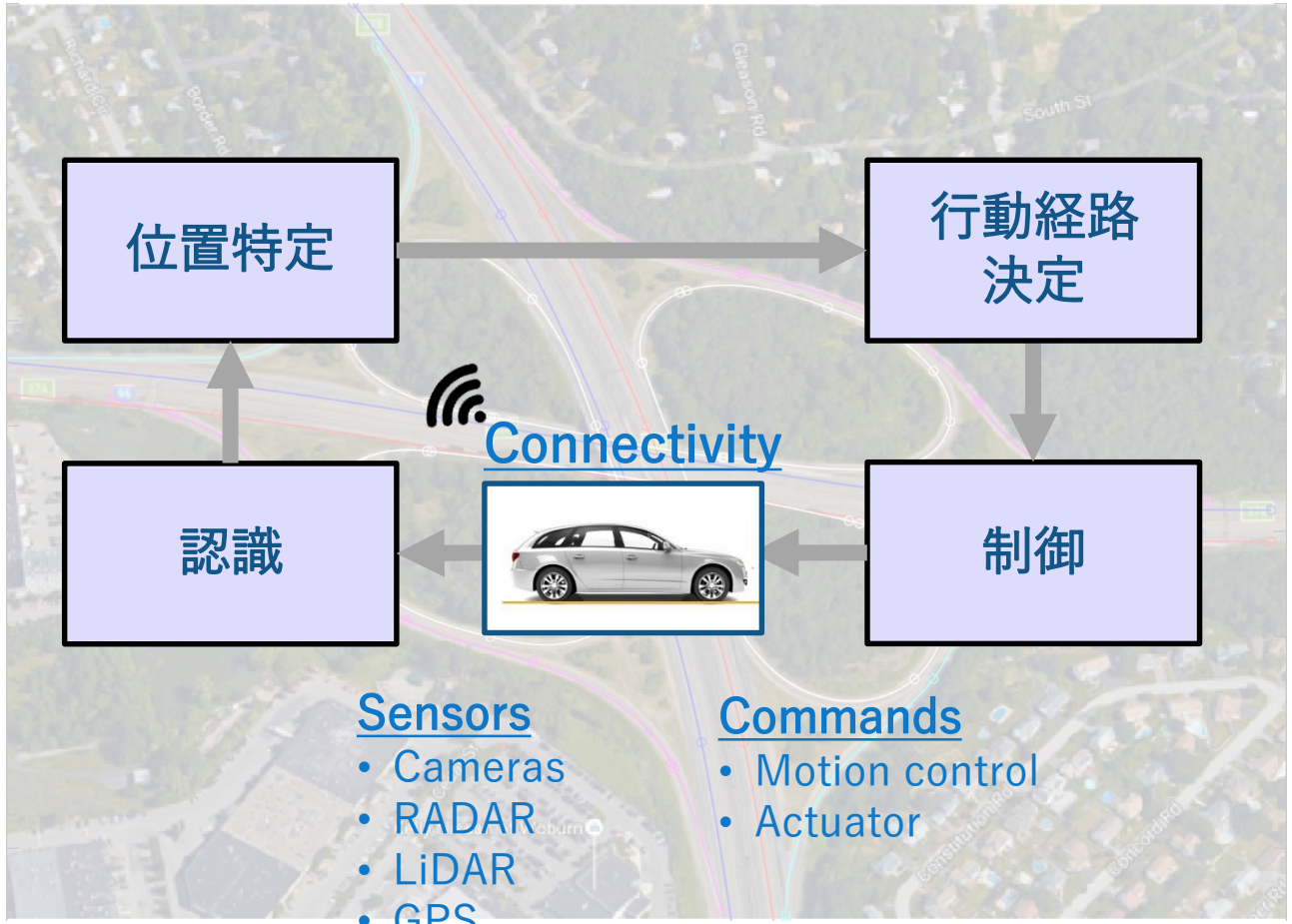
MATLAB EXPO 2018

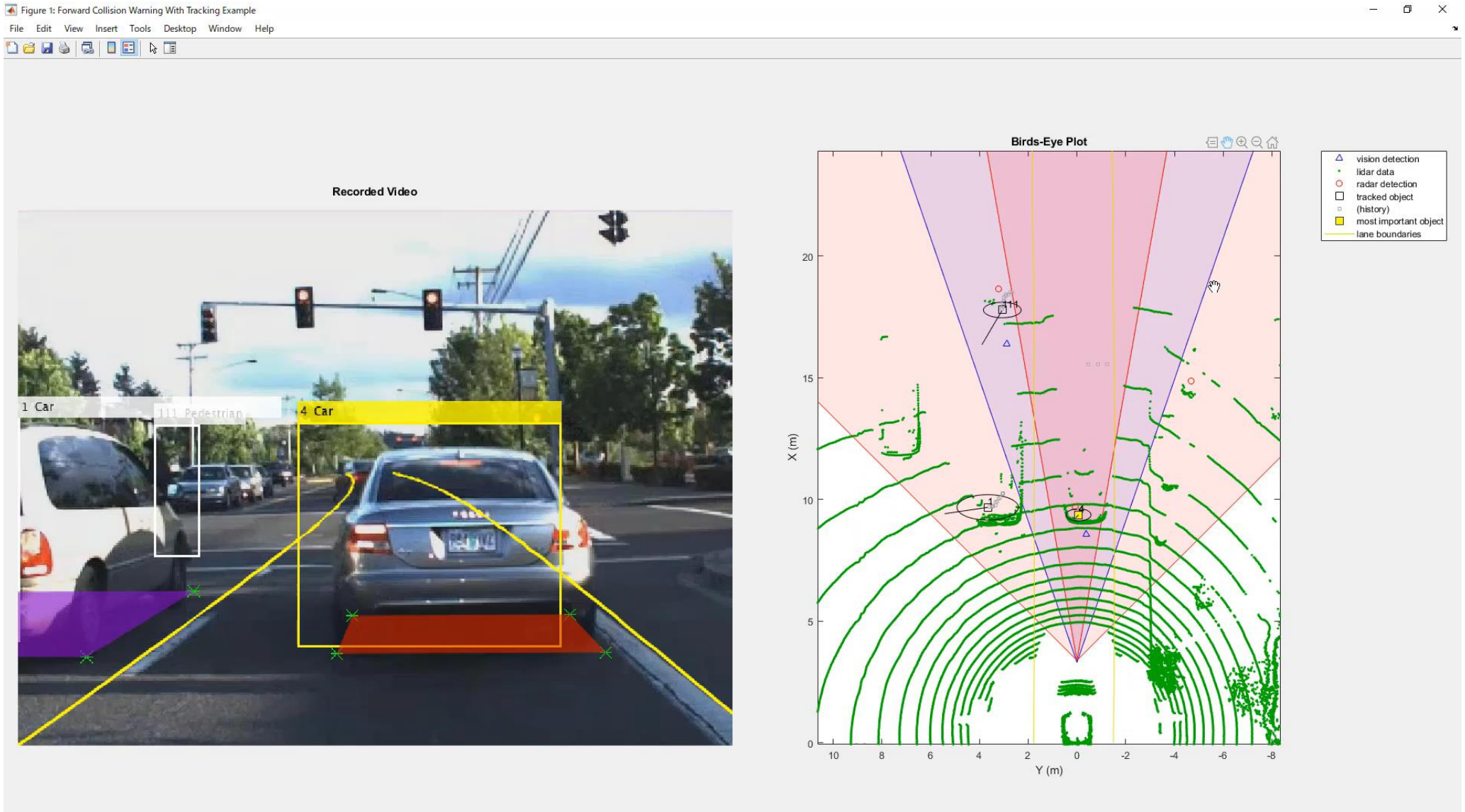
ADAS・自動運転の開発・検証を加速するMATLAB/Simulink

アプリケーションエンジニアリング部
大塚　慶太郎

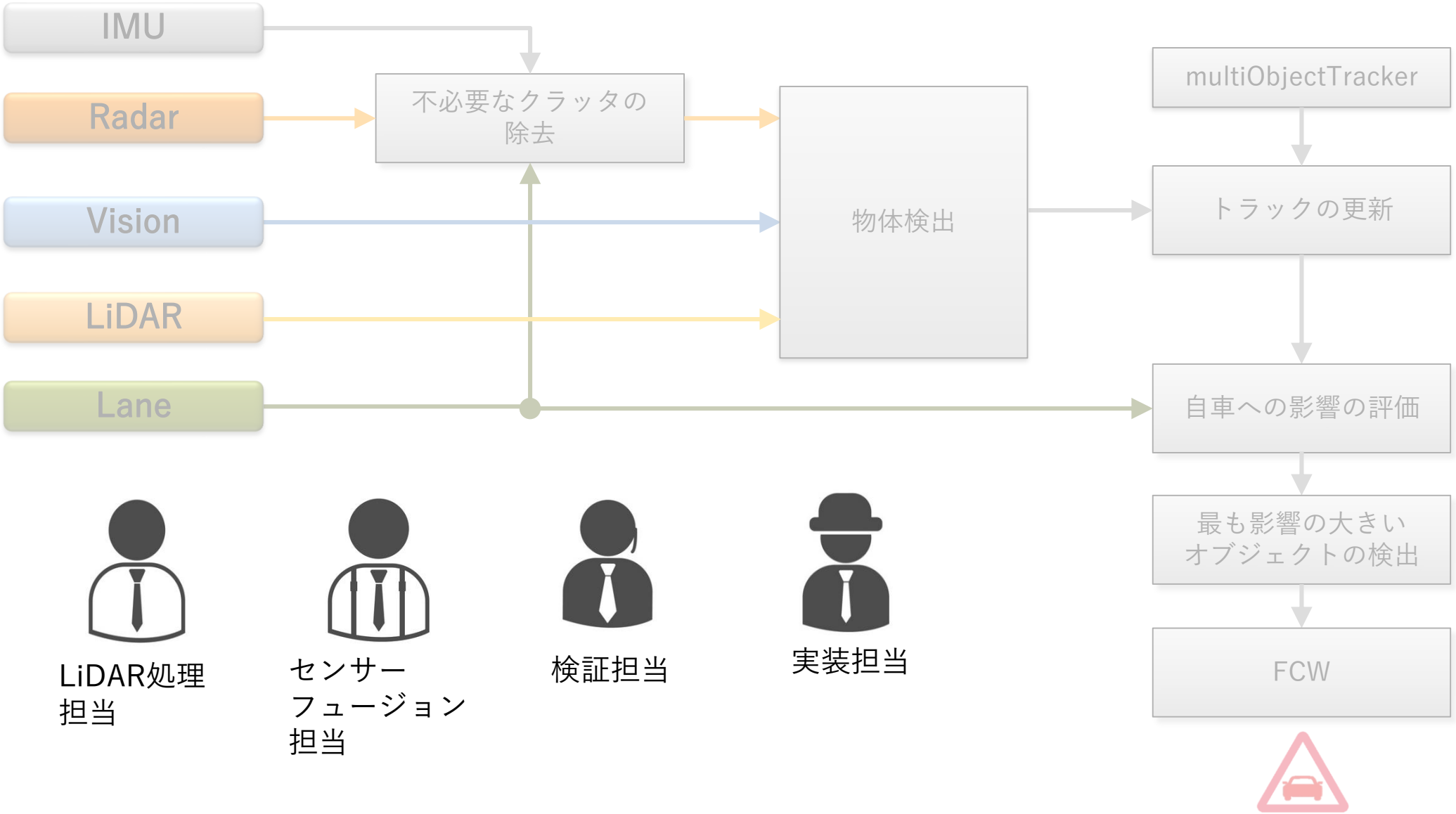
自動運転システムブロック図



前方車両接近警報(FCW)



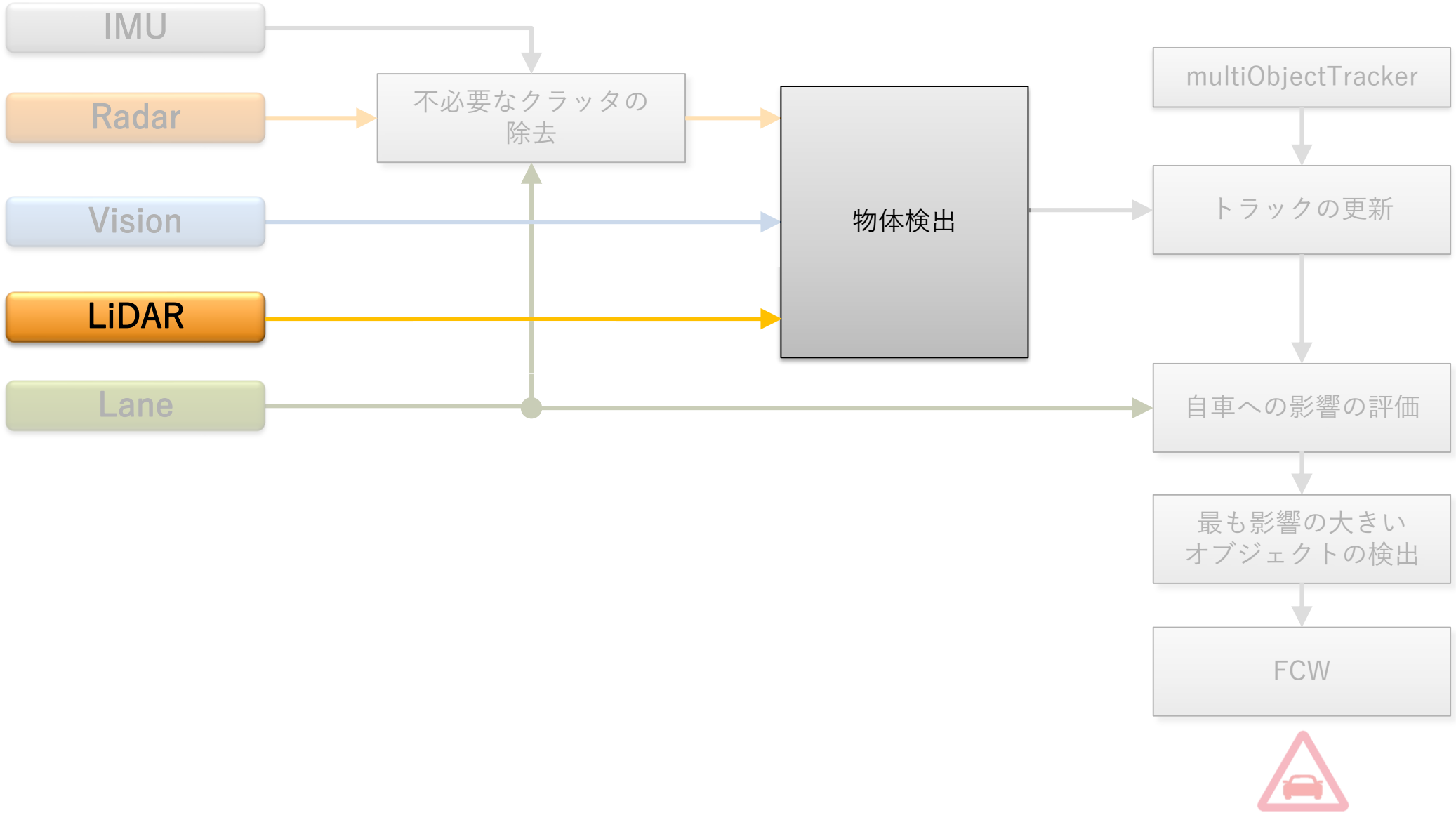
前方車両接近警報(FCW)



Agenda

- LiDARデータの取り扱い
- センサーフュージョン
- シナリオ生成とシミュレーション
- 豊富な実装オプション
- まとめ

前方車両接近警報(FCW)



