

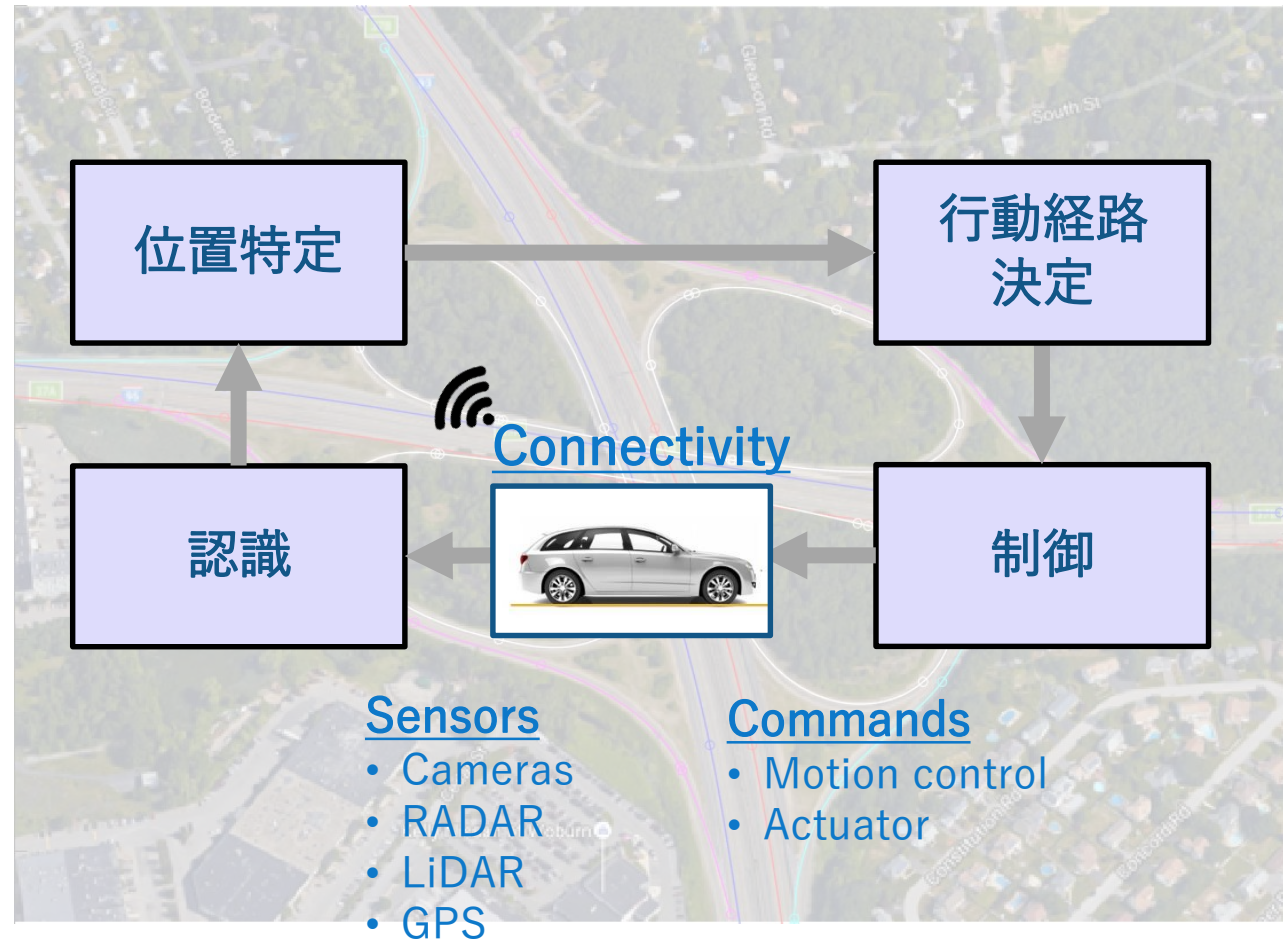
MATLAB EXPO 2018

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



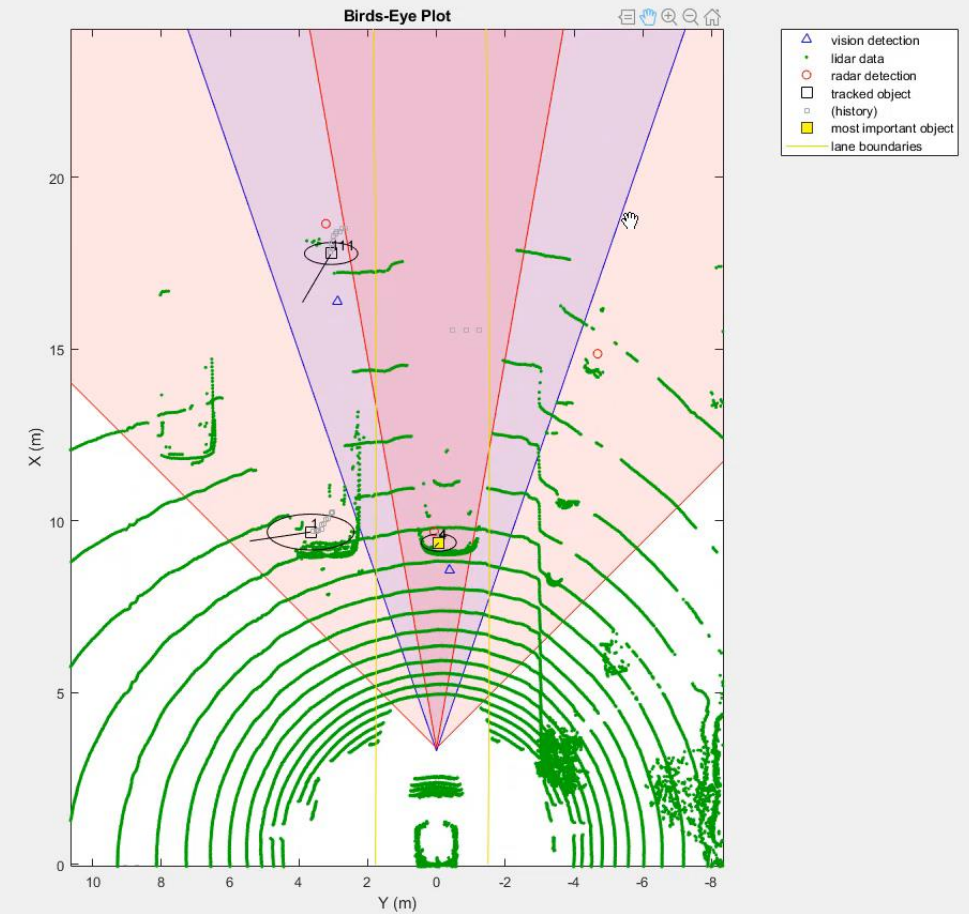
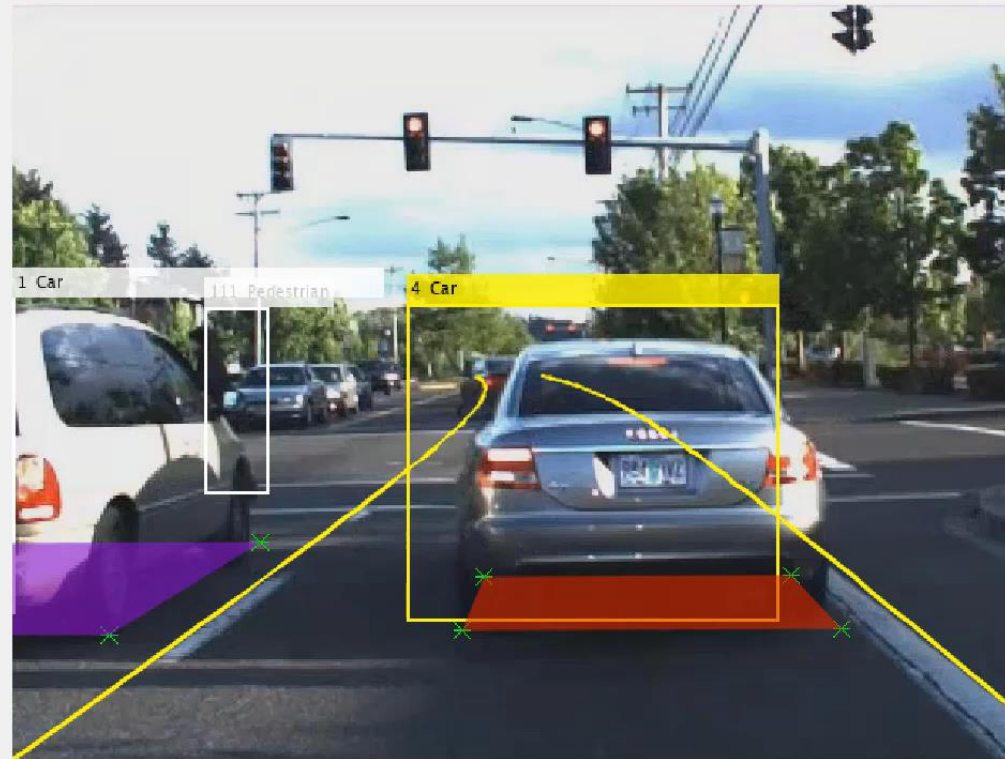
前方車兩接近警報(FCW)

Figure 1: Forward Collision Warning With Tracking Example

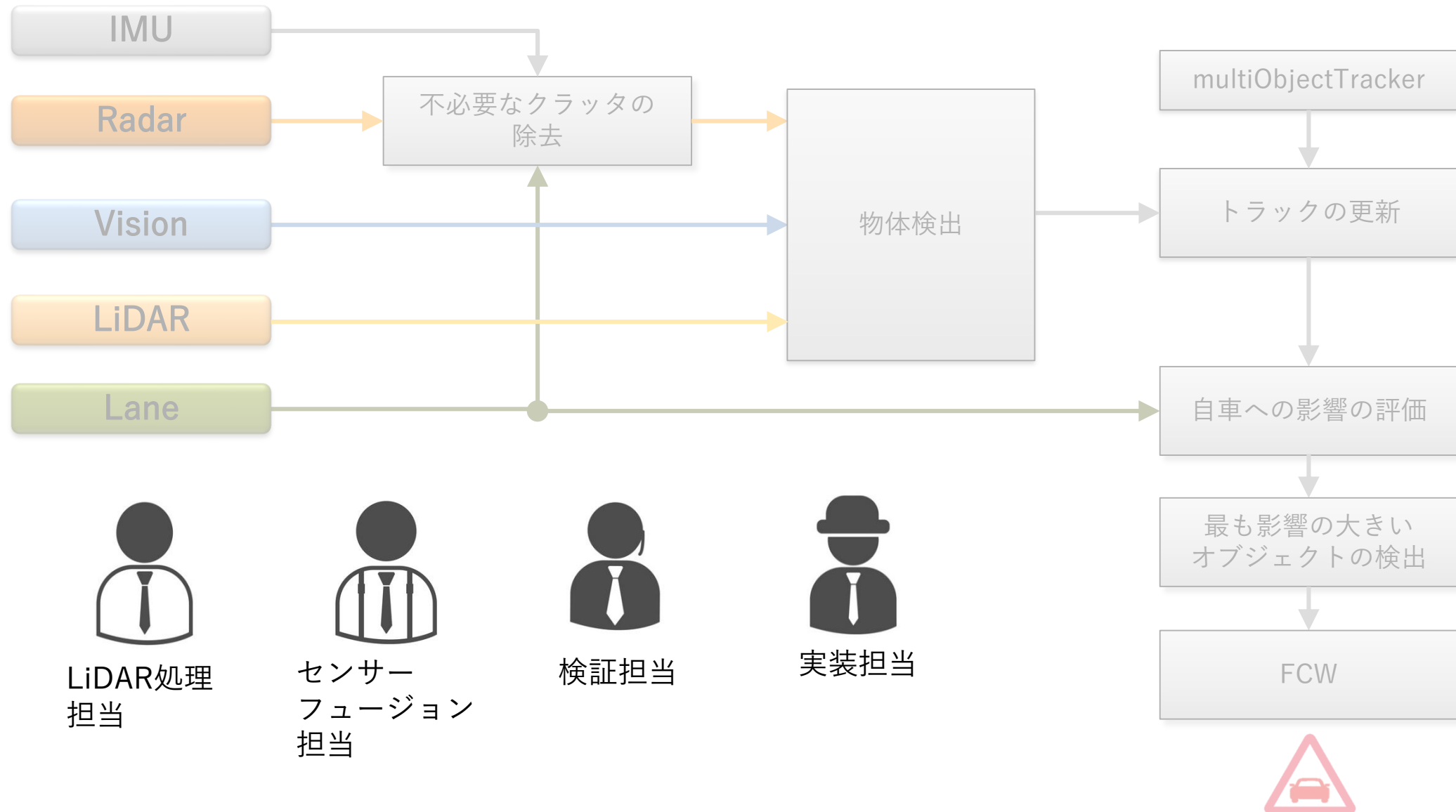
File Edit View Insert Tools Desktop Window Help



Recorded Video



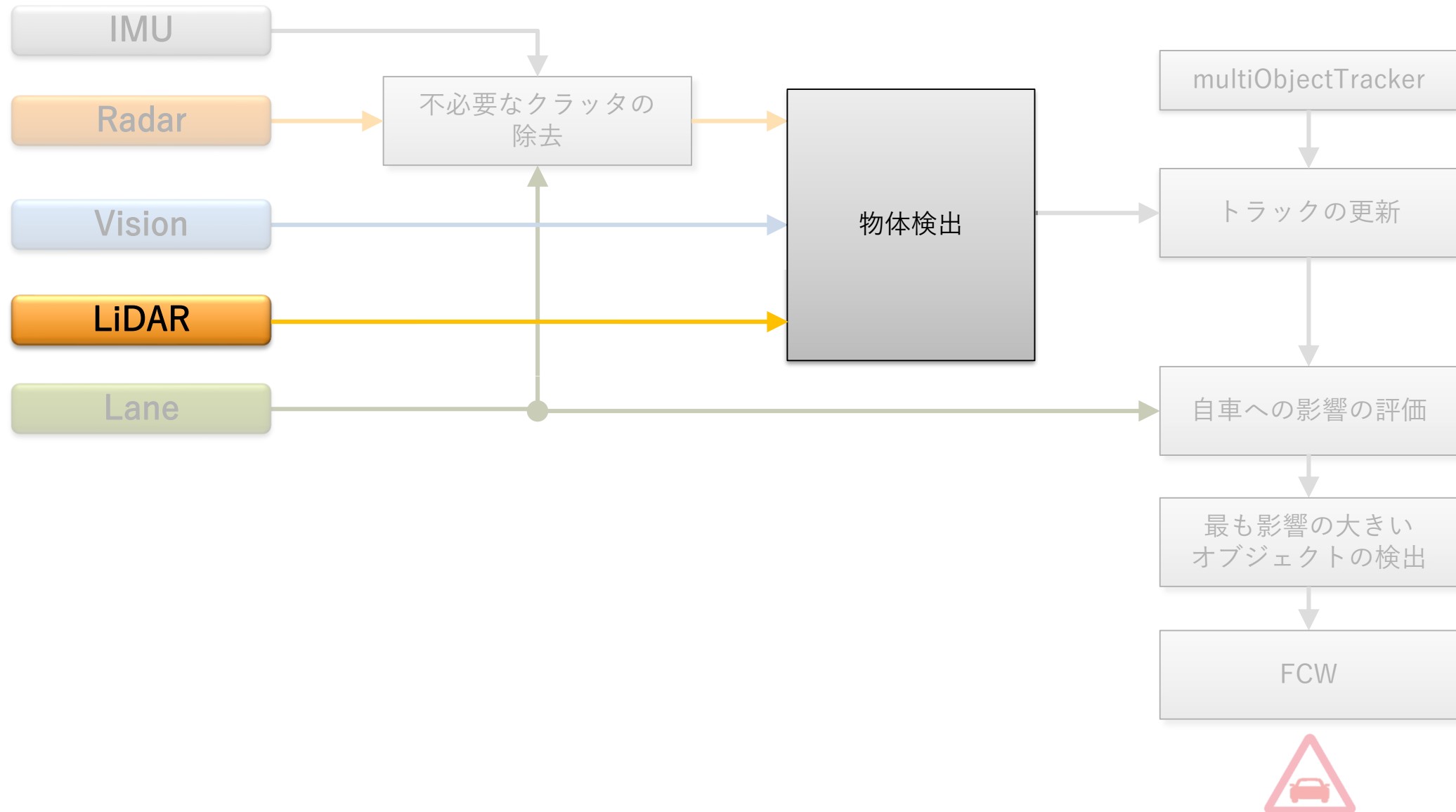
前方車両接近警報(FCW)



