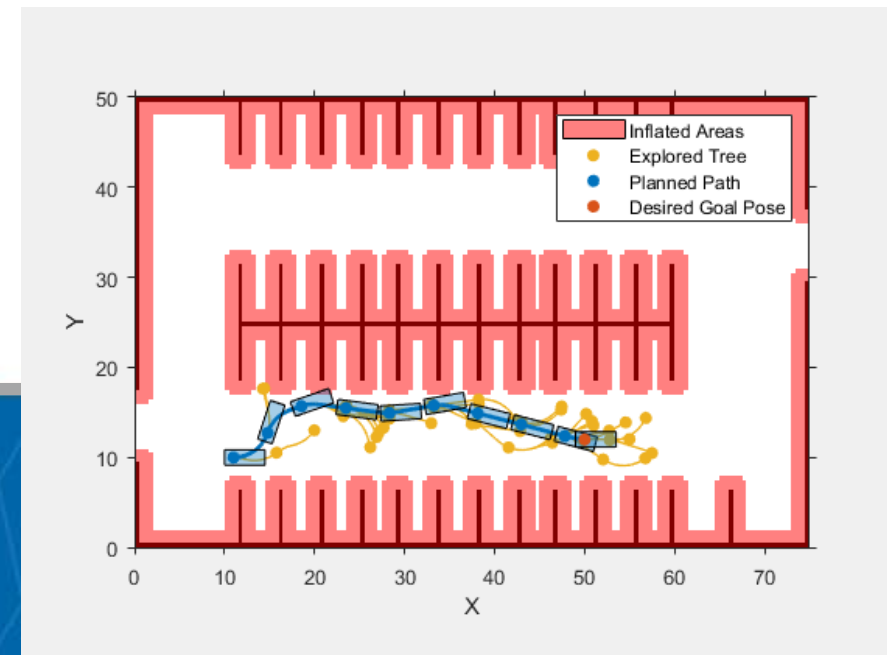


Evaluate Path Planner and Controller for Automated Parking

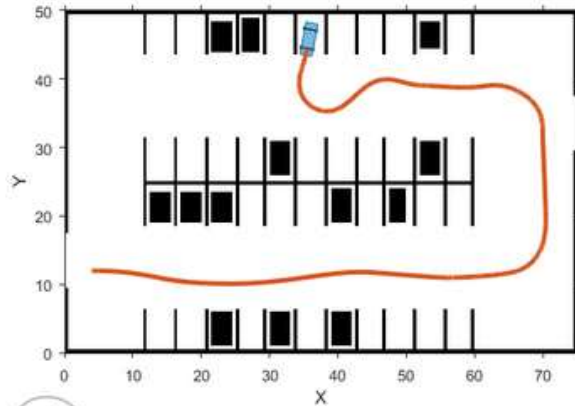
Shusen Zhang

Application Engineering, MathWorks



Learn about path planning with these examples

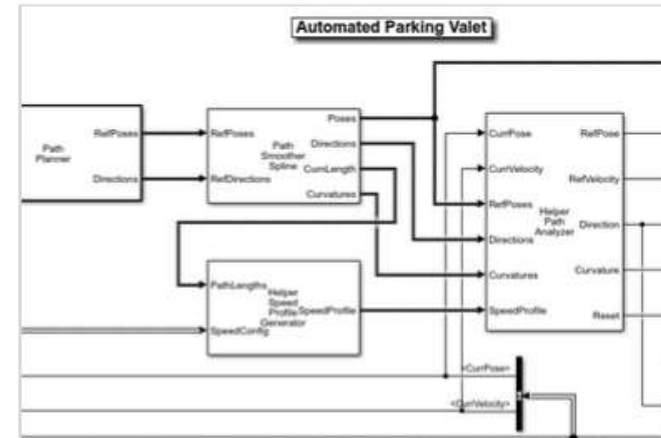
MATLAB
example



Automated Parking Valet

Construct an automated parking valet system using path planning, trajectory generation, and vehicle control techniques.

[Open Script](#)



Automated Parking Valet in Simulink

Construct an automated parking valet system in Simulink with Automated Driving Toolbox.

[Open Model](#)

Simulink test
bench

Learn about path planning with these examples



Code Generation for Path Planning and Vehicle Control

Generate C++ code for a path planning and vehicle control Simulink model, and verify the generated code using software-in-

[Open Script](#)

How robust is the algorithm?

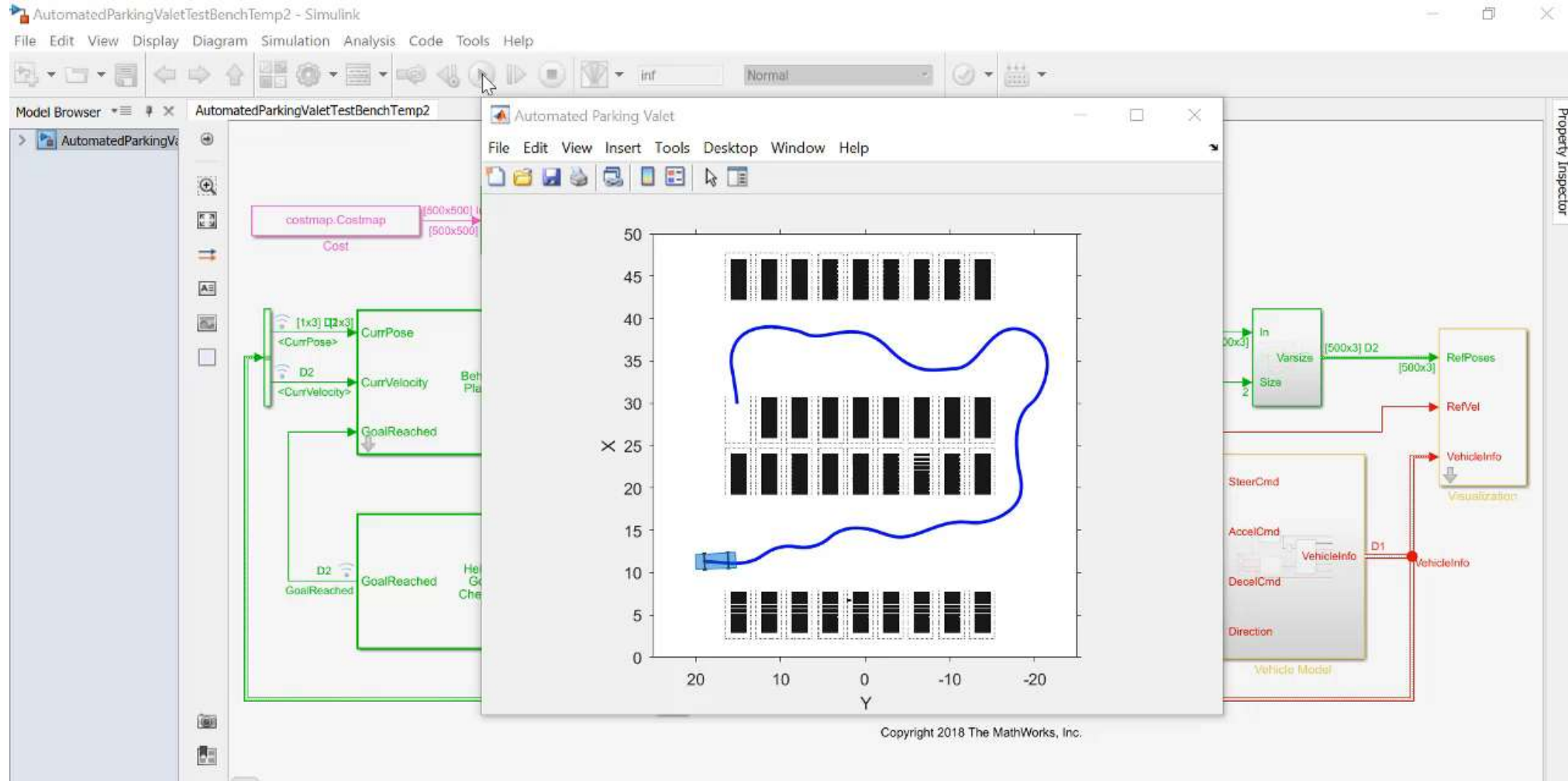
How can I handle moving pedestrian

How can I automate the tests?

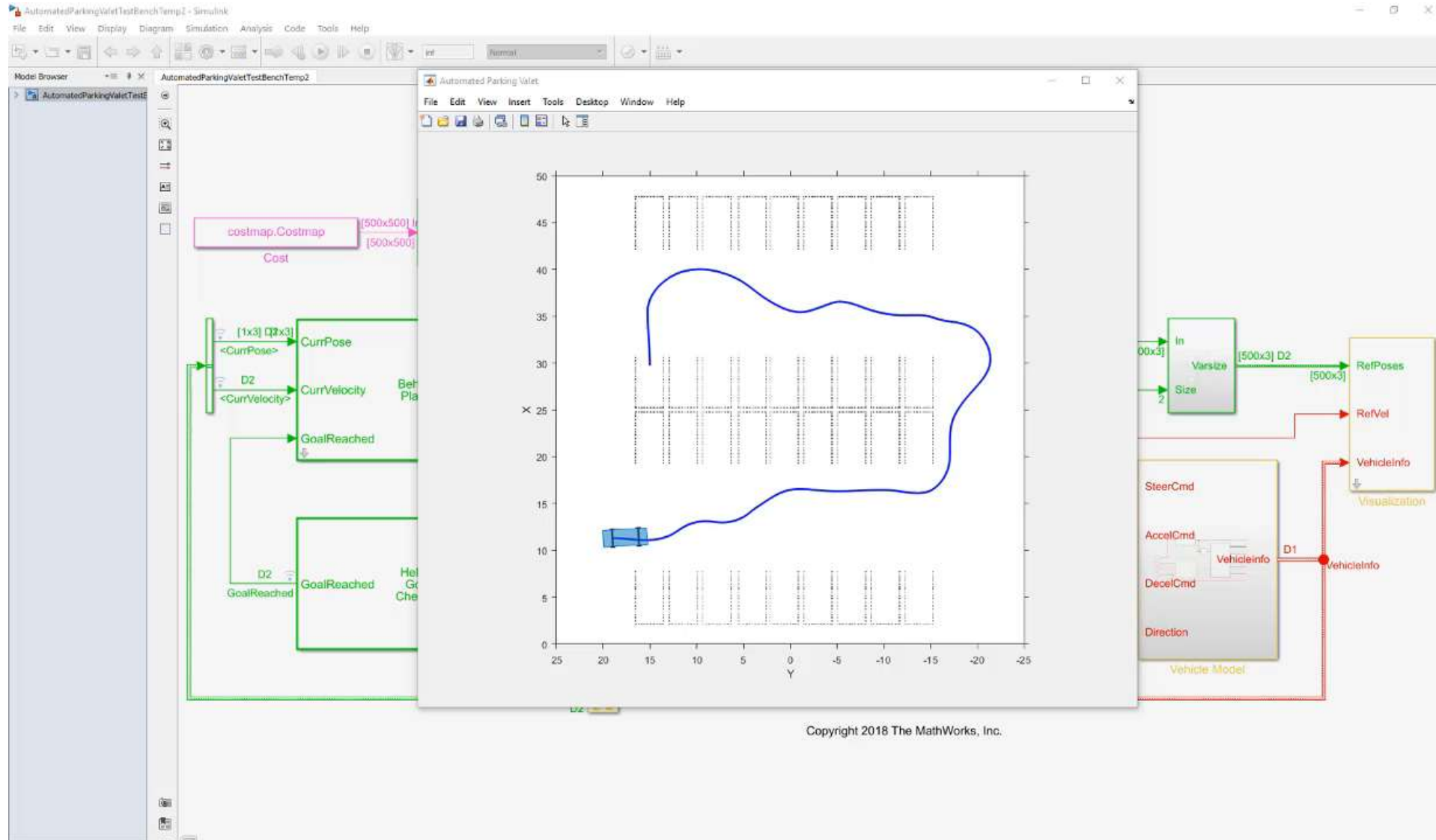
Evaluate Path Planner and Controller for Automated Parking

- Explore system robustness with simulation
- Improve design to handle moving pedestrian
- Test automation for regression tests

