

# MATLABではじめる 自律移動システムのためのパスプランニング

MathWorks Japan

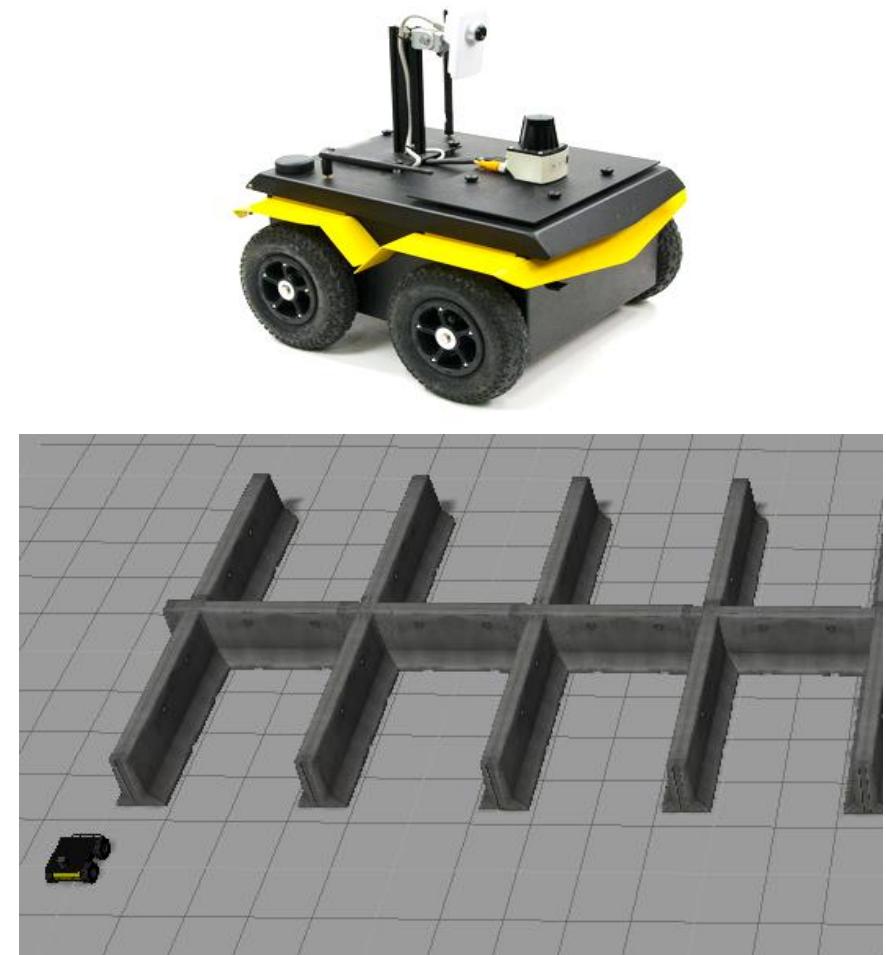
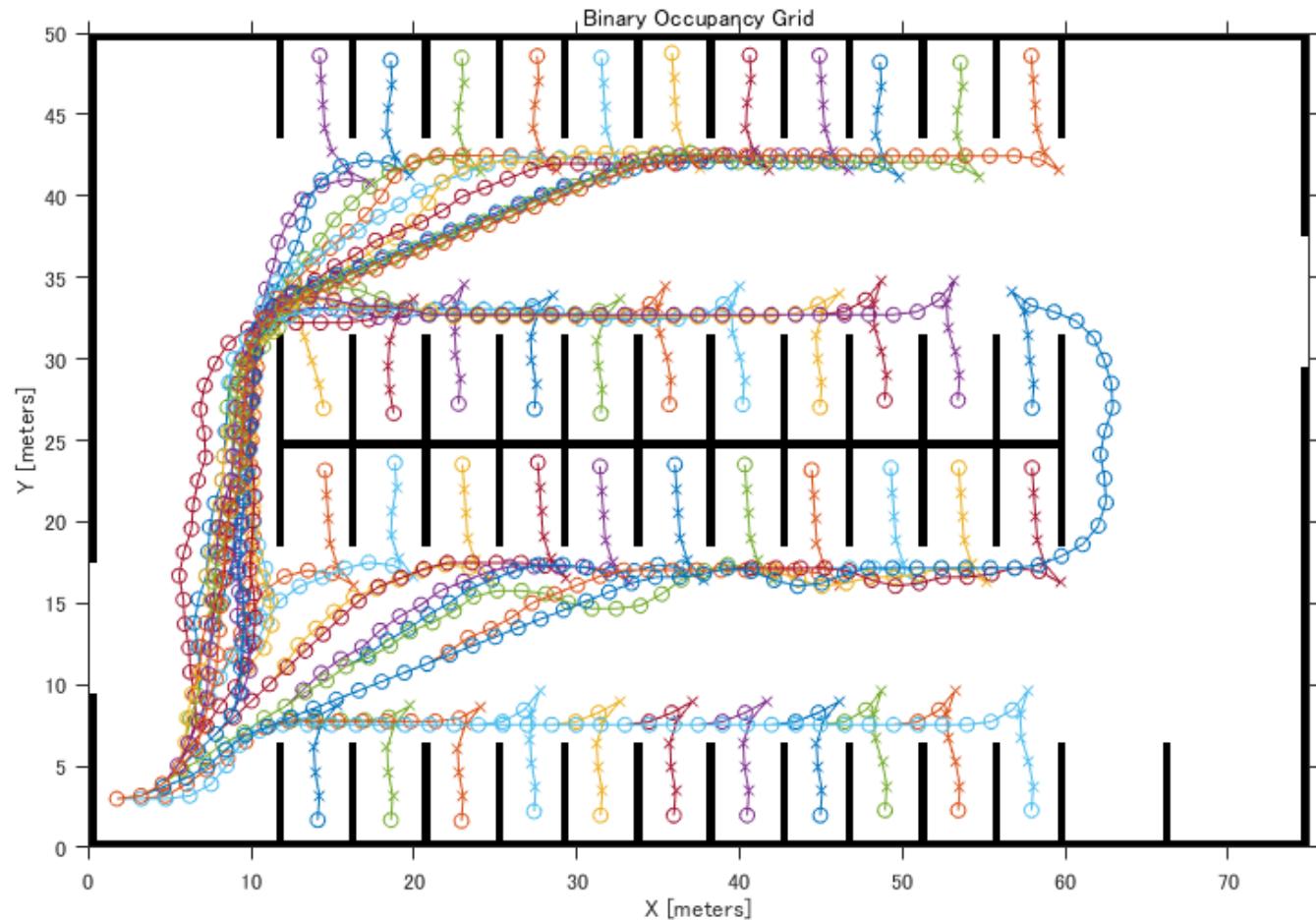
アプリケーションエンジニアリング部

アプリケーションエンジニア

木川田 亘

# パスプランニングによる自動駐車デモ

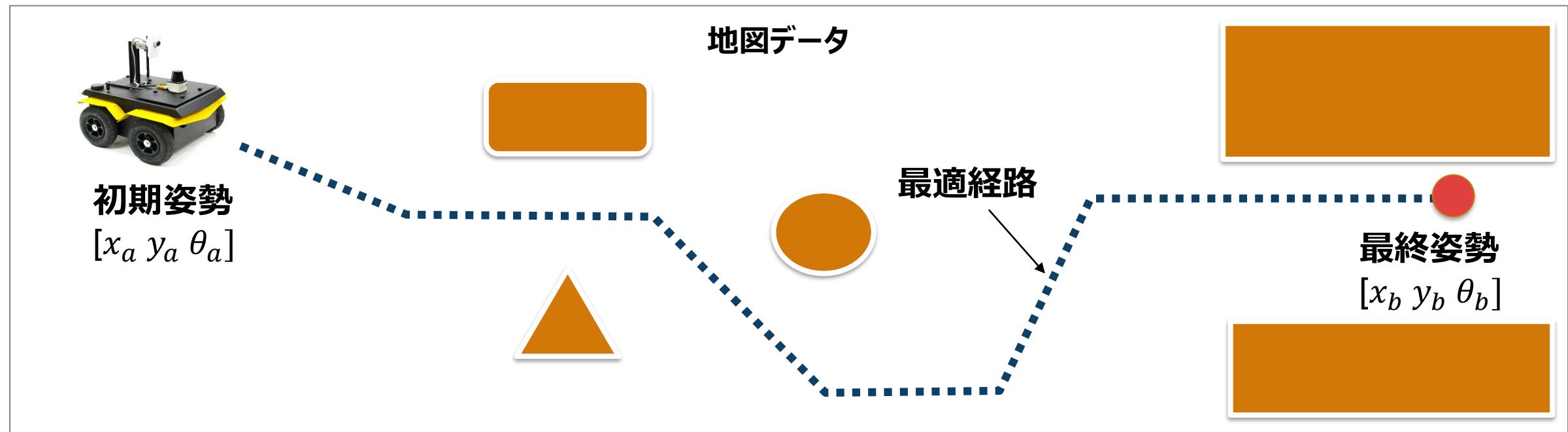
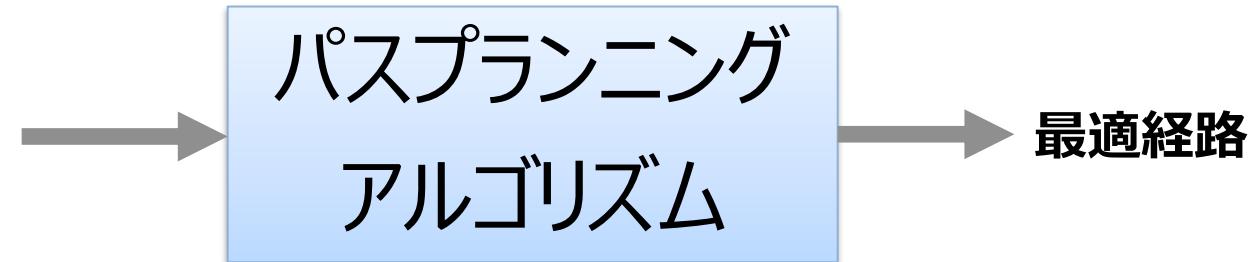
Robotics System Toolbox™



パスプランニングアルゴリズム開発におけるMATLAB/Simulink活用方法を紹介

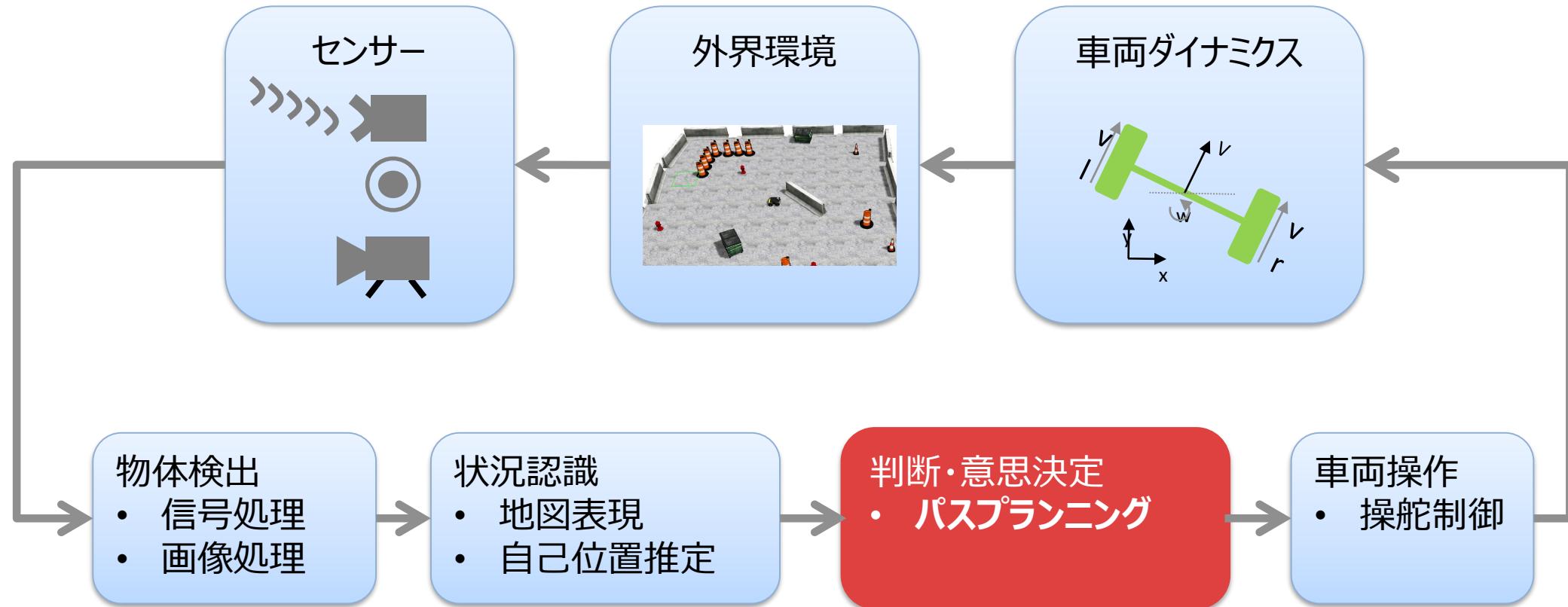
# パスプランニングとは？

地図データ(障害物表現)  
初期姿勢(自己位置推定)  
最終姿勢(目標姿勢)