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

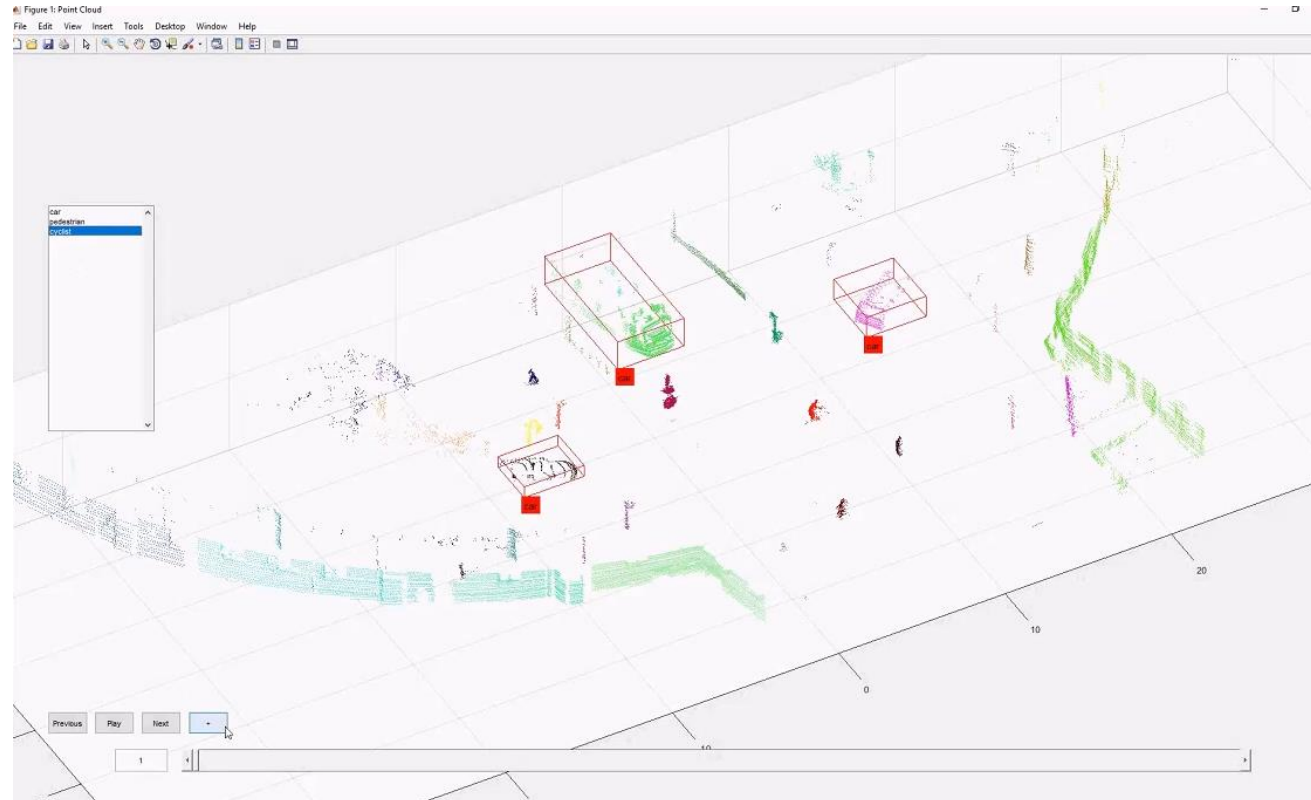
Image Processing Toolbox™
Computer Vision System Toolbox™

ファイル入出力

- pointCloud
- **velodyneFileReader**
- pcread
- pcwrite

前処理

- pcdenoise
- pcdownsampling
- pctransform



位置合わせ

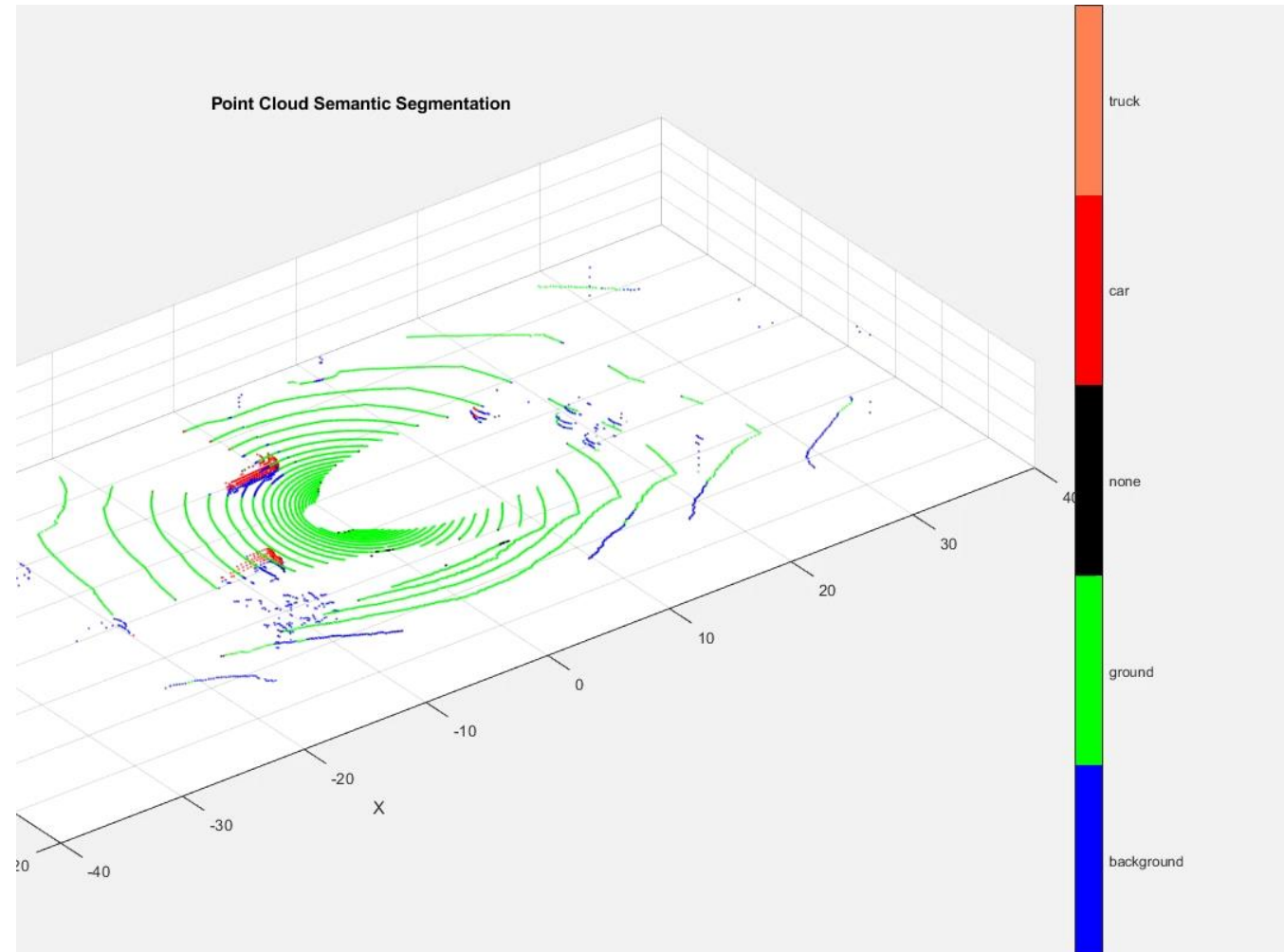
- pcregistericp
- pcregisterndt
- **pcregistercpd**

セグメンテーション

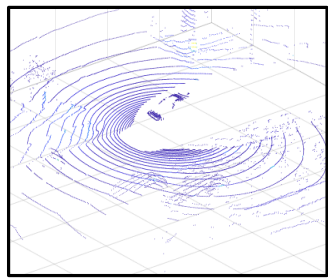
- pcsegdist
- segmentLidarData
- pcfitplane
- **segmentGroundFromLidarData**

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

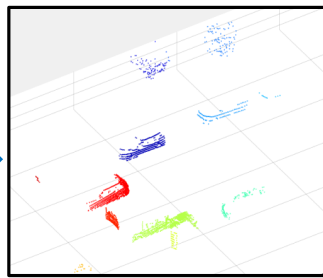
LiDAR Semantic Segmentation



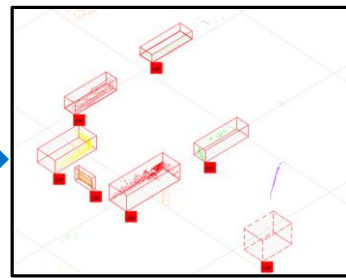
データの前処理・ラベリング



データアクセス



前処理



ラベリング

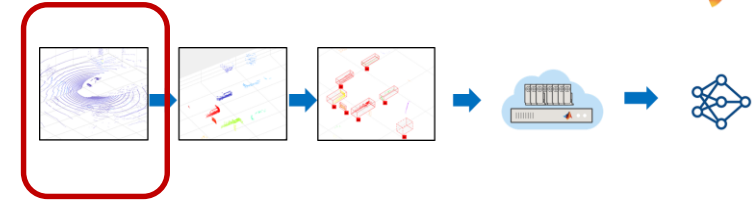


学習



学習済
ネットワーク

LiDARデータへのアクセス、可視化

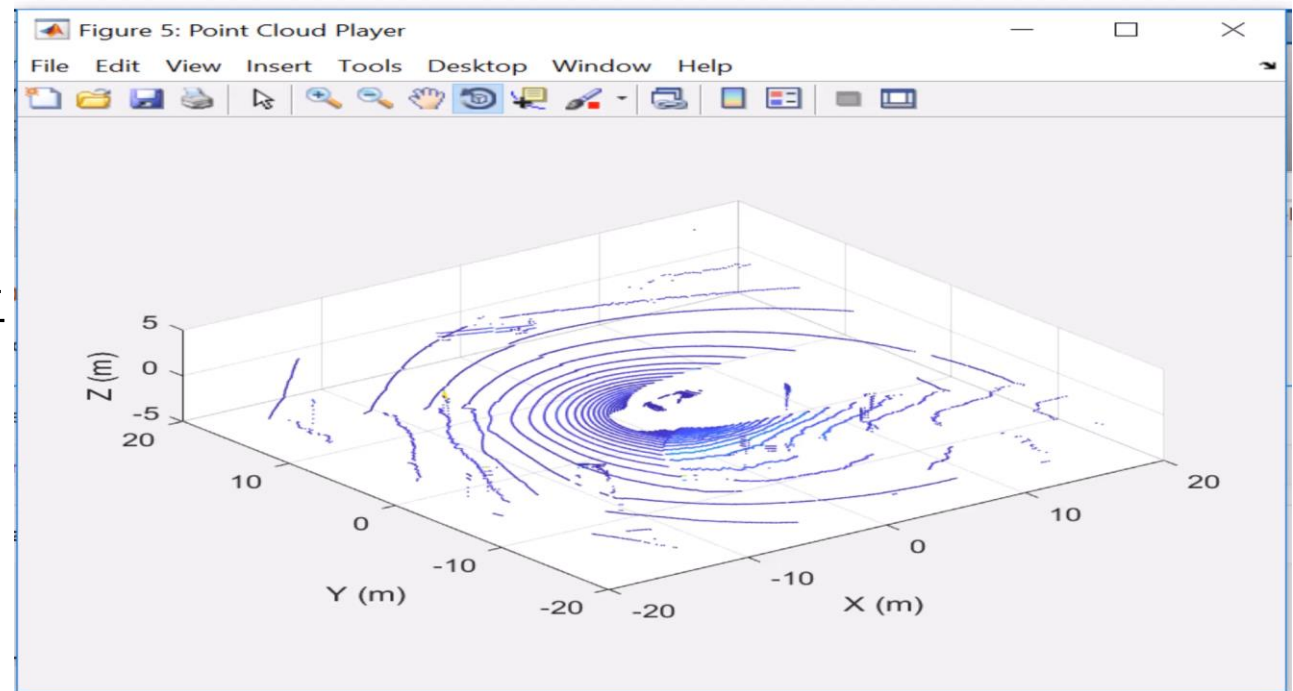


LiDAR データへのアクセス

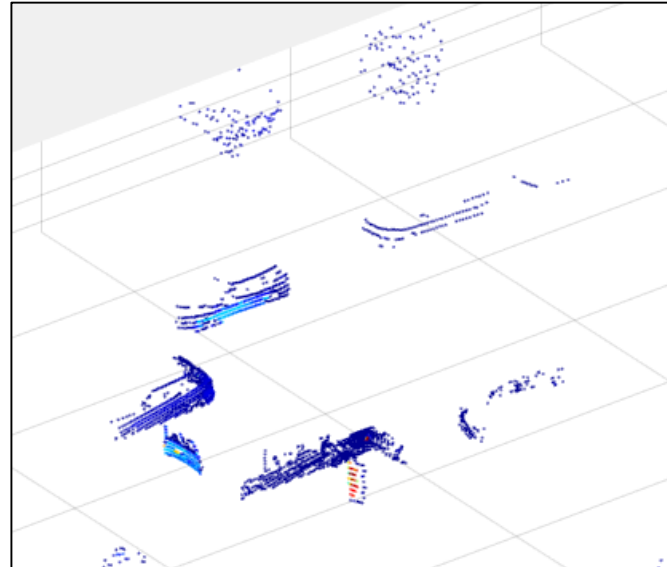
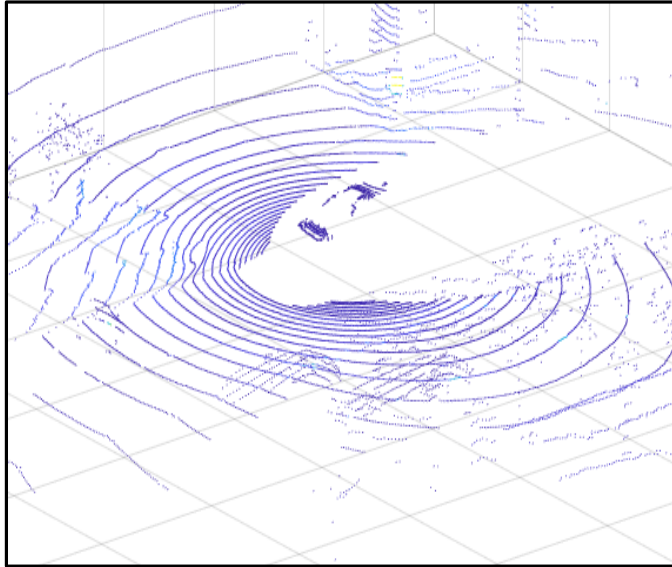
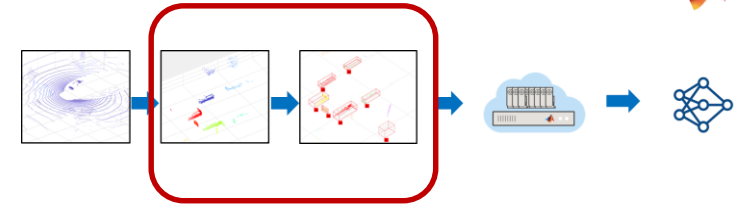
- Velodyne専用ファイル (pcap)
- 点群データフォーマット (.pcd,ply)
- バイナリファイルI/O