Requirement 1: vehicle can only move forward

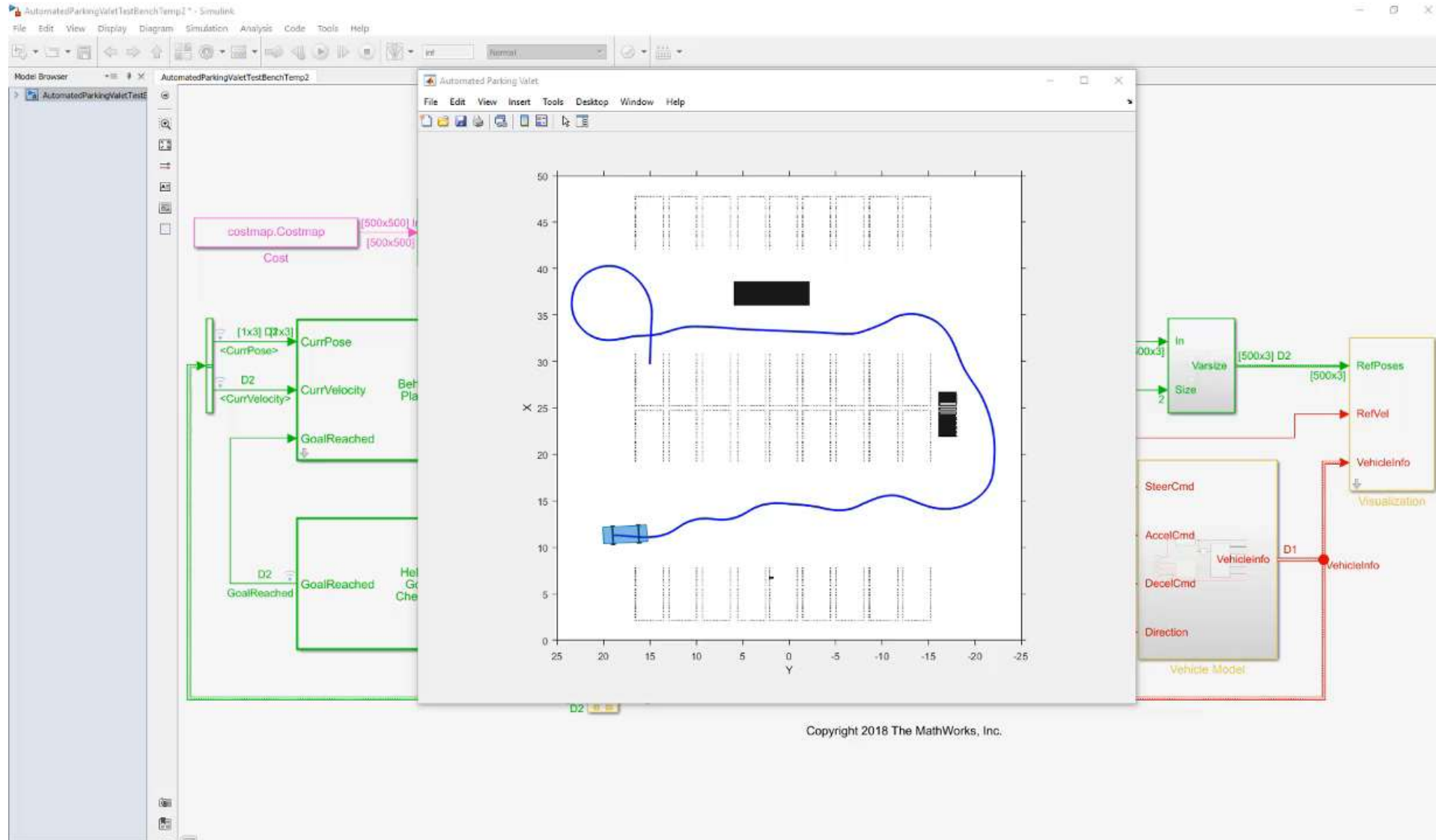


Requirement 2: vehicle can't cross parking lanes



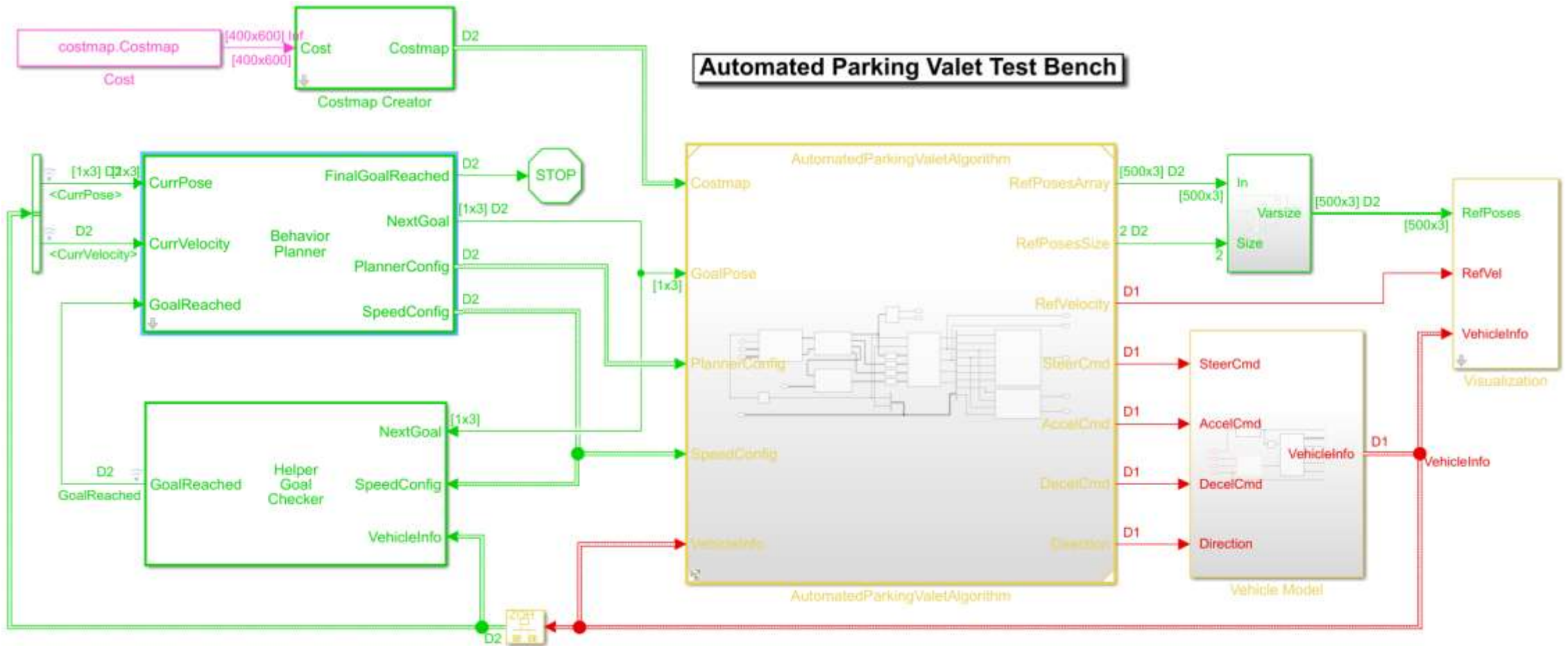
Copyright 2018 The MathWorks, Inc.

Requirement 3: algorithm must be able to handle loop.



Copyright 2018 The MathWorks, Inc.

Baseline model in the product



Copyright 2018 The MathWorks, Inc.

Single test case in the baseline model



Block Parameters: BehaviorPlanner

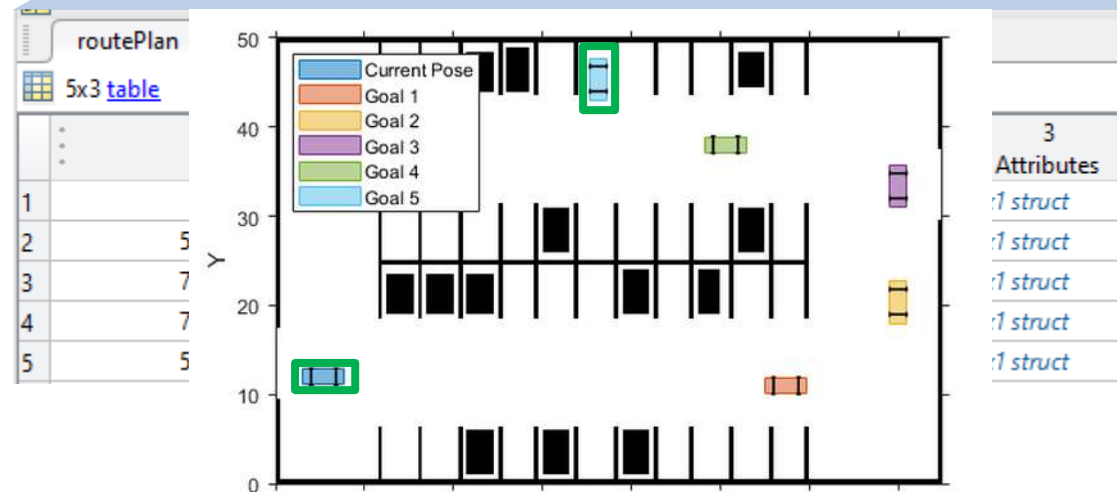
Subsystem (mask)
 Activate a sequence of navigation tasks from global route plan.

Parameters

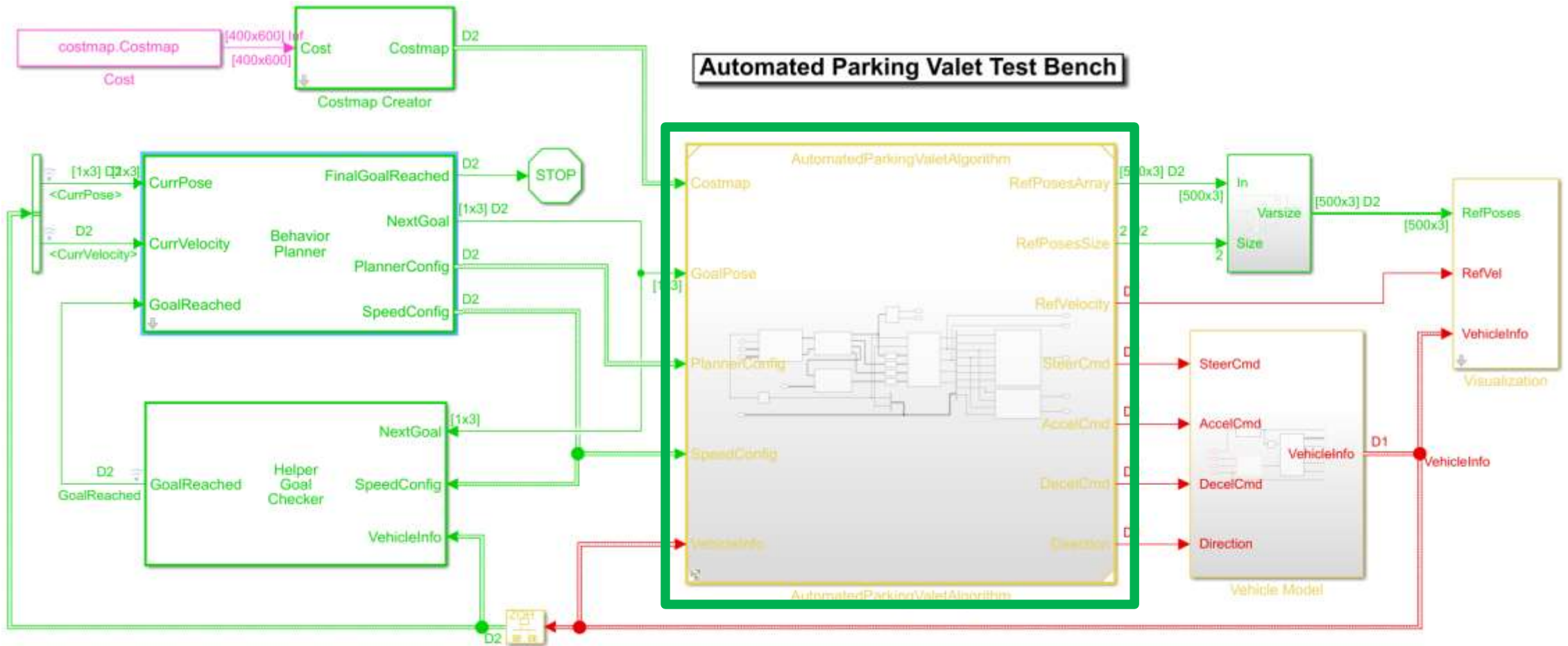
Route plan:

Initial vehicle speed [m/s]:

