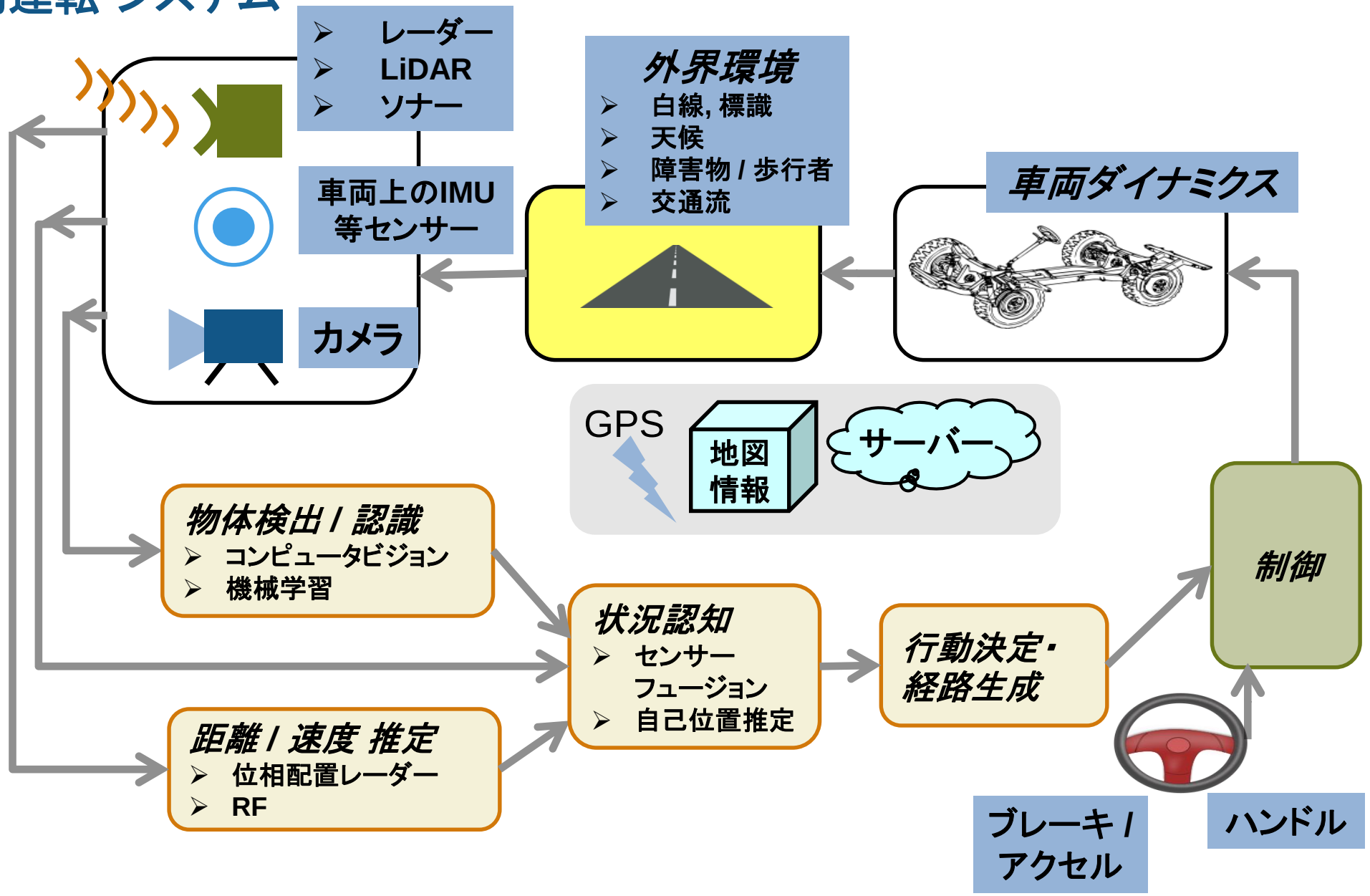
自動運転・ADASの開発・検証ソリューション

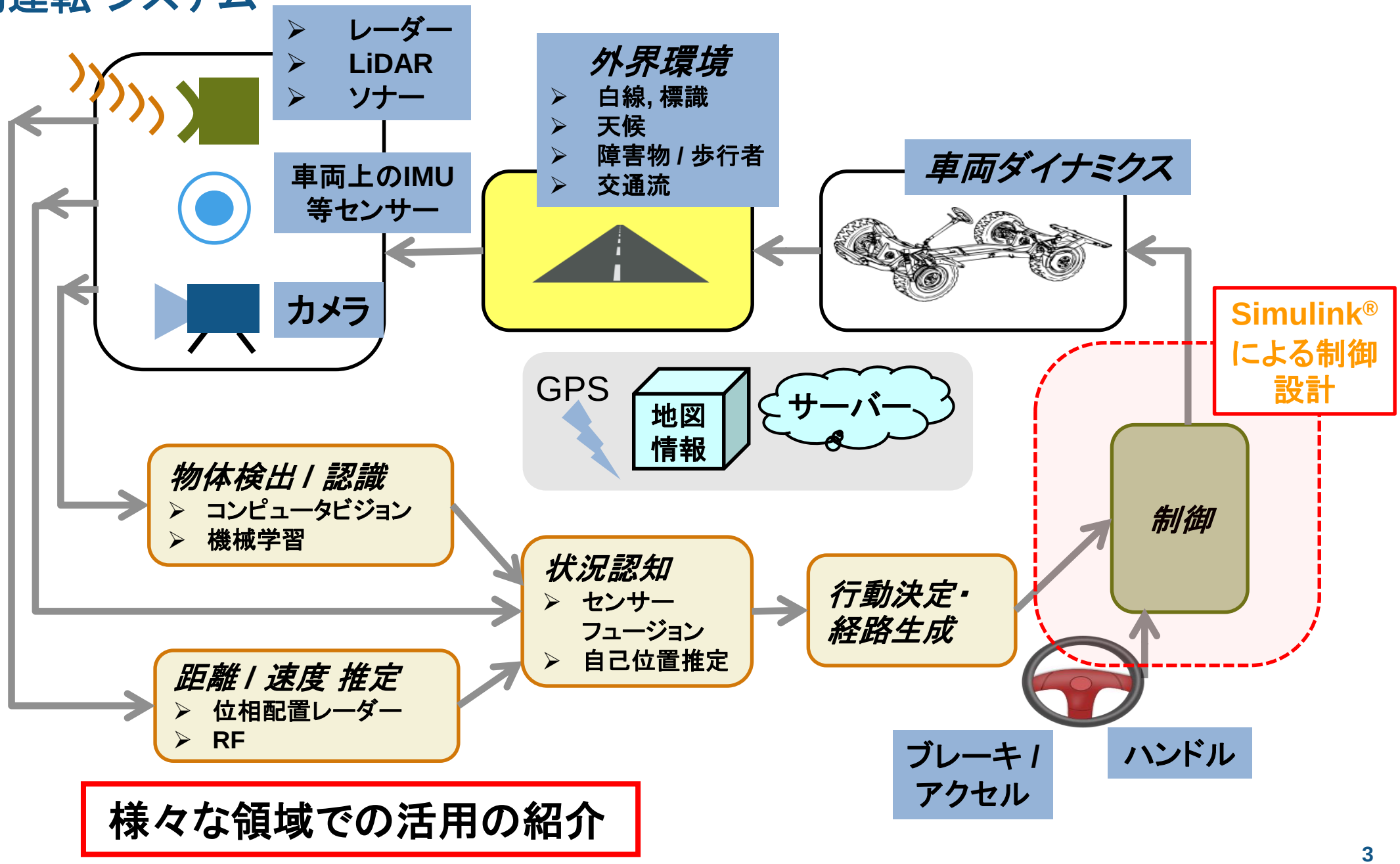
MathWorks Japan
アプリケーションエンジニアリング部　信号処理・通信
乙部　雅則

Automated Driving System Toolbox™

ADAS/自動運転 システム



ADAS/自動運転 システム

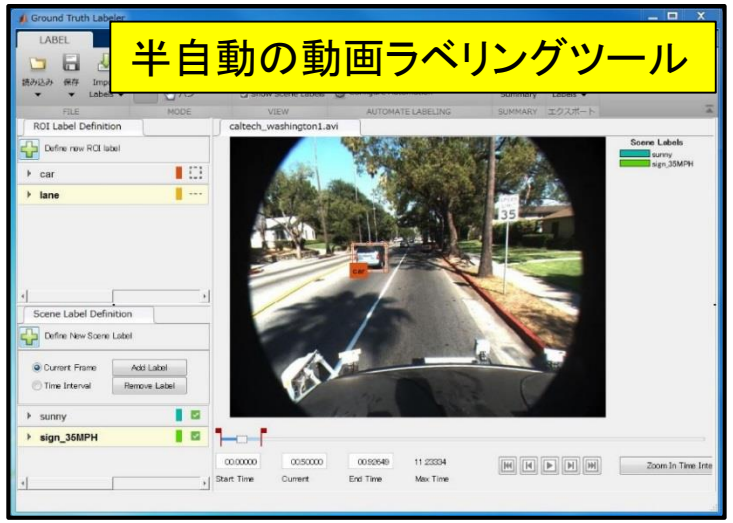


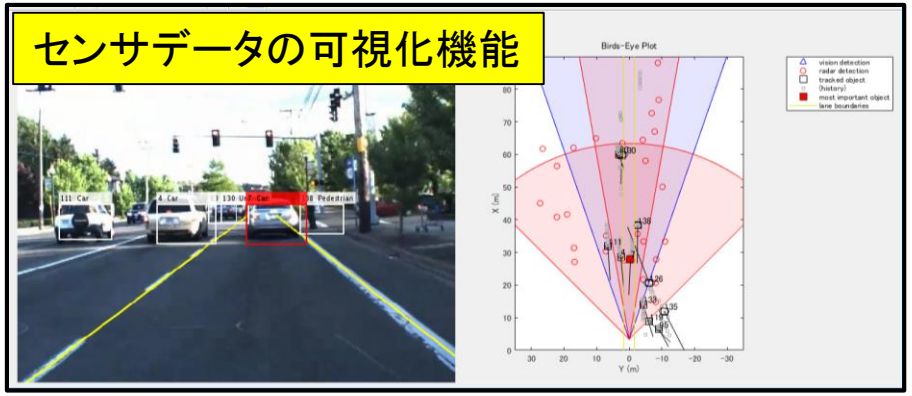
様々な領域での活用の紹介

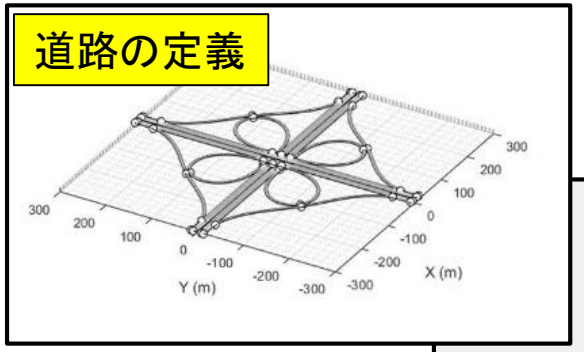
Automated Driving System Toolbox

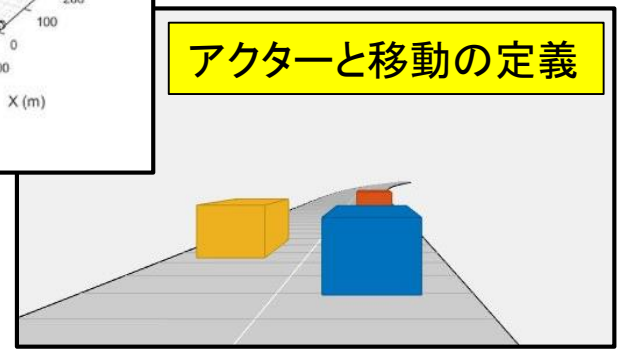
コンセプト：ADAS/自動運転 開発・検証環境の統合プラットフォーム

1. 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
2. Ground Truth ラベリングツール
3. センサデータの可視化
4. テストシナリオ生成







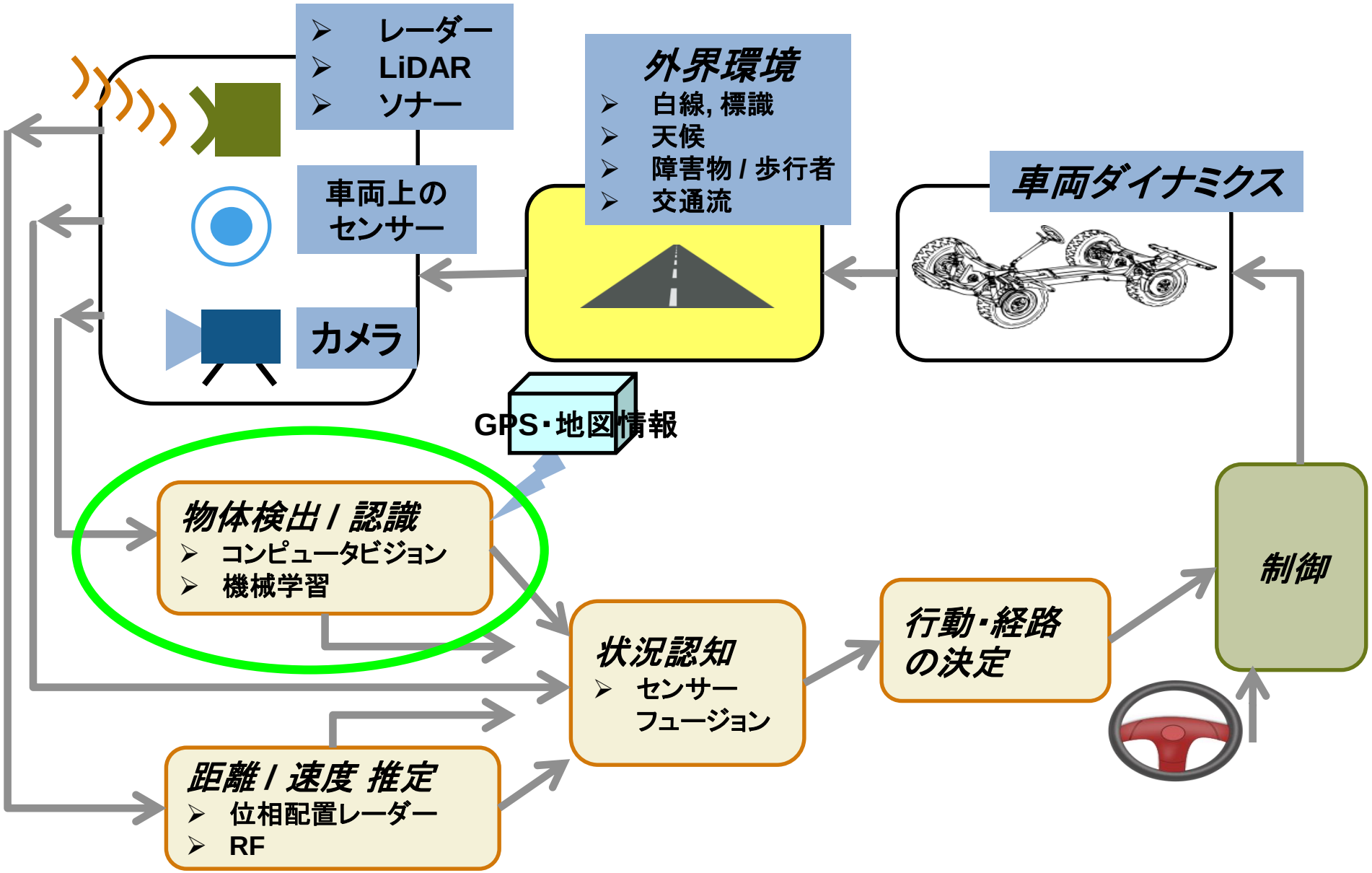


アジェンダ

- 画像処理・画像認識領域での活用
- 検出器の評価・検証
- センサーデータ可視化
- シナリオ生成

- まとめ

Automated Driving System Toolbox

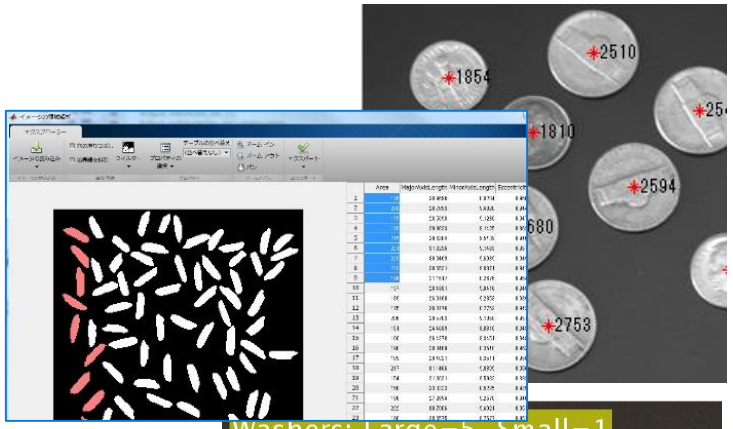


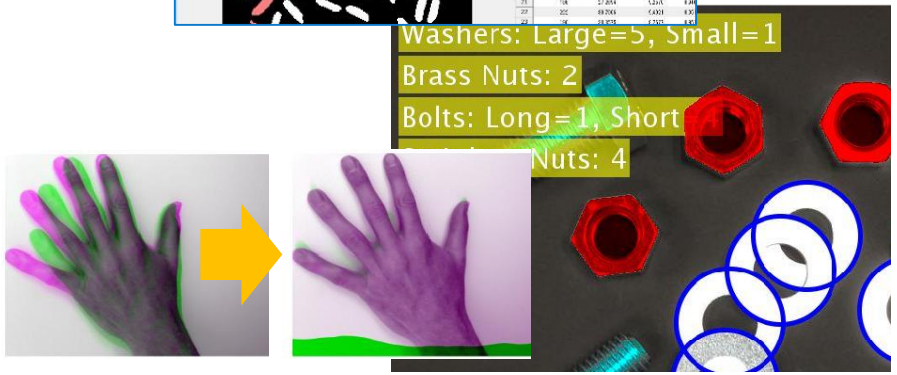
画像処理･コンピュータビジョン用オプション製品群

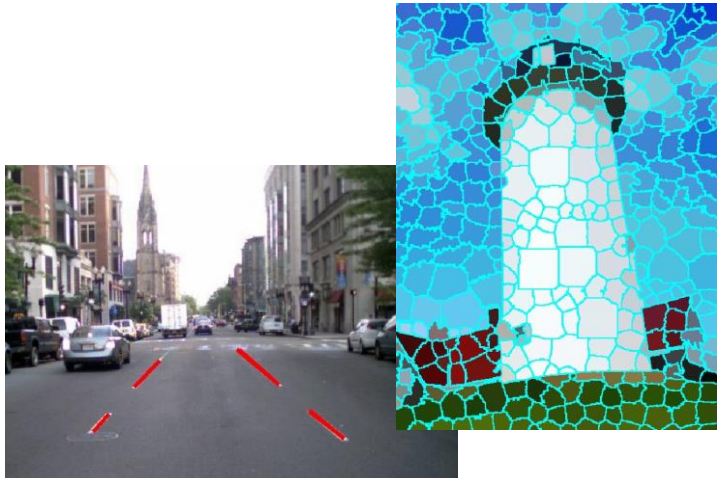
（Automated Driving System Toolbox の前提オプション製品）

画像処理のベースとなる関数群
Image Processing Toolbox™

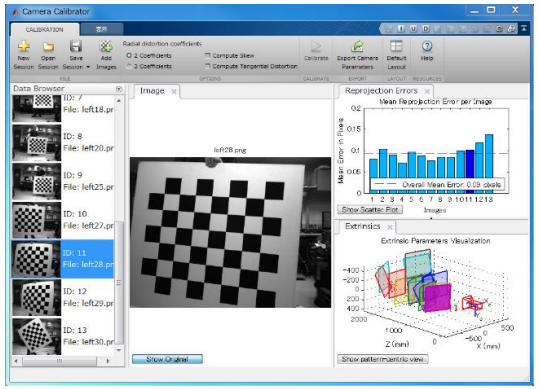
- 各種画像データの読み書き
- 画像調査用の各種アプリケーション
- 画像（色・コントラスト等）の調整・変換
- 幾何学的変換（位置や形の変換）
- レジストレーション（位置合せ）
- 各種画像フィルタ処理
- モルフォロジー処理（膨張・収縮等の様々な形態処理）
- オブジェクト（物体）検出・解析
- セグメンテーション（領域切出し）
- 画像の領域の定量評価
- ROIベースの処理（特定領域処理）



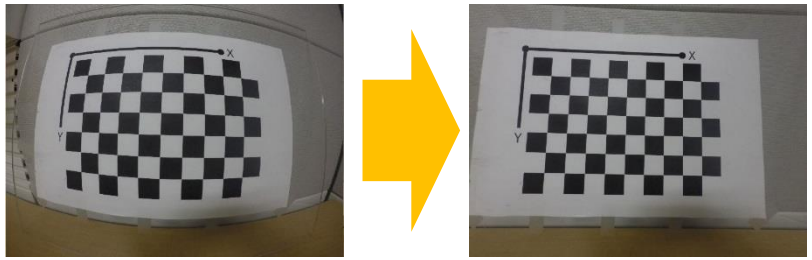


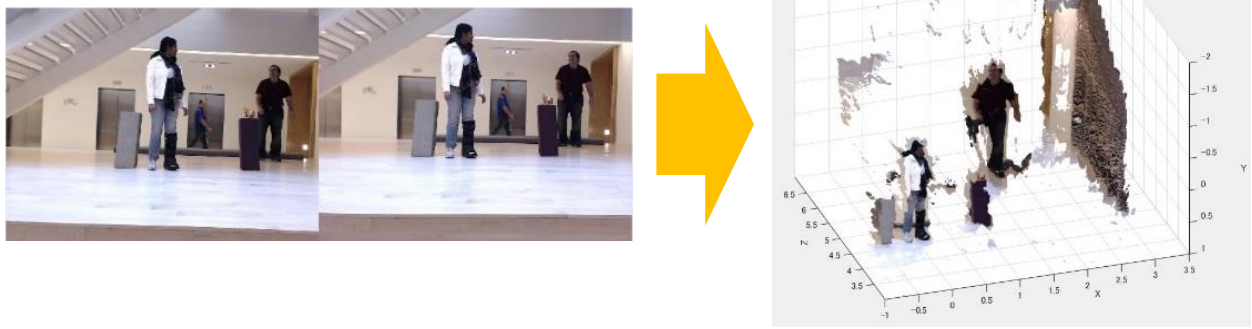


画像処理やコンピュータビジョンのための機能・高速ストリーミング処理
Computer Vision System Toolbox™



- カメラキャリブレーション
- グラフィックス
- 特徴点検出・特徴量抽出、
  マッチング・レジストレーション
- 物体認識、文字認識（OCR）
  - 顔・人物認識、機械学習による物体認識
  - Bag-of-Visual Wordsによる物体認識・画像検索
  - 深層学習による物体検出（R-CNN）
- 動画ストリーミングデータの高速処理
- 物体のトラッキング・動体検出
- ステレオビジョン向けワークフロー
- 3次元点群処理 ・ Structure from Motion
- 画像処理用のSimulink ブロックセット