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

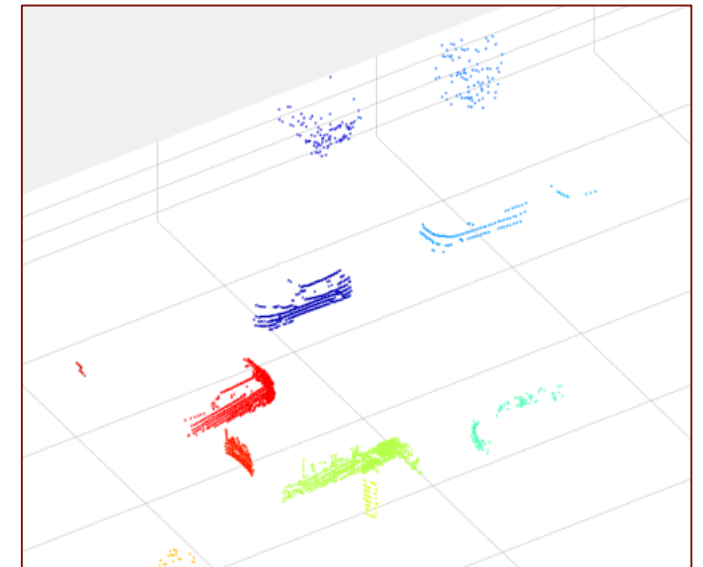


LiDAR データ前処理



地表面の除去

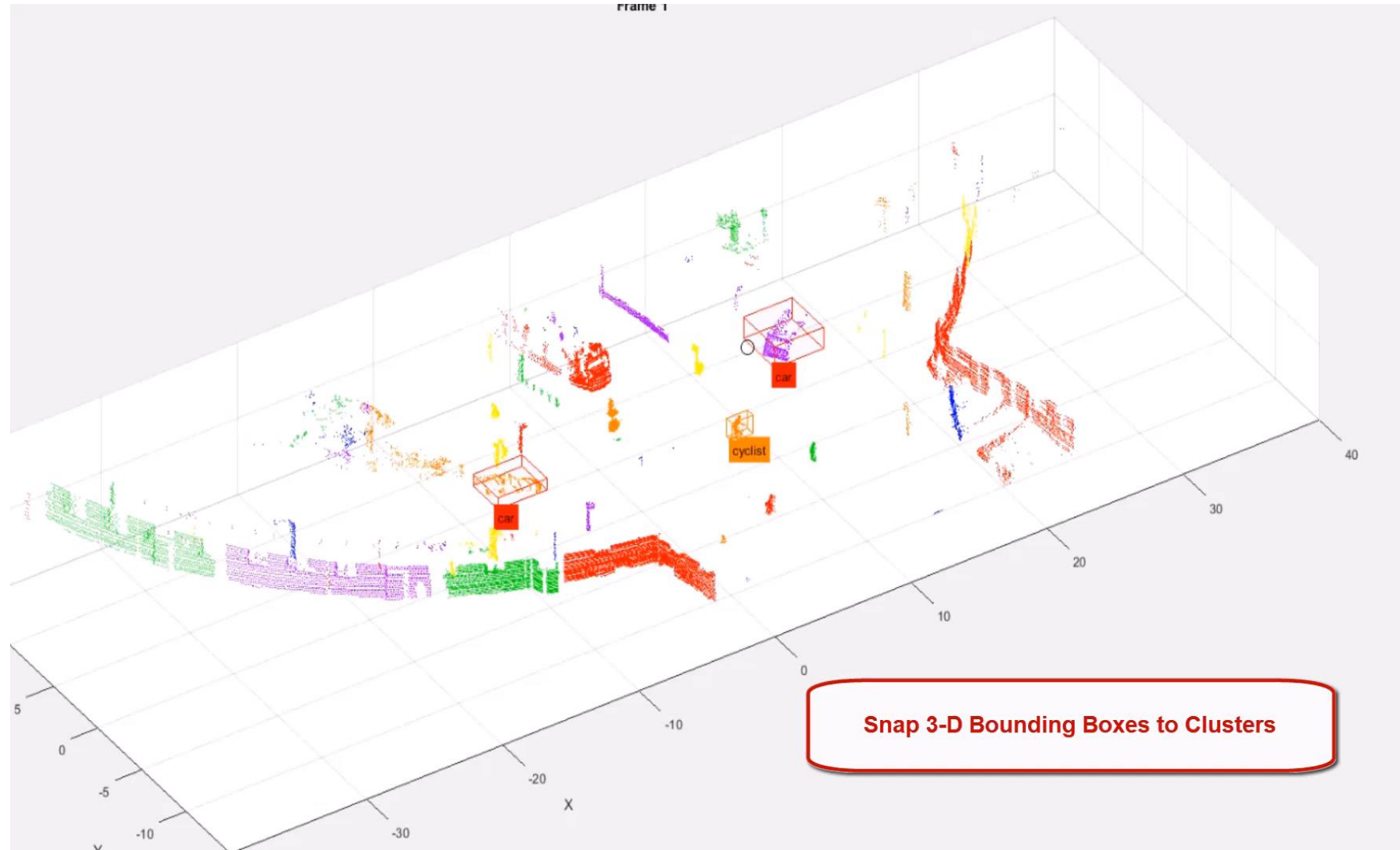
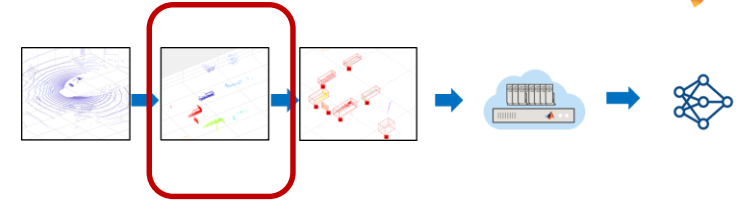
- RANSACを利用したモデルフィッティング



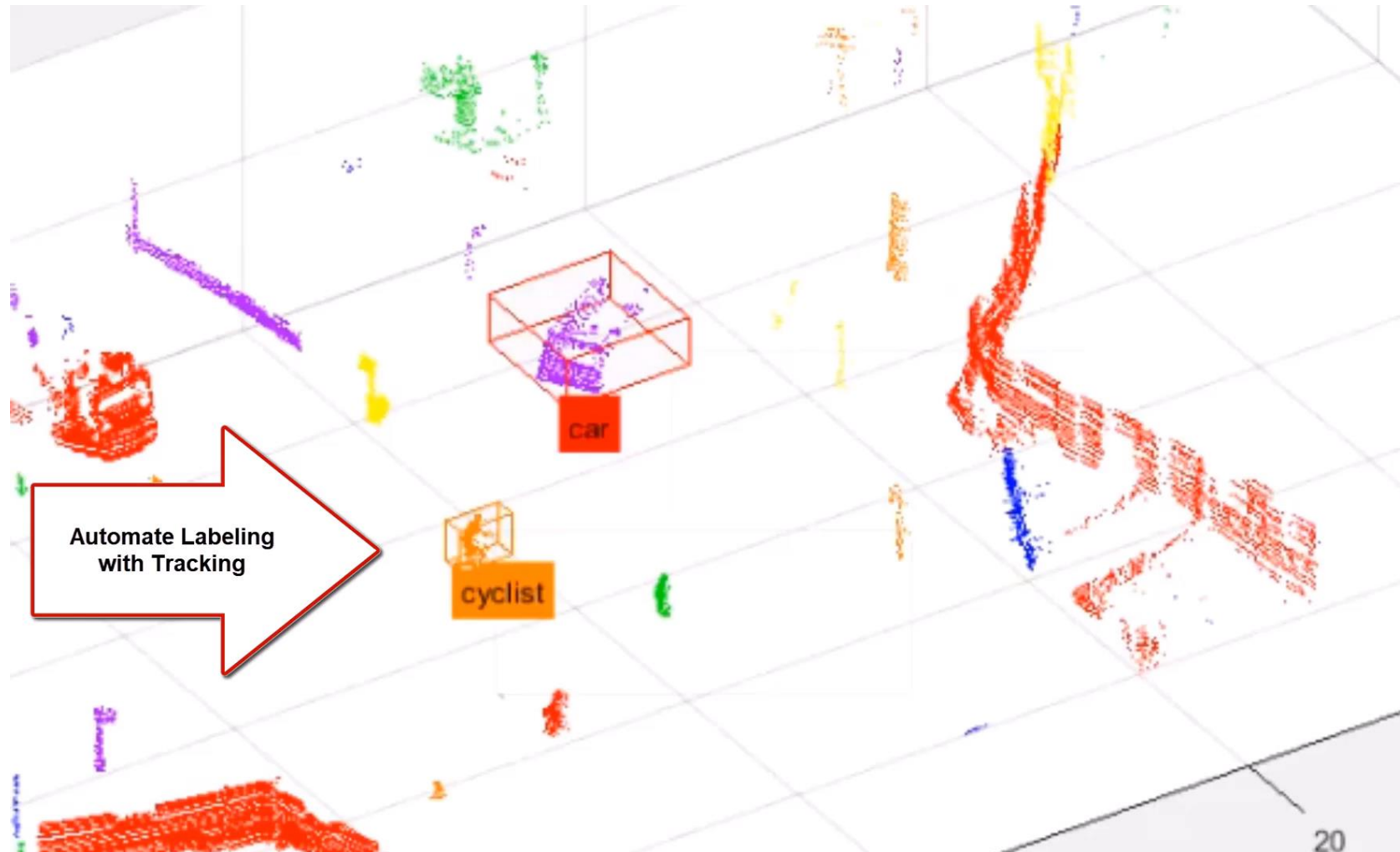
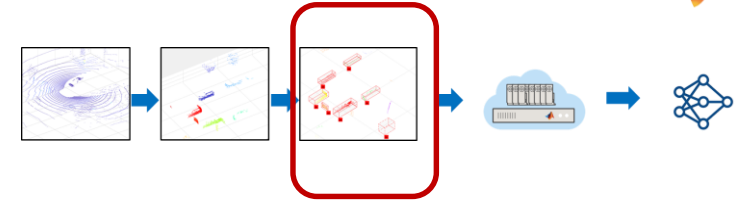
セグメンテーション

- ユークリッド距離を利用したクラス作成

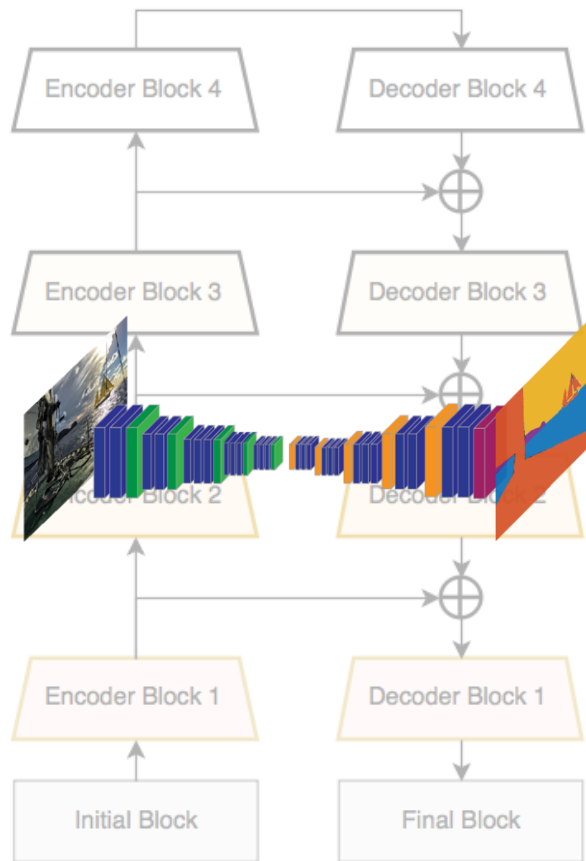
LiDAR データ 真値ラベリング



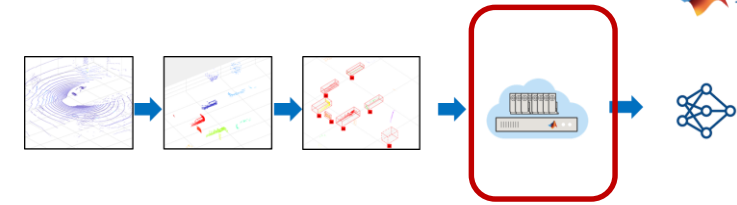
LiDAR データ 真値ラベリング



ネットワークの定義

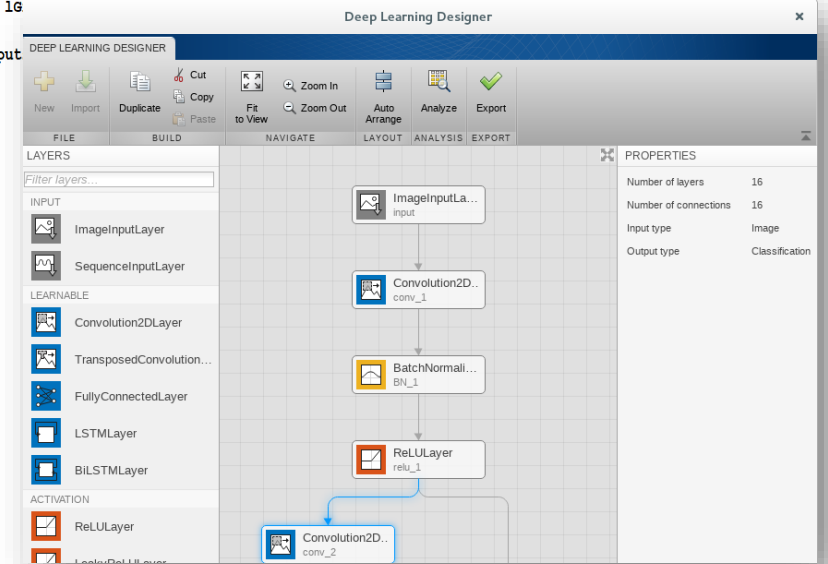


Easy MATLAB API to
create network

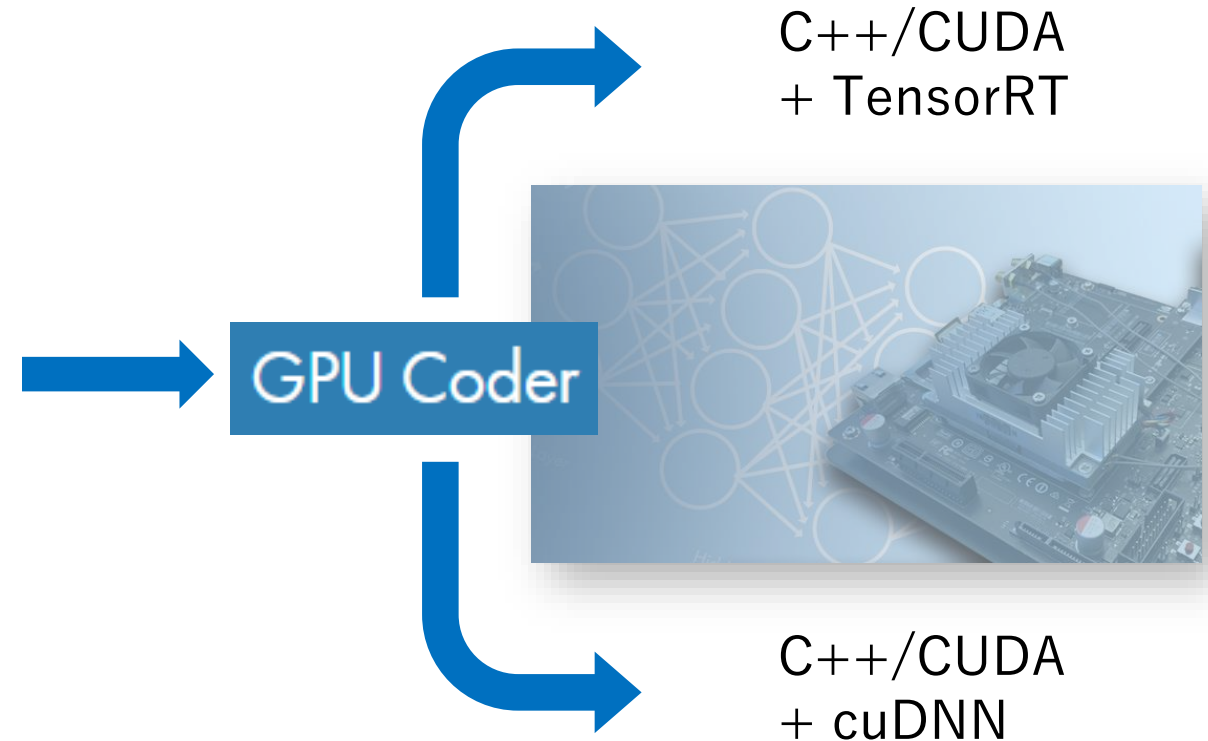
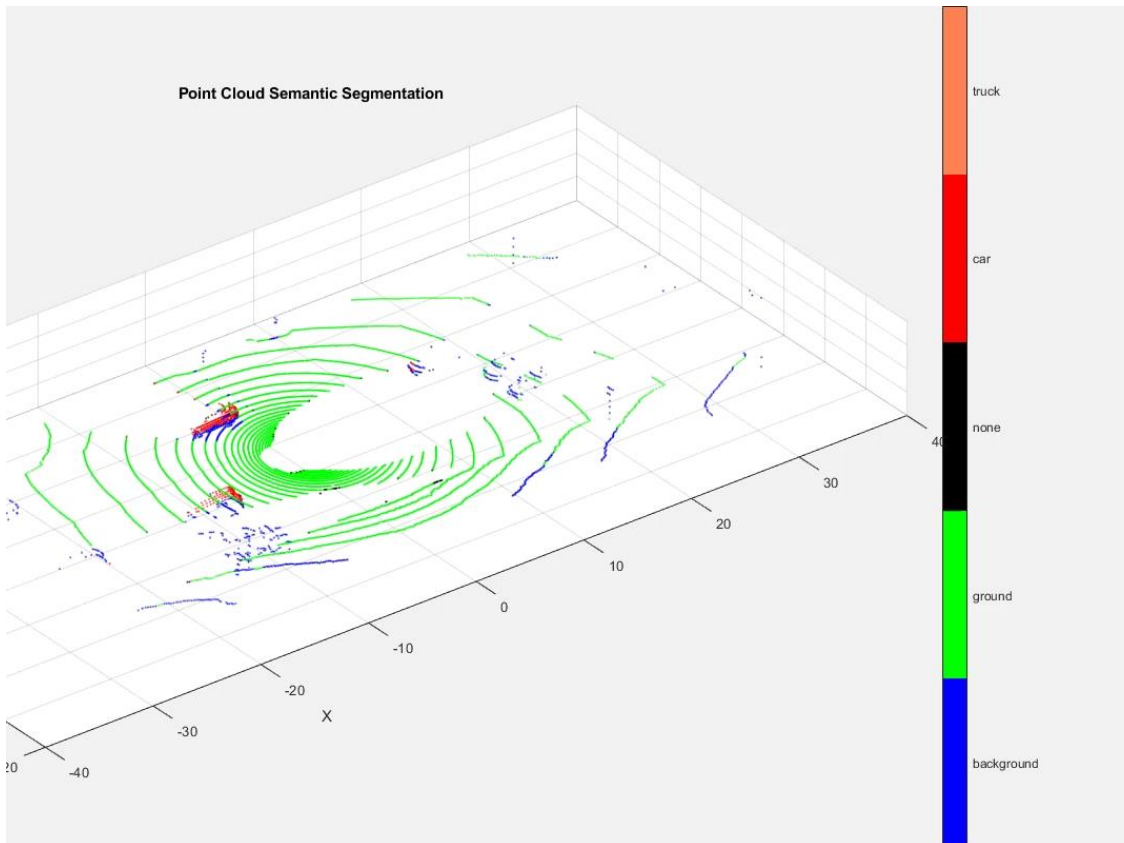


```
%build encoder
nOutputs = 64;
inputLayerName = 'init_maxpool';
for blockIdx = 1:encoderDepth
    [lGraph, layerOutName] = encoderBlock(lGraph, blockIdx, nOutputs, inputLayerName);
    nOutputs = nOutputs * 2;
    inputLayerName = layerOutName;
end

%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']);
    end
    nInputs = nOutputs;
end
lG
```