複合制約の下、ロボットの最適経路を探索する  
アプリケーション：倉庫内搬送、車線変更、追い越し、自動駐車

# 自律システムの開発検証



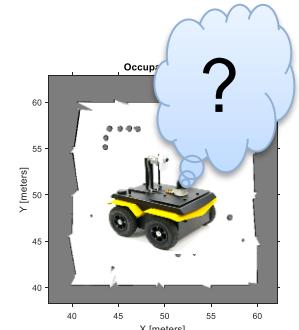
自律システムはClosed-loopで構成される複雑なシステム  
パスプランニング単体での検証だけでなく、  
システムレベルでの妥当性の検証が必要

# パスプランニングアルゴリズムの開発フロー

地図表現/自己位置推定  
(アルゴリズム開発に必要な周辺要素)



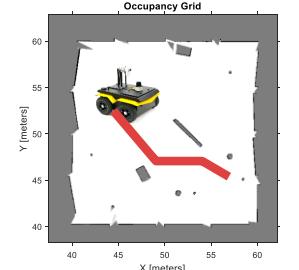
障害物表現  
センサー信号処理  
状態推定



パスプランニング  
(アルゴリズム単体開発・検証)



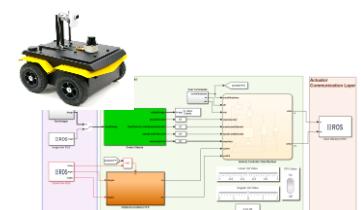
経路計画アルゴリズム  
最適化計算



システムへの統合  
(システムレベルの検証)

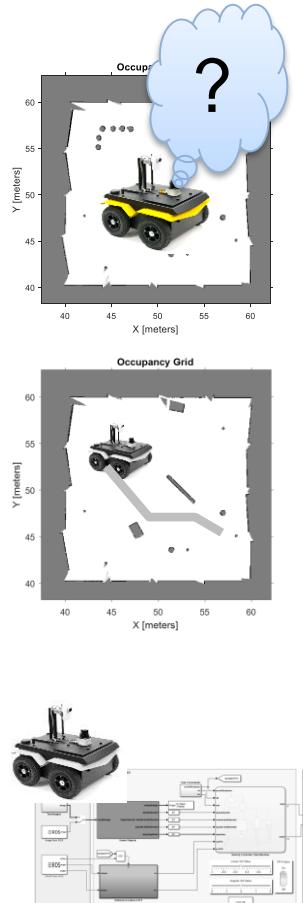
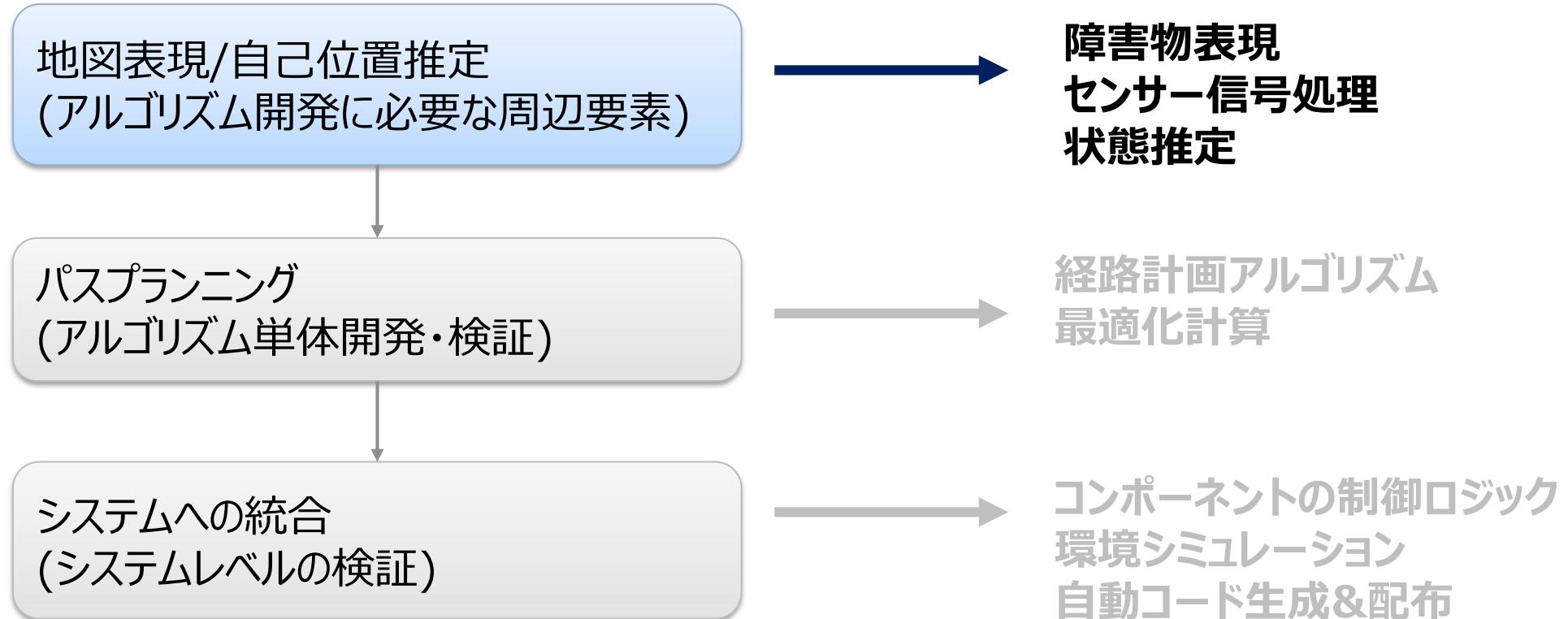


コンポーネントの制御ロジック  
環境シミュレーション  
自動コード生成&配布



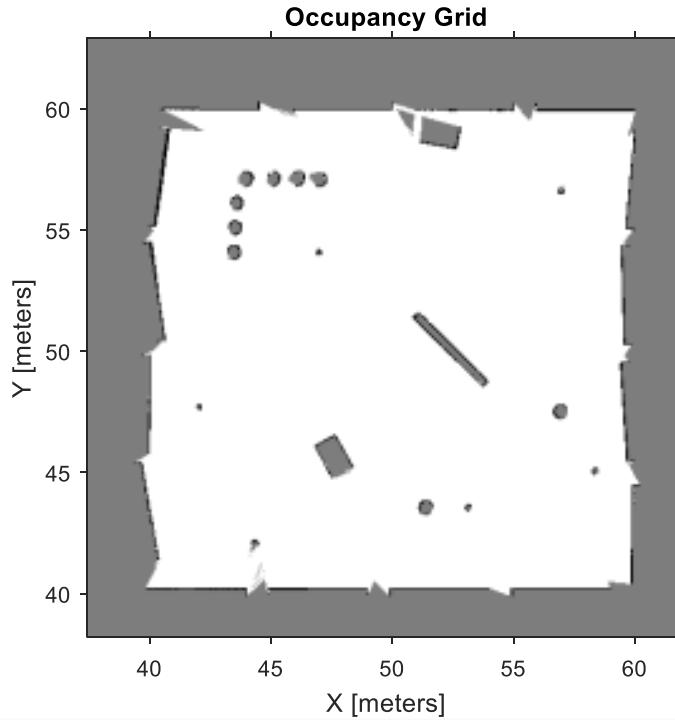
MATLAB/Simulinkを活用することで单一環境で自律システム開発

# パスプランニングアルゴリズムの開発フロー



# さまざまな地図表現

## 2Dマップ (グリッド/ベクター)



障害物を0～1で表現  
プログラムでの取り扱いが容易

Robotics System Toolbox™

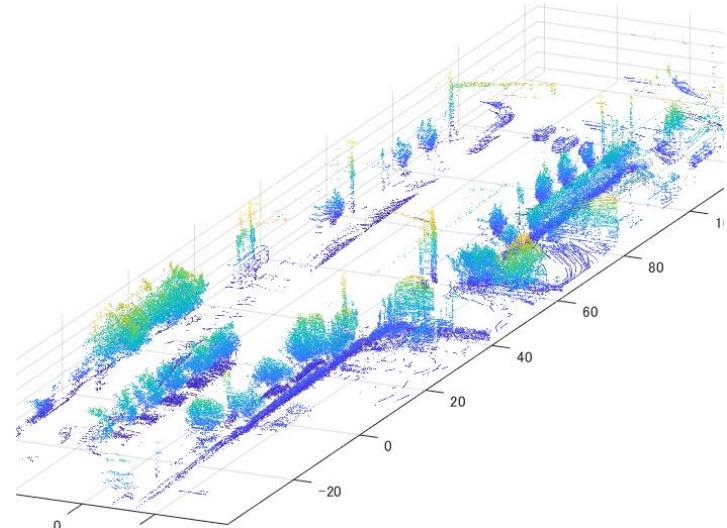
## 2Dマップ (航空写真/ベクター)



一般的な地理データ  
道路や建物なども表現

Mapping Toolbox™

## 3Dマップ (グリッド/点群/ベクター)



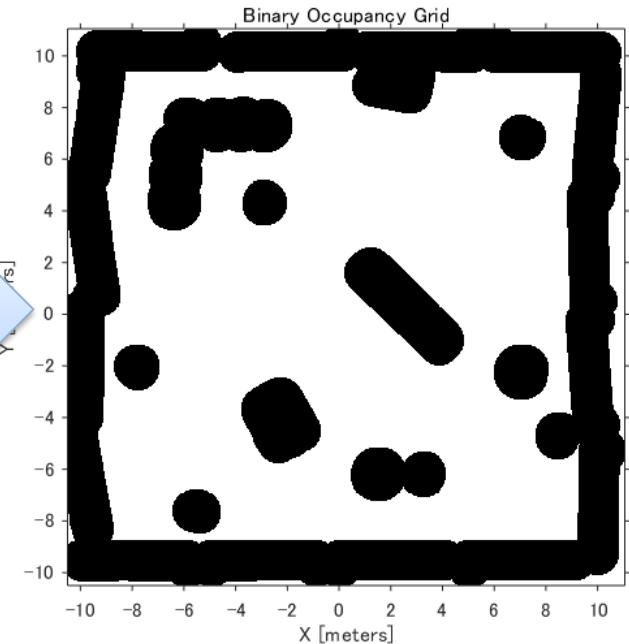
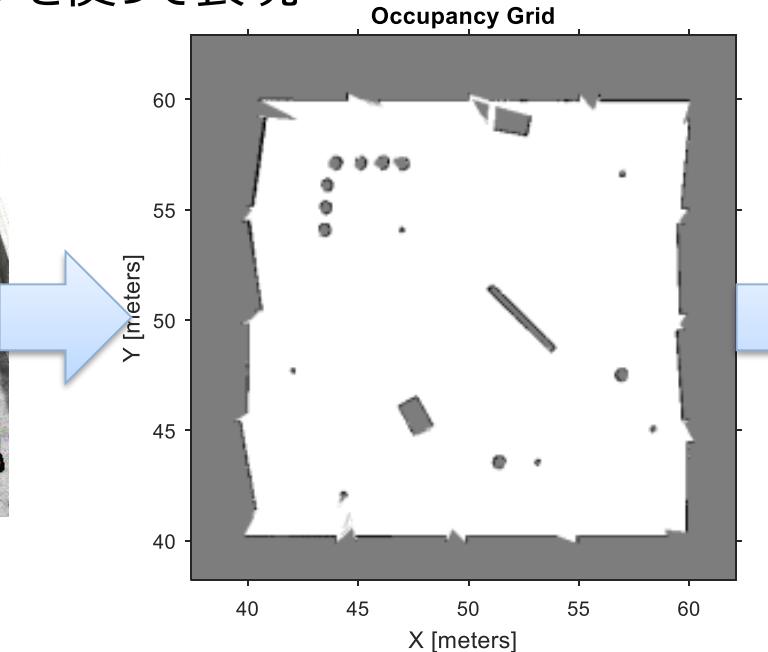
点群やベクターデータ  
より高精細な地図