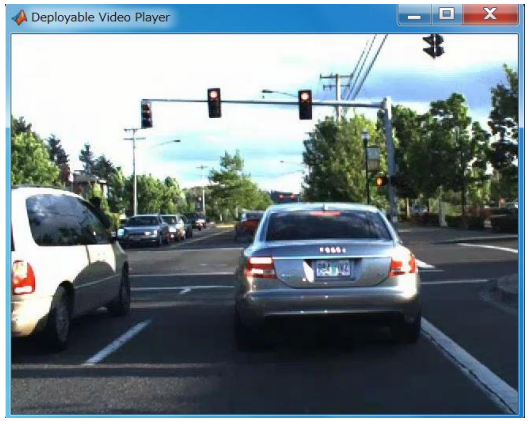


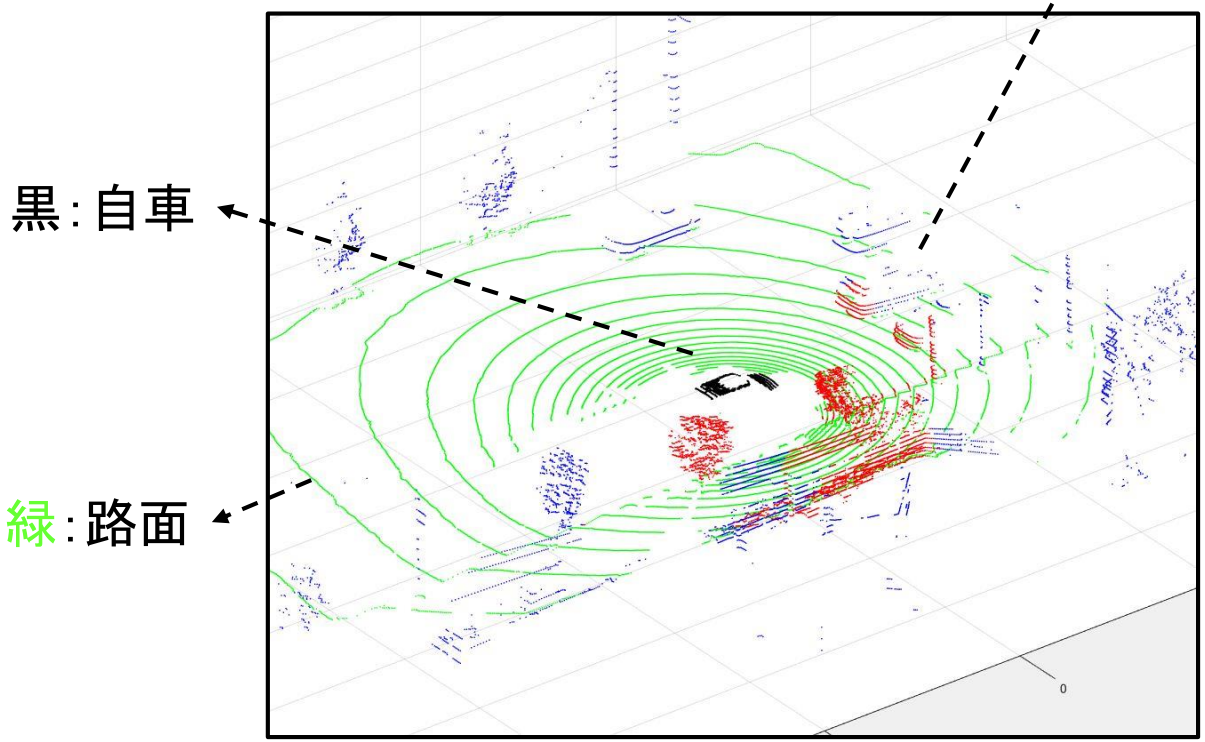
LiDAR信号の可視化･各種処理

- LiDAR: **Li**ght **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

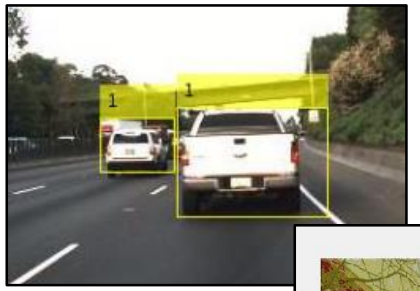


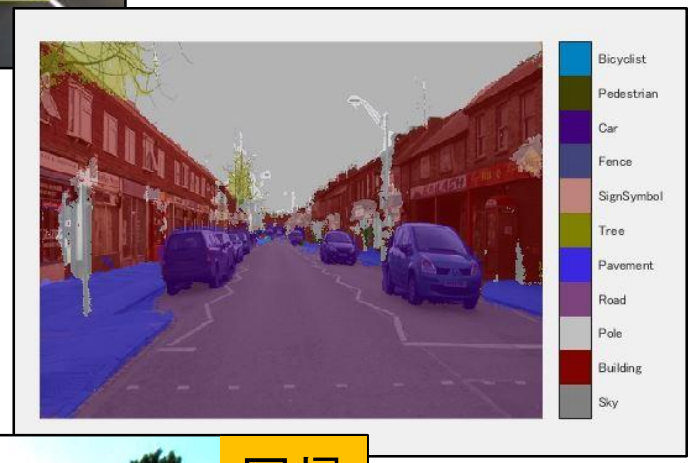


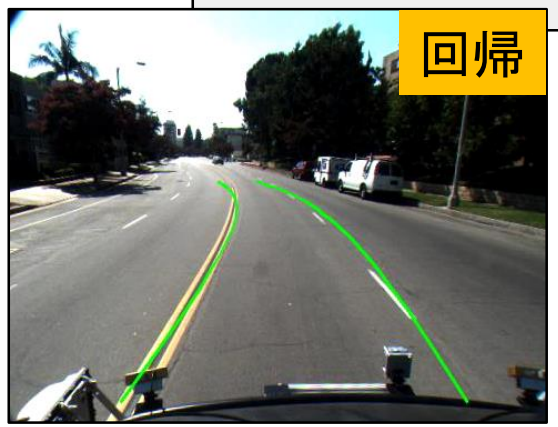
深層学習（Deep Learning for Automated Driving ）

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

- 深層学習フレームワーク
  - 回帰 R2017a
  - DAG (Directed Acyclic Graph) Network R2017b
  - Semantic Segmentation（SegNet, FCN） R2017b
  - LSTM（Long Short-Term Memory Networks） R2017b
  - カスタムレイヤーの定義 R2017b
- 学習済みの深層学習ネットワーク
  - AlexNet R2017a
  - VGG-16 Network、VGG-19 Network R2017b
  - GoogLeNet R2017b Oct
  - ResNet-50 R2017a
  - 車検出器（Automated Driving System Toolbox）
- ネットワークのインポート
  - Caffe モデル R2017a
  - TensorFlow-Keras R2017b Oct
- 深層学習の応用
  - 画像からのノイズ除去（DnCNN） R2017b
- 複数GPUでの学習高速化 R2017a

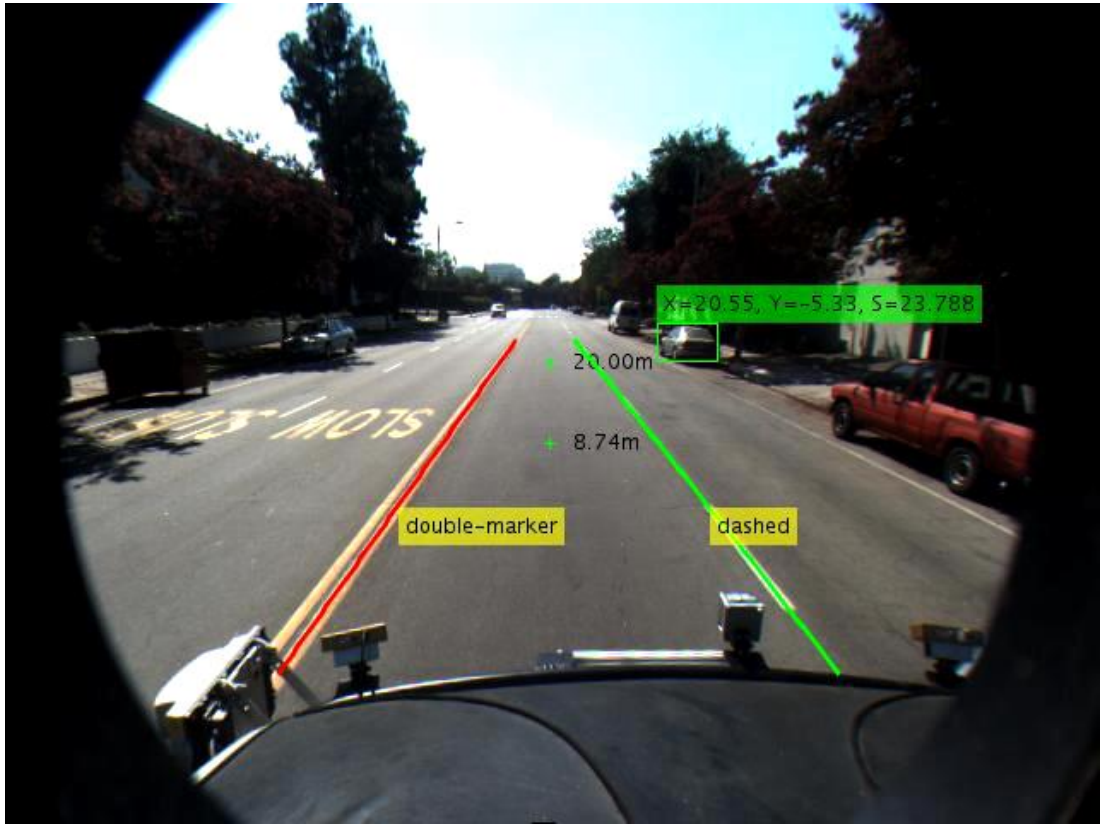


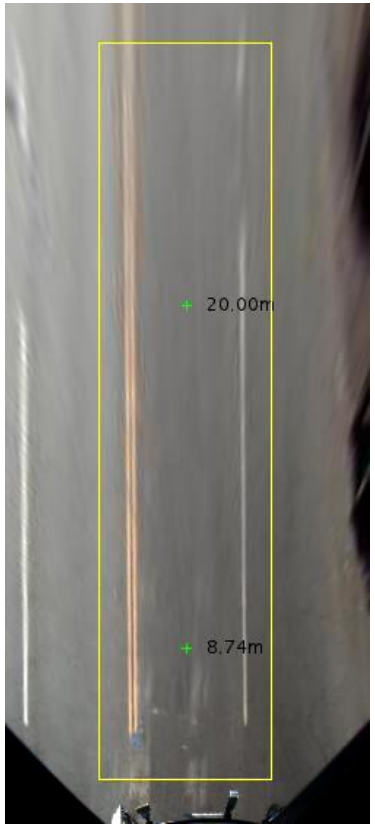




Automated Driving System Toolboxの提供する画像認識関連機能
～単眼カメラによる認識アルゴリズムモデル ～

デモ





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（birdsEyeViewクラス）

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）

画像から物体認識や距離推定するための豊富な部品機能
カスタマイズも可能
容易な画像認識ラピッドプロトタイピング
動画からのデータ抽出

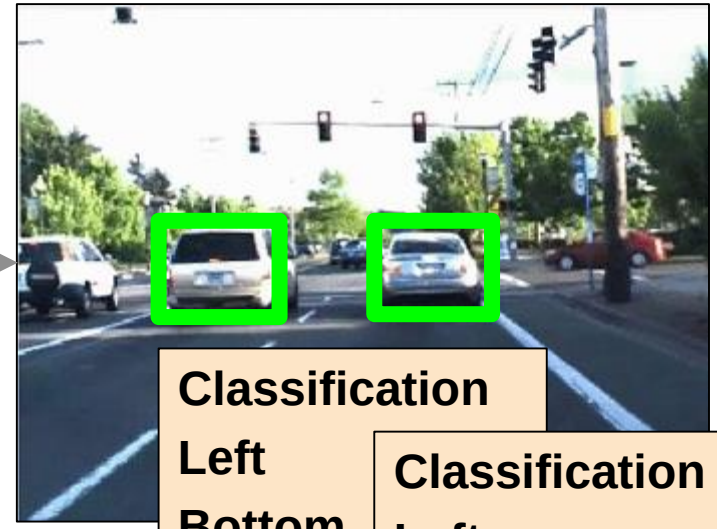
アジェンダ

- 画像処理・画像認識領域での活用
- 検出器の評価・検証
- センサーデータ可視化
- シナリオ生成

- まとめ

物体検出

検出器



Classification
Left
Bottom
Width
Height

Classification
Left
Bottom
Width
Height

検出器の評価

例）10万枚 / 時間 (30fps)

