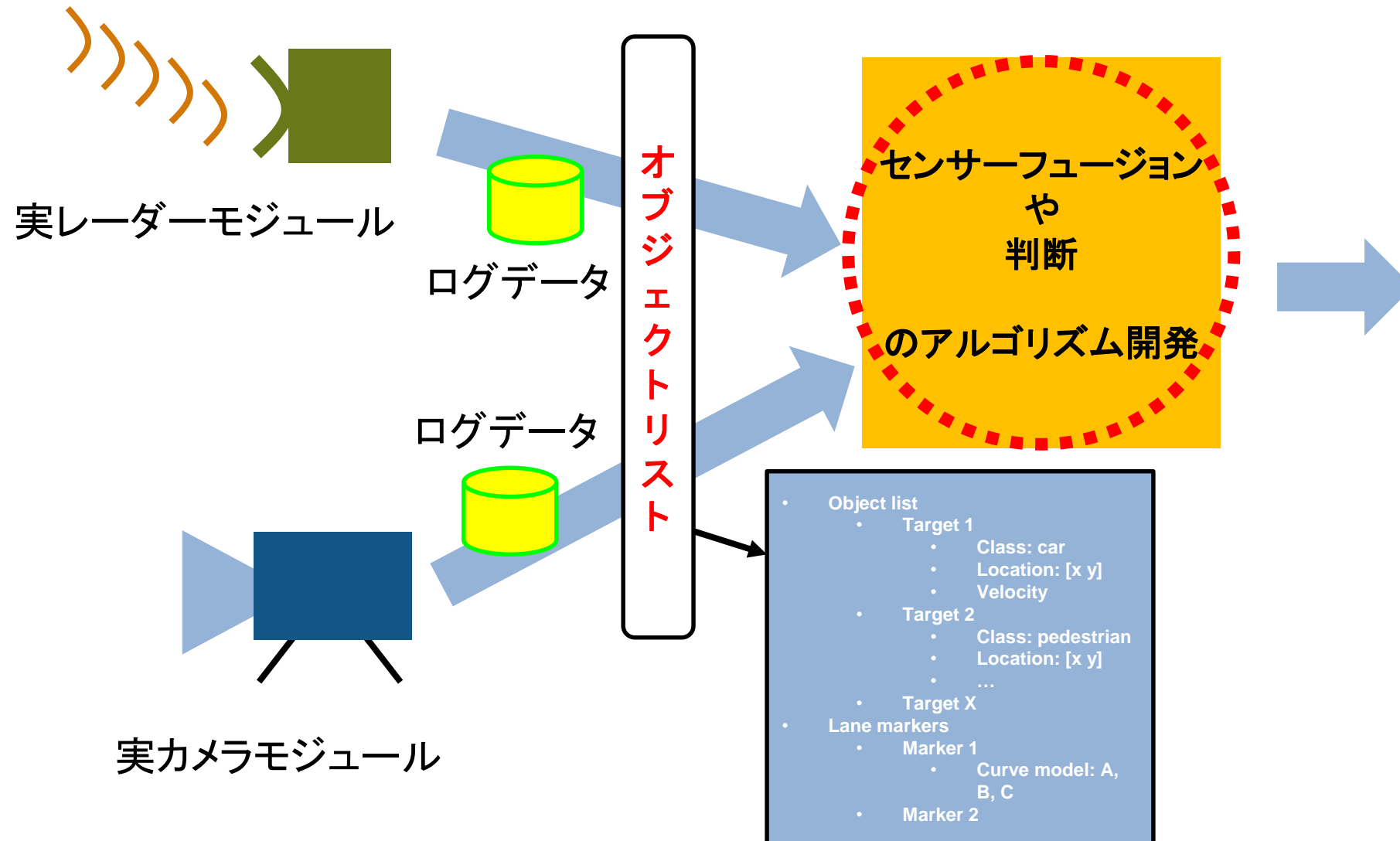
デモ

複数オブジェクトのトラッキング



ユースケース 1 センサーフュージョン アルゴリズム 開発

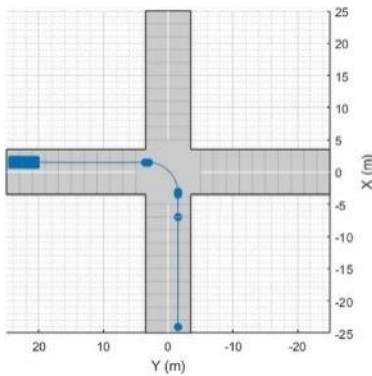
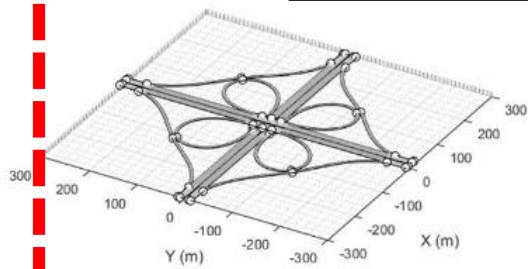
～ 多様な入力データを用いた、センサーフュージョン開発環境の提供 (1)～



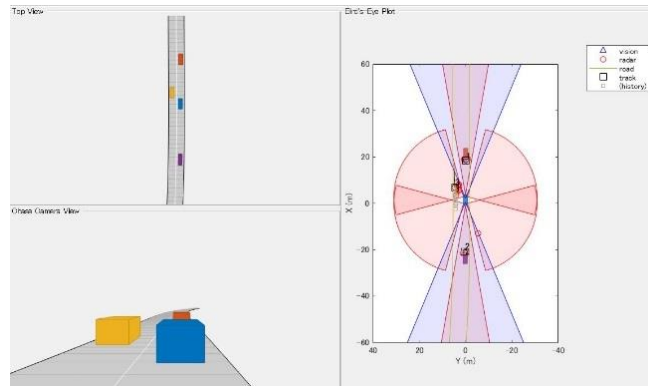
ユースケース 2 センサーフュージョン アルゴリズム 開発

～ 多様な入力データを用いた、センサーフュージョン開発環境の提供(2) ～

シナリオ生成



レーダーモデル



カメラモデル

オブジェクトリスト

- Object list
 - Target 1
 - Class: car
 - Location: [x y]
 - Velocity
 - Target 2
 - Class: pedestrian
 - Location: [x y]
 - ...
 - Target X
- Lane markers
 - Marker 1
 - Curve model: A, B, C
 - Marker 2

センサーフュージョン
や
判断
のアルゴリズム開発

デモ

道路の定義

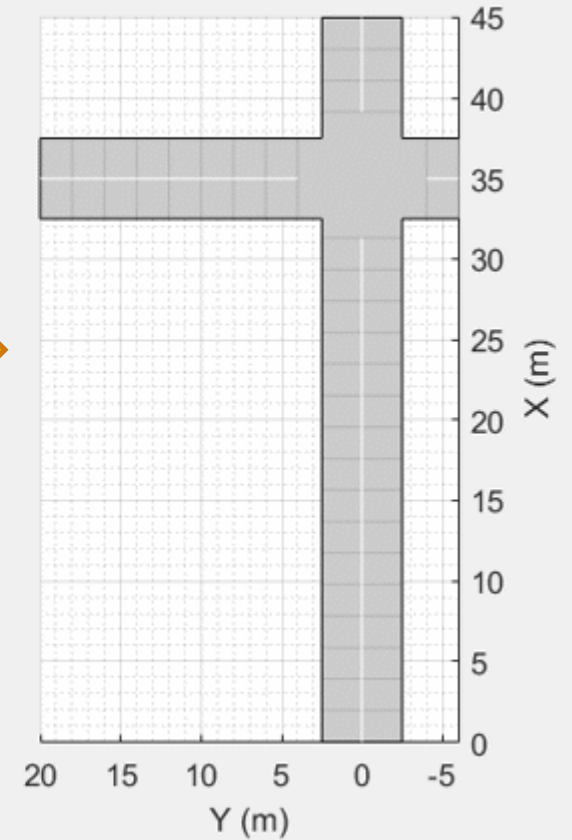
```
%% Create a new scenario
s = drivingScenario('SampleTime', 0.05);

%% 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];
```