GPU Coderを利用した組み込み機器への実装



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

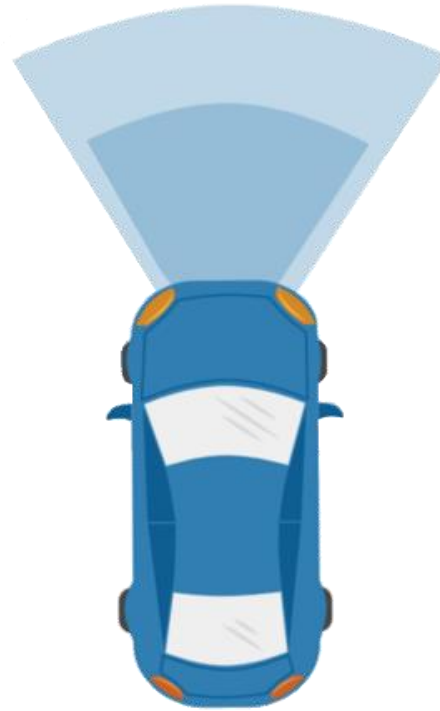
レーダー

カメラ：物体検出

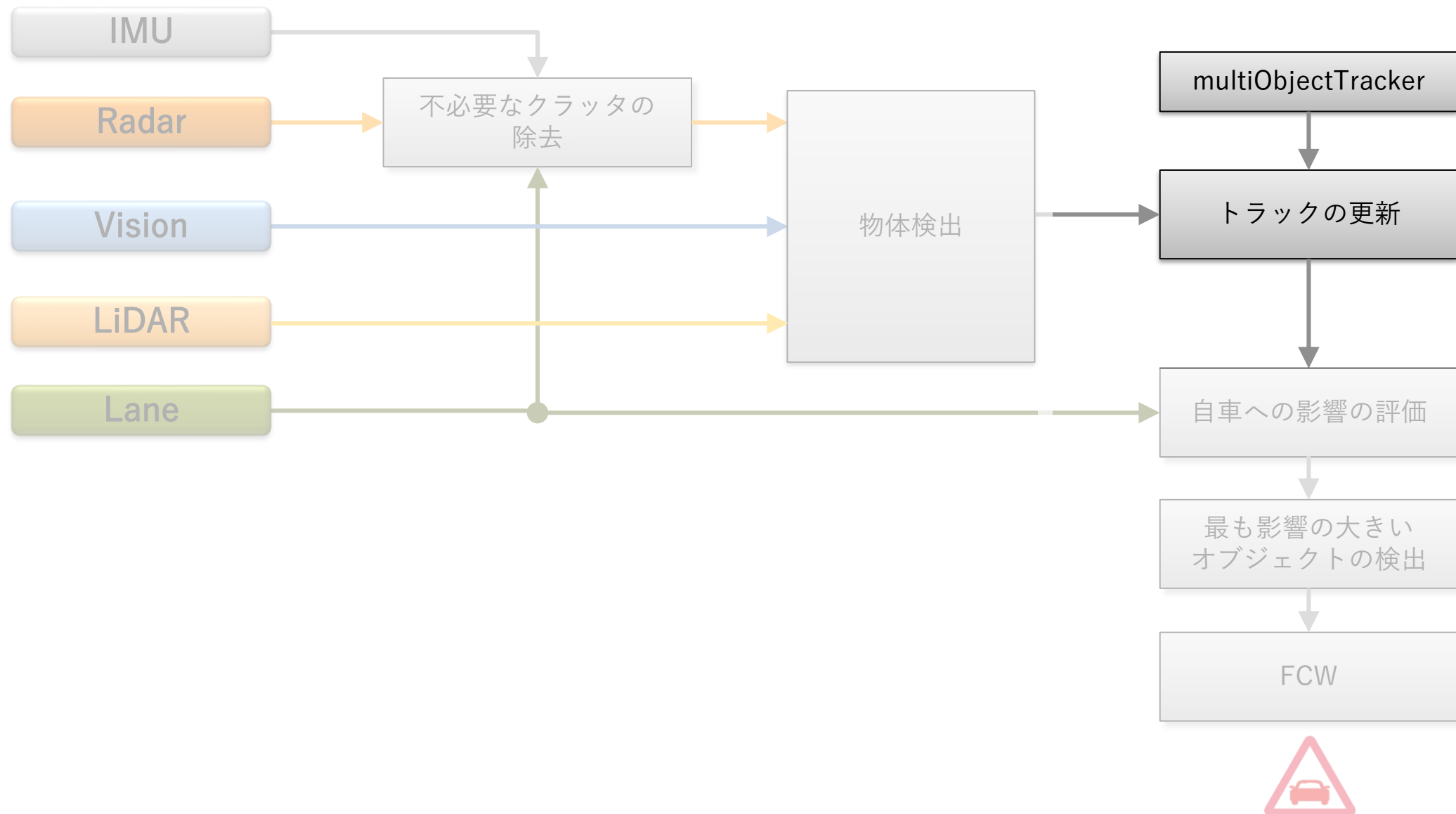
LiDAR：物体検出

カメラ：区画線検出

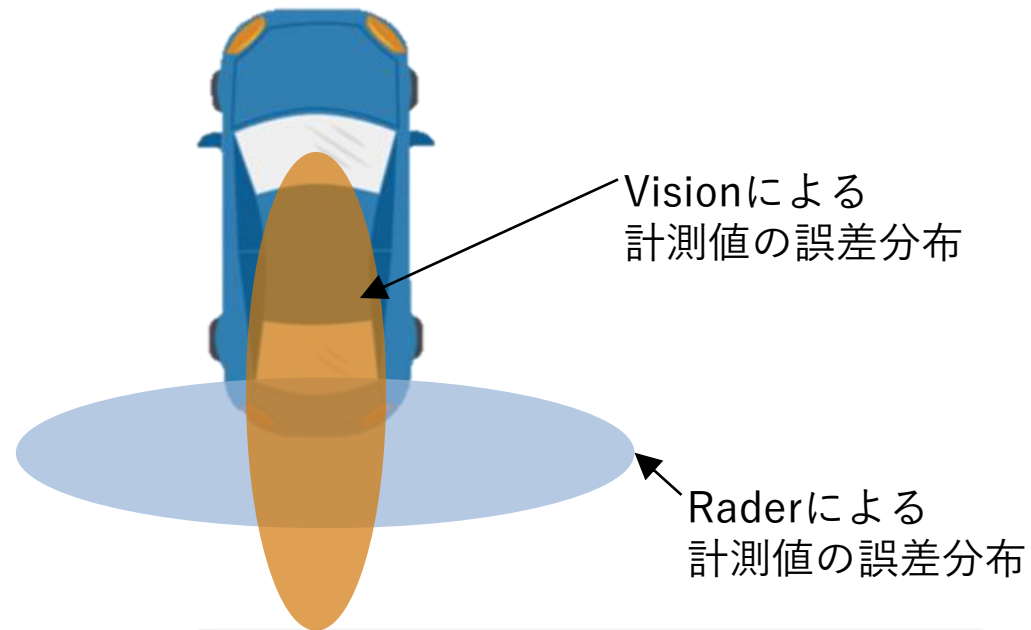
IMU（角速度等）



前方車両接近警報(FCW)

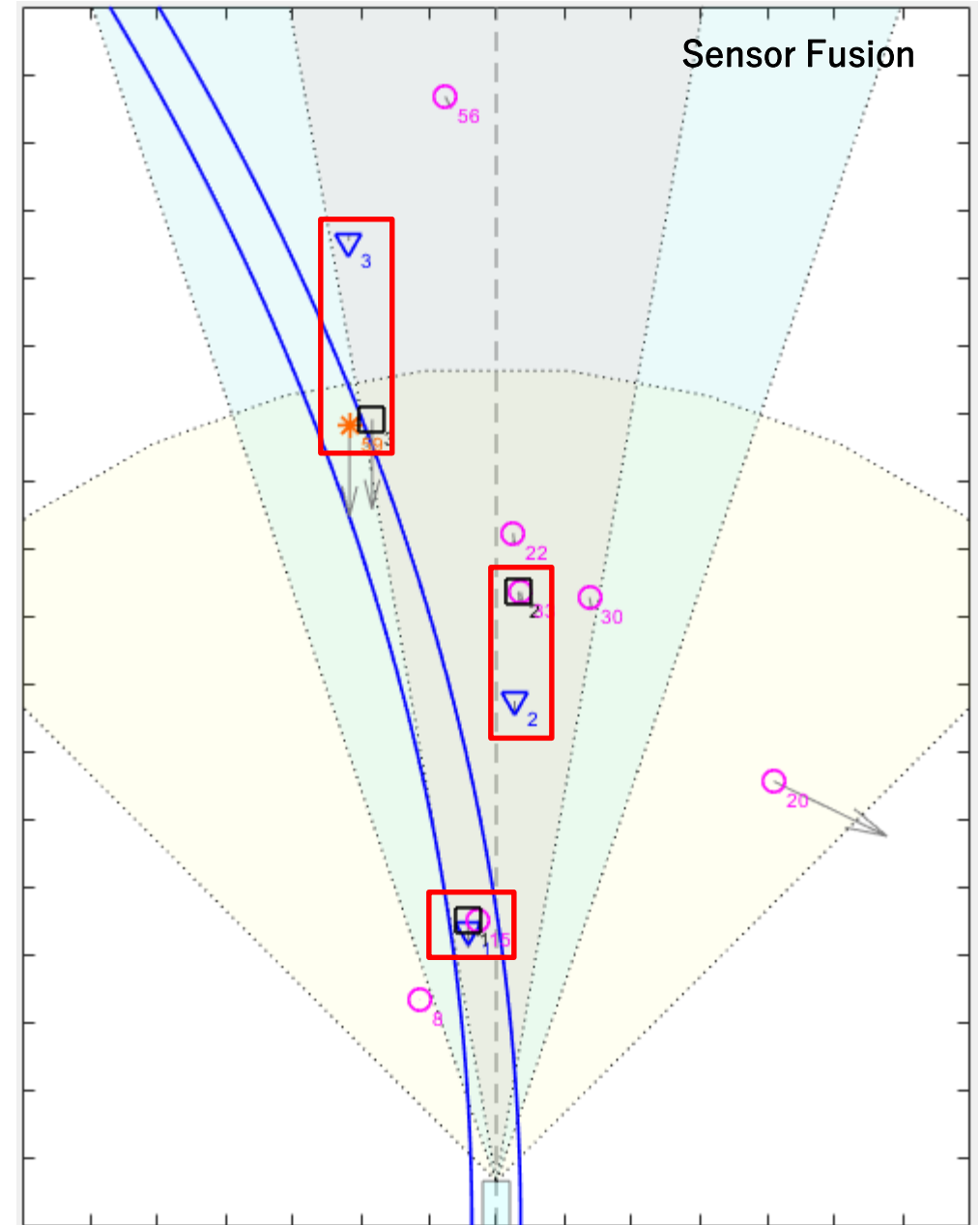


センサーフュージョンの重要性について

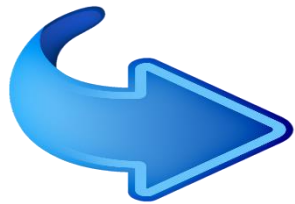
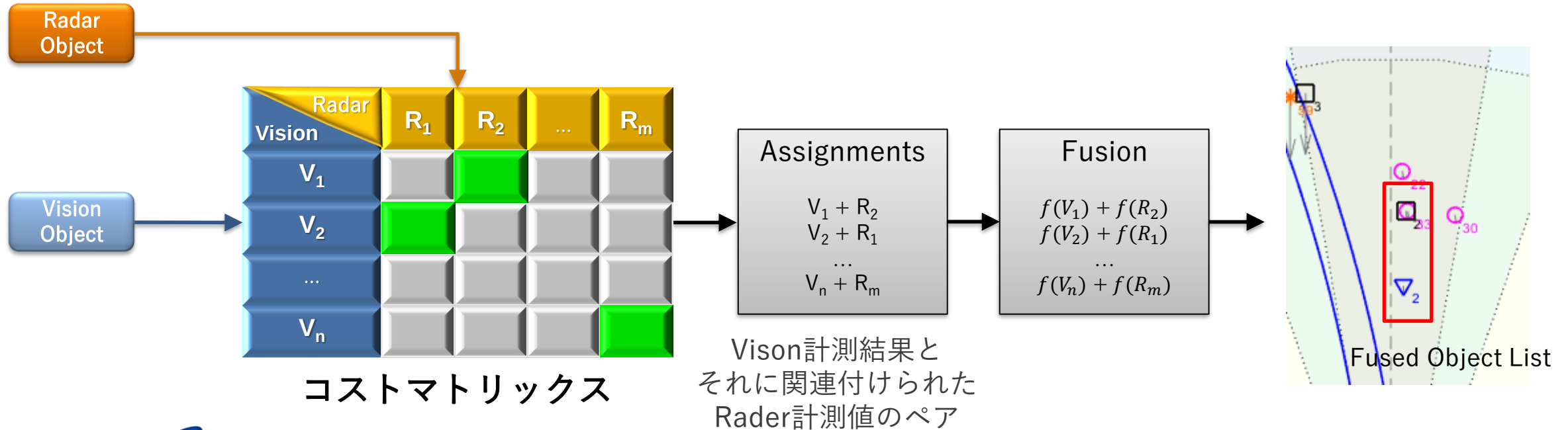


Birds-Eye View object notations

- * 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)