動画

検出器

認識結果

検証

Ground Truth
ラベリングツール

Ground truth

どのように作成？

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

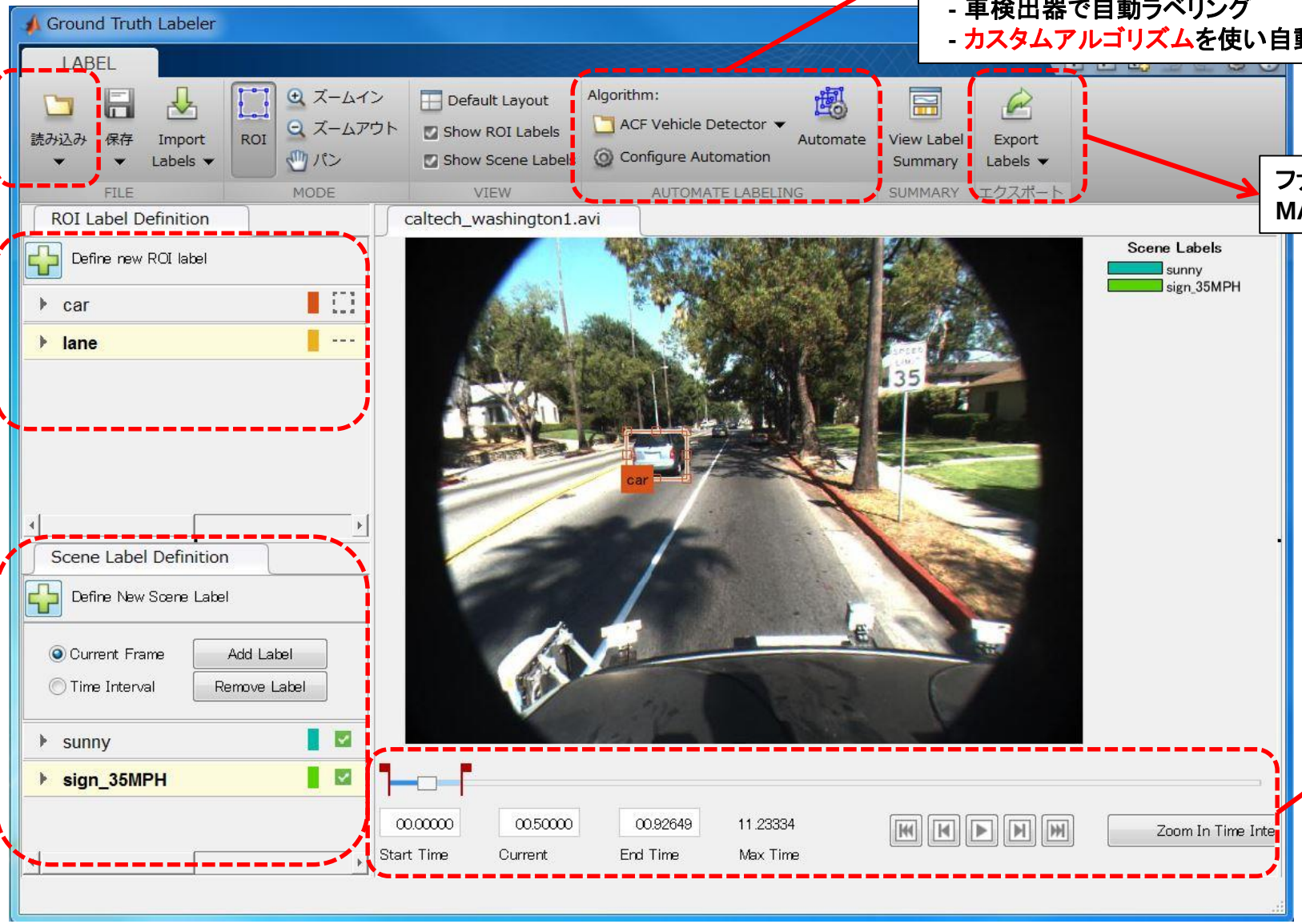
動画ファイル
連番静止画ファイル
カスタム関数による読込み

ROIラベルの定義
（矩形 もしくは ポリライン）

シーン ラベルの定義
（1フレームずつ もしくは 区間）

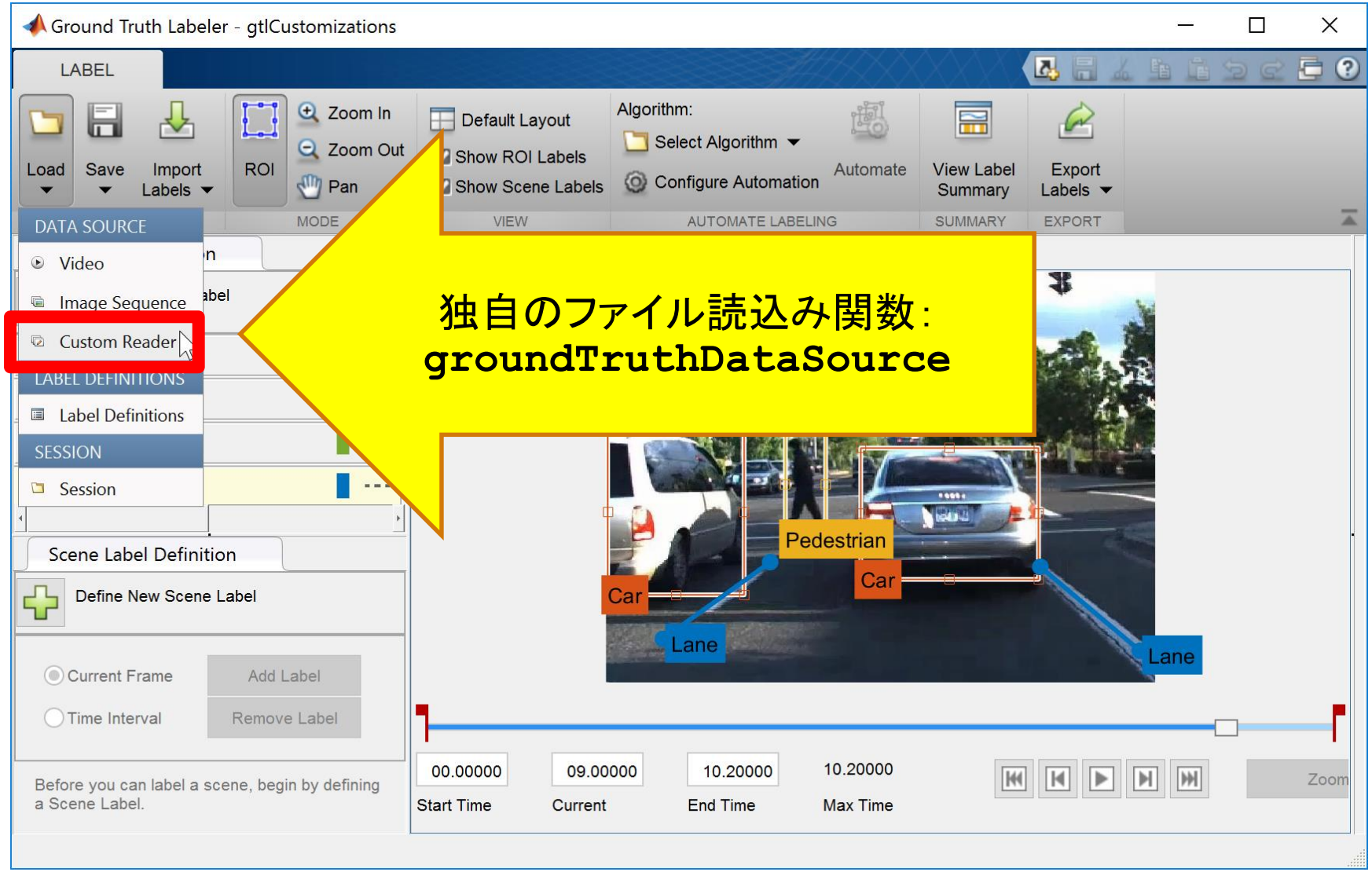
ファイル もしくは
MATLABへ 結果の出力

ビデオコントロール

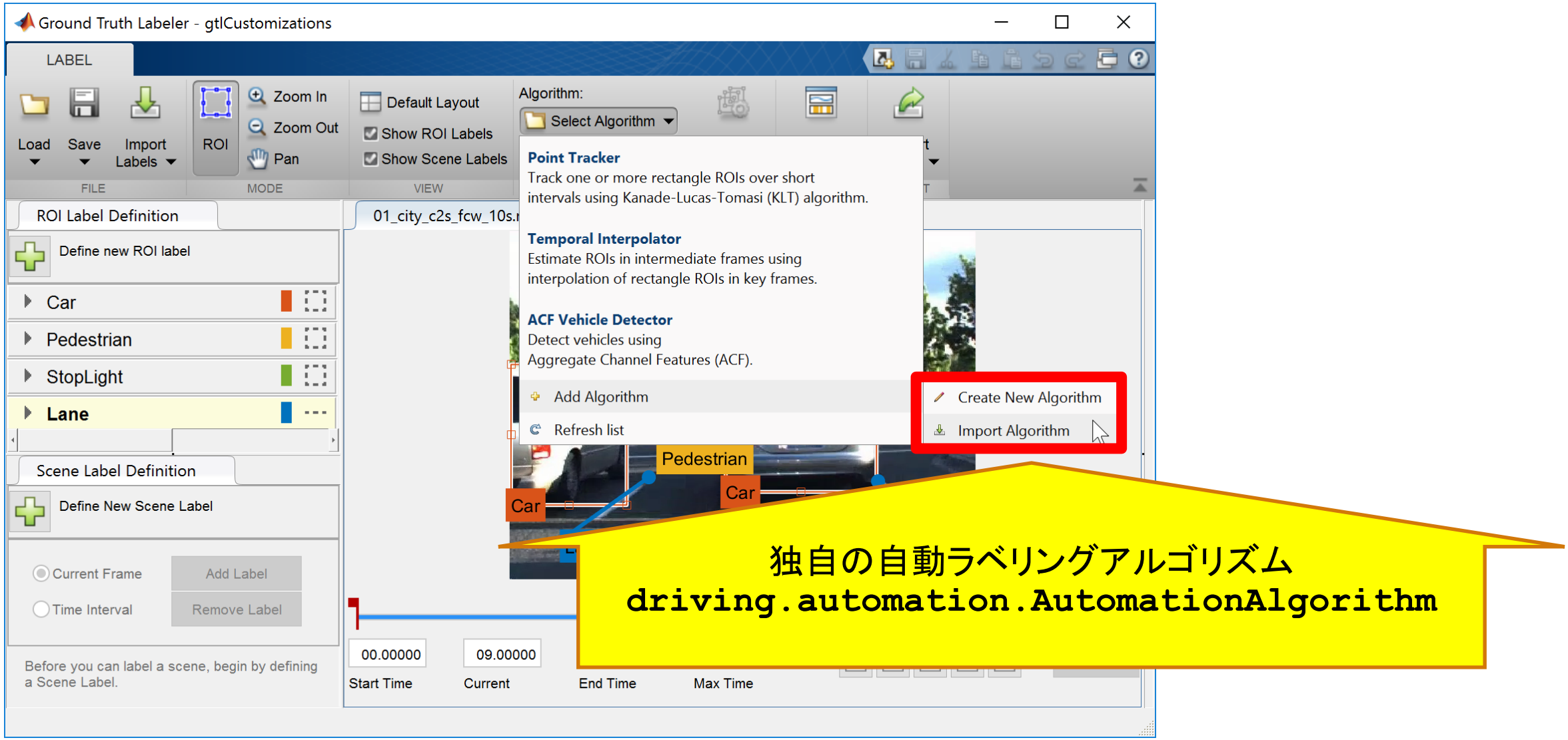


デモ

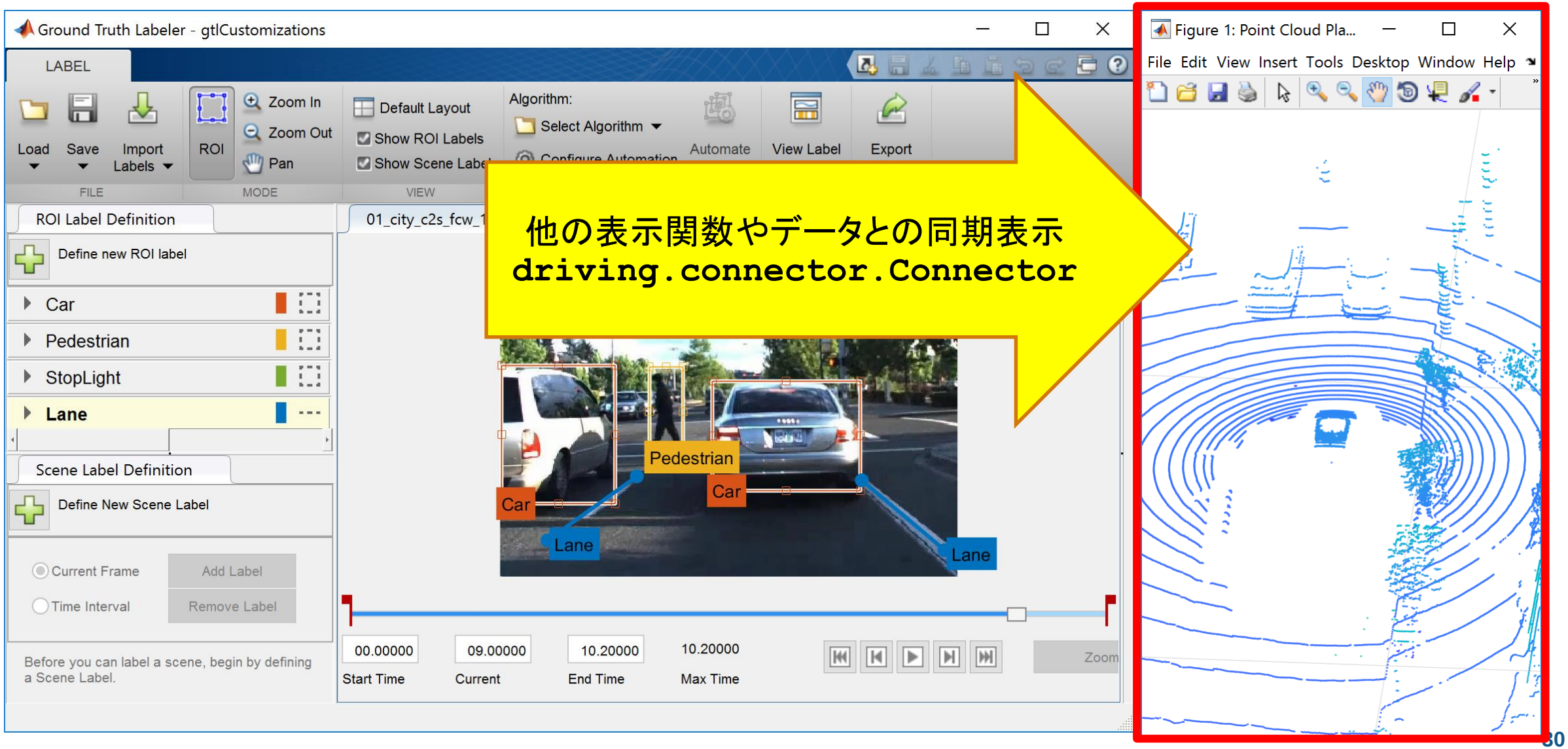
各種カスタマイズ：Ground Truth Labeler App



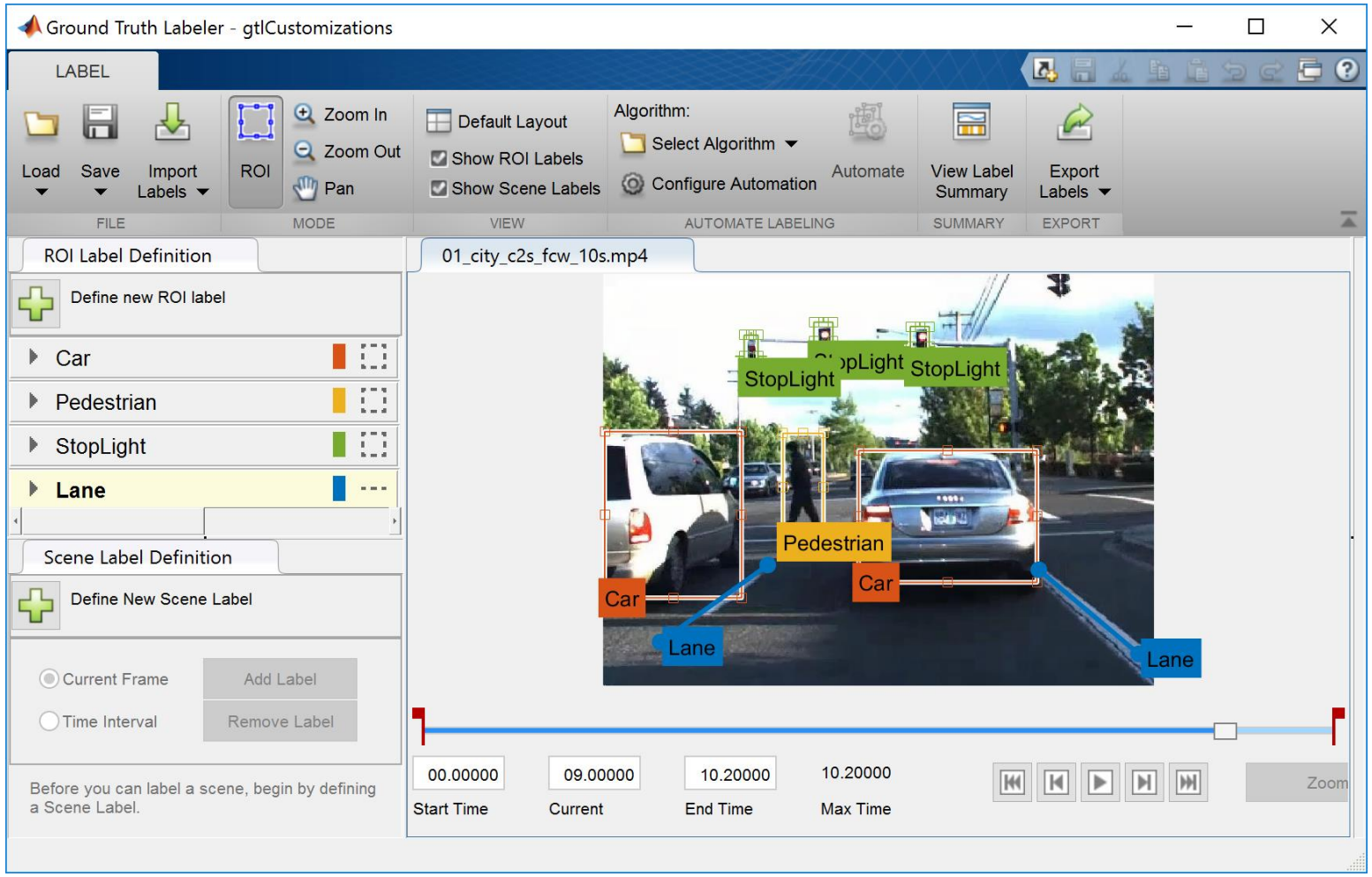
各種カスタマイズ：Ground Truth Labeler App



各種カスタマイズ：Ground Truth Labeler App

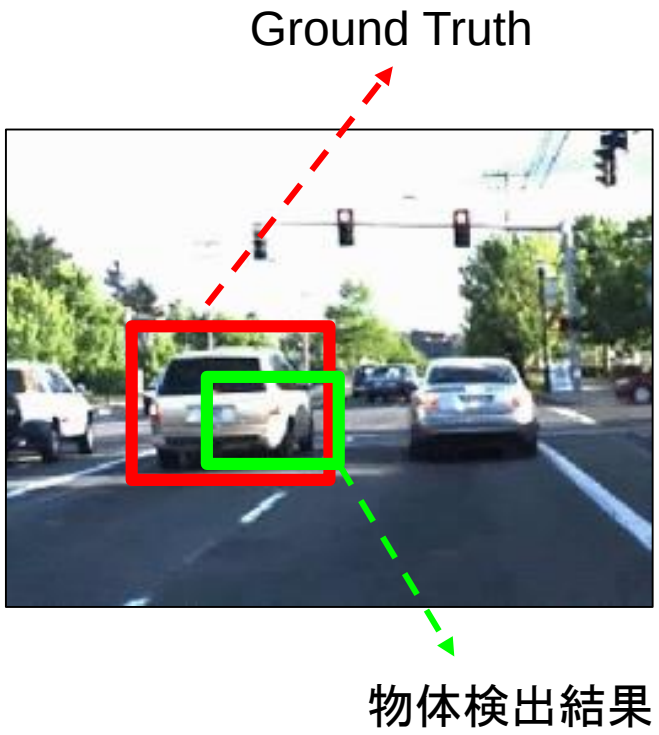


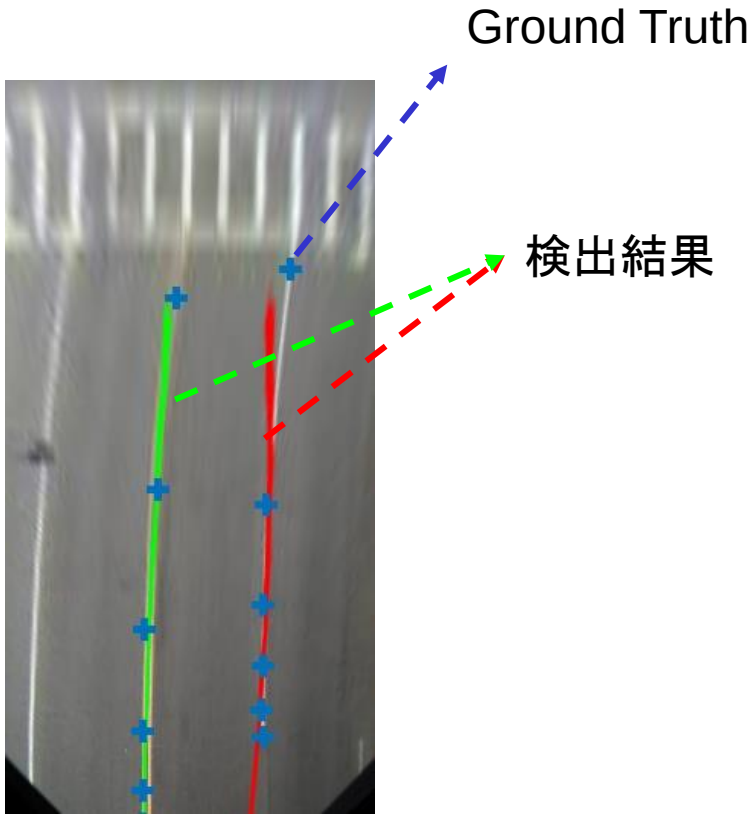
カスタマイズ可能な半自動Ground Truth ラベリング ツール



使いやすく、独自ファイル読込み・自動化アルゴリズム追加・表示機能の拡張等、様々なカスタマイズが可能

評価用の関数





- 物体検出結果の評価
  - evaluateDetectionPrecision
  - evaluateDetectionMissRate
- 区画線検出結果の評価
  - evaluateLaneBoundaries

アジェンダ

- 画像処理・画像認識領域での活用
- 検出器の評価・検証
- センサーデータ可視化
- シナリオ生成

- まとめ

自動運転車両のセンサー例

カメラ：動画像

カメラ：物体検出

カメラ：区画線検出

レーダー

Lidar

IMU（角速度等）

自動運転車両のセンサー例

カメラ：動画像 (640 x 480 x 3)

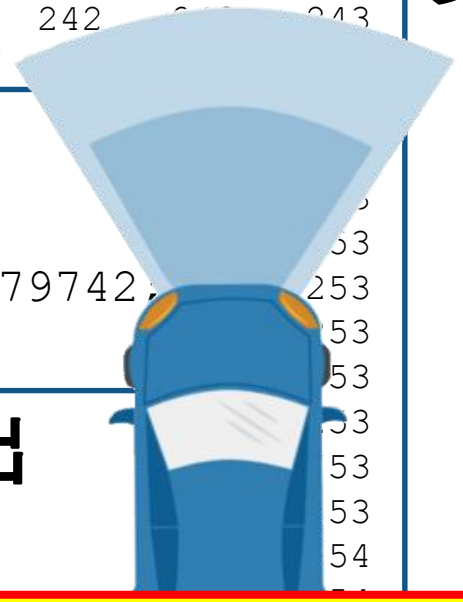
239 239 237 238 241 241 241 242
252 252 251 252 252 253 253

カメラ：物体検出

SensorID = 1;
Timestamp = 146163469637974
NumDetections = **6**;
Dete
Tr
Cl
Po
Ve
Si
Detec
Tr
Cl
Po
Ve
Si

カメラ：区画線検出

Left
IsValid:
Confidence:
BoundaryType
Offset: 1.68
HeadingAngle: 0.002
Curvature: 0.000
Right
IsValid: 1



各種データの統合的可視化の必要性

レーダー

SensorID = 2;
Timestamp = 146163469640752
NumDetections = **23**;
Detection
TrackID
TrackSt
Positio
Velocit
Amplitu
Detection

Lidar (47197 x 3)

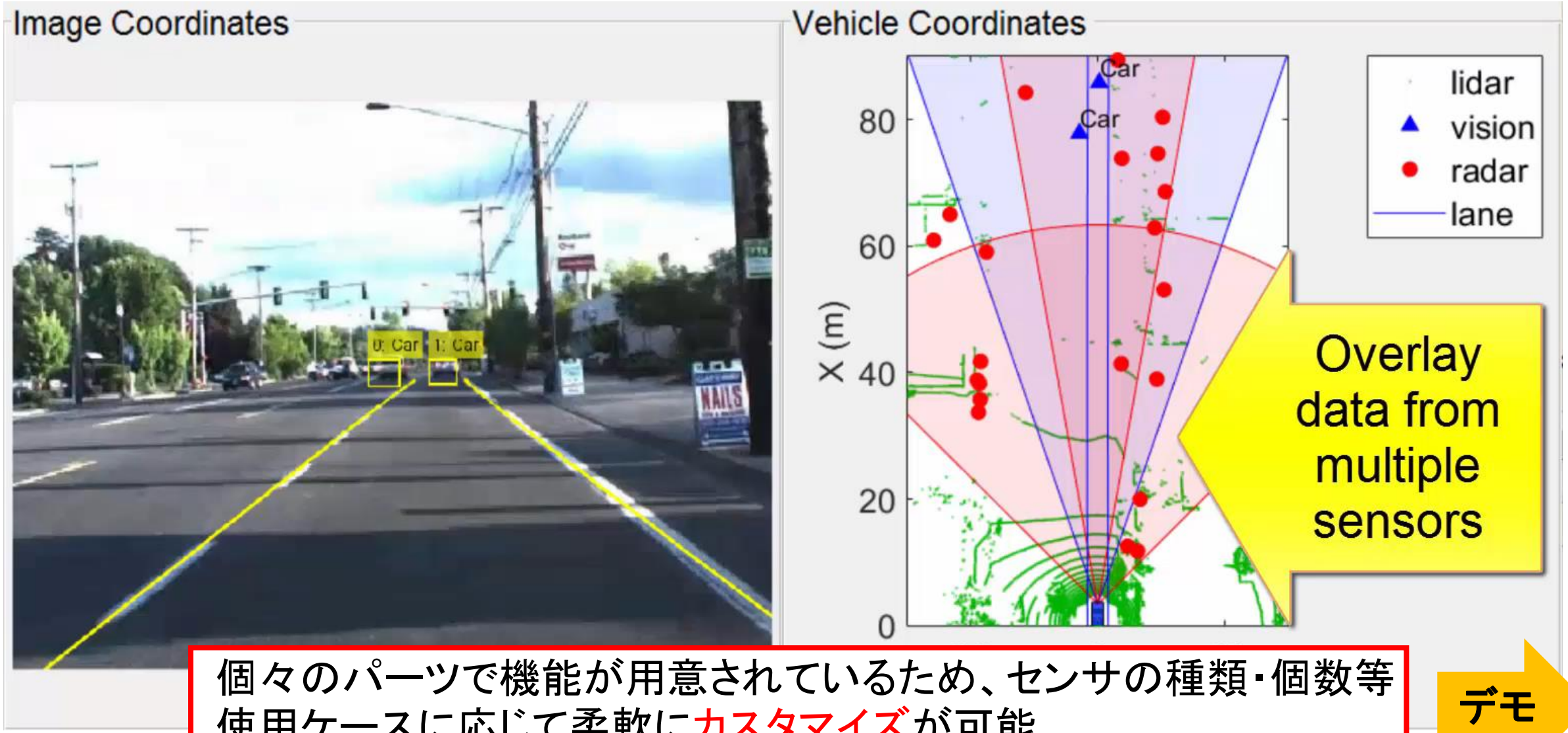
-12.2911 1.4790 -0.59
-14.8852 1.7755 -0.64
-18.8020 2.2231 -0.73
-25.7033 3.0119 -0.92
-0.0632 0.0815 1.25
.0978 0.0855 1.25
.2814 0.1064 1.25
1.26
1.25
1.24
1.23
-0.64
-0.74

IMU

Timestamp: 146163469637974
Velocity: 9.2795
YawRate: 0.0040

各種センサーの検出結果の可視化：差分比較の例

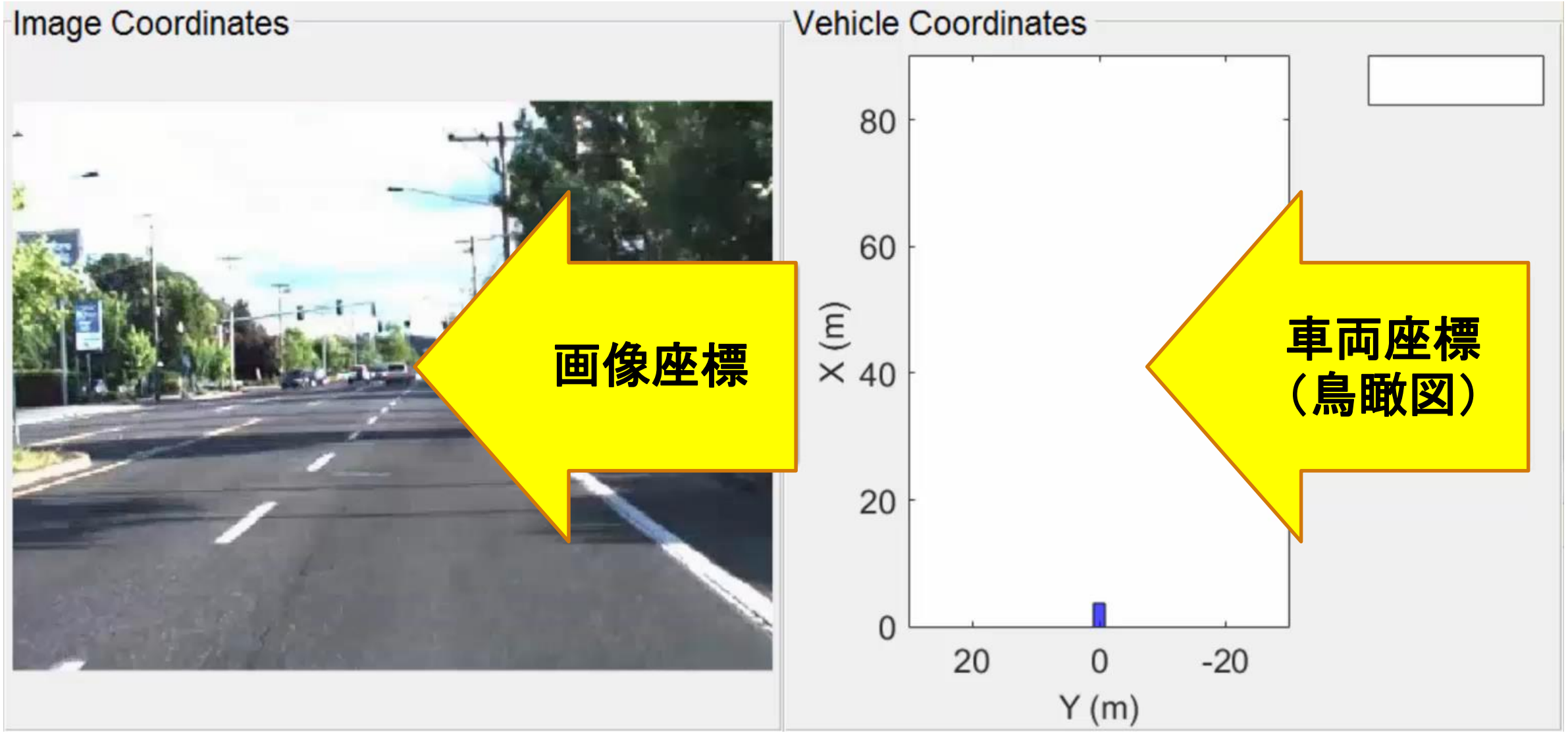
動画



個々のパーツで機能が用意されているため、センサの種類・個数等使用ケースに応じて柔軟にカスタマイズが可能

デモ

センサーデータの可視化例



（準備：センサーデータの取込み）

- 動画データの設定、カメラのパラメータの読込み

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

- センサ（画像・レーダー）データ・センサパラメータの読込み

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

- LiDARの点群データの読込み

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

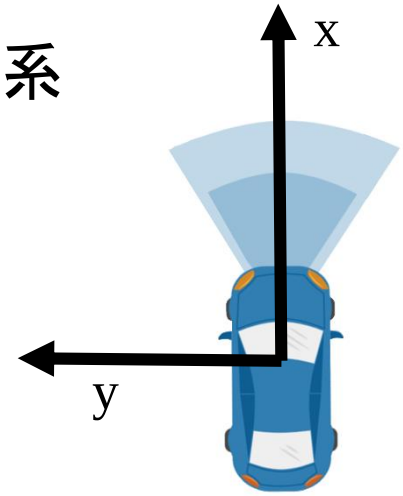
画像座標への表示

```
%% 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);
```