LiDARデータの取り扱い

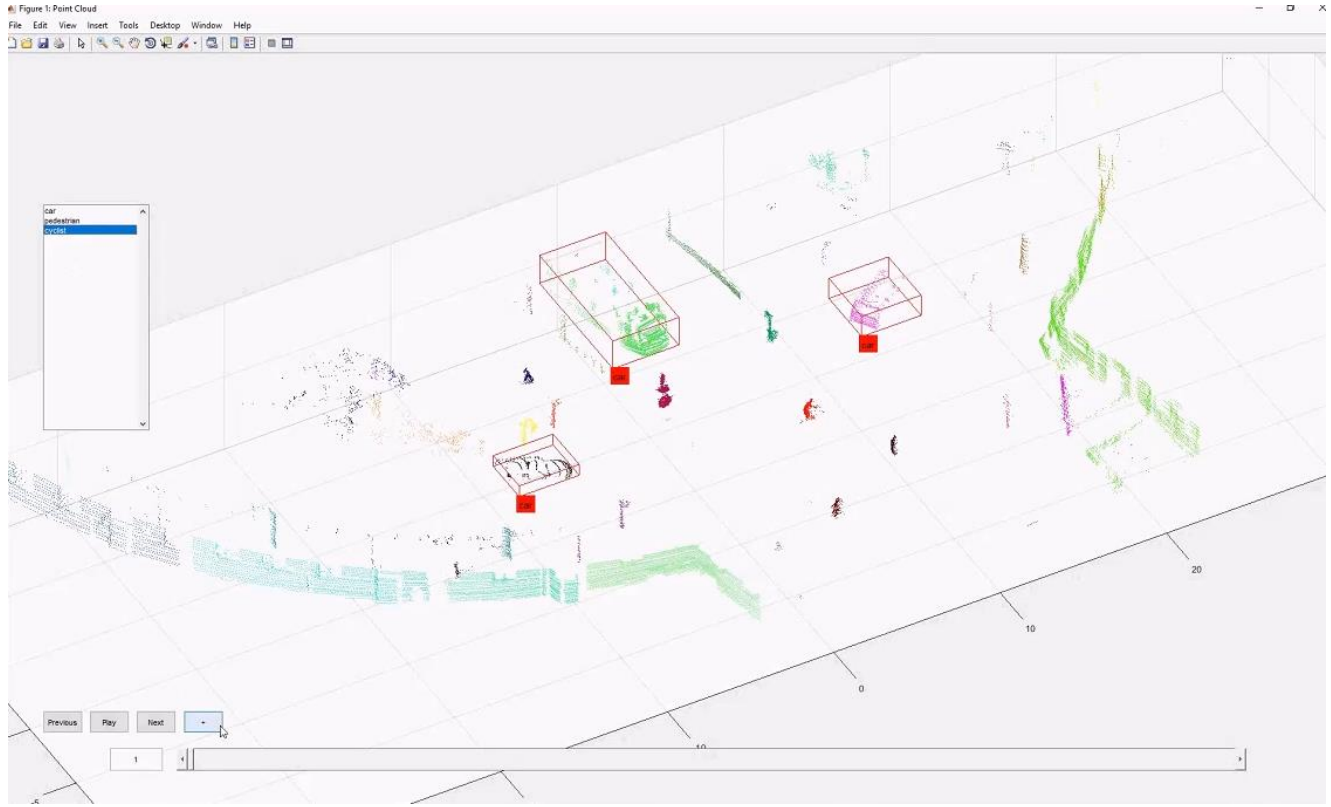
Image Processing Toolbox™
Computer Vision System Toolbox™

ファイル入出力

- pointCloud
- velodyneFileReader
- pcread
- pcwrite

前処理

- pcdenoise
- pcdownsample
- pctransform



位置合わせ

- pcregistericp
- pcregisterndt
- pcregistercpd

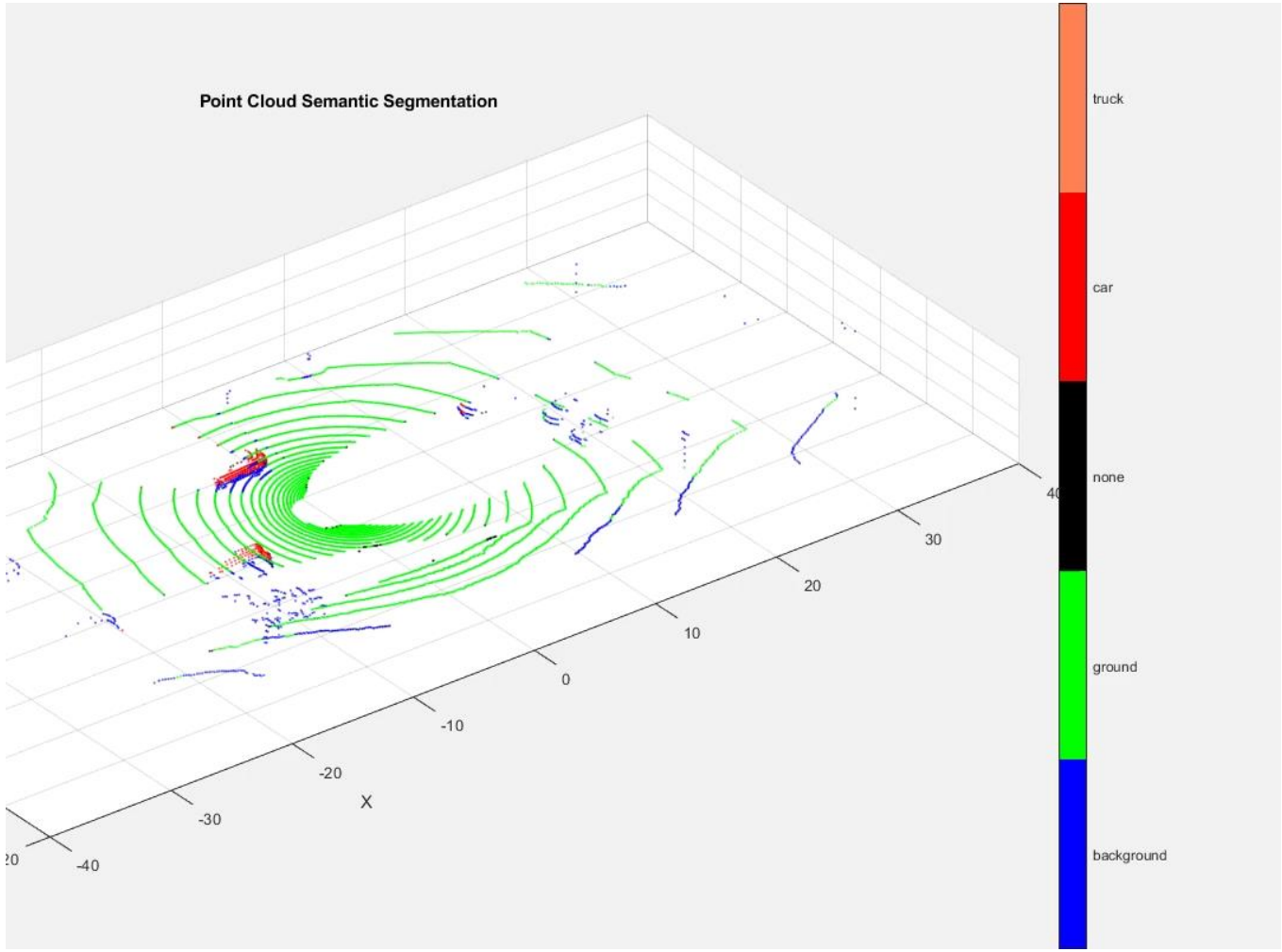
セグメンテーション

- pcsegdist
- segmentLidarData
- pcfitplane
- segmentGroundFromLidarData

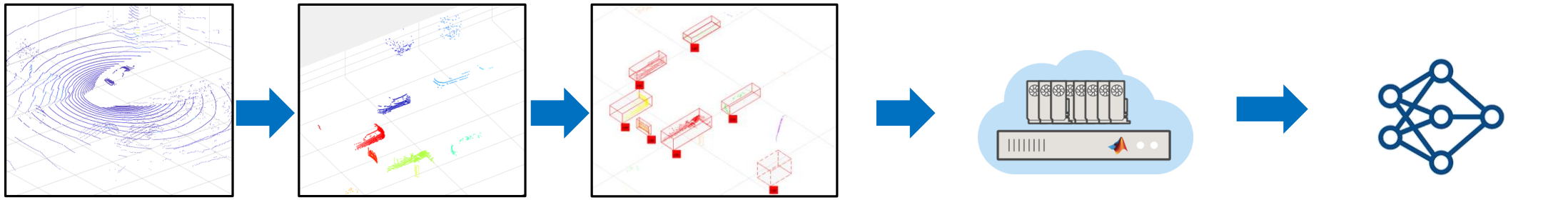
LiDARが初めてでも、使いやすい各種関数による可視化・解析・アルゴリズム開発

*18bで機能追加 or 新規追加

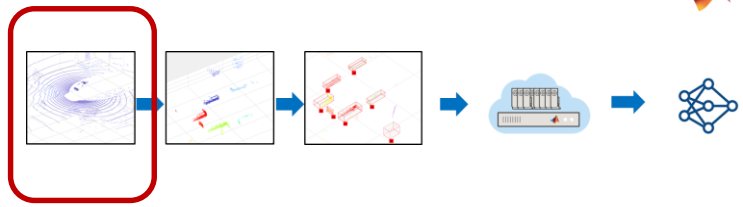
LiDAR Semantic Segmentation



データの前処理・ラベリング



LiDARデータへのアクセス、可視化

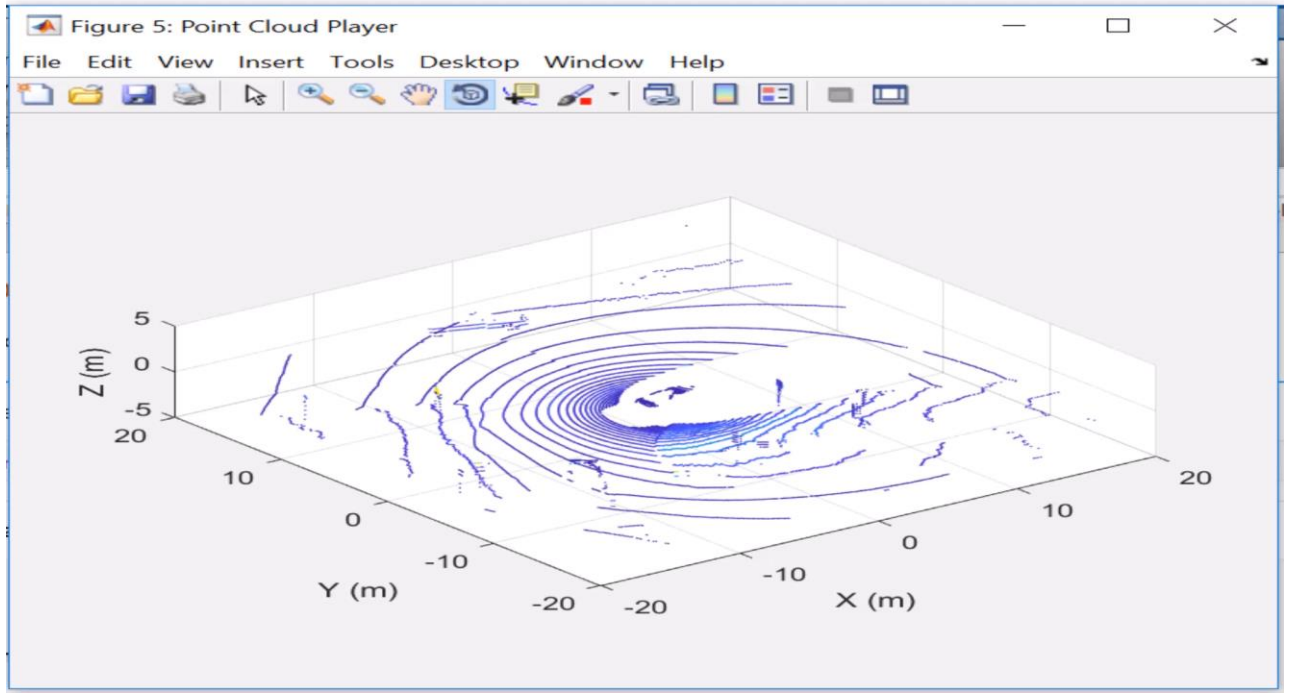


LiDAR データへのアクセス

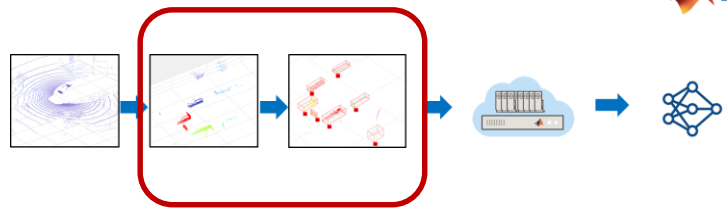
- Velodyne専用ファイル (pcap)
- 点群データフォーマット (.pcd,ply)
- バイナリファイルI/O