OK Cancel Help Apply



Software model

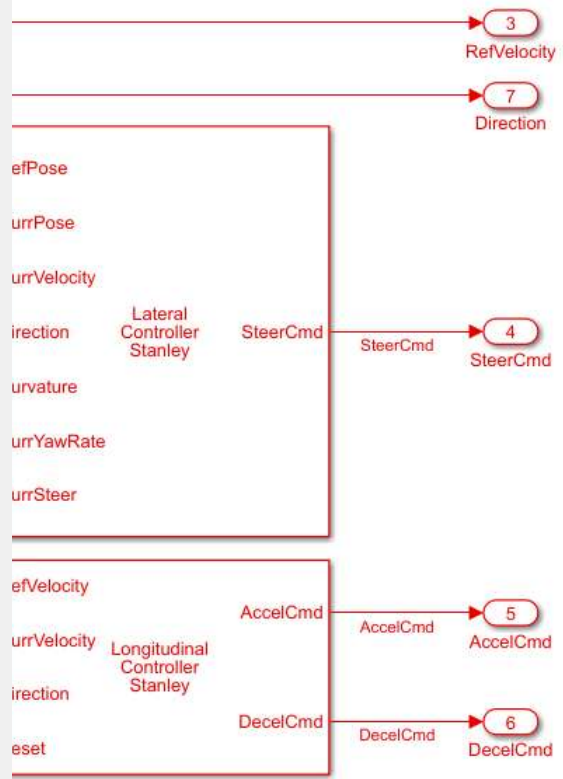
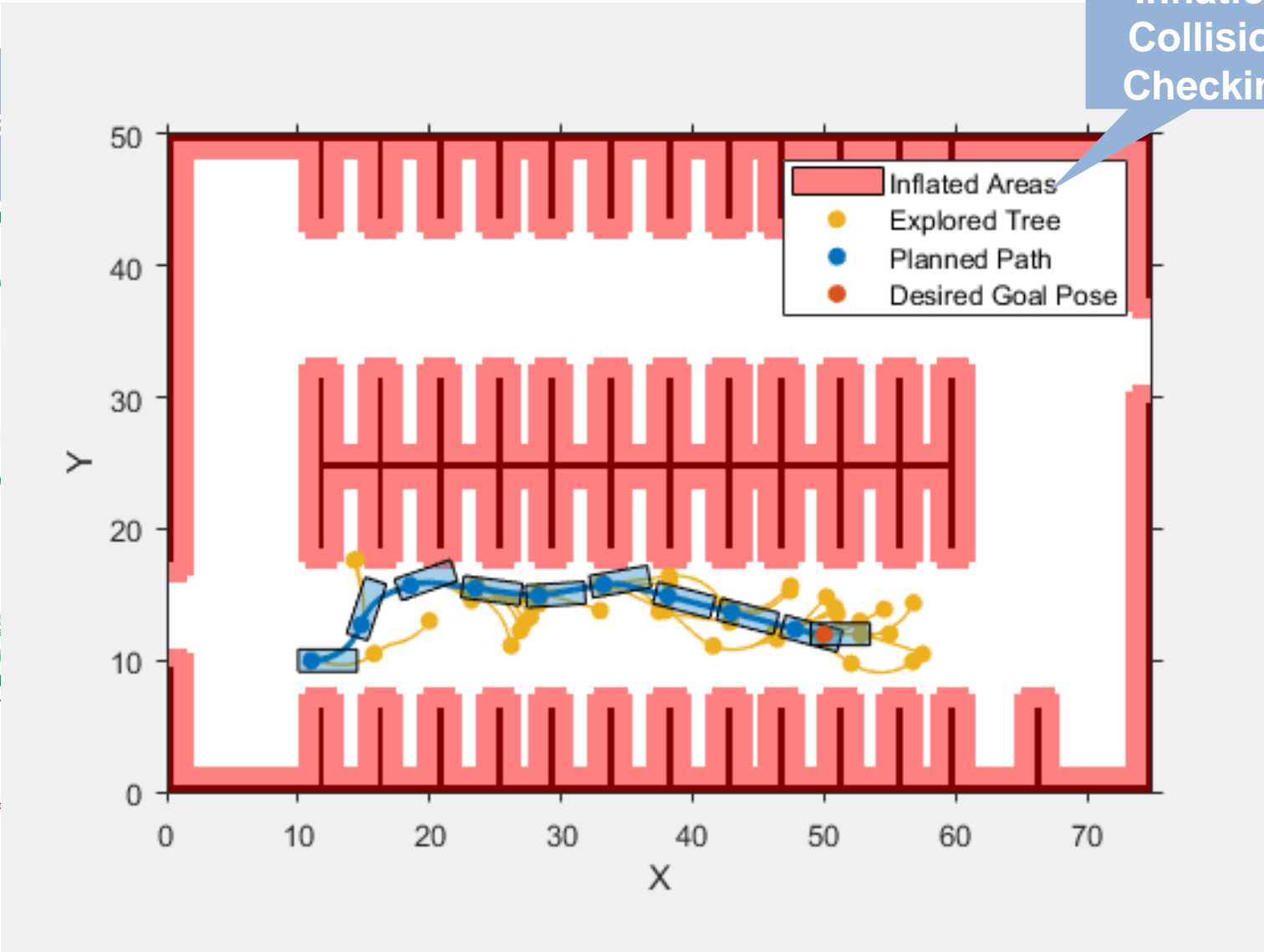
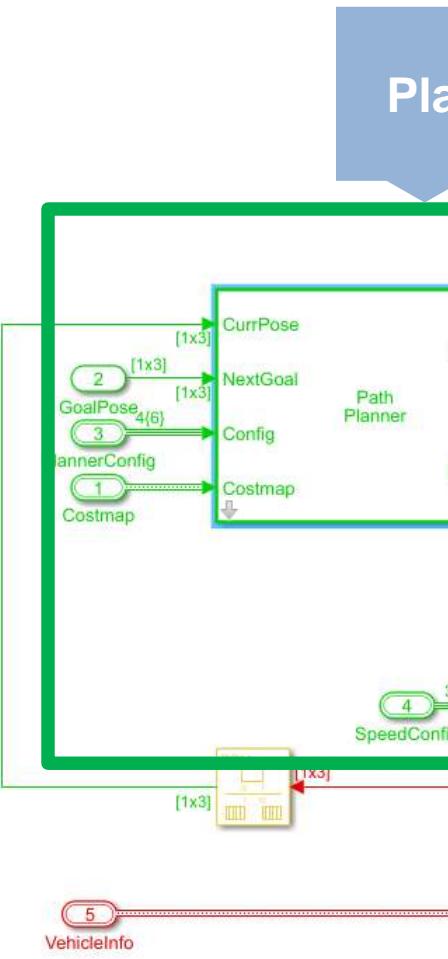


Copyright 2018 The MathWorks, Inc.