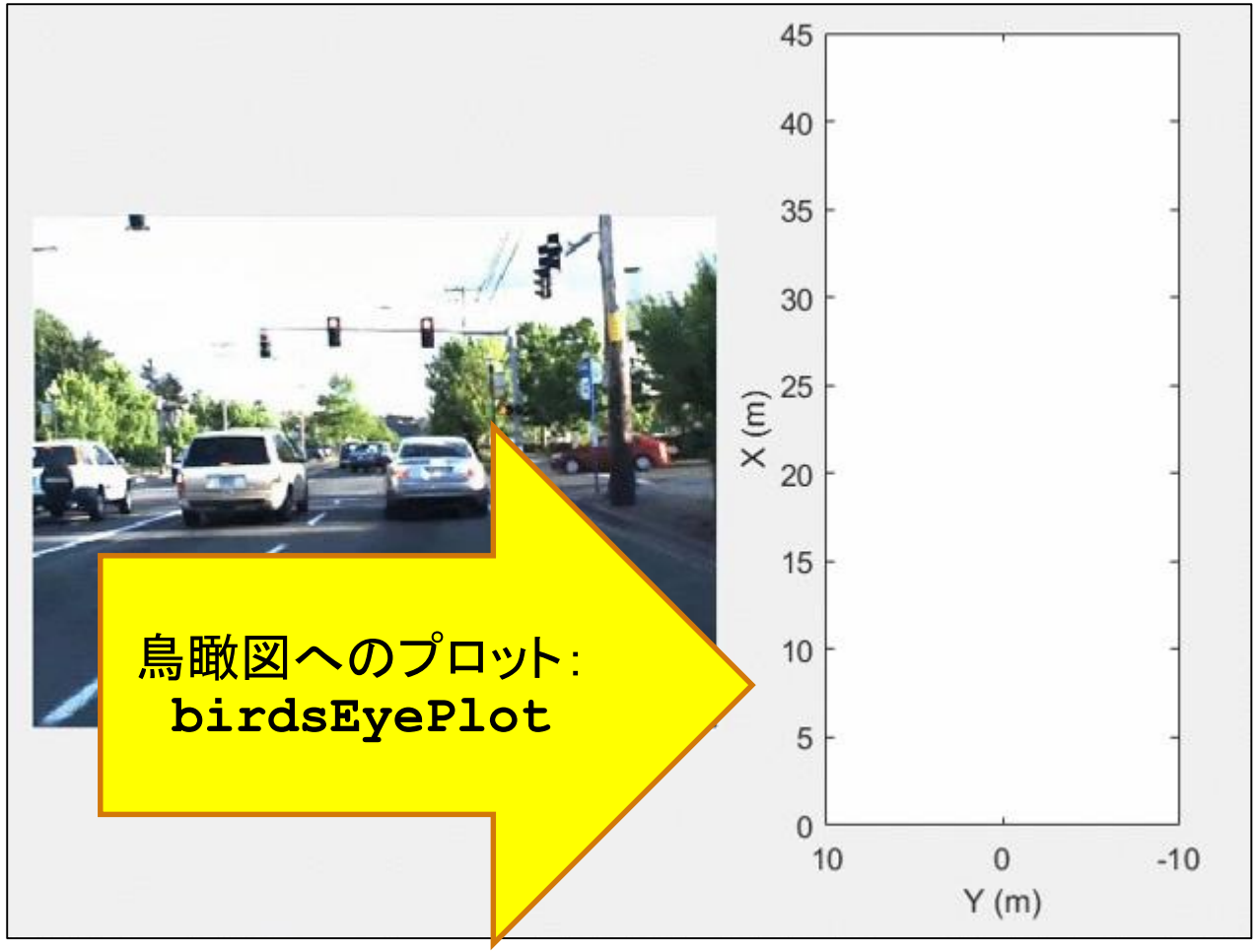
画像座標への表示
（各種画像処理用関数）:
imshow 等

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

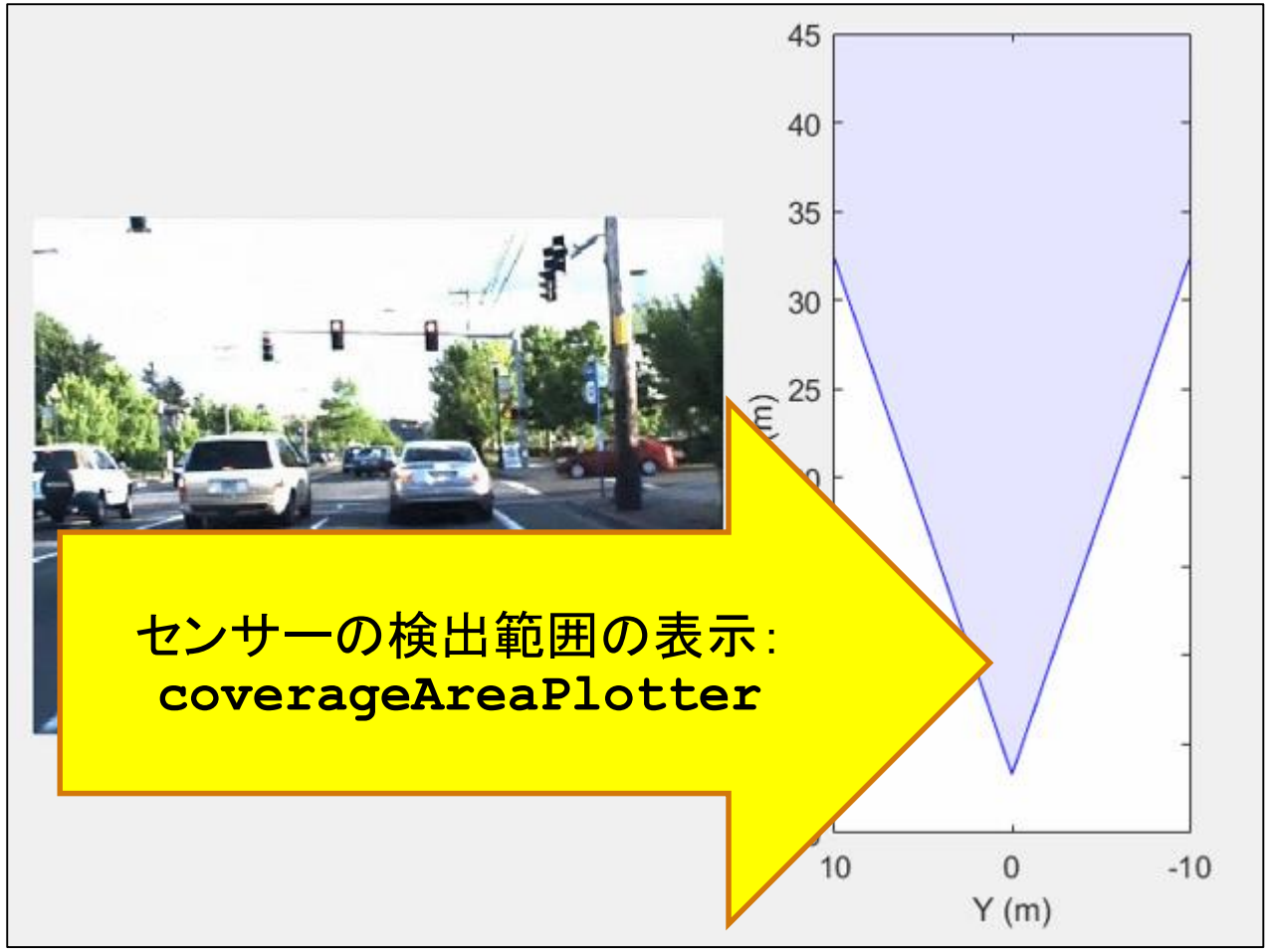



```
%% Plot in vehicle coordinates
ax2 = axes(...
    'Position',[0.6 0.12 0.4 0.85]);
bep = birdsEyePlot(...
    'Parent',ax2,...
    'Xlimits',[0 45],...
    'Ylimits',[-10 10]);
legend('off');
```



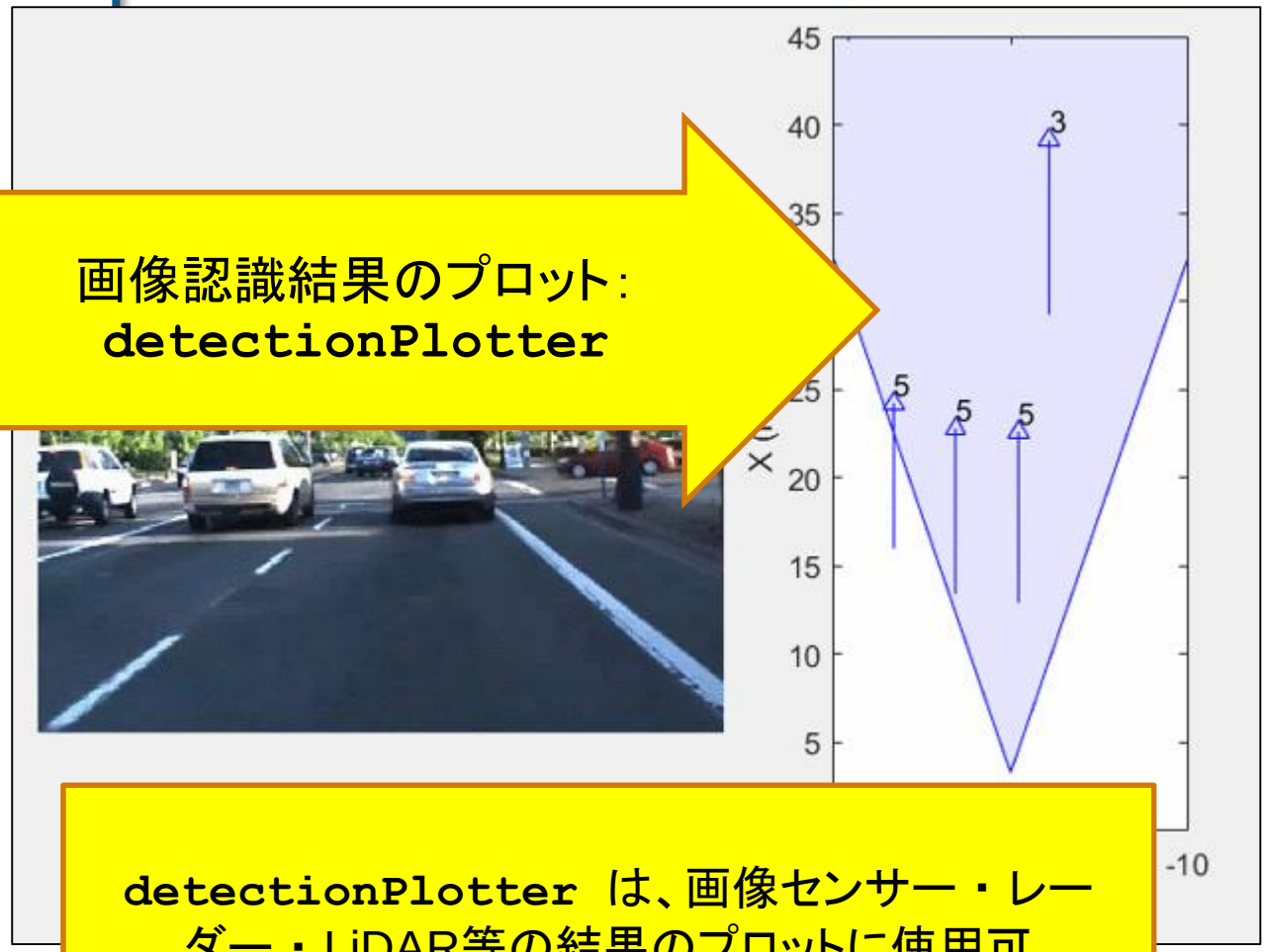
センサーの検出範囲の可視化 (鳥瞰図)

```
%% 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 (鳥瞰図)

```
%% 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);
```

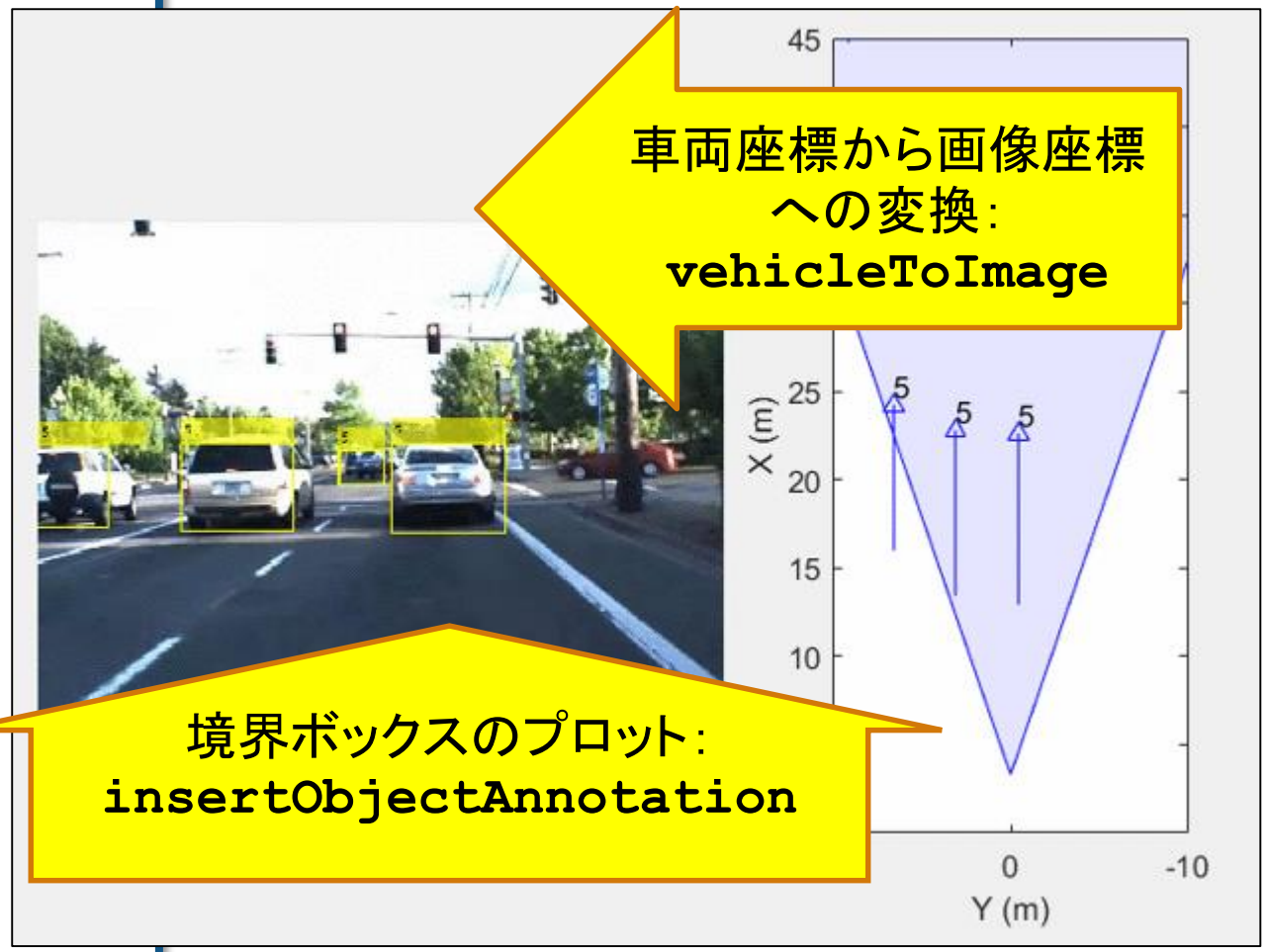


画像認識結果のプロット:
detectionPlotter

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)

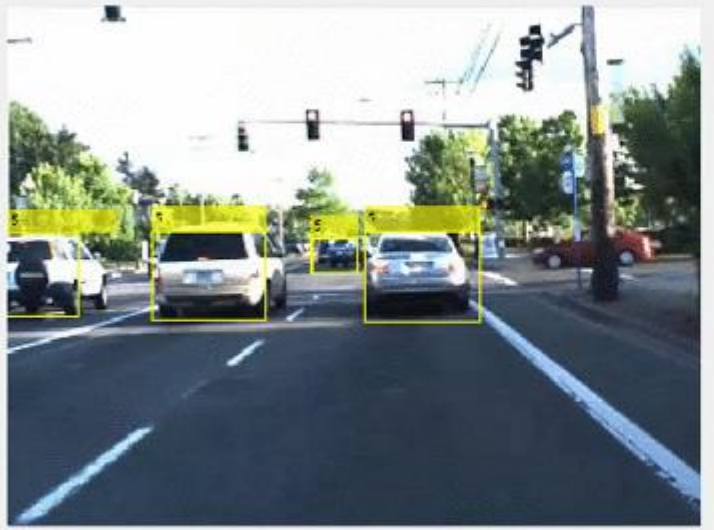
```
%% Bounding box positions in image coordinates
imBoxes = zeros(numDets,4);
for k = 1:numDets
  if vision(n).object(k).classification == 5
    vehPosLR = vision(n).object(k).position(1:2)';
    imPosLR = vehicleToImage(sensor, vehPosLR);
    boxHeight = 1.4 * 1333 / vehPosLR(1);
    boxWidth  = 1.8 * 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;
```

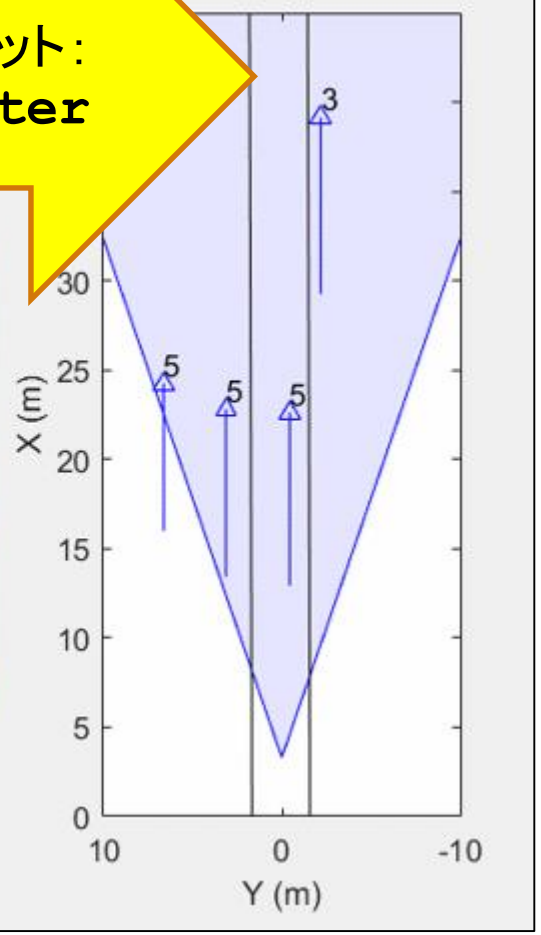


区画線の可視化 (鳥瞰図)