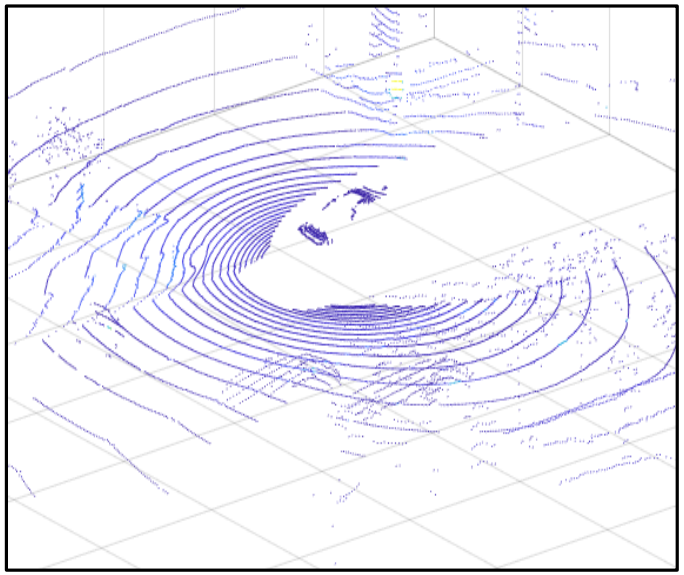
LiDAR データの可視化

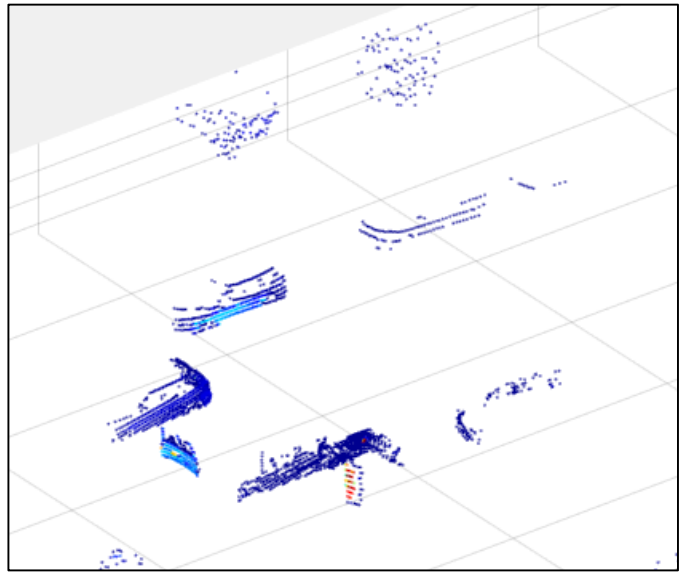
- 3次元点群データストリーミング再生
- 3次元点群データプロット



LiDAR データ前処理

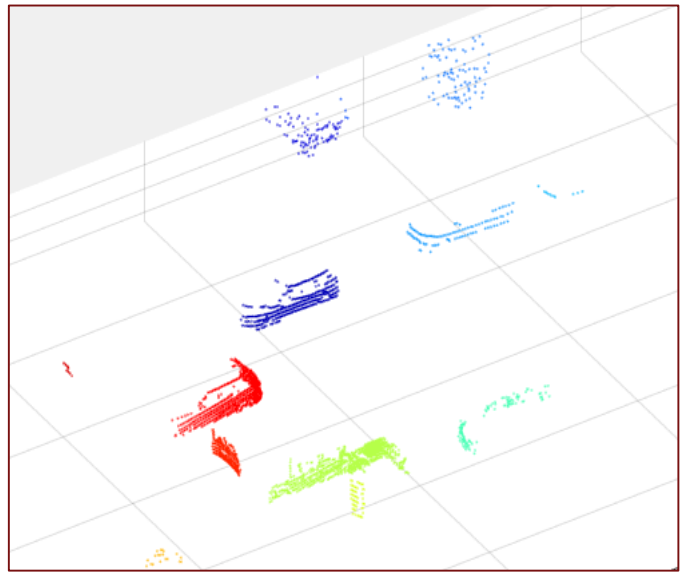






地表面の除去

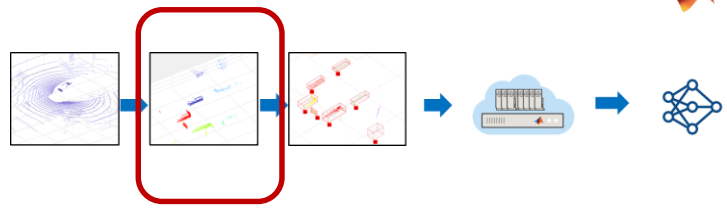
- RANSACを利用した
  モデルフィッティング

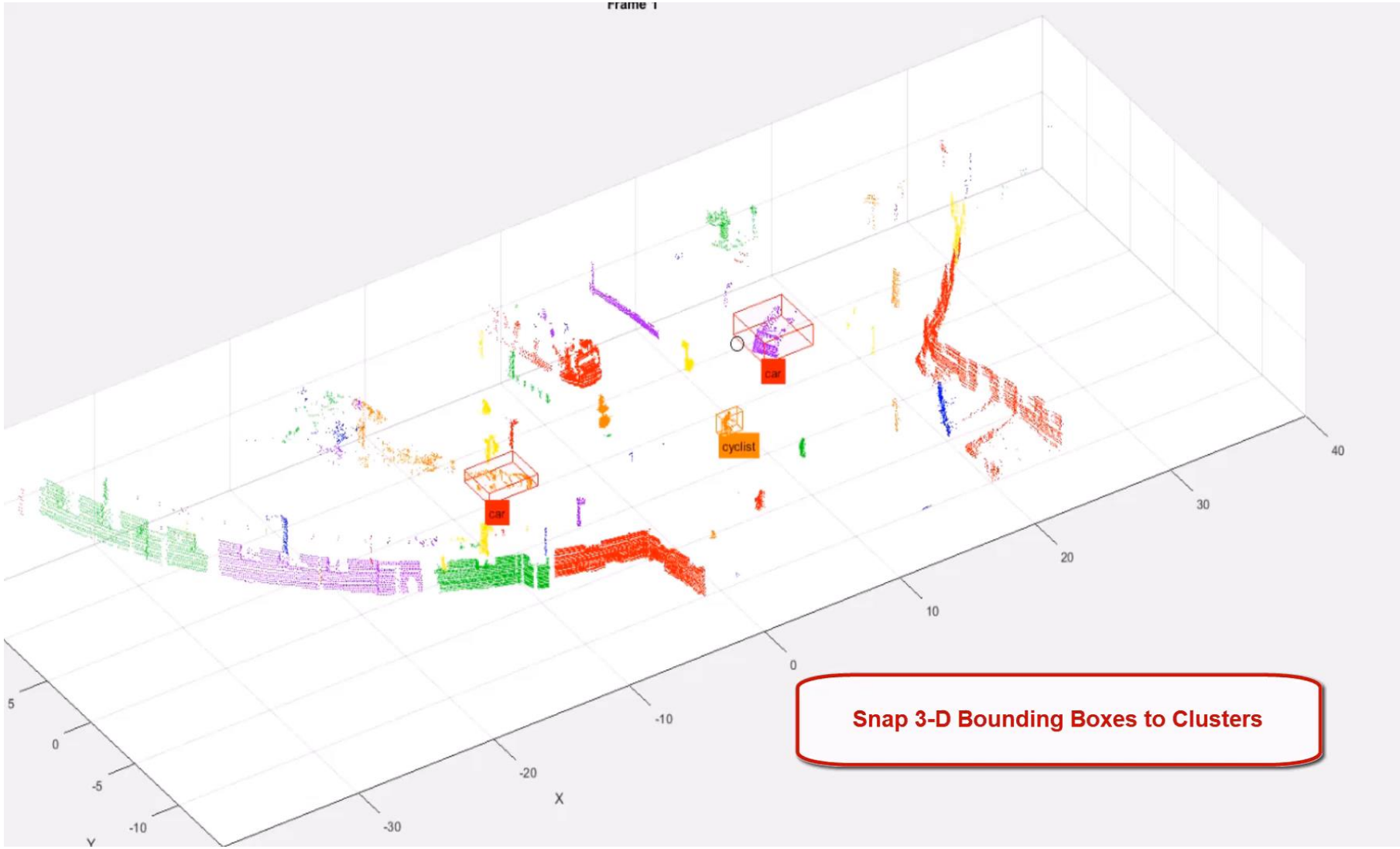


セグメンテーション

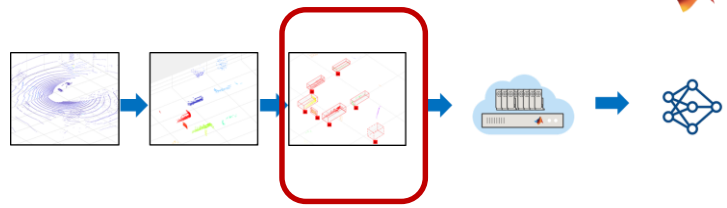
- ユークリッド距離を
  利用したクラスタ作成

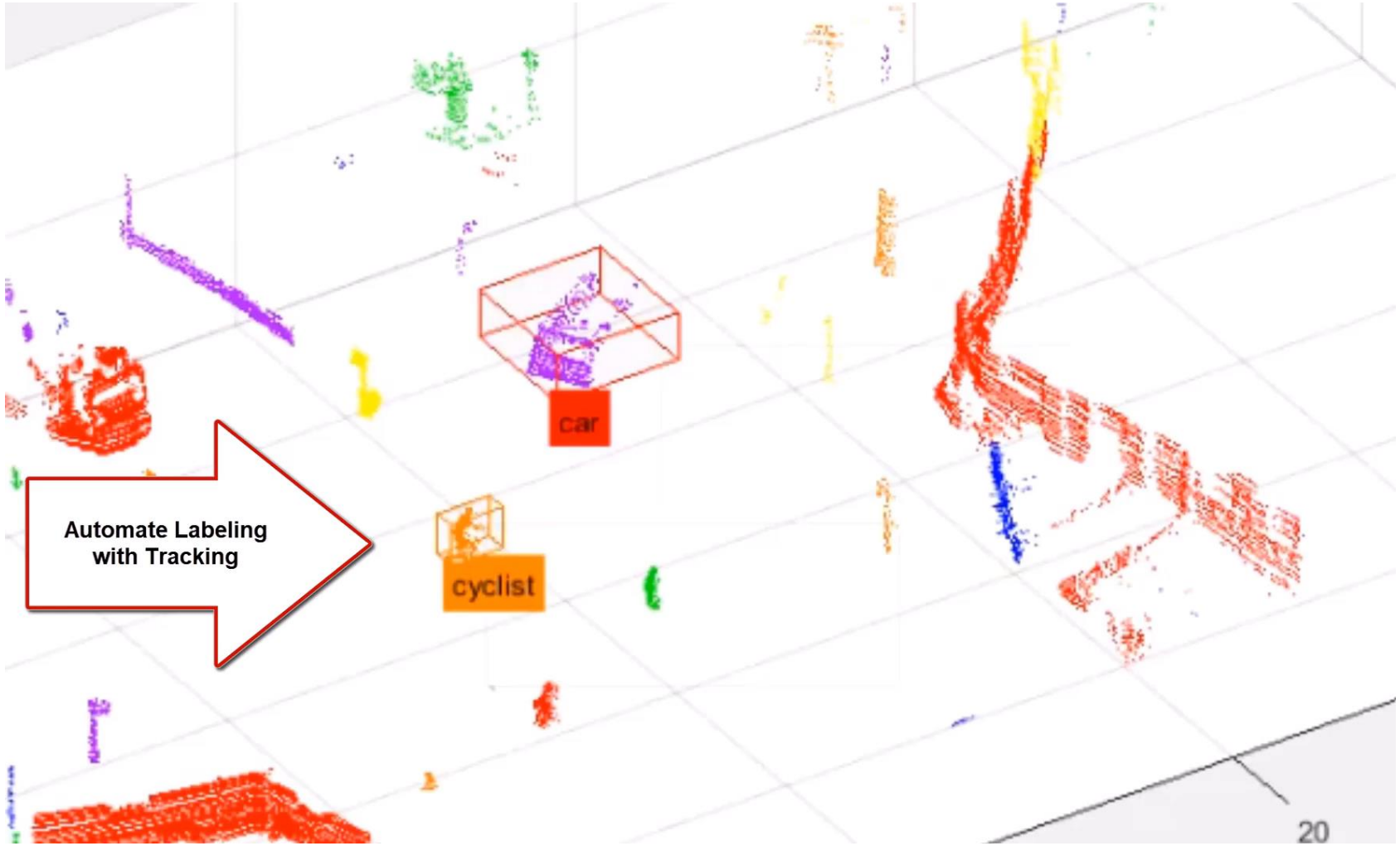
LiDAR データ 真値ラベリング



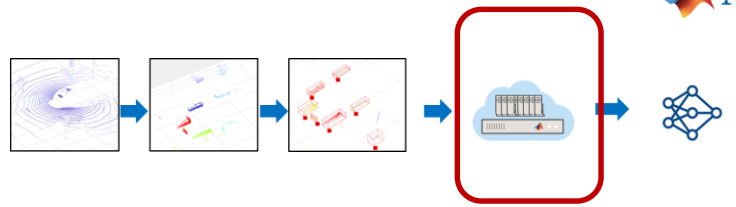


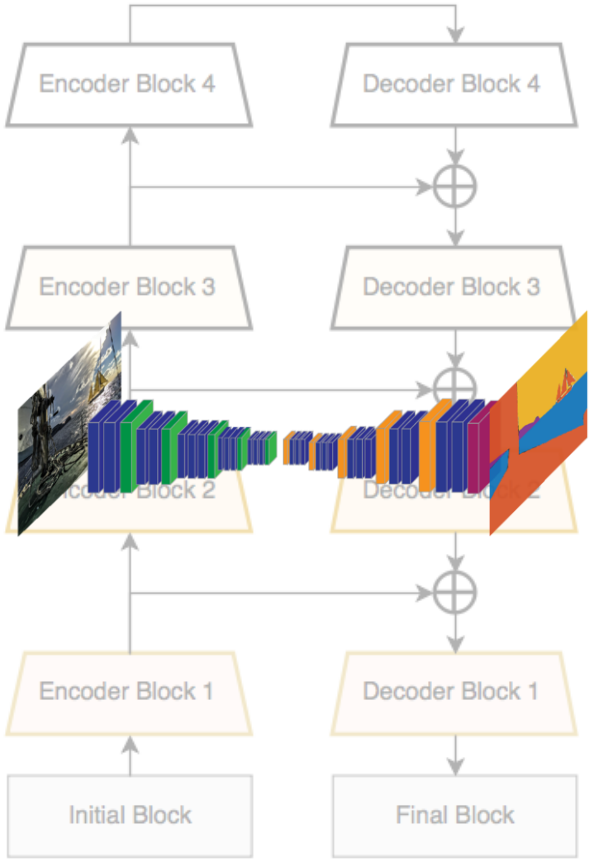
LiDAR データ 真値ラベリング





ネットワークの定義

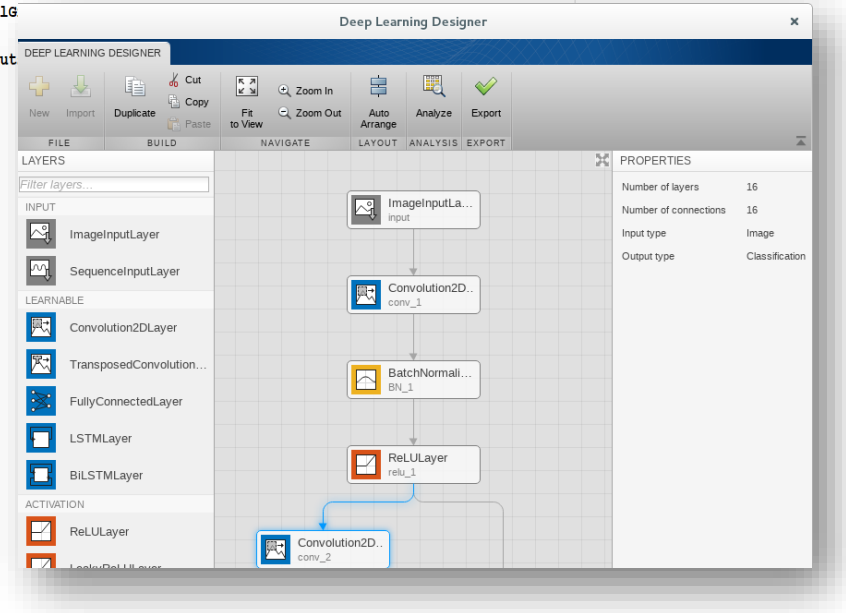




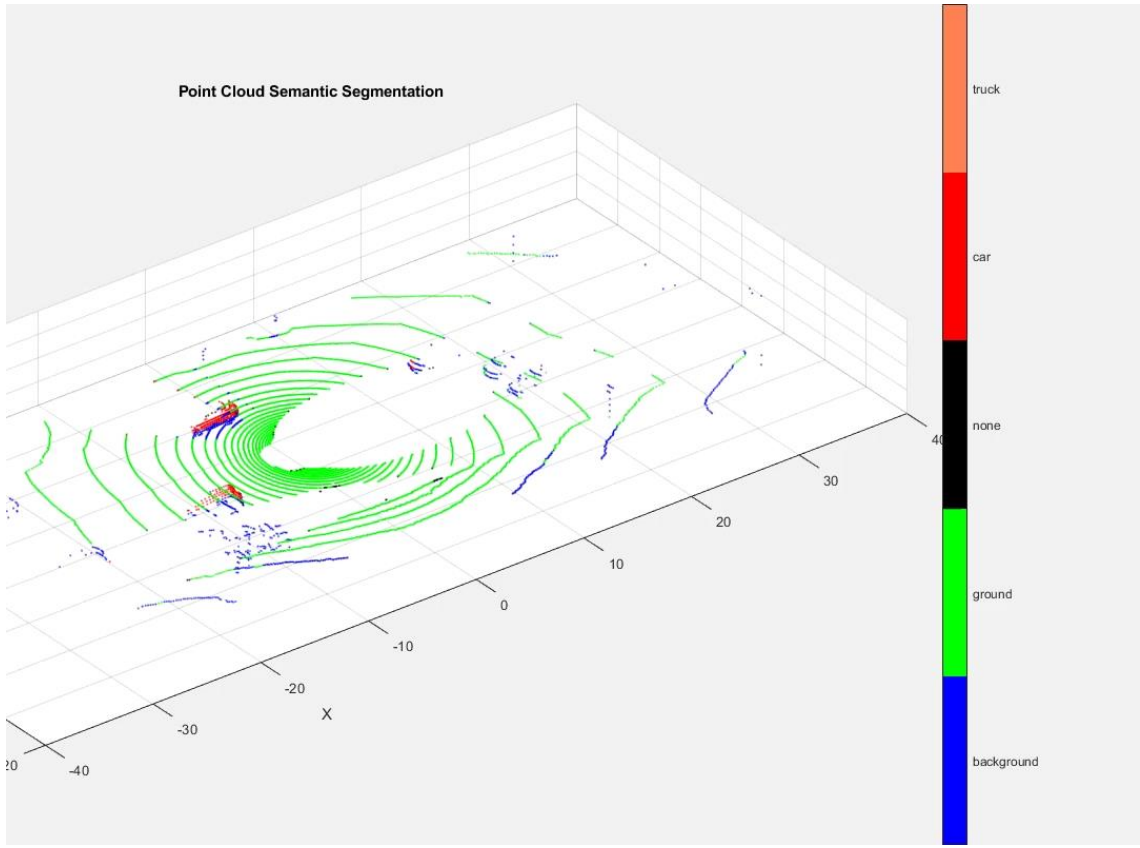
Easy MATLAB API to create network

```
%build encoder
nOutputs = 64;
inputLayerName = 'init_maxpool';
for blockIdx = 1:encoderDepth
    [lGraph, layerOutName] = encoderBlock(lGraph, blockIdx, nOutputs, inputLayerName);
    nOutputs = nOutputs * 2;
    inputLayerName = layerOutName;
end

%build decoder
nInputs = nOutputs;
inputLayerName = layerOutName;
for blockIdx = encoderDepth:-1:1
    nOutputs = min(nInputs/2, 64);
    [lGraph, decoderLayerOutName] = decoderBlock(lGraph, blockIdx, nInputs, nOutputs, inputLayerName);
    if blockIdx ~= 1
        inputLayerName = ['res_add' num2str(blockIdx)];
        lGraph = addLayers(lGraph, additionLayer(2, 'Name', inputLayerName) );
        lGraph = connectLayers(lGraph, ['enc' num2str(blockIdx-1) '_addout'],  [inputLayerName '/in2']);
        lG
    end
    nInput
end
```