道路端の座標・道幅を指定:
road



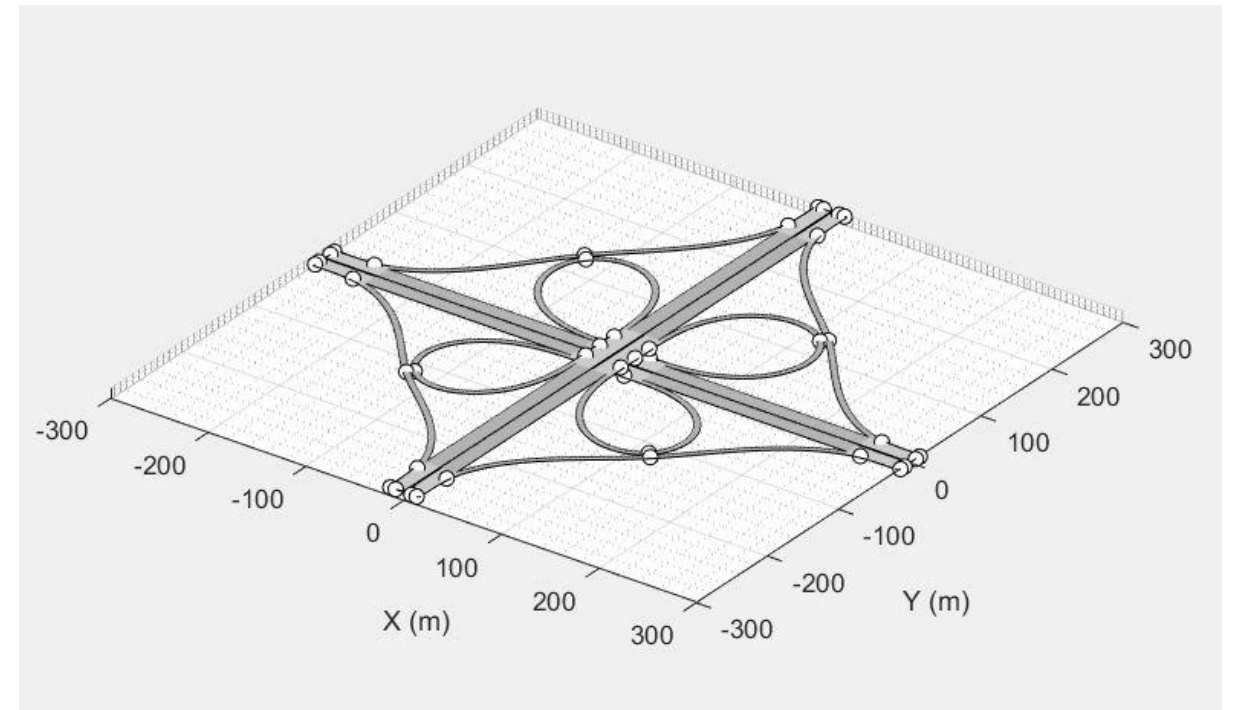
道路の定義

```
s = drivingScenario;

% Highways
road(s, [-300 -8 0; 300 -8 0], 15); % north
road(s, [-300 8 0; 300 8 0], 15); % south
road(s, [-8 -300 8; -8 300 8], 15); % east
road(s, [ 8 -300 8; 8 300 8], 15); % west

% Inner ramps
rampNE = [0 -18 0; 20 -18 0; 120 -120 4; 18 -20 8; 18 0
8];
rampNW = [ 1 -1 1] .* rampNE(end:-1:1,:);
rampSW = [-1 -1 1] .* rampNE;
rampSE = [ 1 -1 1] .* rampSW(end:-1:1,:);
innerRamps = [rampNE(1:end-1,:)
              rampNW(1:end-1,:)
              rampSW(1:end-1,:)
              rampSE];
road(s, innerRamps, 5.4);

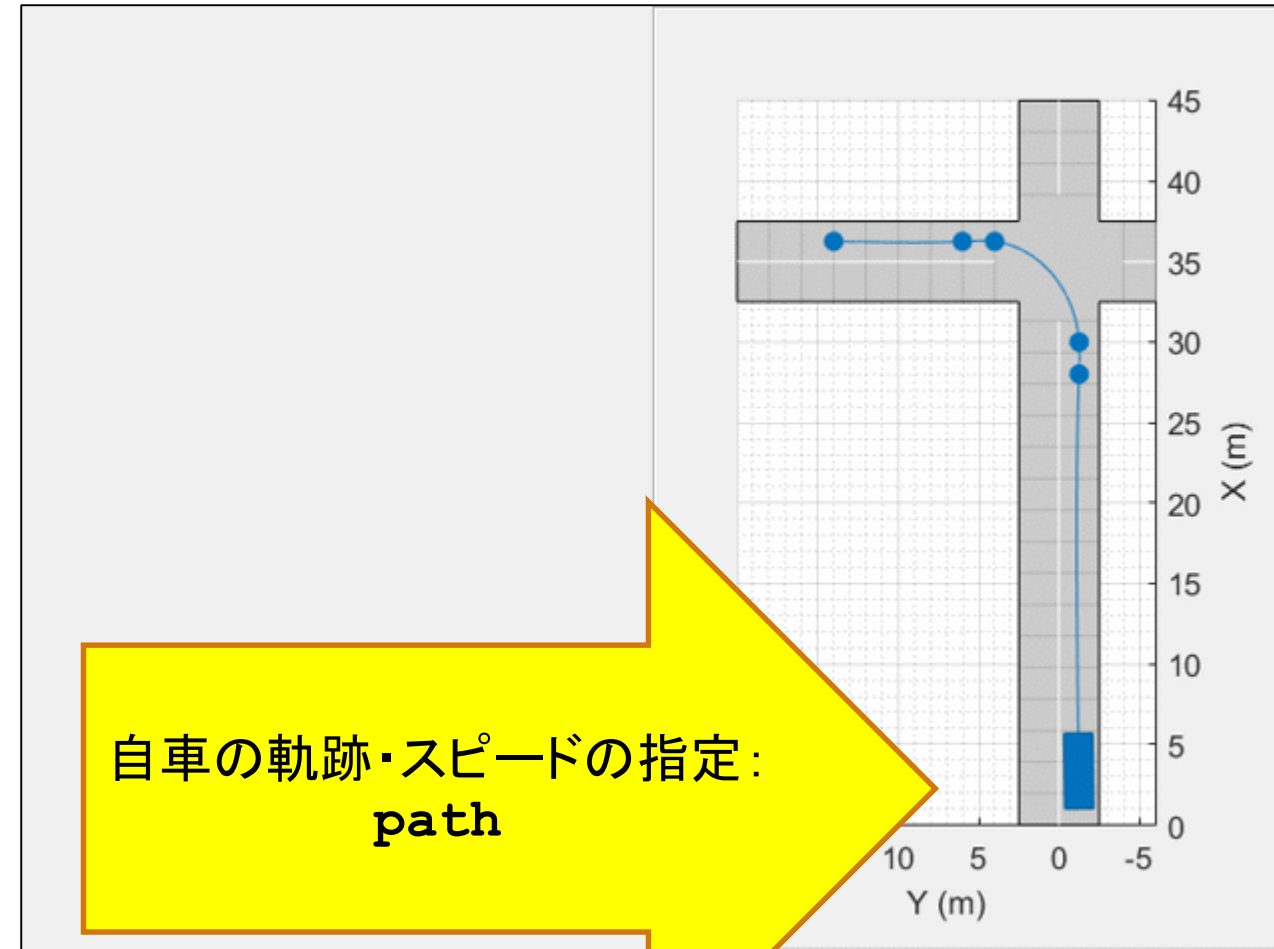
% Outer ramps
roadCenters = [13.5 -300 8; 15 -260 8; 125 -125 4; 260 -15
0; 300 -13.5 0];
road(s, [ 1 1 1] .* roadCenters, 5.4);
road(s, [ 1 -1 1] .* roadCenters, 5.4);
road(s, [-1 -1 1] .* roadCenters, 5.4);
road(s, [-1 1 1] .* roadCenters, 5.4);
```