```
%% 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])
```

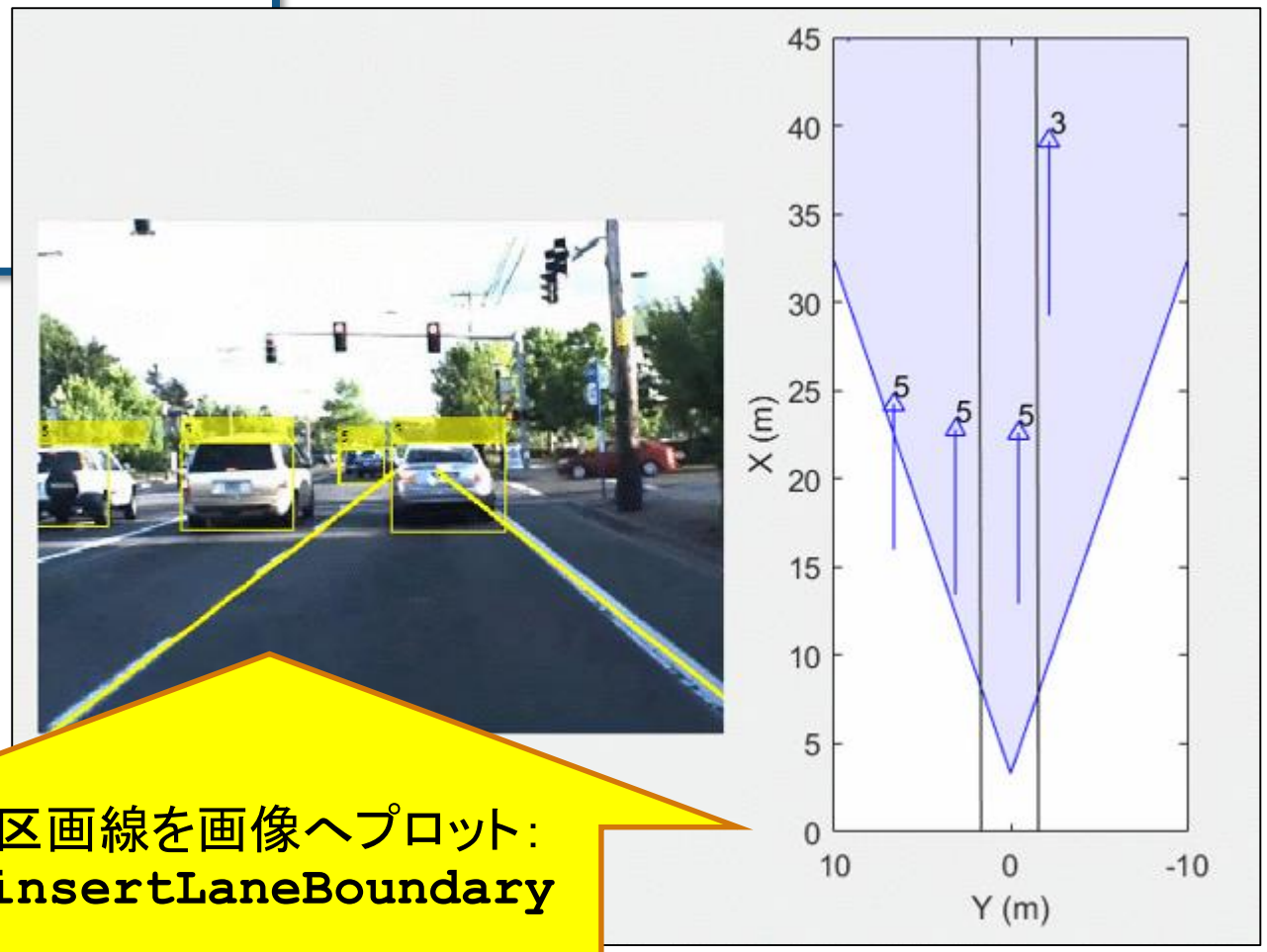
区画線を鳥瞰図へプロット:
laneBoundaryPlotter





区画線の可視化 (画像座標)

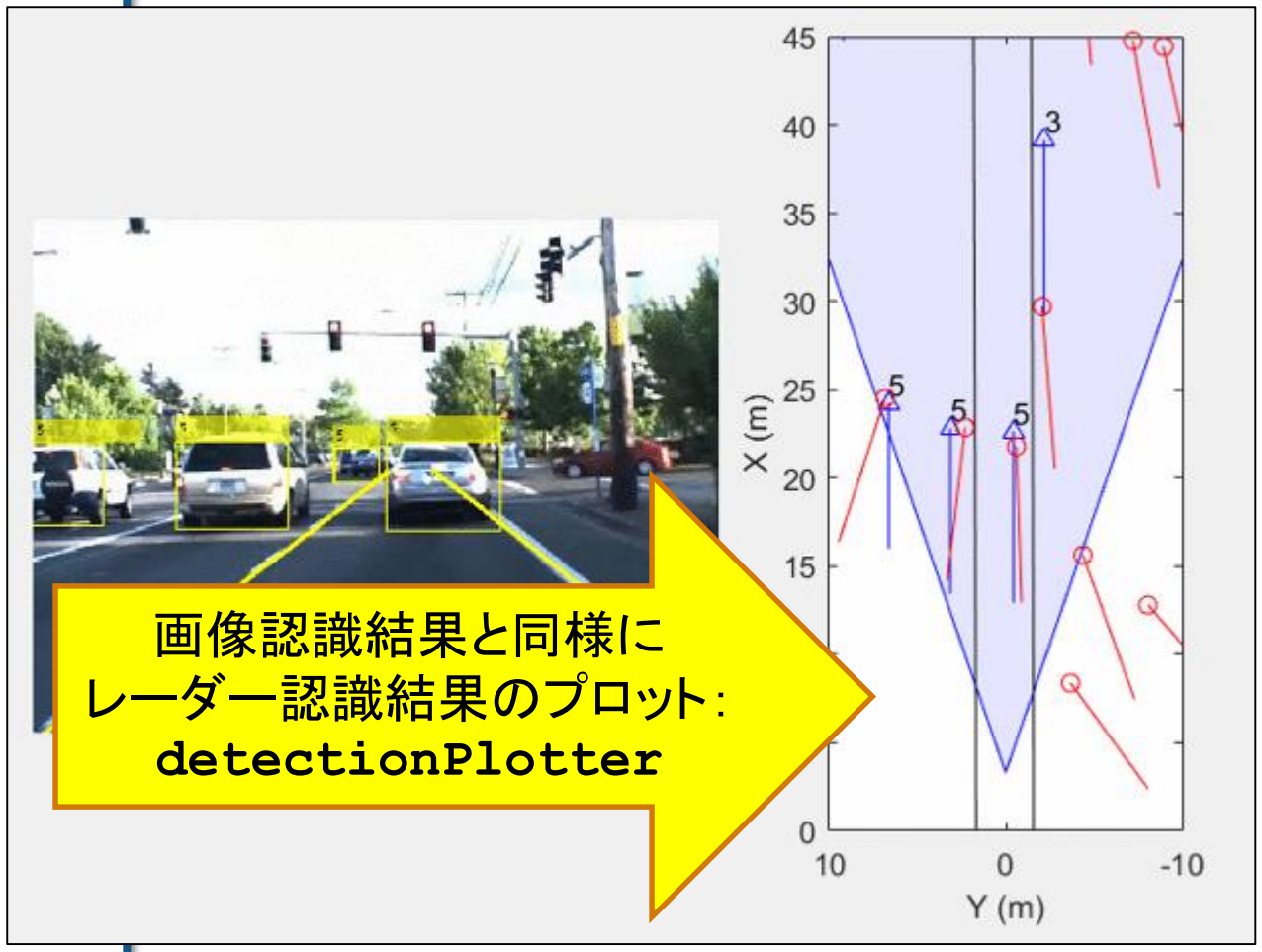
```
%% Draw in image coordinates
frame = insertLaneBoundary(frame, [lb rb], ...
    sensor,(1:100), 'Color','green', ...
    'LineWidth',7);
im.CData = frame;
```



区画線を画像へプロット:
insertLaneBoundary

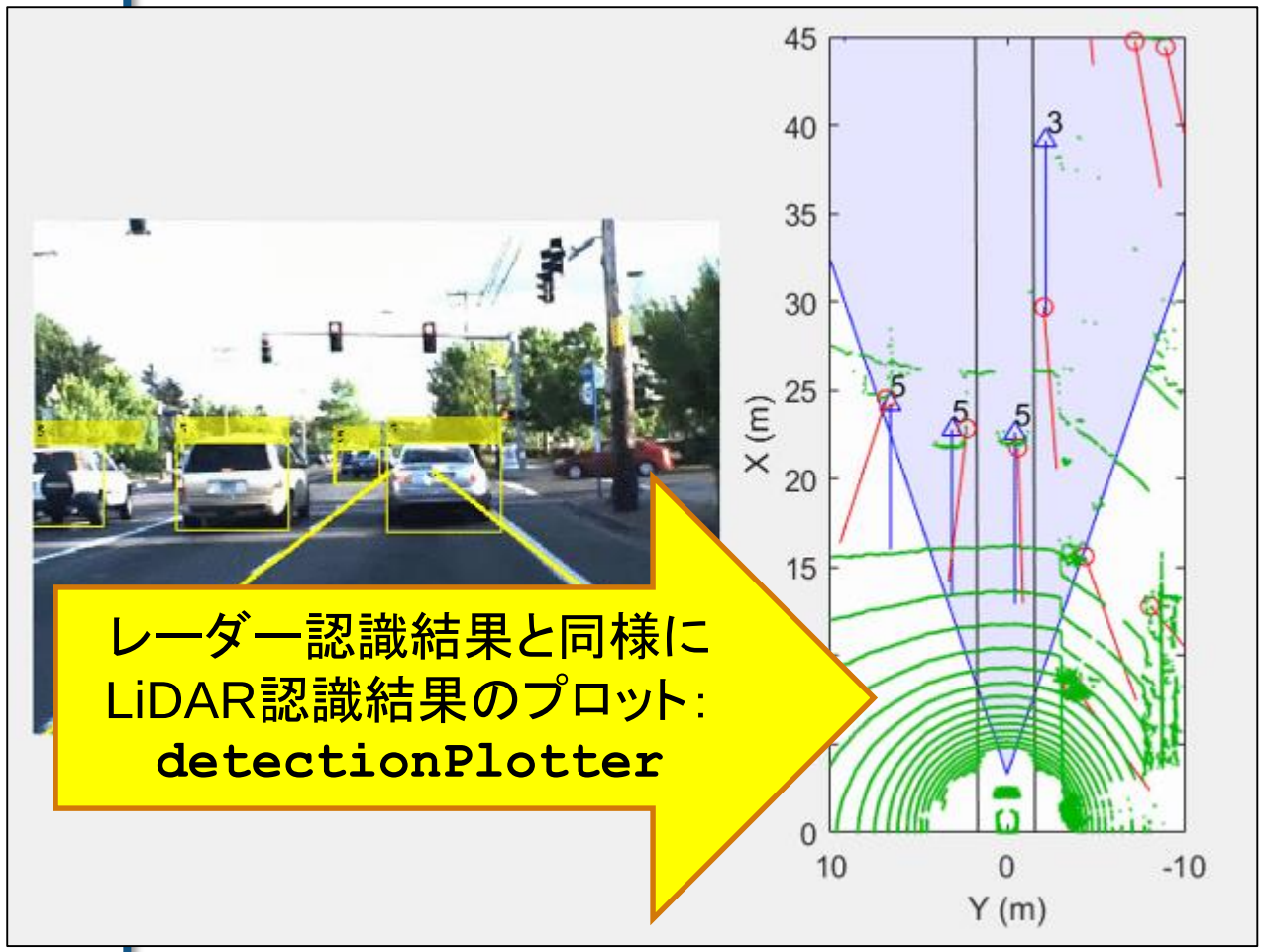
レーダーの検出結果の可視化 (鳥瞰図)

```
%% 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);
```

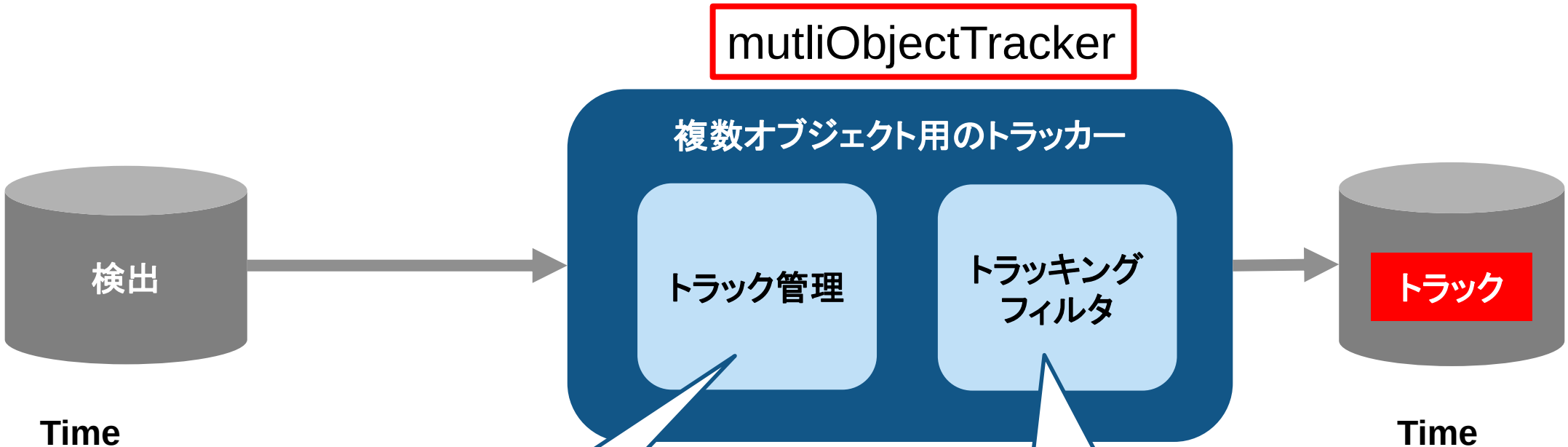


LiDARのデータ(3次元点群)のプロット (鳥瞰図)

```
%% 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);
```



複数オブジェクトのトラッキング専用の関数



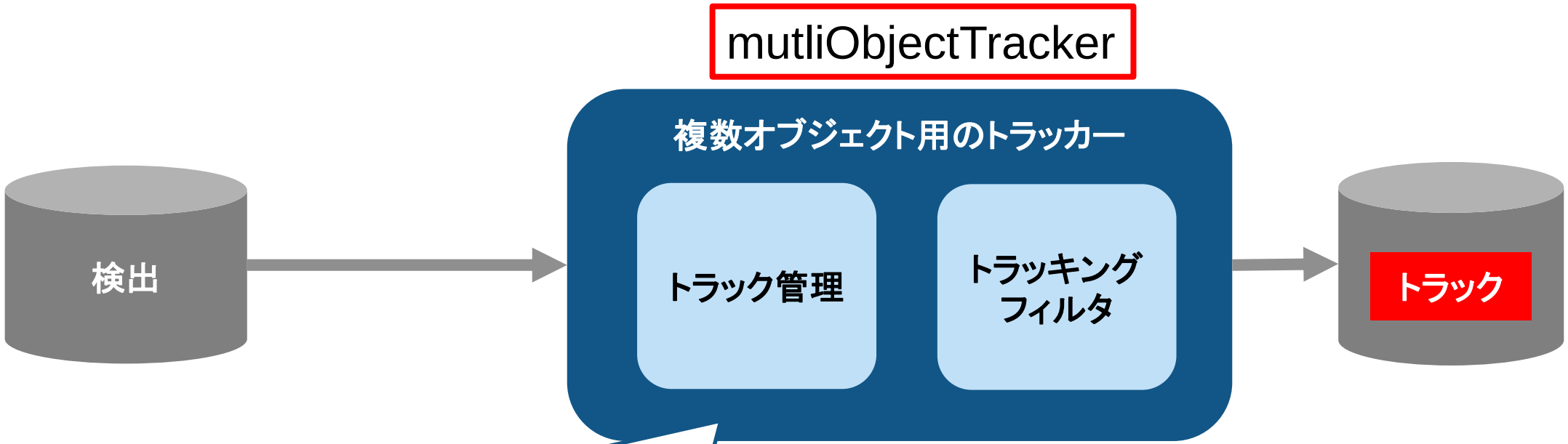
Time
Measurement
Measurement Noise

- 検出結果のトラックへの割当て
- 新しいトラックの生成
- 現有トラックの更新
- 古いトラックの削除

- トラックの状態の予測・更新
- カルマンフィルター:
  線形・拡張・unscented

Time
State
State Covariance
Track ID
Age
Is Confirmed
Is Coasted

複数オブジェクトのトラッキング設定の例




```
tracker = multiObjectTracker(...
  'FilterInitializationFcn', @initcaekf, ... % カルマンフィルタへの関数ハンドル
  'AssignmentThreshold',     35,    ...   % 割り当てに使用する、正規化した距離閾値
  'ConfirmationParameters', [2 3] ...   % 初検出後の、誤検出による削除条件(3回中2回検出必要)
  'NumCoastingUpdates',      5);         % 非検出でトラッキングする最大フレーム数(その後削除)
```

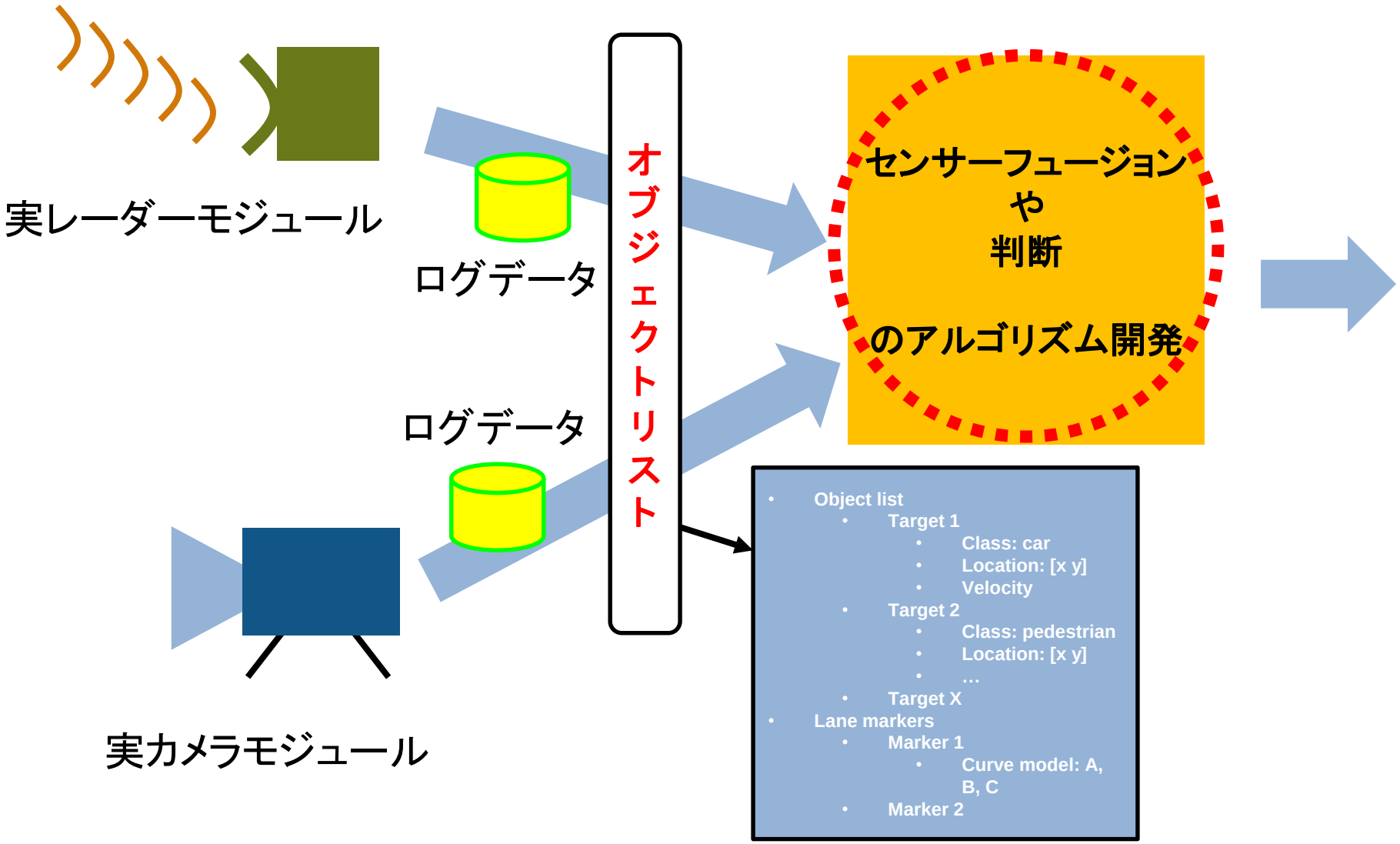
アジェンダ

- 画像処理・画像認識領域での活用
- 検出器の評価・検証
- センサーデータ可視化
- シナリオ生成

- まとめ

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

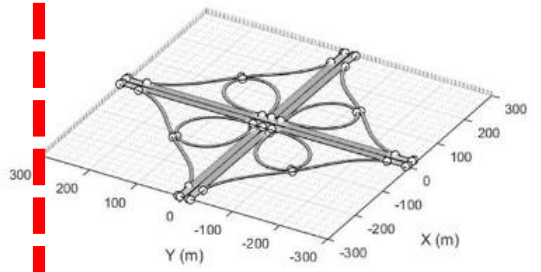
～ 多様な入力データを用いた、センサーフュージョン開発環境の提供（1）～

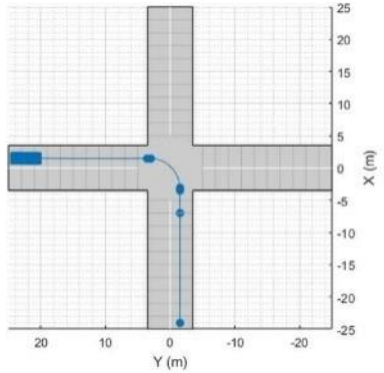


ユースケース 2　センサーフュージョン アルゴリズム 開発

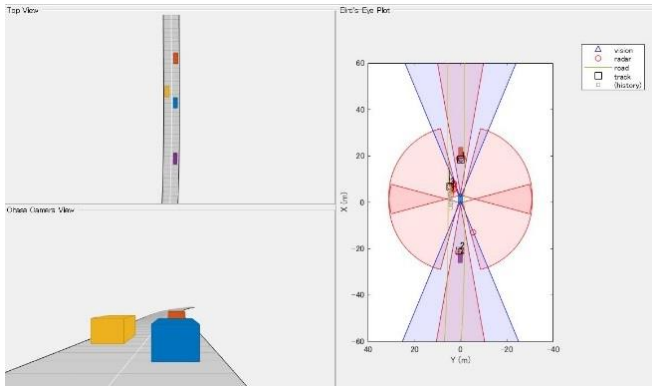
～ 多様な入力データを用いた、センサーフュージョン開発環境の提供（2）～

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

センサーフュージョン
や
判断
のアルゴリズム開発

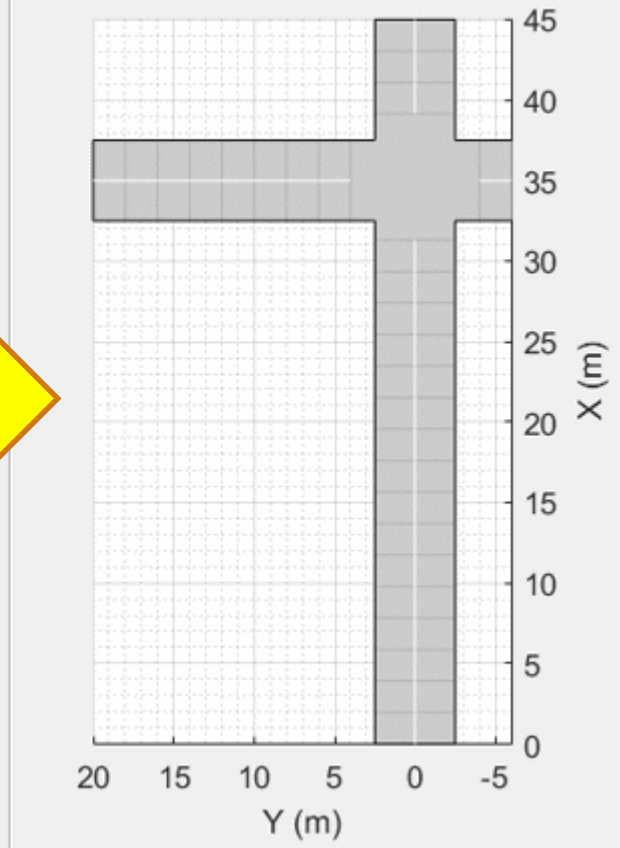
- Object list
  - Target 1
    - Class: car
    - Location: [x y]
    - Velocity
  - Target 2
    - Class: pedestrian
    - Location: [x y]
    - …
  - Target X
- Lane markers
  - Marker 1
    - Curve model: A, B, C
  - Marker 2

デモ

道路の定義

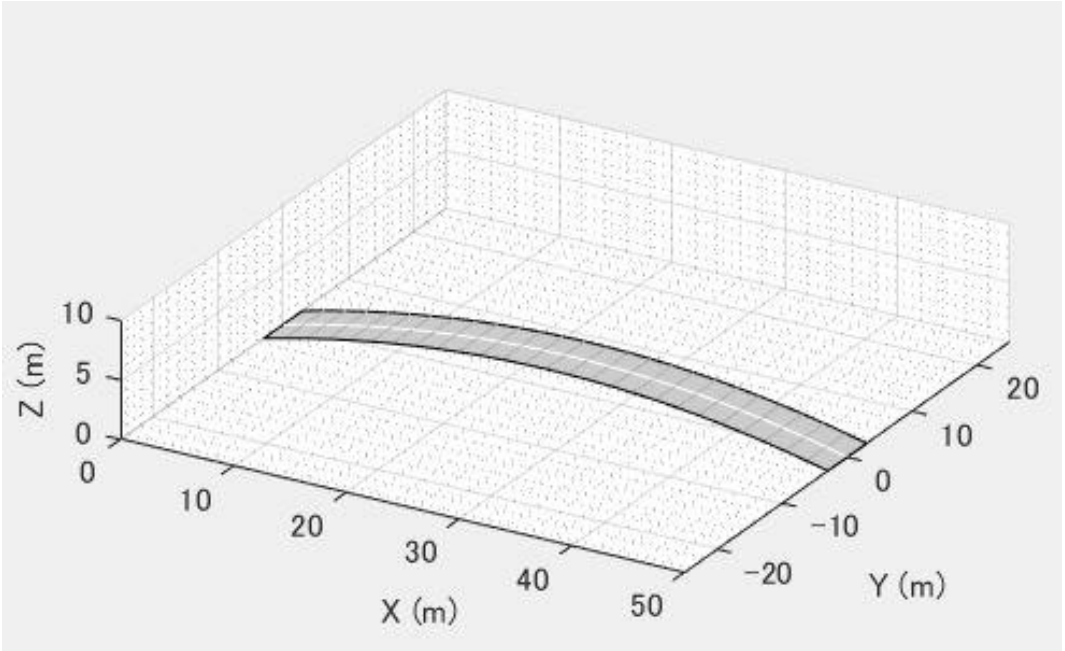
```
%% Create a new scenario
s = drivingScenario('SampleTime', 0.05);

