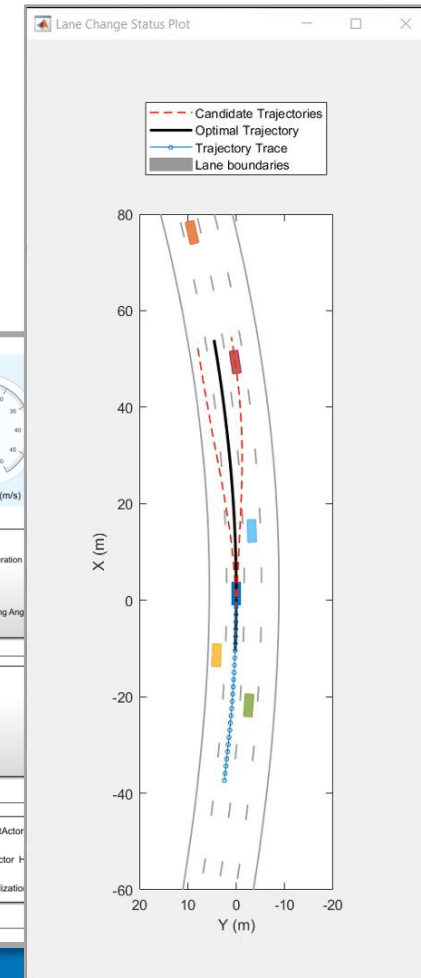
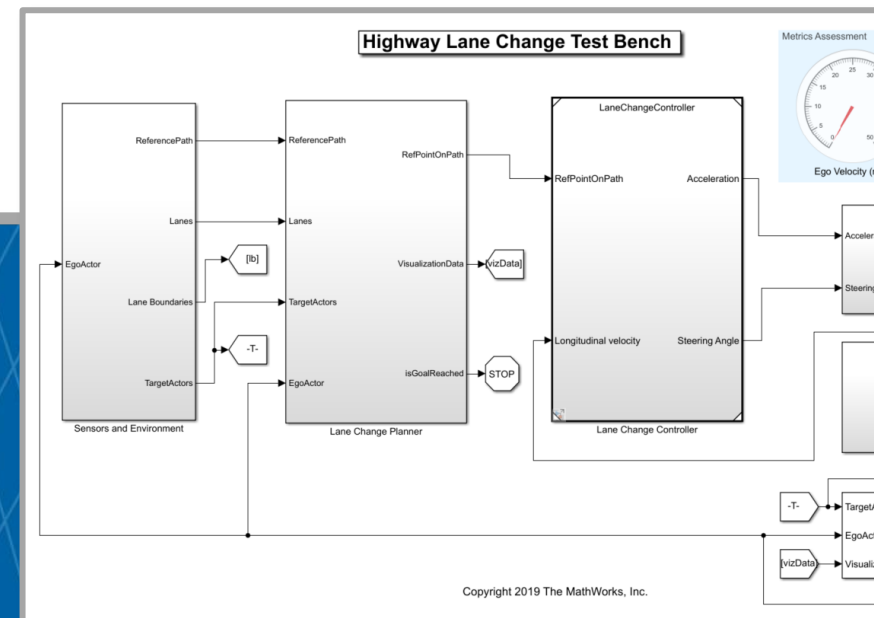
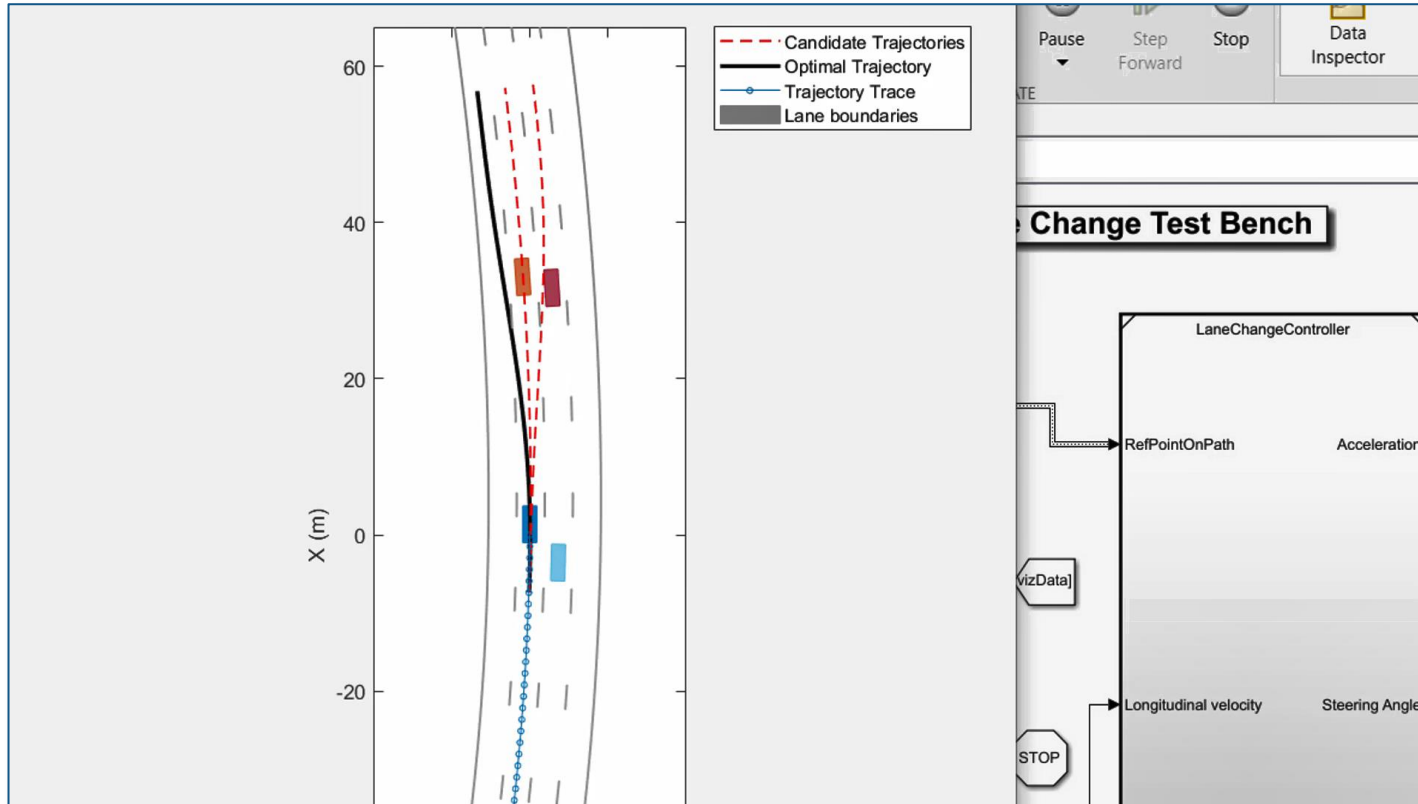


# Developing Planning and Controls for Highway Lane Change Maneuvers

**Seo-Wook Park**  
 Application Engineering  
 ADAS / Automated Driving  
[spark@mathworks.com](mailto:spark@mathworks.com)



# Highway Lane Change



- Automated lane change maneuver (LCM) system for highway driving scenario (straight & curved roads)
- Generates a collision-free, optimal trajectory for lane change using trajectoryOptimalFrenet
- Behavior layer to configure motion planner
- State validator for collision checking
- Lane following controller using MPC

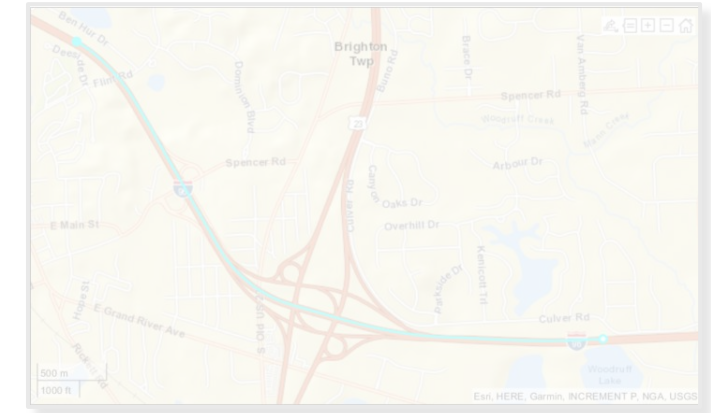
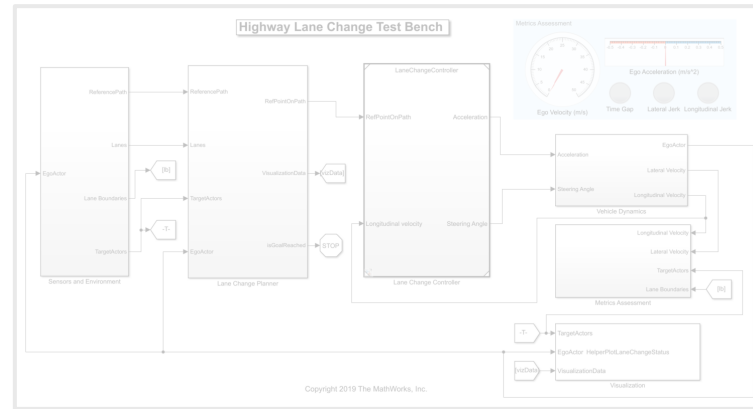
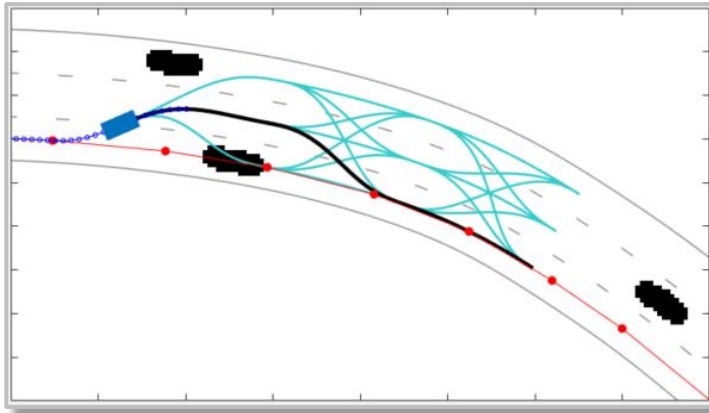
## [Highway Lane Change](#)

*Navigation Toolbox™*

*Model Predictive Control Toolbox™*

*Automated Driving Toolbox™*

# Highway Lane Change



## Learn motion planner

- Optimal trajectory generation in Frenet coordinate
- Planner parameters
- Simulate trajectory planning with occupancy map

## Integrate motion planner and controller

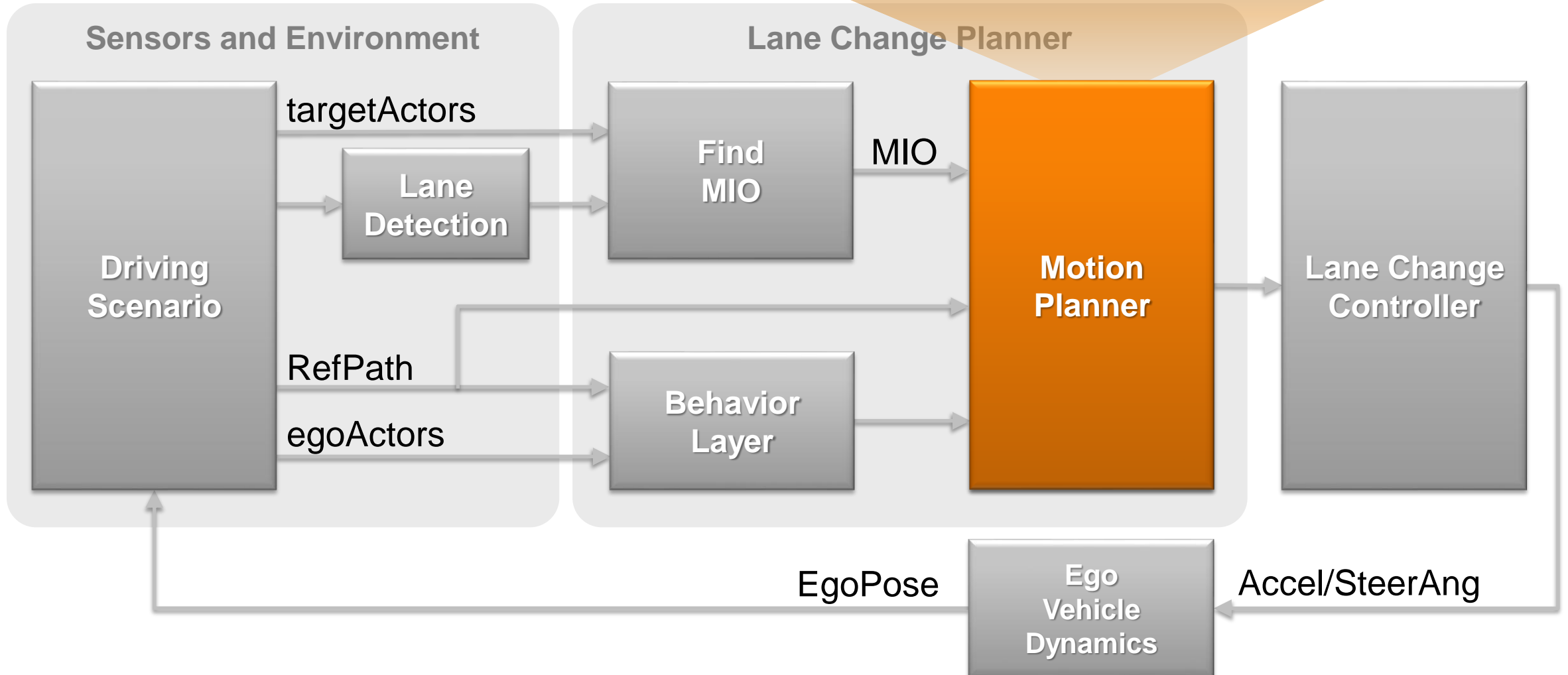
- Learn through reference example
- Architecture of highway lane change
- Simulate closed-loop controller with test scenarios

## Test with real-world scenario

- Create scenario from HD map
- Update setup script and model
- Run simulation with new scenario

# Architecture of Highway Lane Change

```
planner = trajectoryOptimalFrenet(refPath, validator)
```



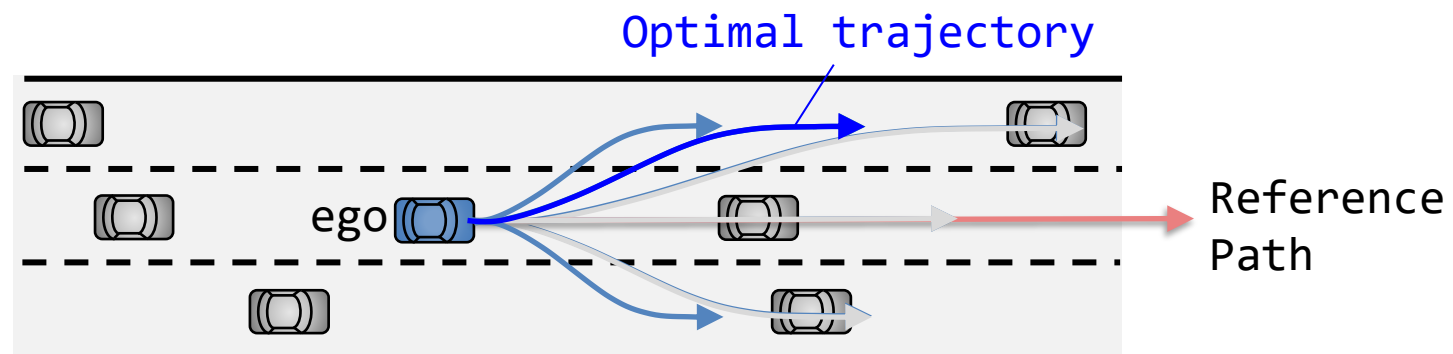
# Optimal Trajectory Generation

- trajectoryOptimalFrenet function generates an **optimal, feasible, and collision-free** trajectory for the reference path

```
planner = trajectoryOptimalFrenet(refPath, validator)
```

reference path

state validator  
for collision checking



# State Validator for collision checking

```
% Create a state validator object for collision checking
stateValidator = validatorOccupancyMap;
```

```
% Create and Assign binaryOccupancyMap to the state validator
stateValidator.Map = binaryOccupancyMap(width,height);
```

1) Assign 2-D occupancy grid map with binary values

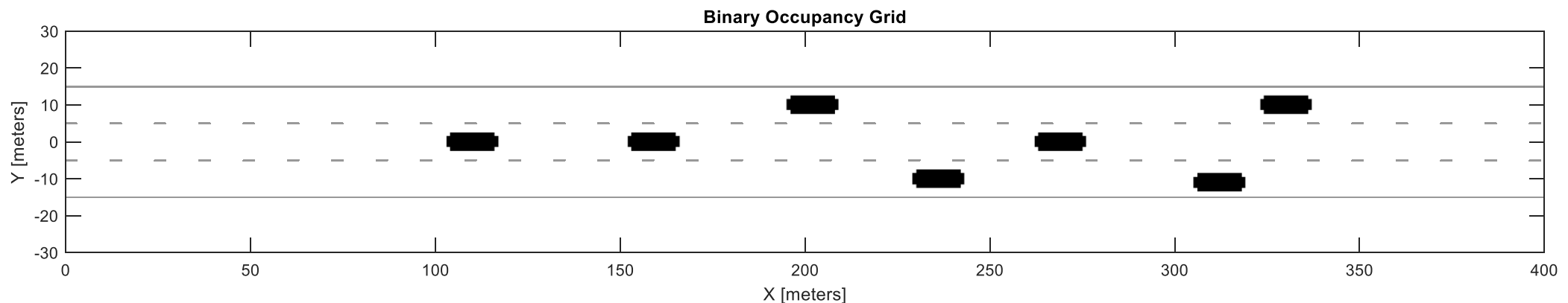
```
% Or, use occupancyMap
stateValidator.Map = occupancyMap(width,height);
```

2) Assign 2-D occupancy grid map with probabilistic values (0..1)

```
% Or, create a custom state validator object
stateValidator = MyCustomStateValidator;
```

3) Use custom state validator

```
classdef MyCustomStateValidator < nav.StateValidator & ...
    matlabshared planning_internal_EnforceScalarHandle
```