Main modules

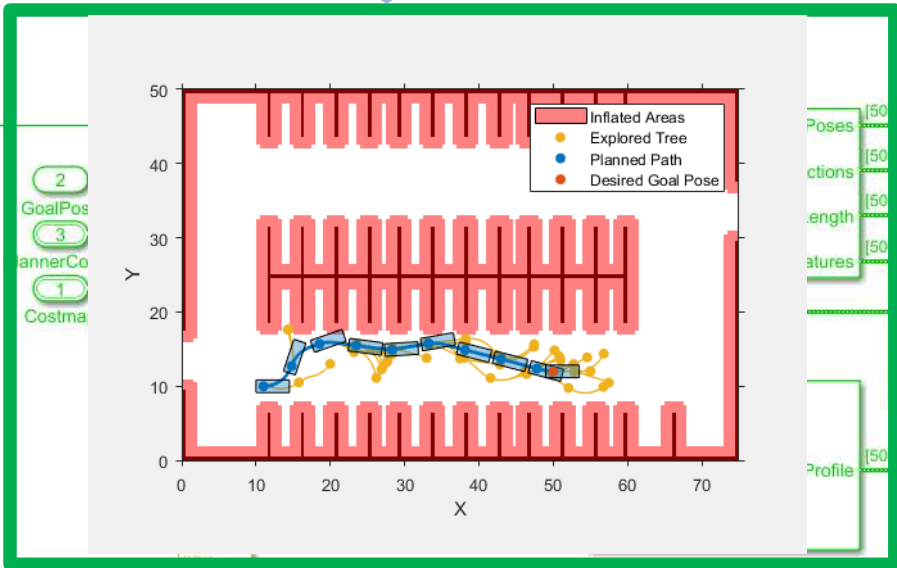
Plan

Inflation
Collision
Checking

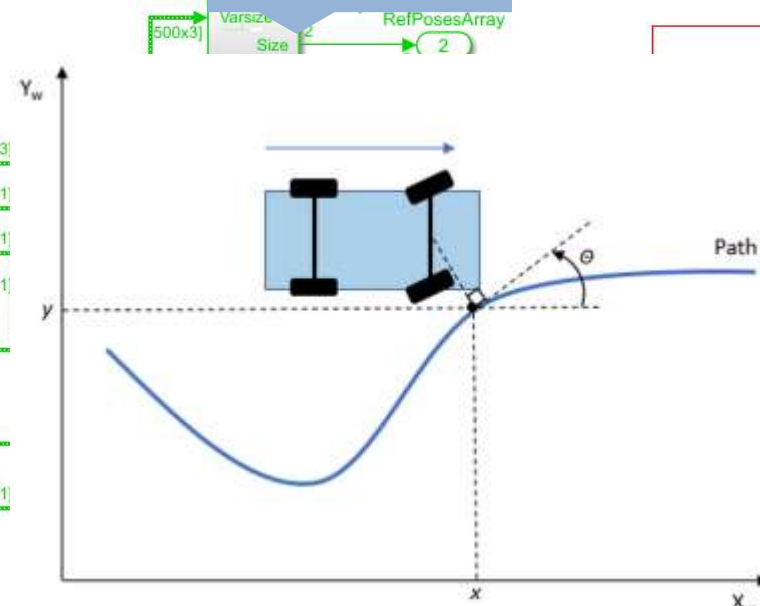


Main modules

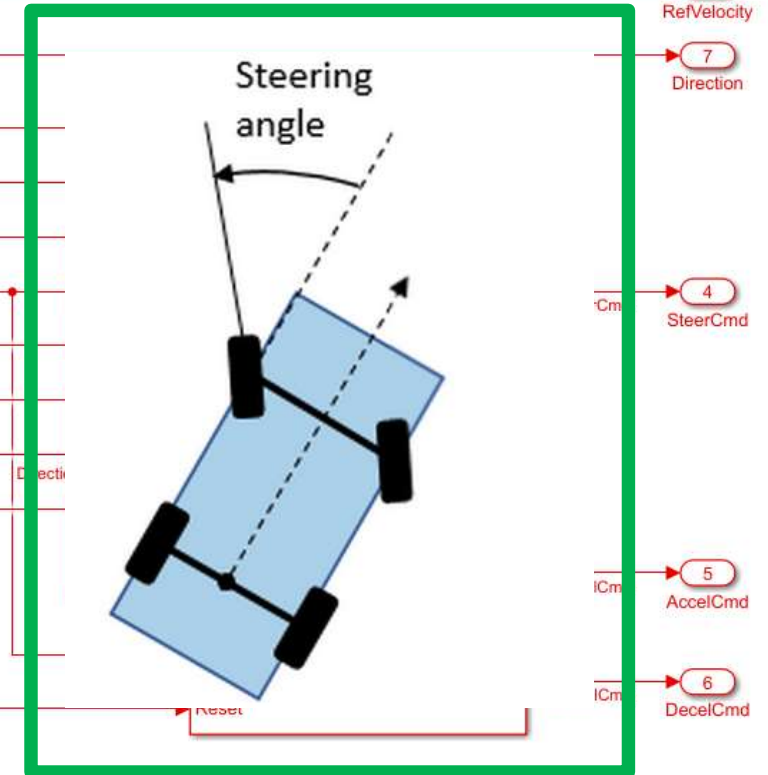
Planning



Path Analyzer



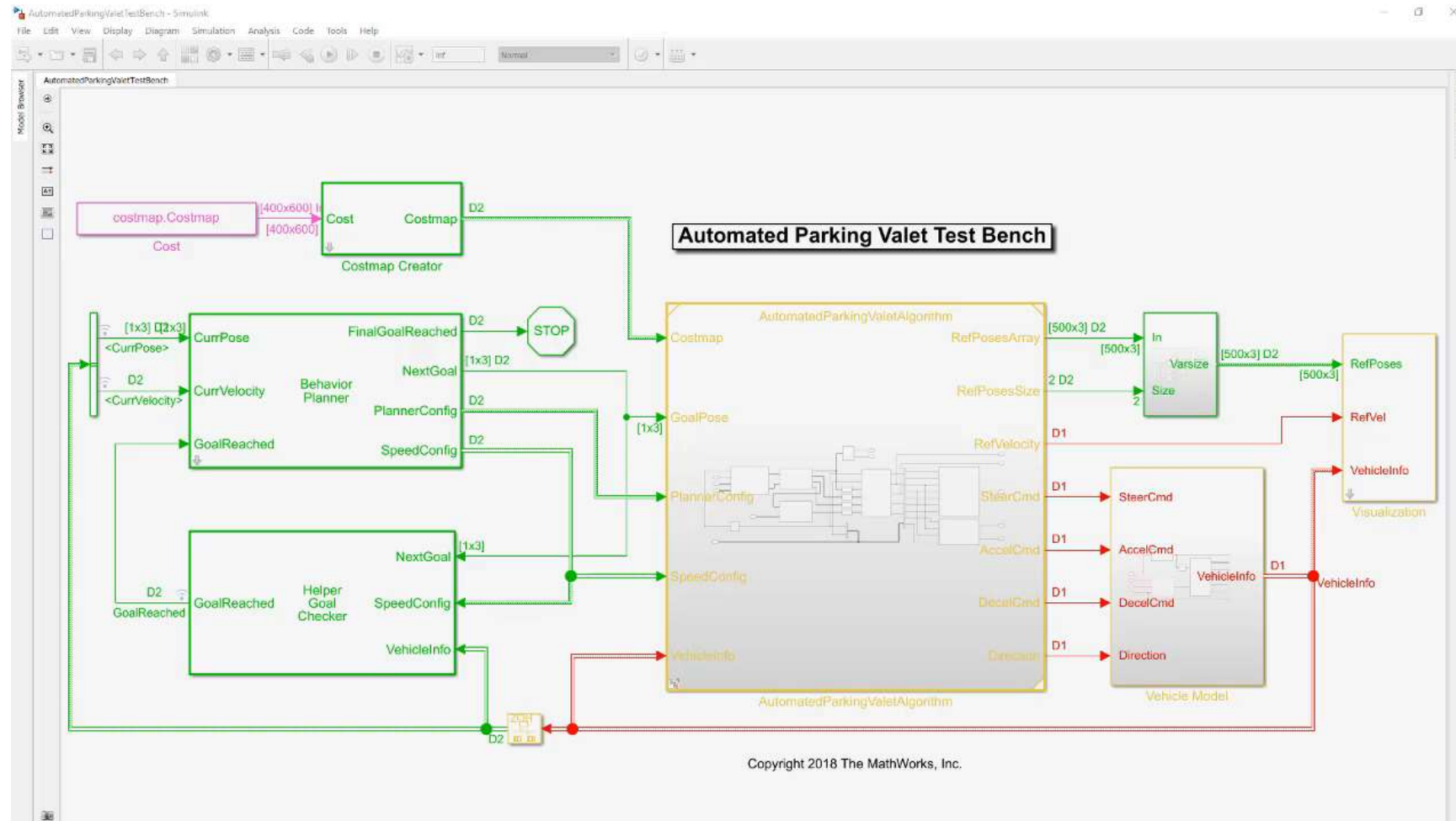
Control



5 VehicleInfo

[1x3]
<CurrVelocity>

Explore baseline behavior with multiple goal poses

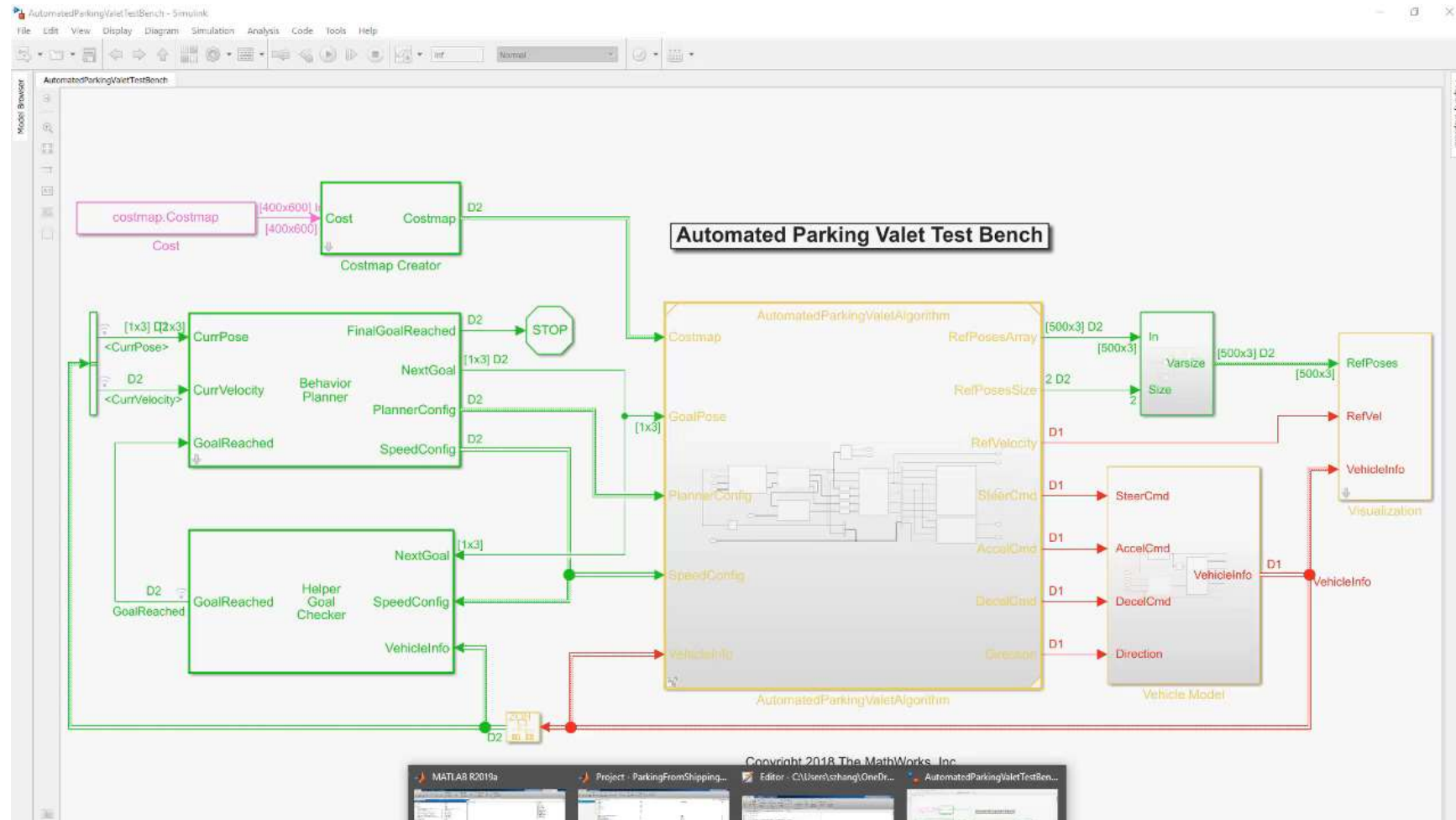


Explore behavior with single goal pose

The image shows a MATLAB/Simulink interface. On the left, a 'Behavior Planner' block is highlighted with a green border. Its inputs and outputs are: CurrPose, CurrVelocity, GoalReached, FinalGoalReached, NextGoal, and SpeedConfig. A blue callout points to the 'Block Parameters: BehaviorPlanner' dialog box. In this dialog, the 'Route plan' parameter is set to 'routePlan' and the 'Initial vehicle speed [m/s]' is set to '0'. Below the dialog, a 'routePlan' table is displayed with 5 rows and 7 columns. The first three columns are grouped under '1 StartPose' and the last three under '3 Attributes'. Two rows (1 and 5) are highlighted with green boxes.

	1 StartPose			3 Attributes		
1	4	12	0	36.3000	44	90 1x1 struct
2	56	11	0	70	19	90 1x1 struct
3	70	19	90	70	32	90 1x1 struct
4	70	32	90	52	38	180 1x1 struct
5	53	38	180	36.3000	44	90 1x1 struct

Explore behavior with single goal pose



Changing testing condition



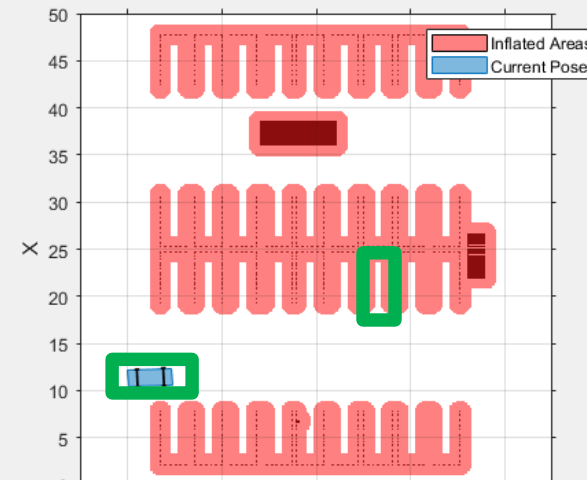
Block Parameters: BehaviorPlanner [X]

Subsystem (mask)
 Activate a sequence of navigation tasks from global route plan.

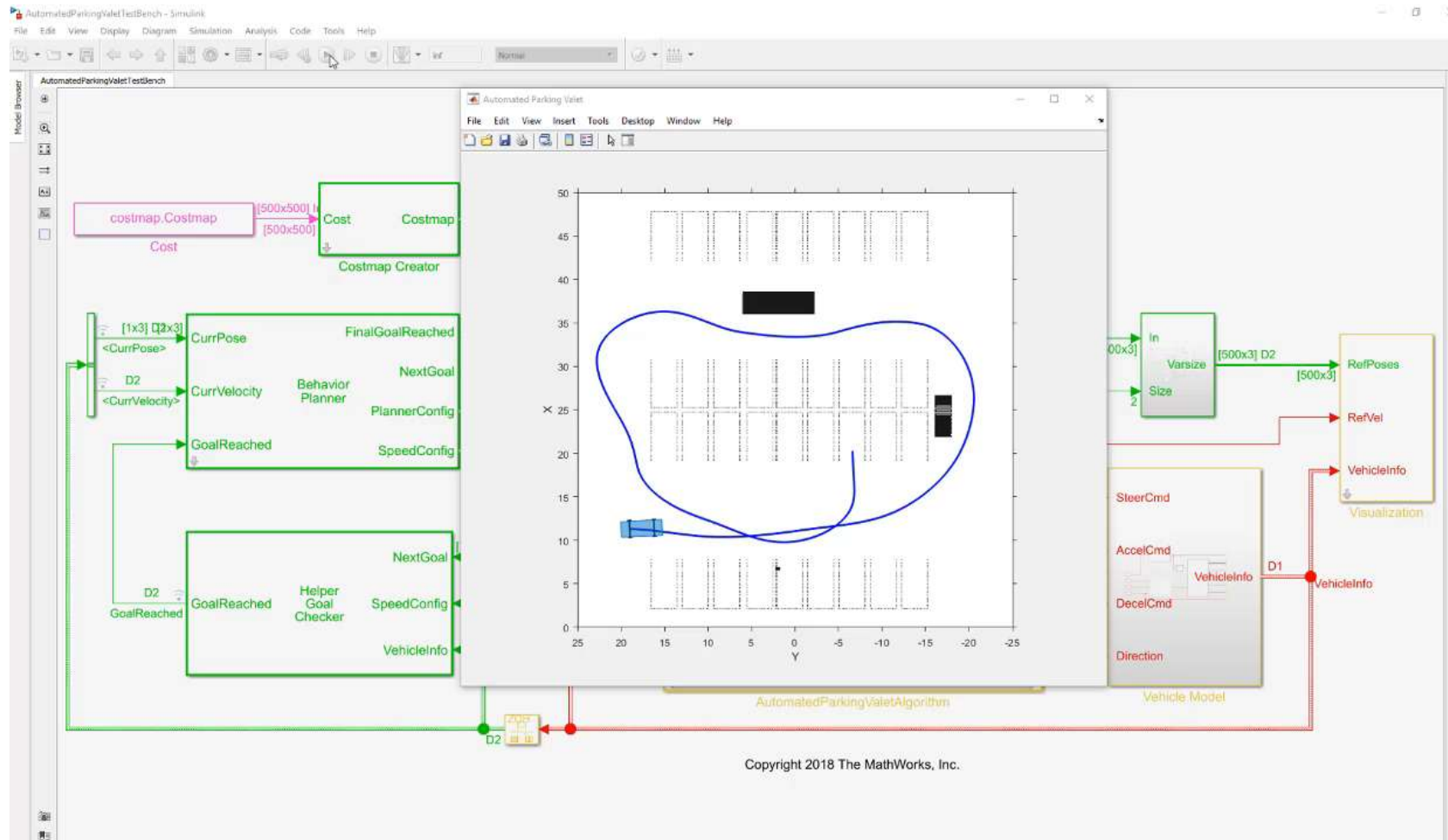
Parameters

Route plan: [v]

Initial vehicle speed [m/s]: [v]

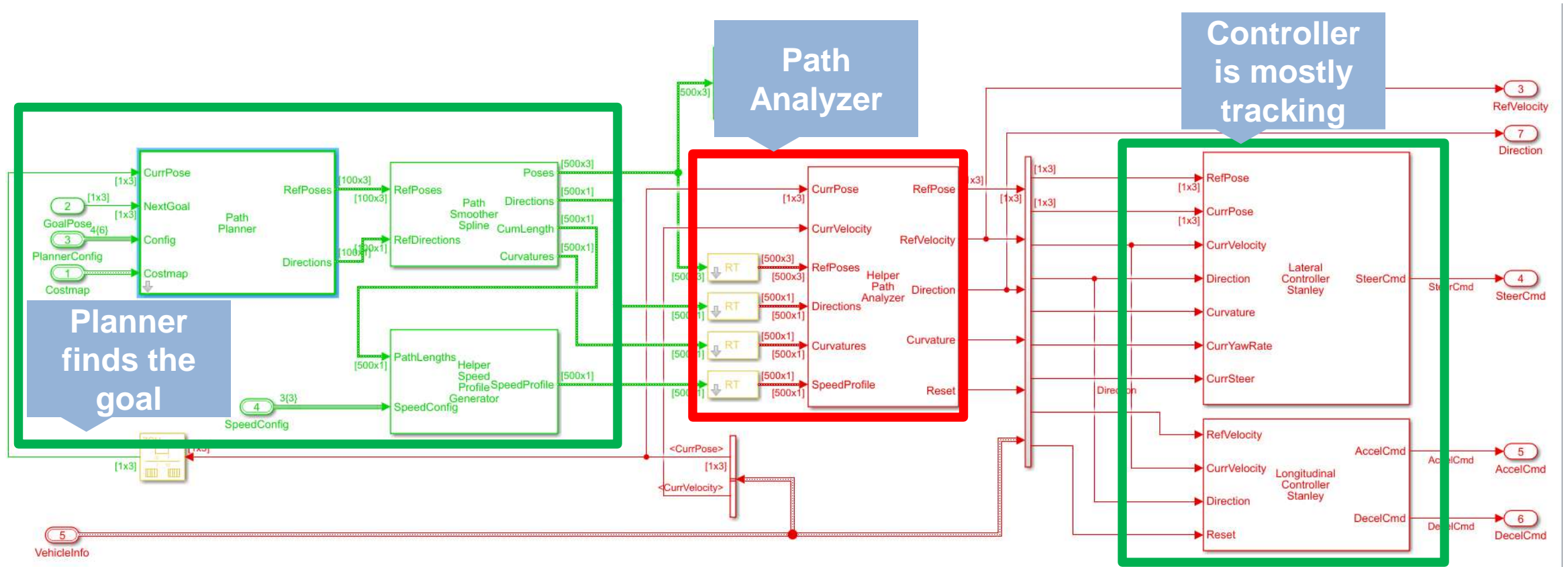


Explore behavior with goal pose that generates a loop in path



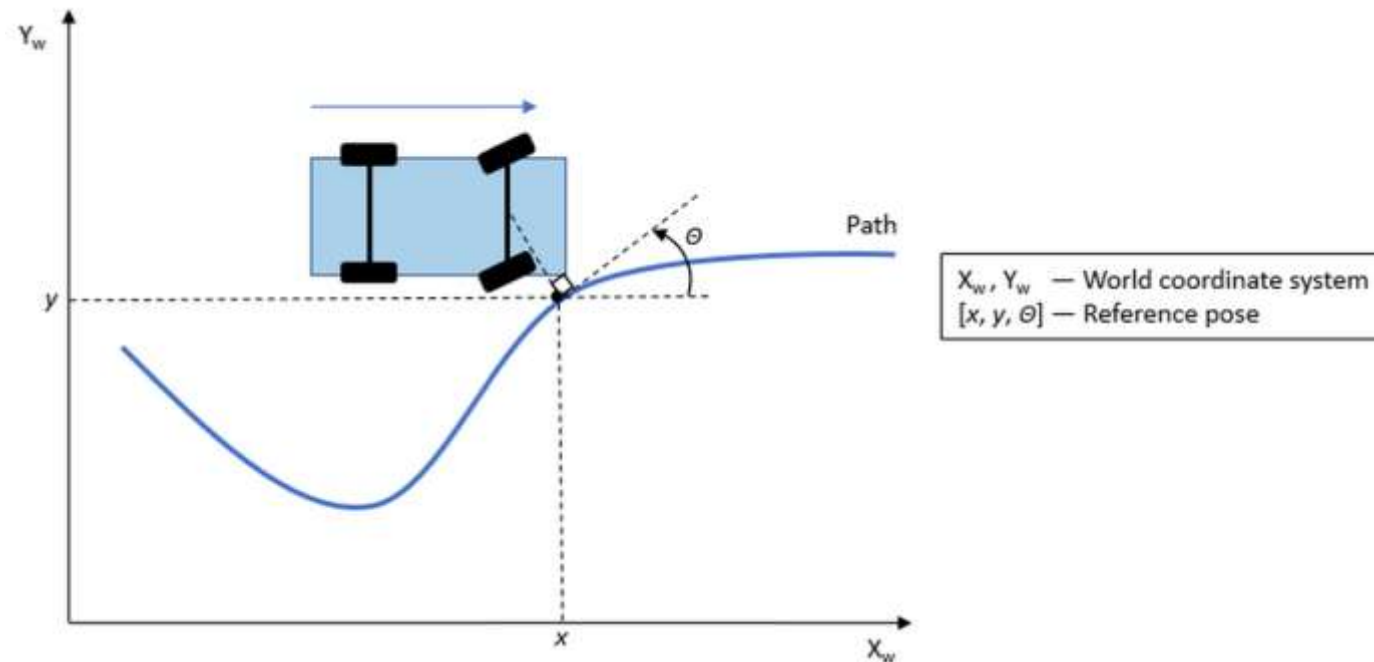
Copyright 2018 The MathWorks, Inc.