Computer Vision System Toolbox™

# 2Dグリッドマップ表現

Robotics System Toolbox

移動体の環境を確率的グリッドマップを使って表現



自律移動体の周辺環境地図

グリッドマップ作成

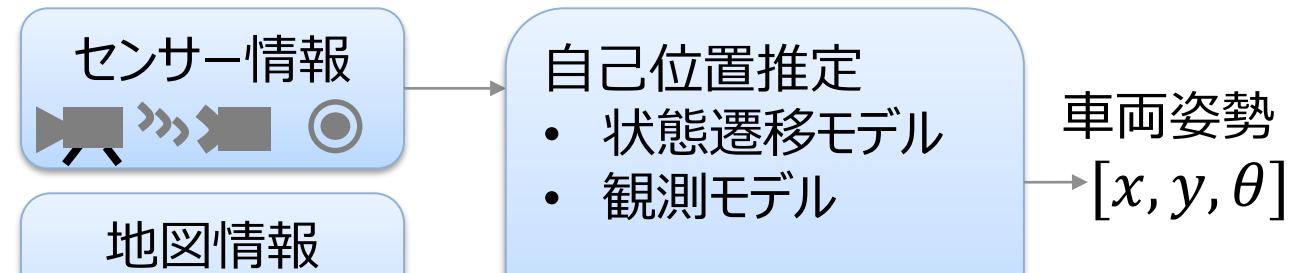
車両マージン分膨張

```
map = robotics.BinaryOccupancyGrid(binaryImage, 20);  
inflate(map, 0.75)
```

便利なグリッドマップオブジェクトを提供

# 自己位置推定とは？

- 自己位置推定(Localization)とは周辺地図上のロボットの姿勢をセンサー情報を基に推定する技術
- LiDARによる姿勢推定
  - スキャンマッチング
    - ICPアルゴリズム\*1
    - NDTアルゴリズム\*2
  - ベイズフィルタ
    - パーティクルフィルタ\*2  
(モンテカルロローカリゼーション)
    - カルマンフィルタ  
(線形/拡張/Unscented) \*3



\*1 Computer Vision System Toolbox, \*2 Robotics System Toolbox, \*3 Automated Driving System Toolbox™

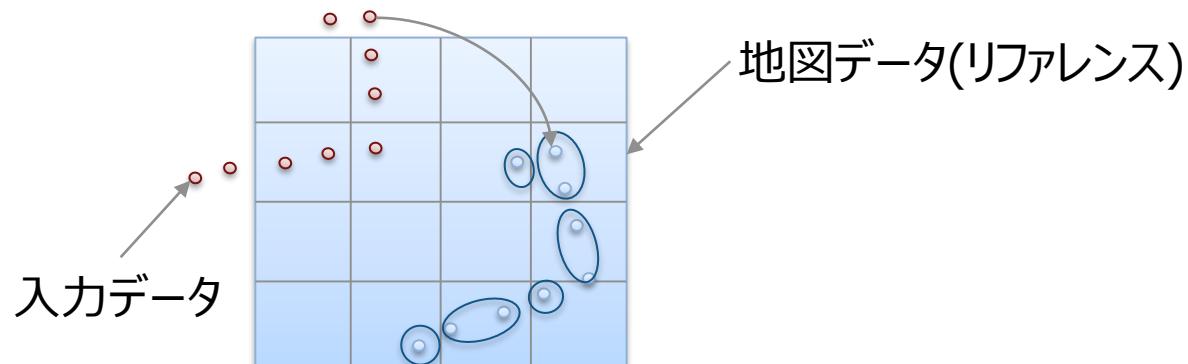
# 自己位置推定：NDTアルゴリズム

Robotics System Toolbox  
Automated Driving System Toolbox™

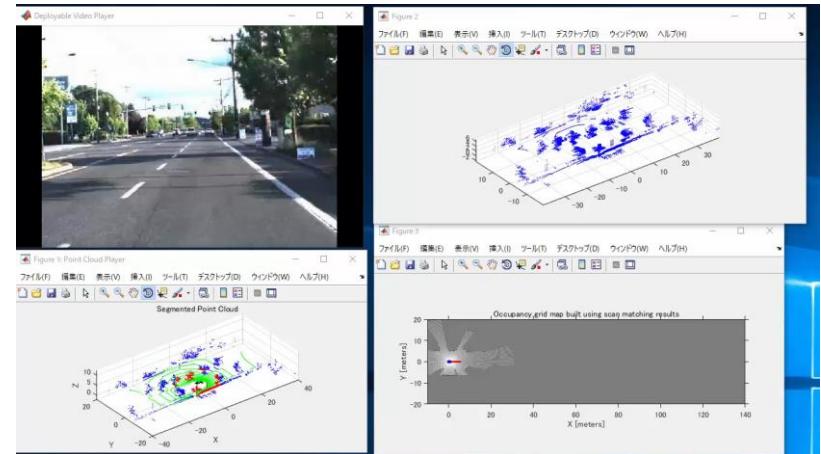
R2017a

- Normal Distributions Transform(NDT)

- 地図空間をグリッド分割し、正規分布(平均・分散)で近似
- 対応要素をニュートン法で探索
- グリッドサイズによって収束性能が大きく異なる



関数を活用することで点群同士の容易なマッチングを実現



NDTによる自己位置推定の例

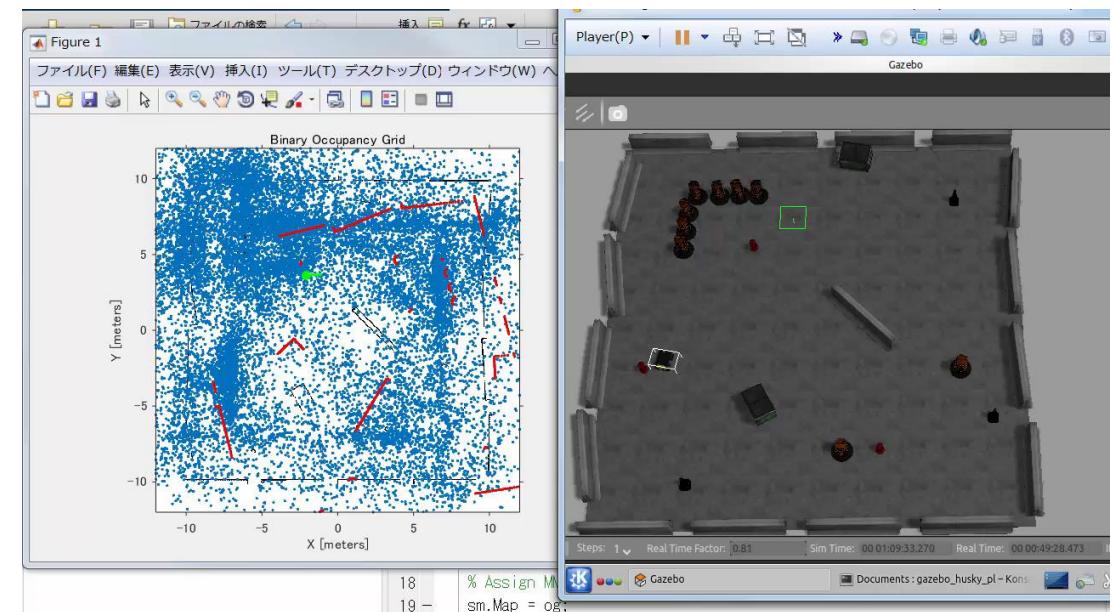
```
pose =  
matchScans (currScan, refScan)
```

# 自己位置推定：パーティクルフィルタ

Robotics System Toolbox

R2016a

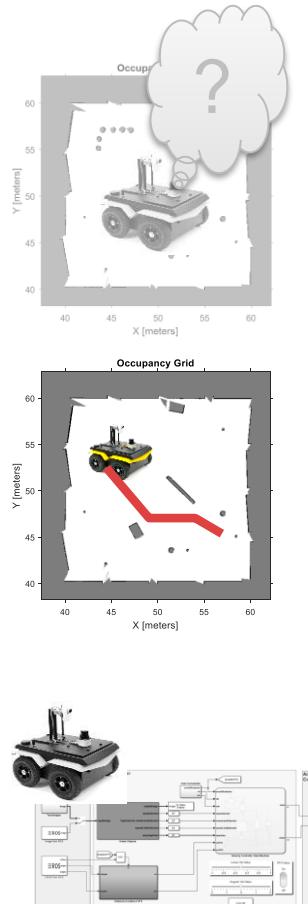
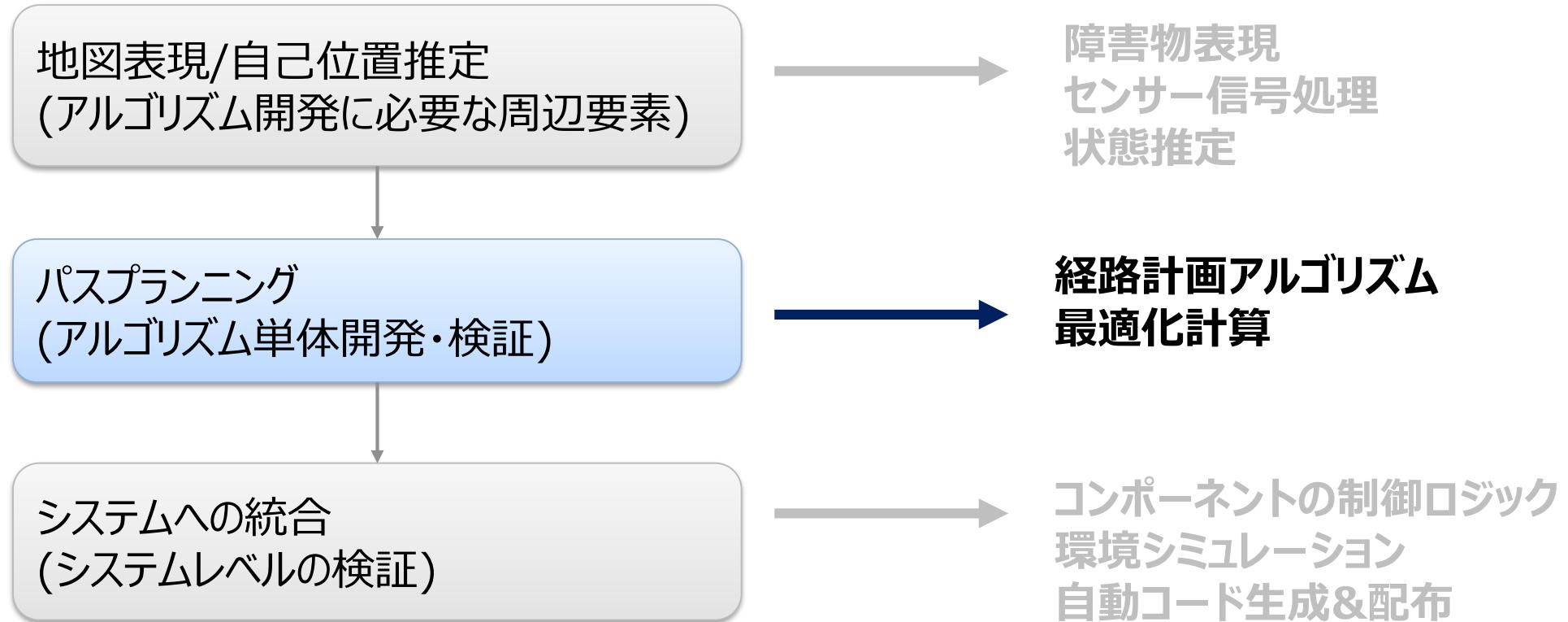
- モンテカルロローカリゼーション
  - パーティクルフィルタによる位置推定
  - 非線形システムにも幅広く応用可能
  - ただし、計算コストが高い
- アルゴリズム概要
  - パーティクル散らす(確率分布を近似)
  - 運動モデルでパーティクルの挙動を予測
  - 尤度関数で予測したパーティクルのもつもらしさを決定
  - もつもらしさにしたがって再度パーティクル生成(リサンプリング)



```
mcl = robotics.MonteCarloLocalization()
```