# Planner parameters

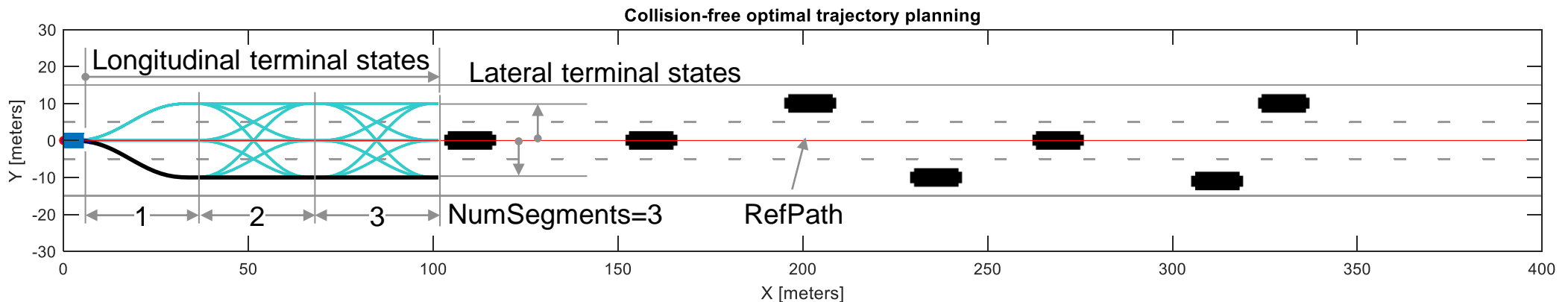
- Terminal states
- No. of longitudinal segments
- Feasibility parameters

```

% Assign terminal states for longitudinal state, lateral deviation,
% speed, time, and acceleration
planner.TerminalStates.Longitudinal = 100;
planner.TerminalStates.Lateral = -10:10:10;
planner.TerminalStates.Speed = 8;
planner.TerminalStates.Time = 7;
planner.TerminalStates.Acceleration = 0;

% Assign number of partitions for the longitudinal terminal state
planner.NumSegments = 3;

% Assign maximum acceleration and curvature values for feasibility
planner.FeasibilityParameters.MaxCurvature = 0.1;
planner.FeasibilityParameters.MaxAcceleration = 10;
    
```



# Optimal Trajectory Generation

```
trajectory = plan(planner, startFrenetState)
```

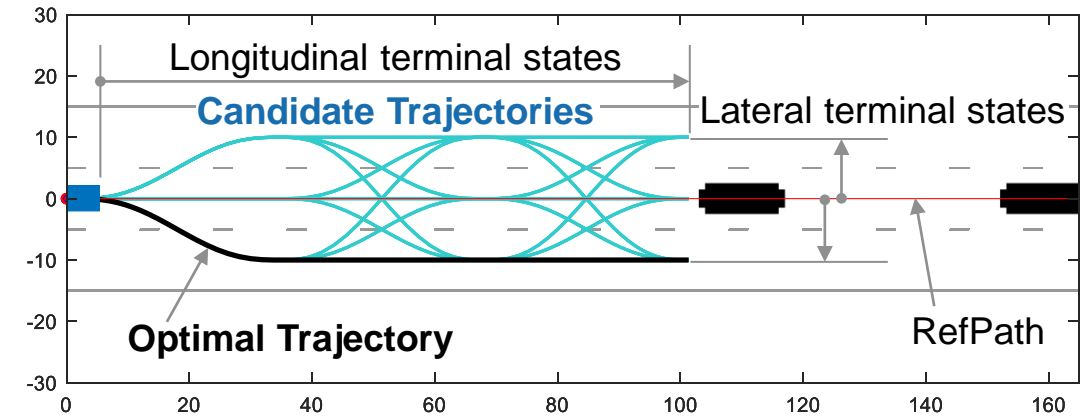
Samples multiple candidate trajectories for each pair of states (start, terminal states)

Evaluate **cost** for all candidate trajectories

Check **feasibility** for all candidate trajectories

Check **collision** using **state validator**

Choose a **feasible trajectory with the least cost**  
→ optimal, feasible, and collision-free trajectory



## Cost weights

- Time, Arc length, Deviation
- Lateral/longitudinal smoothness

## Feasibility parameters:

- MaxCurvature, MaxAcceleration

validatorOccupancyMap

Custom state validator



# Generate trajectory in Frenet coordinate using quintic polynomial

- Quintic polynomial

$$s(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

$$\dot{s}(t) = 5a_5 t^4 + 4a_4 t^3 + 3a_3 t^2 + 2a_2 t + a_1$$

$$\ddot{s}(t) = 20a_5 t^3 + 12a_4 t^2 + 6a_3 t + 2a_2$$

where  $s$  = longitudinal or lateral distance

- Start boundary conditions ( $t = 0$ )

$$a_0 = s_{start}$$

$$a_1 = \dot{s}_{start}$$

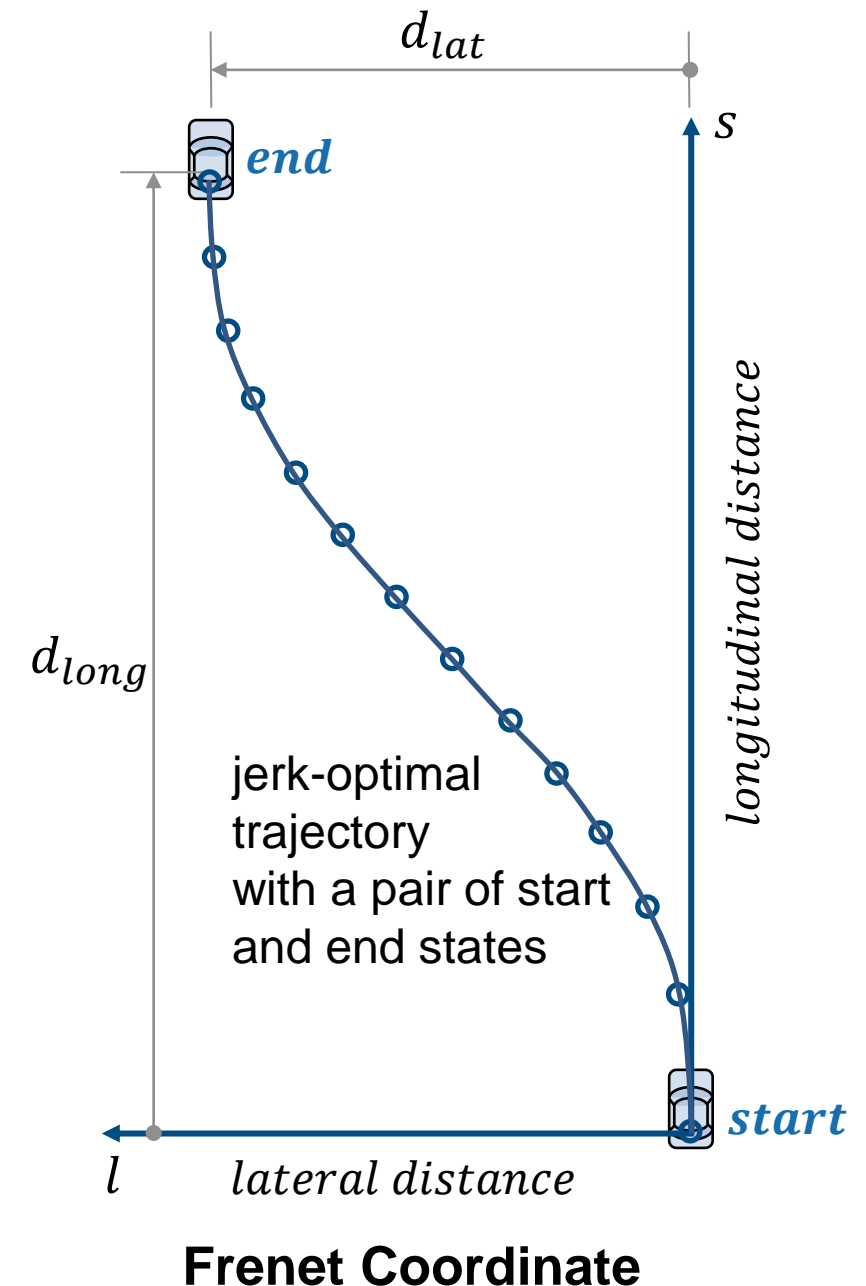
$$2a_2 = \ddot{s}_{start}$$

- End boundary conditions ( $t = t_f$ )

$$a_5 t_f^5 + a_4 t_f^4 + a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0 = s_{end}$$

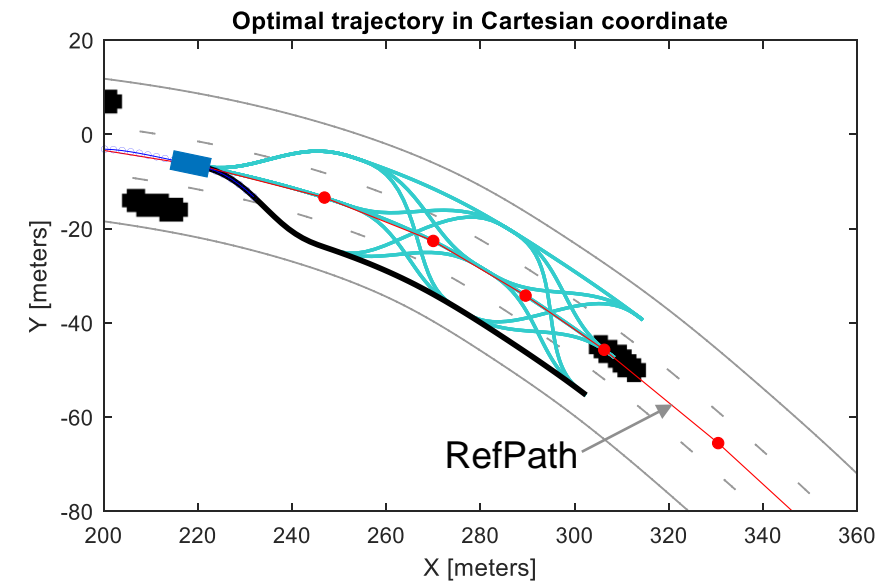
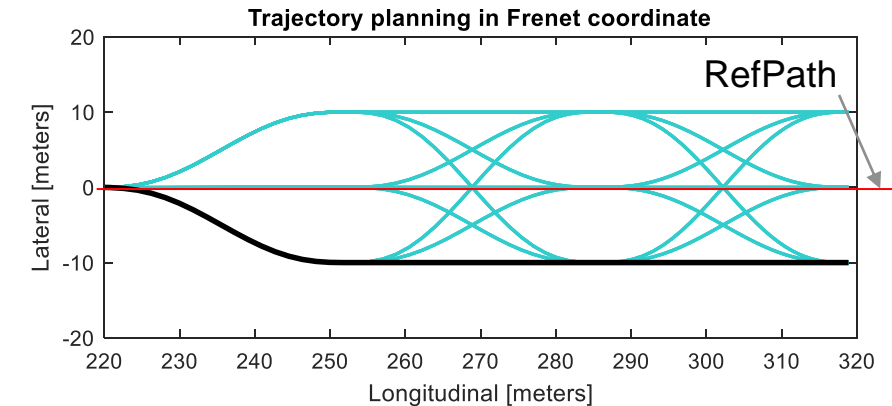
$$5a_5 t_f^4 + 4a_4 t_f^3 + 3a_3 t_f^2 + 2a_2 t_f + a_1 = \dot{s}_{end}$$

$$20a_5 t_f^3 + 12a_4 t_f^2 + 6a_3 t_f + 2a_2 = \ddot{s}_{end}$$

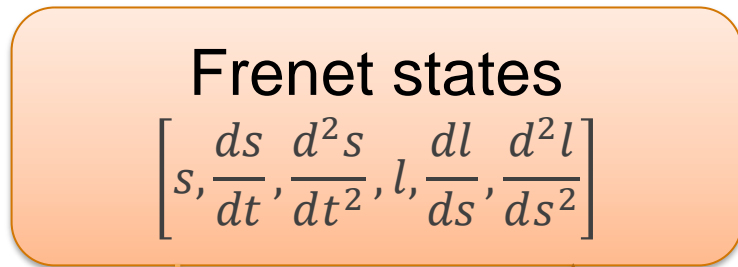


# Advantage of trajectory generation in Frenet coordinate

- Frenet system represents an object and its trajectory with respect to the reference path (road center or lane center).
- This approach dramatically simplifies the problem of trajectory generation when a car is traveling on a curved road.



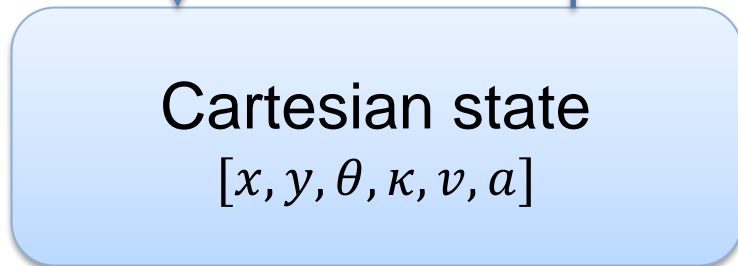
# Convert Frenet states $\leftrightarrow$ Cartesian states



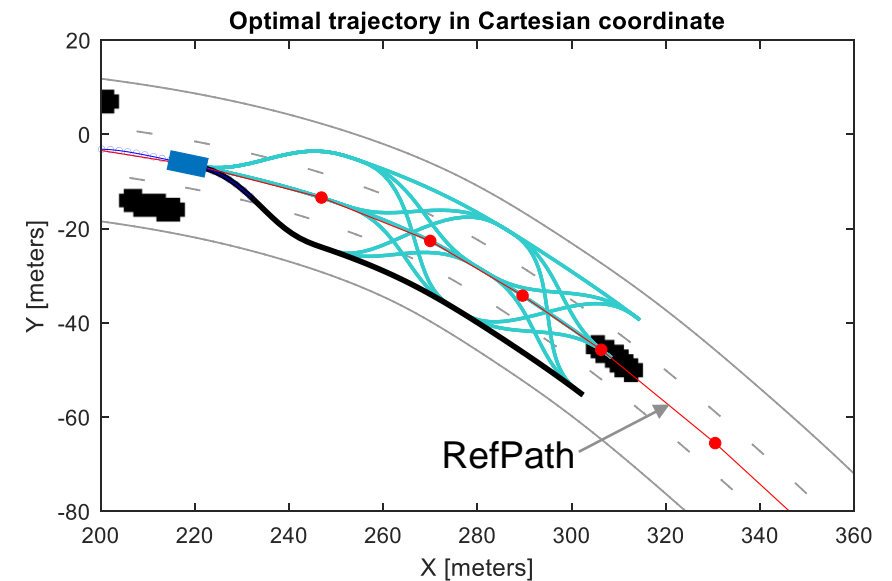
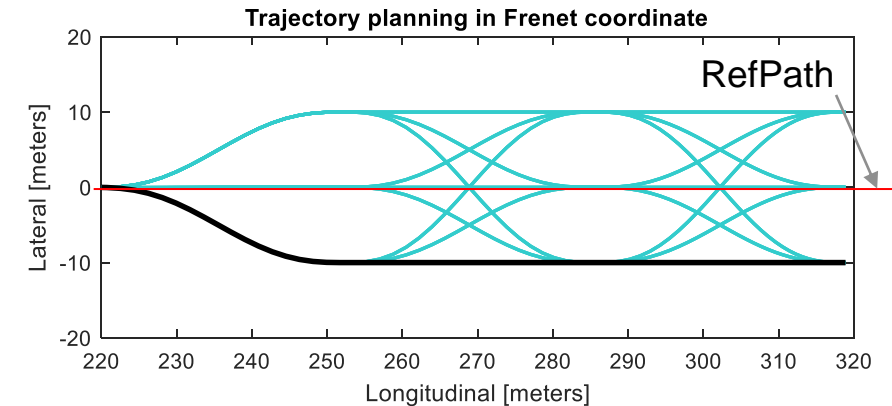
- $s$  : arc length (longitudinal distance)
- $l$  : normal distance (lateral distance) from reference path

frenet2cart

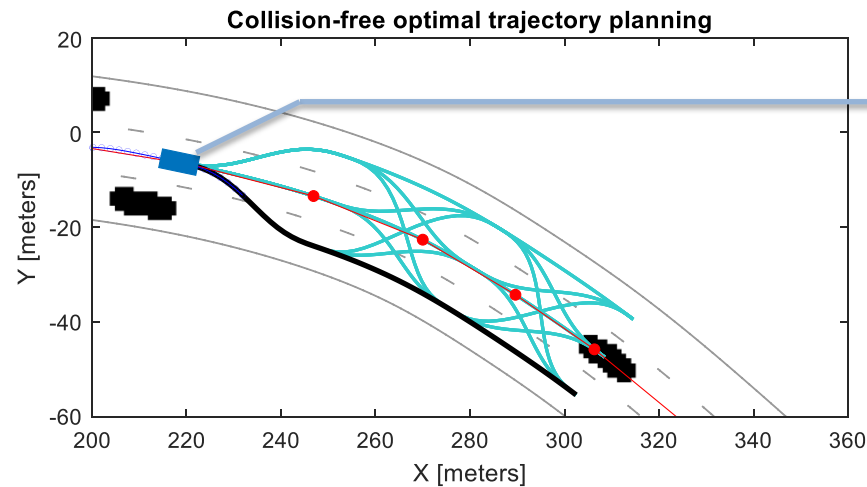
cart2frenet



- $x, y$  : position in meters
- $\theta$  : orientation angle in radians
- $\kappa$  : curvature in  $m^{-1}$
- $v$  : velocity in  $m/s$
- $a$  : acceleration in  $m/s^2$



# Convert Frenet states $\leftrightarrow$ Cartesian states



Current Cartesian state  
 $[x, y, \theta, \kappa, v, a]$

cart2frenet

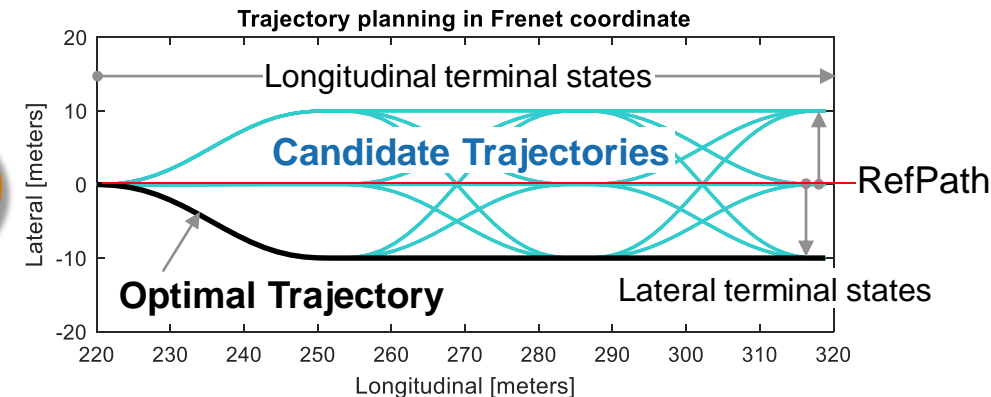
Start Frenet state  
 $\left[ s, \frac{ds}{dt}, \frac{d^2s}{dt^2}, l, \frac{dl}{ds}, \frac{d^2l}{ds^2} \right]$