%% 道の定義
road(s, [ 0   0; ...      % 路端座標1 [x,y] (m)
          45  0],...      % 路端座標2 [x,y] (m)
          5);             % 道幅 (m)
road(s, [35  20; ...
          35 -10],...
          5);

%% 表示
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

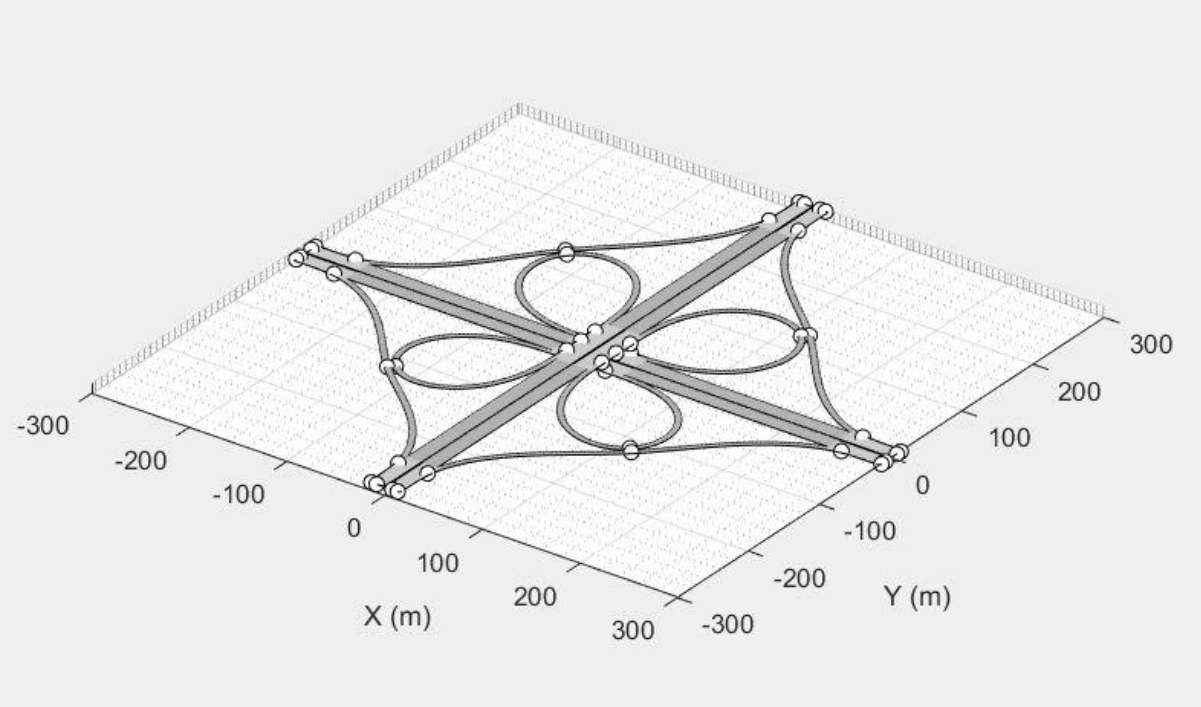
```
%% Create a new scenario
s = drivingScenario;

%% 道の定義
road(s, [ 0   0   0; ...   % Centers [x,y,z] (m)
          25  0   3; ...
          50  0   0]);

%% 表示
plot(s,'Centerline','on','Waypoints','on')
view(30,24);
```



z座標を定義することで、3次元構造も生成可能

道路の定義

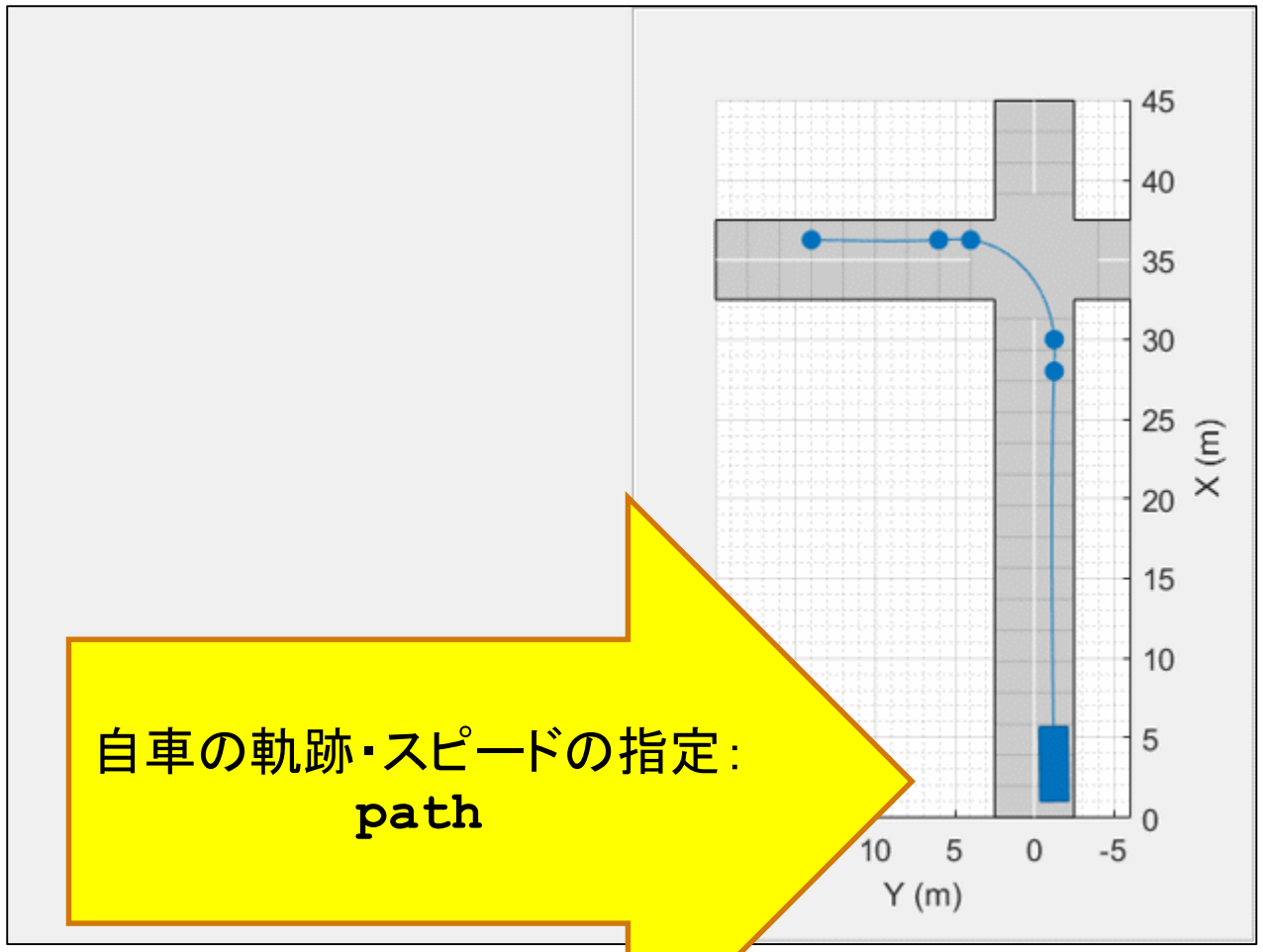
```
s = drivingScenario;

% 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);
```



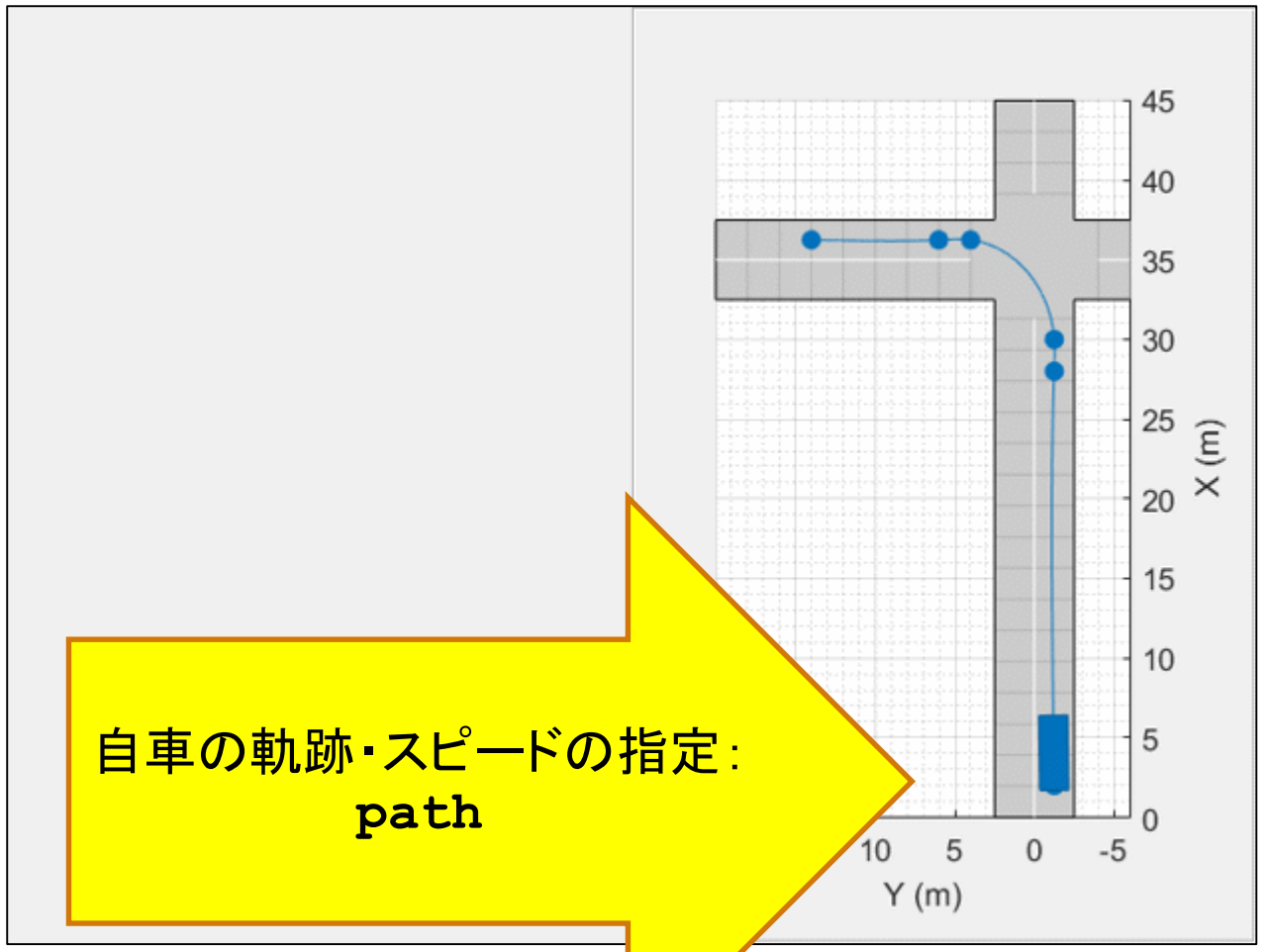
自車の軌跡・スピードの指定：
path

後車軸の中心が、車両の位置・回転中心
速度は、スカラ定数もしくは、waypointごとにベクトルで指定（waypoint間を直線補完）

シナリオ再生

アニメーション

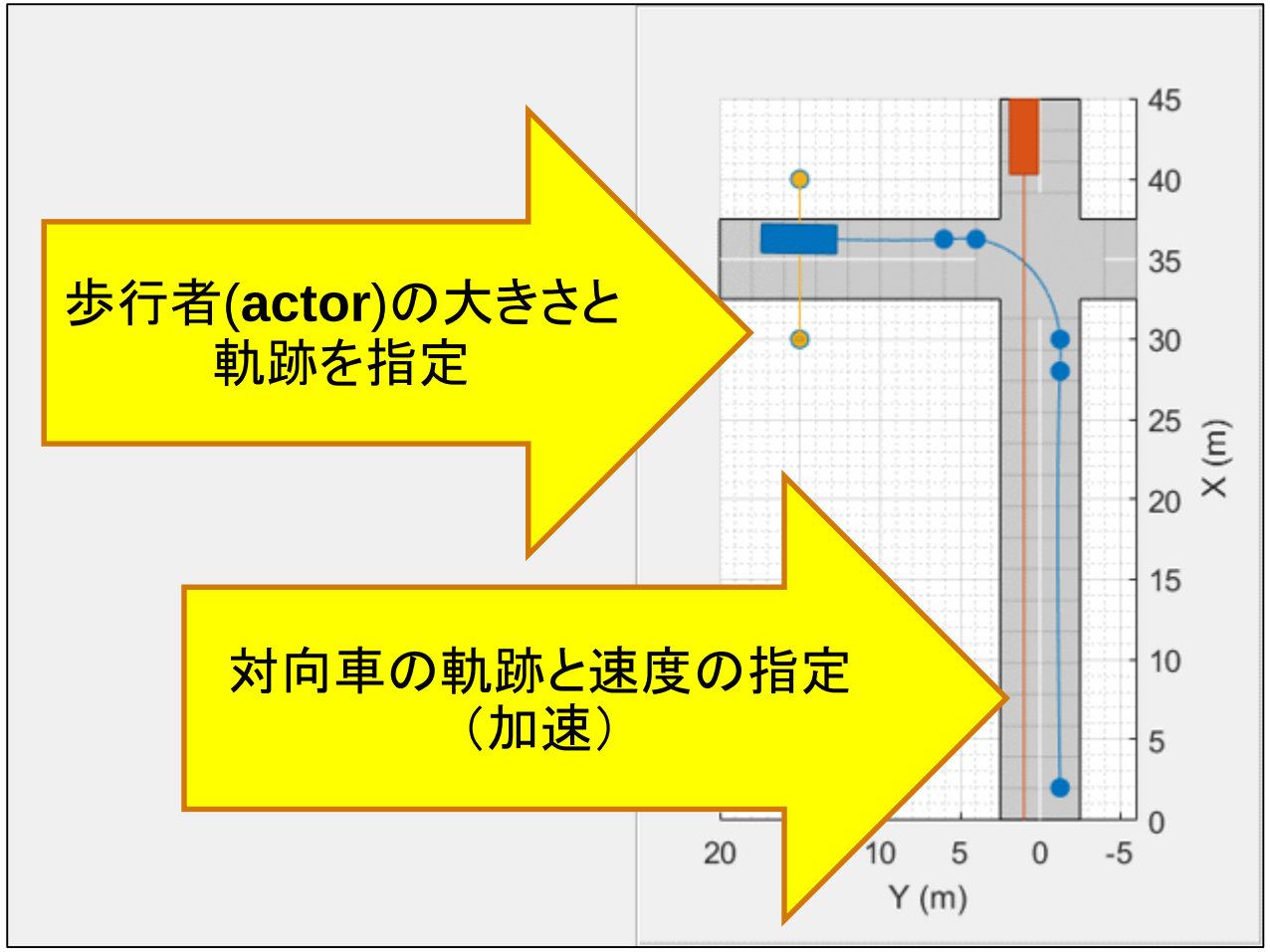
```
%% Add ego vehicle
egoCar = vehicle(s);
waypoints = [ 2   -1.25;... % [x y] (m)
              28  -1.25;...
              30  -1.25;...
              36.25   4;...
              36.25   6;...
              36.25  14];
speed = 13.89; % (m/s) = 50 km/hr
path(egoCar, waypoints, speed);

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



自車の軌跡・スピードの指定：
path

対向車と歩行者の指定

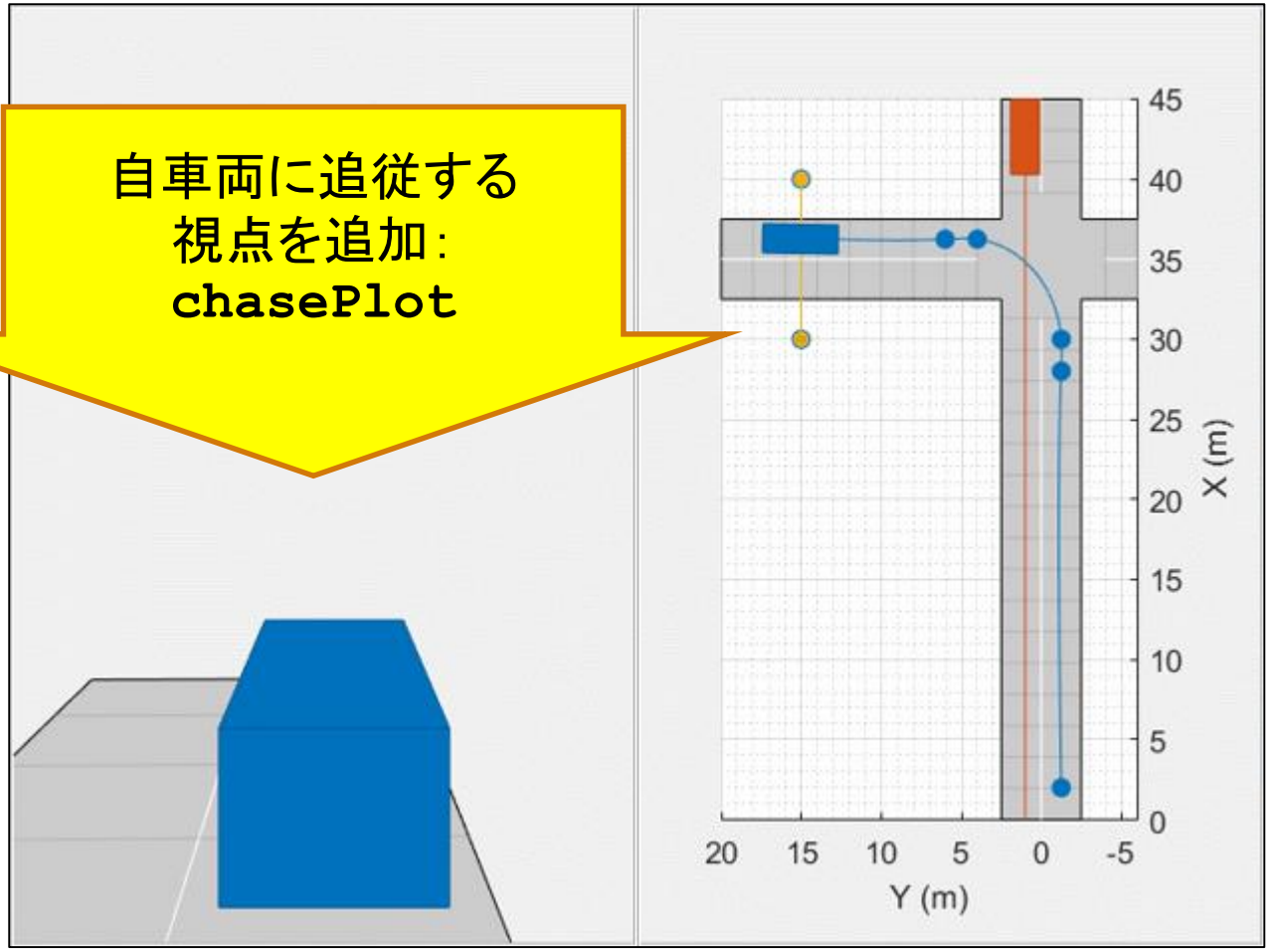
```
%% Add Target vehicle
targetVehicle = vehicle(s);
path(targetVehicle,...
      [44 1; -4 1],... % Waypoints (m)
      [5  ; 14]);      % Speeds (m/s)