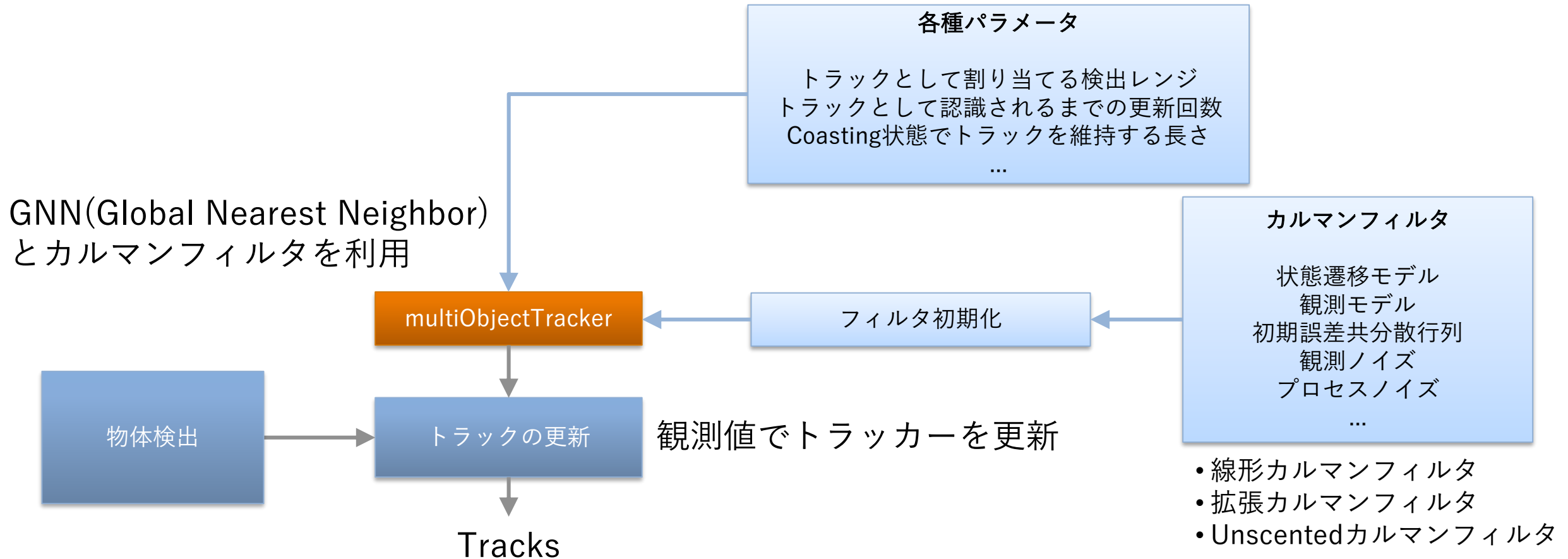


計測値の最適割り当てについて

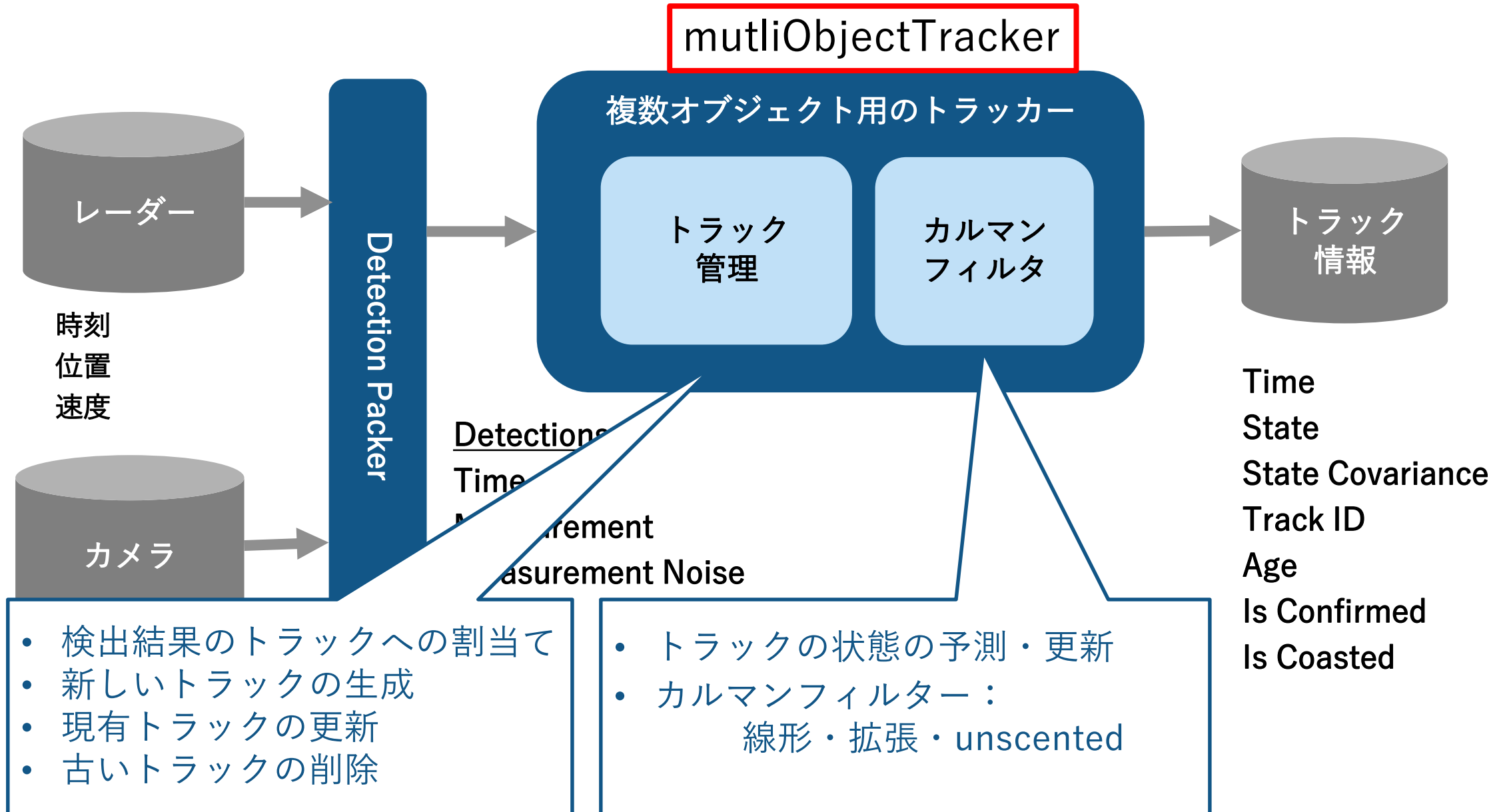


```
[assignments, unassignedVisions, unassignedRadars] = ...
    assignDetectionsToTracks(costMatrix, param.costOfNonAssignment);
```

複数オブジェクトのトラッキング

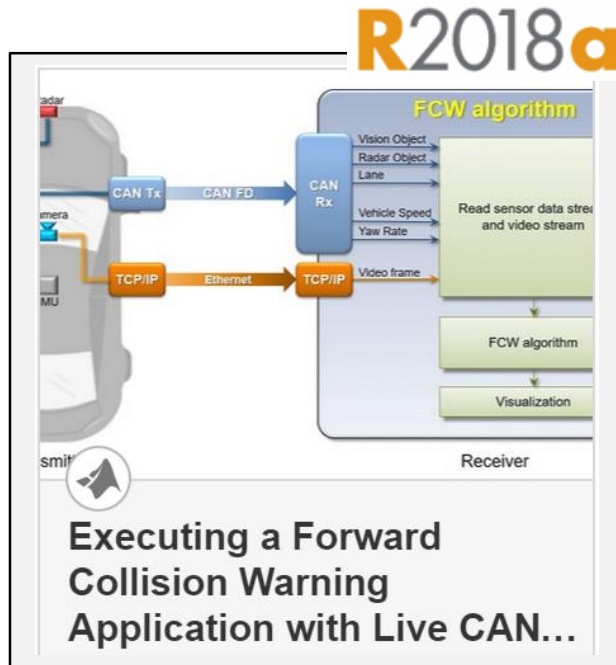


multiObjectTracker



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™



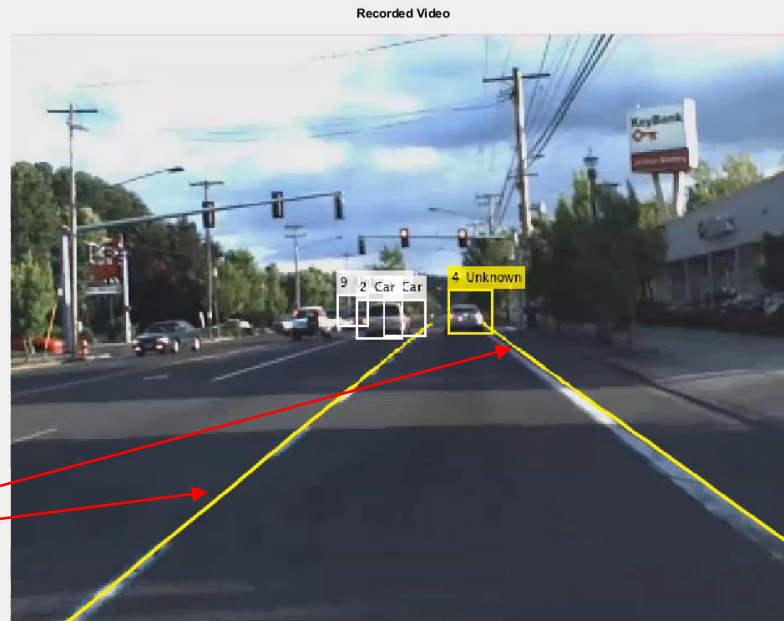
- CAN FD データのストリーミングとアルゴリズムのプロトタイピング



センサーフュージョンによるFCW(前方車接近警報)

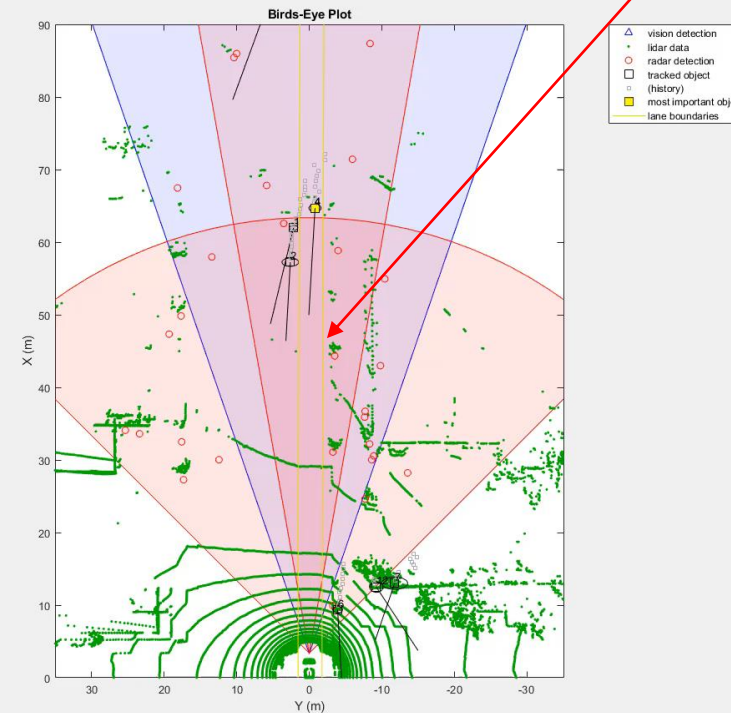
Automated Driving System Toolbox™

Figure 1: Forward Collision Warning With Tracking Example



データ可視化

- `parabolicLaneBoundary`
- `insertLaneBoundary`
- `vehicleToImage`
- `insertObjectAnnotation`

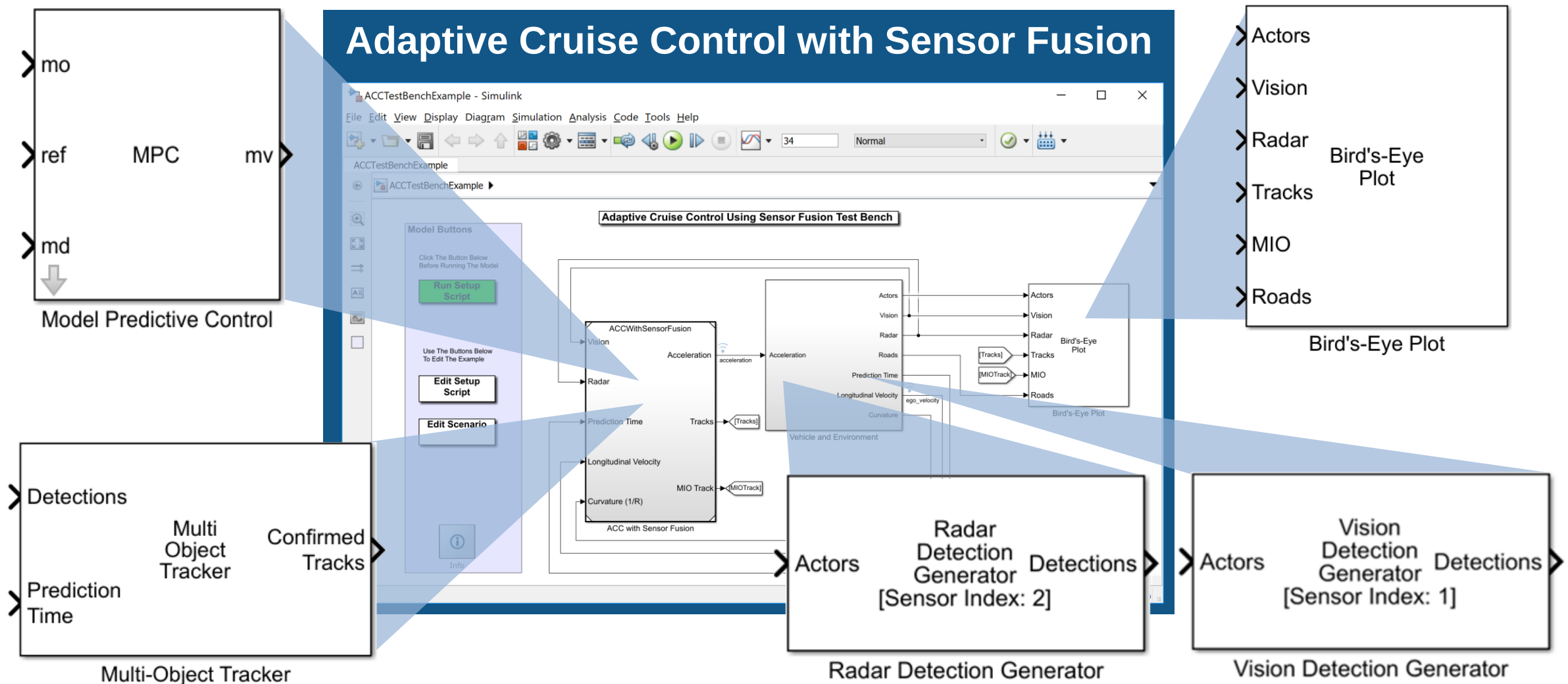


フュージョン&トラッキング

- `objectDetection`
- `trackingEKF`
- `trackingUKF`
- `trackingKF`
- `updateTracks`

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

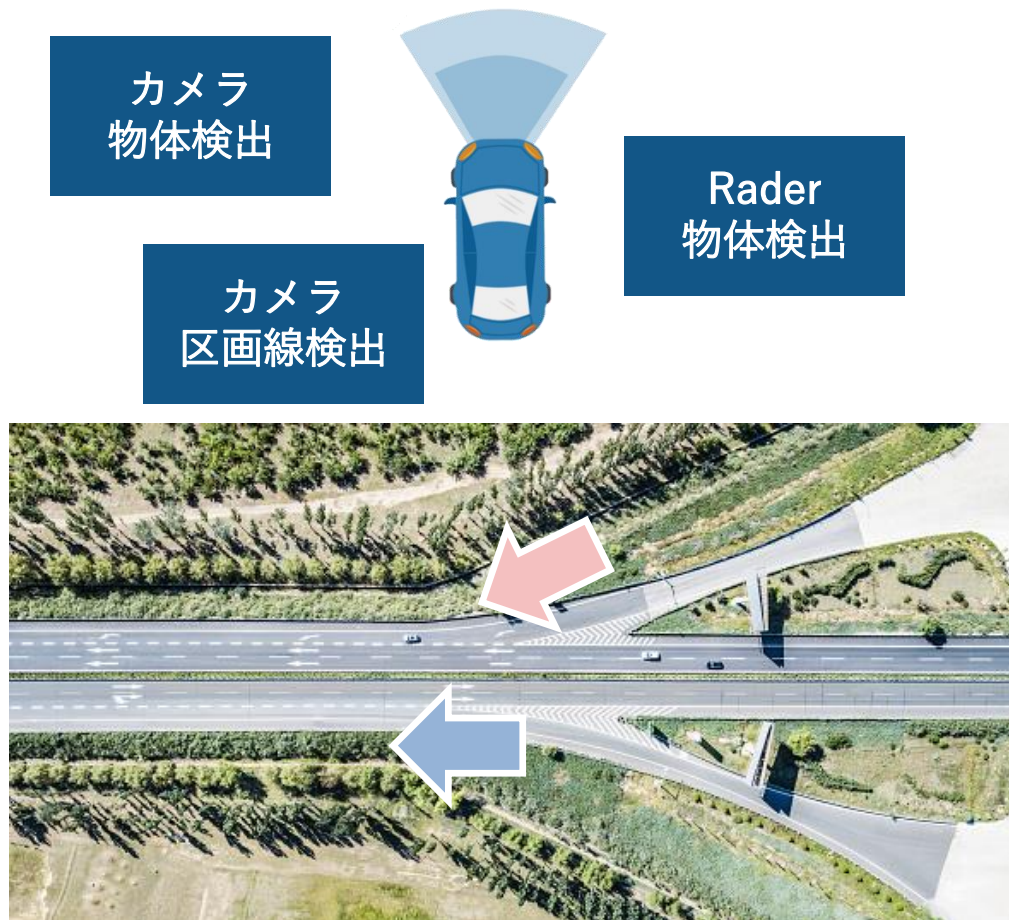
Simulinkブロックも利用可能



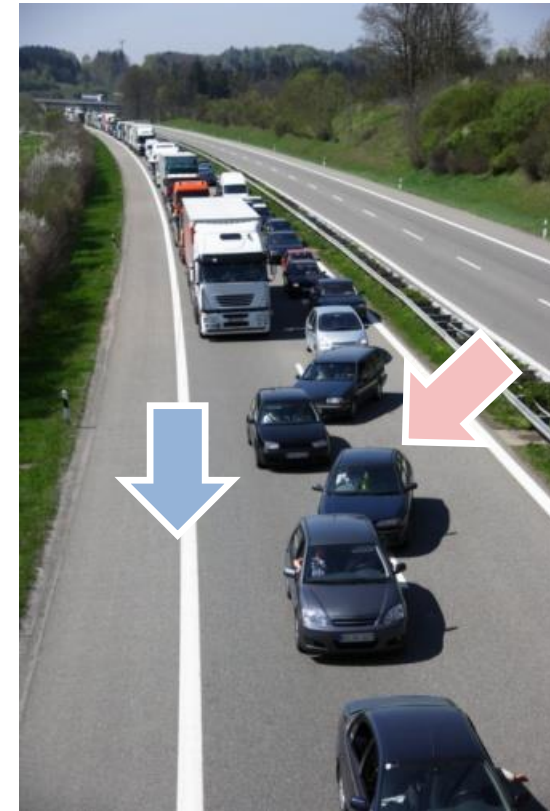
Agenda

- 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



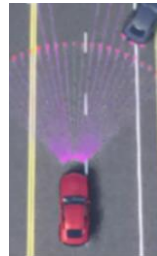
- ・ コーナーケースでのフュージョンアルゴリズム検証
- ・ 異なる特性のセンサの組み合わせ、配置位置の検証



- ・ 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

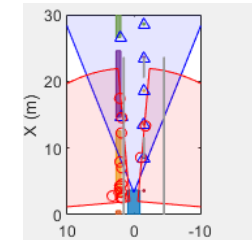
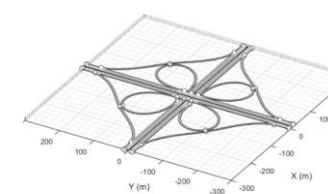