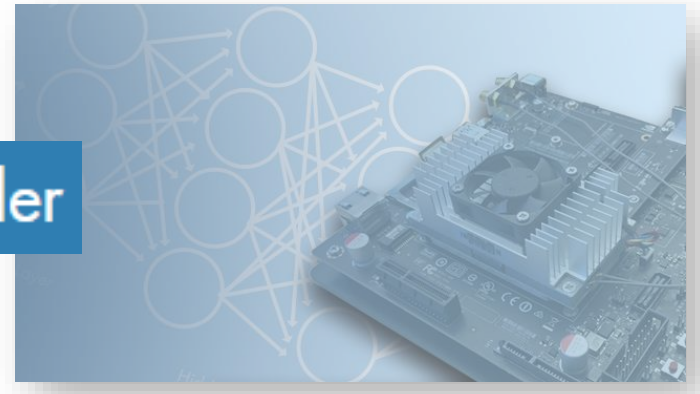


GPU Coderを利用した組み込み機器への実装



GPU Coder

C++/CUDA
+ TensorRT



C++/CUDA
+ cuDNN

Agenda

- LiDARデータの取り扱い
- センサーフュージョン
- シナリオ生成とシミュレーション
- 豊富な実装オプション
- まとめ

自動運転車両のセンサー例

カメラ：動画像

レーダー

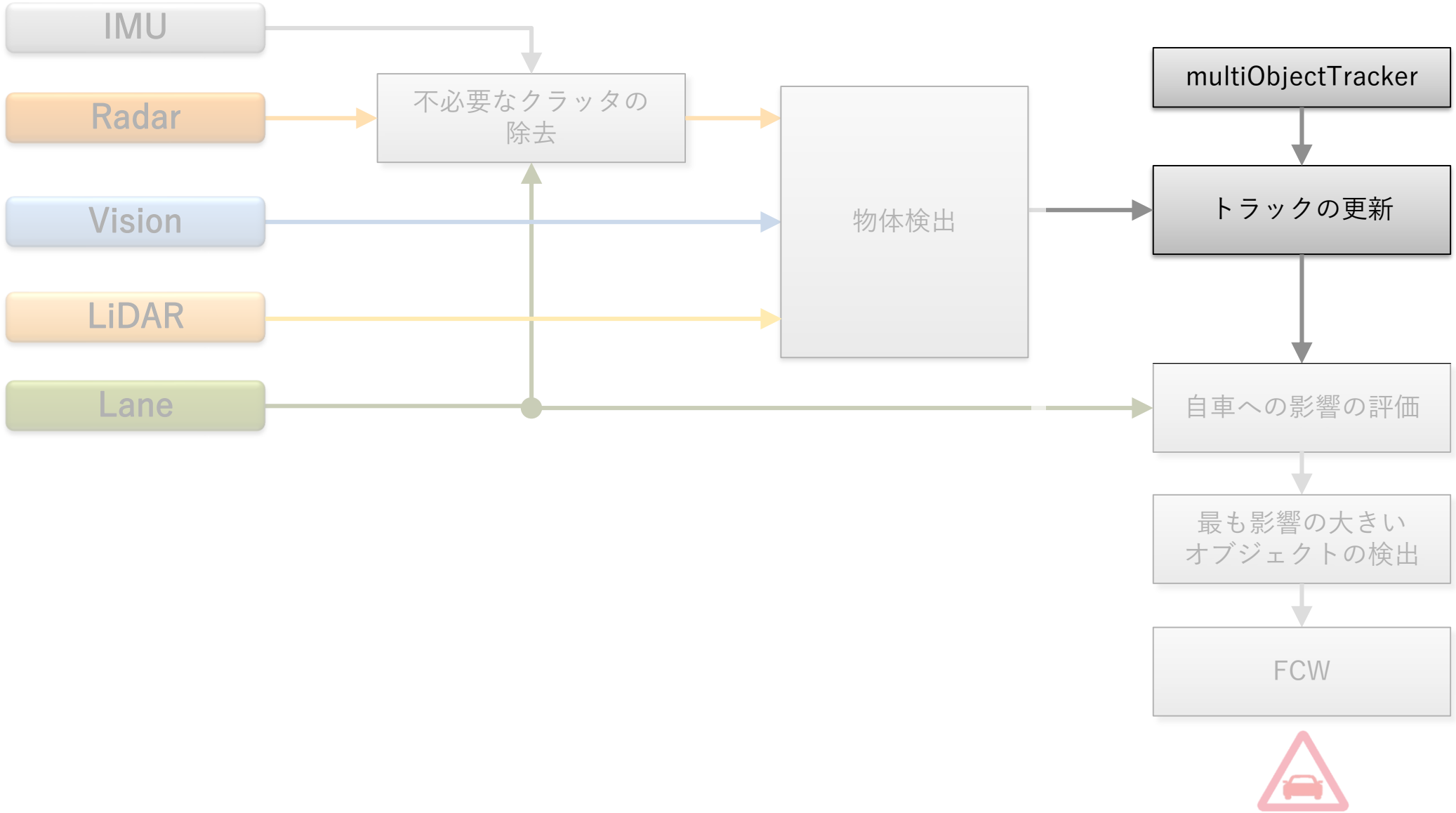
カメラ：物体検出

LiDAR : 物体検出

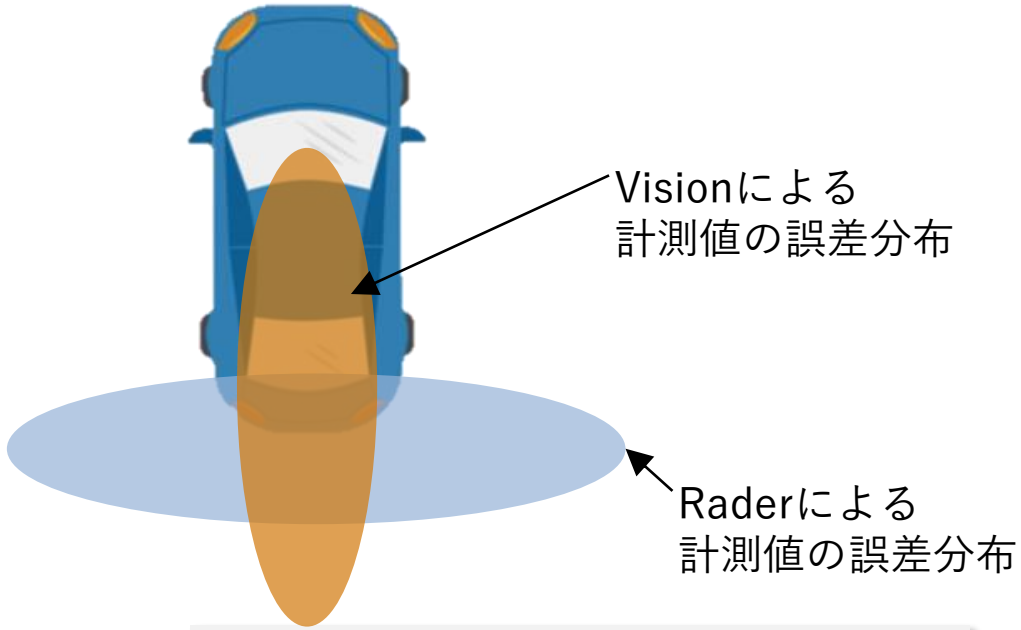
カメラ：区画線検出

IMU（角速度等）

前方車両接近警報(FCW)

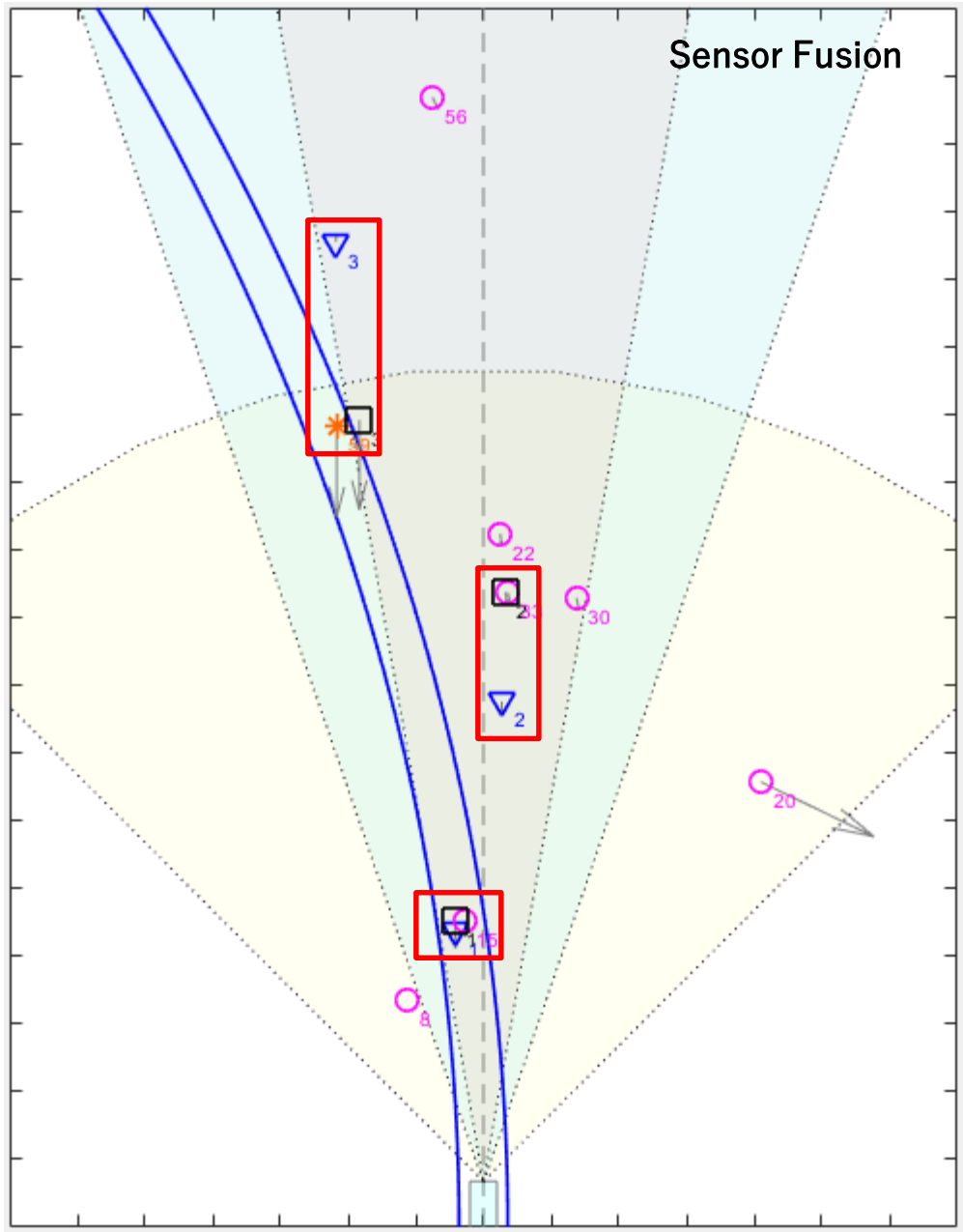


センサーフュージョンの重要性について

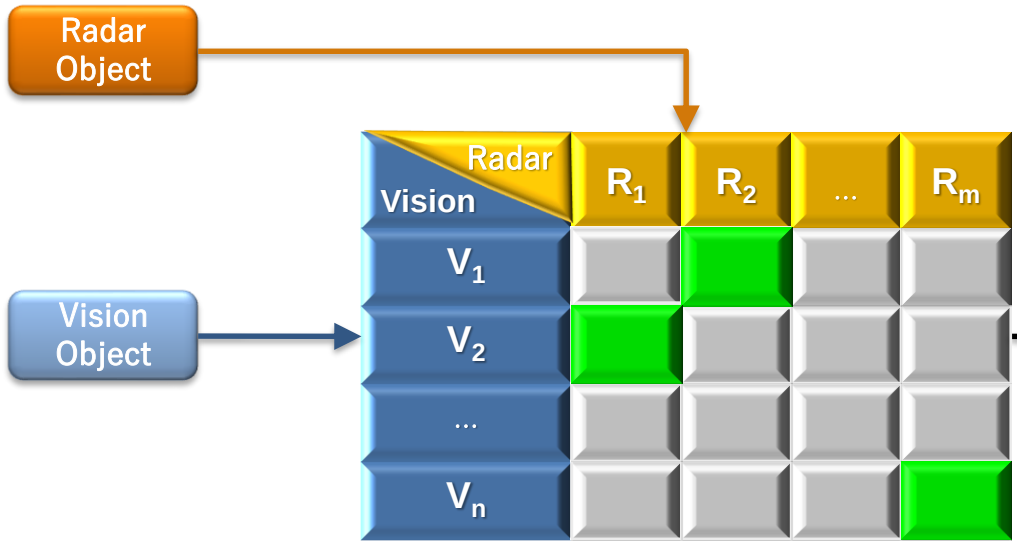


Birds-Eye View object notations

- ✳ Radar object (stationary)
- ○ Radar object (moving)
- ▽ Vision object
- ■ Fused object (safe zone)
- ■ Fused object (warn zone)
- ■ Fused object (FCW zone)
- ⊠ Fused object (most important object)



計測値の最適割り当てについて



コストマトリックス

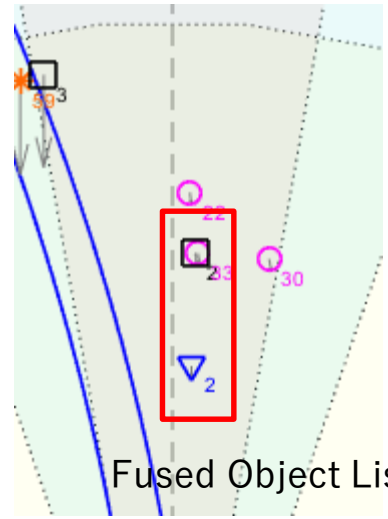
Assignments

$V_1 + R_2$
$V_2 + R_1$
...
$V_n + R_m$

Vison計測結果と
それに関連付けられた
Rader計測値のペア

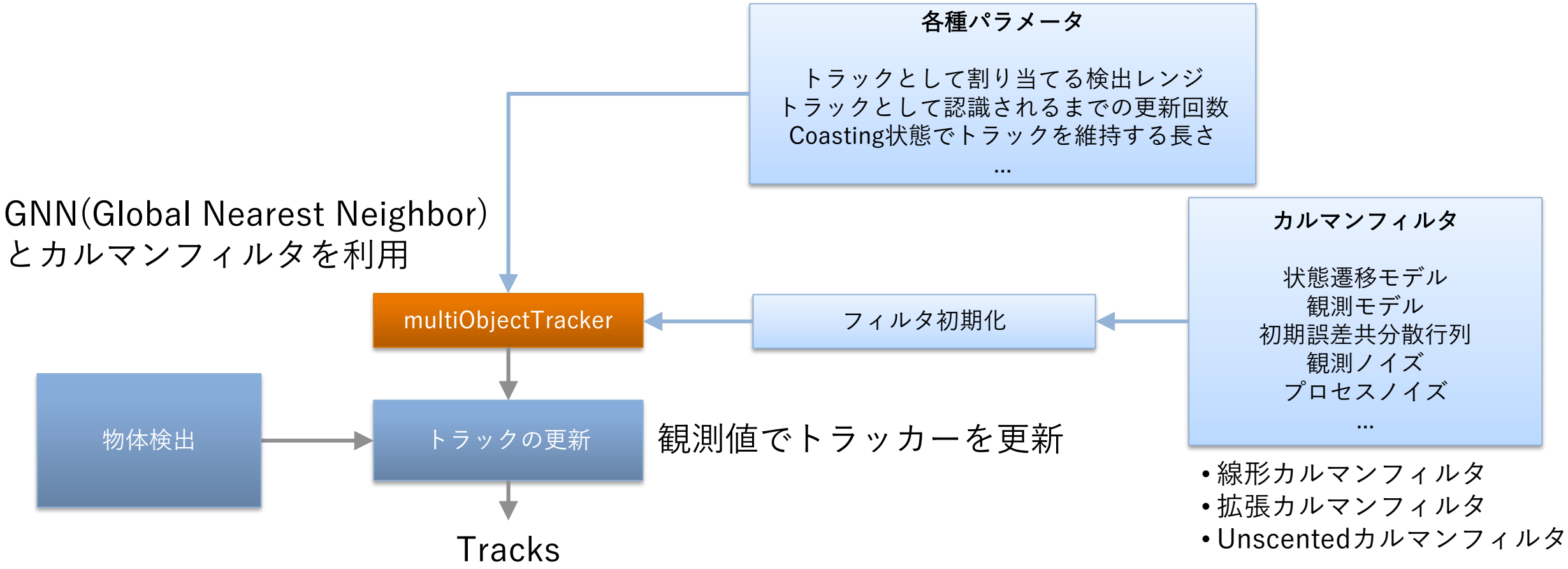
Fusion

$f(V_1) + f(R_2)$
$f(V_2) + f(R_1)$
...
$f(V_n) + f(R_m)$

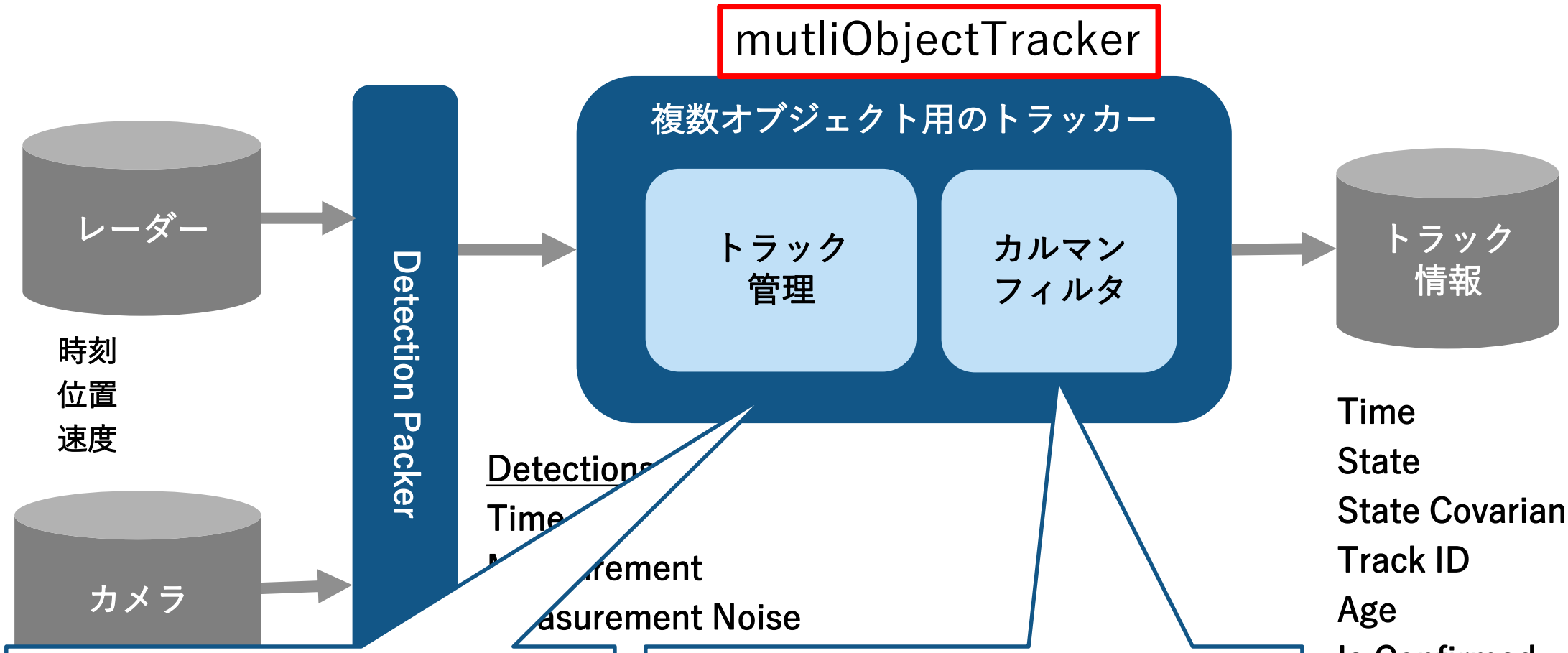



```
[assignments, unassignedVisions, unassignedRadars] = ...
    assignDetectionsToTracks(costMatrix, param.costOfNonAssignment);
```