%% 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)
```

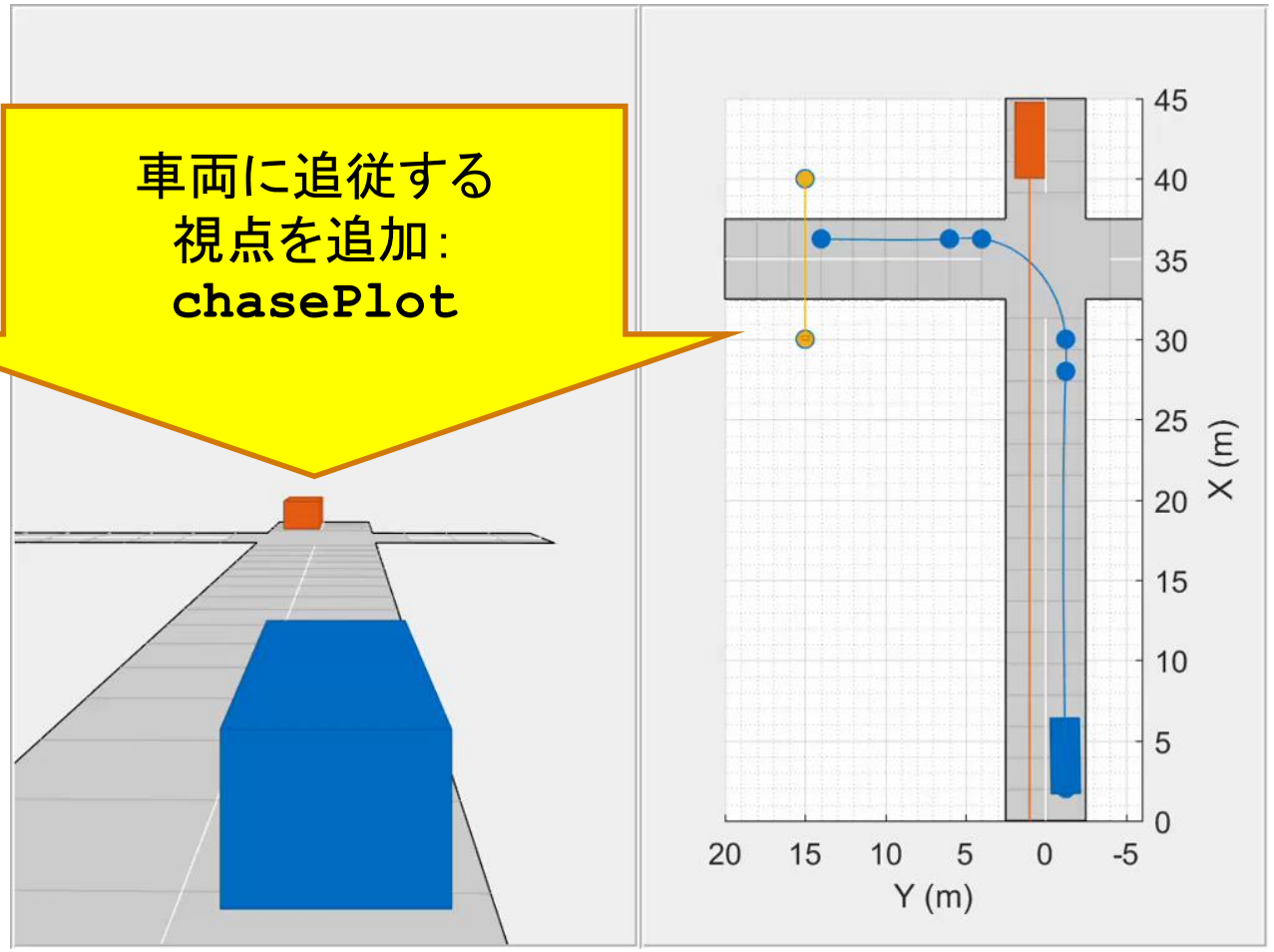
自車両に追従する
視点を追加:
chasePlot



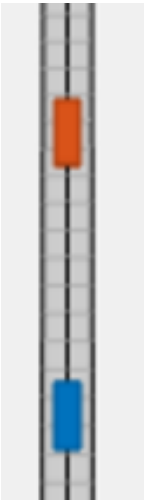
後方から車両に追従する視点の追加

```
%% 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
```

車両に追従する
視点を追加:
chasePlot

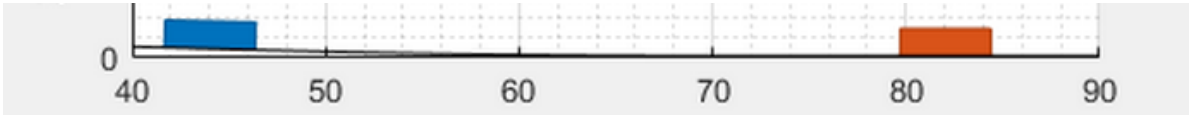


センサーの特性：カメラモジュール（画像センサー）の場合



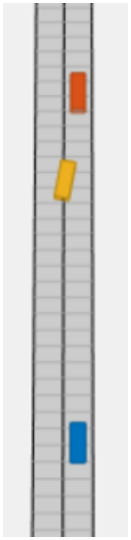
距離への効果

- 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

- 部分的に隠れると、認識しにくい

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出来ます

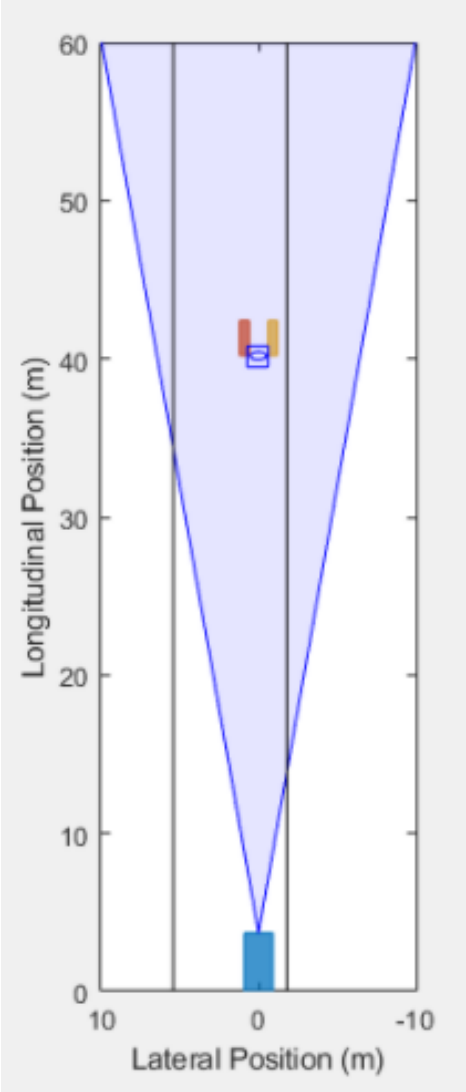
センサーの特性：レーダーモジュールの場合

距離への効果

- 検出は、物体の大きさとRCSに依存
- 近距離に場合、一つの物体から複数の検出が生成される
- 遠距離の場合、複数の物体から一つしか検出されない
- 横方向の角度の精度は、距離と共に悪くなる

速度への効果

- 縦方向の速度の精度は、距離に依存せず高い
- 横方向の速度の精度は、距離が離れると悪くなる



センサーモデル：レーダー、カメラ

カメラモデルのプロパティー（visionDetectonGenerator）

```
      SensorIndex: 1
   UpdateInterval: 0.1000

SensorLocation: [1.9000 0]
        Height: 1.1000
           Yaw: 0
         Pitch: 1
          Roll: 0
    Intrinsics: [1×1 cameraIntrinsics]

   FieldOfView: [43.6028 33.3985]
      MaxRange: 150
      MaxSpeed: 50
MaxAllowedOcclusion: 0.5000
MinObjectImageSize: [15 15]

DetectionProbability: 0.9000
FalsePositivesPerImage: 0.1000

BoundingBoxAccuracy: 5
ProcessNoiseIntensity: 5

        HasNoise: true
MaxNumDetectionsSource: 'Auto'
DetectionCoordinates: 'Ego Cartesian'

  ActorProfiles: [1×1 struct]
```

センサのID番号

測定周期

センサの車両座標位置(x,y)

タイヤの面からの高さ

車両座標系に対するカメラの角度

縦方向・横方向のFOV(°)（intrinsicから自動計算）

レーダモデルのプロパティー（radarDetectonGenerator）

```
      SensorIndex: 1
   UpdateInterval: 0.1000

SensorLocation: [3.4000 0]
        Height: 0.2000
           Yaw: 0
         Pitch: 0
          Roll: 0

   FieldOfView: [20 5]
      MaxRange: 150
RangeRateLimits: [-100 100]

DetectionProbability: 0.9000
FalseAlarmRate: 1.0000e-06
ReferenceRange: 100
ReferenceRCS: 0

AzimuthResolution: 4
RangeResolution: 2.5000
RangeRateResolution: 0.5000

AzimuthBiasFraction: 0.1000
RangeBiasFraction: 0.0500
RangeRateBiasFraction: 0.0500

        HasElevation: false
        HasRangeRate: true
           HasNoise: true
       HasFalseAlarms: true
MaxNumDetectionsSource: 'Auto'
DetectionCoordinates: 'Ego Cartesian'

  ActorProfiles: [1×1 struct]
```

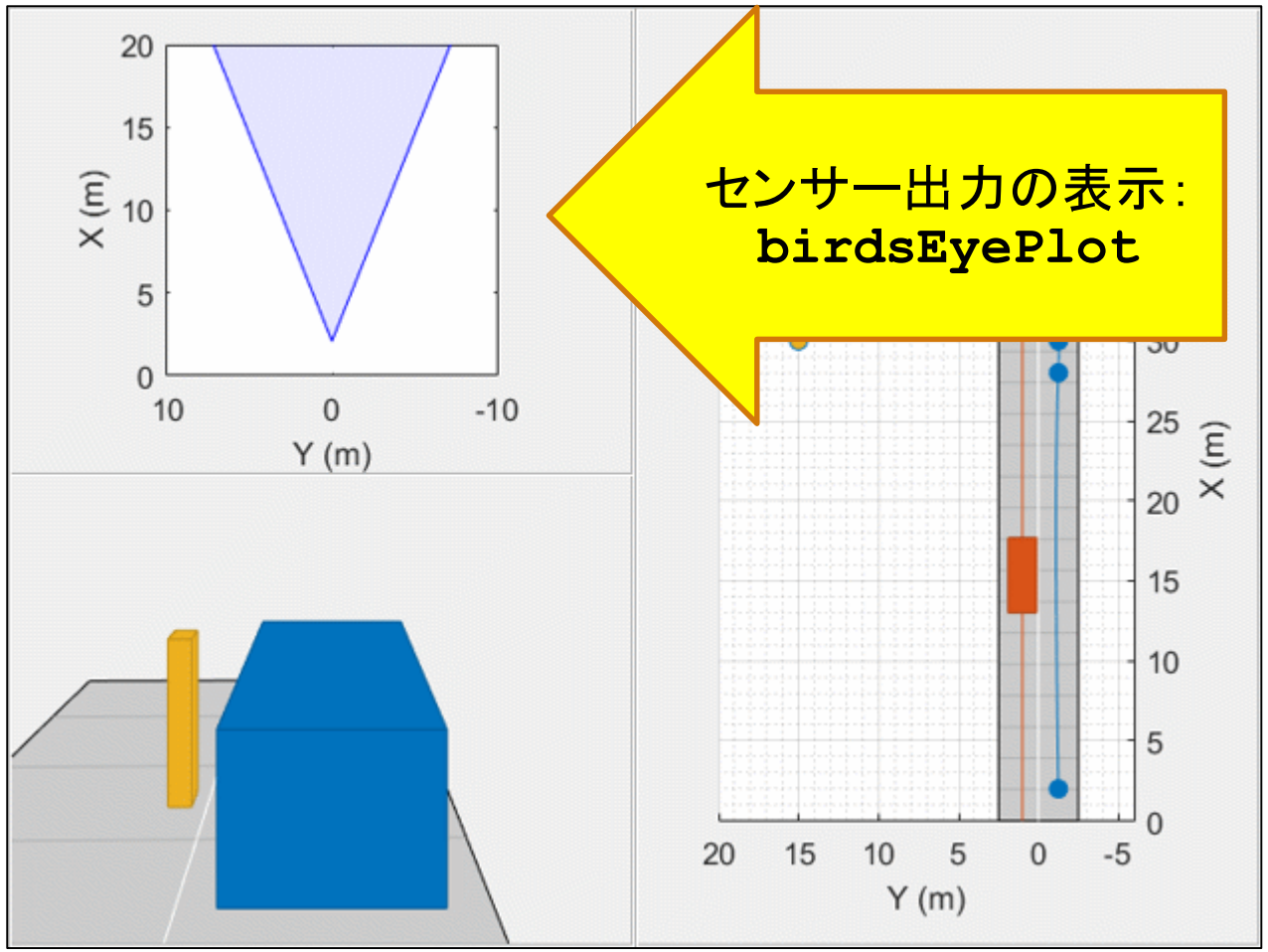
画像センサーの定義

```
%% 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));
```

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

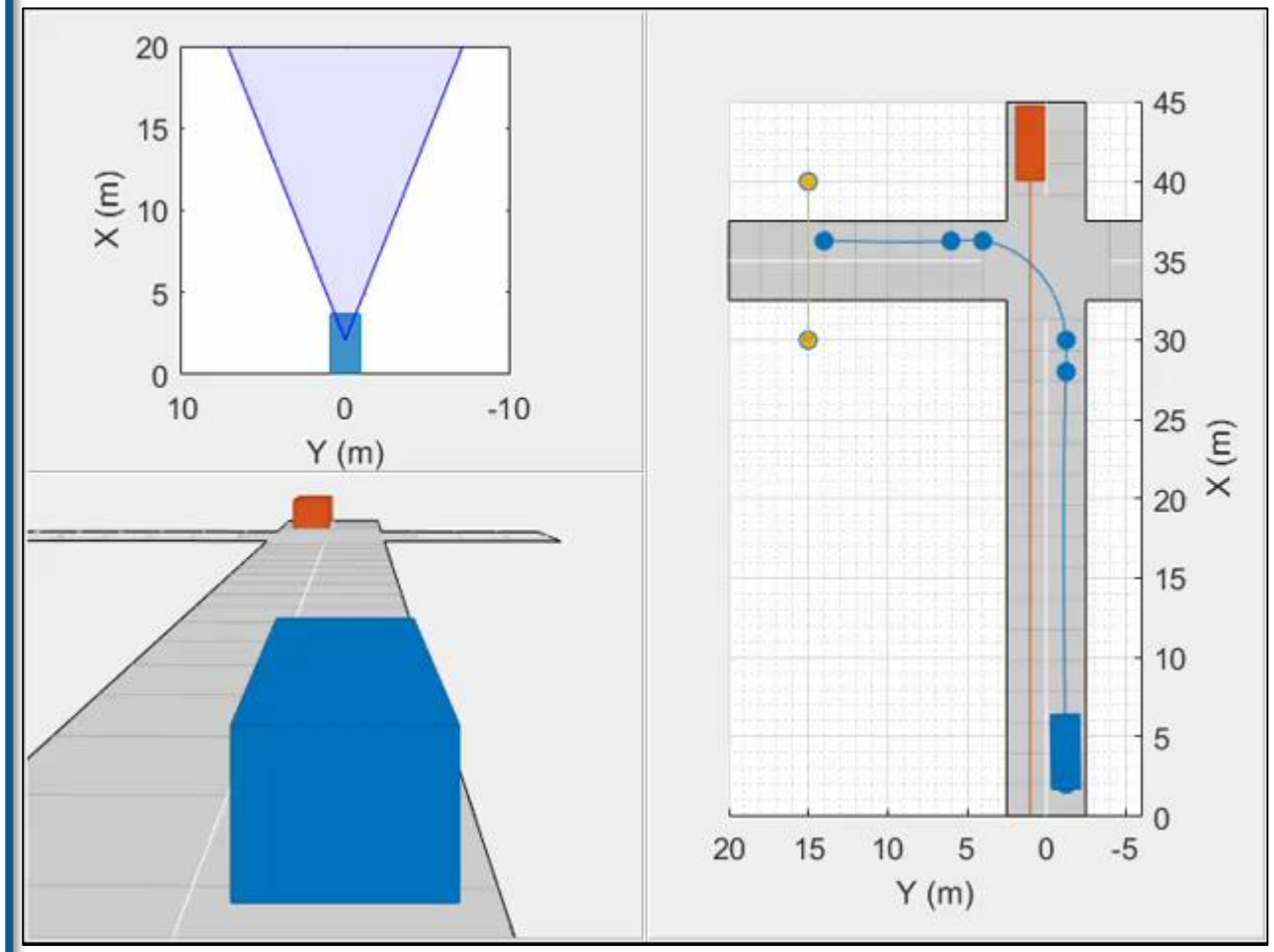
センサー出力の表示用に、鳥瞰図を生成

```
%% 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); %アクター表示用
```

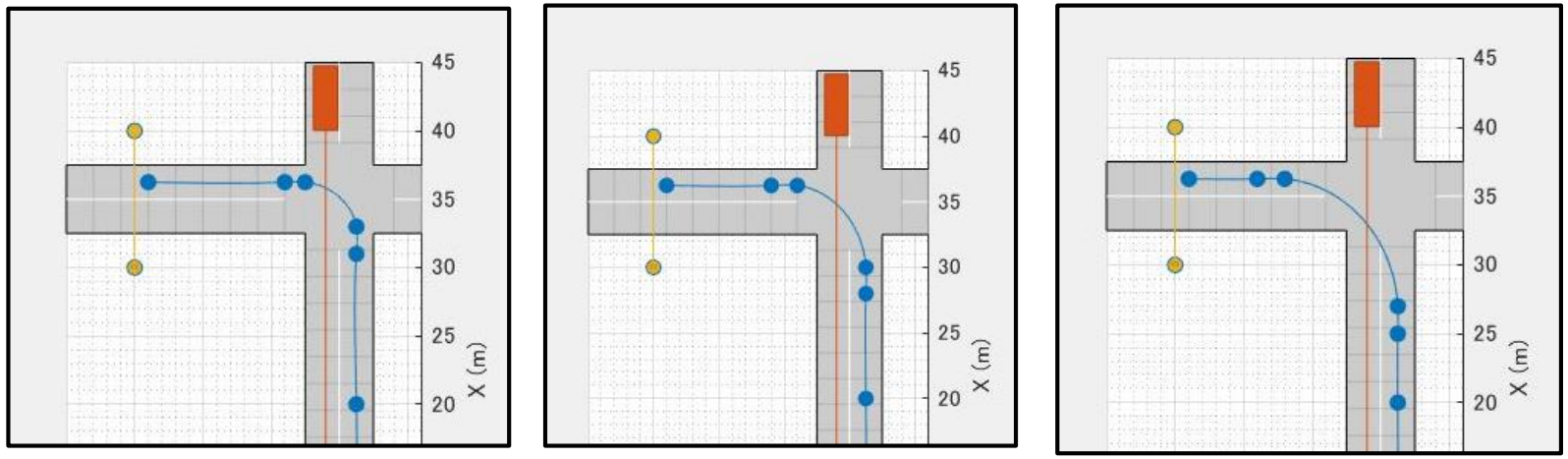


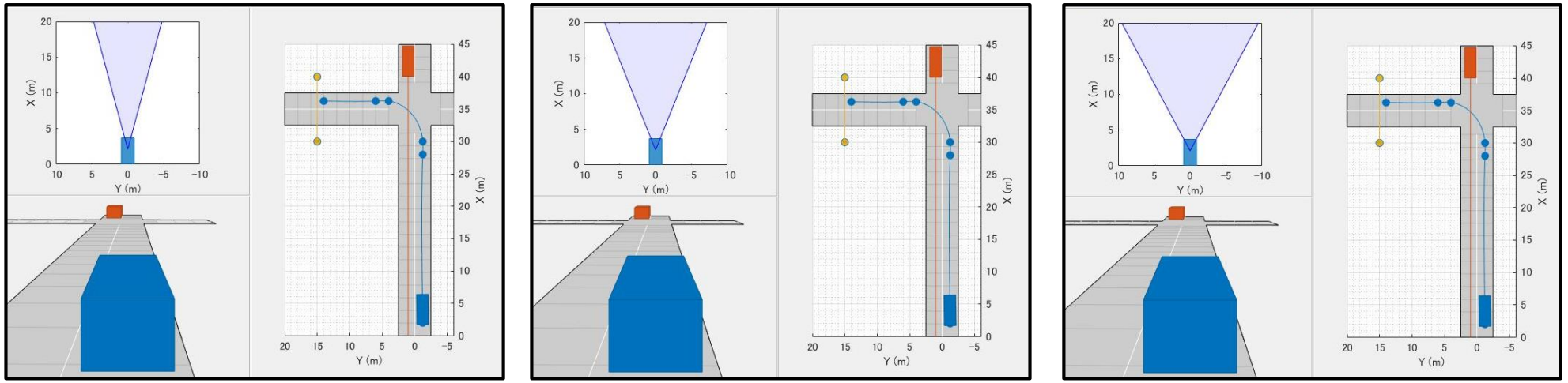
センサーモデルと共にシミュレーション