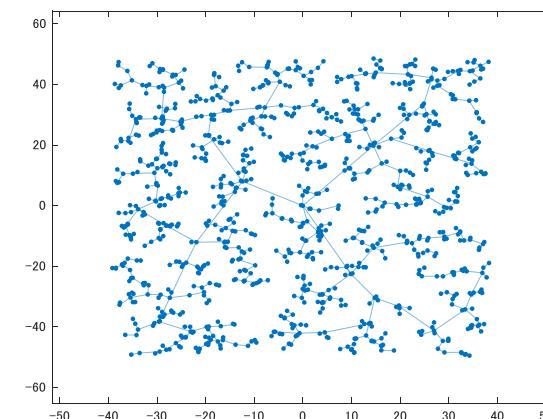
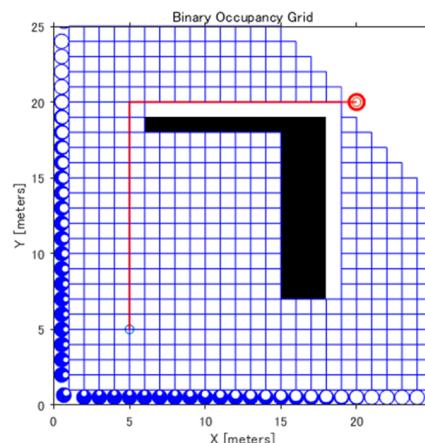
逐次的にロボットの姿勢を推定

# パスプランニングアルゴリズムの開発フロー



# パスプランニングアルゴリズム

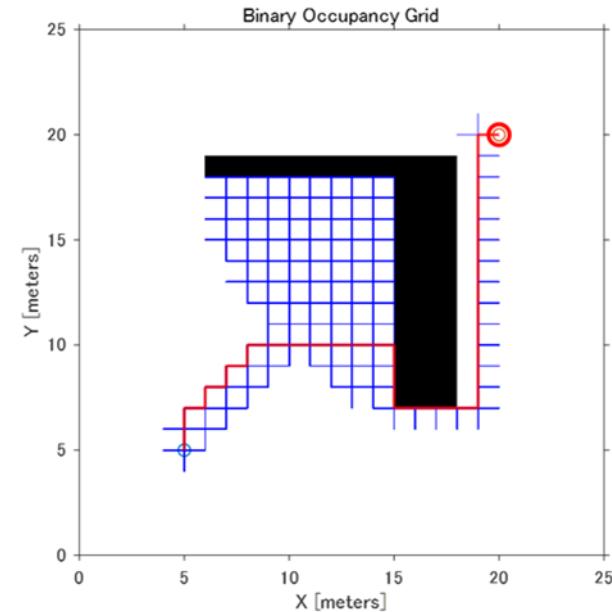
- グリッドベースの探索
  - 探索空間をグリッドで分割
  - カーナビや電車の経路探索など広く利用
  - 探索空間が連続の場合や  
次元数が高い場合には時間がかかる
- 代表的なアルゴリズム
  - Dijkstra法、**A\*** (**A Star**)
- サンプルベースの探索
  - 探索空間をランダムにサンプリング
  - サンプリングした点をつなげてパスを作る
  - 探索空間が連続の場合や  
多次元の場合に有効
- 代表的なアルゴリズム
  - **PRM, RRT, RRT\***



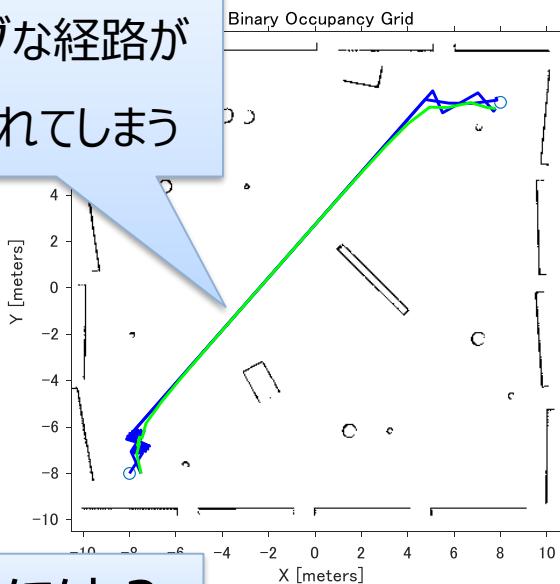
# パスプランニング：A\*アルゴリズム

- グリッドベースの経路探索手法
- ダイクストラ法の拡張版
- コストを最小化する経路：
  - $f(n) = g(n) + h(n)$
- $g(n)$ :スタートノードから  $n$  までの最小コスト
- $h(n)$ :ヒューリスティック関数
- ヒューリスティック関数は経験的に決める  
例：移動体ならゴールまでのユークリッド距離

$$h(n) = \sqrt{x_n^2 + y_n^2}$$



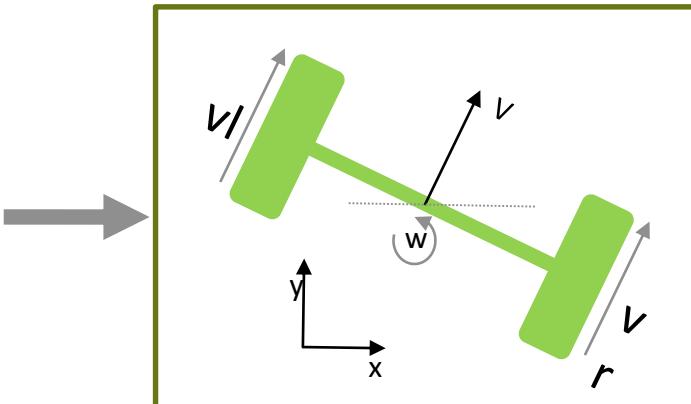
ジグザグな経路が  
生成されてしまう



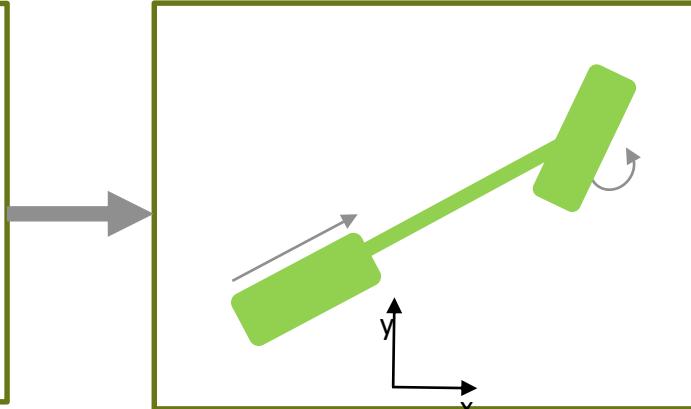
車両追従性を考慮したパスを生成するには？

# 移動体の数式モデル(運動学モデル)

Symbolic Math Toolbox™



$$\begin{aligned}x_{t+1} &= x_t + v_{cmd} \cos \theta \Delta t \\y_{t+1} &= y_t + v_{cmd} \sin \theta \Delta t \\\theta_{t+1} &= \theta_t + w_{cmd} \Delta t\end{aligned}$$



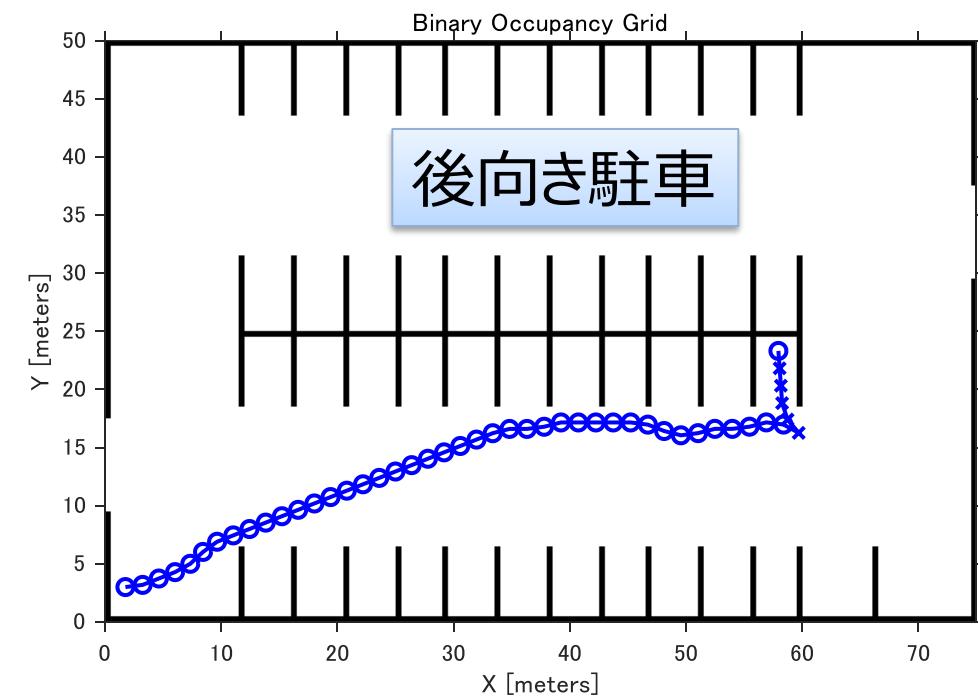
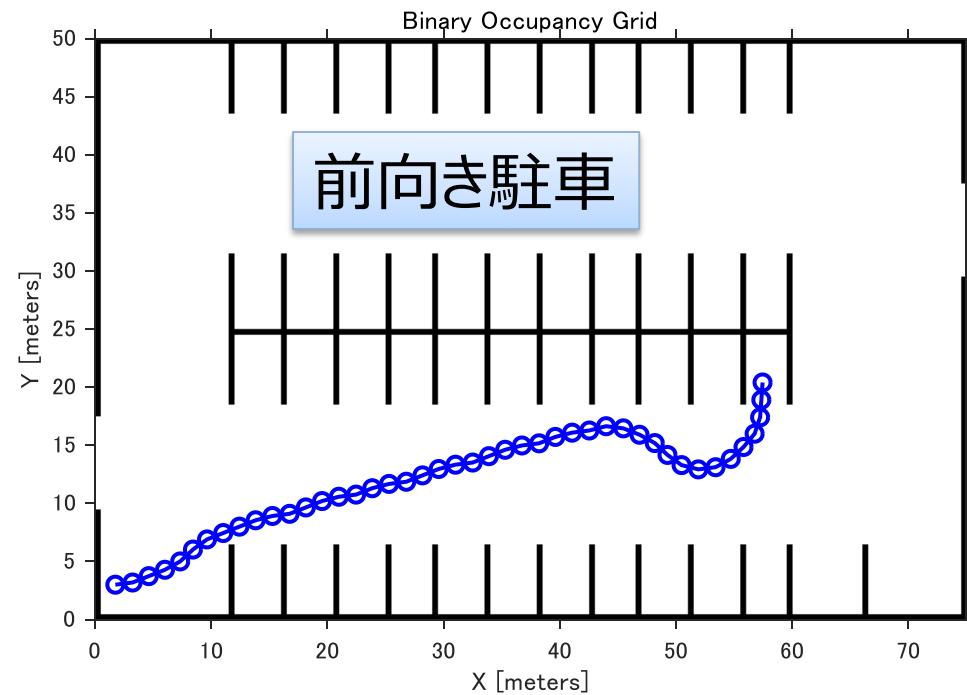
```
%% Dynamic model
Vx_dot = +theta_dot*Vy + 1/Mass * ( (Fxfl+Fxfr)*cos_delta ...
- (Fyfl+Fyfr)*sin_delta...
+ (Fxrl+Fxrr) ...
- Ca*Vx^2);

Vy_dot = -theta_dot*Vx + 1/Mass * ( (Fyfl+Fyfr)*cos_delta ...
+ (Fxfl + Fxfr)* sin_delta ...
+ (Fyrl+Fyrr));
```