Optimal trajectory  
 in Cartesian state  
 $[x, y, \theta, \kappa, v, a, t]$

`trajectory = plan(planner, startFrenetState)`

frenet2cart

Trajectory generation  
 in Frenet states



# Optimal Trajectory Planning in Frenet Space

## - shipping example

```
% Step 1) Create and Assign Map to State Validator
```

```
% Step 2) Initialize the planner object and set planner parameters
```

```
% Step 3) Trajectory Planning
```

```
% Step 4) Trajectory Visualization
```

```
% Visualize the map and the trajectories.
```

```
show(map)
```

```
hold on
```

```
show(planner, 'Trajectory', 'all')
```

```
% and the custom cost function.
```

```
planner =
```

```
trajectoryOptimalFrenet(refPath, stateValida
```

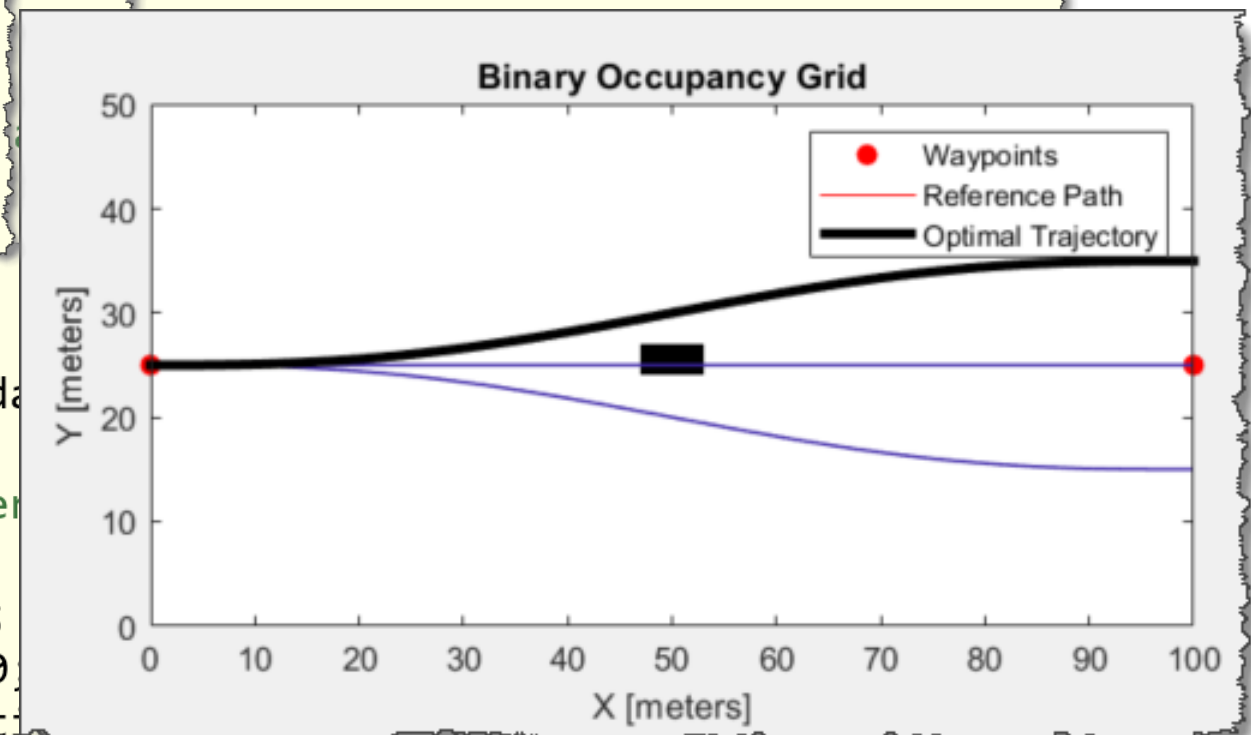
```
% Assign longitudinal terminal state, later  
% values.
```

```
planner.TerminalStates.Longitudinal = 100;
```

```
planner.TerminalStates.Lateral = -10:10:10
```

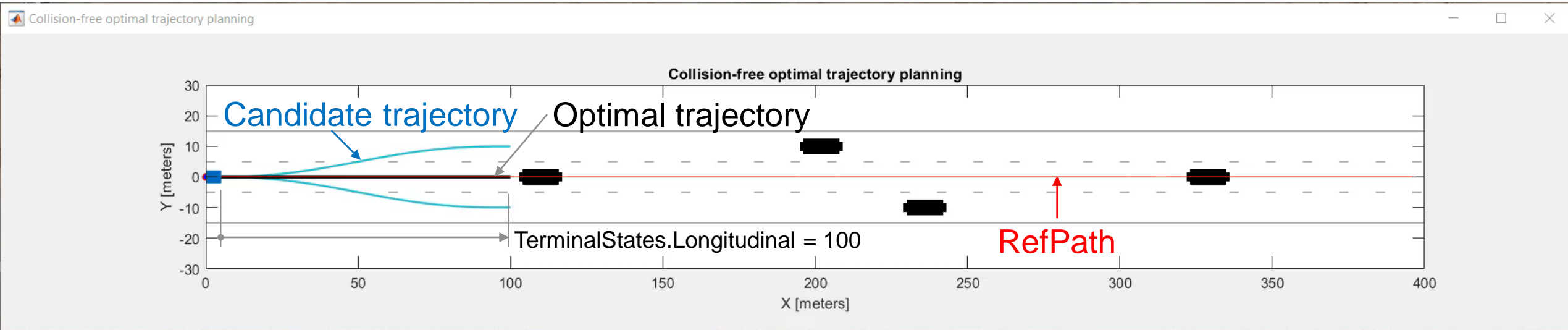
```
planner.FeasibilityParameters.MaxAccelerat
```

llow.



# Optimal Trajectory Planning with binary occupancy map : Straight road

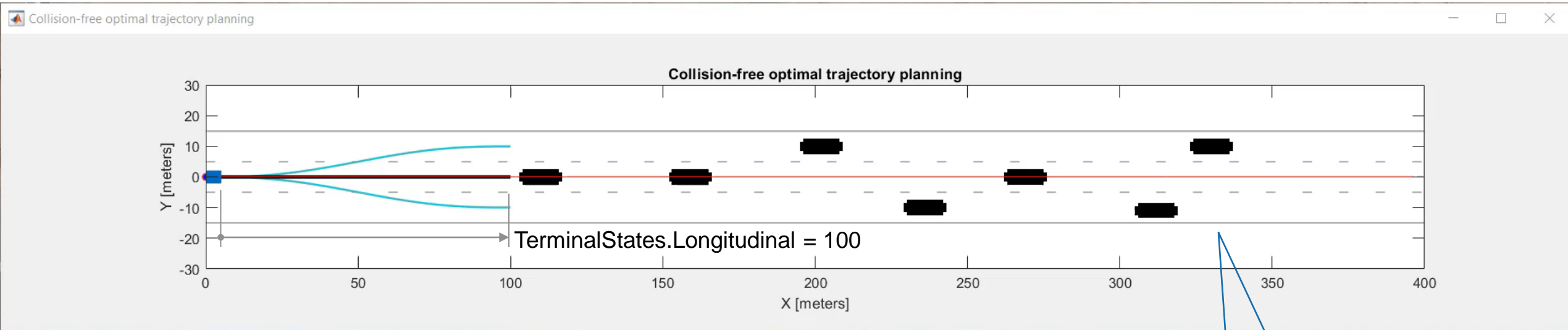
- TerminalStates.Longitudinal = 100
- NumSegments = 1



# Optimal Trajectory Planning with binary occupancy map

## : Straight road with dense traffic condition

- TerminalStates.Longitudinal = 100
- NumSegments = 1

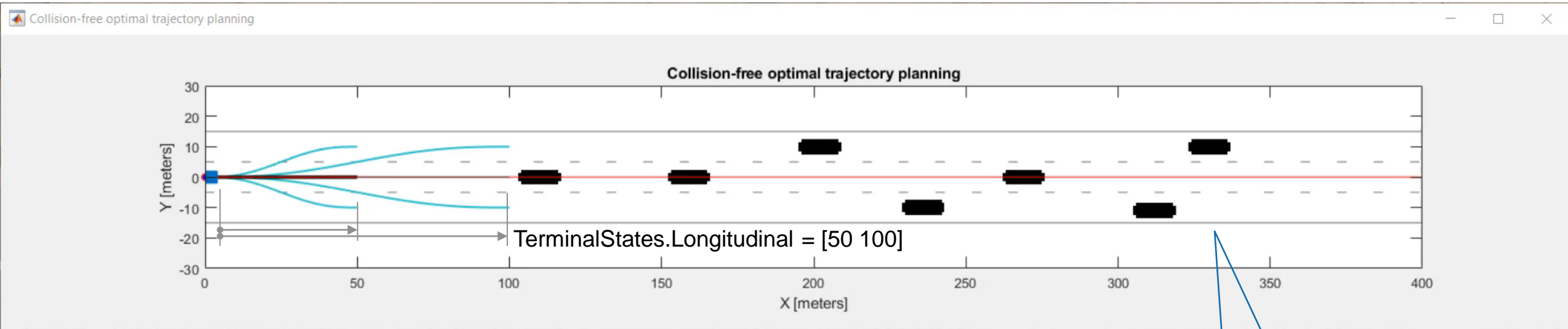


Dense traffic condition

# Optimal Trajectory Planning with binary occupancy map

## : Straight road with dense traffic condition

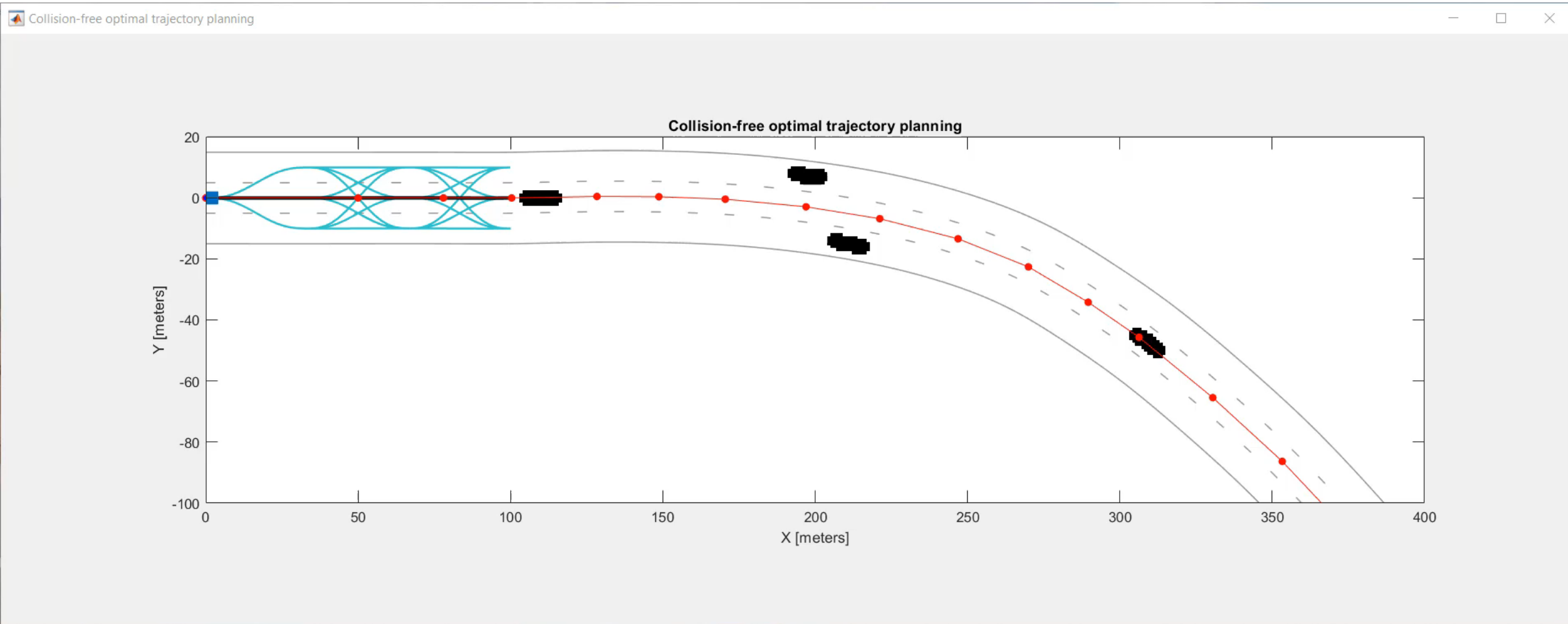
- TerminalStates.Longitudinal = [50 100]
- NumSegments = 1



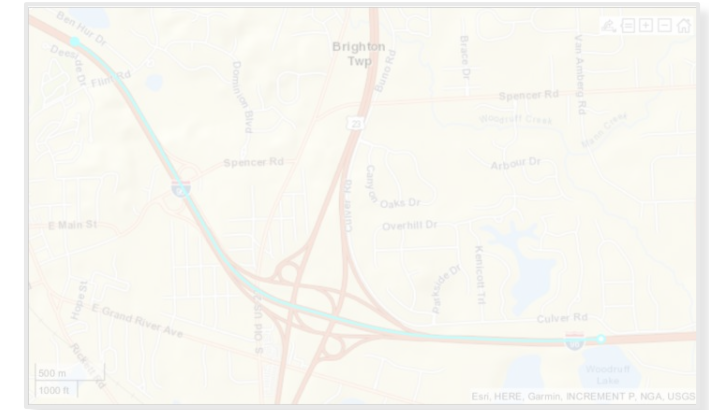
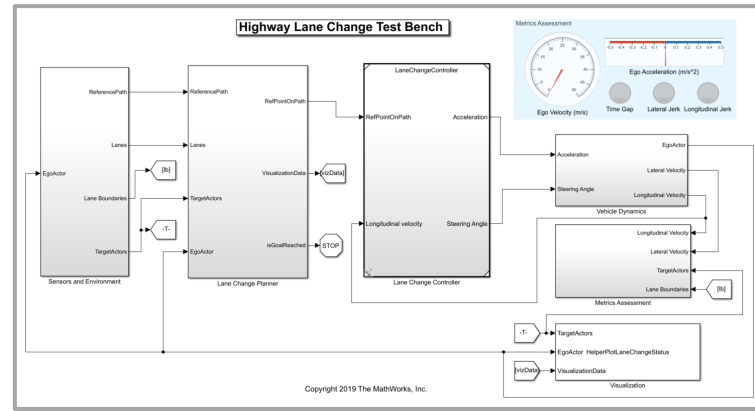
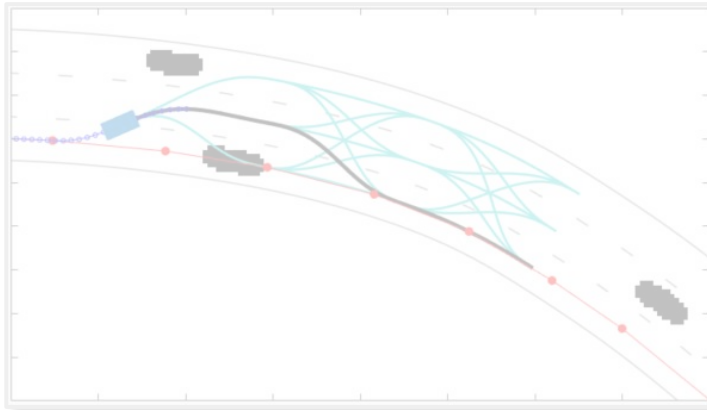


# Optimal Trajectory Planning with binary occupancy map : Curved road

- TerminalStates.Longitudinal = 100
- NumSegments = 3



# Highway Lane Change



## Learn motion planner

- Optimal trajectory generation in Frenet coordinate
- Planner parameters
- Simulate trajectory planning with occupancy map

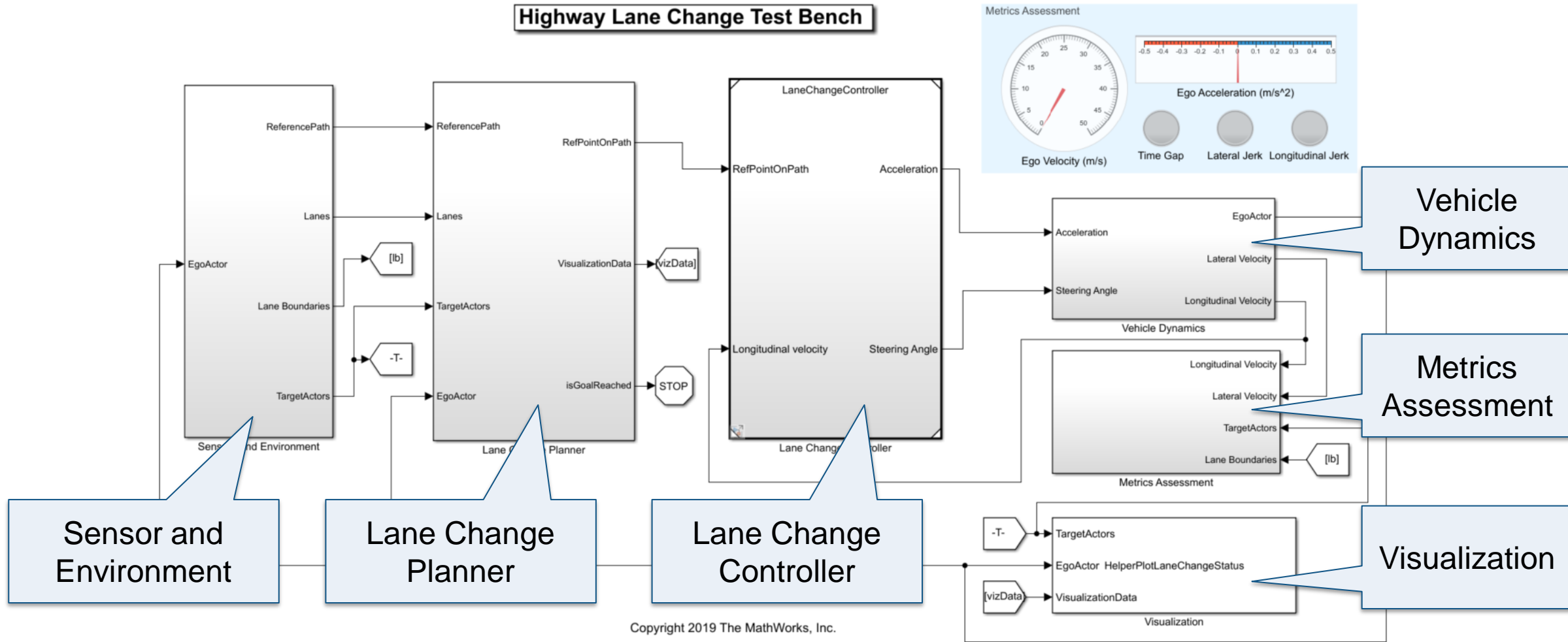
## Integrate motion planner and controller