より“リアル”な環境、
物理センサーモデル

- 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

シナリオ作成

Driving Scenario Designer

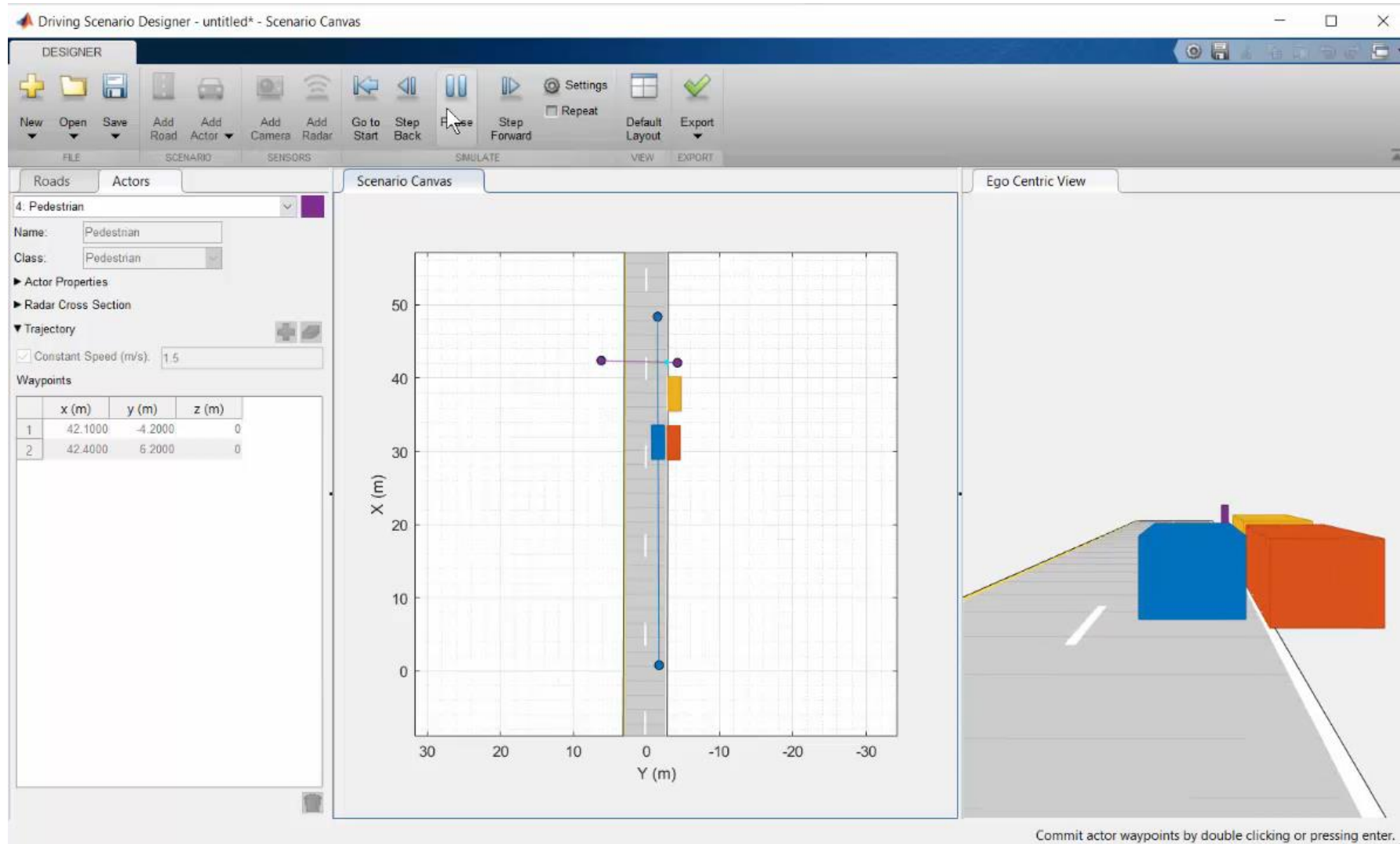


抽象度の高い、簡易的表現
確率的センサーモデル

- 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

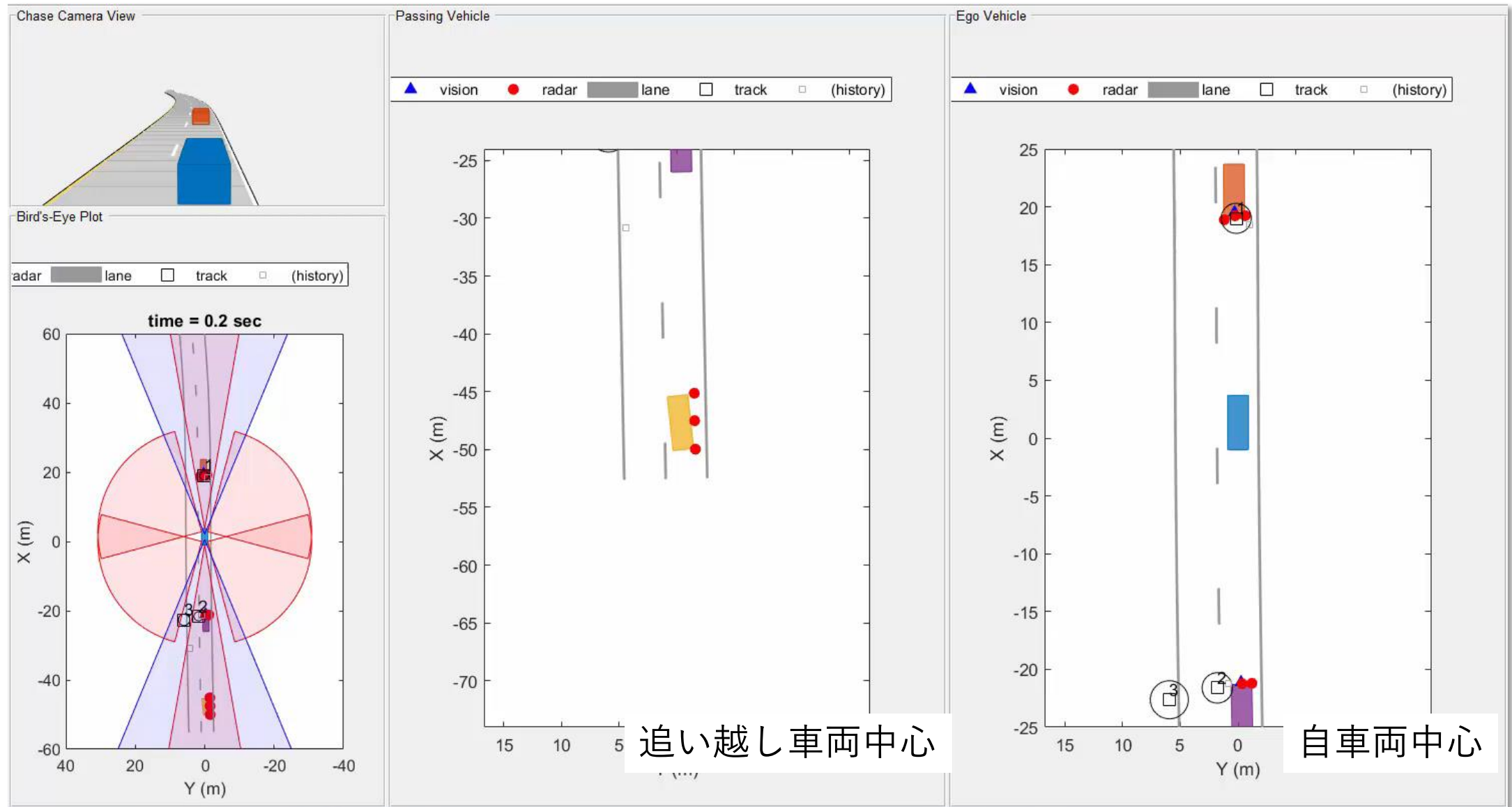
Driving Scenario Designer : シナリオ作成専用App



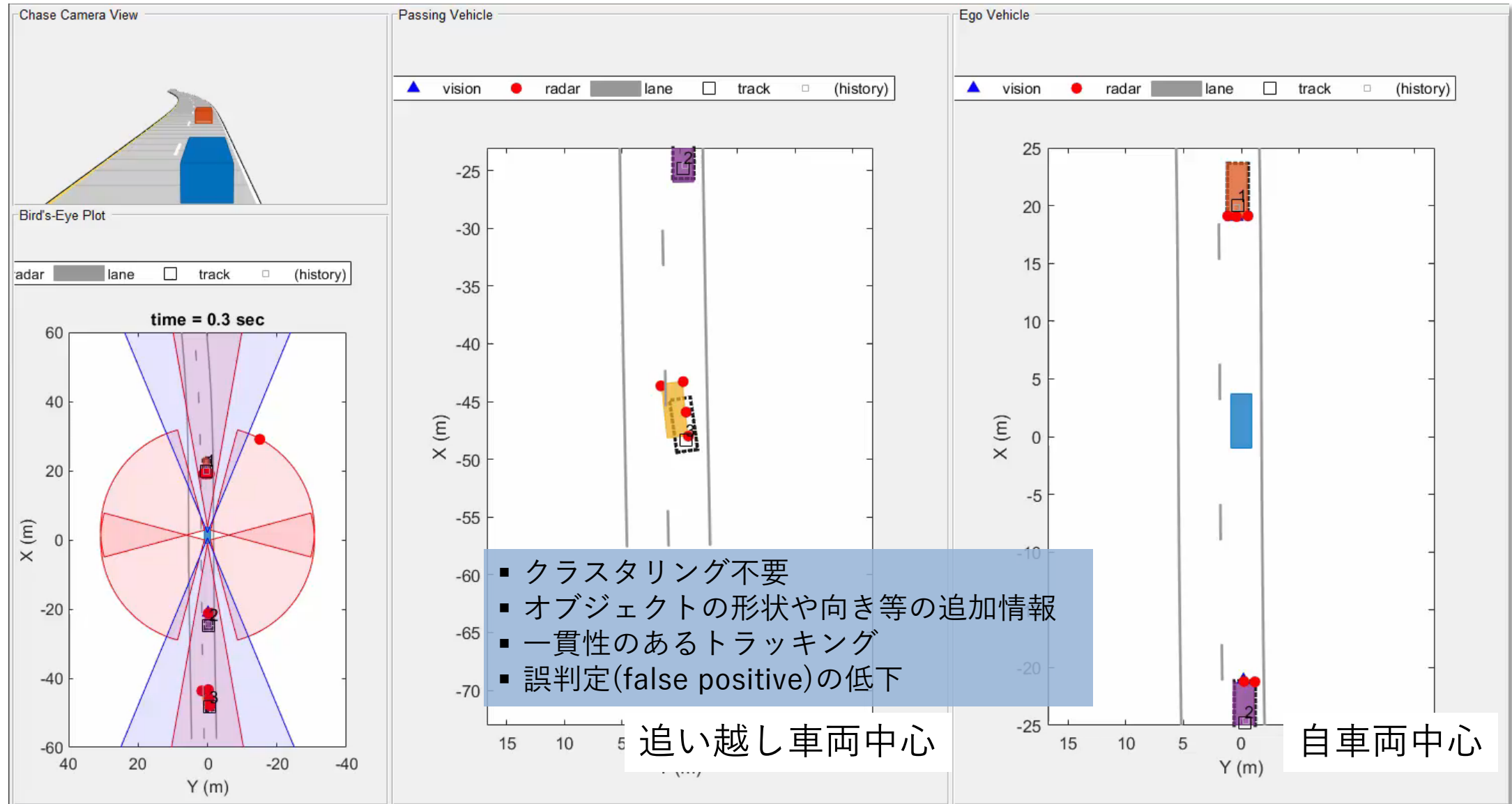
-マウス操作で様々な形状の道やセンサーを定義できます

Point Object Tracker

Automated Driving System Toolbox™



Extended Object Tracker



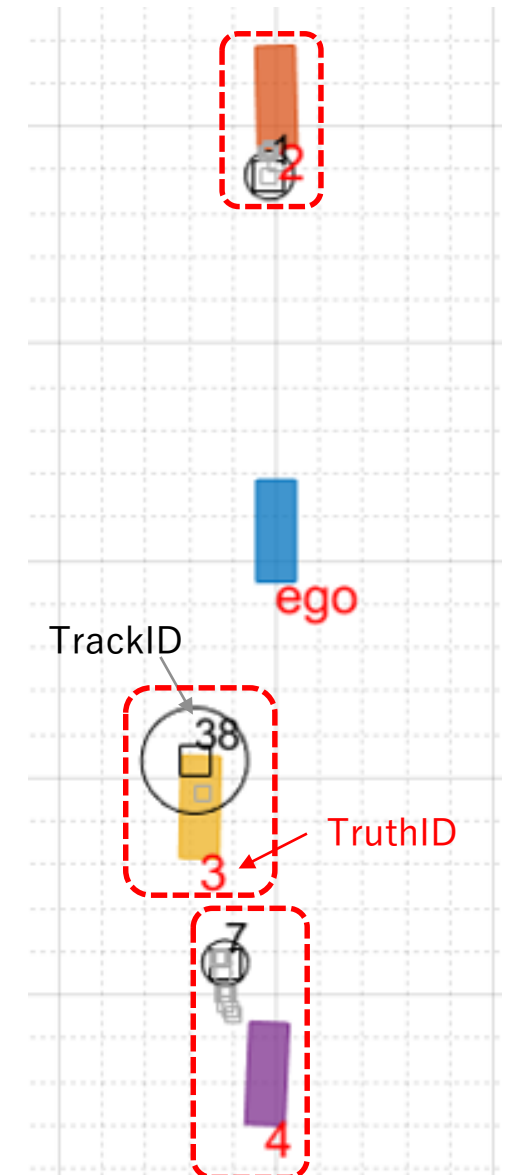
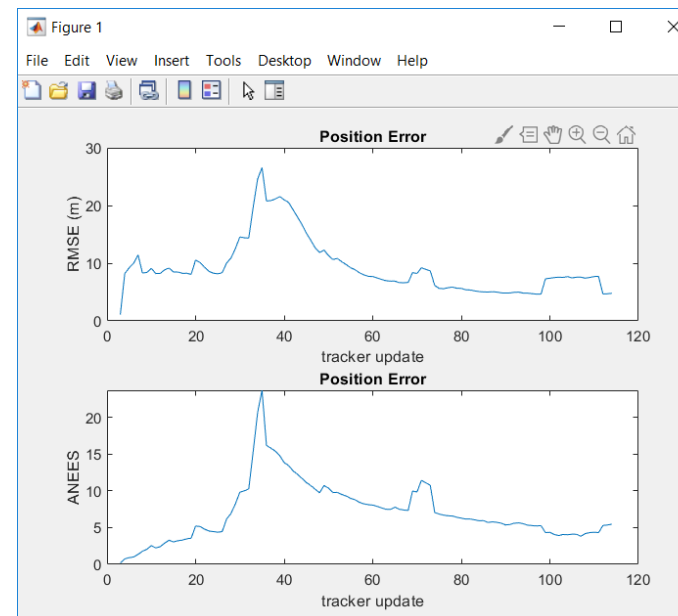
トラッキングパフォーマンスの解析