運動方程式の導出など、数式処理も同一環境で実現

# パスプランニング：A\*アルゴリズム

- 2輪等価運動学モデルを使った自動駐車

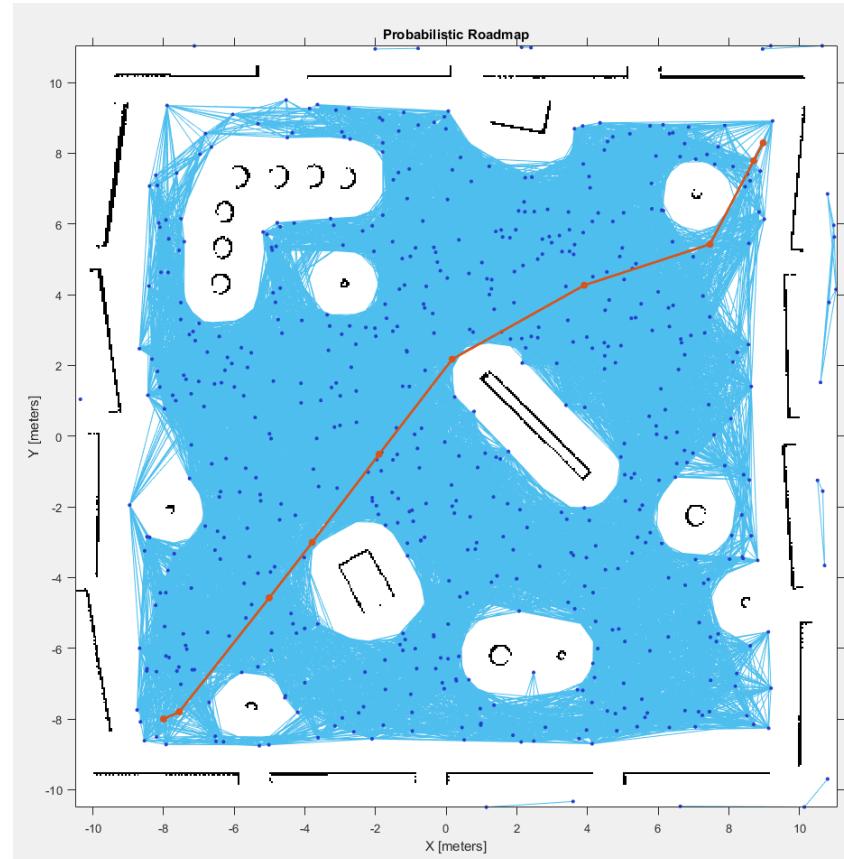


車両運動学を考慮した経路生成が可能  
ただし、探索空間の次元数が増えると計算時間が爆発⇒サンプルベース探索

# パスプランニング：PRM(Probabilistic Roadmap)

Robotics System Toolbox

- Nサンプルランダムに状態を生成
- 生成したサンプル同士を結合し、障害物にぶつからないエッジを生成
- 構築されたグラフから最短経路を探索

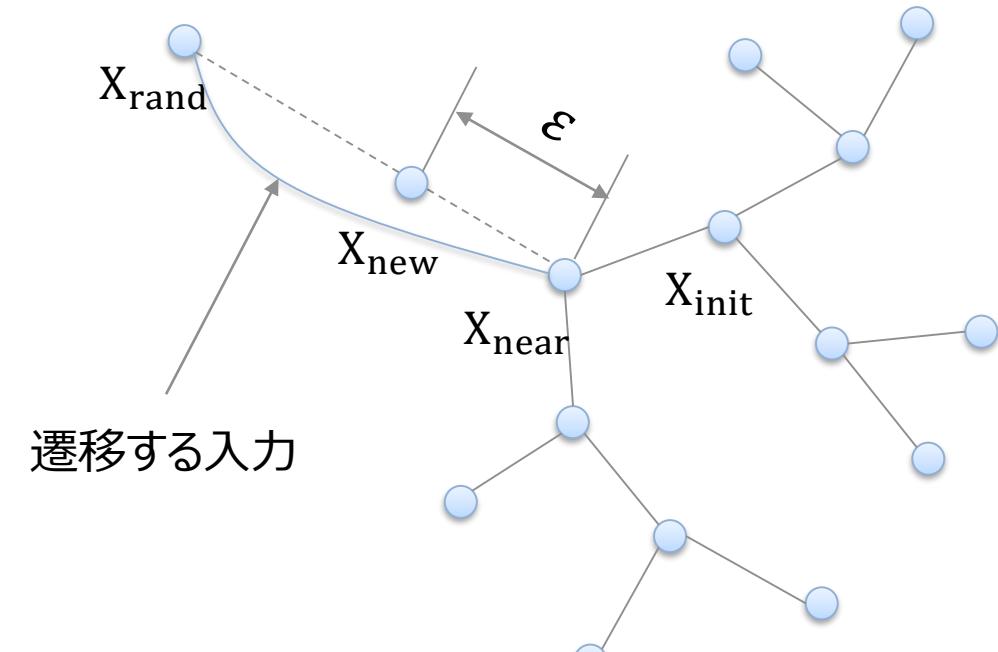


```
planner = robotics.PRM;
```

Toolbox内蔵のパスプランニングアルゴリズムを活用してすばやく開発

# パスプランニング：RRT(Rapidly-Exploring Random Tree)

- サンプリングベースの経路探索手法
  - 初期地点を $X_{init}$ としてグラフを構築
  - 全状態空間からランダムに1サンプルとり、 $X_{rand}$ とする
  - $X_{rand}$ に一番近いノードを $X_{near}$ とする
  - $X_{near}$ から $X_{rand}$ に遷移する入力 $u$ を求める
  - $X_{near}$ から $X_{rand}$ に直線を引き、距離 $\epsilon$ の点を $X_{new}$ としてグラフに追加
  - これを繰り返してゴールに到達するまでグラフを拡張していく



遷移する入力

グラフベースのアルゴリズムをどのように実装すればよいか？

# グラフオブジェクトによる無向/有向グラフサポート

MATLAB®

R2015b

## グラフオブジェクトの作成と最短経路探索

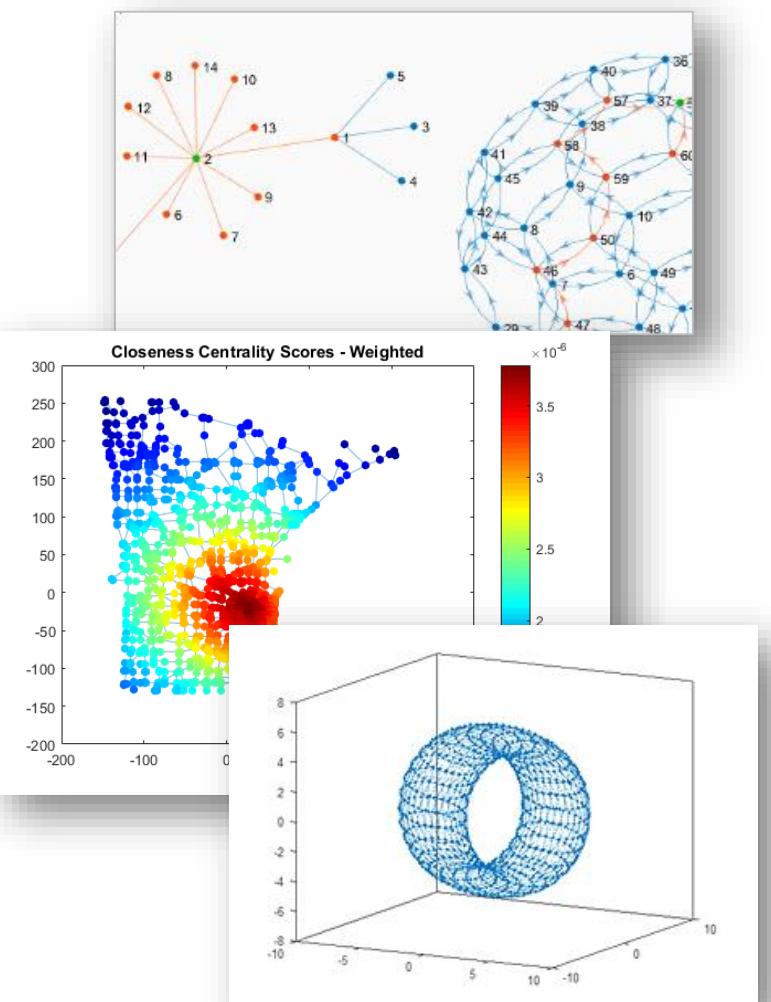
```
G = digraph(lhstotal,rhstotal,'OmitSelfLoops');
G.Nodes = pagenames;

cleve = shortestpath(G,'Kevin_Bacon','Cleve_Moler');
jack = shortestpath(G,'Kevin_Bacon','John_N._Little');
```