- Learn through reference example
- Architecture of highway lane change
- Simulate closed-loop controller with test scenarios

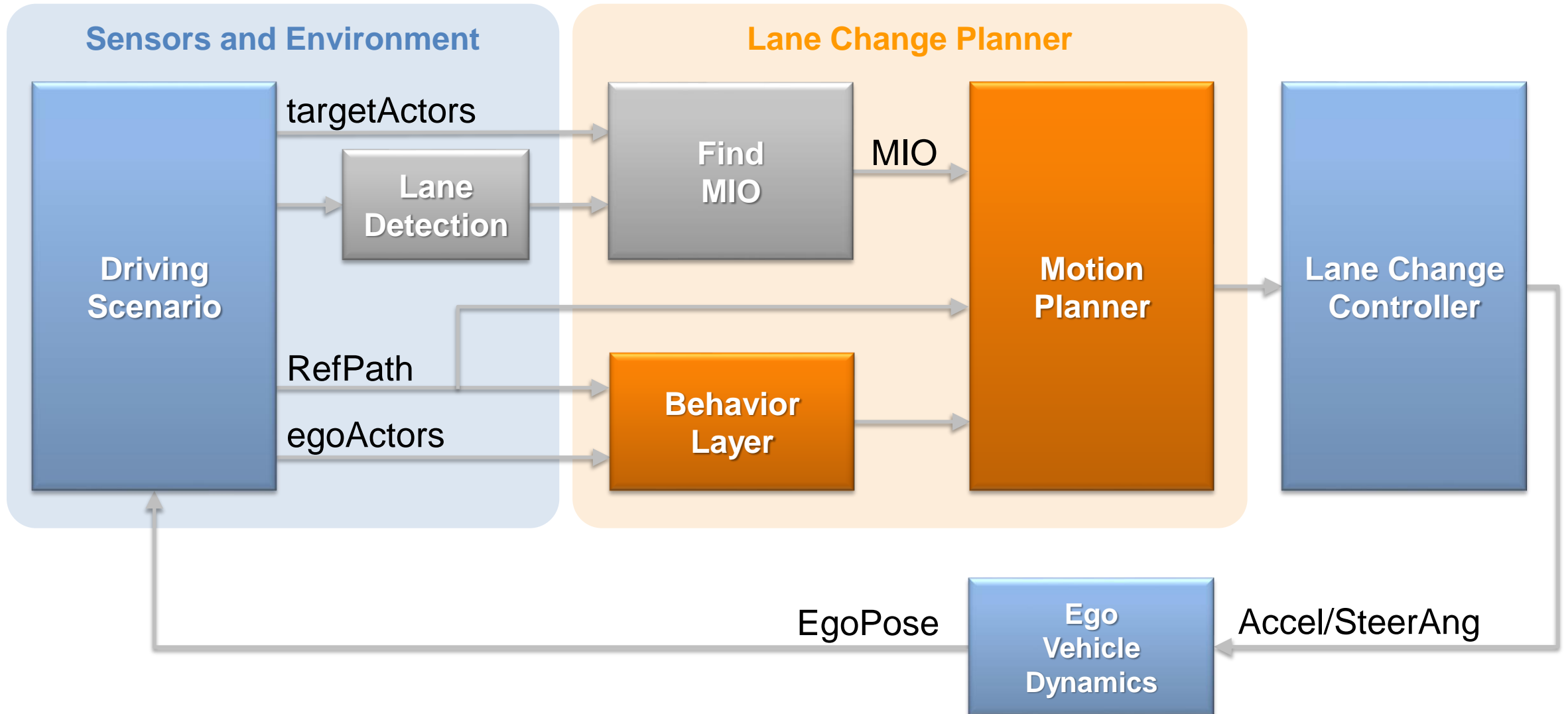
## Test with real-world scenario

- Create scenario from HD map
- Update setup script and model
- Run simulation with new scenario

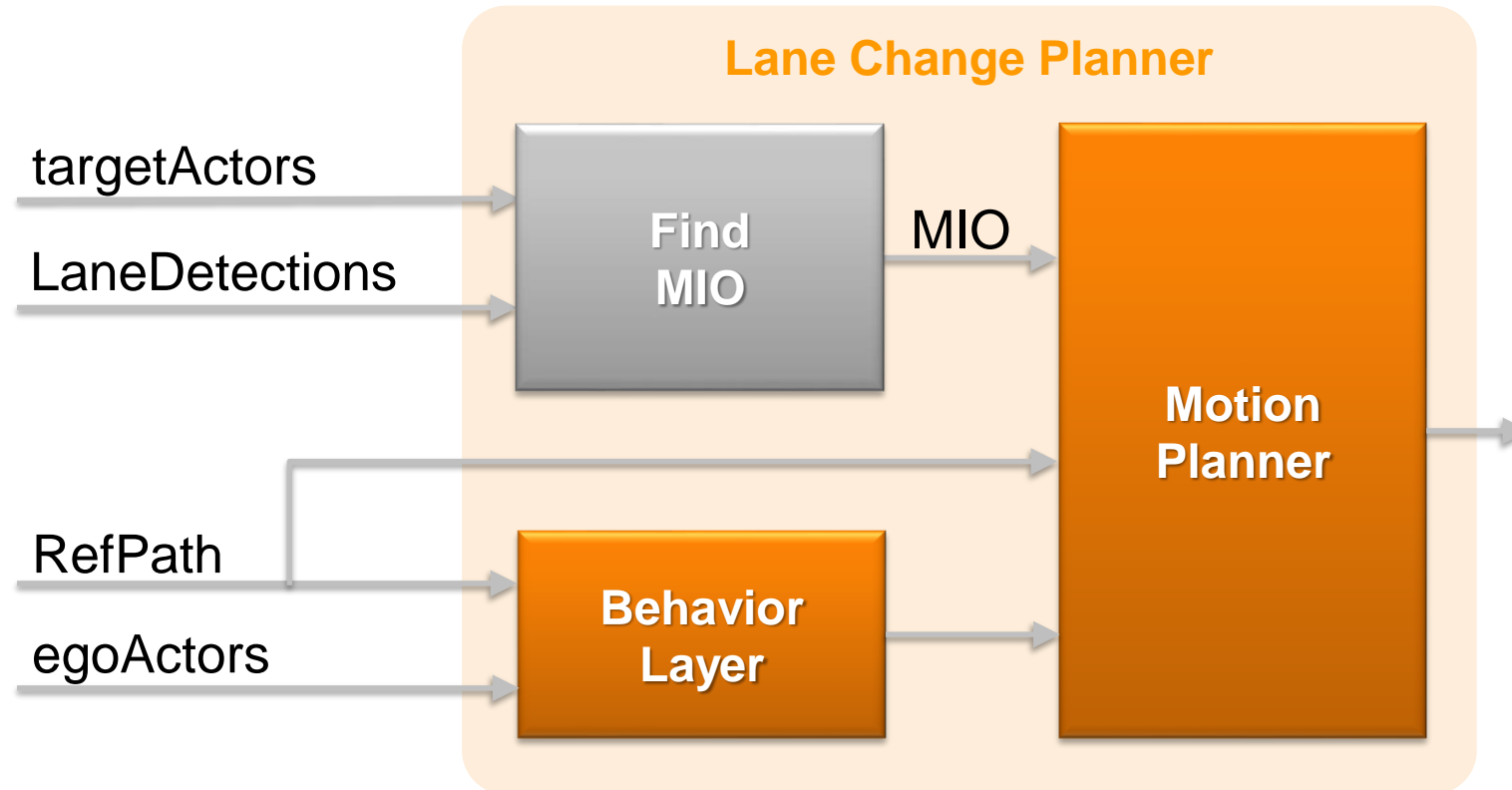
# Test Bench Model for Highway Lane Change