複数オブジェクトのトラッキング



multiObjectTracker

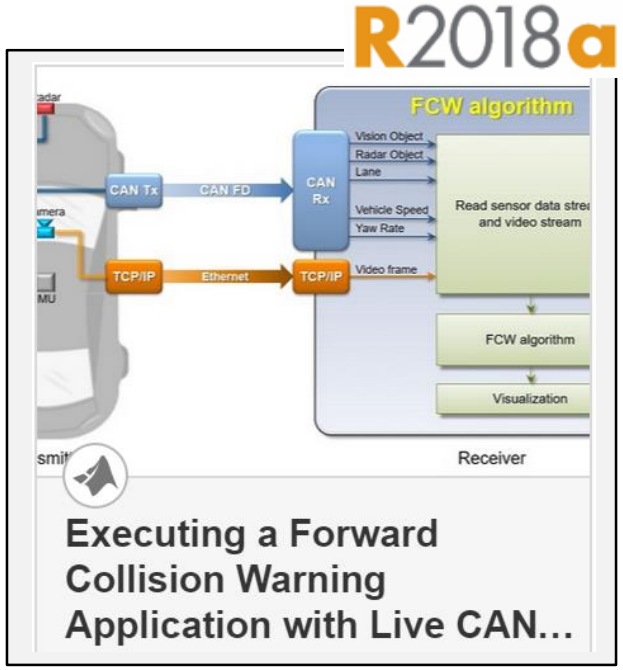


- 検出結果のトラックへの割当て
- 新しいトラックの生成
- 現有トラックの更新
- 古いトラックの削除

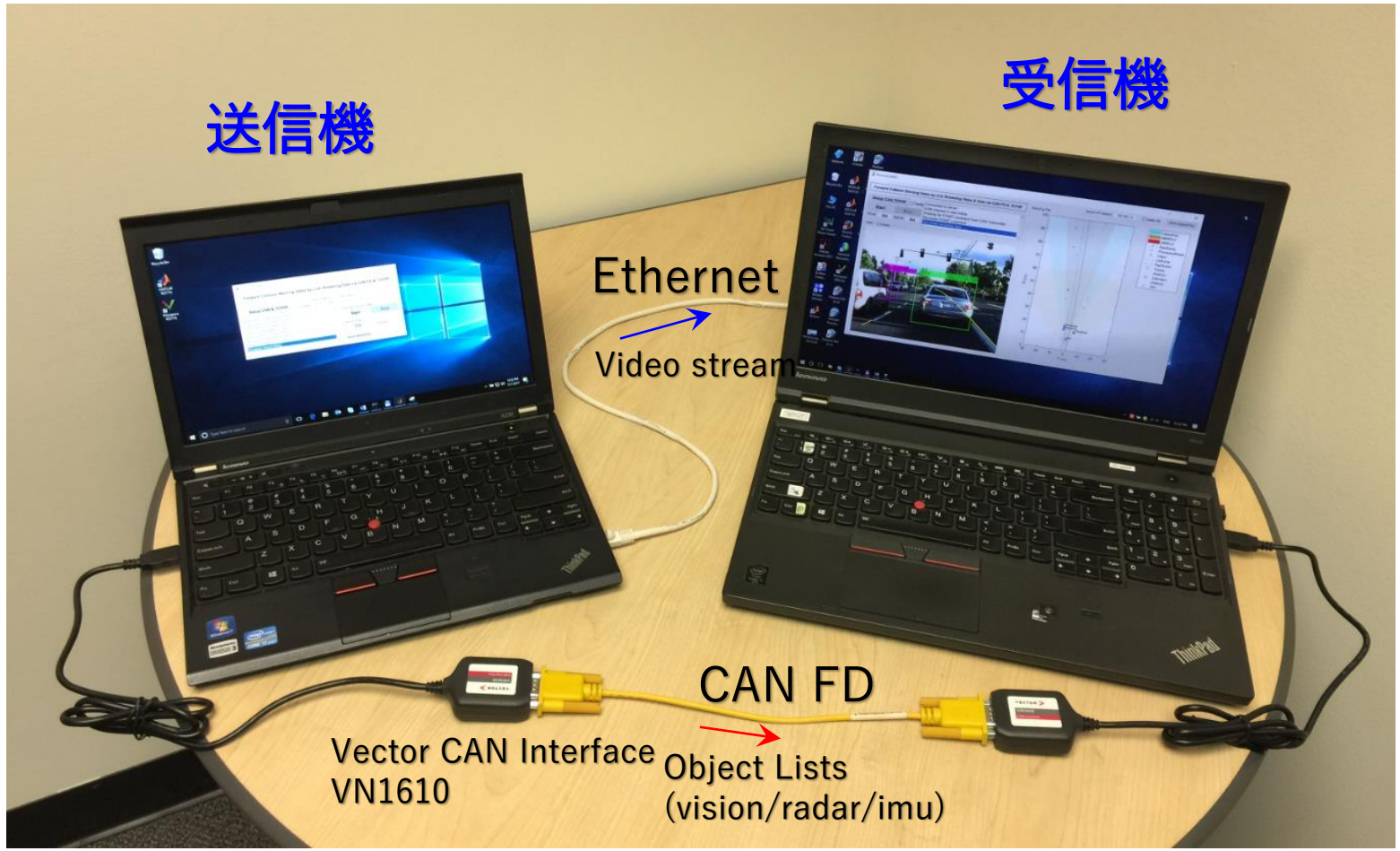
- トラックの状態の予測・更新
- カルマンフィルター：
  線形・拡張・unscented

CAN FDデータを利用したフュージョンアルゴリズムの開発

Automated Driving System Toolbox™
Instrument Control Toolbox™
Vehicle Network Toolbox™

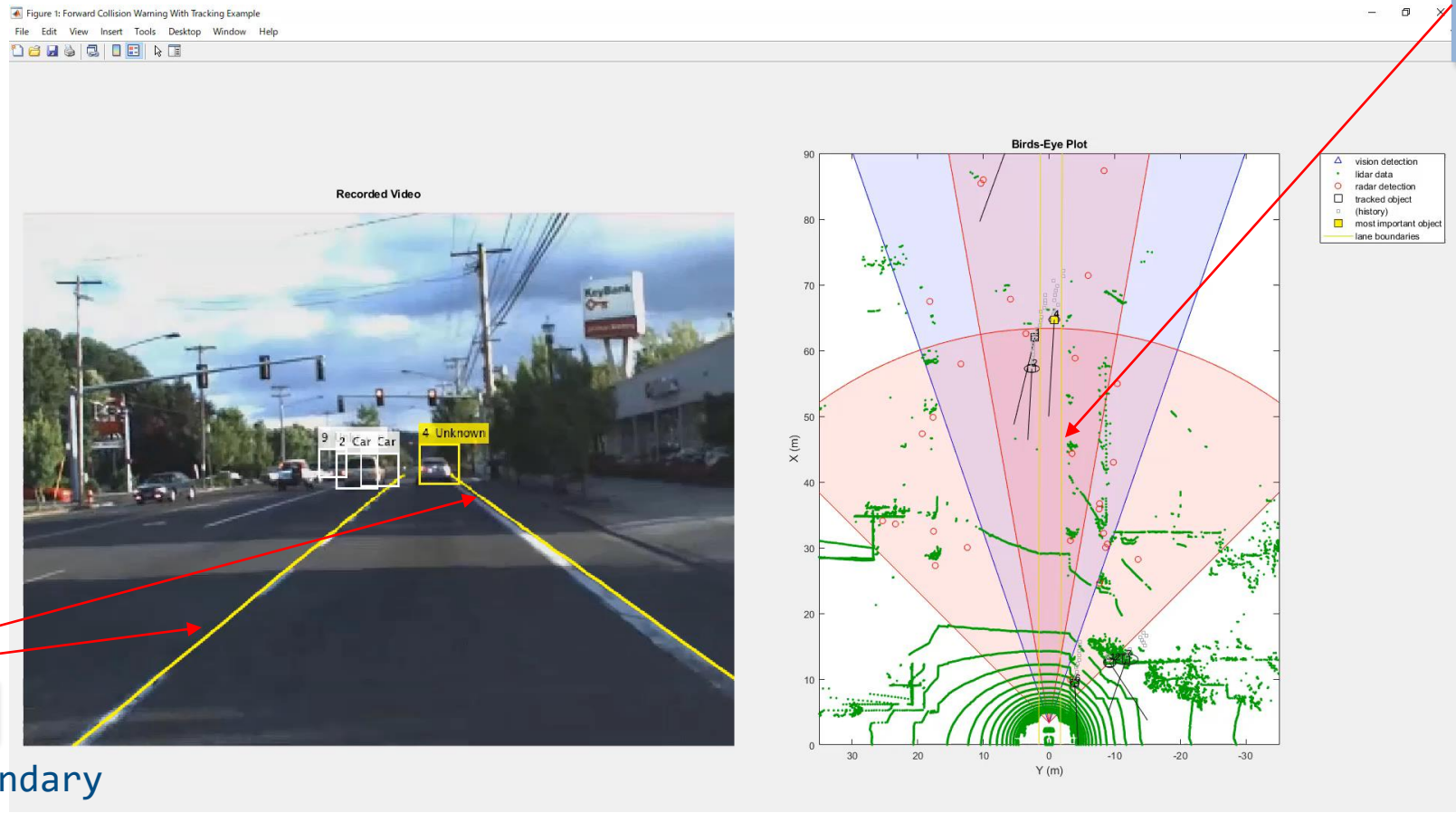


- CAN FD データのストリーミングとアルゴリズムのプロトタイピング



センサーフュージョンによるFCW(前方車接近警報)

Automated Driving System Toolbox™



フュージョン&トラッキング

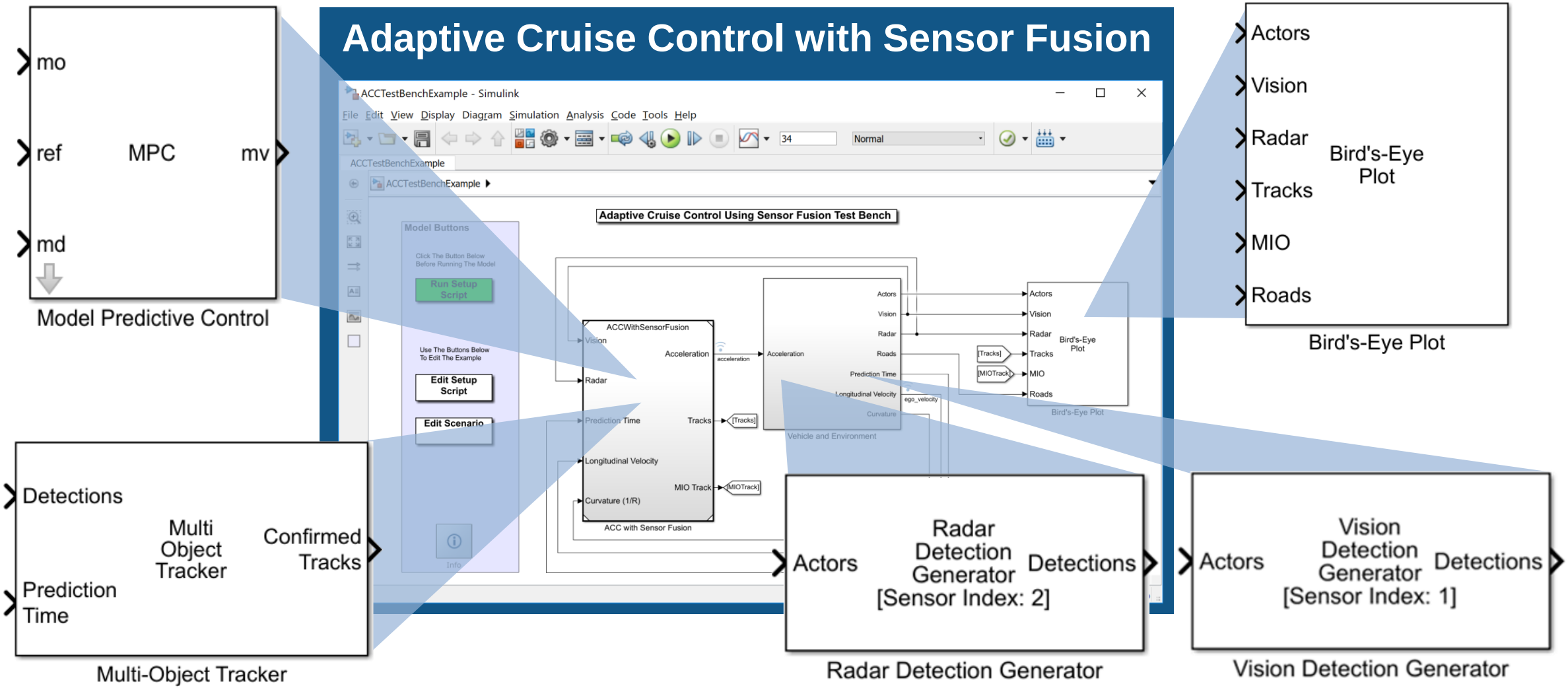
- objectDetection
- trackingEKF
- trackingUKF
- trackingKF
- updateTracks

データ可視化

- parabolicLaneBoundary
- insertLaneBoundary
- vehicleToImage
- insertObjectAnnotation

豊富な可視化機能、使いやすいフュージョン用フレームワークで開発エンジニアをサポートします

Simulinkブロックも利用可能



https://jp.mathworks.com/help/releases/R2018b/driving/examples/adaptive-cruise-control-with-sensor-fusion.html

Agenda

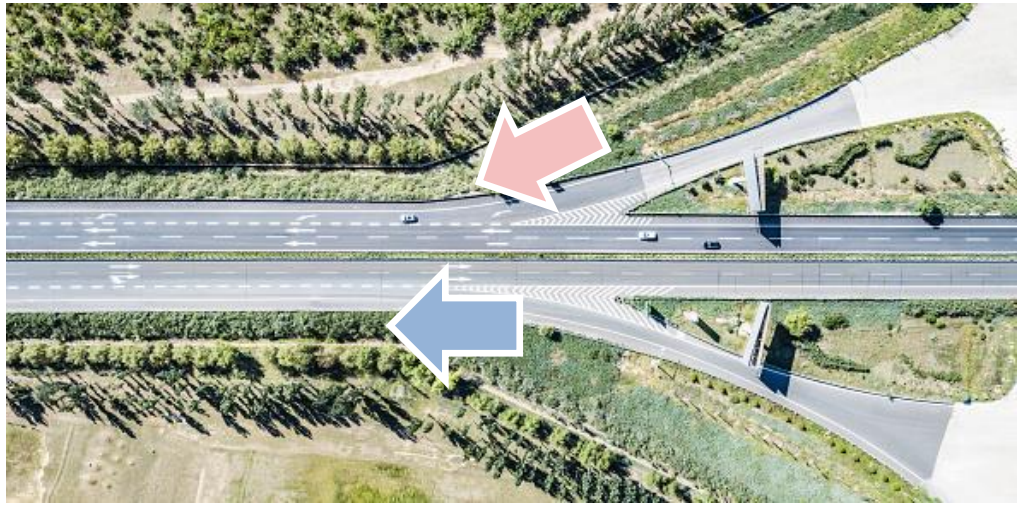
- LiDARデータの取り扱い
- センサーフュージョン
- シナリオ生成とシミュレーション
- 豊富な実装オプション
- まとめ