## グラフの可視化

```
plot(G)
```

- 3次元の可視化サポート
- 連結要素の判定 R2016b
- グラフの同型性判定



グラフオブジェクトを利用してパスプランニングアルゴリズム開発

# パスプランニング：RRTを独自に実装

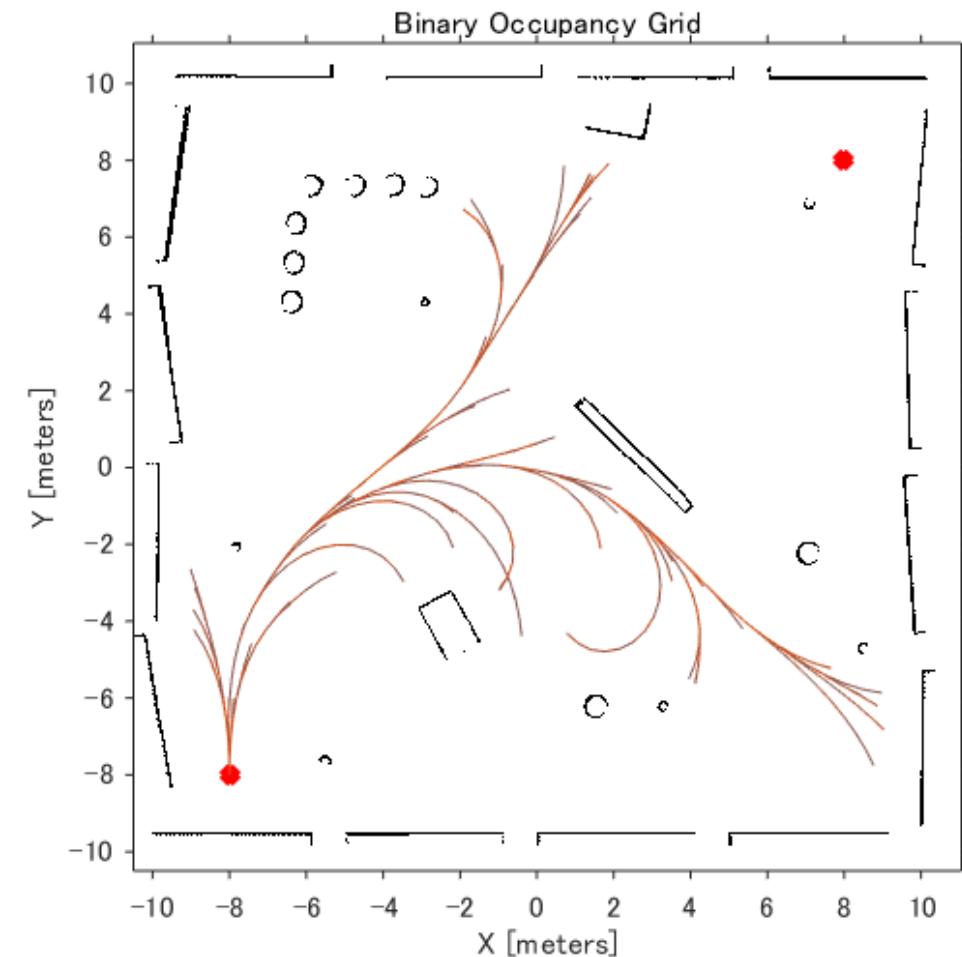
```
% Define root state [x,y]
startState = [0;0];

% Define boundaries for each state
statesMin = [-40; -50];
statesMax = [ 40; 50];

% Create graph
g = graph;

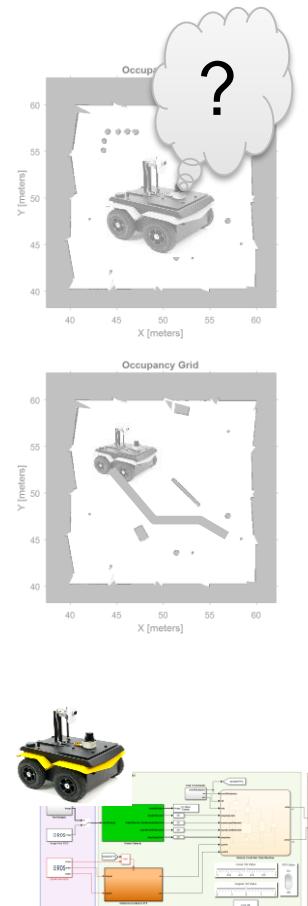
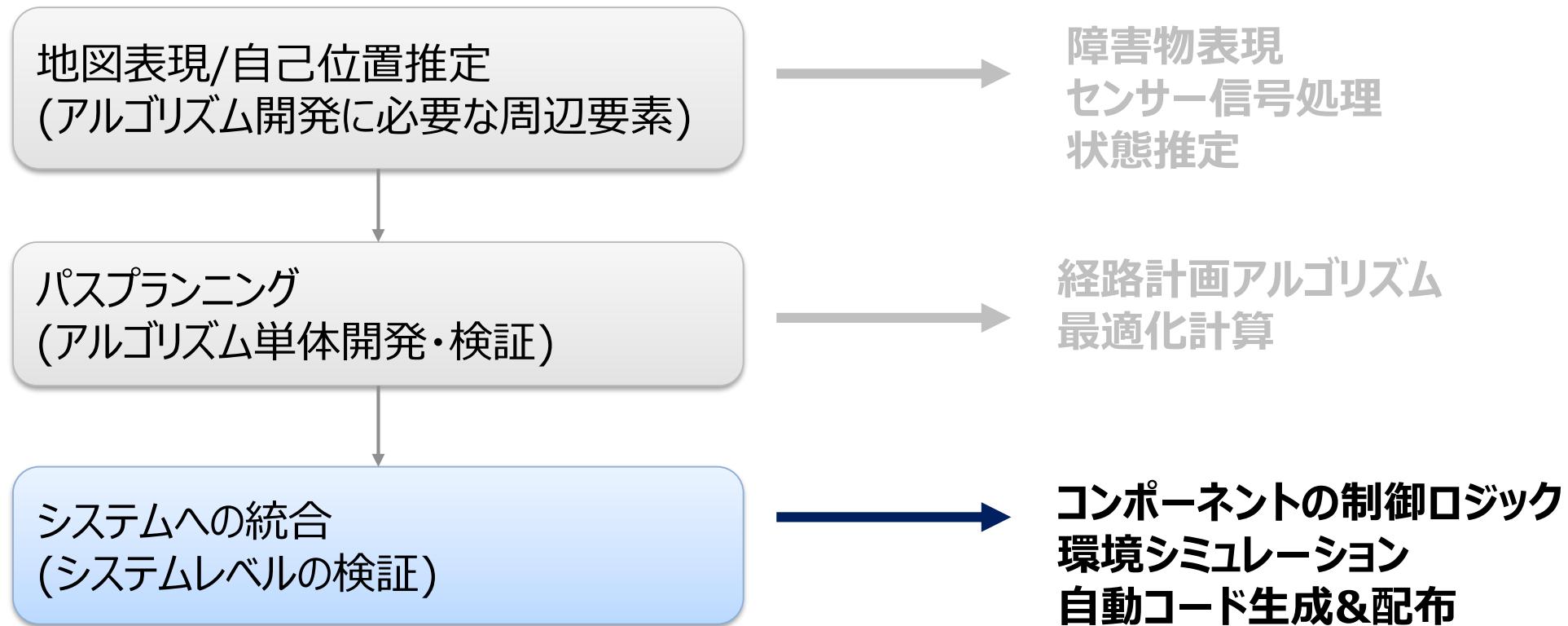
% Initialize tree with root node
g = addnode(g,1);
g.Nodes.State = {startState};

% Grow tree
stepSize = 0.5;
for ii = 1:1000
    randomState = generateRandomState(statesMin,
statesMax);
    g = extend(g, randomState, stepSize);
end
```