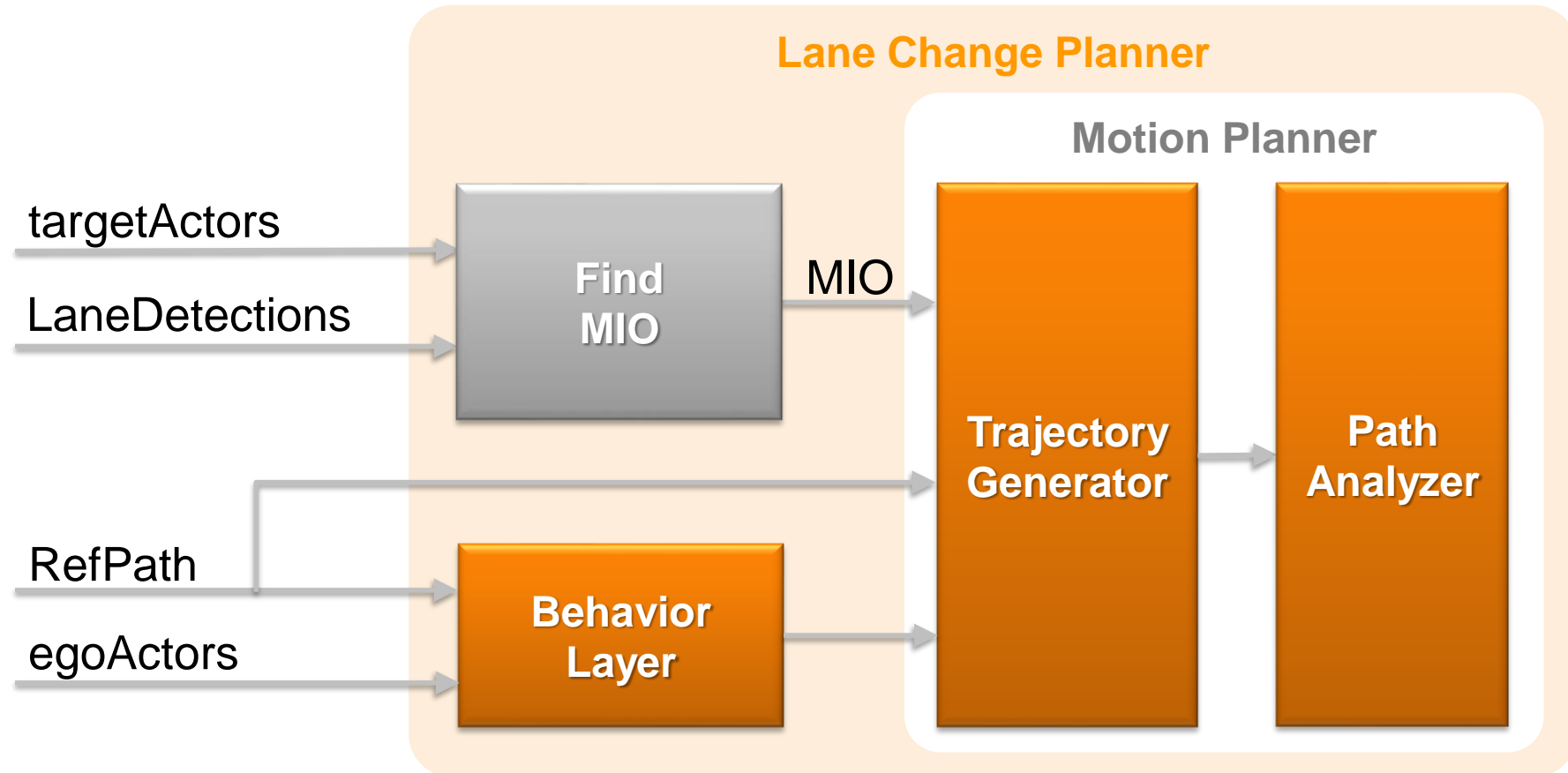
# Architecture of Highway Lane Change



# Lane Change Planner

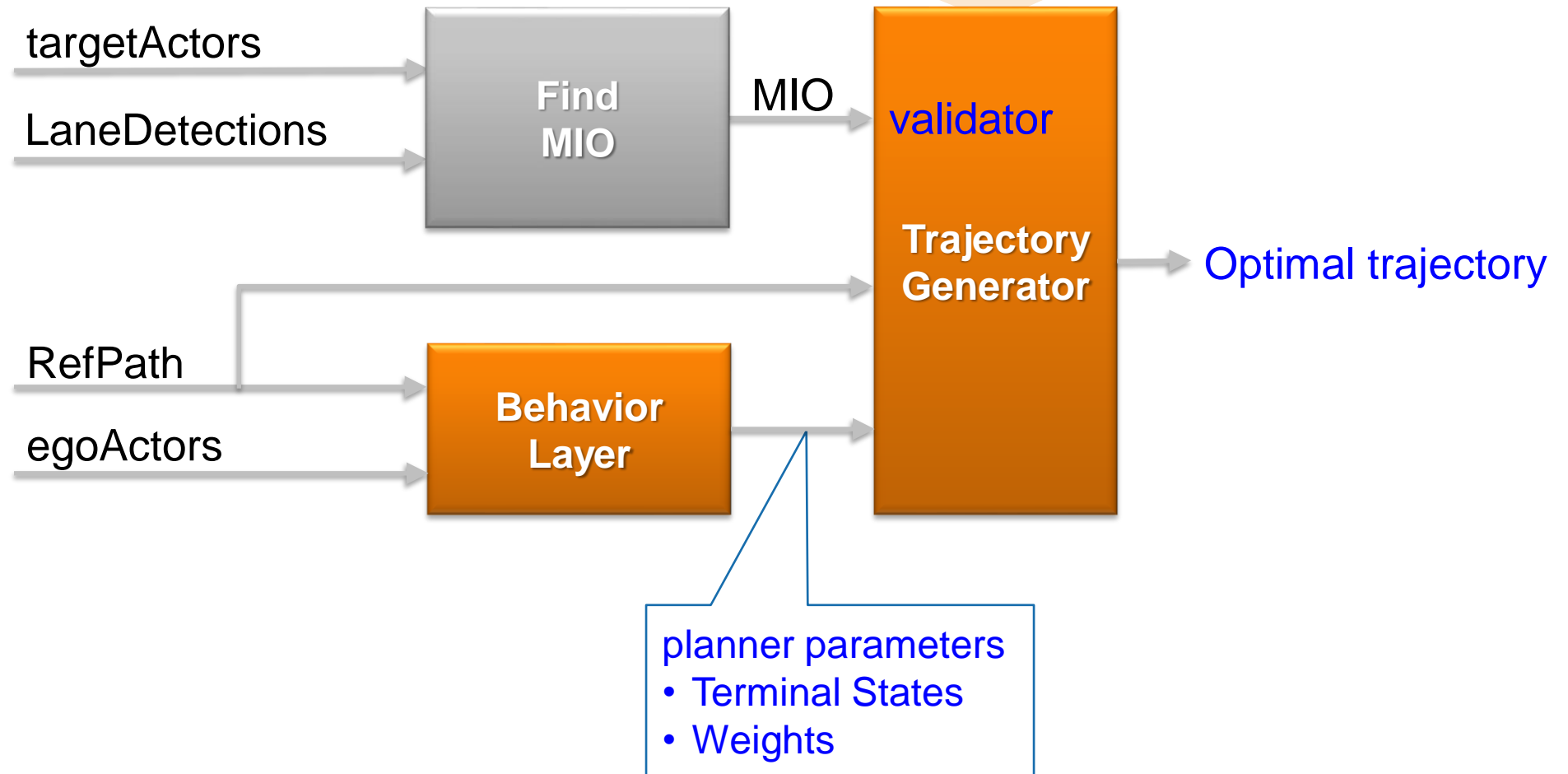


# Lane Change Planner

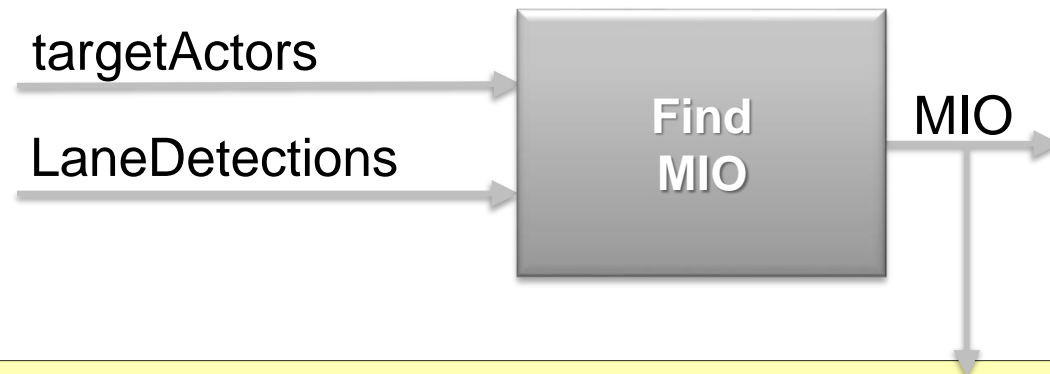


# Lane Change Planner

```
planner = trajectoryOptimalFrenet(refPath, validator)
```

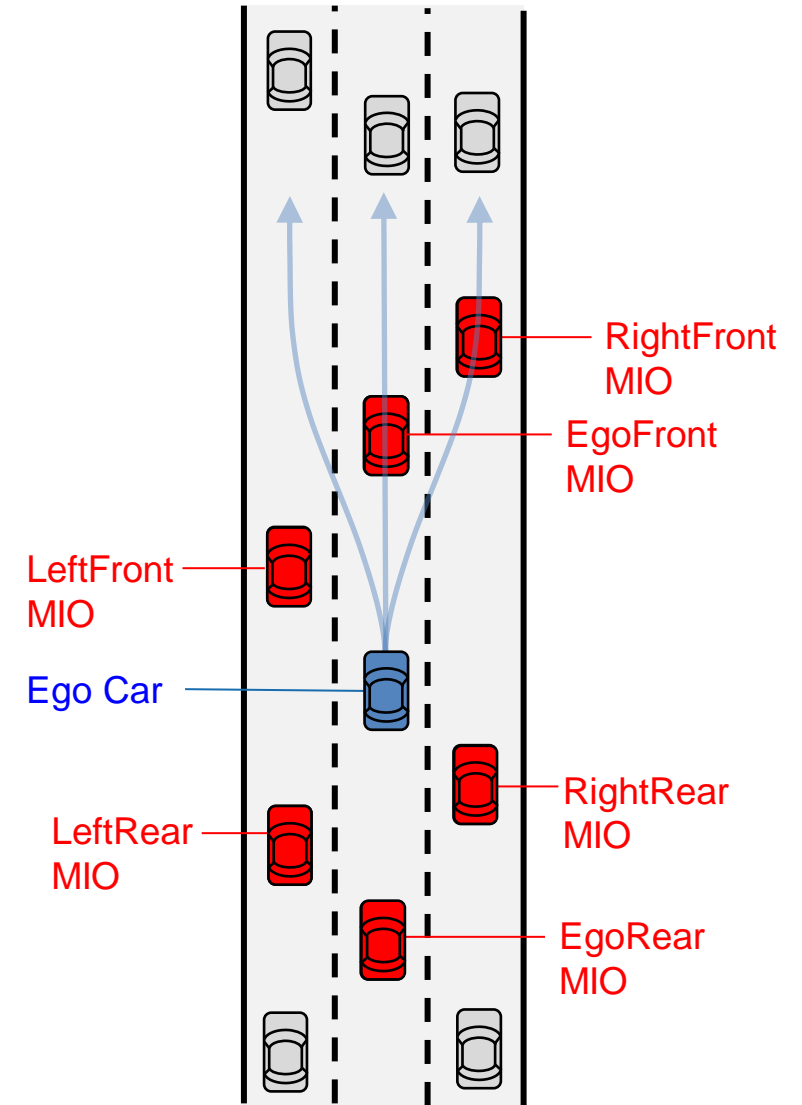


# Find Most Important Objects (MIO)



```
planner = trajectoryOptimalFrenet(refPath, validator)
```

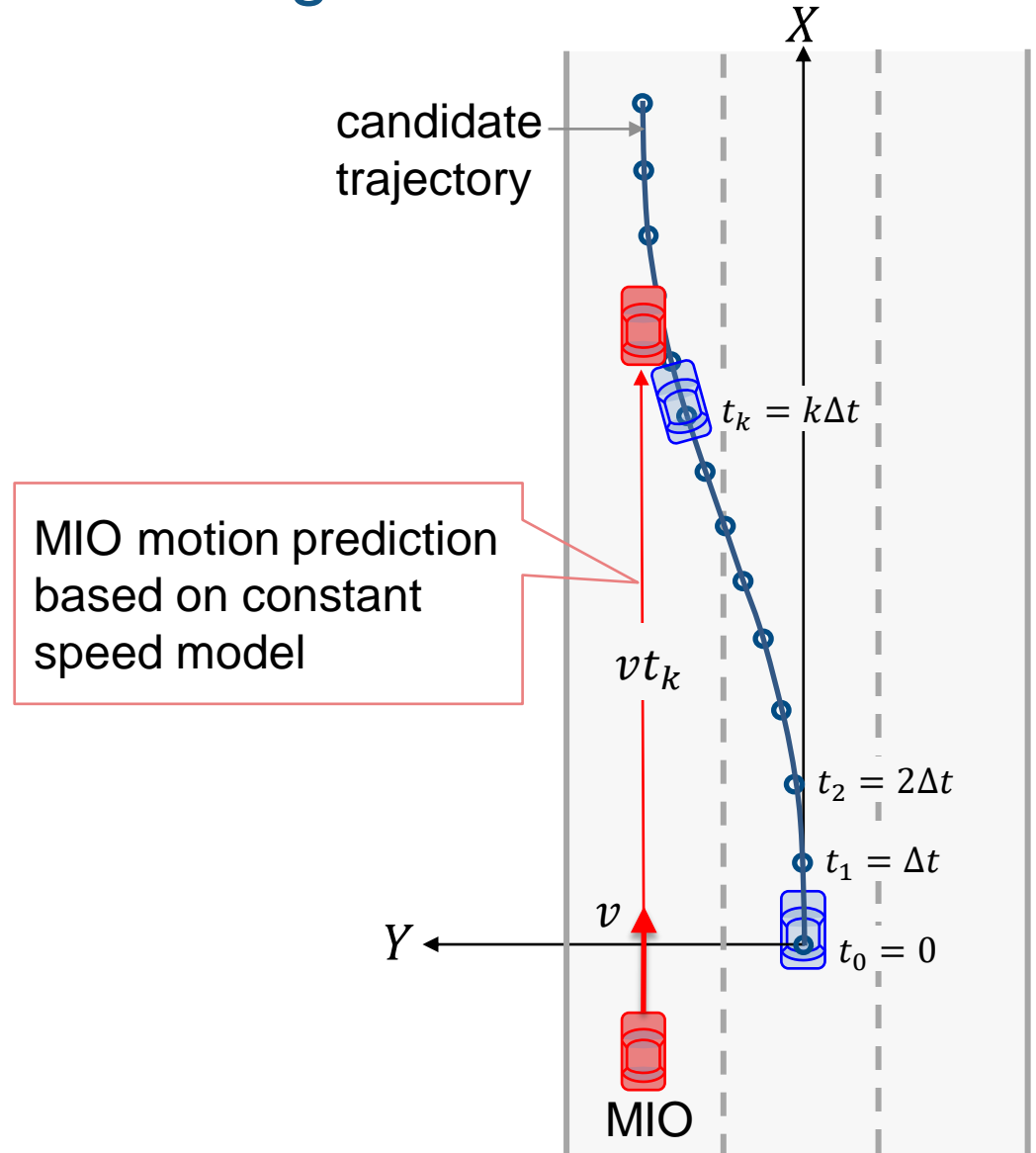
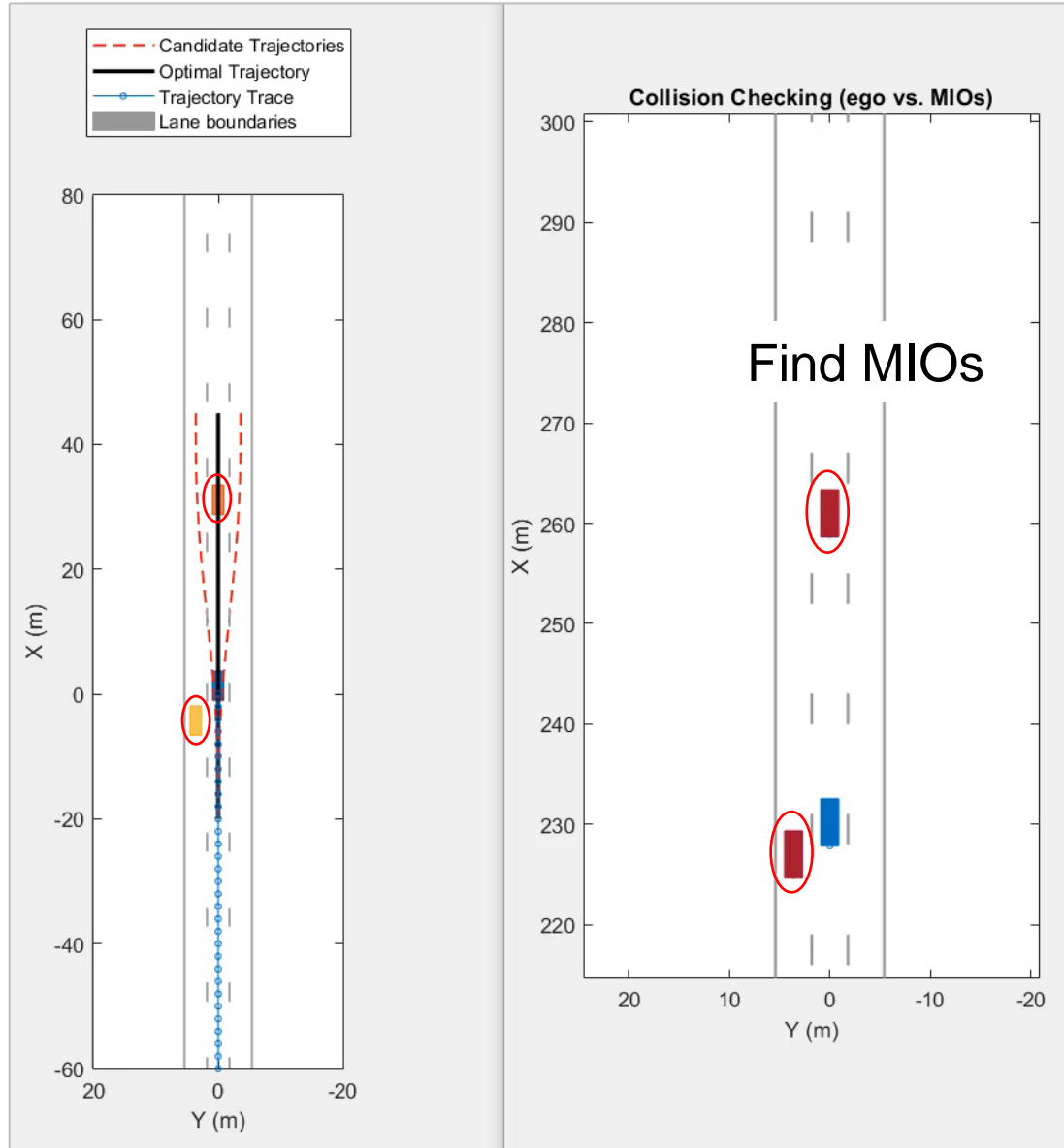
- Find MIOs in ego and neighbor lanes
- MIOs are provided for the custom state validator for collision checking.



Max. 6 MIOs



# Custom State Validator for Collision checking



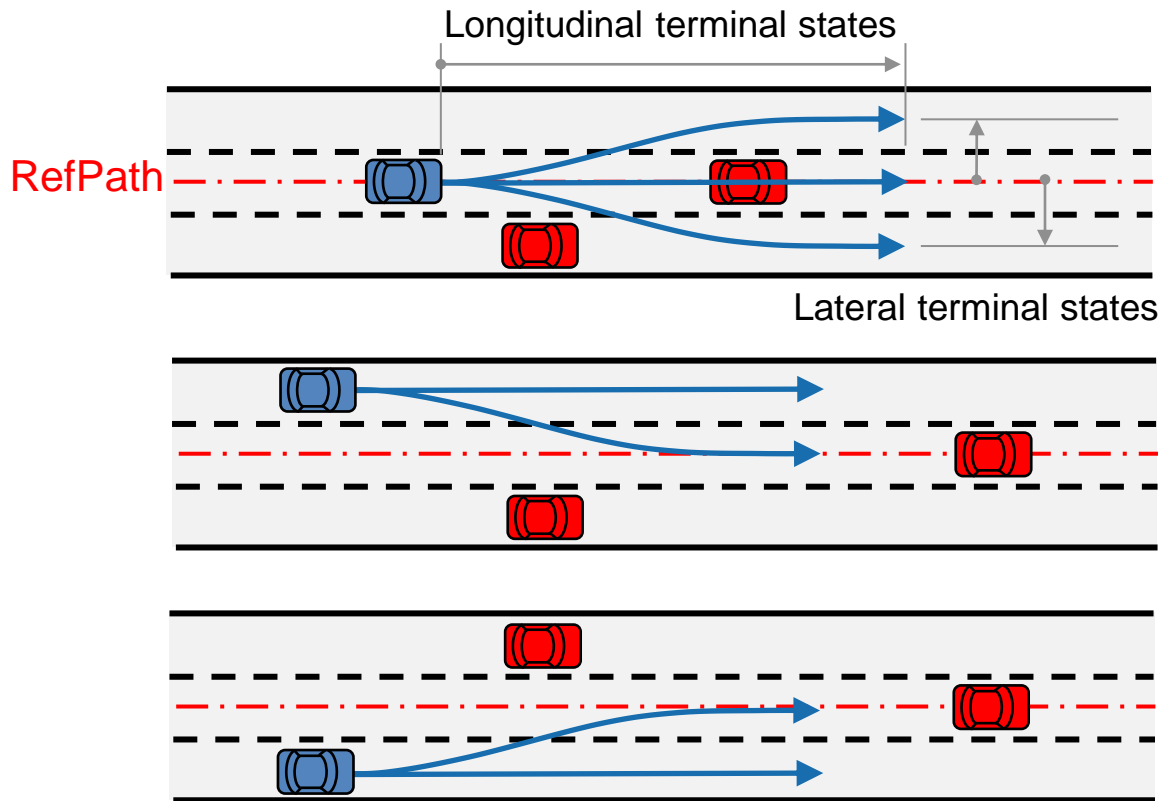
# Behavior Layer

RefPath

egoActors



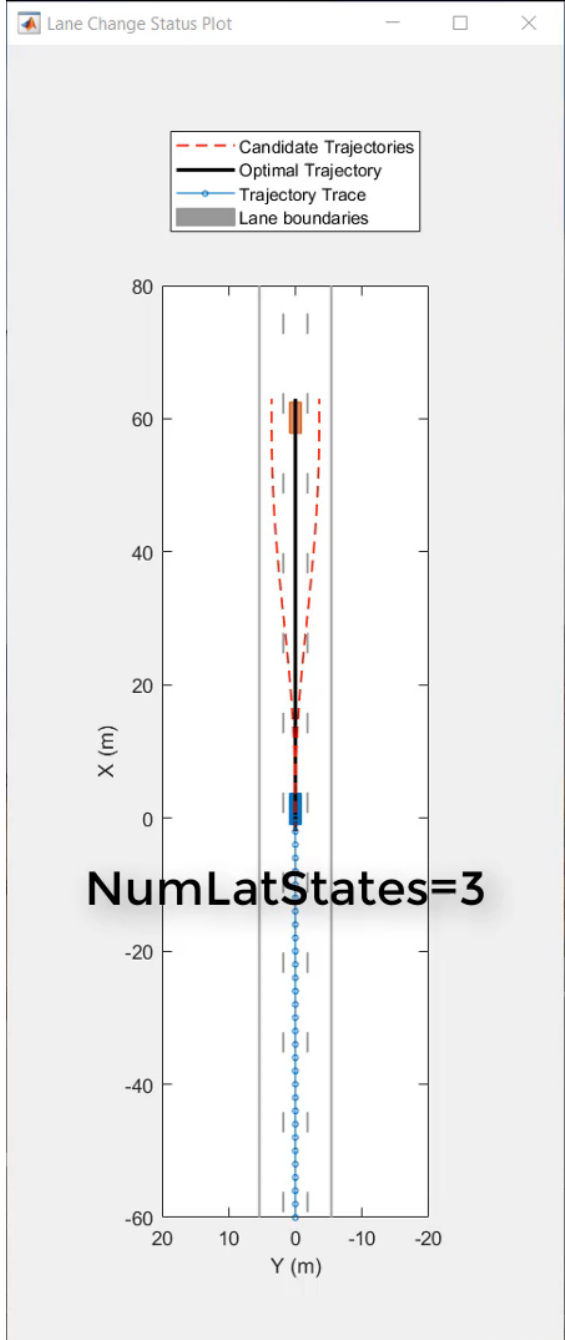
planner parameters  
 • Terminal States  
 • Weights



NumLatStates = 3

NumLatStates = 2  
(No Left Lane)

NumLatStates = 2  
(No Right Lane)



# Motion Planner ▷ Trajectory Generator

```
planner = trajectoryOptimalFrenet(refPath, validator)
```

MIO

RefPath

Planner Parameters

- Terminal States
- Weights

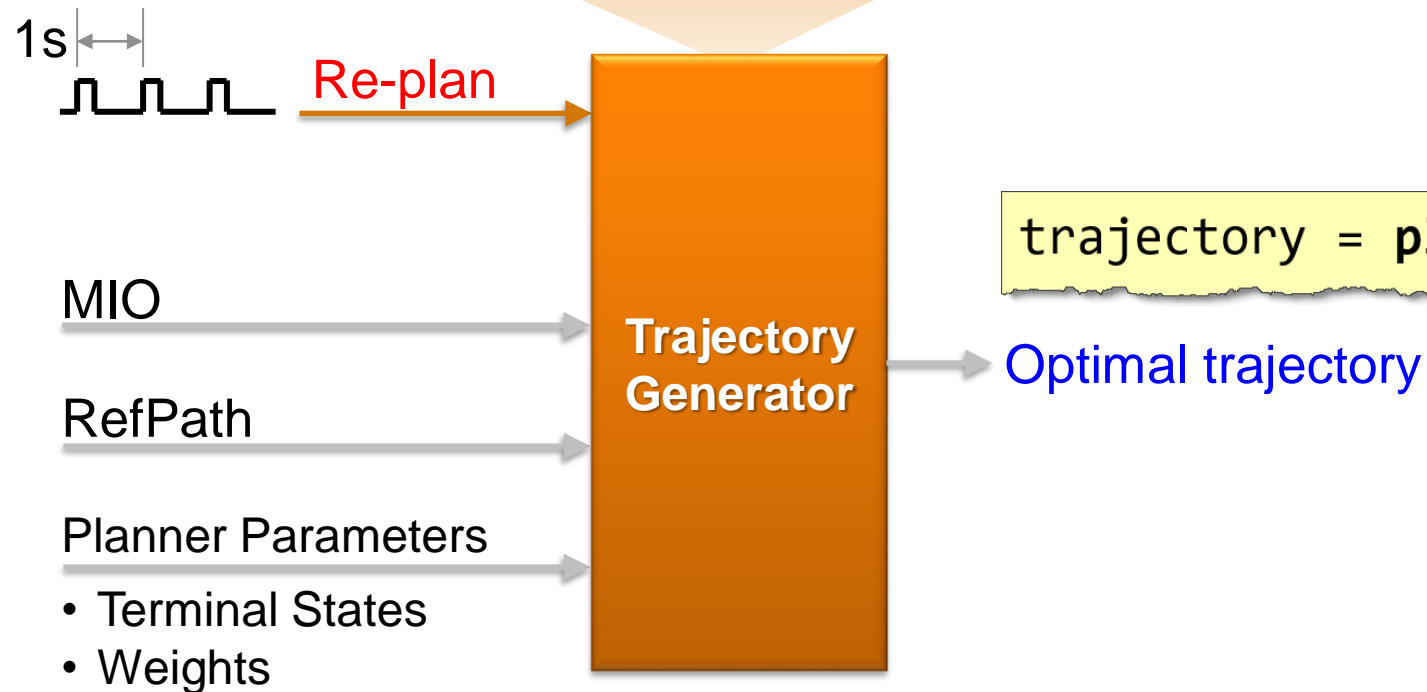
Trajectory  
Generator

```
trajectory = plan(planner, startFrenetState)
```