3次元構造も可能

自車の定義

```
%% 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);
```

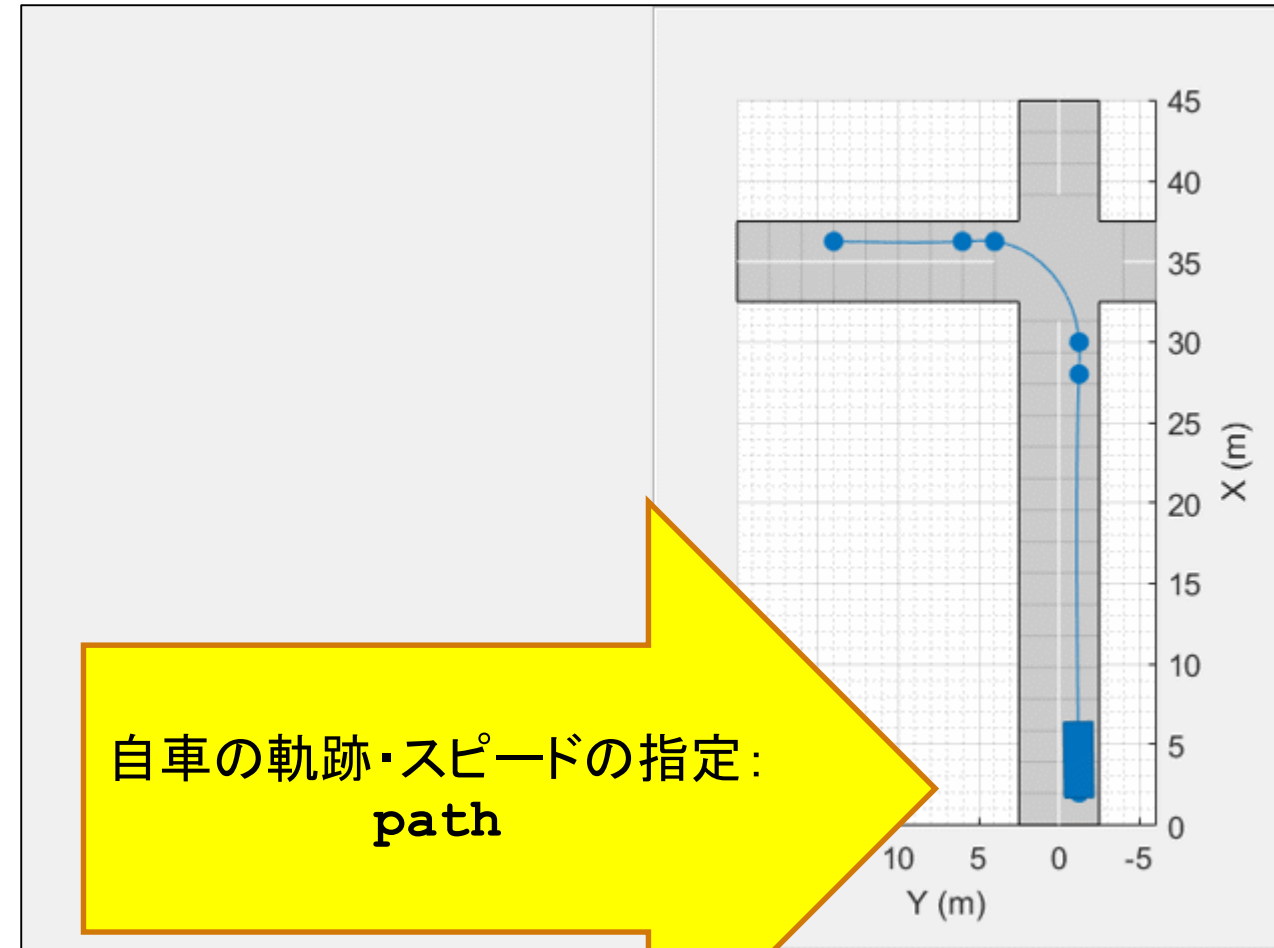


自車の定義

```
%% 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
```



対向車と歩行者の指定

```

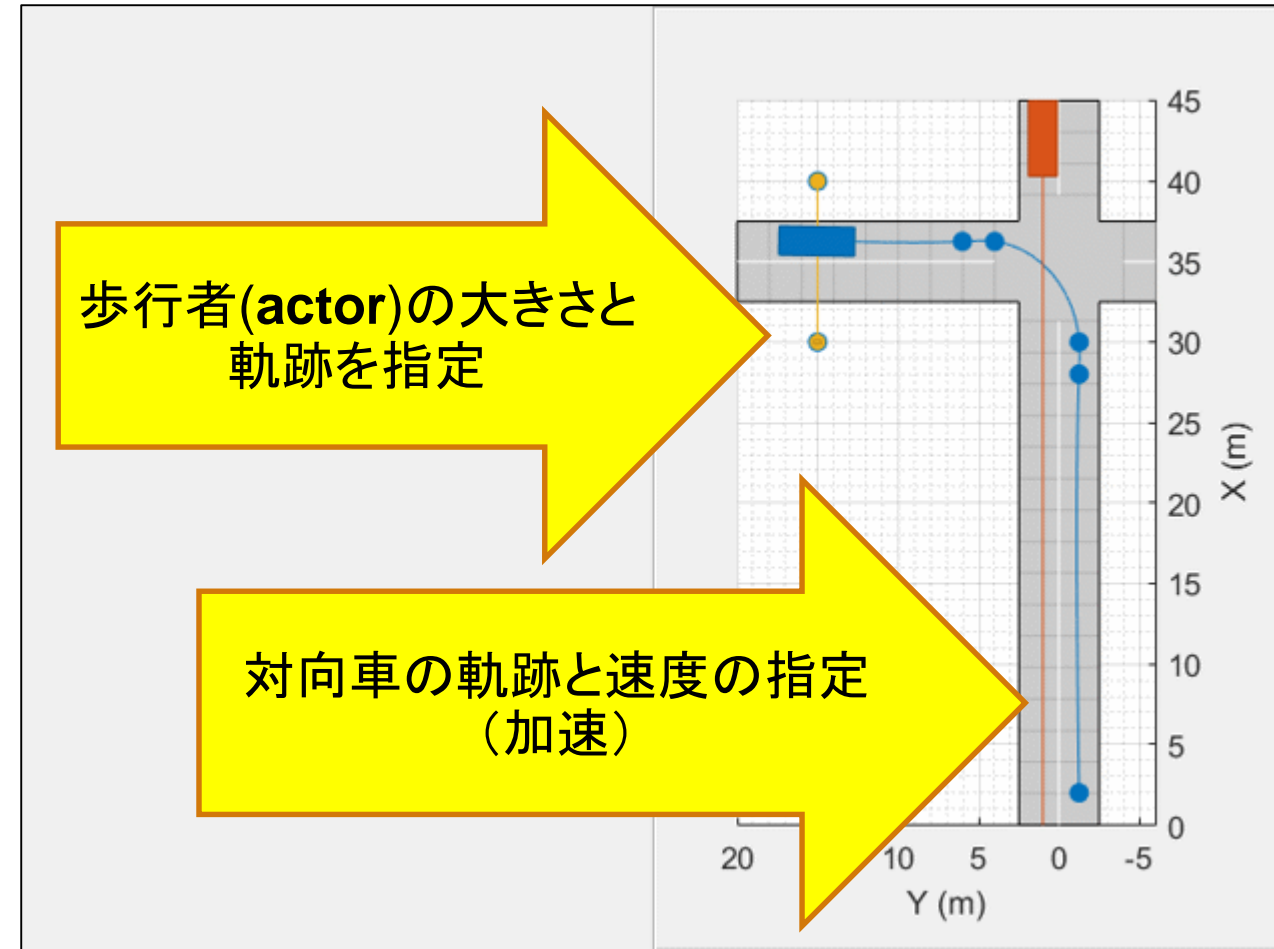
%% Add Target vehicle
targetVehicle = vehicle(s);

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

```



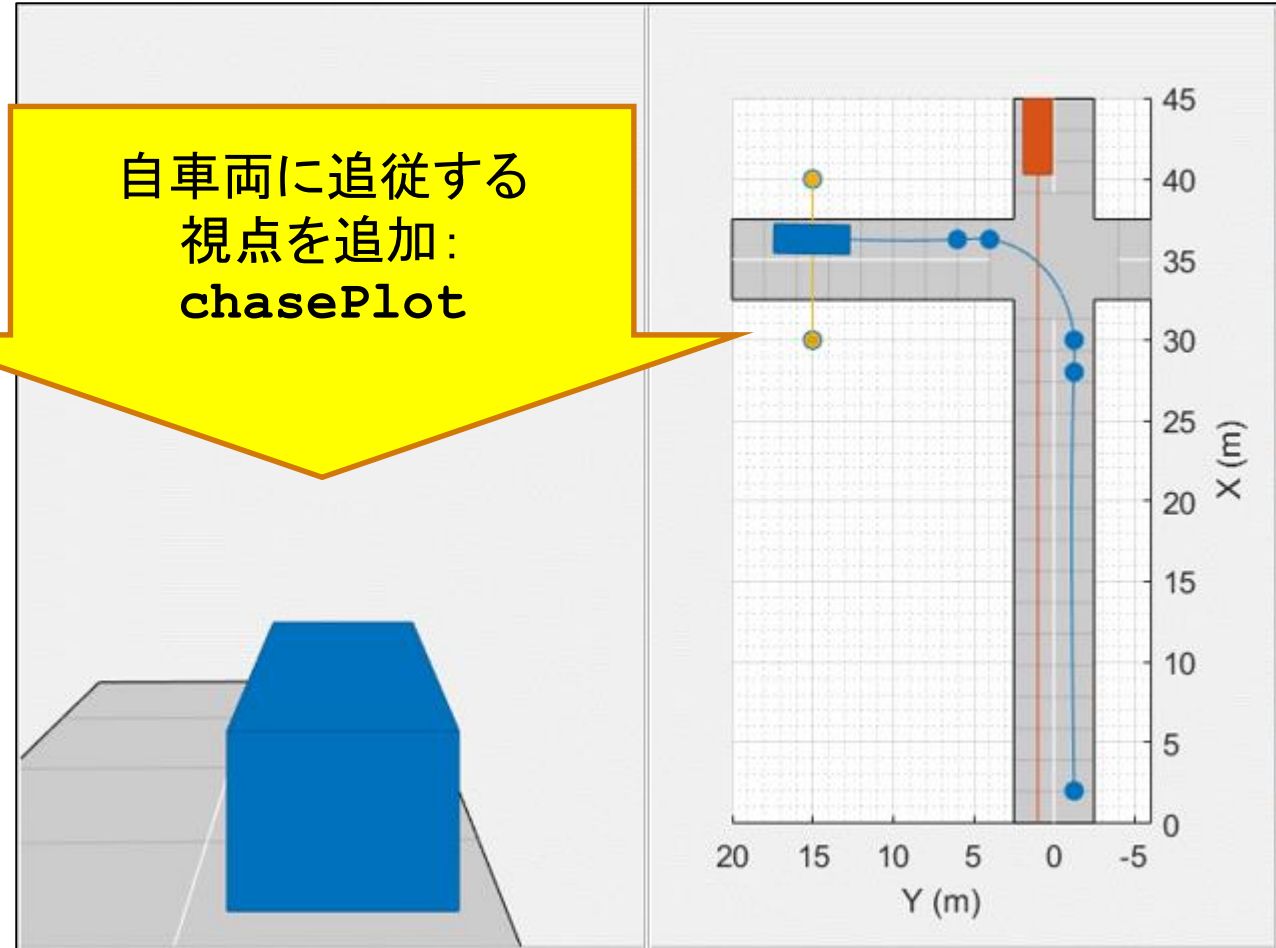
後方から車両に追従する視点の追加

```

%% 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)