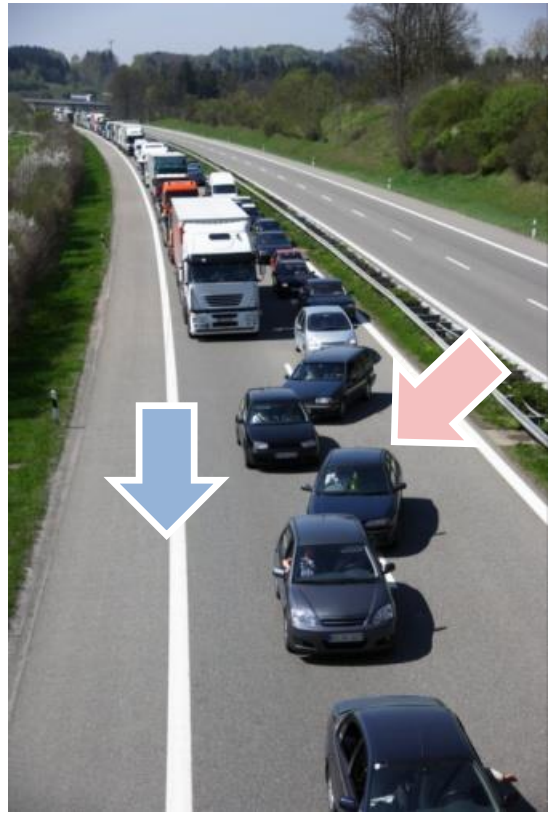
運転シナリオシミュレーションの重要性

カメラ
物体検出

カメラ
区画線検出

Rader
物体検出





・コーナーケースでのフュージョンアルゴリズム検証
・異なる特性のセンサの組み合わせ、配置位置の検証

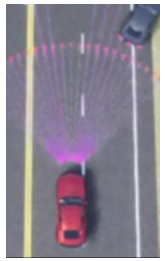
・より多くのテストケースバリエーションでの検証
・実車データ取得の難しい、危険なシナリオ

目的に応じた環境の選択

Gaming Engine Integration

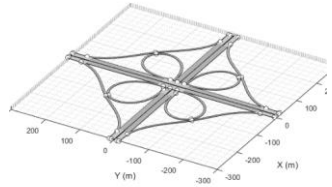
シナリオ作成

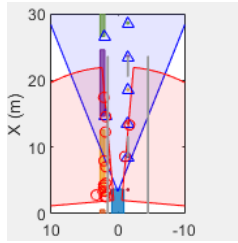
Driving Scenario Designer



より“リアル”な環境、
物理センサーモデル

- 認識系アルゴリズムを含む、より現実に近いシミュレーション
- 実行時間は低速、環境構築のため専門知識が必要



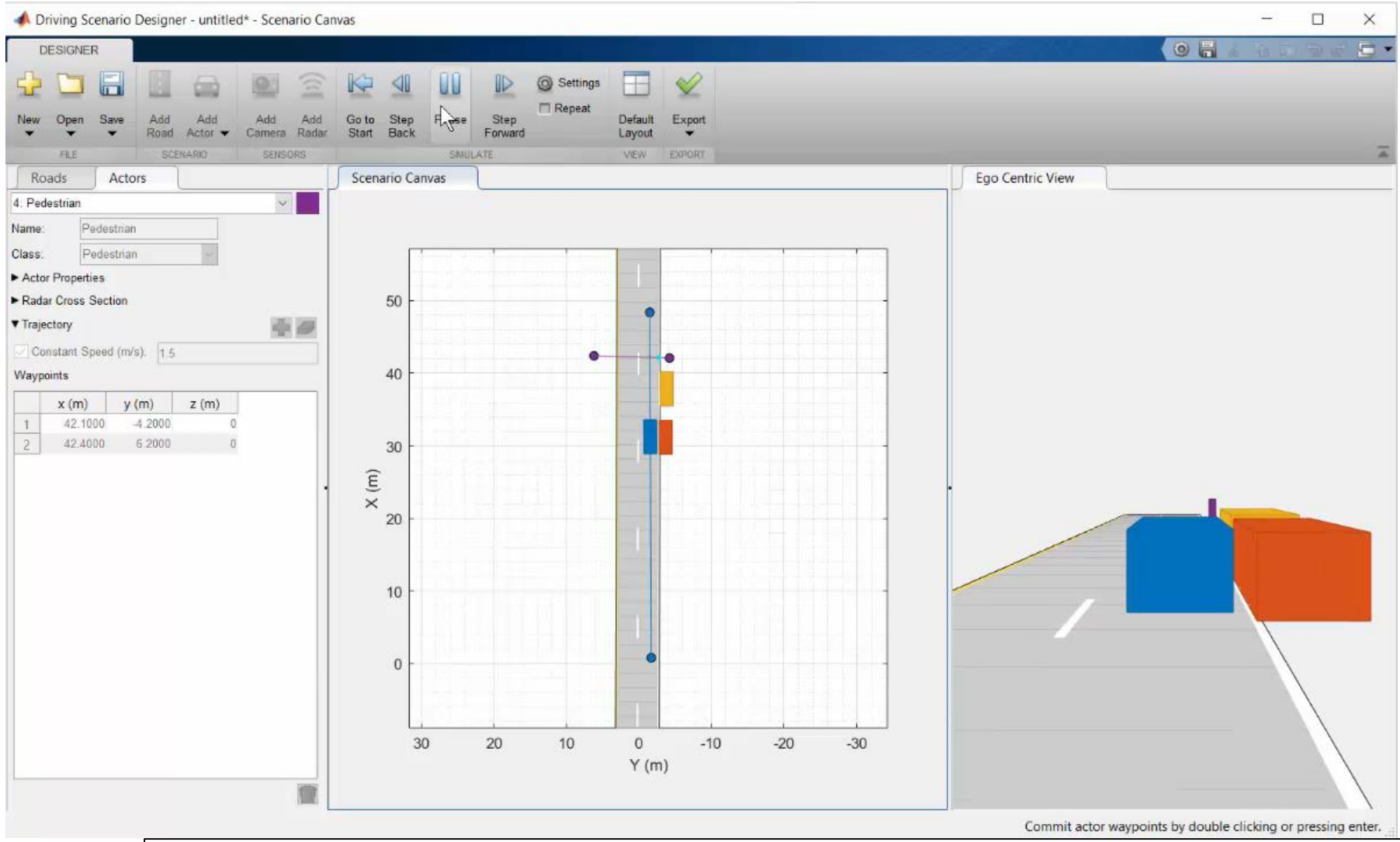


抽象度の高い、簡易的表現
確率的センサーモデル

- 統計的なセンサー特性に基づいた簡易シミュレーション
- 実行時間は高速、マウス操作でシナリオ生成が可能

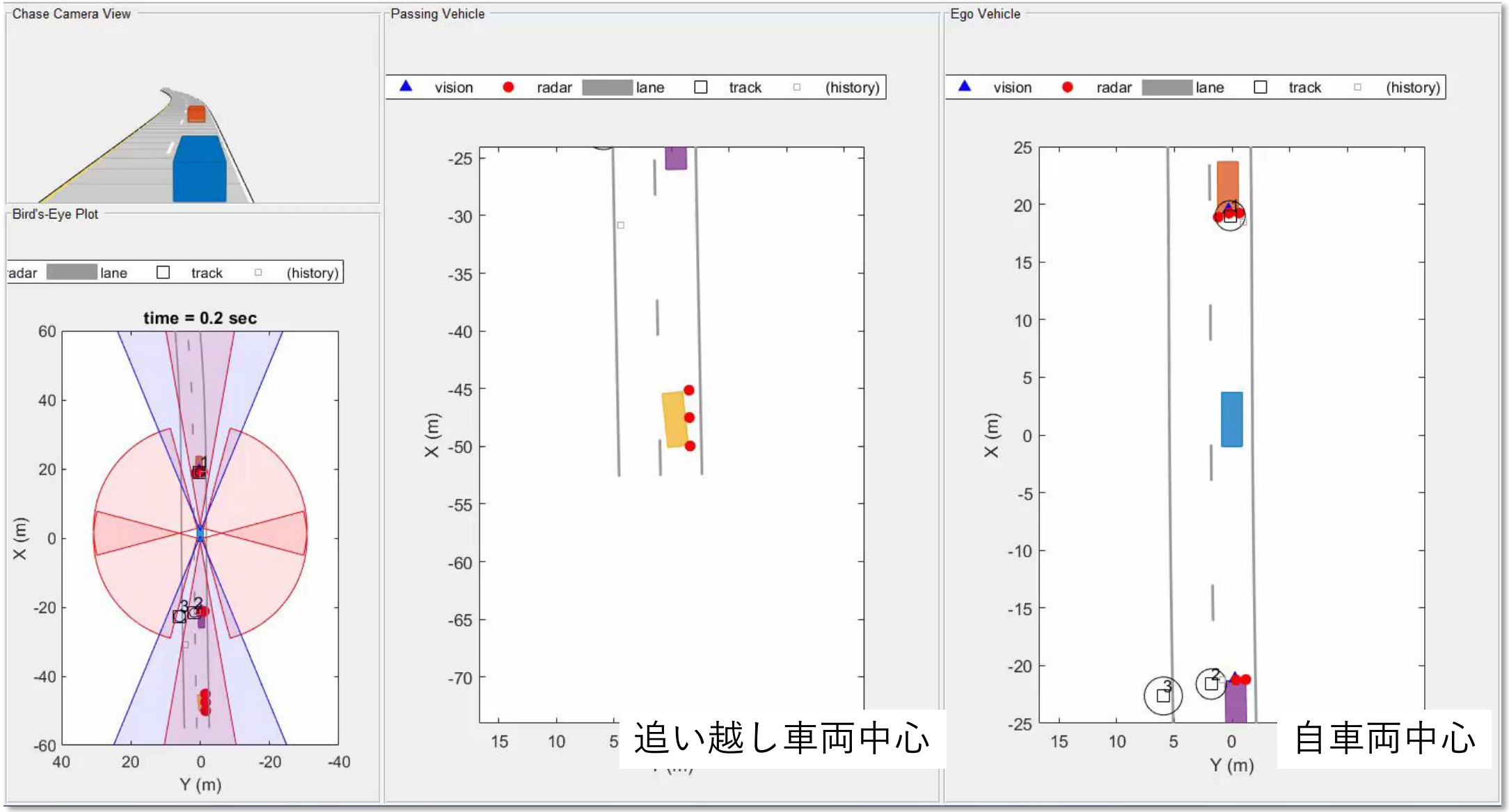
センサー特性や取り付け位置の検討、
センサーフュージョンや制御系アルゴリズムの検証など
幅広くお使い頂けます

Driving Scenario Designer : シナリオ作成用専用App



-マウス操作で様々な形状の道やセンサーを定義できます

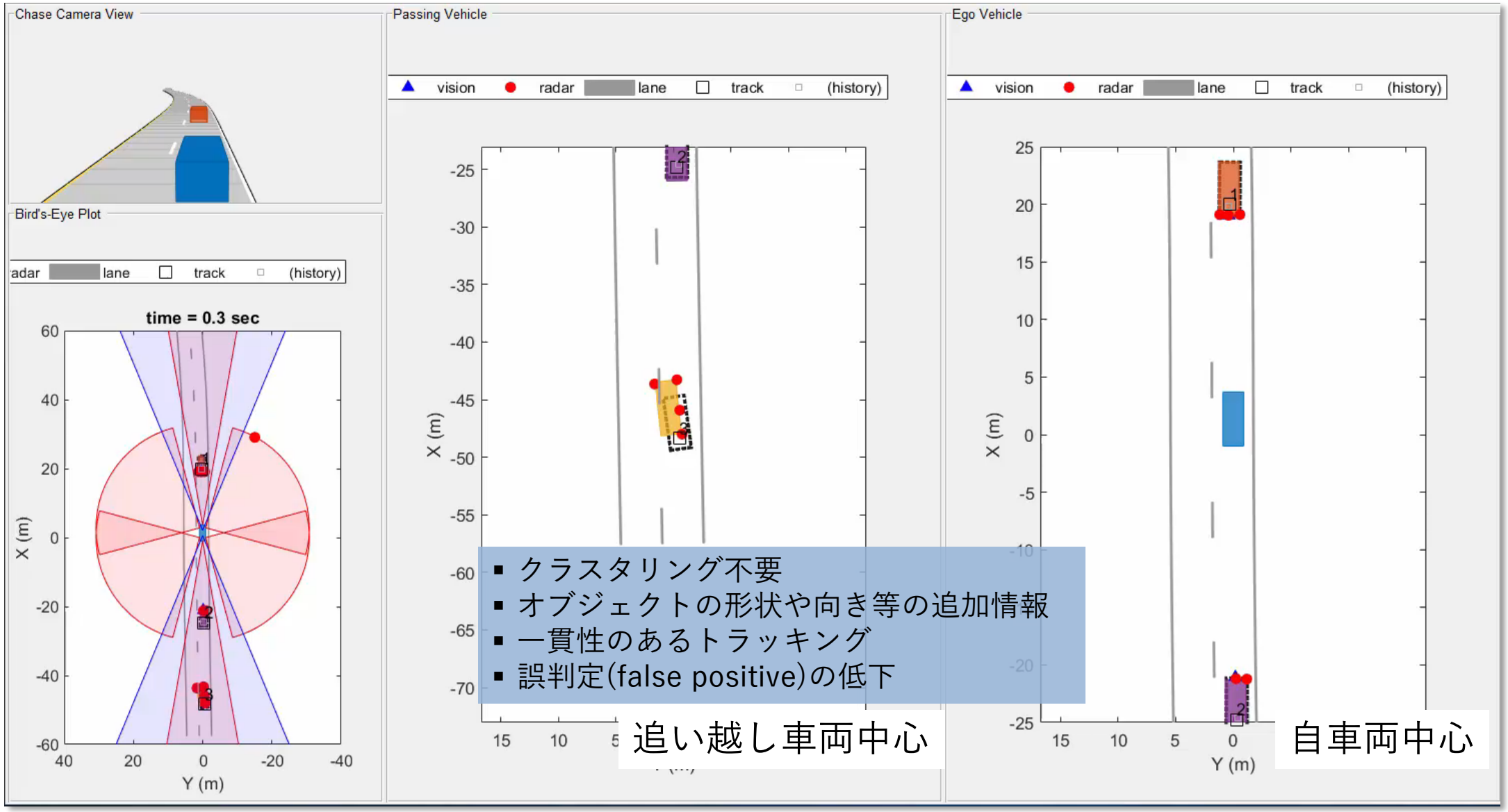
Point Object Tracker



Extended Object Tracker

Automated Driving System Toolbox™
Sensor Fusion and Tracking Toolbox™

MathWorks R2018b



- クラスタリング不要
- オブジェクトの形状や向き等の追加情報
- 一貫性のあるトラッキング
- 誤判定(false positive)の低下

トラッキングパフォーマンスの解析

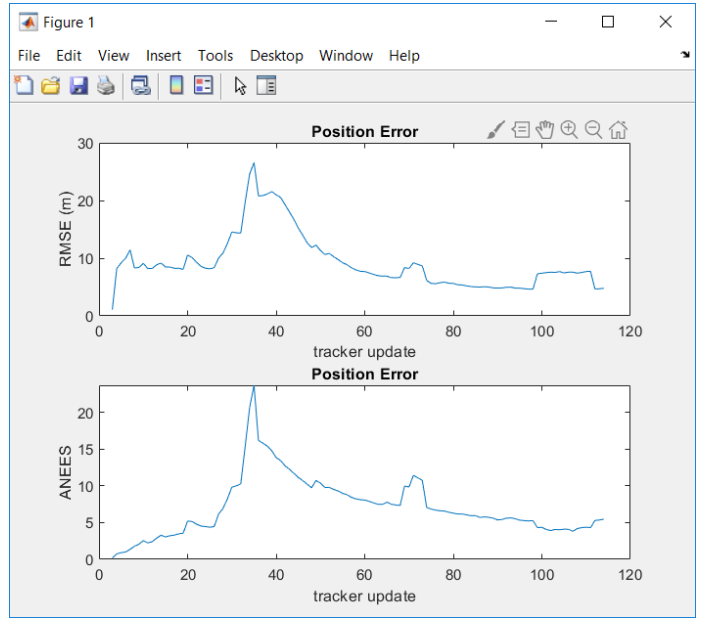
Sensor Fusion and Tracking Toolbox™ R2018b