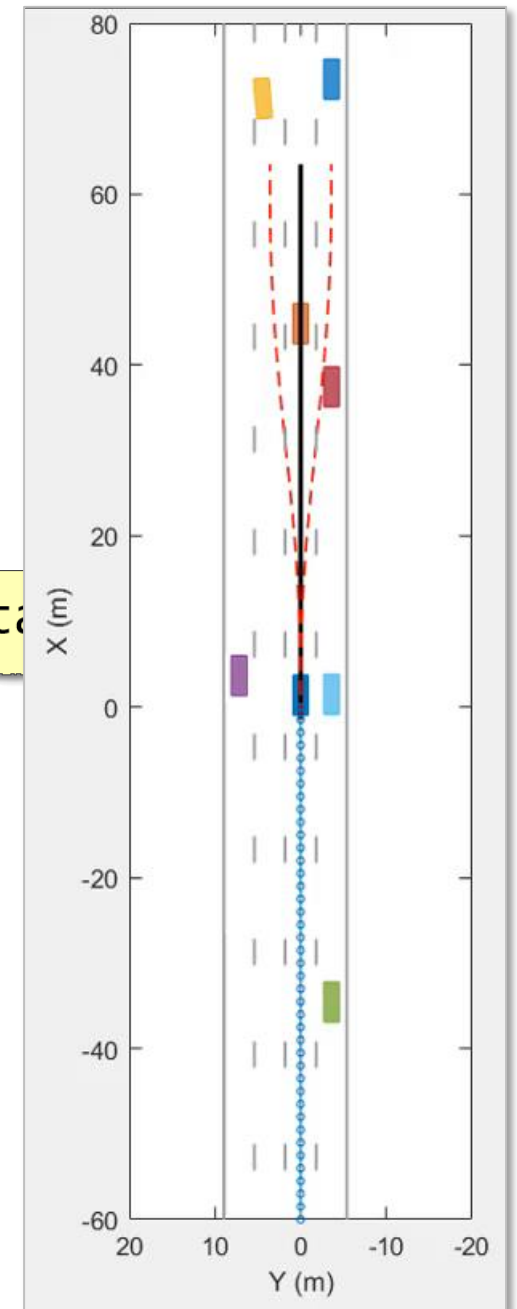
Optimal trajectory

# Motion Planner ▷ Periodic Re-planning

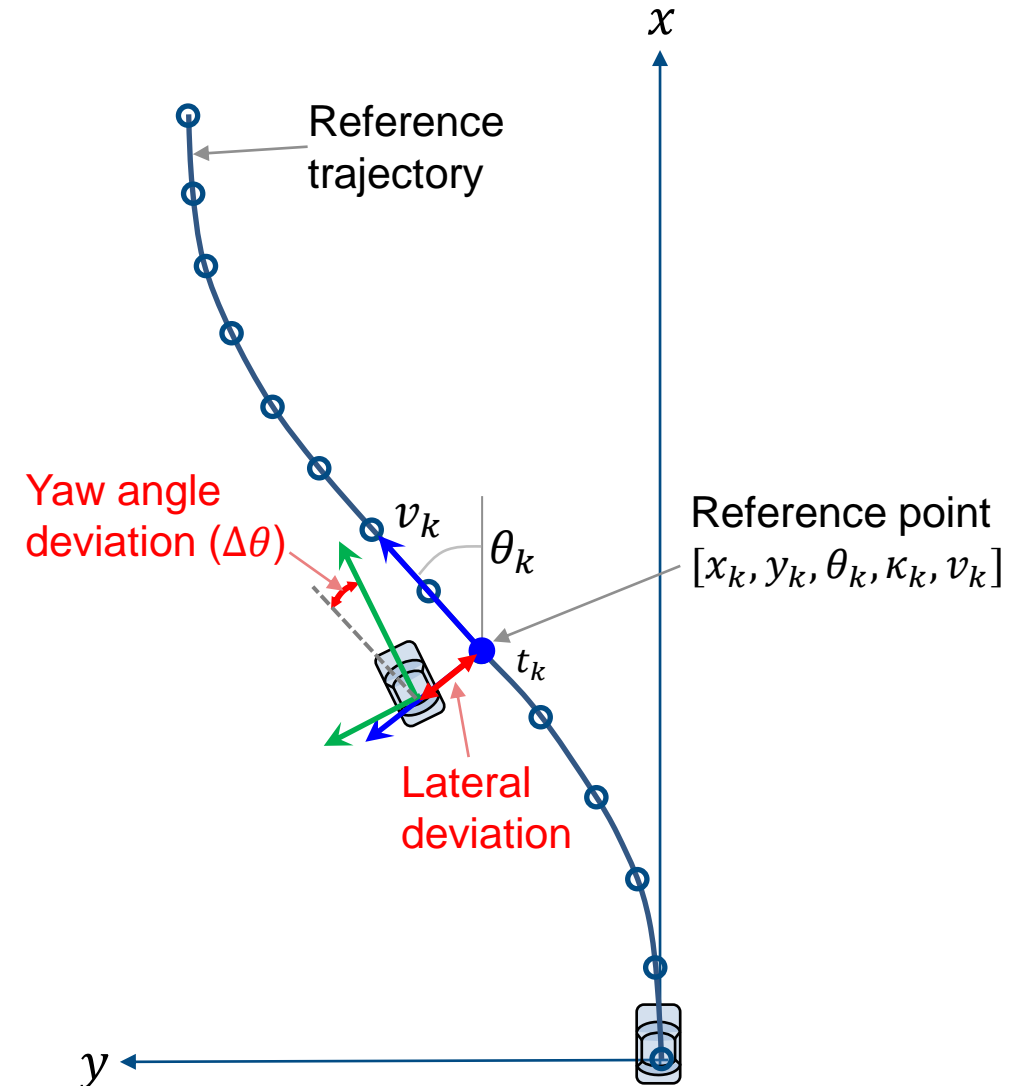
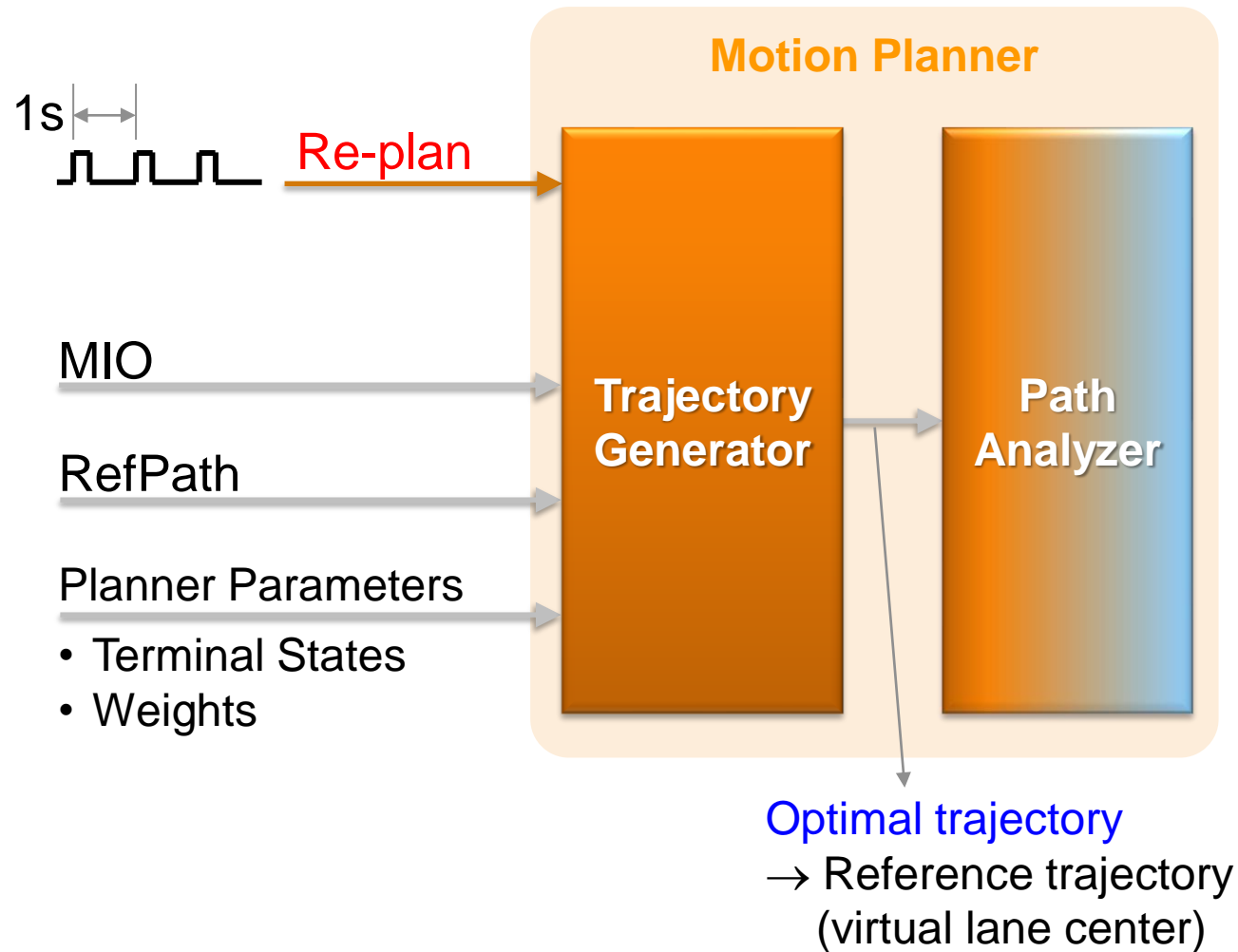
```
planner = trajectoryOptimalFrenet(refPath, validator)
```



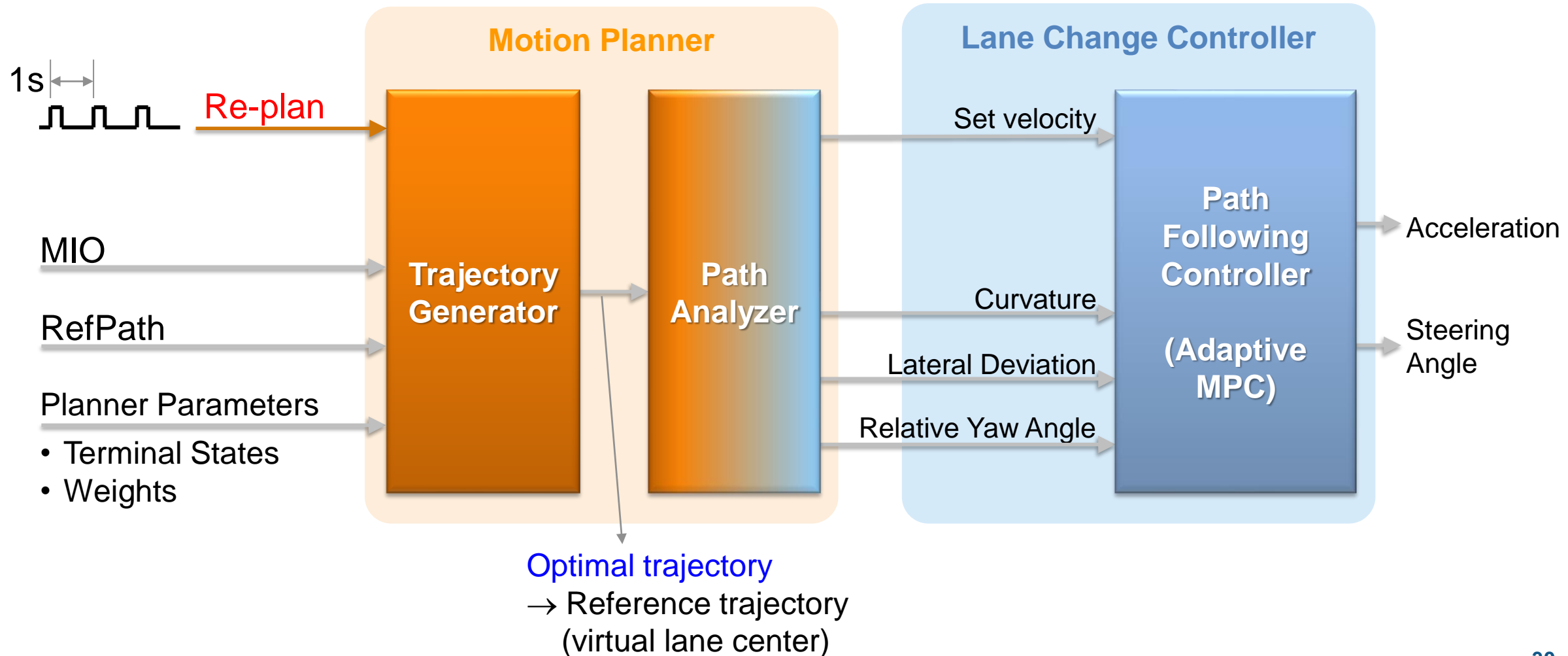
```
trajectory = plan(planner, start, end)
```



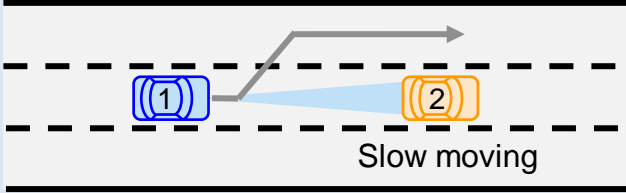
# Motion Planner ▷ Path Analyzer



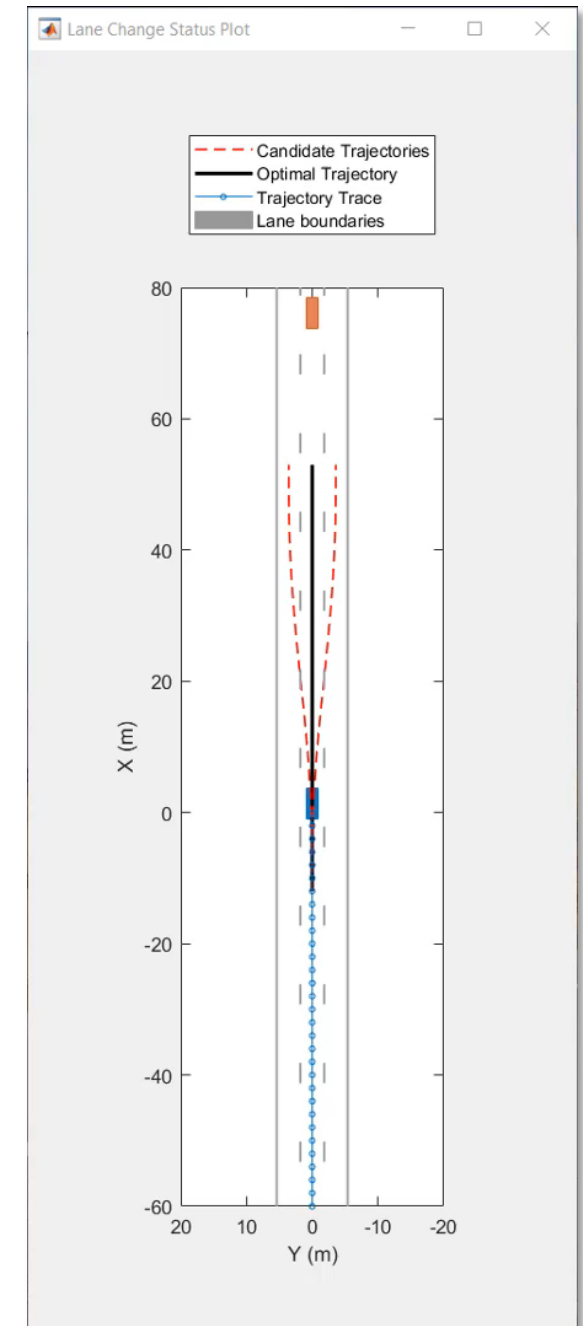
# Motion Planner and Lane Change Controller



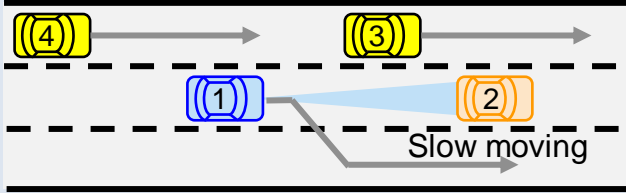
# Simulate Highway Lane Change with Test Scenarios

Test Name	scenario_LC_01_SlowMoving
Test Description	Passing slow moving lead car 
(1) Host Car	initial velocity = 20m/s HWT = 6.5sec HW = 130m v_set = 20m/s
(2) Lead Car	constant velocity = 10m/s
Other Cars	None

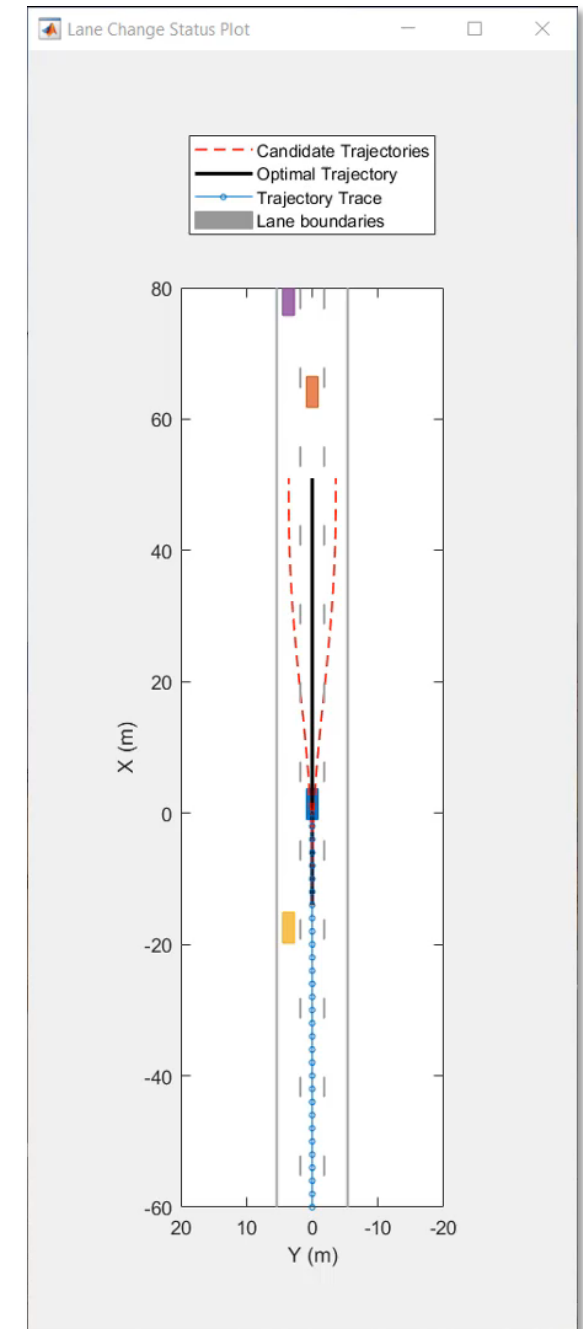
HWT : Headway time  
 HW : Headway  
 v\_set : set velocity for ego car