```

自車両に追従する
視点を追加：
chasePlot



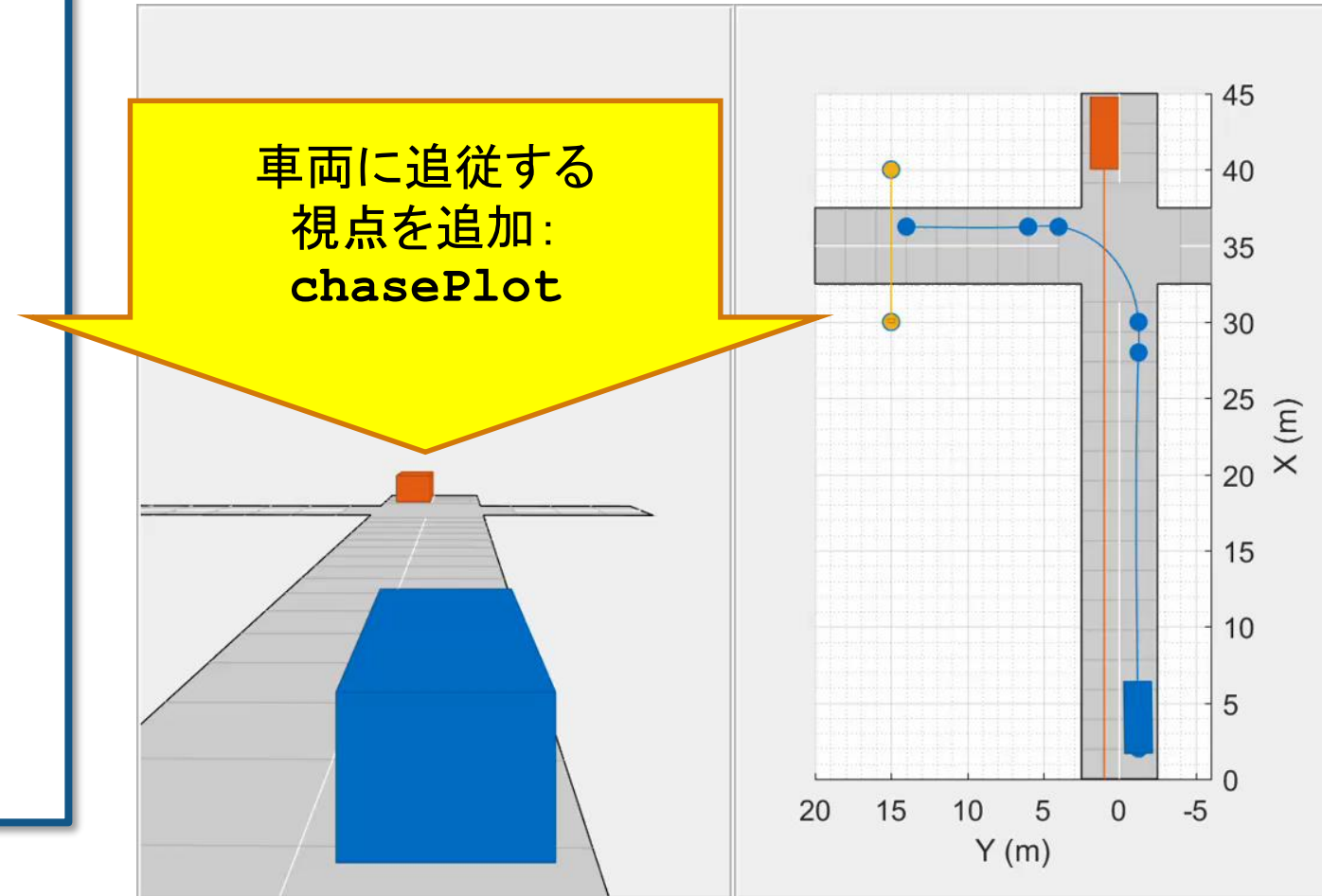
後方から車両に追従する視点の追加

```

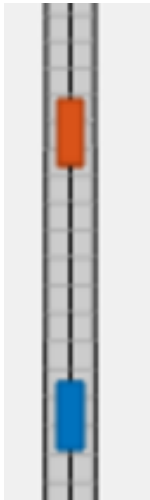
%% Add chase view (left)
p2 = uipanel('Position',[0 0 0.5 1]);
a2 = axes('Parent',p2);
chasePlot(egoCar,...
    'Parent',a2,...
    'Centerline','on',...
    'ViewHeight',3.5,... % (m)
    'ViewLocation',[-8 0]); % [x y] (m)

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

```

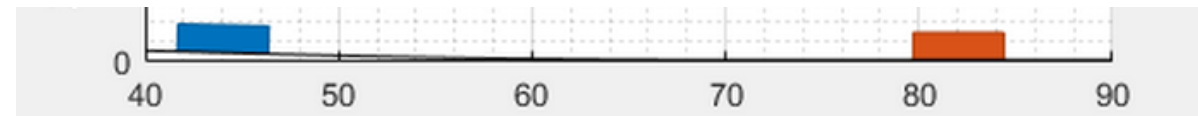


センサーの特性: カメラモジュール(画像センサー)の場合



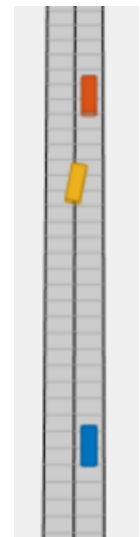
距離の効果

- 距離推定の精度は、物体までの距離が離れるに従い劣化
- 横方向の角度の精度は、検出範囲内で一定



道路勾配効果

- 水平線より高く見える場合、検出精度の劣化
- 勾配が異なることで、距離推定精度の劣化



隠れの効果

- 部分的を含む隠れに認識が弱い

センサーモデルには、パラメータで、認識可能距離・精度やノイズ含め様々な特性を持たせることができます

画像センサーの定義

```
%% 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));
```

カメラの設置パラメータ

cameraIntrinsics
により検出範囲を規定

レーダーのモデルの定義の場合：
radarDetectionGenerator

センサー出力の表示用に、鳥瞰図を生成

```

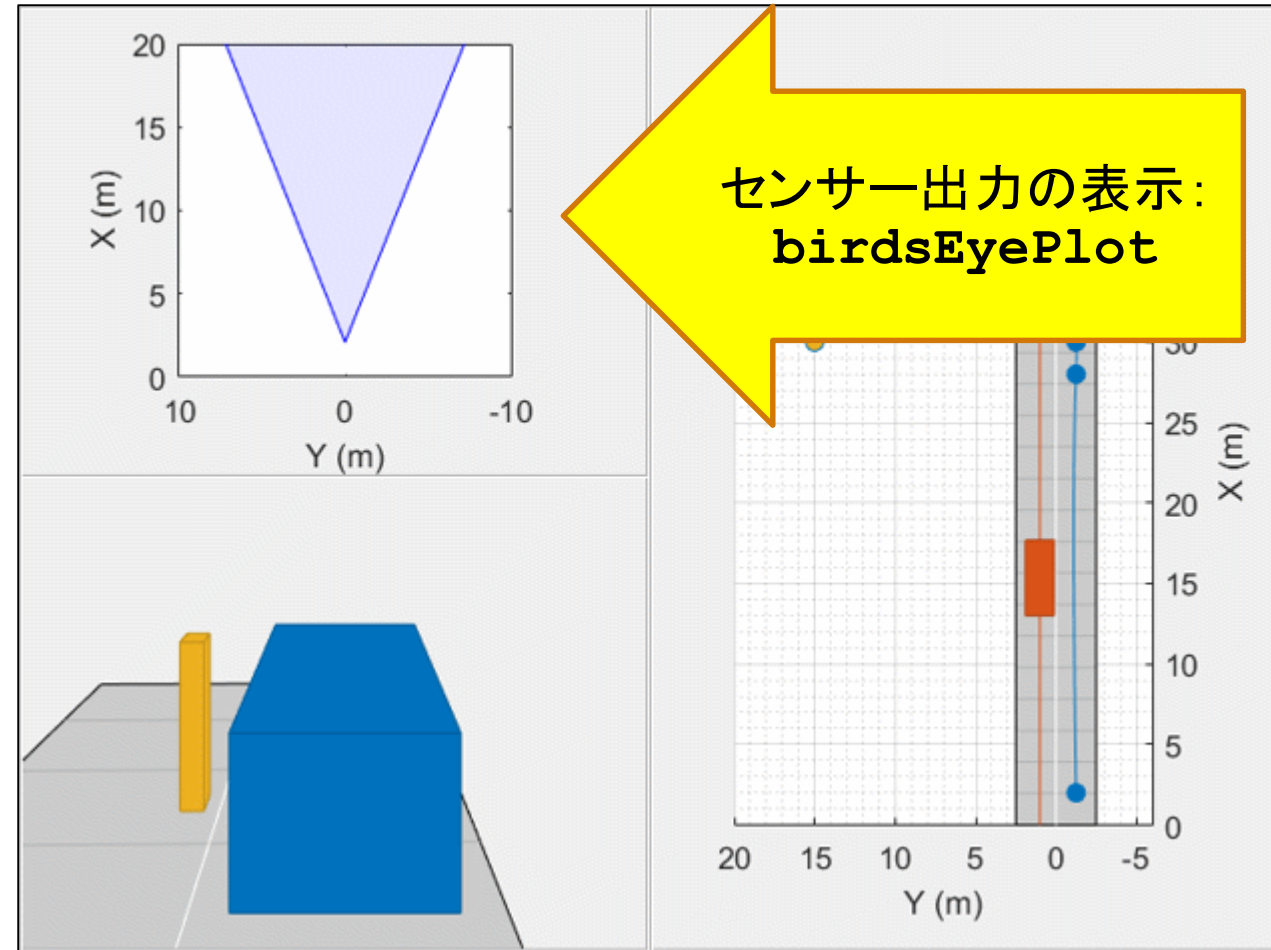
%% Add sensor birds eye plot (top left)
p3 = uipanel('Position',[0 0.5 0.5 0.5]);
a3 = axes('Parent',p3);
bep = birdsEyePlot('Parent',a3,...
    'Xlimits',[0 20],...
    'Ylimits',[-10 10]);
legend(a3,'off');

% 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);