Isolate issue based on simulation results



Existing path analyzer

- Find a point on the path for the vehicle to follow

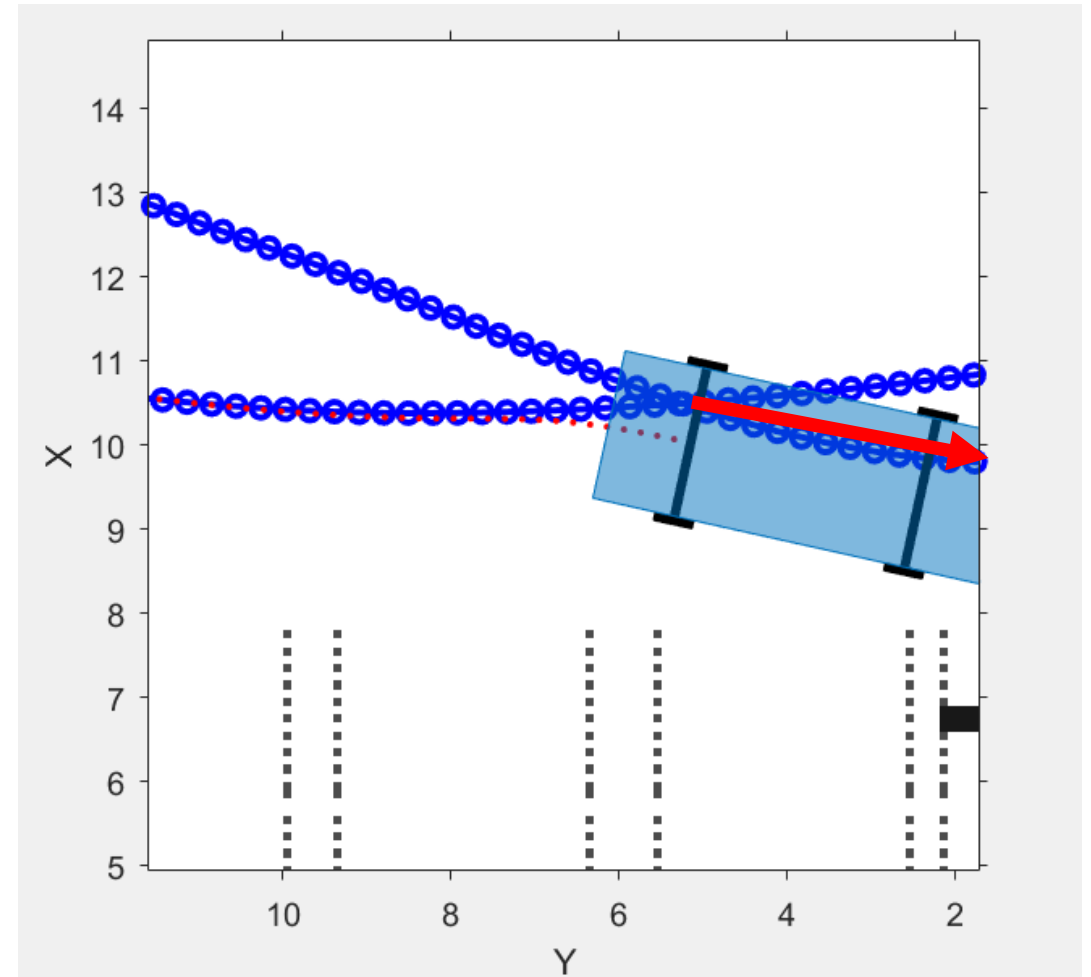
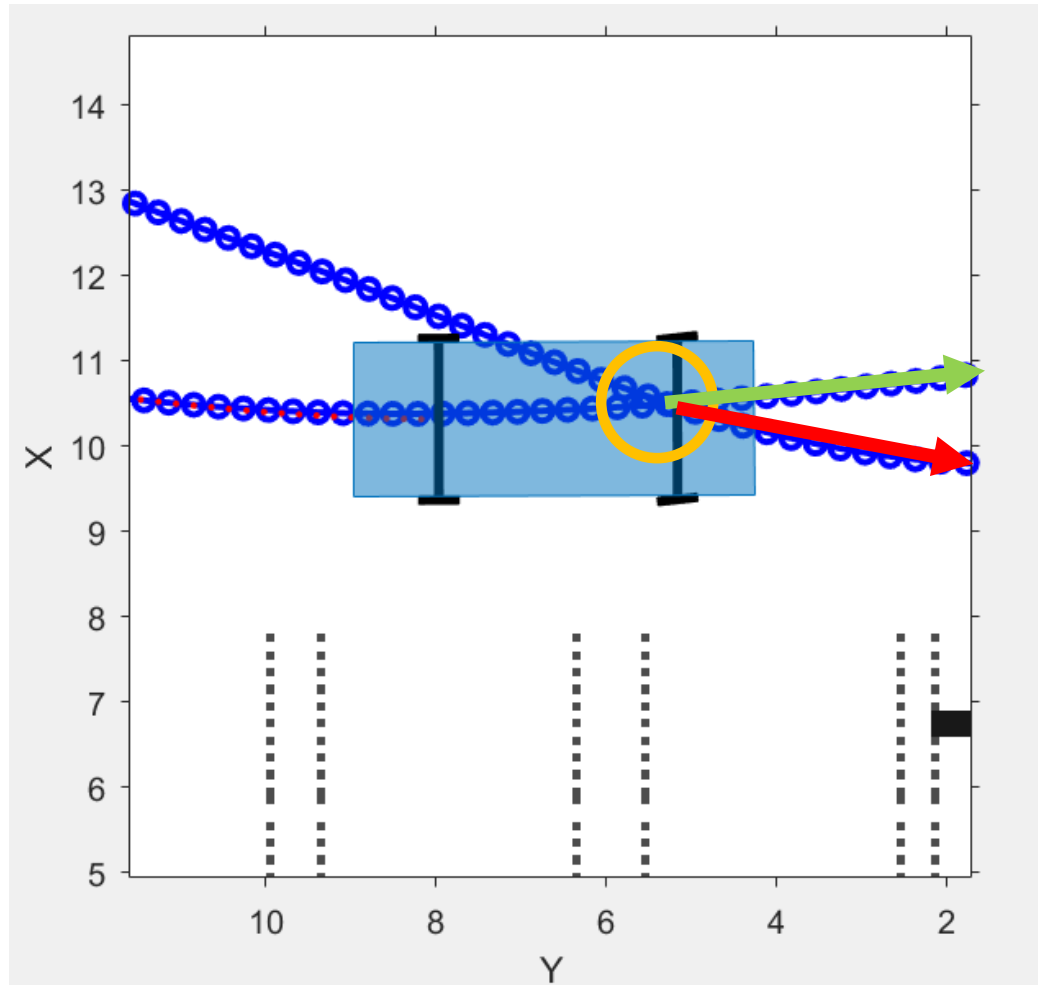


```
% Find the closest point on the reference path
[closestIdx, ~] = find(dis2PointsSquare == min(dis2PointsSquare), 1);

% Enforce to be a scalar in Simulink
closestIdx = closestIdx(1);
```