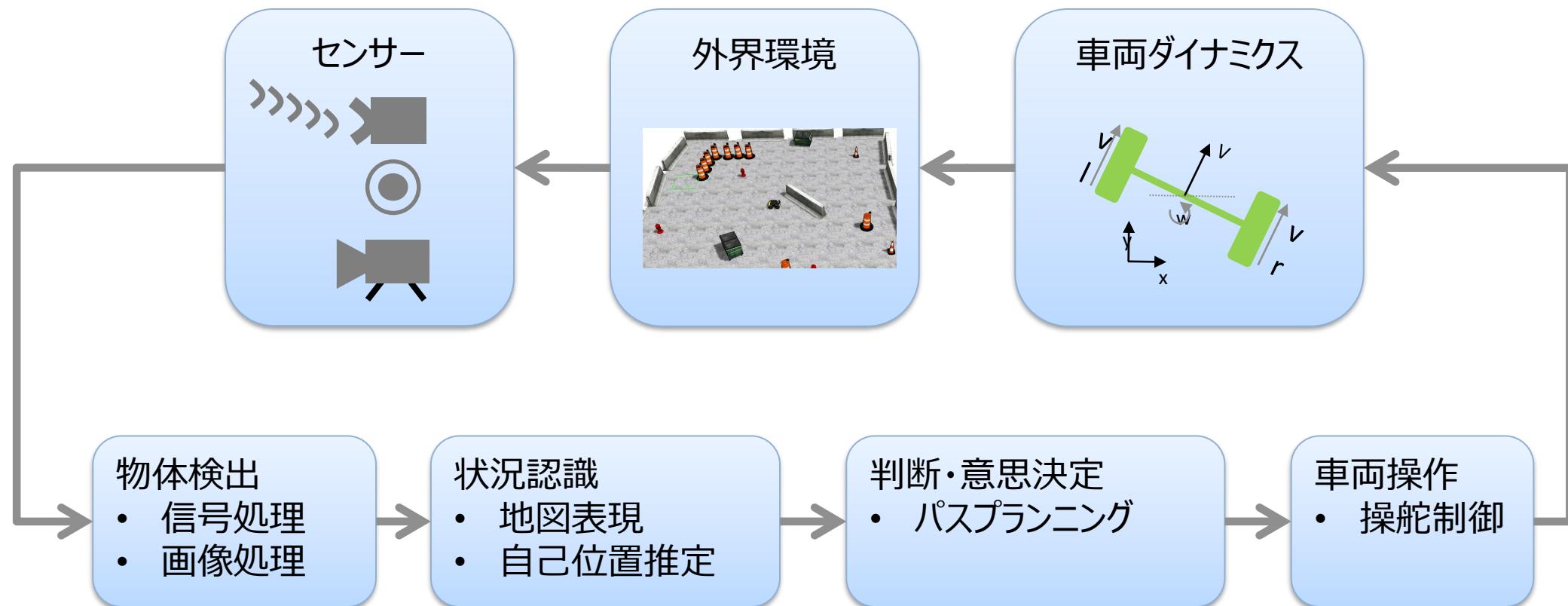


パスプランニングアルゴリズムを独自に実装可能

# パスプランニングアルゴリズムの開発フロー



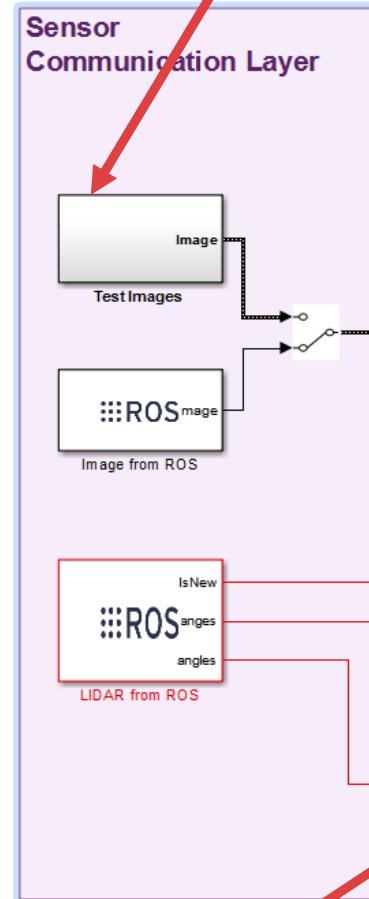
# 自律システムの検証・プロトタイピング



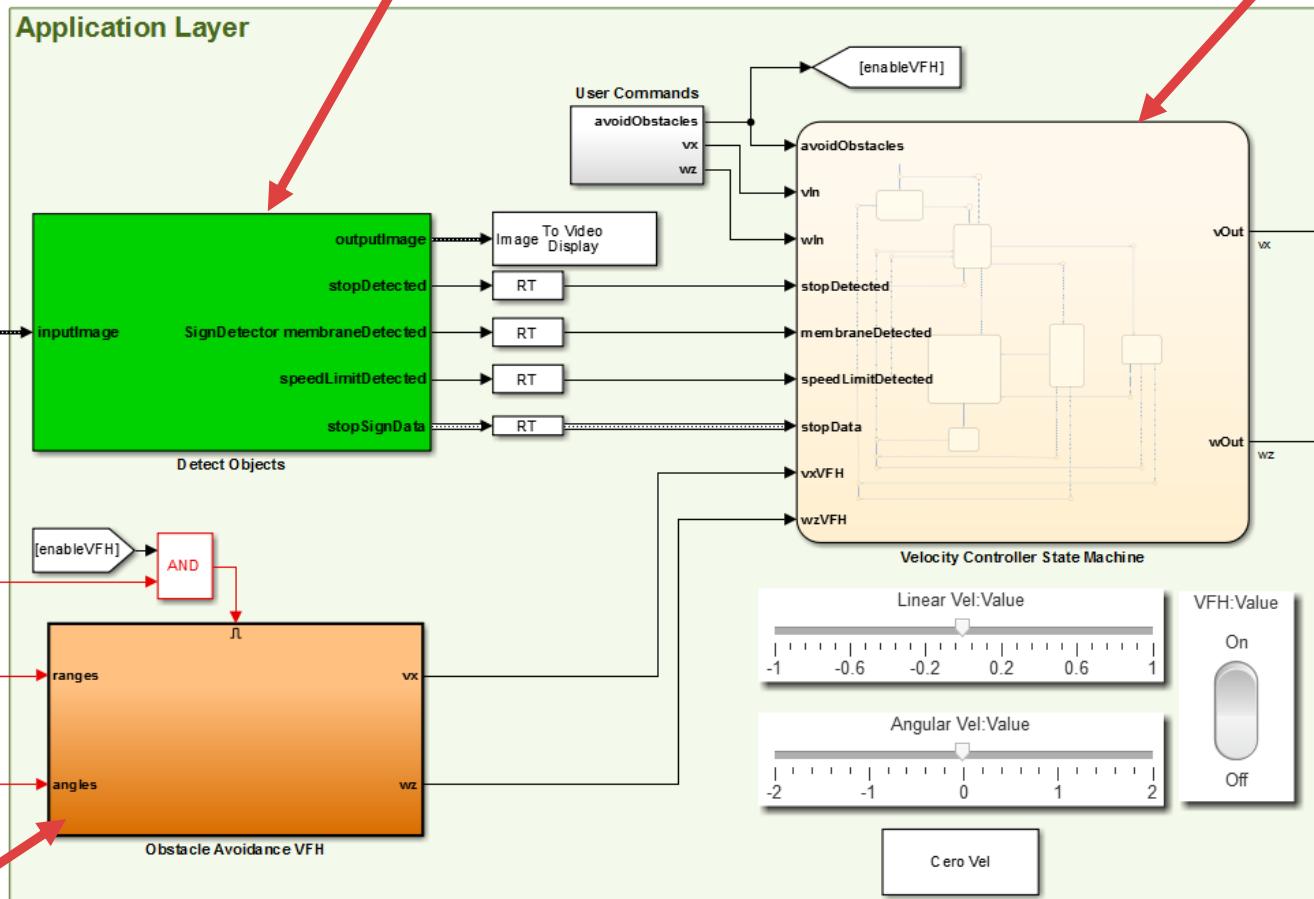
各コンポーネントを統合してシステムレベルの検証

# システムレベル設計環境

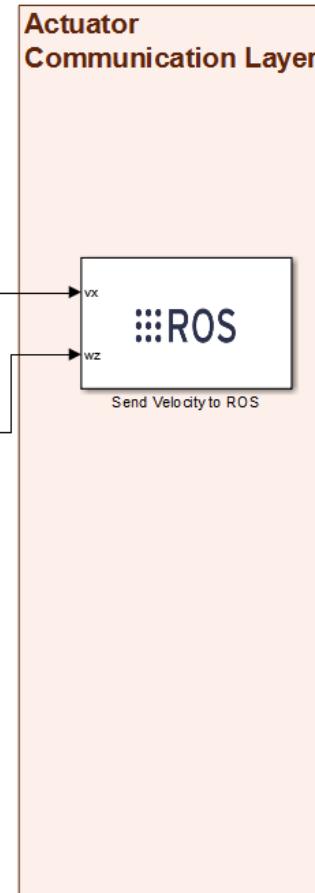
ROSによるデータ交換



物体識別器



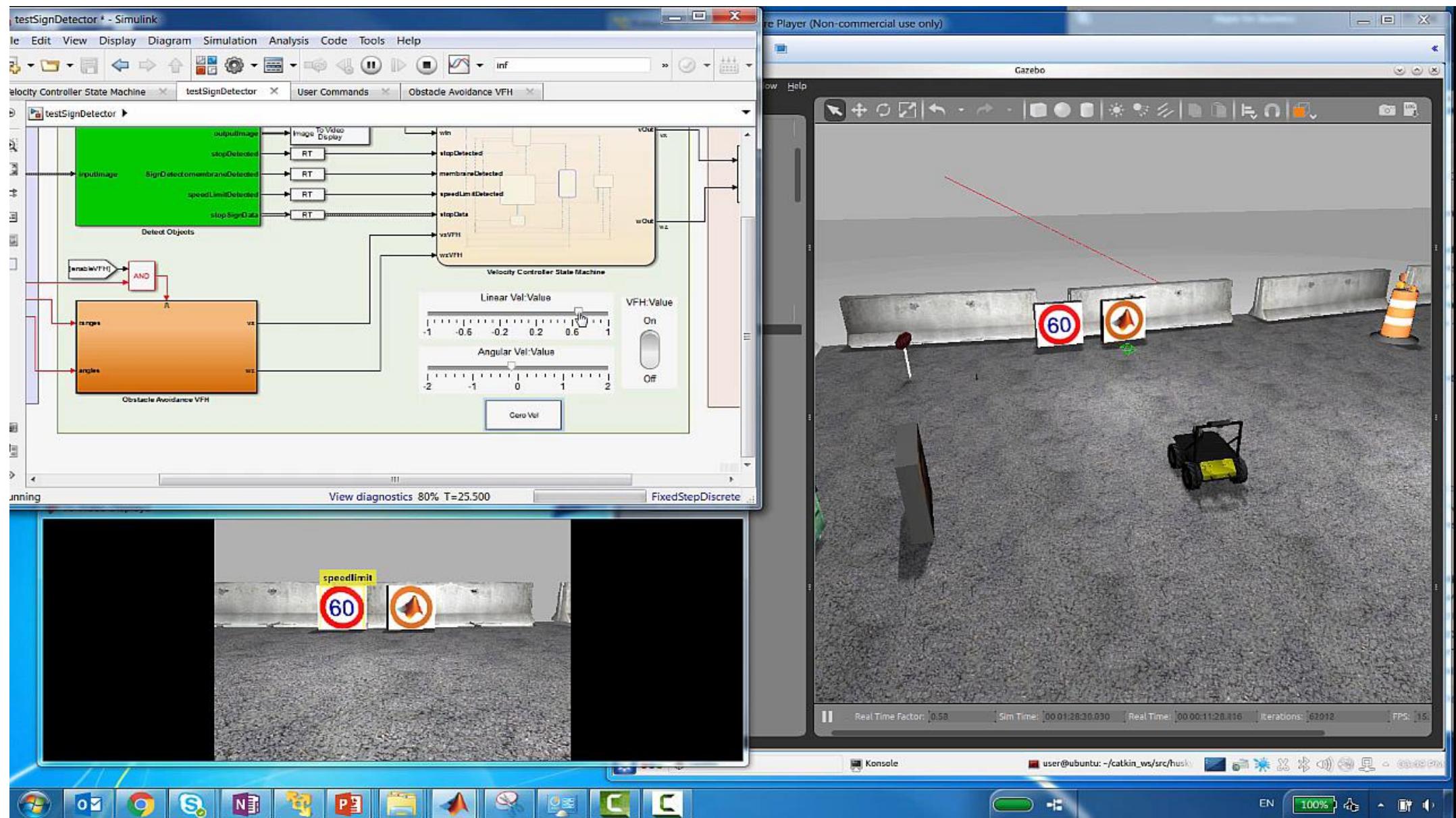
パスプランニングアルゴリズムや  
ステートマシンによる状態管理



衝突回避

ブロック線図環境で直感的にコンポーネントを配置・統合

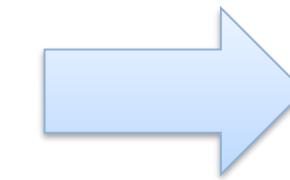
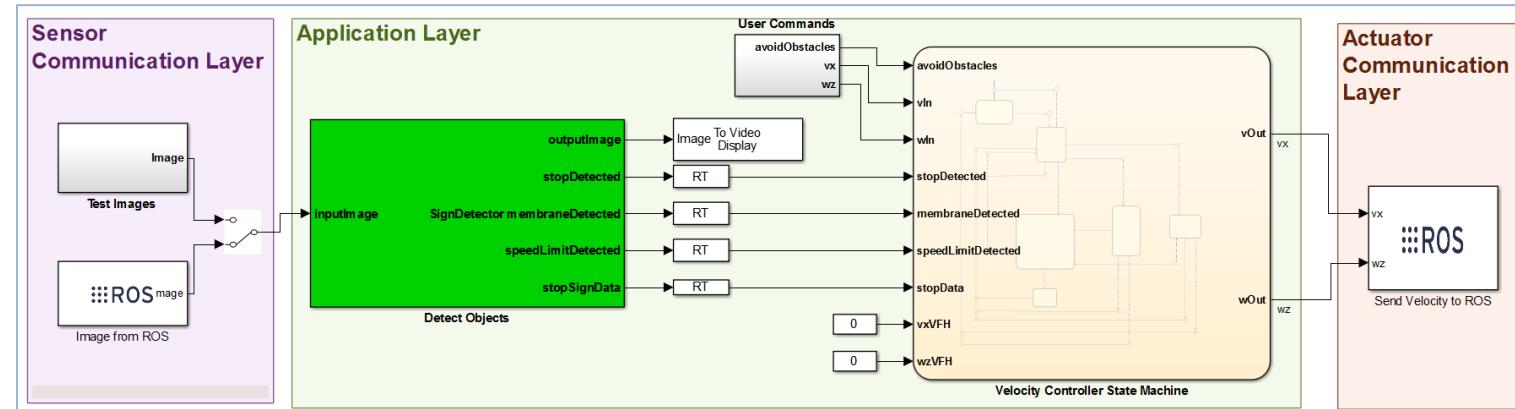
# ROSによる協調シミュレーション



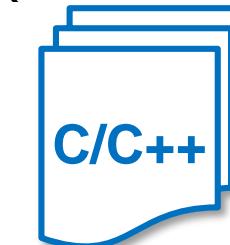
# コード生成&実機プロトタイピング

MATLAB Coder™, Simulink Coder™  
Embedded Coder®  
Simulink Report Generator™

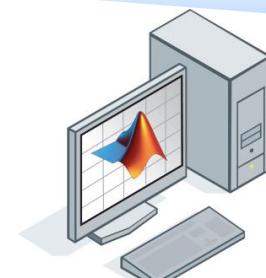
- コード生成機能でROSノードを生成し、ROS対応ロボットに直接実装



ROSノード  
(Cコード生成)



ROSシステムに  
ROSノードとして  
実装



開発用PC  
(MATLAB/Simulink)

作成したモデルを自動コード生成機能を使って容易に実装

## まとめ

- ・ さまざまな地図表現や自己位置推定機能などすぐに使える周辺機能
- ・ グラフの取り扱いから可視化までパスプランニングアルゴリズム開発使える柔軟な機能
- ・ システムレベルシミュレーションによる動作検証とコード生成による実機検証



MATLAB/Simulinkをパスプランニングアルゴリズムの開発にお役立てください！

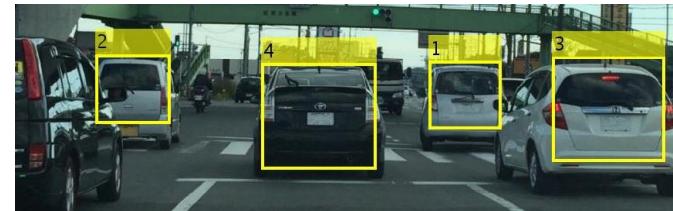
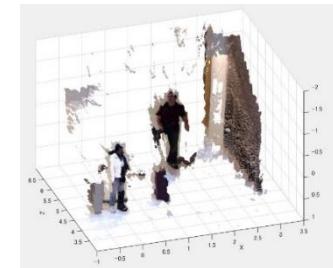
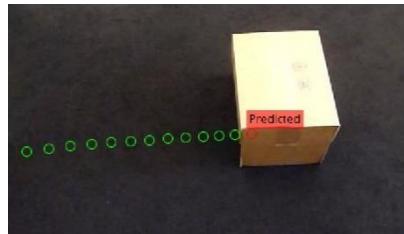
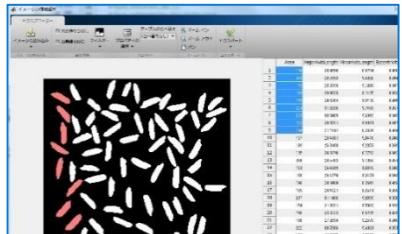
# 画像処理・コンピュータービジョン無料セミナー