```



センサーモデルと共にシミュレーション

```

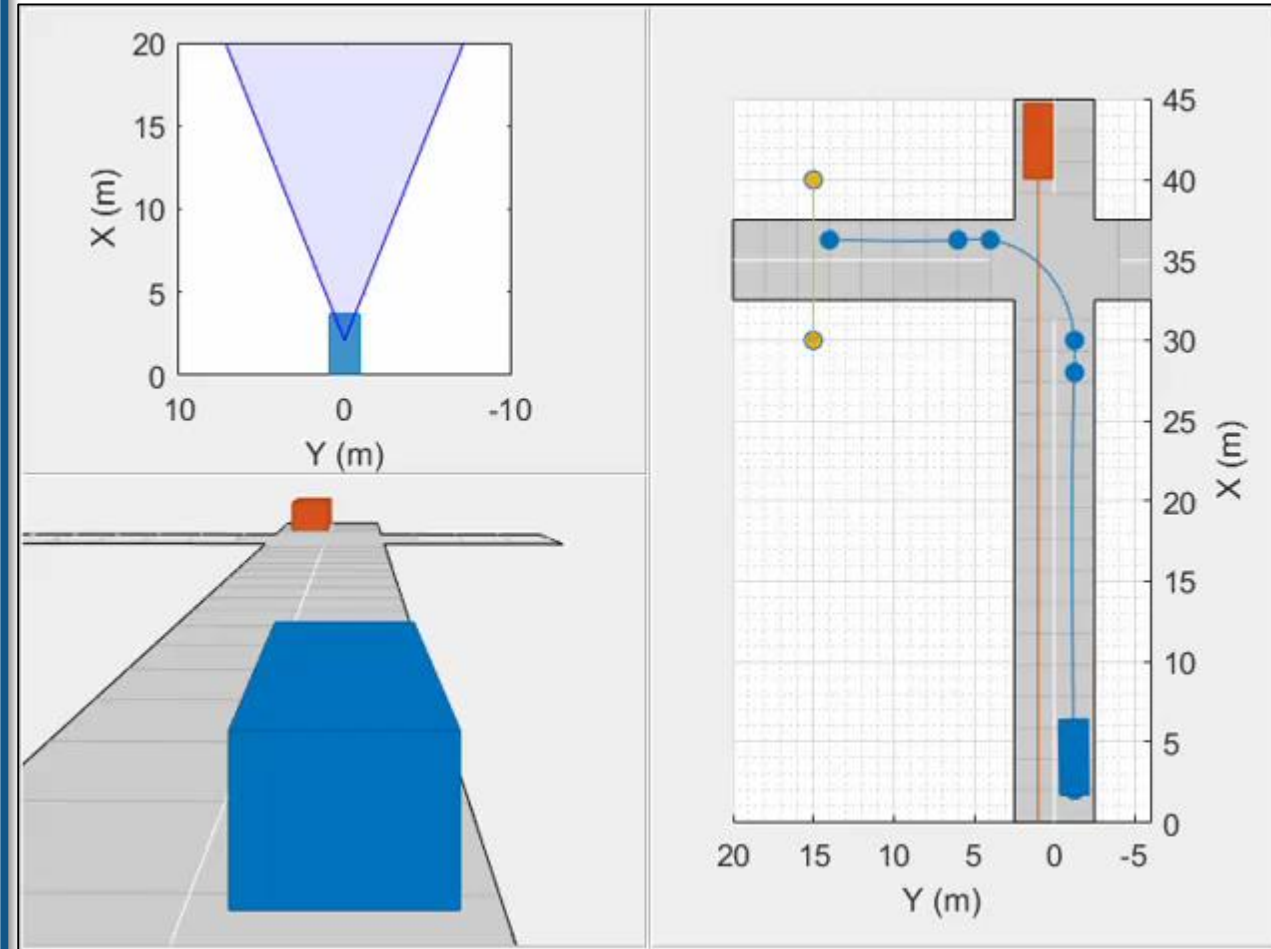
restart(s)
while advance(s)
    % Get detections in ego vehicle coordinates
    det = sensor(targetPoses(egoCar),...
                s.SimulationTime);

    % 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

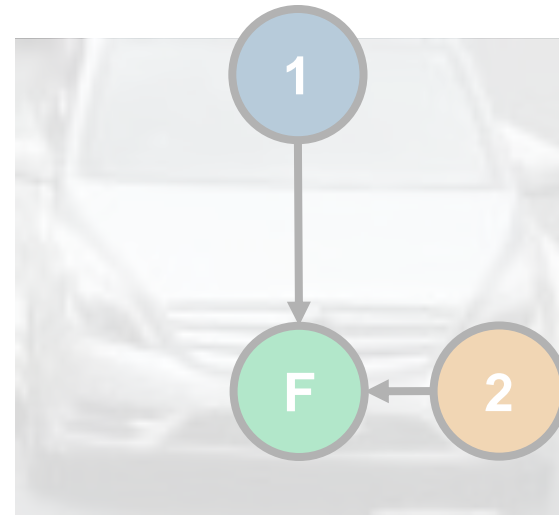
```