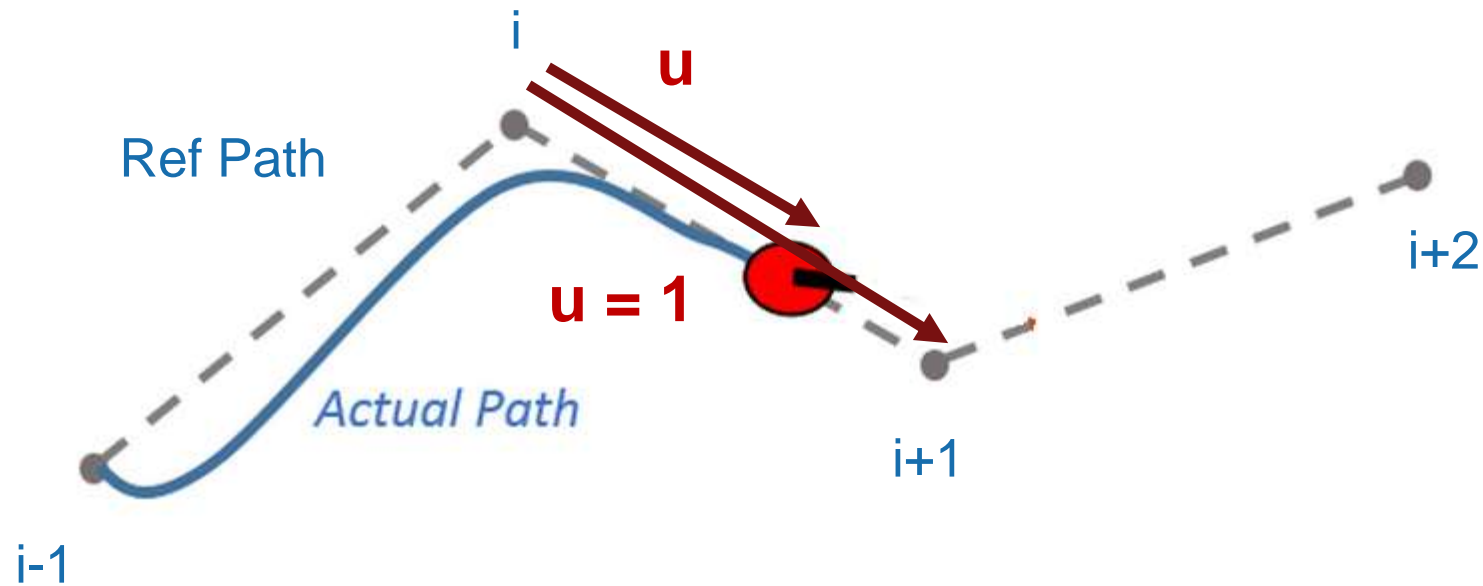
Minimum
distance to entire
path

Existing path analyzer



Modified path analyzer

- Find a point on the path for the vehicle to follow

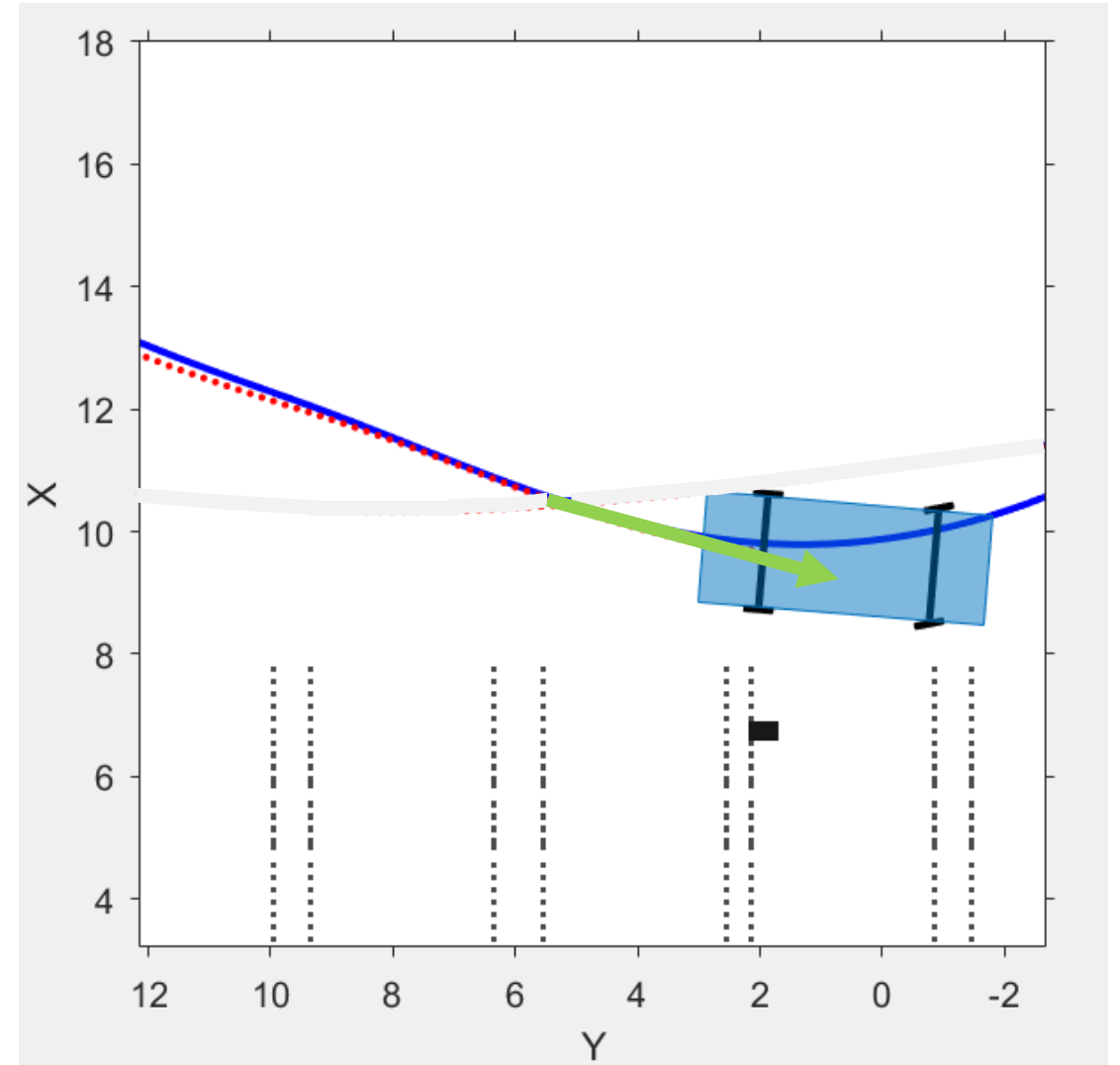
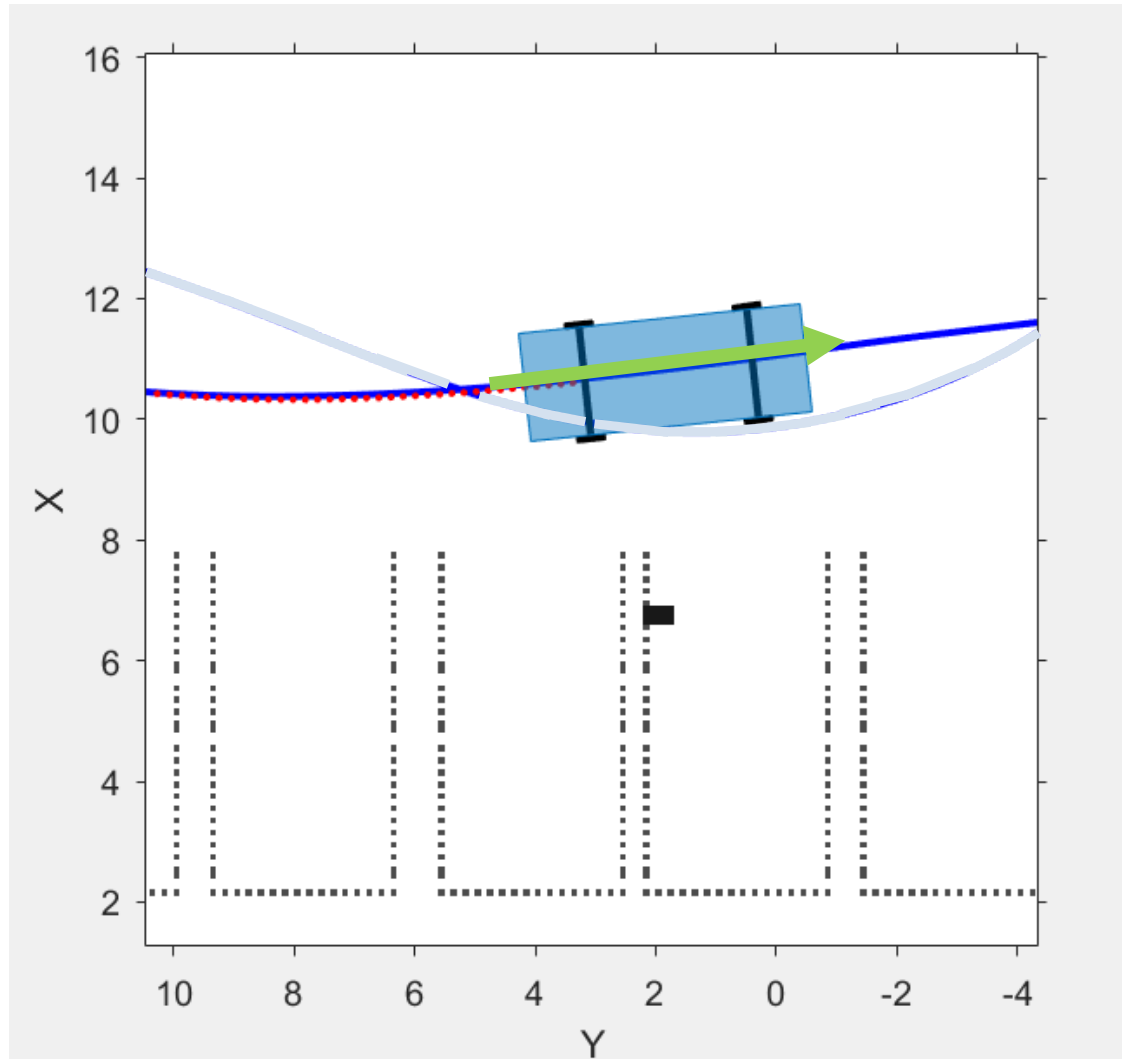


```
% Normalized distance between current position and section starting point
u = (RXY.*DeltaXY) / (DeltaXY.*DeltaXY);
```

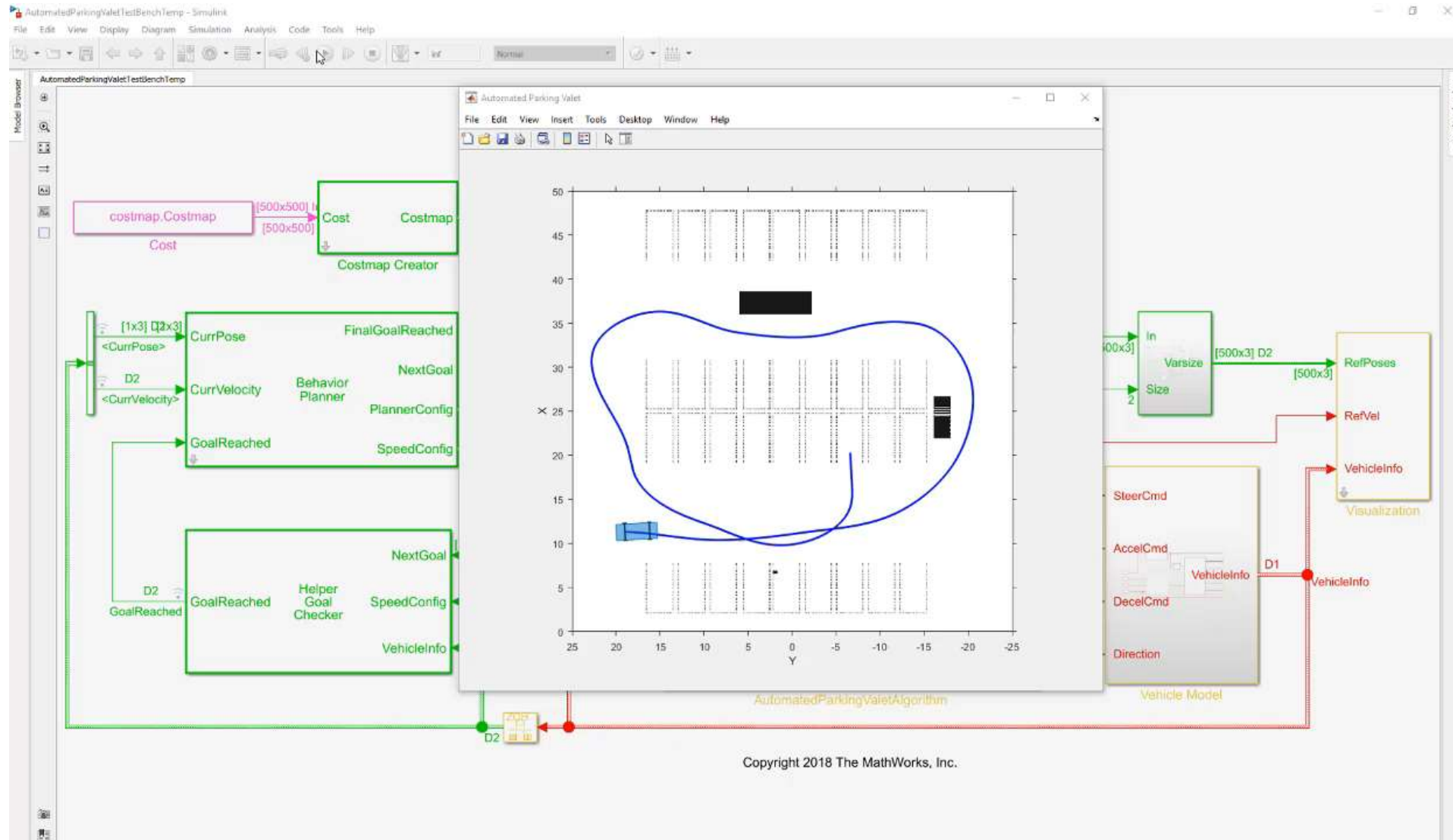
```
% Find section ending point
indexIncrement = ceil(u-1);
```

Incremental
projection

Modified path analyzer

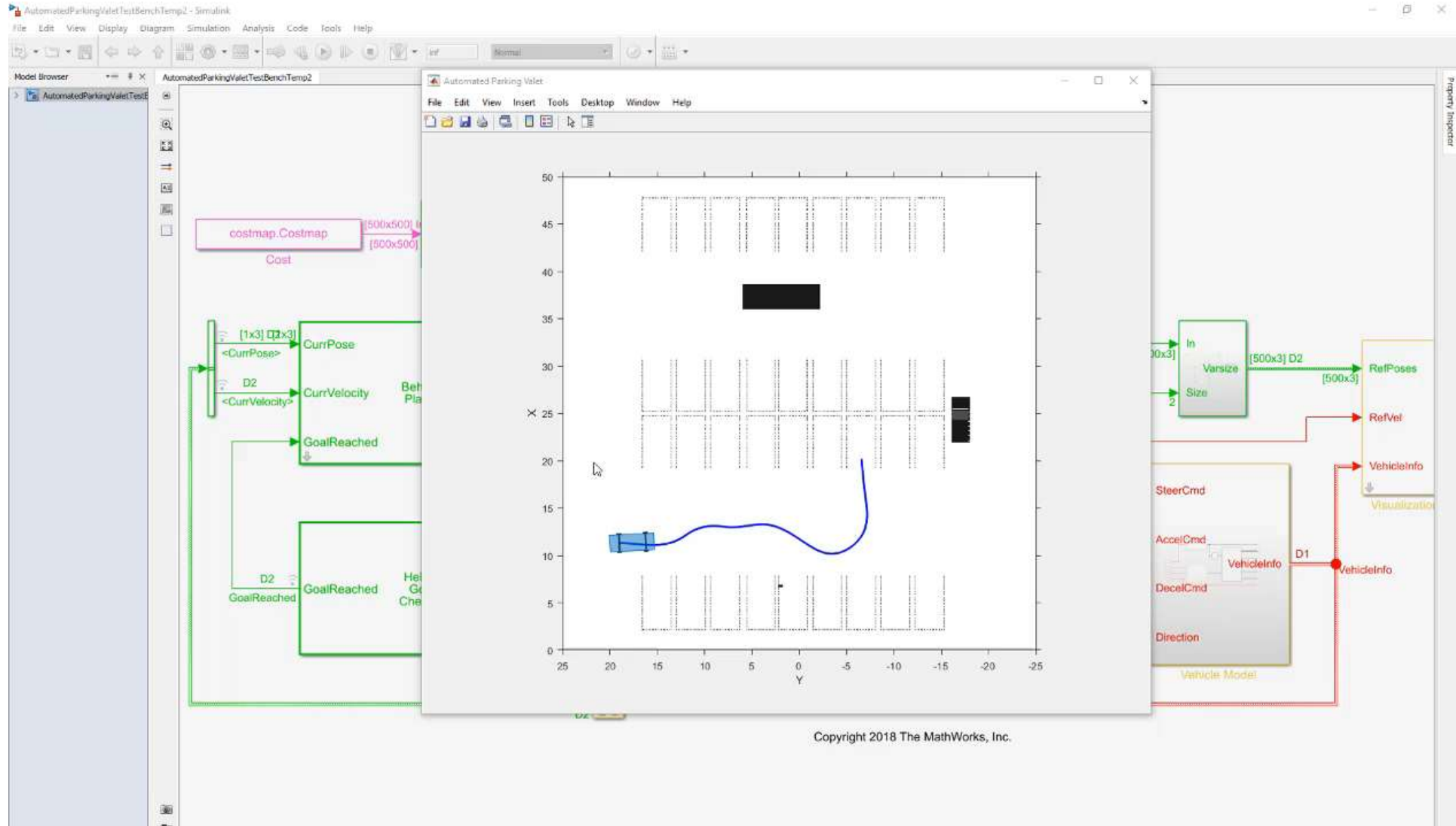


Explore behavior with improved path analyzer



Copyright 2018 The MathWorks, Inc.