```
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
```



スクリプトでシナリオを自動生成

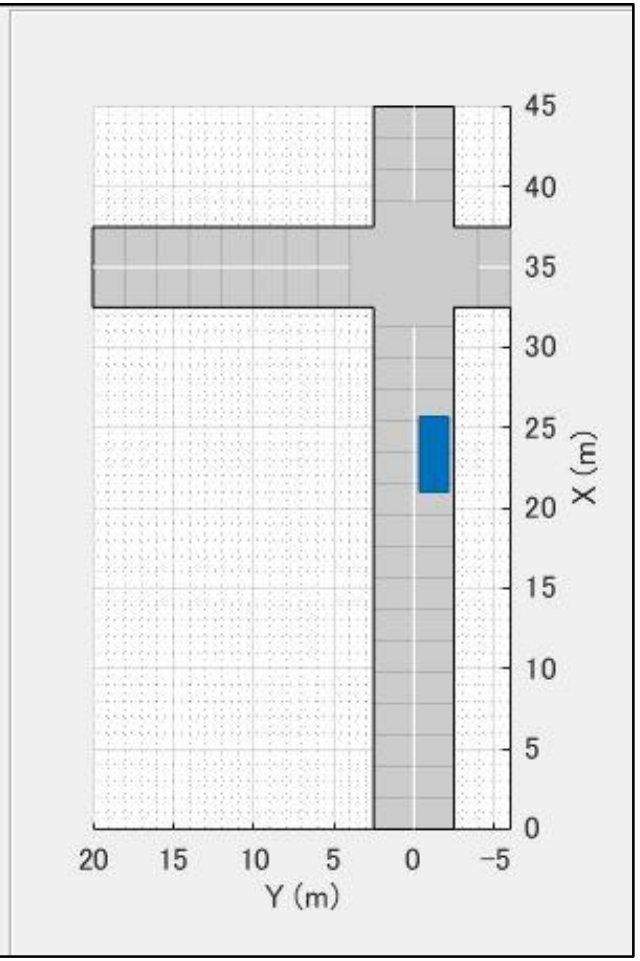




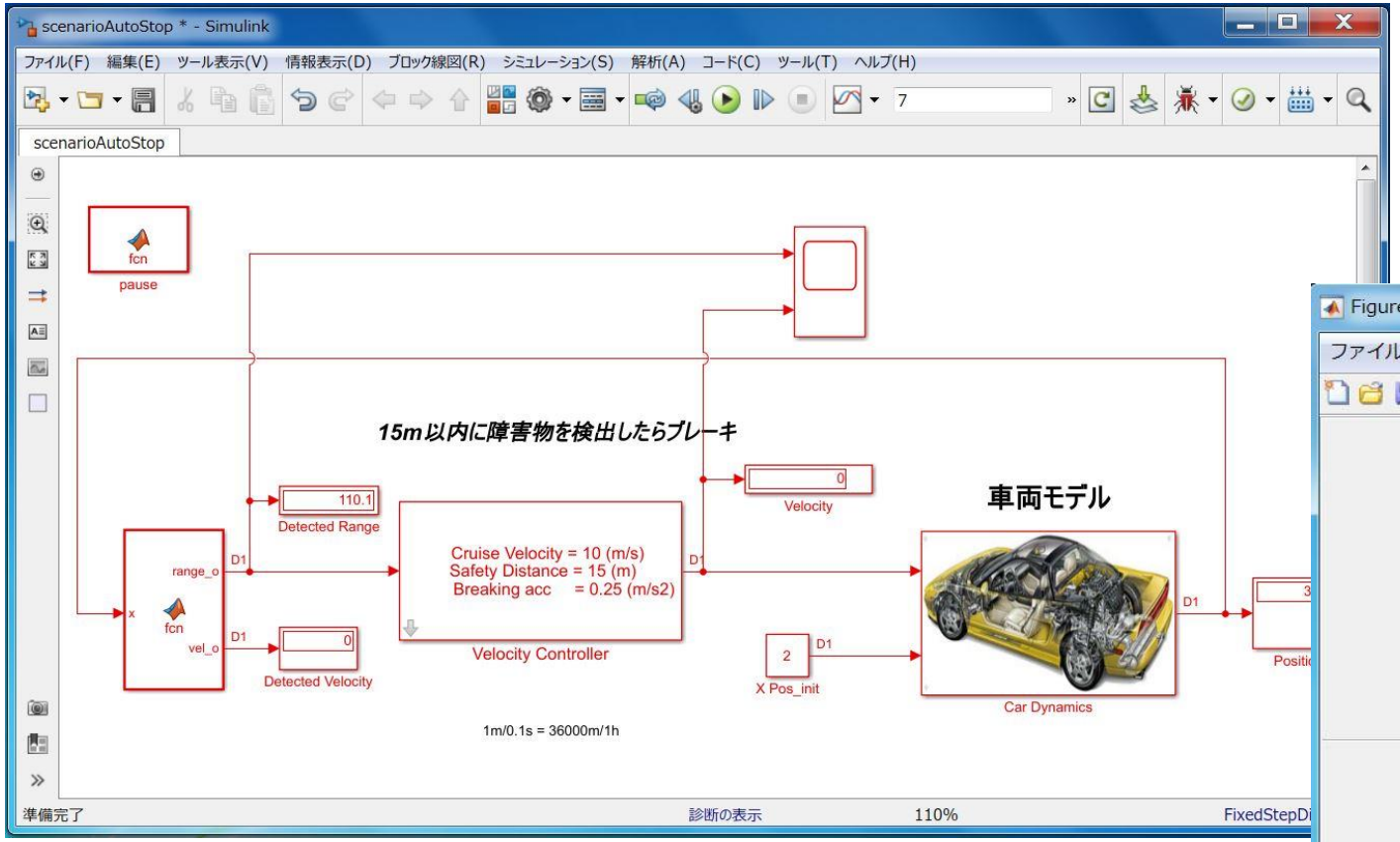
スクリプトによるフレキシブルなシナリオバリエーションの自動生成

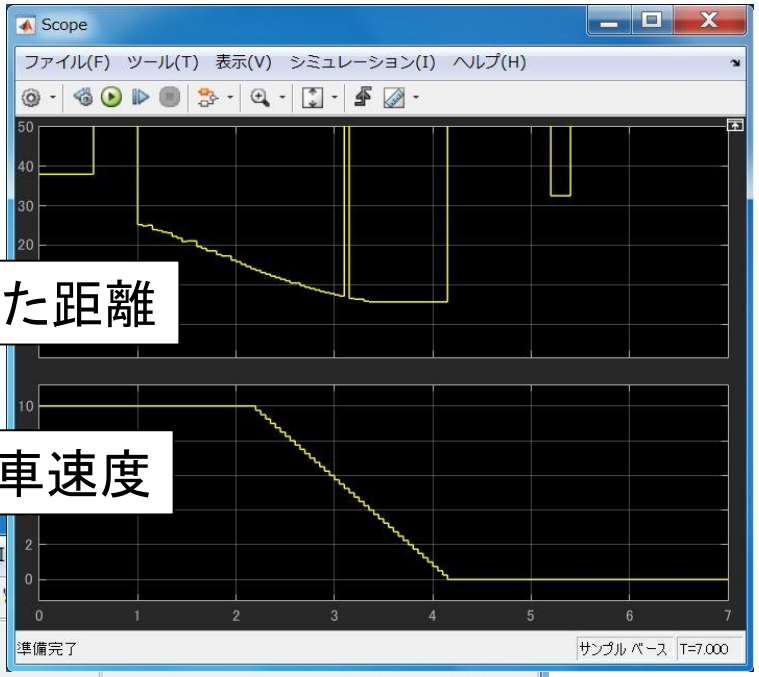
自車の定義（座標指定して移動の場合：自動運転制御結果の可視化）

```
%% Add ego vehicle
egoCar = vehicle(s);
egoCar. Position = [2 -1.25 0];   % 初期位置

%% Play scenario
for advance(s)
  % タイムステップごとに0.5m前進
  egoCar.Position = egoCar.Position + [0.5 0 0];

  pause(s.SampleTime);
end
```



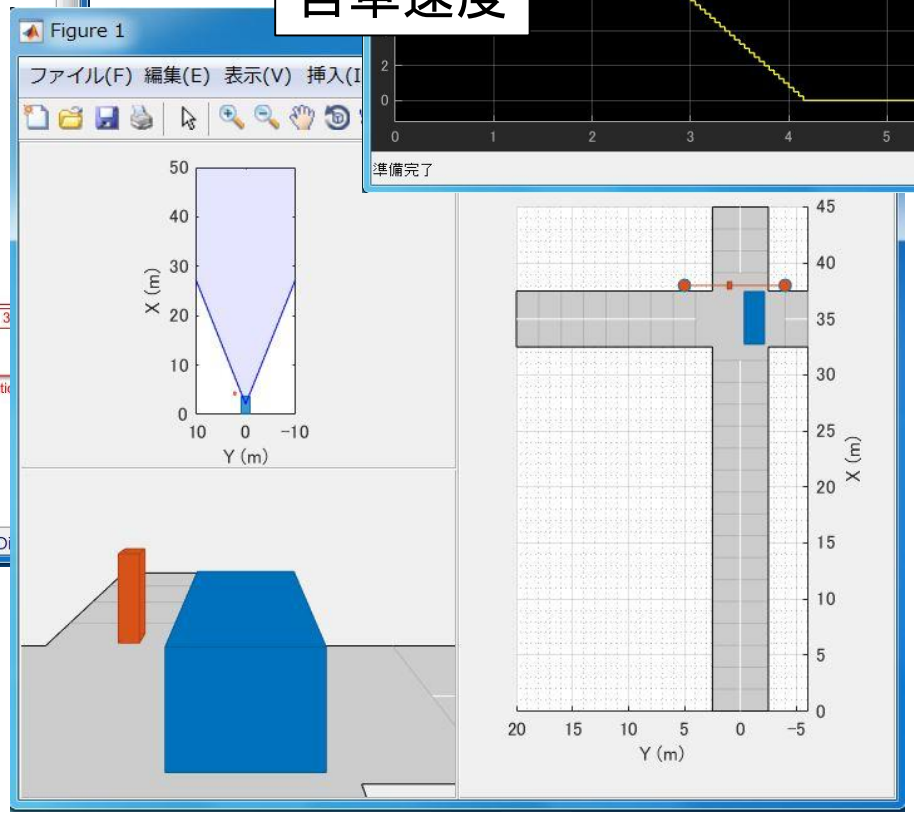
自動運転制御結果の可視化：Simulink デモ





検出した距離

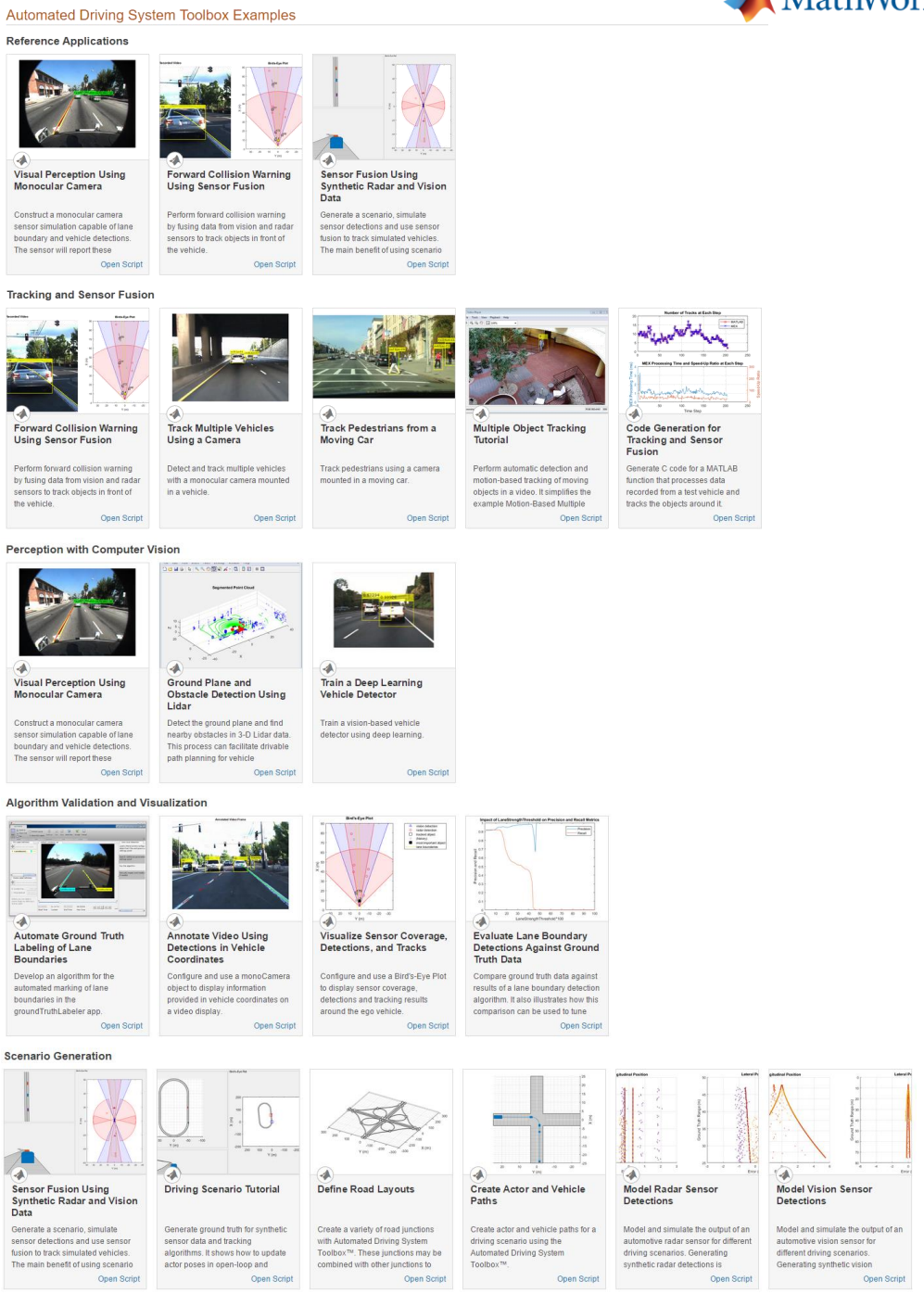
自車速度



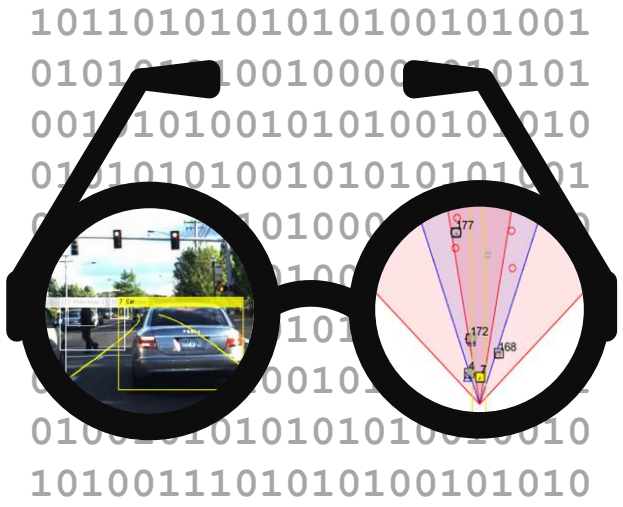
Simulinkの制御モデルとのシームレスな連携

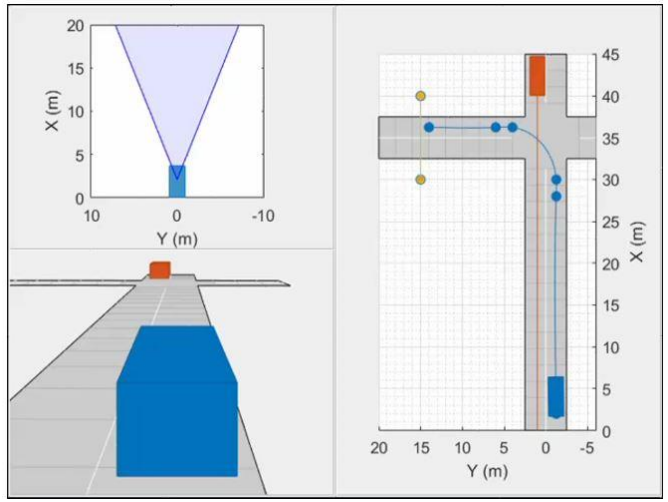
多くのサンプルプログラムを同梱

- ADAS/自動運転 開発に関連する機能をサンプルとして提供
  - スターティングポイントとして
  - 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





認識の開発･検証

センサーデータの可視化

シナリオ生成

- カスタマイズ容易な多くの部品で提供
- MATLAB ® / Simulink プラットフォーム上でシームレスに動く開発環境 : 既存資産･各種オプション製品との連携

Automated Driving System Toolbox の必要オプション構成

必須: Computer Vision System Toolbox
Image Processing Toolbox（Computer Vision System Toolboxの前提製品）

[深層学習ベースの車検出器を使用する場合]
必須: Neural Network Toolbox
強く推奨: Parallel Computing Toolbox
Compute Capability3.0以上のCUDA GPU

概要紹介Webinar
https://jp.mathworks.com/videos/development-and-verification-platform-for-automated-driving-and-adas-automated-driving-system-toolbox-1498054419919.html

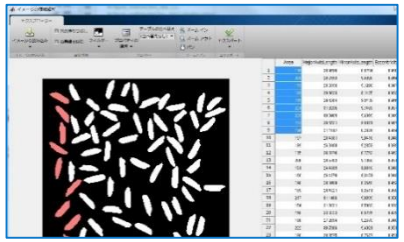
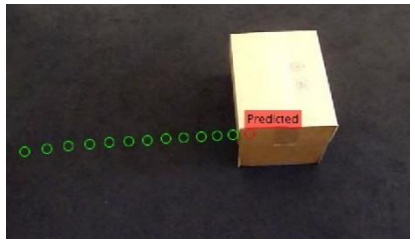
画像処理・コンピュータービジョン無料セミナー

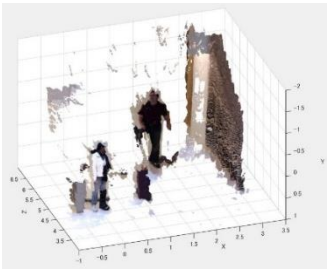
申し込みは弊社ウェブサイトより（メールアドレスを入力後、送信ボタン）

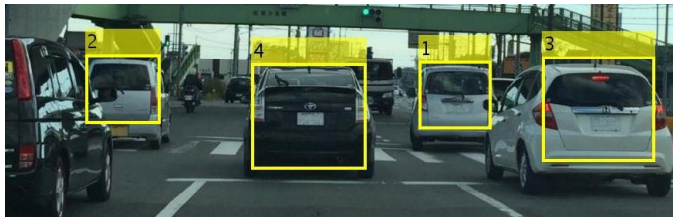
https://jp.mathworks.com/company/events/seminars/ipcv-tokyo-2258970.html

具体例で分かる！MATLABによる画像処理・コンピュータビジョン・機械学習

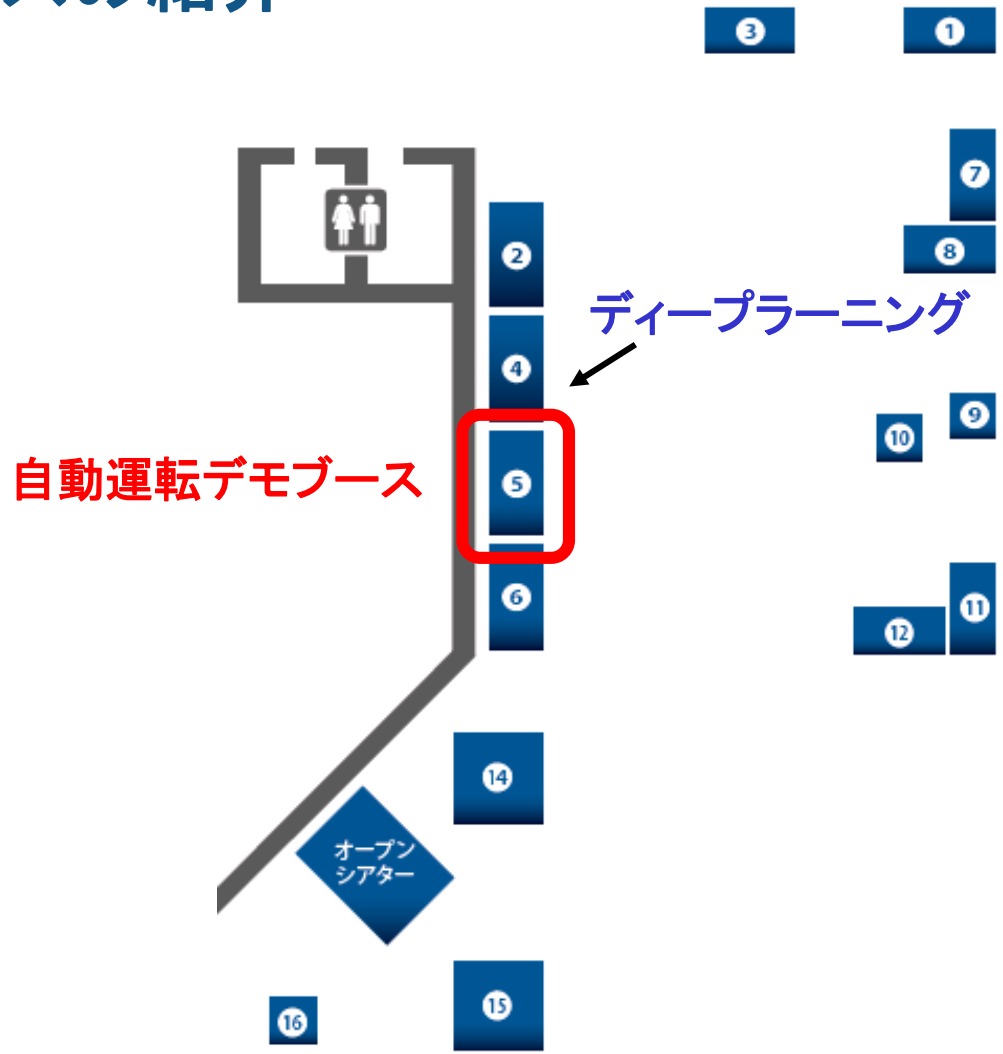
- 日時：2017年11月21日　13:30–17:00（受付　13:00–）
- 場所：品川シーズンテラスカンファレンス（タワー棟3F　カンファレンス A＋B＋C）
  （アクセス：JR品川駅　港南口（東口）より徒歩6分　http://www.sst-c.com/access/index.html　）
- **画像処理、コンピュータービジョン、機械学習の機能をご紹介！**
  - MATLABではじめる画像処理ワークフロー
  - 例題で実感するMATLABの画像処理機能
  - MATLABで試す！機械学習の応用例

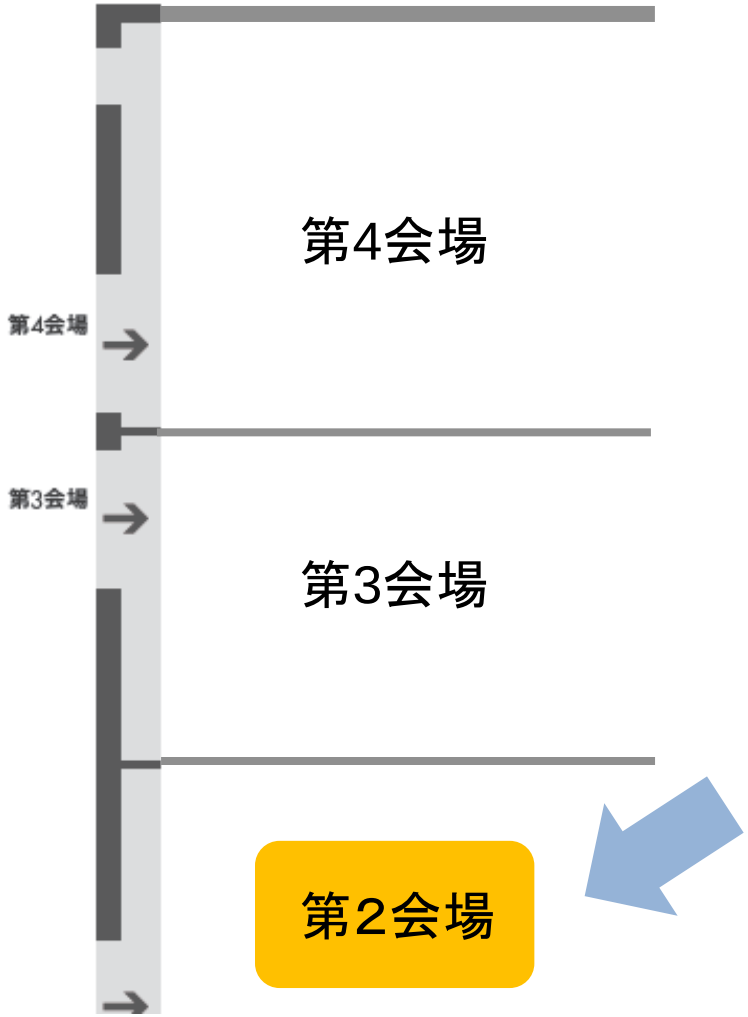







デモブースの紹介


3
1
7
2
8
ディープラーニング
4
9
10
自動運転デモブース
5
6
11
12
14
オープン
シアター
16
15


第4会場
第4会場
第3会場
第3会場
第2会場