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

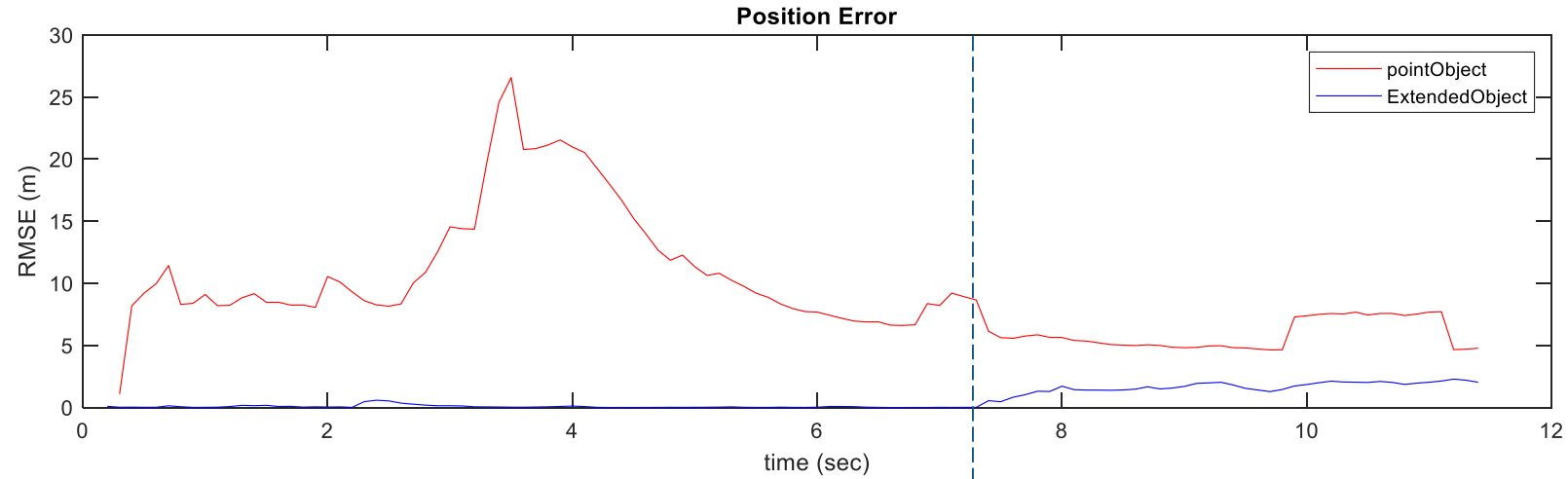
trackErrorMetrics

- 各トラックに対するエラーメトリクス取得

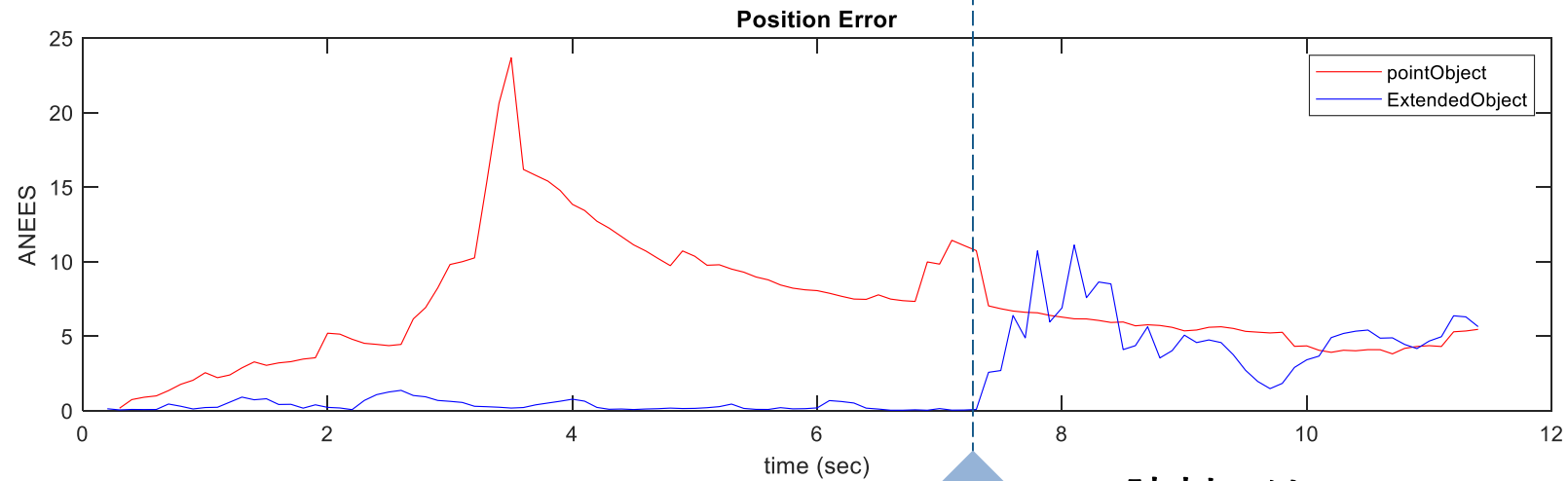


トラックエラー メトリクス

Root Mean
Squared Error



Average
Normalized-
Estimation Error
squared



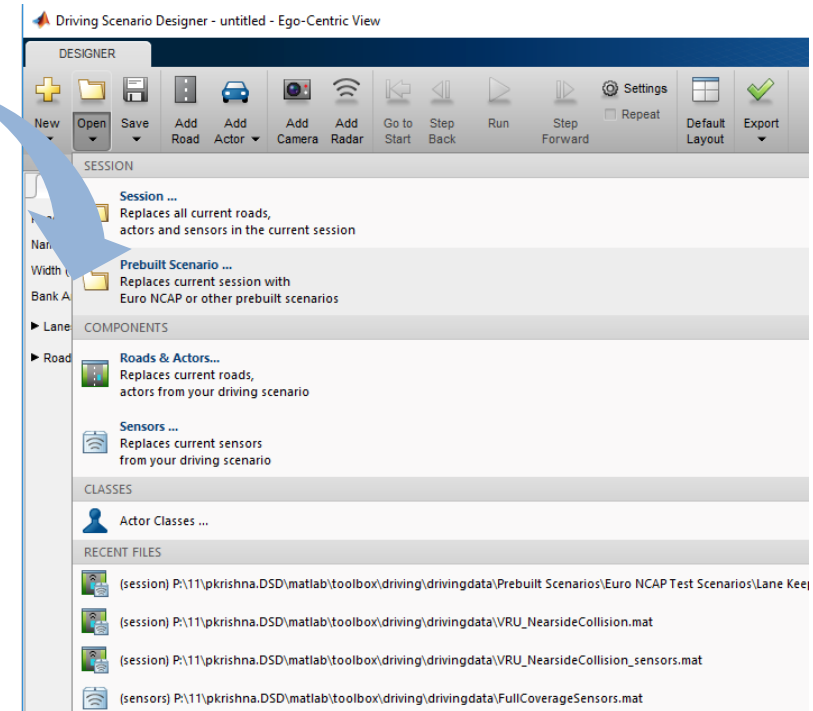
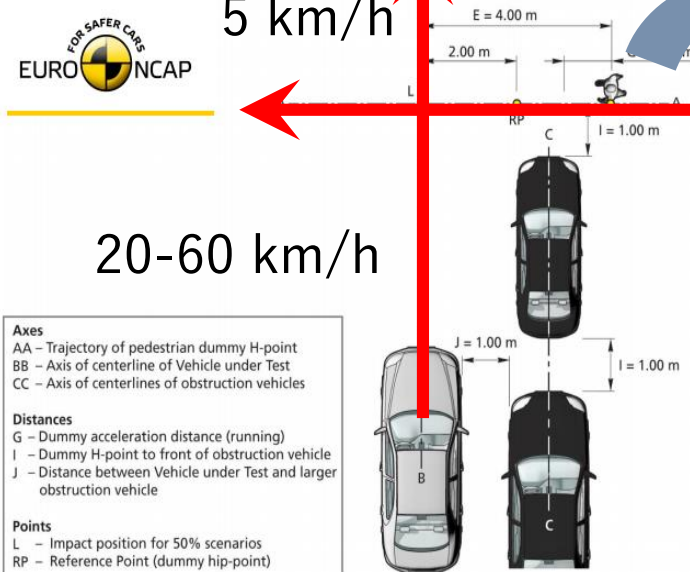
この時刻では、
追い越し車両が自車よりも十分離れている

既存のシナリオの活用

R2018b

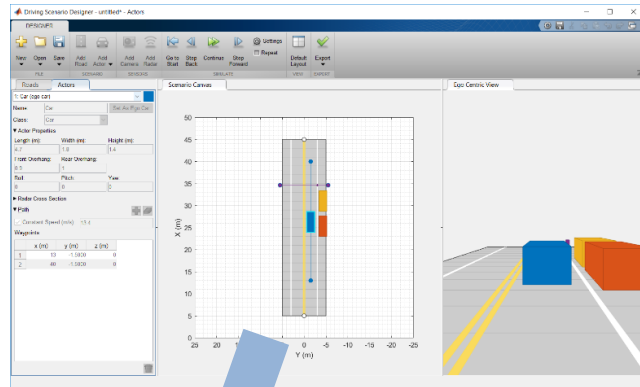
- Euro NCAPのテストシナリオがDriving Scenario Designerですぐに利用可能

AEB Scenario: Vulnerable Road User



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

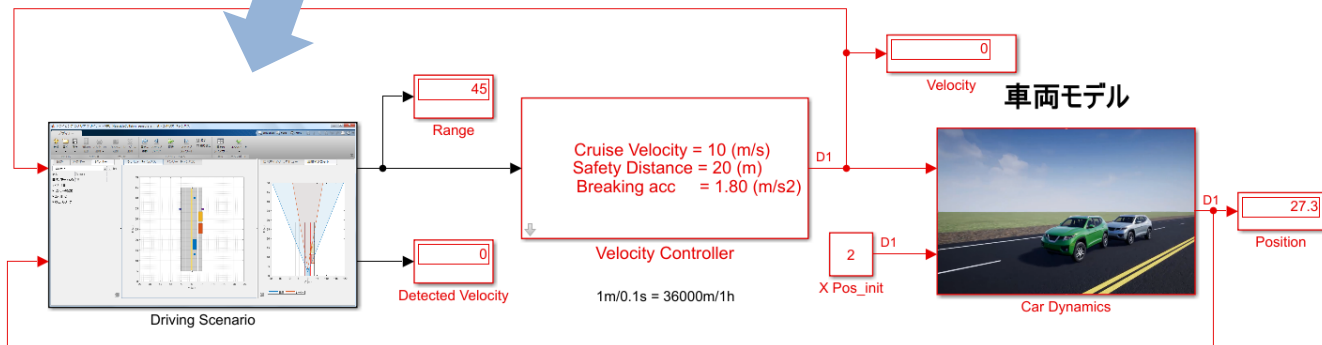
Driving Scenario Designerで作成したシナリオの活用



Driving Scenario Designer

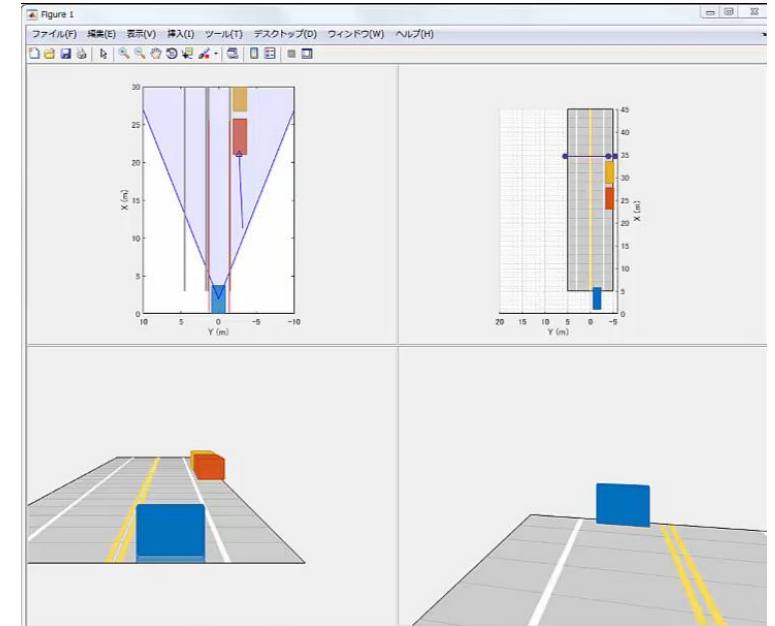
関数としてエクスポート

2回連続で、20m以内に障害物を検出したらブレーキ



モデルプロパティ: InitFcn で、ML 関数の初期化

Simulink モデル



シミュレーション実行&
鳥瞰図プロット

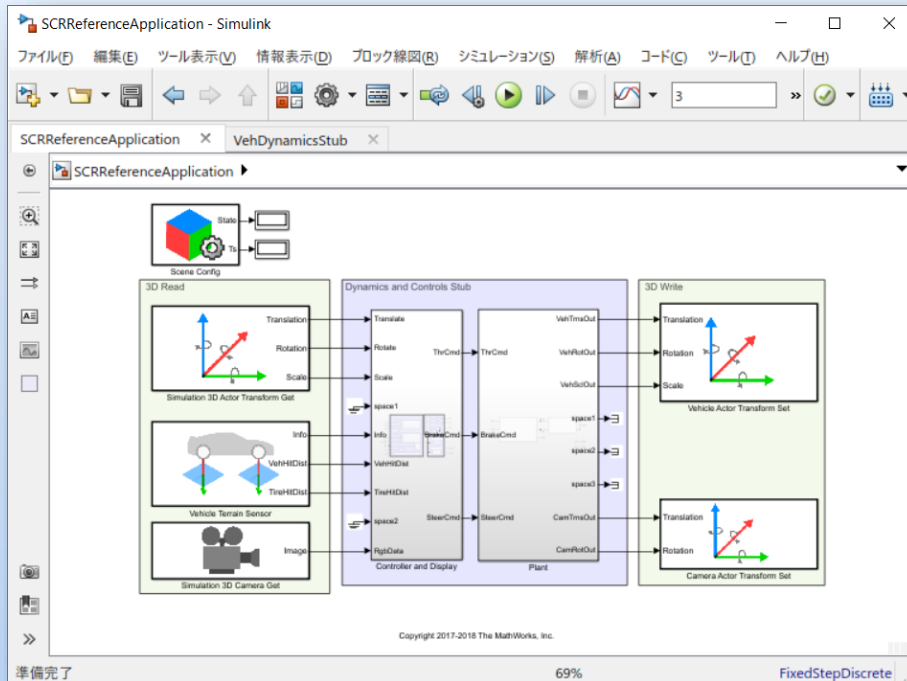
-作成したシナリオは関数としてエクスポート可能、車両ダイナミクスモデルや既存の制御アルゴリズムとSimulink上で統合してシミュレーション可能です。

ゲーミングエンジンとの協調シミュレーション

Vehicle Dynamics Blockset™

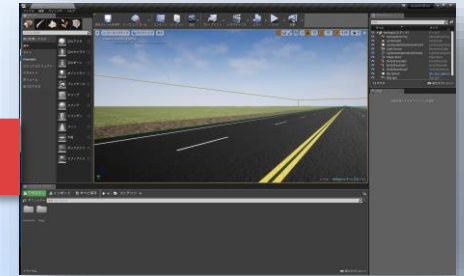
- ゲーミングエンジンと閉ループ構築可能

Vehicle Dynamics Blocksetに同梱



Simulinkモデル

コンパイル



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

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

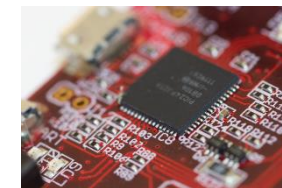
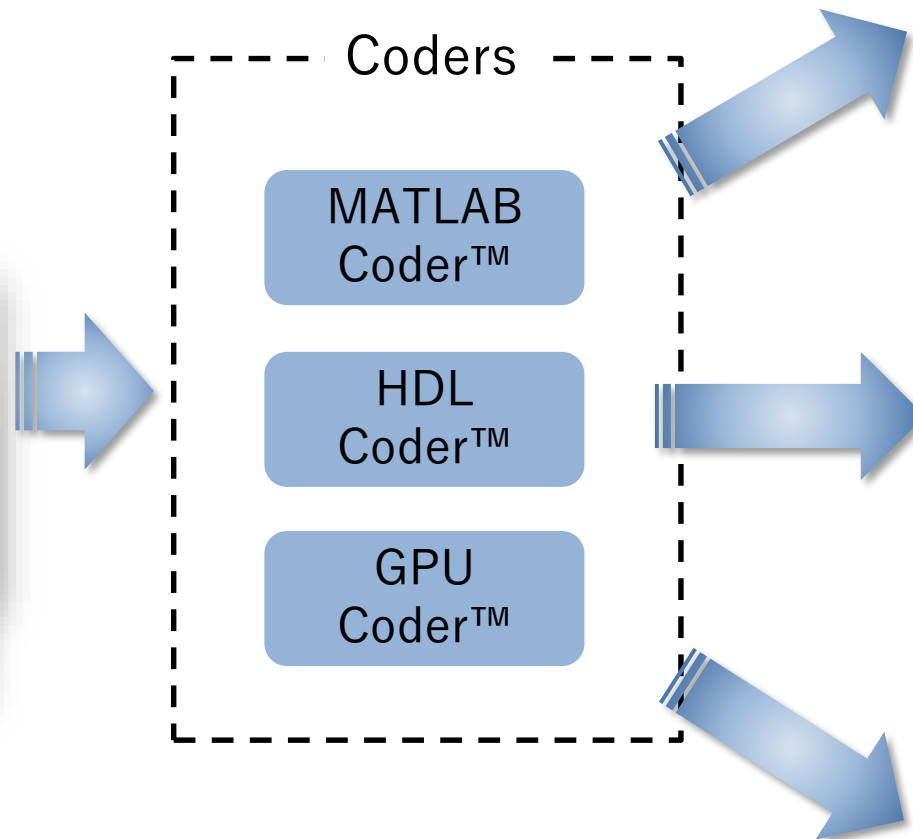
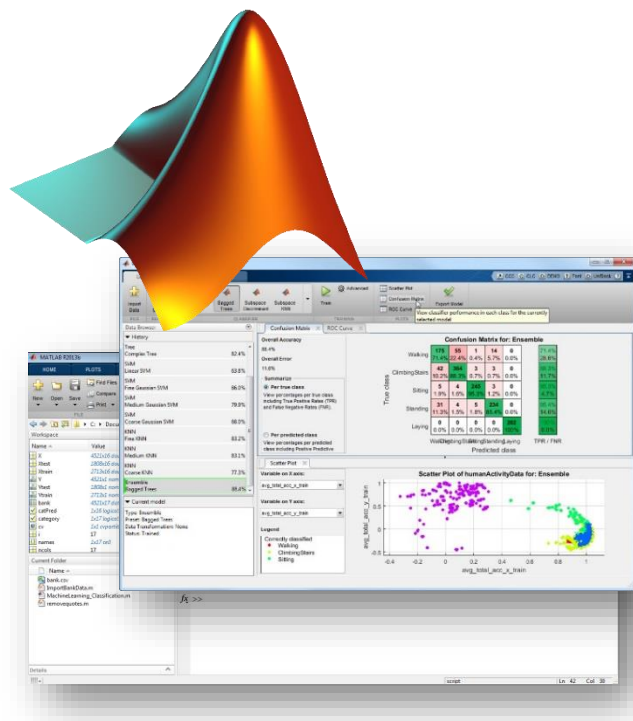