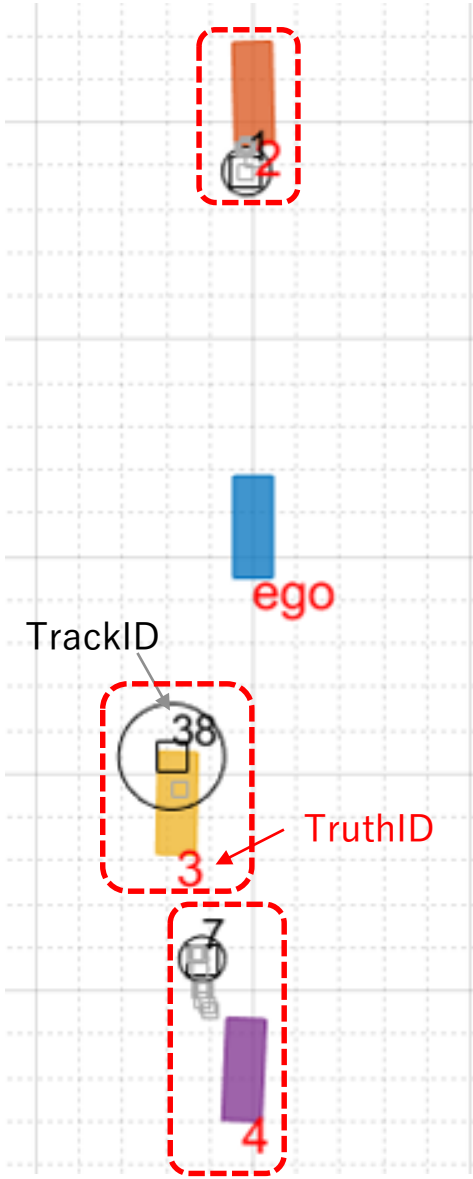
trackAssignmentMetrics

- トラッキングシステムのパフォーマンス解析および比較
- 既知の値(真値)とトラッキングシステムが生成したトラックの比較

trackErrorMetrics

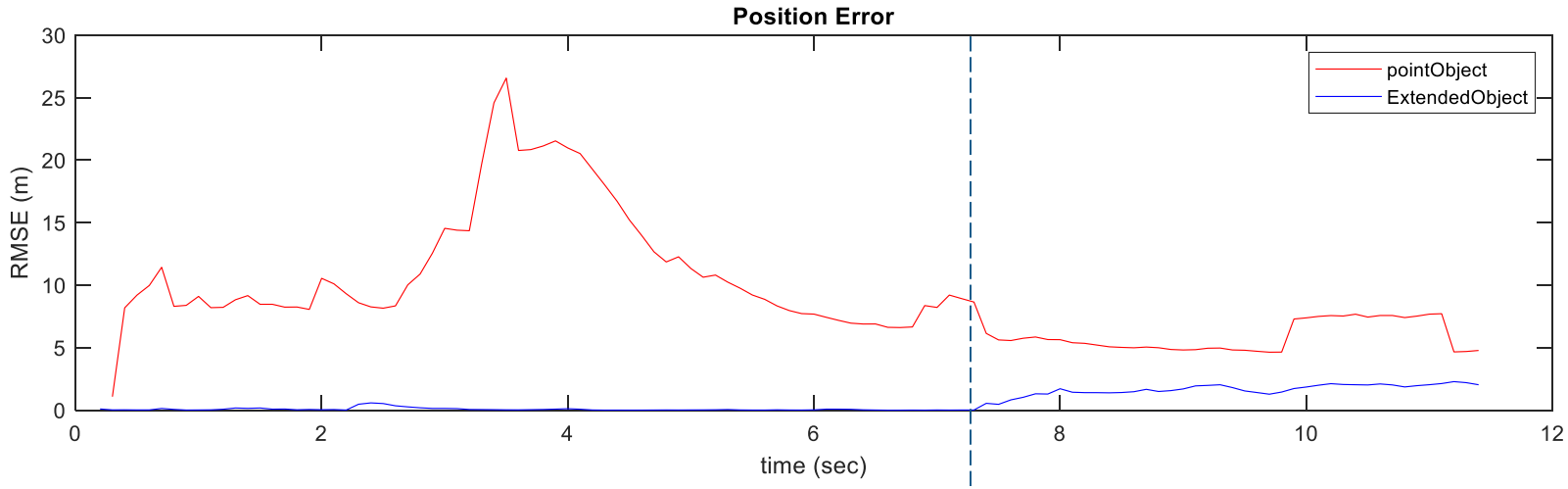
- 各トラックに対するエラーメトリクスの取得



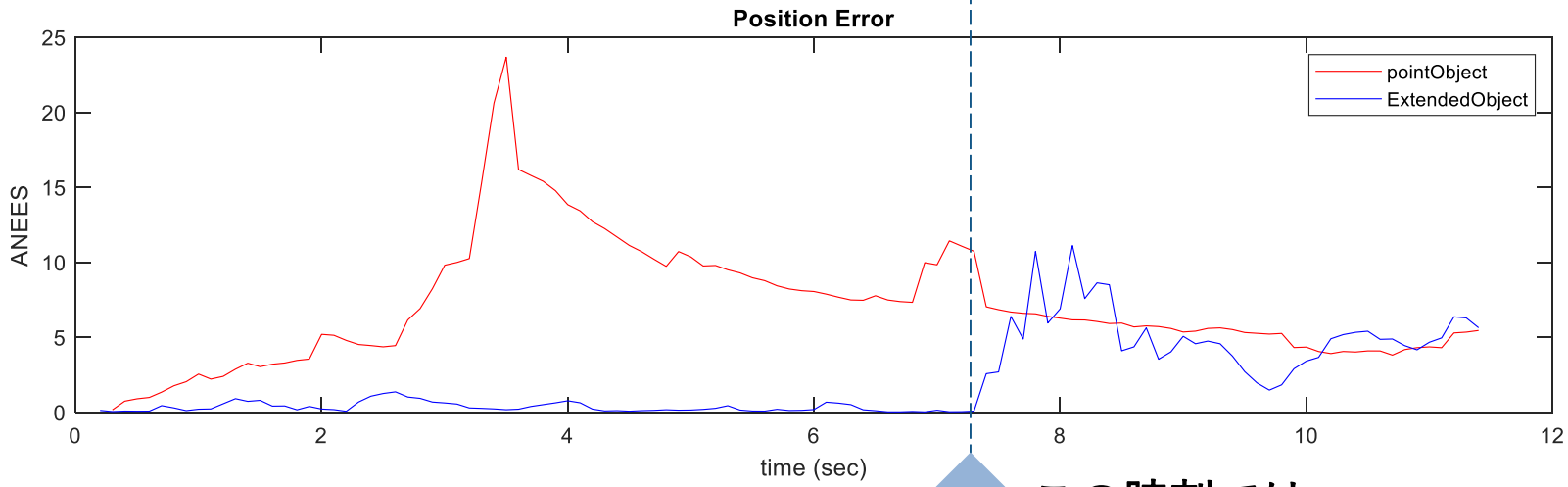


トラックエラー メトリクス

Root Mean Squared Error



Average Normalized-Estimation Error squared

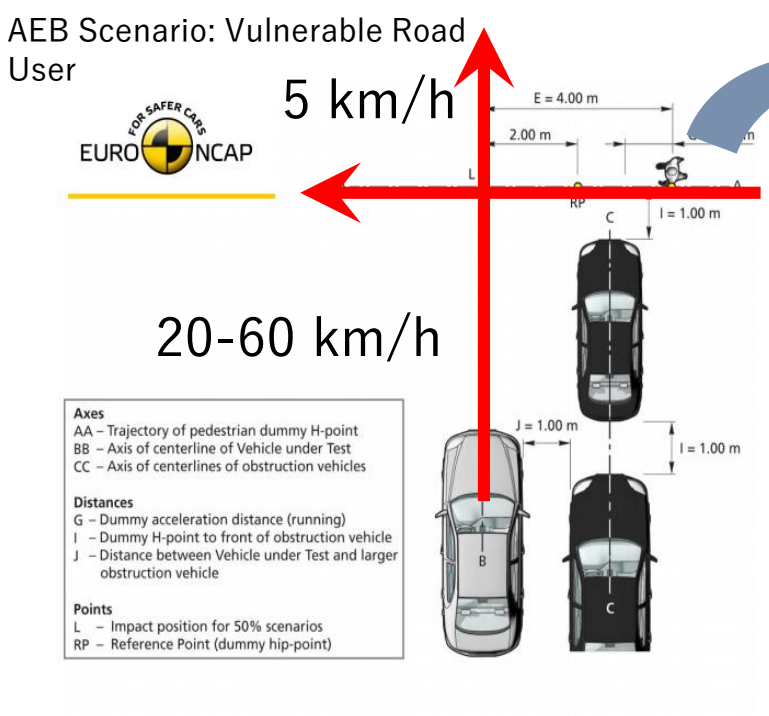


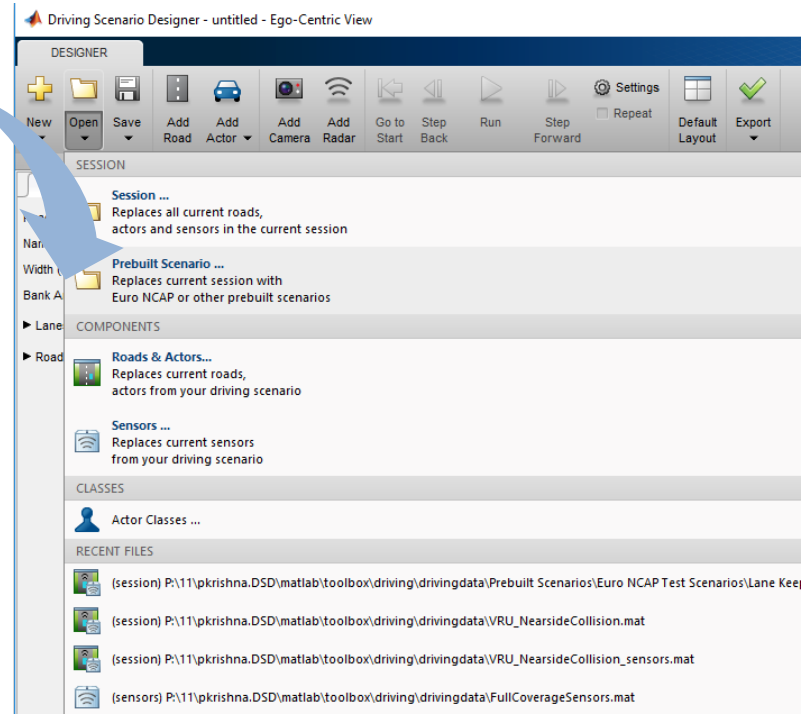
この時刻では、
追い越し車両が自車よりも十分離れている

既存のシナリオの活用

R2018b

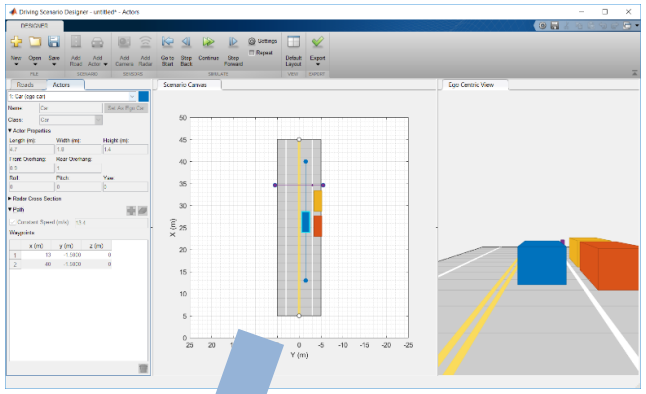
- Euro NCAPのテストシナリオが
  Driving Scenario Designerですぐに利用可能





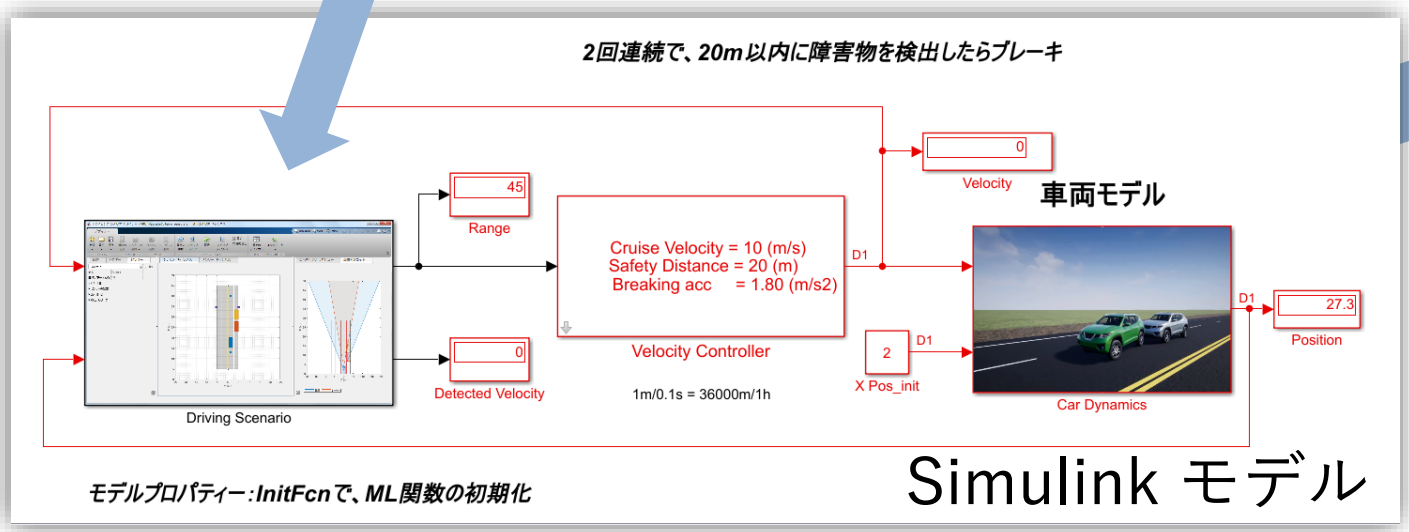
- OpenDRIVE®のroad networkをDriving Scenario Designerに読み込み(.xml, .xodr形式)

Driving Scenario Designerで作成したシナリオの活用

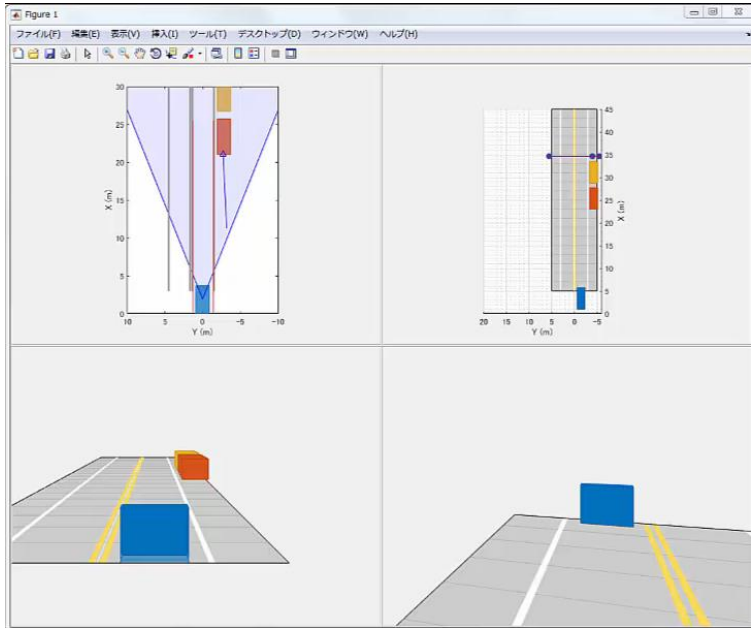


Driving Scenario Designer

関数としてエクスポート



Simulink モデル



シミュレーション実行&
鳥瞰図プロット

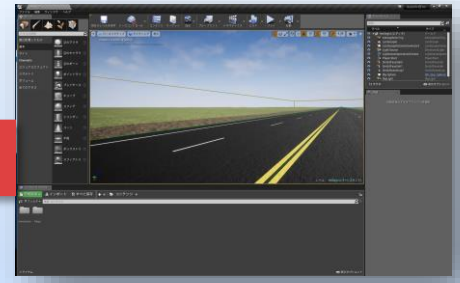
-作成したシナリオは関数としてエクスポート可能、車両ダイナミクスモデルや既存の制御アルゴリズムとSimulink上で統合してシミュレーション可能です。