申し込みは弊社ウェブサイトより

<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・コンピュータービジョン・機械学習

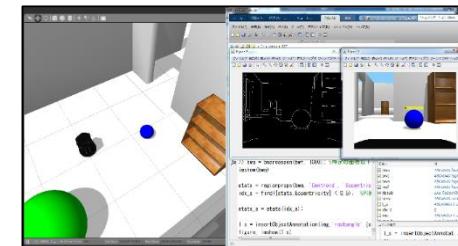
- 日時：2017年11月21日 13:30-17:00 (受付 13:00-)
- 場所：品川シーザンテラスカンファレンス（タワー棟3F カンファレンス A+B+C）  
(アクセス：JR品川駅 港南口（東口）より徒歩6分 <http://www.sst-c.com/access/index.html>)
- **画像処理、コンピュータービジョン、機械学習の機能をご紹介！**
  - MATLABではじめる画像処理ワークフロー
  - 例題で実感するMATLABの画像処理機能
  - MATLABで試す！機械学習の応用例



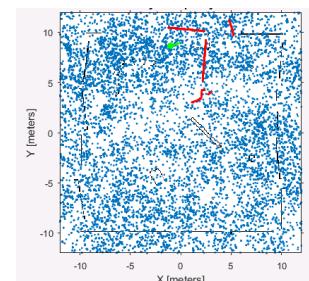
# ロボットアルゴリズム開発ソリューション

## Robotics System Toolbox

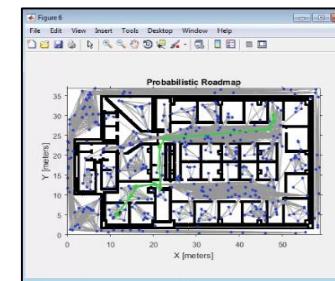
- ROSのインターフェイス提供
  - MATLABをROSマスター、ノードとして起動
  - 直接ROSネットワークに接続して検証
- ROSノード生成
  - SimulinkモデルからC++ ROSノードを生成
- ロボットアルゴリズム開発の支援
  - オイラー角、クオータニオン、座標変換などの便利な関数群
  - パスプランニング、VFH+、モンテカルロローカリゼーション(MCL)などの高度な移動用ロボットアルゴリズム
  - ツリー構造表現、逆運動学解析などのマニピュレーターアルゴリズム



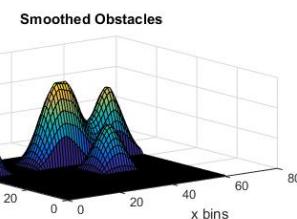
シミュレータや実機とのROS連携



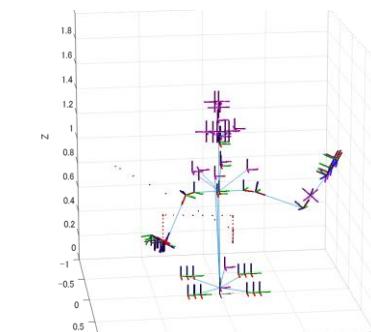
自己位置推定(AMCL)



確率的ロードマップ法(PRM)



衝突回避(VFH+)



ロボットマニピュレーターアルゴリズム開発

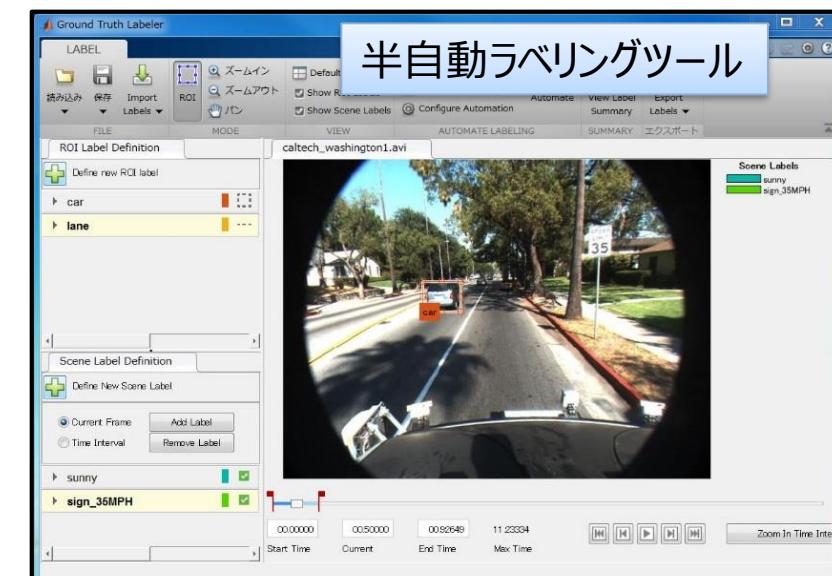
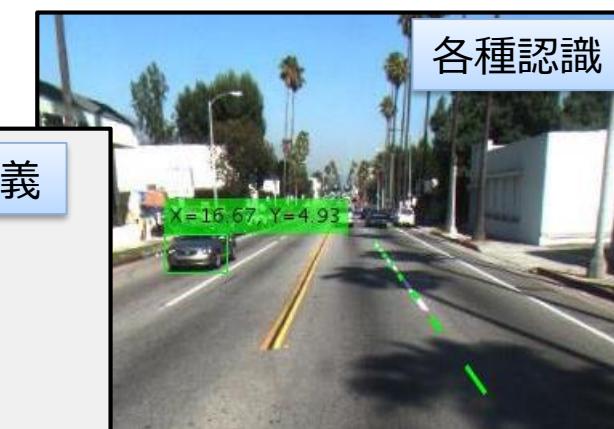
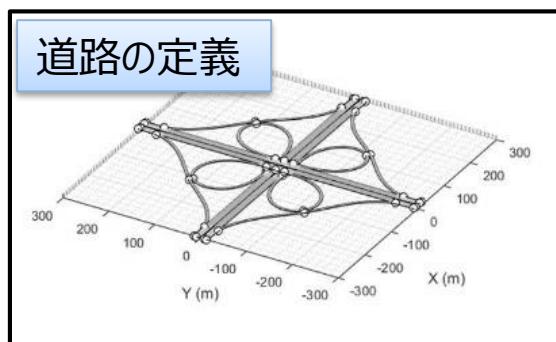


# 自律移動/ADAS/自動運転向けアルゴリズム開発検証

R2017a

## Automated Driving System Toolbox™

- データの可視化
  - 鳥瞰図変換、白線検出・可視化
- 自動車での画像処理・トラッキングに特化したアルゴリズム
  - 拡張カルマンフィルタ、Unscentedカルマンフィルタ
- Ground Truth Labeler
  - 動画に対してラベル付け
- テストシナリオ生成



# 物理モデリング・制御ロジック

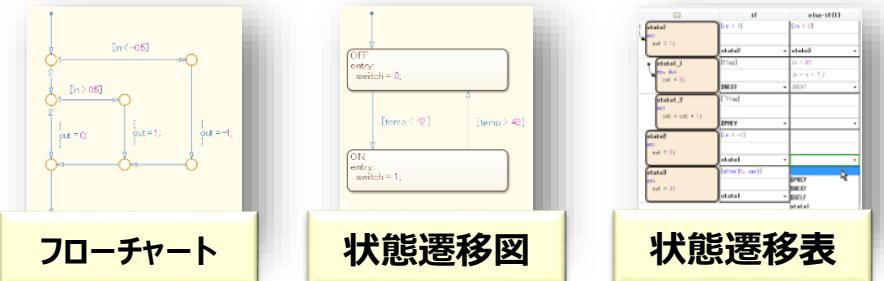
## Simulink

- ブロック線図モデリング
- 豊富なブロックライブラリ
- 時系列の統合シミュレーション環境



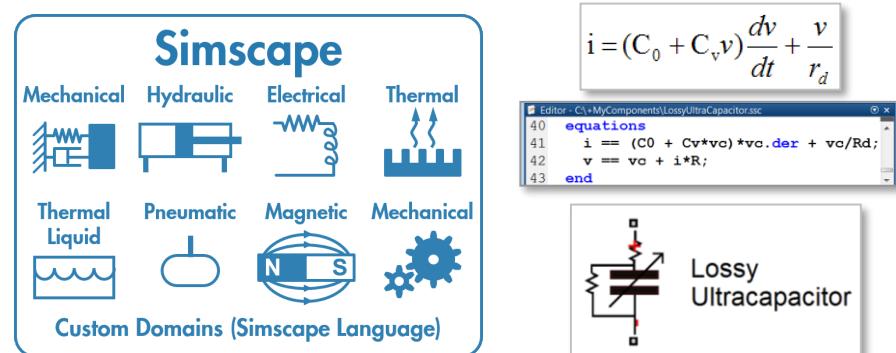
## Stateflow

- モードロジックの素早い設計 & 検証
- 状態遷移図、表、フローチャート機能
- コード生成、モデル検証オプション機能



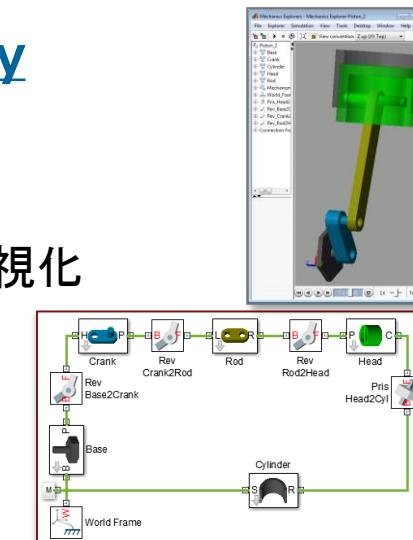
## Simscape

- 複合物理領域のプラントモデリング



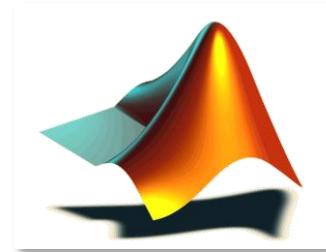
## Simscape Multibody

- 機構 (3-D)
- 剛体・ジョイント
- アニメーションによる可視化
- CADインポート機能
- 順動力学解析
- 逆動力学解析

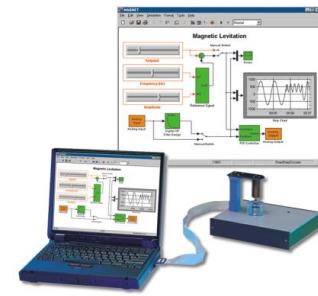


# 自動コード生成ソリューション

## MATLAB Coder

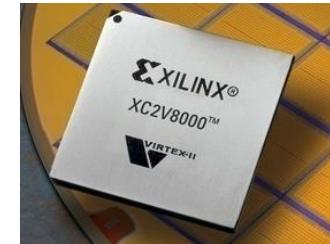


## Simulink Coder



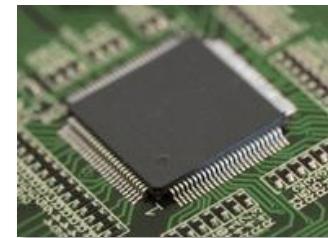
RCP/HILS

## HDL Coder



FPGA/ASIC

## Embedded Coder



MCU/DSP

## Simulink PLC Coder



PLC