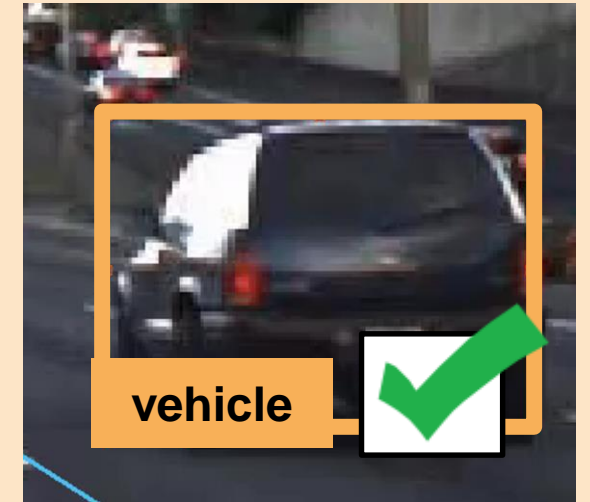
ADAS/自動運転 開発に関して良くある悩み



センサーデータの
可視化を
どう行うか？



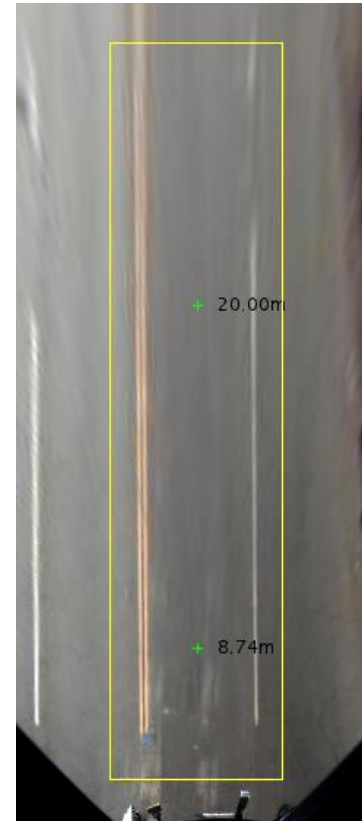
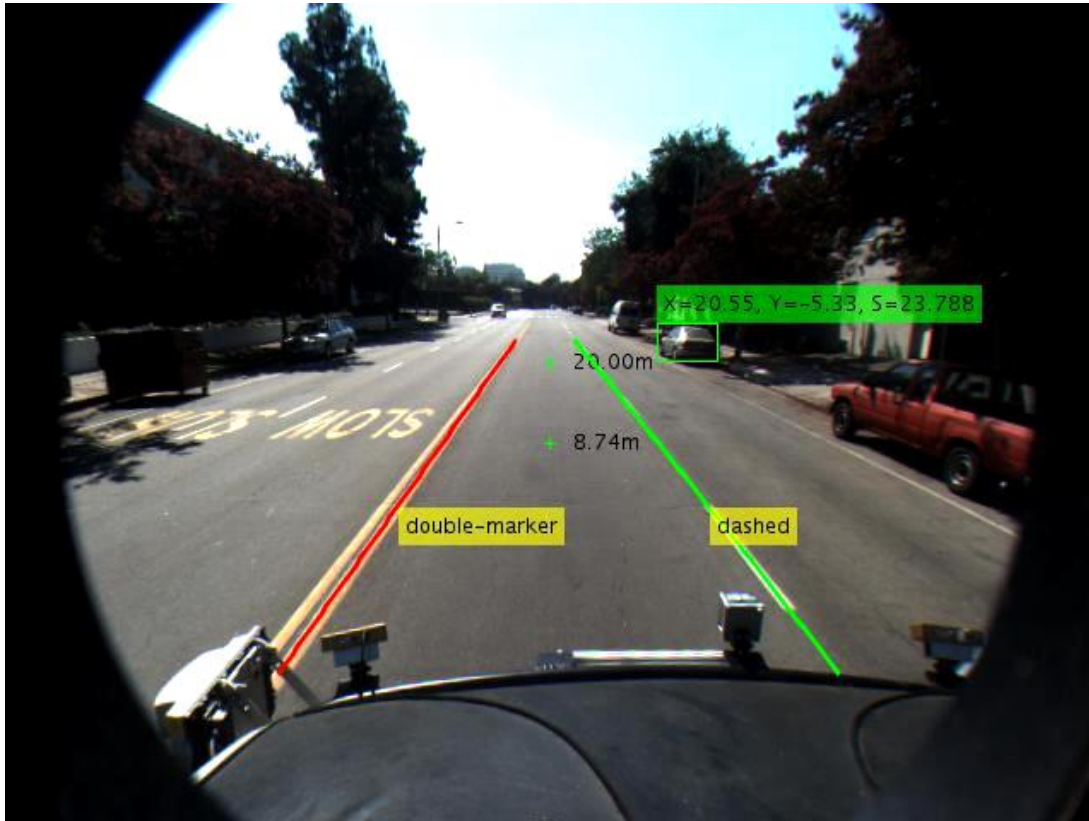
センサーフュージョン・
判断ロジック
をいかに開発し
検証するか？



認知部分の開発・
検証をいかに
効率よく行うか？

ユースケース 3

単眼カメラによる認識アルゴリズムモデル



提供機能

アルゴリズム

- RANSACによる区画線フィッティング
- 車検出器 (深層学習・ACF)

座標系の変換

- 車両座標系 <-> 画像座標系 の変換
- 単眼画像による、物体までの距離推定

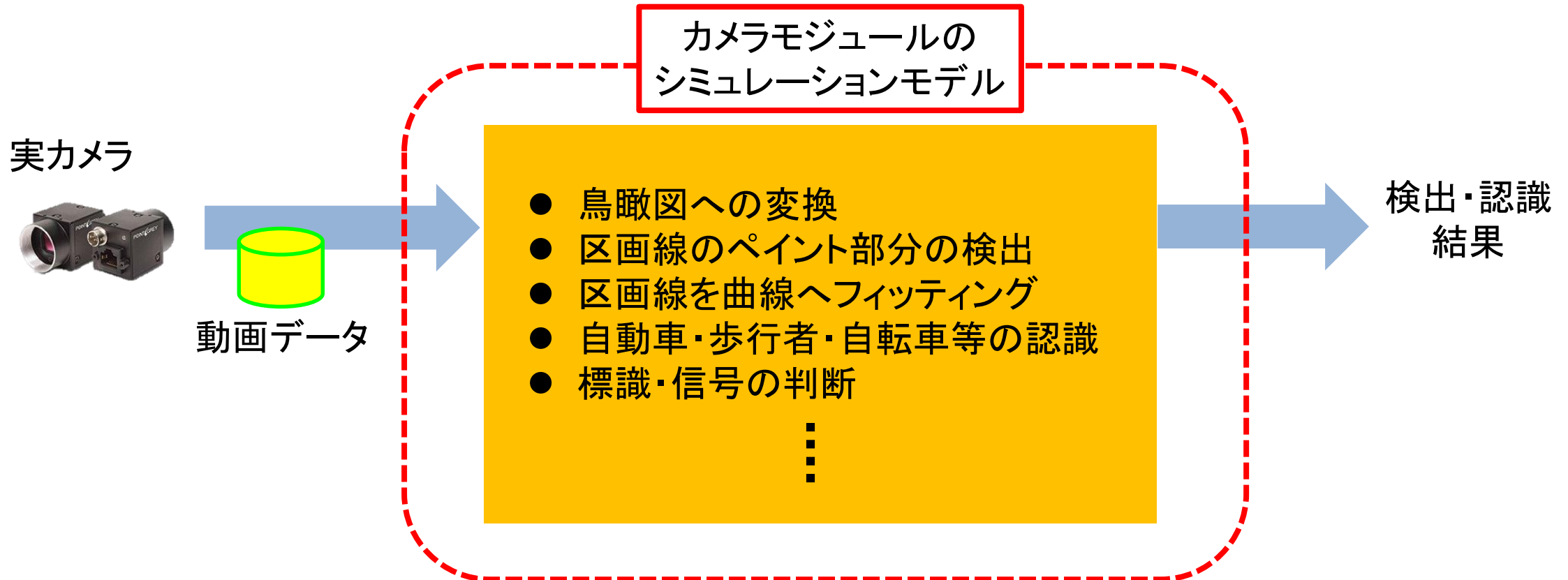
可視化機能

- 区画線
- 鳥瞰図

ビデオデータに対するカメラモジュール動作のシミュレーション
(ビデオデータから、オブジェクトリストの生成)

ユースケース 3a カメラモジュールのプロトタイピング

- カメラモジュールの内部アルゴリズムの理解・プロトタイピング



ユースケース 3b アルゴリズムをラベリングの自動化に適用

The screenshot displays the Ground Truth Labeler software interface. The main window shows a video frame titled "caltech_washington1.avi" with detected lane and car labels. The interface includes a menu bar with options like "FILE", "MODE", "VIEW", "AUTOMATE LABELING", "SUMMARY", and "エクスポート". The "ROI Label Definition" panel on the left lists "car" and "lane" labels. The "Scene Label Definition" panel on the left has a "Define New Scene Label" button. The video frame shows a street scene with lane markings and cars. Two yellow arrows point to the detected lane and car labels, with the text "区画線検出" (Lane Detection) and "車検出" (Car Detection) respectively. The bottom of the interface shows a timeline with "Start Time", "Current", "End Time", and "Max Time" fields, and a "Zoom In Time Interval" button.

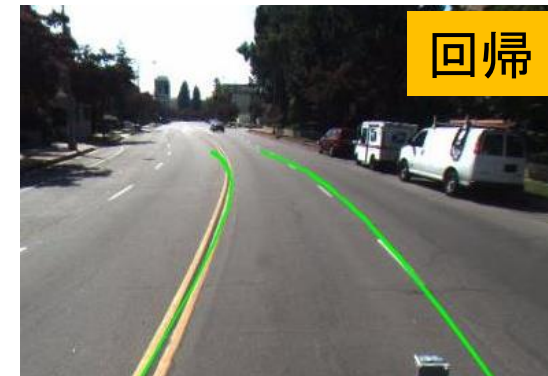
区画線検出

車検出

ユースケース4

深層学習 (Deep Learning for Automated Driving)

- 深層学習フレームワーク
 - R-CNN
 - Fast R-CNN **R2017a**
 - Faster R-CNN **R2017a**
 - 回帰 **R2017a**
- 学習済みの深層学習ネットワーク
 - 車検出器 **R2017a**
- Caffe モデルの取込み **R2017a**
- 学習済みのネットワークの取込み
 - AlexNet
 - VGG-16 Network **R2017a**
 - VGG-19 Network **R2017a**
- 複数GPUでの学習高速化 **R2017a**