ゲーミングエンジンの実行ファイル
(コンパイル済みのモデル)

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



マイコン/DSP

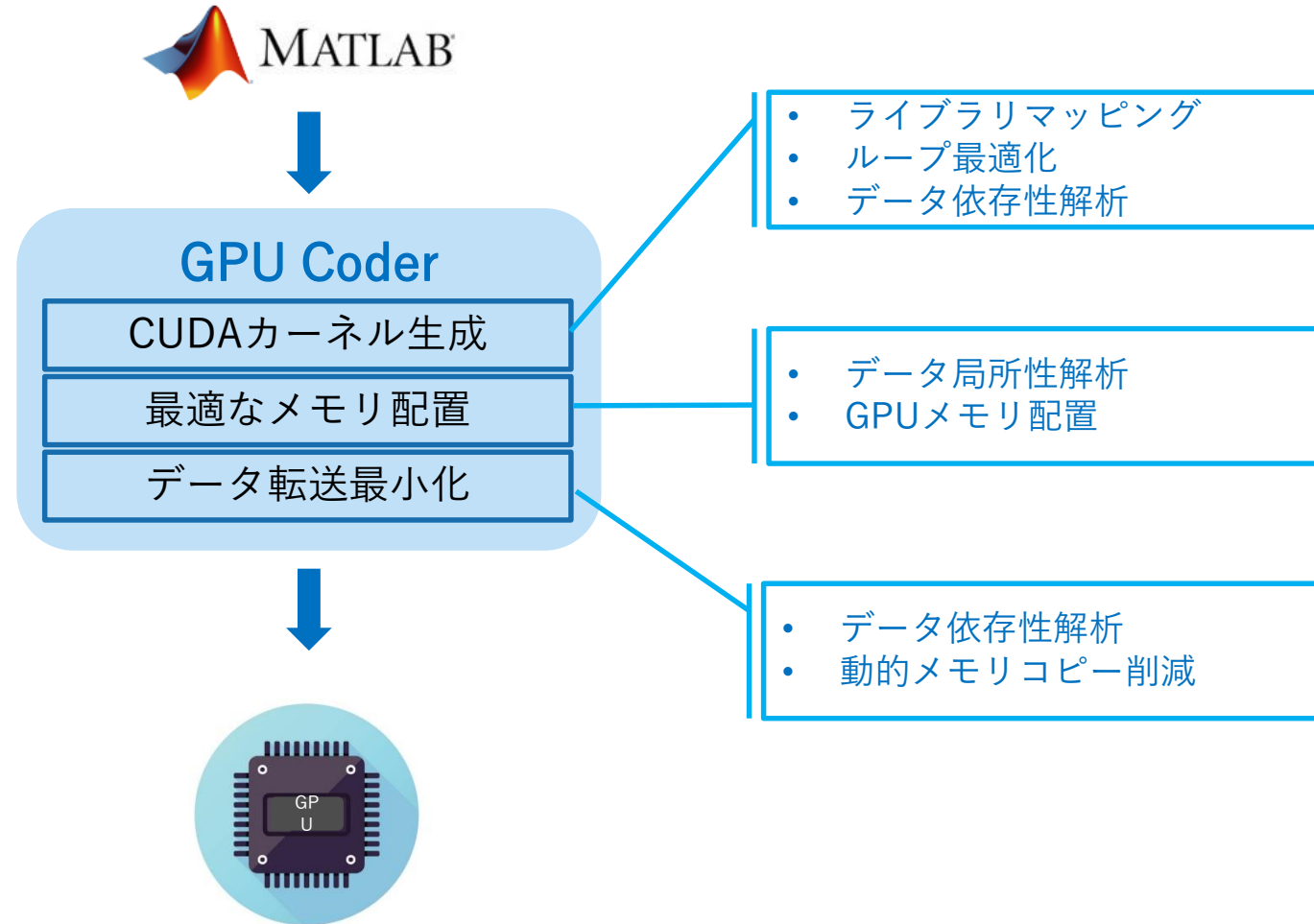


FPGA/ASIC



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

The image displays three overlapping screenshots of the GPU Coder application interface, illustrating the workflow for generating code from a MATLAB function.

Left Screenshot (Selection Step): The "GPU Coder" window shows the "選択" (Select) step. The "エントリーポイント関数:" (Entry point function) field is set to "vecSum". A callout box points to this field with the text: "コード生成対象となる関数を指定" (Specify the function to be generated).

Middle Screenshot (Input Type Definition Step): The "GPU Coder - vecSum.prj" window shows the "入力の型を定義" (Define input types) step. The "MATLAB を GPU に変換するには、エントリーポイント関数ごとに各入力の型を定義し" (To convert MATLAB to GPU, define the type of each input for each entry point function) message is visible. The "入力データの型を指定" (Specify input data type) callout box points to the "single" type selected for the input. A dropdown menu shows options: "100 正確に 100", ":100 最大 100", and ":Inf 制限なし".

Right Screenshot (Execution Confirmation Step): The "GPU Coder - vecSum.prj" window shows the "実行時の問題の確認" (Check for execution problems) step. The "コード生成時の問題を確認" (Check for code generation problems) callout box points to the "問題の確認" (Check for problems) button. The "コードを入力するか、vecSum を実行するスクリプトを選択してください:" (Enter code or select a script to run vecSum) message is visible. The input code is ">> vecSum(rand(1,100,'single'))". The "問題の確認" button is highlighted. Below the code input, there are checkboxes for "MATLAB 行の実行回数の収集" (Collect MATLAB line execution counts) and "次の問題の確認:" (Check for the next problem: CPU GPU). A green message bar states "問題は検出されませんでした。" (No problems were detected). At the bottom, there are three green checkmarks indicating successful steps: "試行コードを生成中" (Generating trial code), "MEX をビルド中" (Building MEX), and "MEX でテスト ファイルを実行中" (Running test file with MEX). The output value "50.4444656" is displayed.

NVIDIA Hardware Support Package (HSP)

GPU Coder™
R2018b

Project Settings

Speed Memory Code Appearance Debugging Custom Code Hardware GPU Code

Host Hardware (CPU)

Hardware Board: NVIDIA Drive

Device: MATLAB Host Computer
NVIDIA Drive
NVIDIA Jetson

[Customize hardware implementation](#)

Build Process

Toolchain: NVCC for NVIDIA Embedded Processors [Validate...](#)

Build Configuration: Faster Runs [Show settings](#)

Minimize run time

NVIDIA Drive Settings

Board Parameters

Device Address: gpusoder-tx1

Username: nvidia

Password:

Build directory: /home/buildDri

ターゲットボードの
選択



MATLABから直接ボードに
アクセス可能

ターゲットボードの
パラメータ入力



Jetson

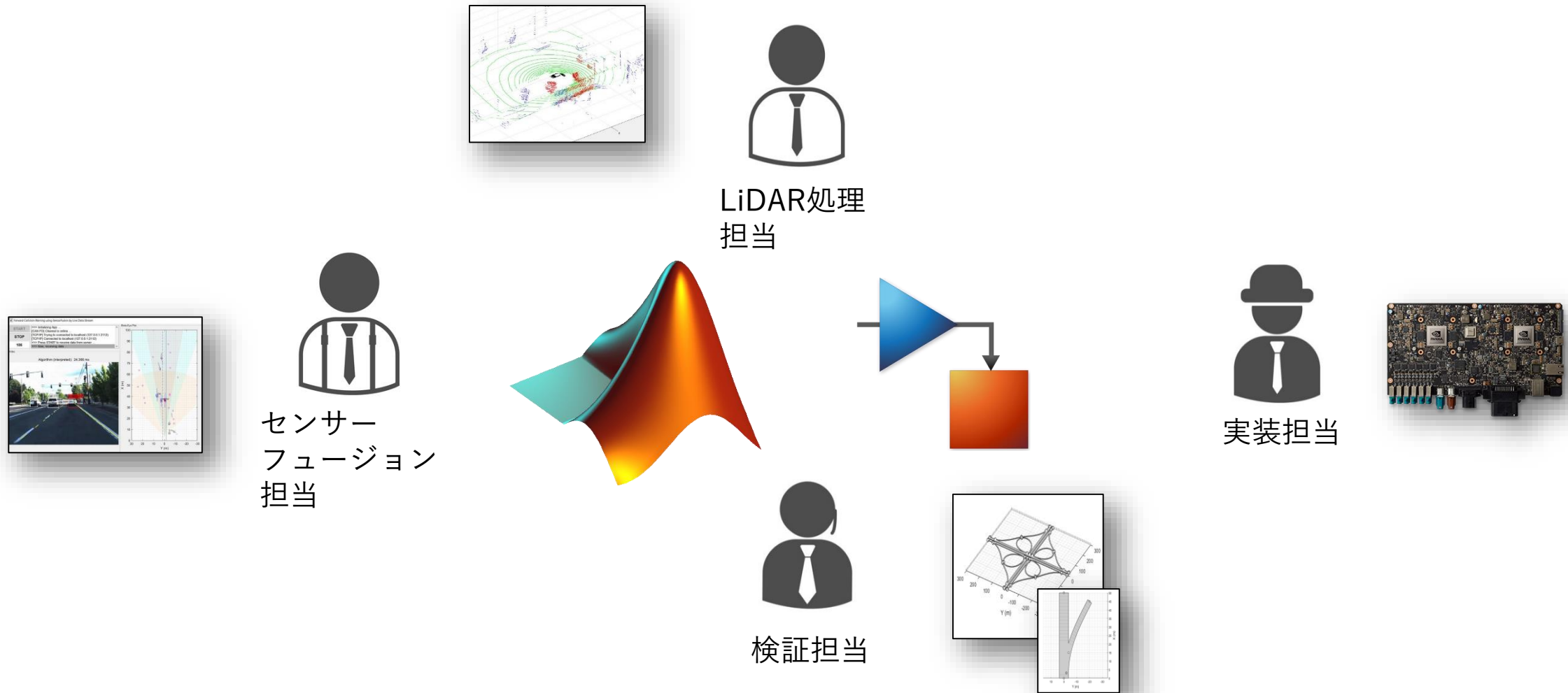


Drive
platform

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

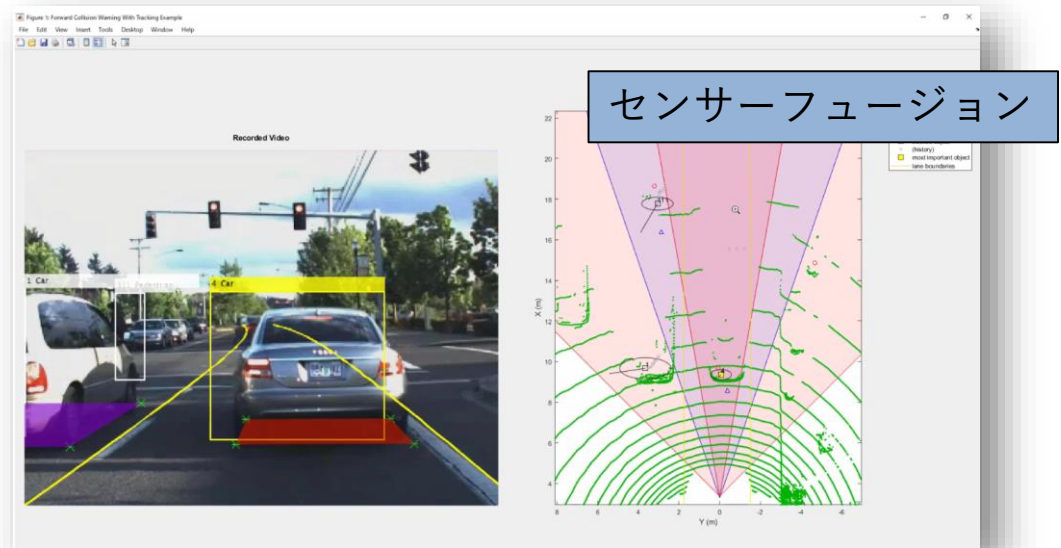
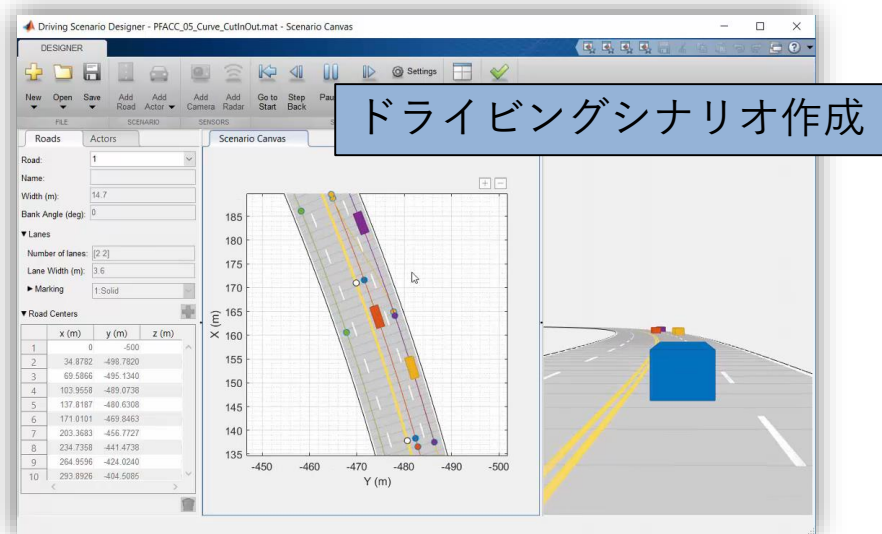
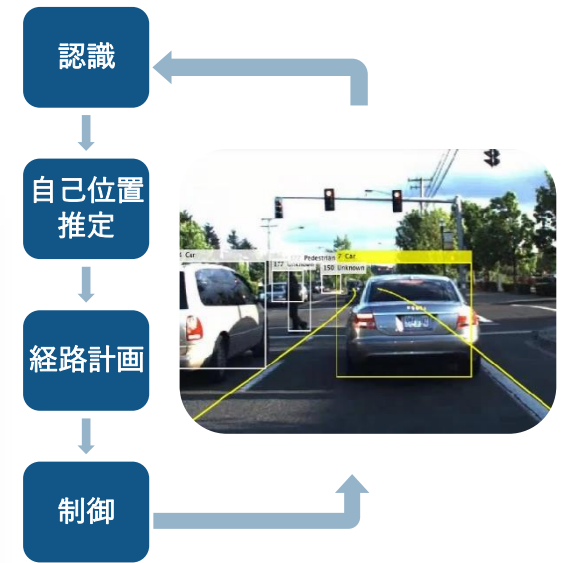
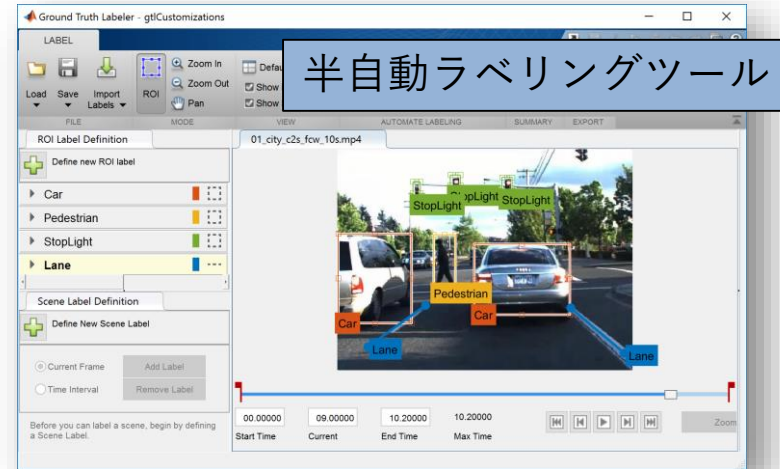
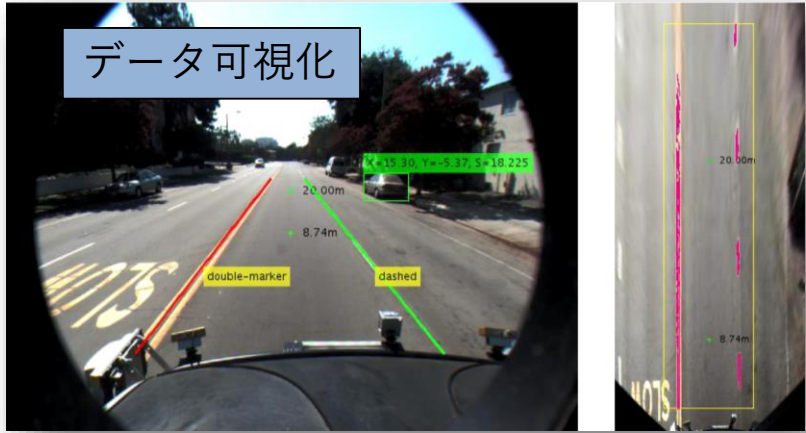
Key Takeaways : ADAS・自動運転開発・検証の統合プラットフォーム



- 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





© 2018 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.