Reduce turning radius and speed

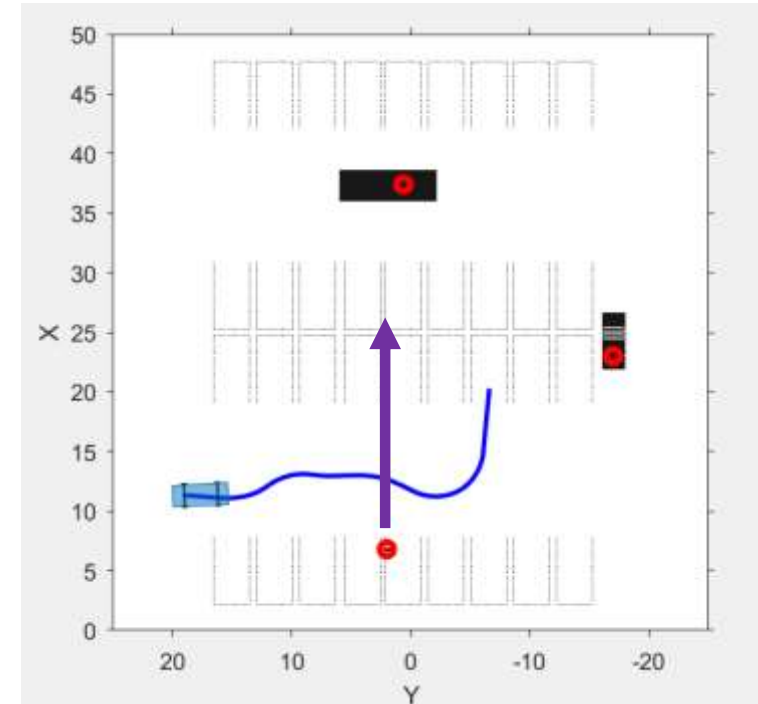
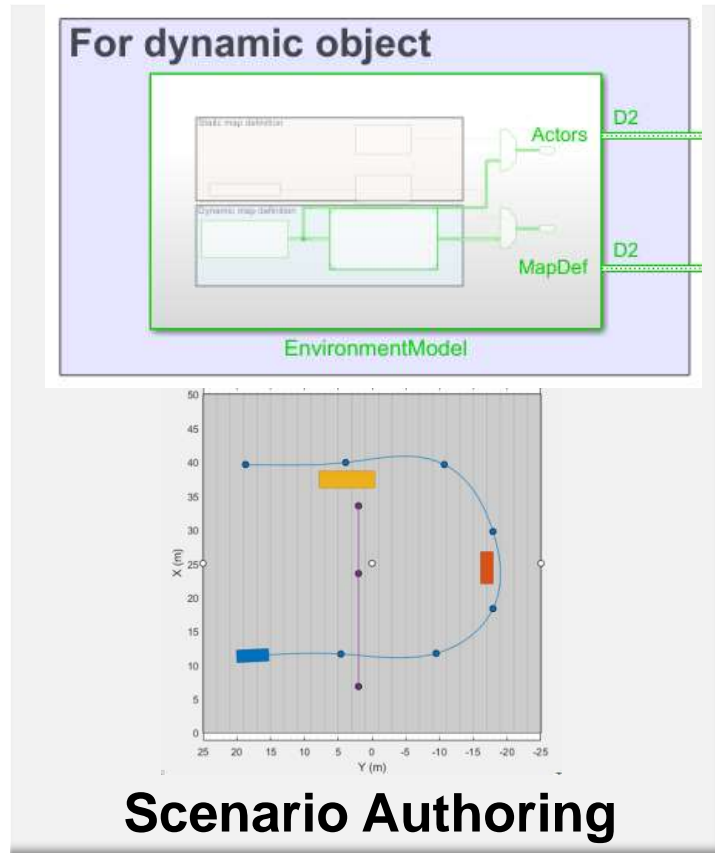
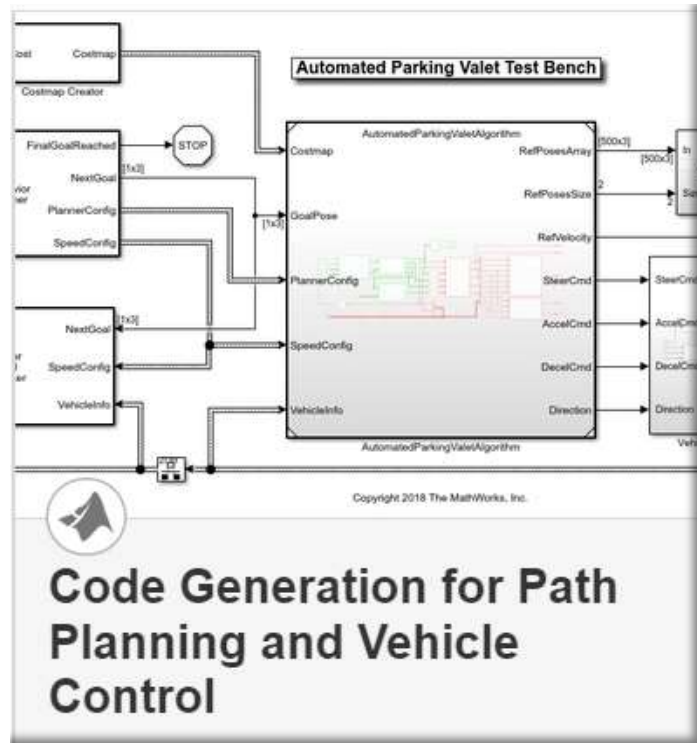


Copyright 2018 The MathWorks, Inc.

Evaluate Path Planner and Controller for Automated Parking

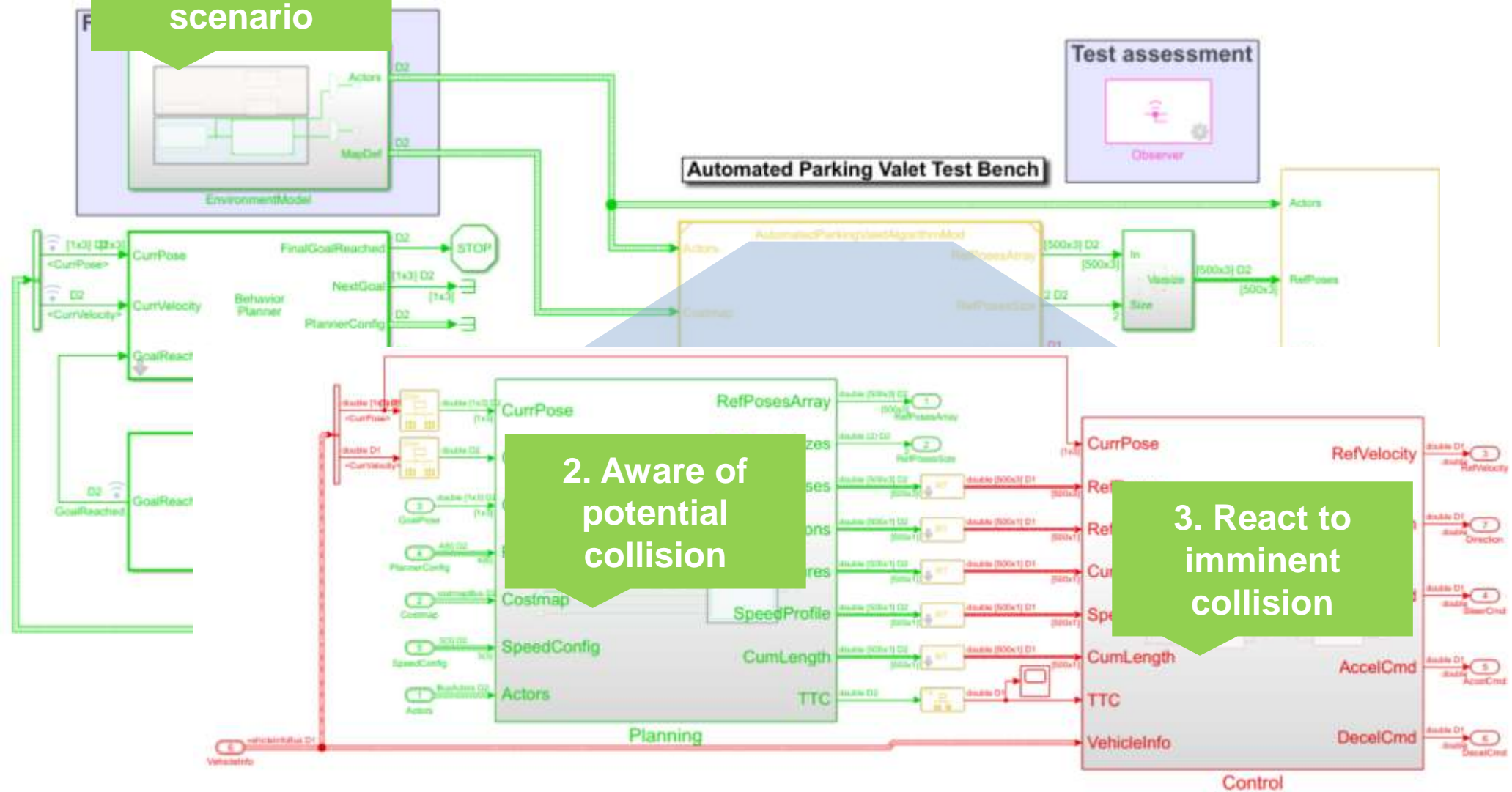
- Explore system robustness with simulation
 - Remove intermedia points
 - Specify different parking maps and spots
 - Identify design issue and improve the design.
- Improve design to handle moving pedestrian
- Test automation for regression tests

Build test bench to test dynamic scenario



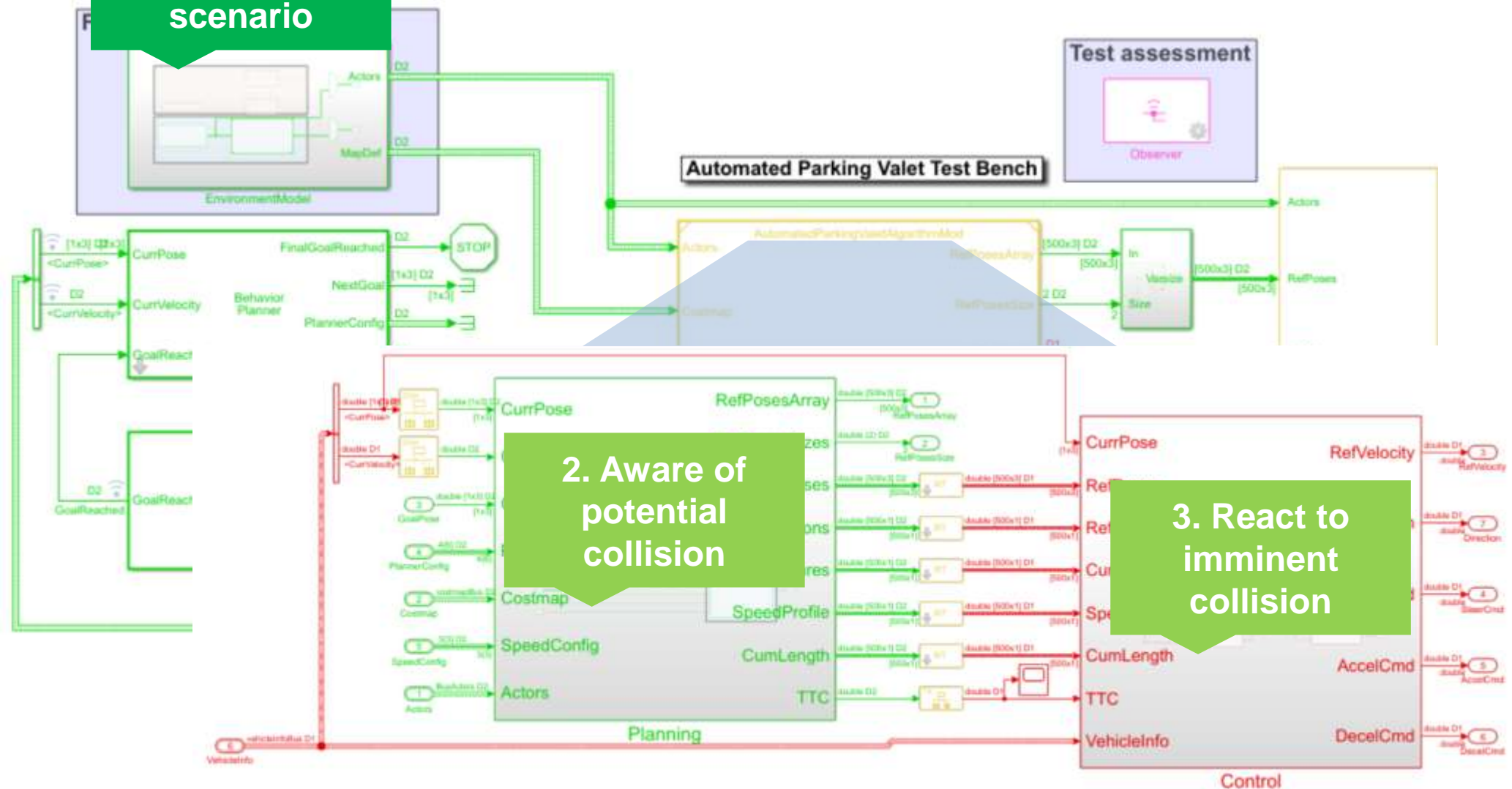
Identify changes to react to pedestrian

1. Update map based on scenario

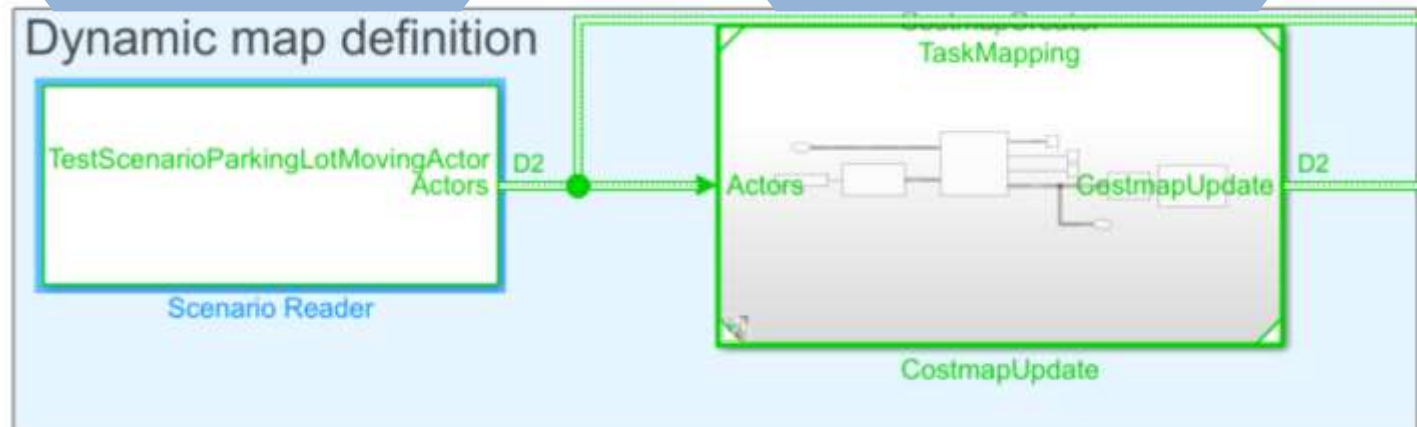
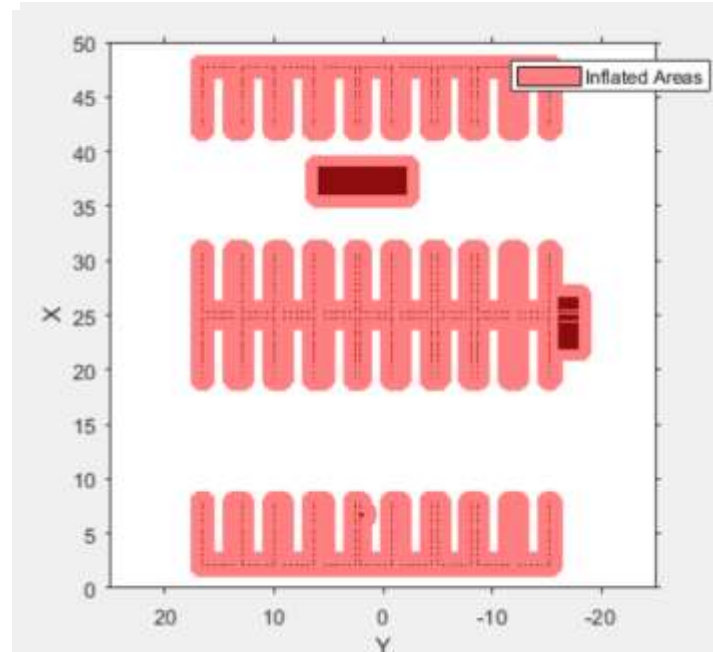
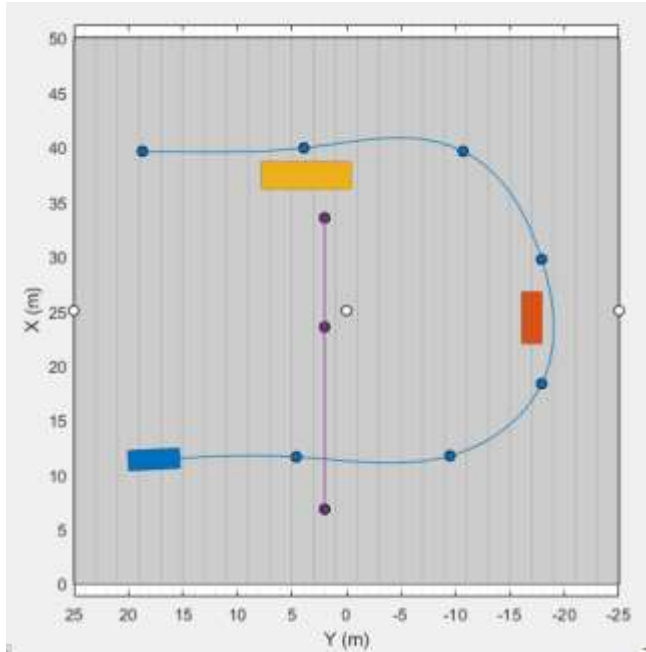