# Simulate Highway Lane Change with Test Scenarios

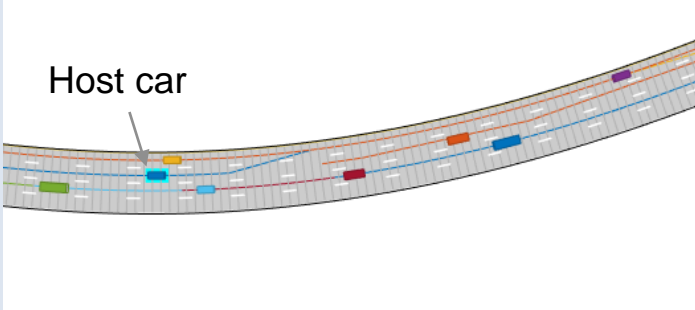
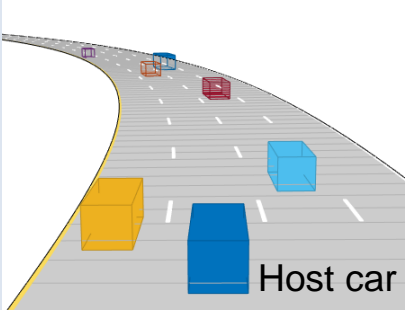
Test Name	scenario_LC_07_RightLaneChange
Test Description	Passing slow moving lead car to right lane 
(1) Host Car	initial velocity = 20m/s HWT = 6sec HW = 120m v_set = 20m/s
(2) Lead Car	constant velocity = 10m/s
Other Cars	Constant velocity = 25m/s (3 <sup>rd</sup> car in left lane) Constant velocity = 24m/s (4 <sup>th</sup> car in left lane)

HWT : Headway time  
 HW : Headway  
 v\_set : set velocity for ego car

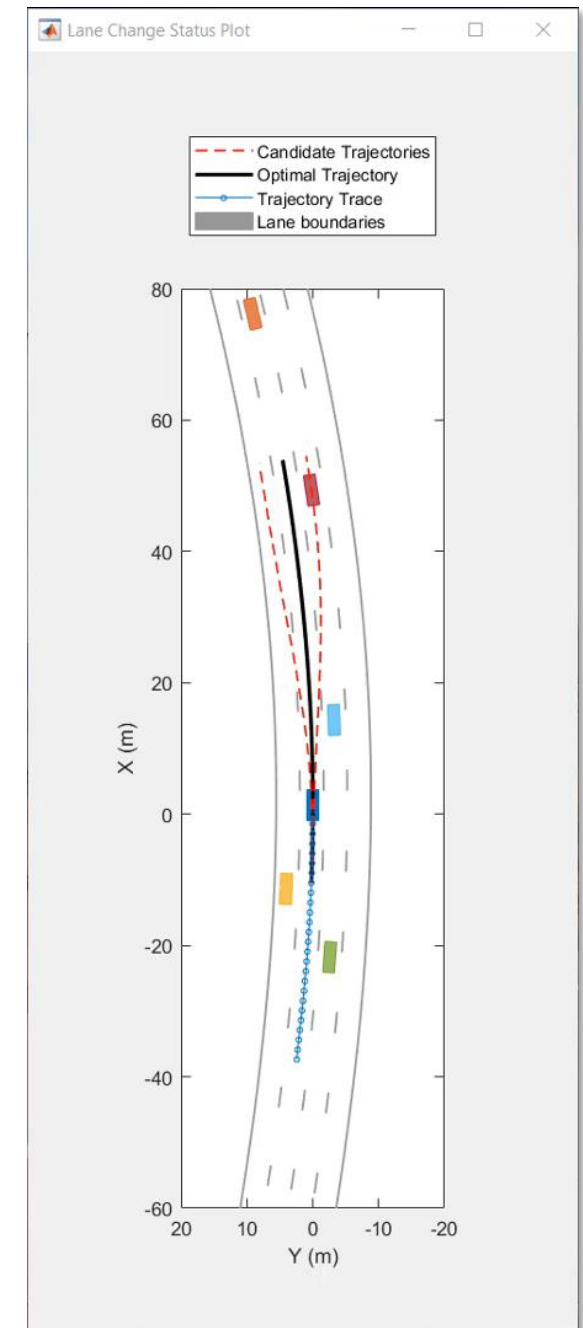




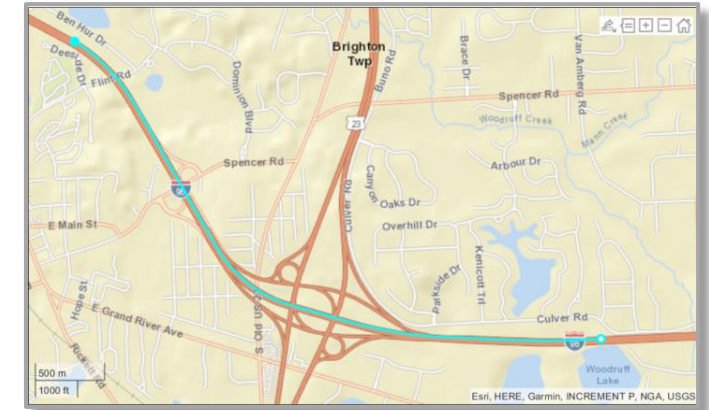
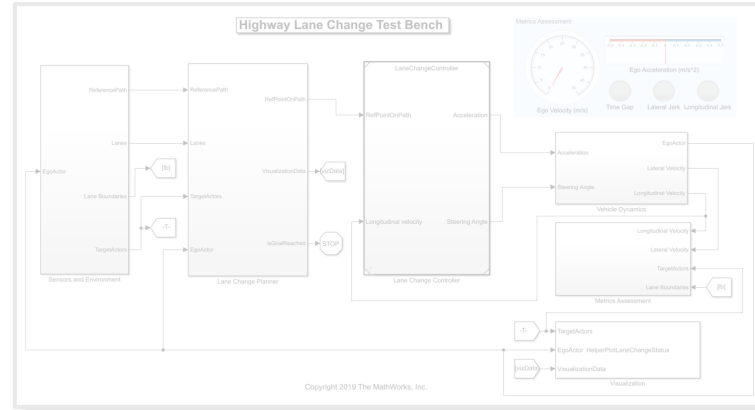
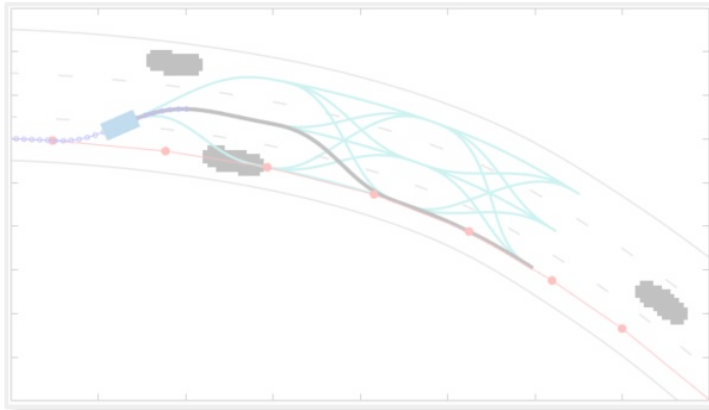
# Simulate Highway Lane Change with Test Scenarios

Test Name	scenario_LC_10_SingleLaneChange_Curved
Test Description	Passing slow moving lead car to left lane in curved road <div style="display: flex; justify-content: space-around; margin-top: 10px;">   </div>
(1) Host Car	initial velocity = 15m/s $v\_set = 15\text{m/s}$
(2) Lead Car	Slow moving
Other Cars	Dense traffic

HWT : Headway time  
 HW : Headway  
 $v\_set$  : set velocity for ego car



# Highway Lane Change



## Learn motion planner

- Optimal trajectory generation in Frenet coordinate
- Planner parameters
- Simulate trajectory planning with occupancy map

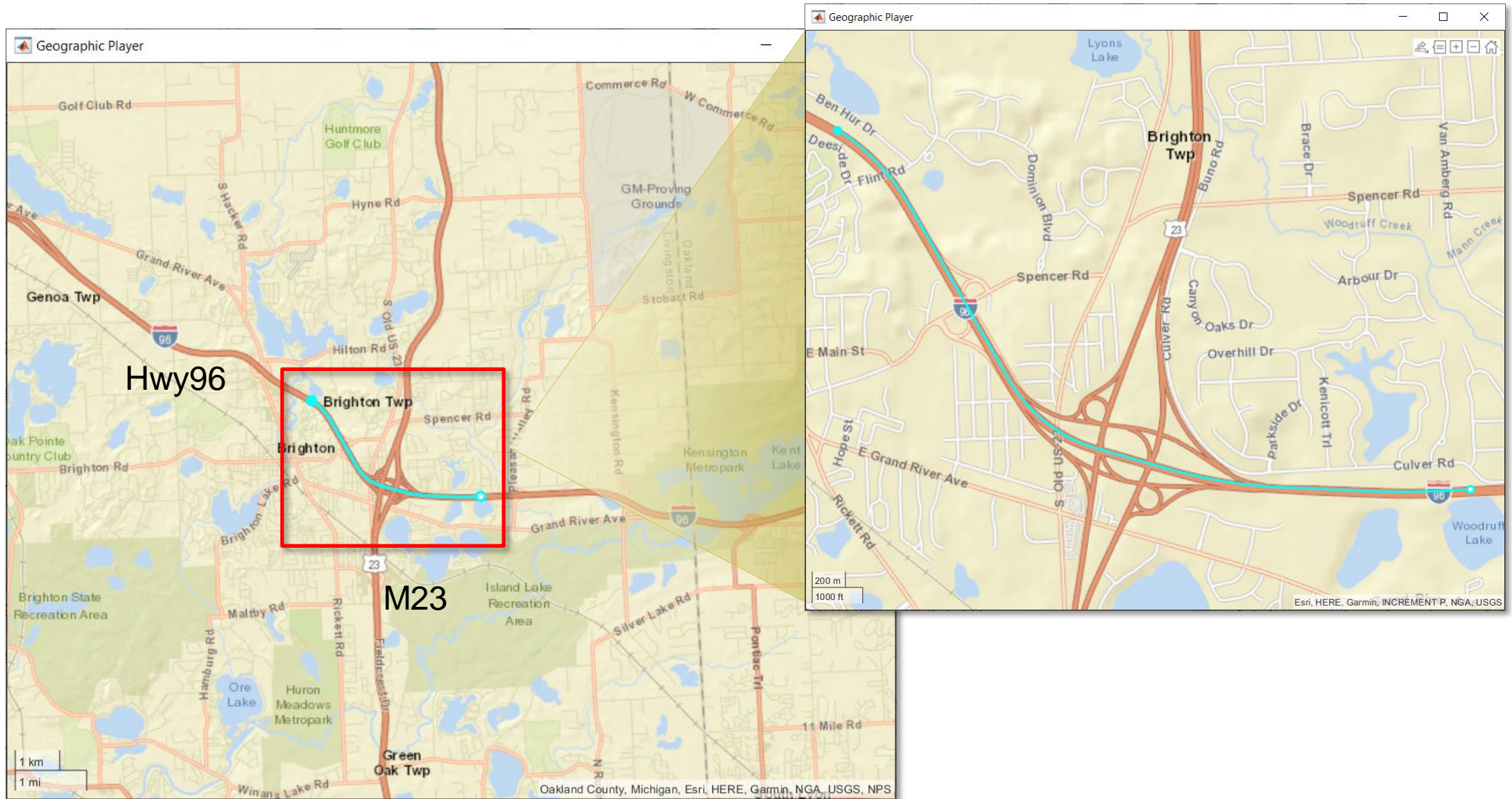
## Integrate motion planner and controller

- Learn through reference example
- Architecture of highway lane change
- Simulate closed-loop controller with test scenarios

## Test with real-world scenario

- Create scenario from HD map
- Update setup script and model
- Run simulation with new scenario

# Create highway road network from HD map



# Import HERE HD Live Map into Driving Scenario

**DESIGNER**

New Open Save Import Add Road Add Actor Add Camera Add Radar

**FILE**

Roads Acto

Road:

Name:

Width (m):

Bank Angle (deg):

Lanes

Road Centers

**OpenDRIVE Road Network**  
Import roads from .xml or .xodr OpenDRIVE files

**HERE HD Live Map**  
Import roads generated from HERE HD Live Map data (credentials required)

**HERE HD Live Map Import**

**Specify Geographic Coordinates**

From Workspace  Input Coordinates

Latitude Latitude Longitude Longitude

**Select Region**

Specify a region around the roads of interest.

1 km  
0.5 mi

Cancel Back

**HERE HD Live Map Import**

**Specify Geographic Coordinates**

From Workspace  Input Coordinates

Latitude Latitude Longitude Longitude

**Select Roads**

Select a road by clicking on it. Select multiple roads with controls in the toolbar.

500 m  
2000 ft

Select All Deselect All Select Nearest Roads

Cancel Back Import

# Add actors and export driving scenario with MATLAB Function

The screenshot shows the Driving Scenario Designer interface. The 'Export MATLAB Function' button is highlighted with a red box. The button text reads: "Export MATLAB Function" and "Generate MATLAB function for your driving scenario and sensors".

The MATLAB code generated by the software is as follows:

```
function [scenario, egoVehicle] = createDrivingScenario()
% createDrivingScenario Returns the drivingScenario defined in
% Generated by MATLAB(R) 9.8 (R2020a) and Automated Driving Tool
% Generated on: 24-Mar-2020 17:27:39
% Construct a drivingScenario object.
scenario = drivingScenario('StopTime', 19, ...
    'SampleTime', 0.1);
% Add all road segments
roadCenters = [57.03213 -1223.454 0;
    52.57648 -1138.823 0;
    40.3209 -862.7384 0;
    35.85877 -764.1326 0;
    35.85681 -746.0532 0;
    35.85499 -728.8004 0;
    30.29402 -607.1907 0;
```

# Setup driving scenario and workspace variables

```
function helperSLHighwayLaneChangeSetup(varargin)

%% Inputs
% Scenario function name
defaultScenarioFcnName = "scenario_LC_06_DoubleLaneChange";
validScenarioFcnNames = [...
    "scenario_LC_01_SlowMoving"
    "scenario_LC_02_SlowMovingWithPassingCar"
    "scenario_LC_03_DisabledCar"
    "scenario_LC_04_CutInWithBrake"
    "scenario_LC_05_SingleLaneChange"
    "scenario_LC_06_DoubleLaneChange"
    "scenario_LC_07_RightLaneChange"
    "scenario_LC_08_SlowmovingCar_Curved"
    "scenario_LC_09_CutInWithBrake_Curved"
    "scenario_LC_10_SingleLaneChange_Curved"
    "scenario_LC_11_Hwy96_M23_WithMergingCar"
    "scenario_LC_12_Hwy96_M23_WithCutInCar"];
```

Add new driving  
scenario

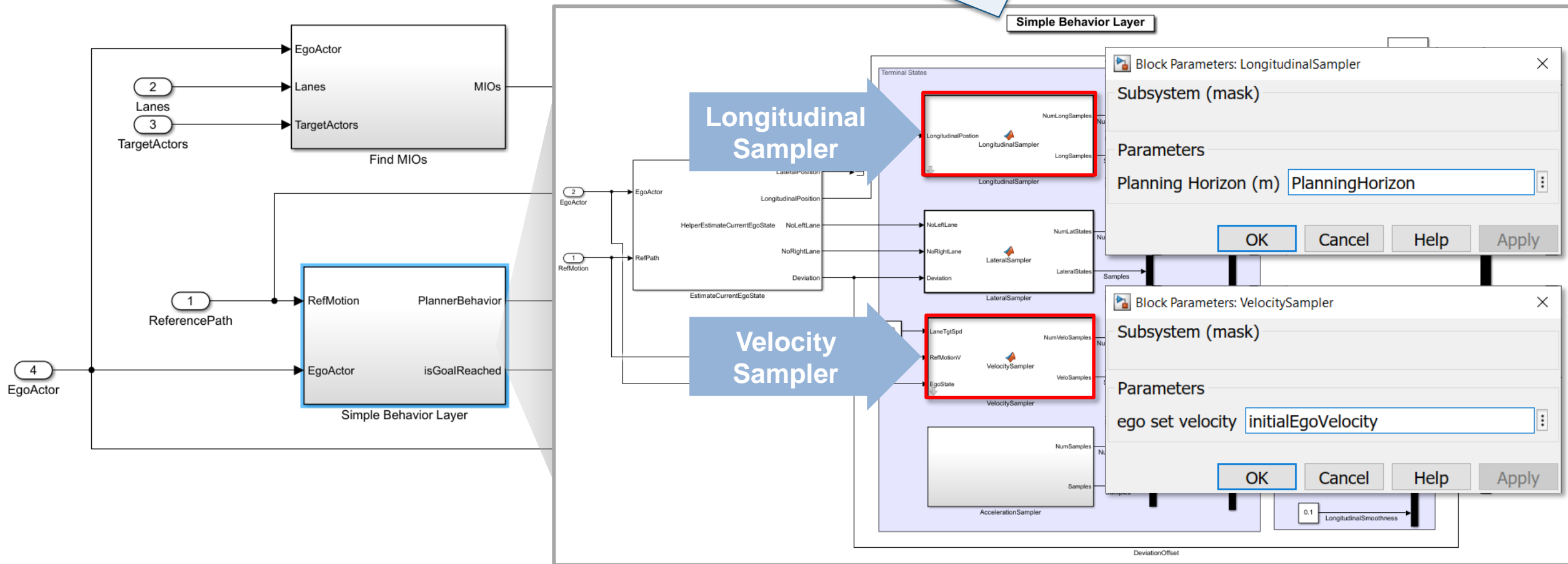
Command Window

```
>> helperSLHighwayLaneChangeSetup("scenario_LC_11_Hwy96_M23_WithMergingCar")
fx>> |
```