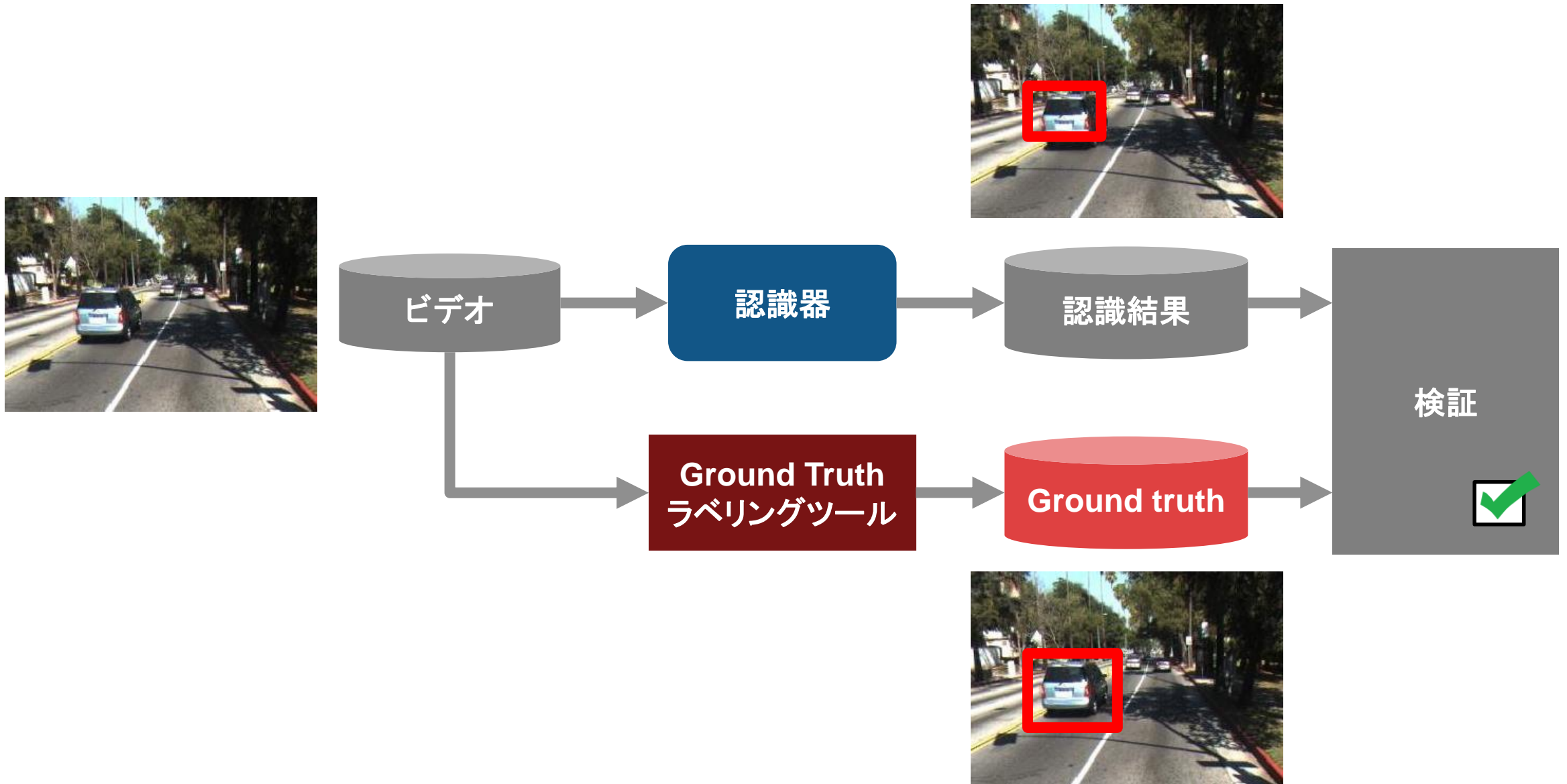
Neural Network Toolbox, Parallel Computing Toolbox™
compute capability 3.0以上のCUDA GPUが必要。

認識器



機械学習	Aggregate Channel Feature	<code>trainACFObjectDetector</code>
	Cascade	<code>trainCascadeObjectDetector</code>
深層学習	R-CNN (Regions with Convolutional Neural Networks)	<code>trainRCNNObjectDetector</code>
	Fast R-CNN	<code>trainFastRCNNObjectDetector</code>
	Faster R-CNN	<code>trainFasterRCNNObjectDetector</code>

認識器の評価



カスタマイズ可能な半自動Ground Truth ラベリング ツール

画像やデータの表示に関しても、APIにより**カスタマイズ可能**

- 半自動ラベリング機能（3つの半自動アルゴリズム）
- 先頭フレームに手動でつけたROIを、後続フレームで自動ラベリング（画像特徴量を使ったトラッキング）
 - 手動で付けたROIの間のフレームでROI位置を直線近似
 - 車検出器で自動ラベリング
 - **カスタムアルゴリズム**を使い自動ラベリングするためのAPI

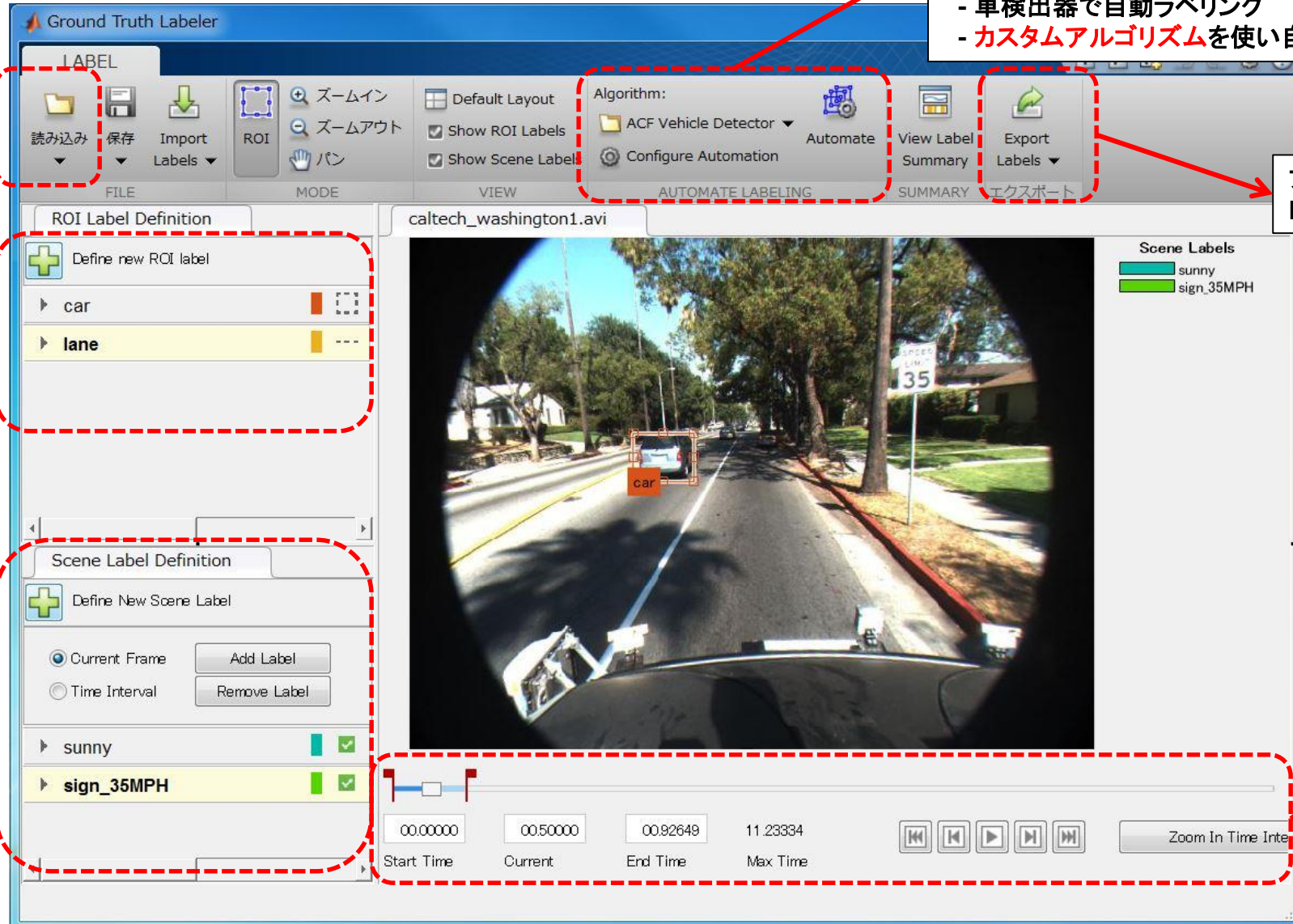
動画ファイル
連番静止画ファイル
カスタム関数による読み込み

ROIラベルの定義
(矩形もしくはポリライン)

ファイルもしくは
MATLABへ 結果の出力

シーン ラベルの定義
(1フレームずつもしくは 区間)

ビデオコントロール



デモ

カスタマイズ: Ground Truth Labeler App

The screenshot displays the Ground Truth Labeler application window titled "Ground Truth Labeler - gtlCustomizations". The interface includes a top toolbar with icons for Load, Save, Import Labels, ROI, Zoom In, Zoom Out, and Pan. Below the toolbar are tabs for DATA SOURCE, LABEL DEFINITIONS, and SESSION. The DATA SOURCE tab is active, and the "Custom Reader" option is highlighted with a red box. A large yellow arrow points from the "Custom Reader" option to a text box containing the text: "独自のファイル読み込み関数: groundTruthDataSource". The main workspace shows a video frame with labeled objects: "Car", "Pedestrian", and "Lane". A timeline at the bottom indicates the current time is 09.00000, with start and end times set to 00.00000 and 10.20000 respectively. The interface also includes a "Scene Label Definition" panel on the left and a "Zoom" button on the right.