ゲーミングエンジンとの協調シミュレーション

Vehicle Dynamics Blockset™

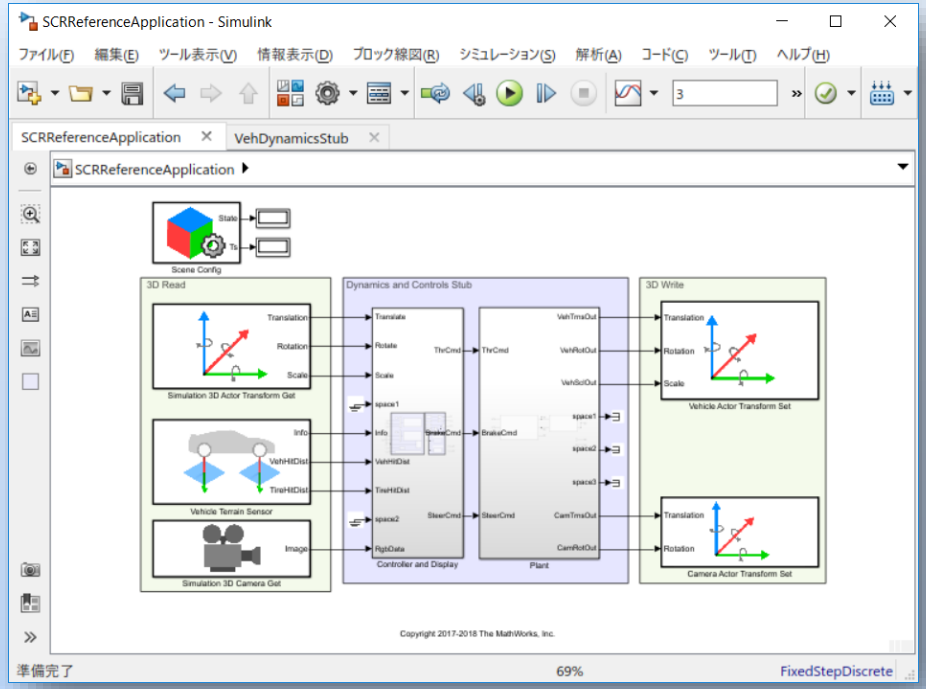
- ゲーミングエンジンと閉ループ構築可能

コンパイル



Unreal Engine
(ゲーミングエンジンのエディター)
ユーザ様インストール

Vehicle Dynamics Blocksetに同梱



Simulinkモデル

カメラモジュール信号(RGB)
車速、
車輪速、
車体傾き、
など

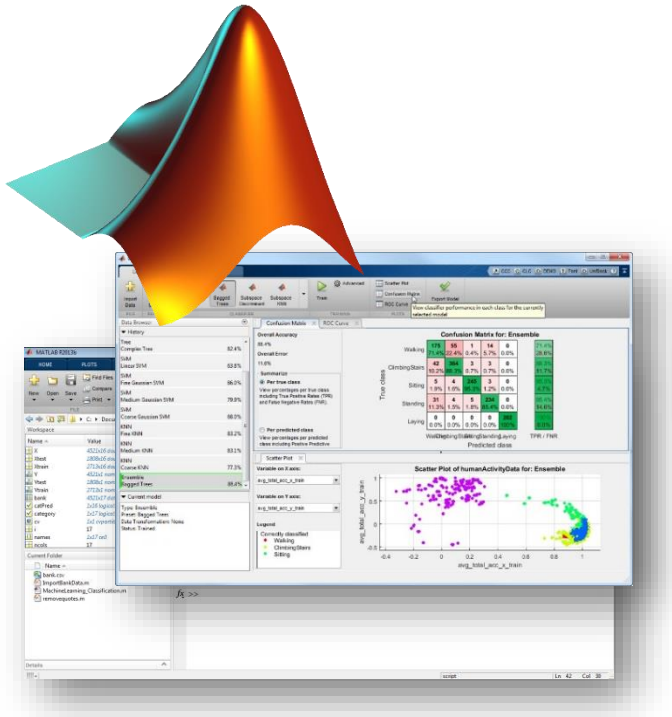
車速、
車輪速、
車体傾き、
など

ゲーミングエンジンの実行ファイル
(コンパイル済みのモデル)

Agenda

- LiDARデータの取り扱い
- センサーフュージョン
- シナリオ生成とシミュレーション
- 豊富な実装オプション
- まとめ

豊富な実装オプション

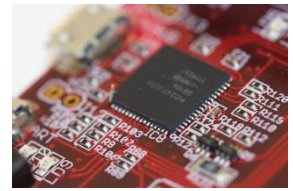


Coders

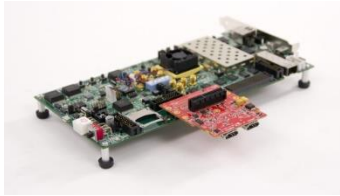
MATLAB Coder™

HDL Coder™

GPU Coder™



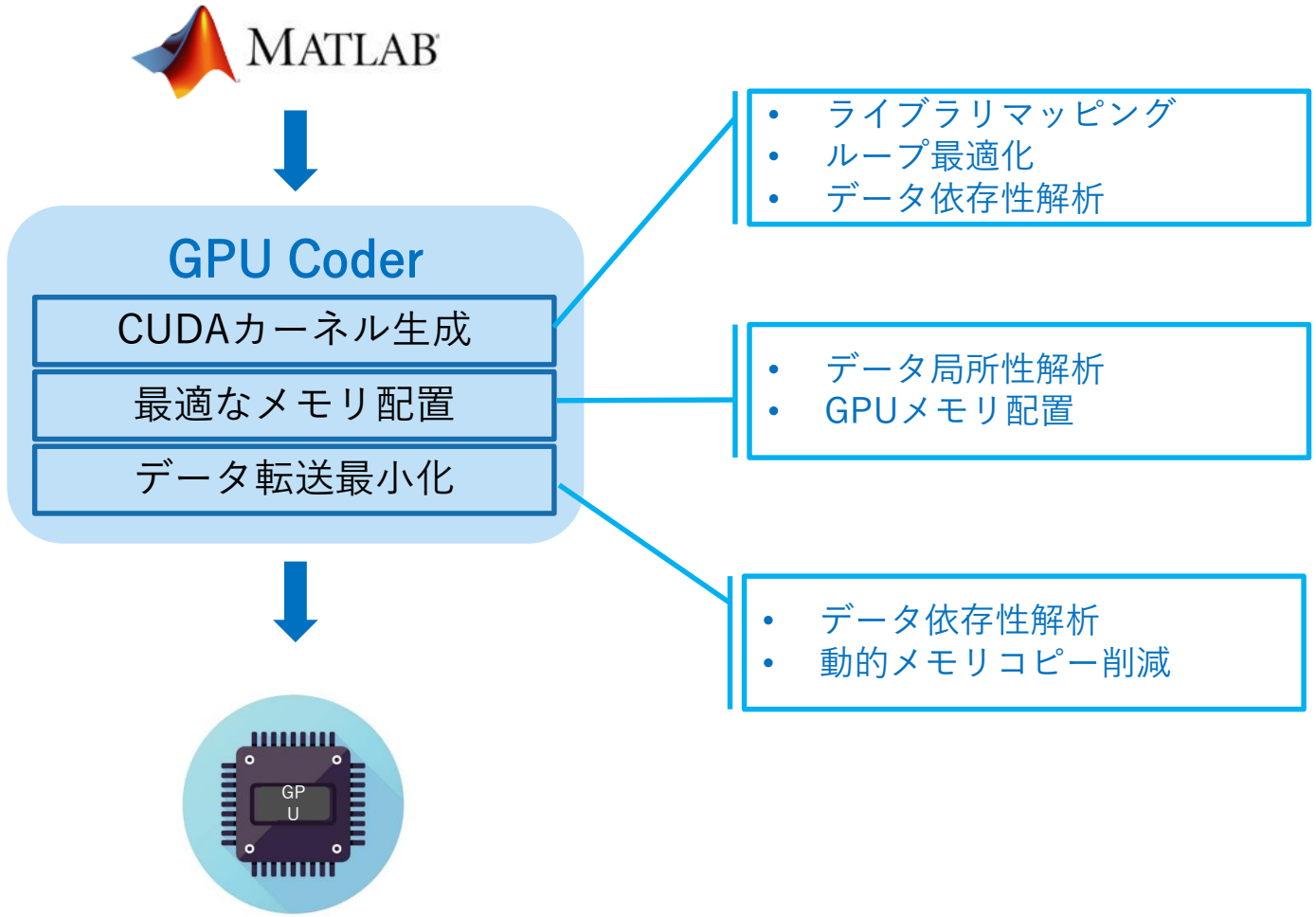
マイコン/DSP



FPGA/ASIC

NVIDIA® GPU

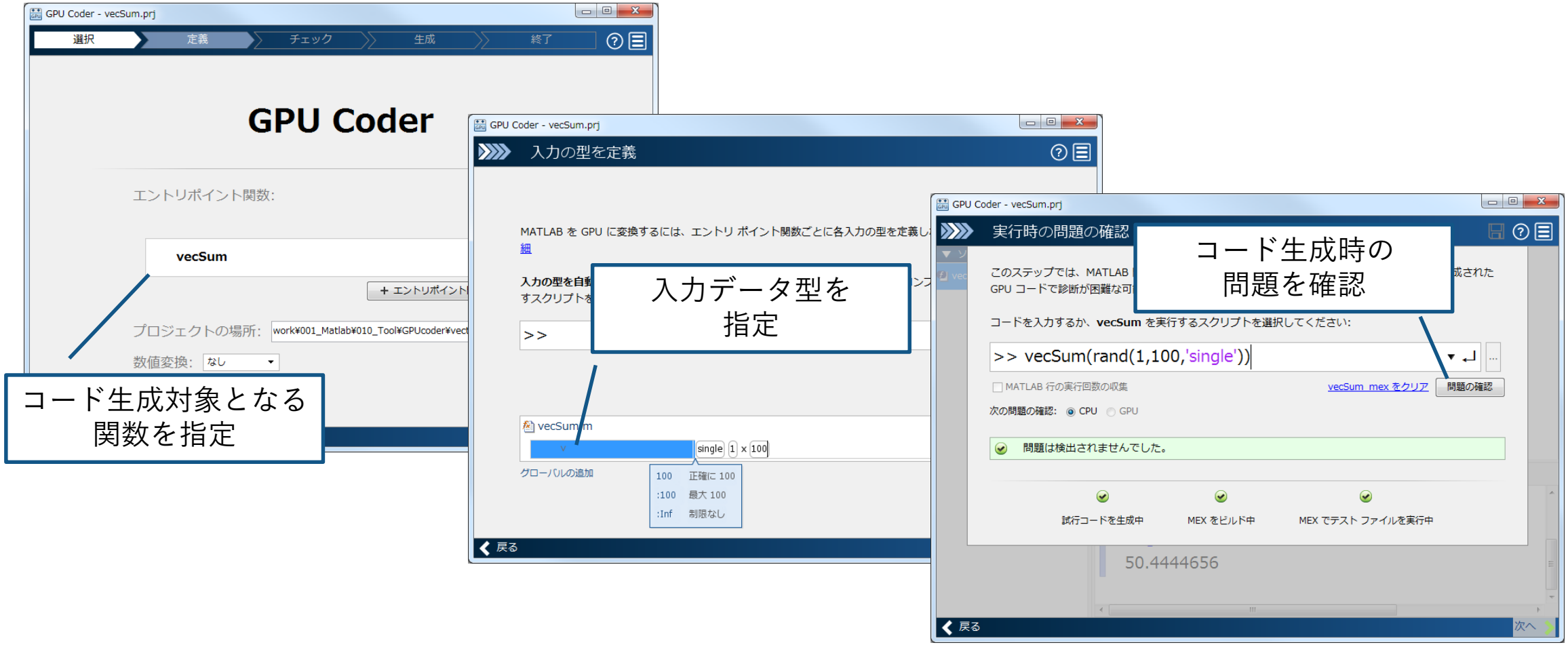
GPU Coder : 高度な関数解析機能が効率の良いコード生成を実現



MATLABからCUDA C/C++コードを自動生成します

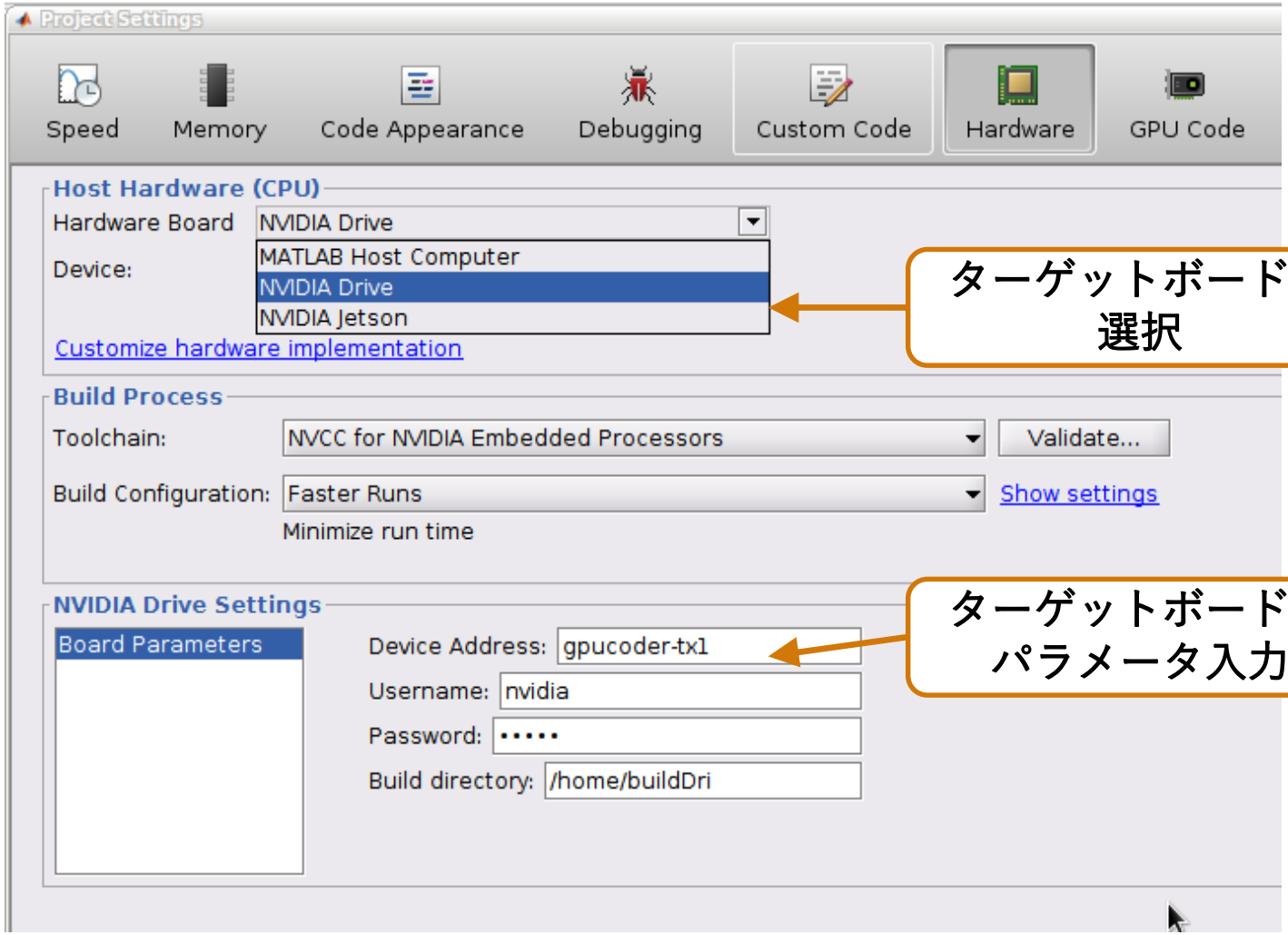
専用Appを利用したコード生成

MATLAB Coder™
GPU Coder™



NVIDIA Hardware Support Package (HSP)

GPU Coder™
R2018b



ターゲットボードの選択

ターゲットボードのパラメータ入力

MATLABから直接ボードにアクセス可能

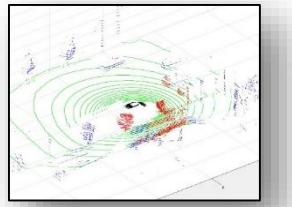
Jetson

Drive platform

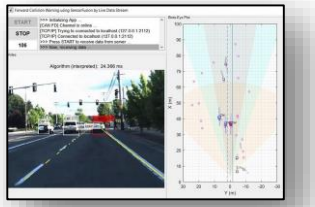
Agenda

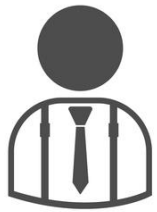
- LiDARデータの取り扱い
- センサーフュージョン
- シナリオ生成とシミュレーション
- 豊富な実装オプション
- まとめ

Key Takeaways : ADAS・自動運転開発・検証の統合プラットフォーム

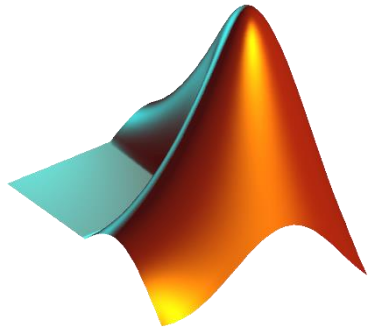


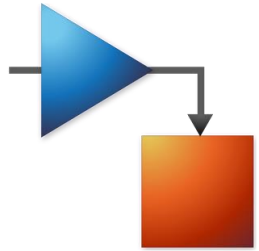
LiDAR処理
担当





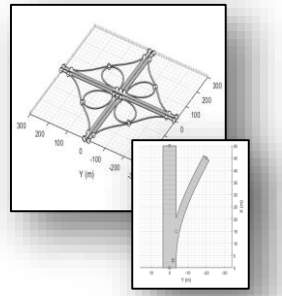
センサー
フュージョン
担当





実装担当

検証担当



-カスタマイズ可能な多くのツール・関数群、MATLAB/Simulink上でシームレスに動作
-既存の資産や各種オプション製品との連携
-アルゴリズム開発からコード生成、実装までを網羅する統合開発環境

Automated Driving System Toolbox™

ADAS/自動運転開発・検証の統合プラットフォーム