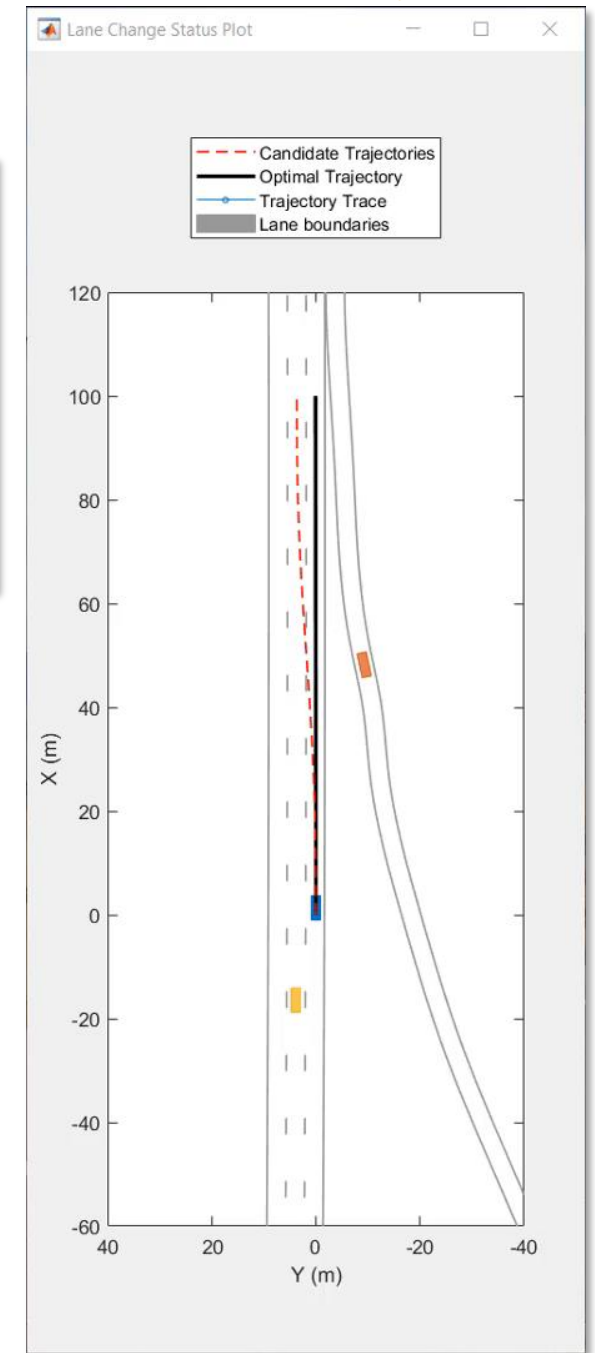
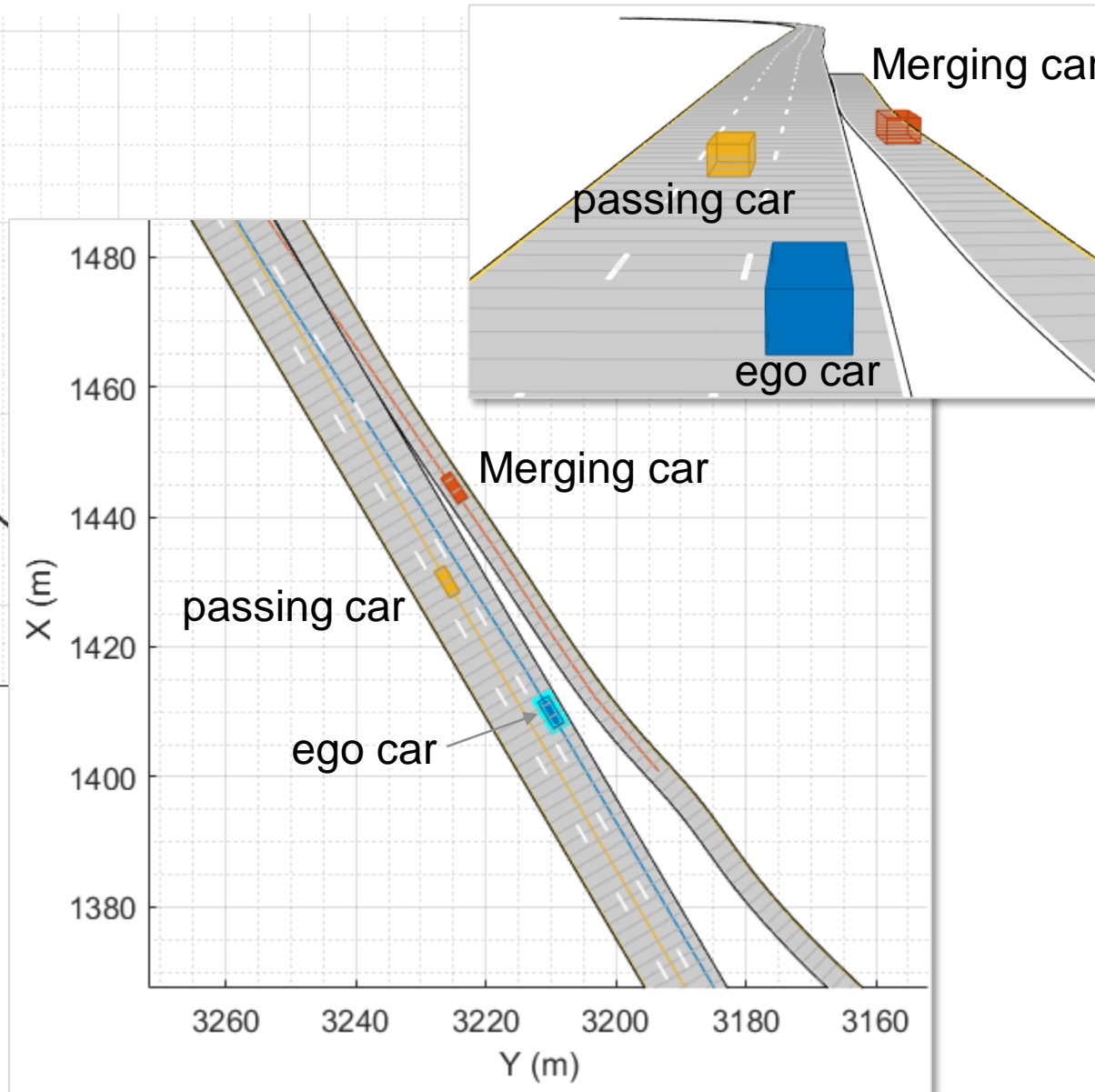
# Update Lane Change Planner ▷ Simple Behavior Layer

## Lane Change Planner

Update longitudinal & velocity sampler to read input parameter

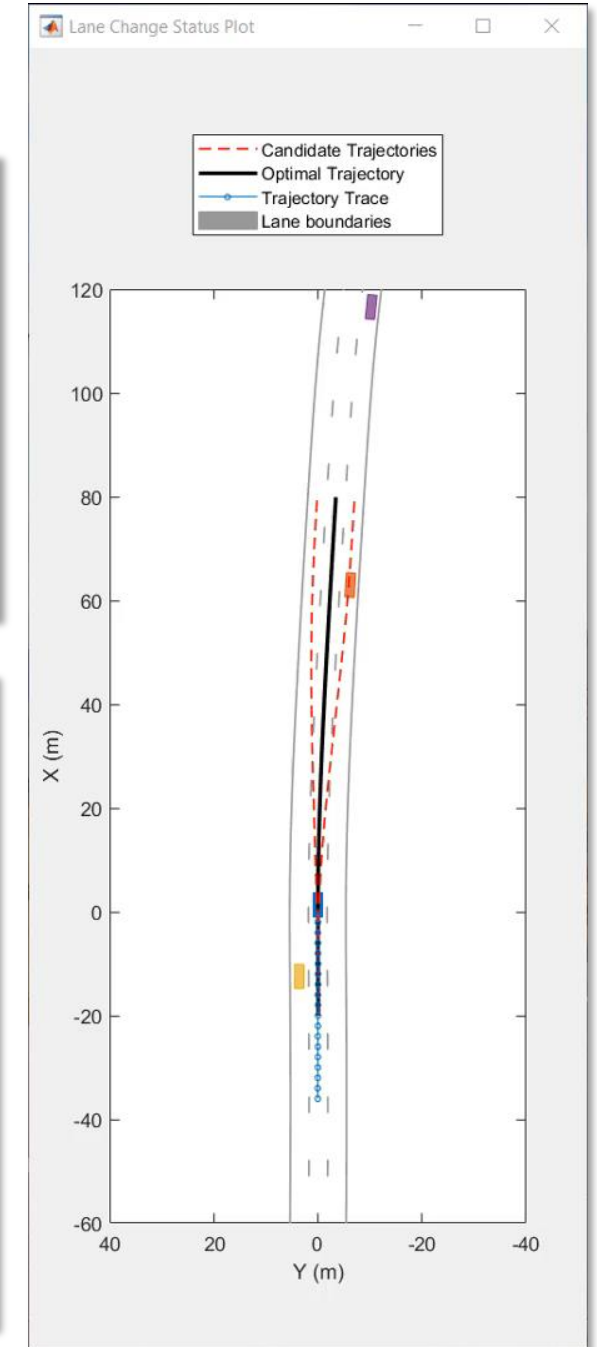
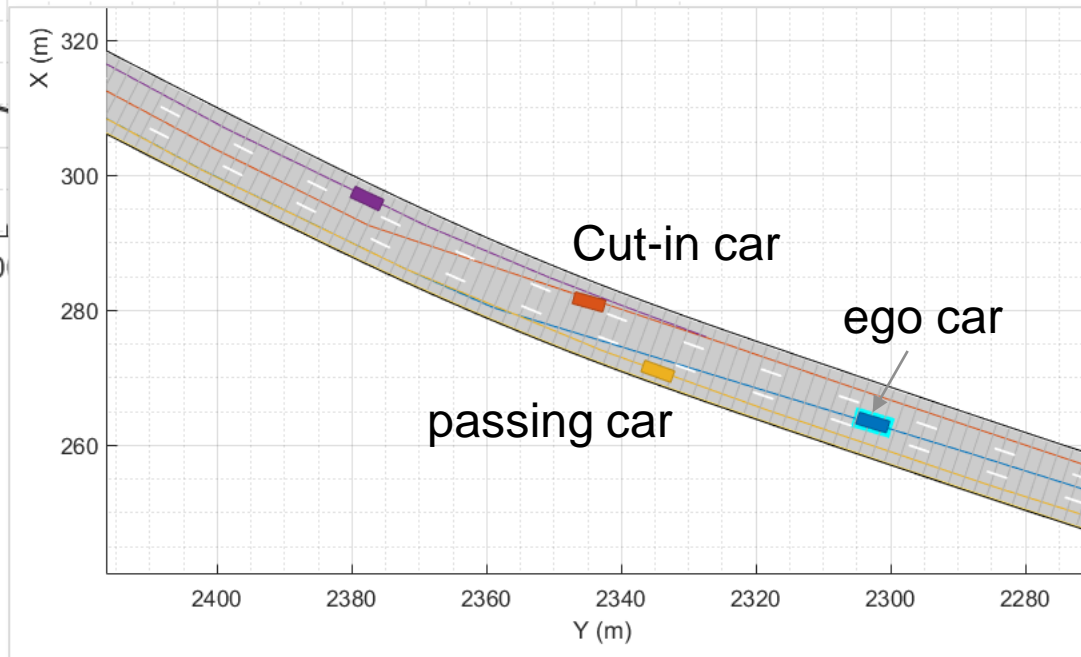
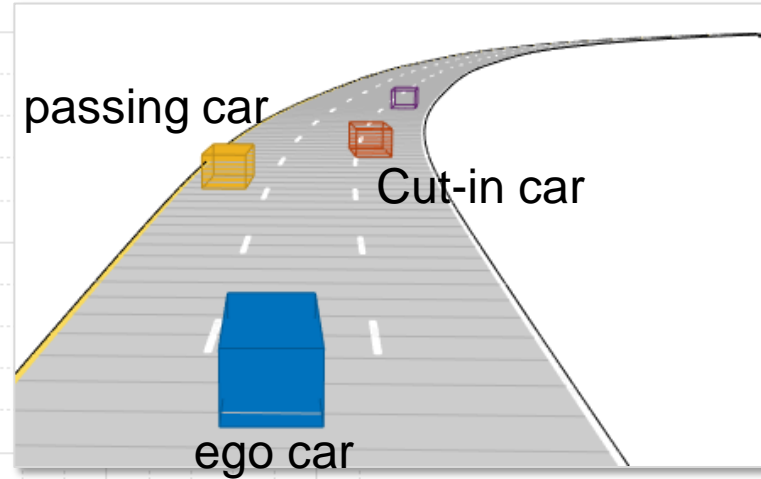
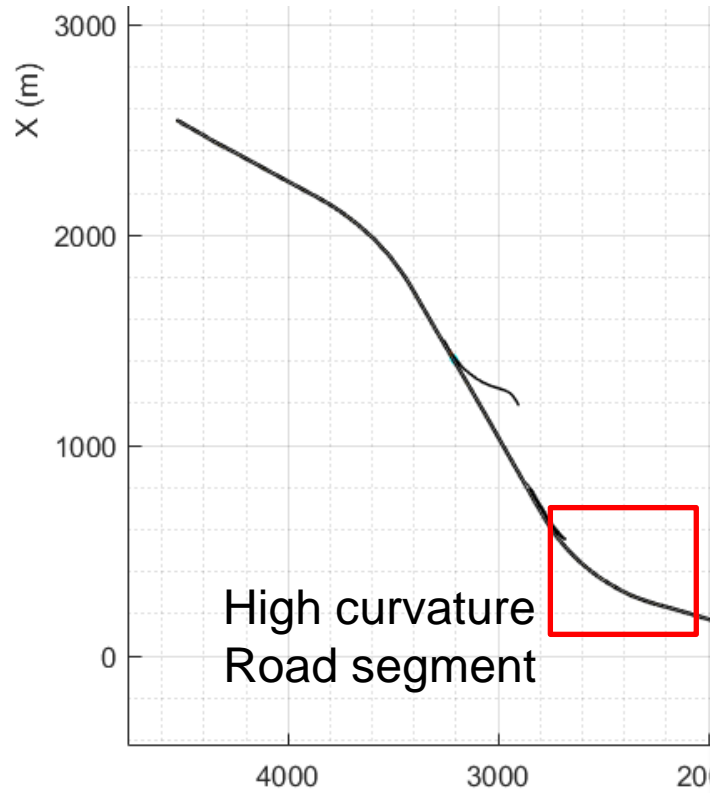


# scenario\_LC\_11\_Hwy96\_M23\_WithMergingCar

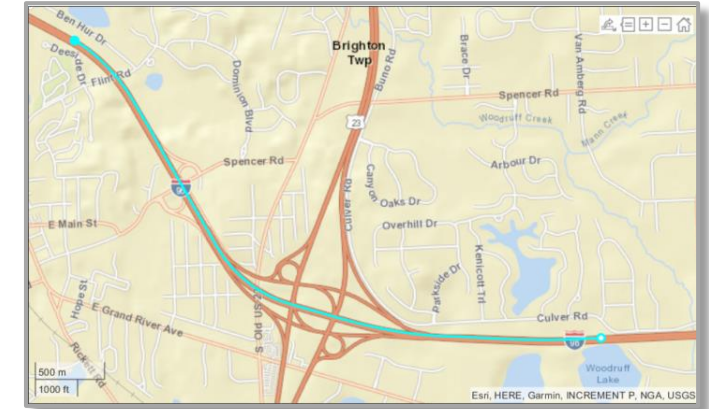
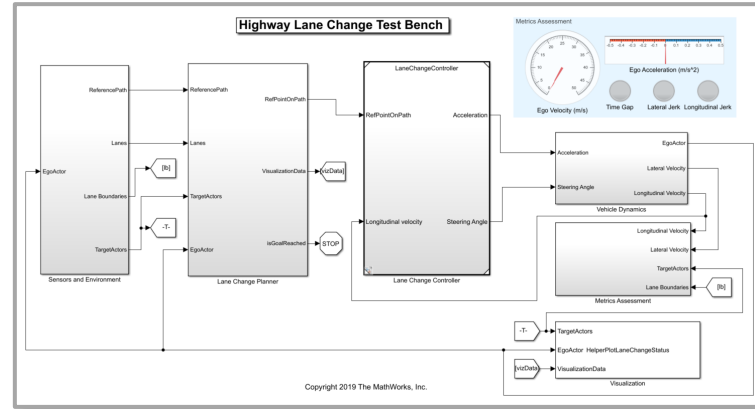
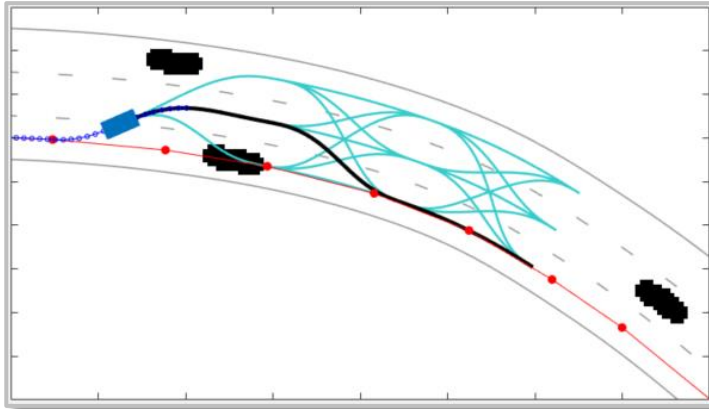




# scenario\_LC\_12\_Hwy96\_M23\_WithCutInCar



# Recap: Highway Lane Change



## Learn motion planner

- Optimal trajectory generation in Frenet coordinate
- Planner parameters
- Simulate trajectory planning with occupancy map

## Integrate motion planner and controller

- Learn through reference example
- Architecture of highway lane change
- Simulate closed-loop controller with test scenarios

## Test with real-world scenario

- Create scenario from HD map
- Update setup script and model
- Run simulation with new scenario

# Presenter contact info and poll questions

Please contact me at [spark@mathworks.com](mailto:spark@mathworks.com) with questions

- Poll question : I found this technique the most interesting
  - a. Motion planner for lane change
  - b. MPC path following controller
  - c. Driving scenario creation
  - d. Import HD map into driving scenario
- If you would like to an individual follow-up, please let us know in the WebEx poll area.