独自のファイル読み込み関数:
groundTruthDataSource

カスタマイズ: Ground Truth Labeler App

The screenshot shows the Ground Truth Labeler app interface. On the left, there are panels for 'ROI Label Definition' and 'Scene Label Definition'. The 'ROI Label Definition' panel lists labels like Car, Pedestrian, StopLight, and Lane. The 'Scene Label Definition' panel has options for 'Current Frame' and 'Time Interval'. The main area shows a video frame with bounding boxes for 'Car' and 'Pedestrian'. A menu is open over the video frame, listing algorithms: 'Point Tracker', 'Temporal Interpolator', and 'ACF Vehicle Detector'. At the bottom of the menu, 'Create New Algorithm' and 'Import Algorithm' are highlighted with a red box. A yellow callout box at the bottom contains the text: '独自の自動ラベリングアルゴリズム driving.automation.AutomationAlgorithm'. The bottom of the interface shows a timeline with 'Start Time' (00.00000), 'Current' (09.00000), 'End Time', and 'Max Time'.

独自の自動ラベリングアルゴリズム
`driving.automation.AutomationAlgorithm`

カスタマイズ: Ground Truth Labeler App

The screenshot displays the Ground Truth Labeler application interface. On the left, there are panels for 'ROI Label Definition' and 'Scene Label Definition'. The 'ROI Label Definition' panel lists labels such as Car, Pedestrian, StopLight, and Lane. The 'Scene Label Definition' panel includes options for 'Current Frame' and 'Time Interval'. The main workspace shows a video frame with bounding boxes and labels for 'Car', 'Pedestrian', and 'Lane'. A yellow arrow points from the text '他の表示関数やデータとの同期表示' to the 'driving.connector.Connector' label. To the right, a window titled 'Figure 1: Point Cloud Pla...' shows a 3D point cloud visualization of the scene with blue lines representing lane markings.

他の表示関数やデータとの同期表示
driving.connector.Connector

多くのサンプルプログラムを同梱

- ADAS/自動運転 開発に関連する機能をサンプルとして提供
 - スタートアップポイントとして
 - 更に機能を拡張・カスタマイズして必要な機能を実装

Automated Driving System Toolbox Examples

Reference Applications



Visual Perception Using Monocular Camera

Construct a monocular camera sensor simulation capable of lane boundary and vehicle detections. The sensor will report these

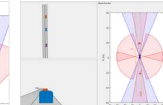
[Open Script](#)



Forward Collision Warning Using Sensor Fusion

Perform forward collision warning by using data from vision and radar sensors to track objects in front of the vehicle.

[Open Script](#)



Sensor Fusion Using Synthetic Radar and Vision Data

Generate a scenario, simulate sensor detections and use sensor fusion to track simulated vehicles. The main benefit of using scenario

[Open Script](#)

Tracking and Sensor Fusion



Forward Collision Warning Using Sensor Fusion

Perform forward collision warning by using data from vision and radar sensors to track objects in front of the vehicle.

[Open Script](#)



Track Multiple Vehicles Using a Camera

Detect and track multiple vehicles with a monocular camera mounted in a vehicle.

[Open Script](#)



Track Pedestrians from a Moving Car

Track pedestrians using a camera mounted in a moving car.

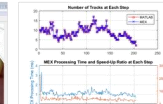
[Open Script](#)



Multiple Object Tracking Tutorial

Perform automatic detection and motion-based tracking of moving objects in a video. It simplifies the example Motion-Based Multiple

[Open Script](#)



Code Generation for Tracking and Sensor Fusion

Generate C code for a MATLAB function that processes data recorded from a test vehicle and tracks the objects around it.

[Open Script](#)

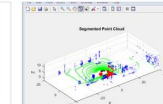
Perception with Computer Vision



Visual Perception Using Monocular Camera

Construct a monocular camera sensor simulation capable of lane boundary and vehicle detections. The sensor will report these

[Open Script](#)



Ground Plane and Obstacle Detection Using Lidar

Detect the ground plane and find nearby obstacles in 3-D Lidar data. This process can facilitate drivable path planning for vehicle

[Open Script](#)



Train a Deep Learning Vehicle Detector

Train a vision-based vehicle detector using deep learning.

[Open Script](#)

Algorithm Validation and Visualization



Automate Ground Truth Labeling of Lane Boundaries

Develop an algorithm for the automated marking of lane boundaries in the groundTruthLabeler app.

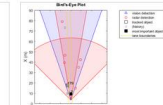
[Open Script](#)



Annotate Video Using Detections in Vehicle Coordinates

Configure and use a monoCamera object to display information provided in vehicle coordinates on a video display.

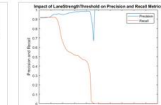
[Open Script](#)



Visualize Sensor Coverage, Detections, and Tracks

Configure and use a Bird's-Eye Plot to display sensor coverage, detections and tracking results around the ego vehicle.

[Open Script](#)

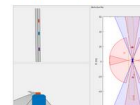


Evaluate Lane Boundary Detections Against Ground Truth Data

Compare ground truth data against results of a lane boundary detection algorithm. It also illustrates how this comparison can be used to tune

[Open Script](#)

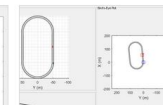
Scenario Generation



Sensor Fusion Using Synthetic Radar and Vision Data

Generate a scenario, simulate sensor detections and use sensor fusion to track simulated vehicles. The main benefit of using scenario

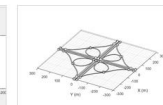
[Open Script](#)



Driving Scenario Tutorial

Generate ground truth for synthetic sensor data and tracking algorithms. It shows how to update actor poses in open-loop and

[Open Script](#)



Define Road Layouts

Create a variety of road junctions with Automated Driving System Toolbox™. These junctions may be combined with other junctions to

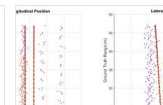
[Open Script](#)



Create Actor and Vehicle Paths

Create actor and vehicle paths for a driving scenario using the Automated Driving System Toolbox™.

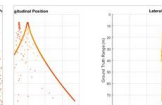
[Open Script](#)



Model Radar Sensor Detections

Model and simulate the output of an automotive radar sensor for different driving scenarios. Generating synthetic radar detections is

[Open Script](#)

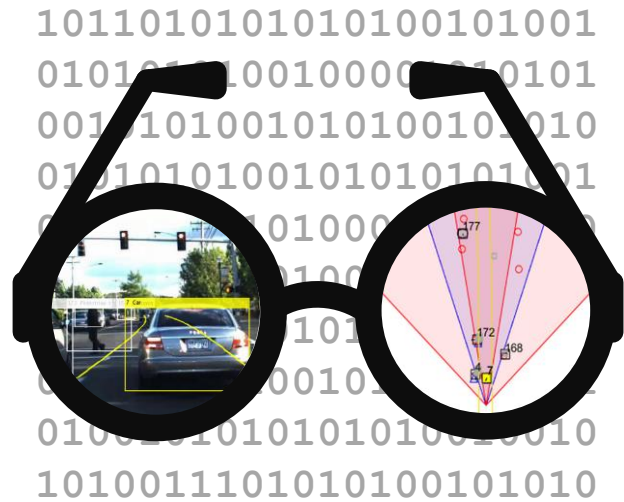


Model Vision Sensor Detections

Model and simulate the output of an automotive vision sensor for different driving scenarios. Generating synthetic vision

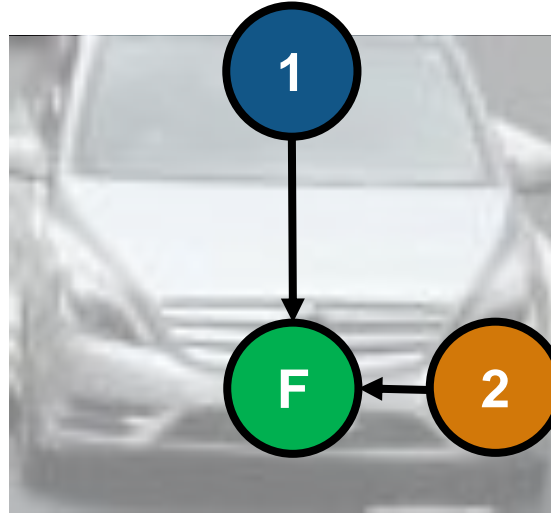
[Open Script](#)

Automated Driving System Toolbox



センサーデータの可視化

- センサーデータのプロット
- 検知領域の表示
- 画像座標と、車両座標(鳥瞰図)の相互変換



センサーフュージョン・判断ロジックの開発

- トラッキングアルゴリズム
- Cコード生成
- シナリオ生成機能



認識の開発・検証

- 様々な認識用のアルゴリズム
- 深層学習
- Ground Truthラベリングツール