Update map based on scenario

1. Update map based on scenario

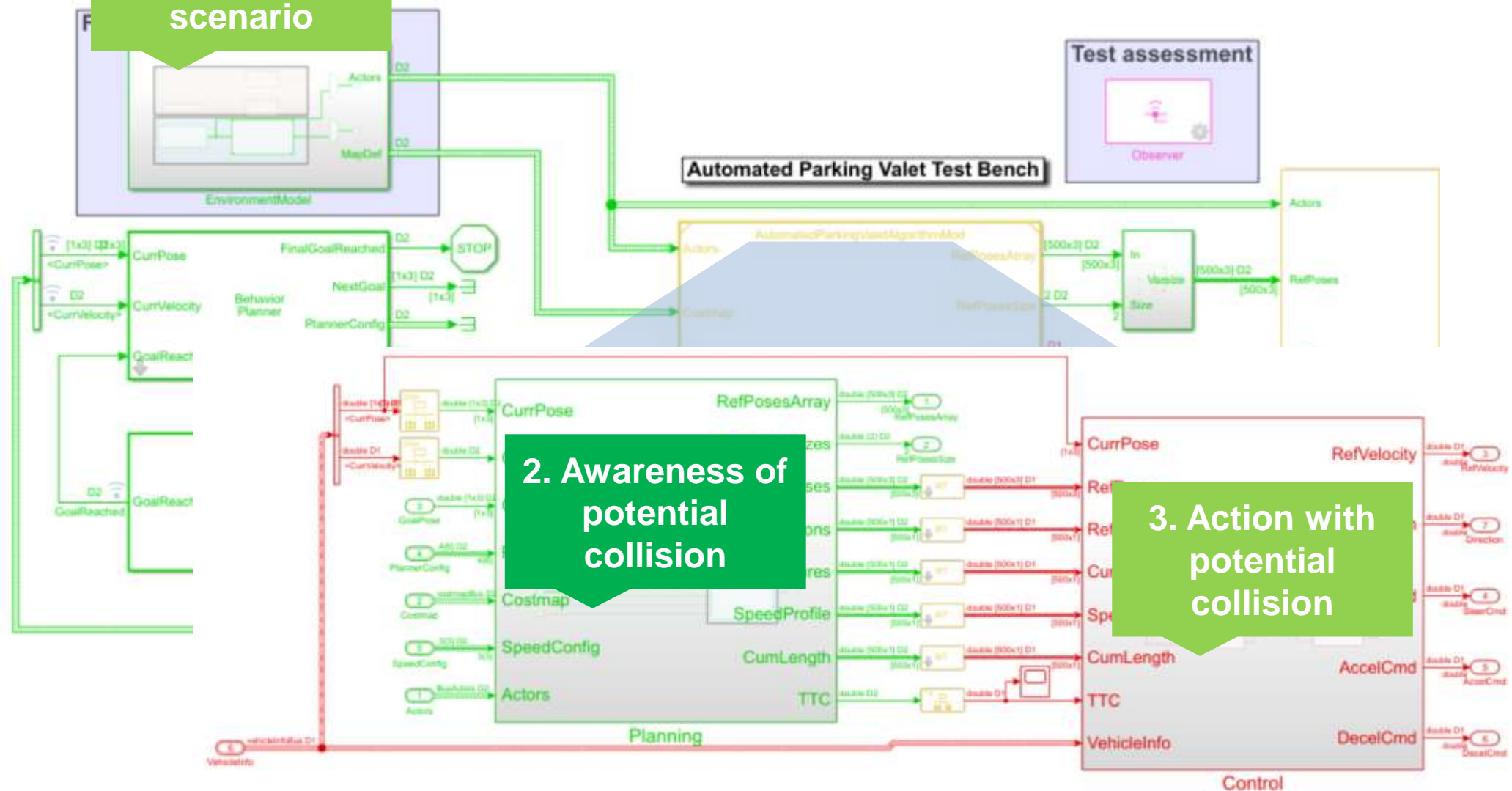


Update map based on scenario



Identify changes to react to pedestrian

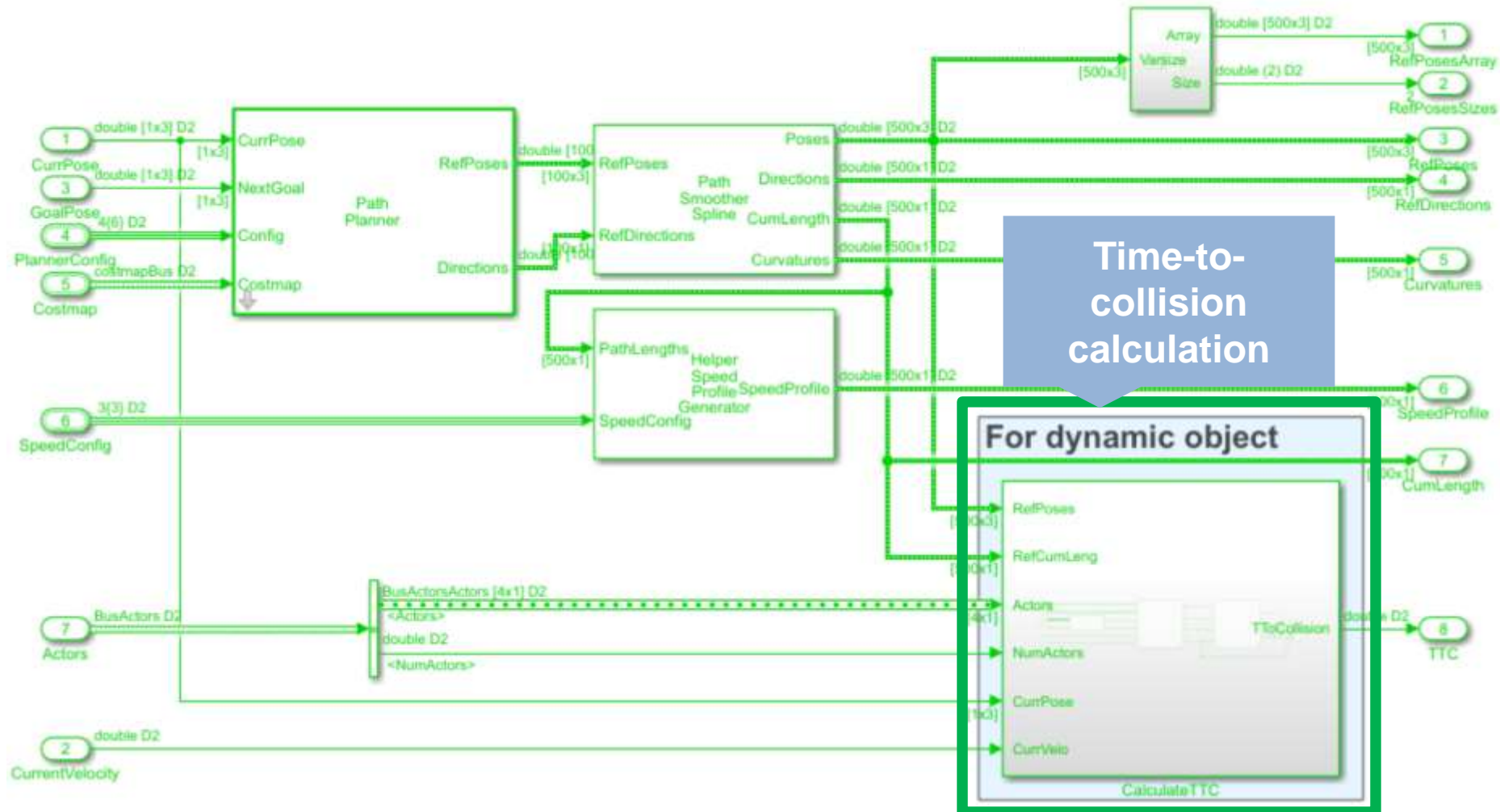
1. Update map based on scenario



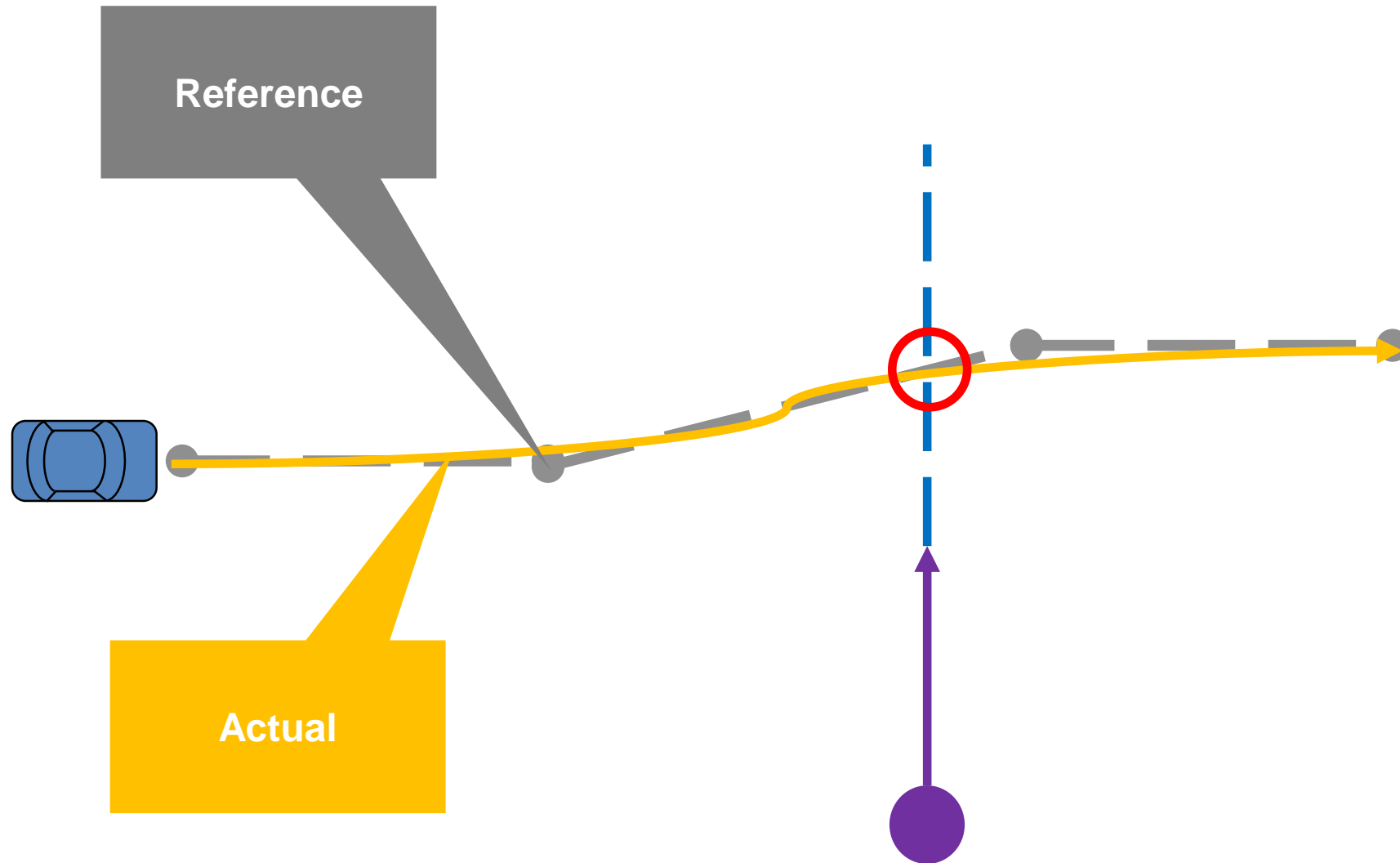
2. Awareness of potential collision

3. Action with potential collision

Awareness of potential collision

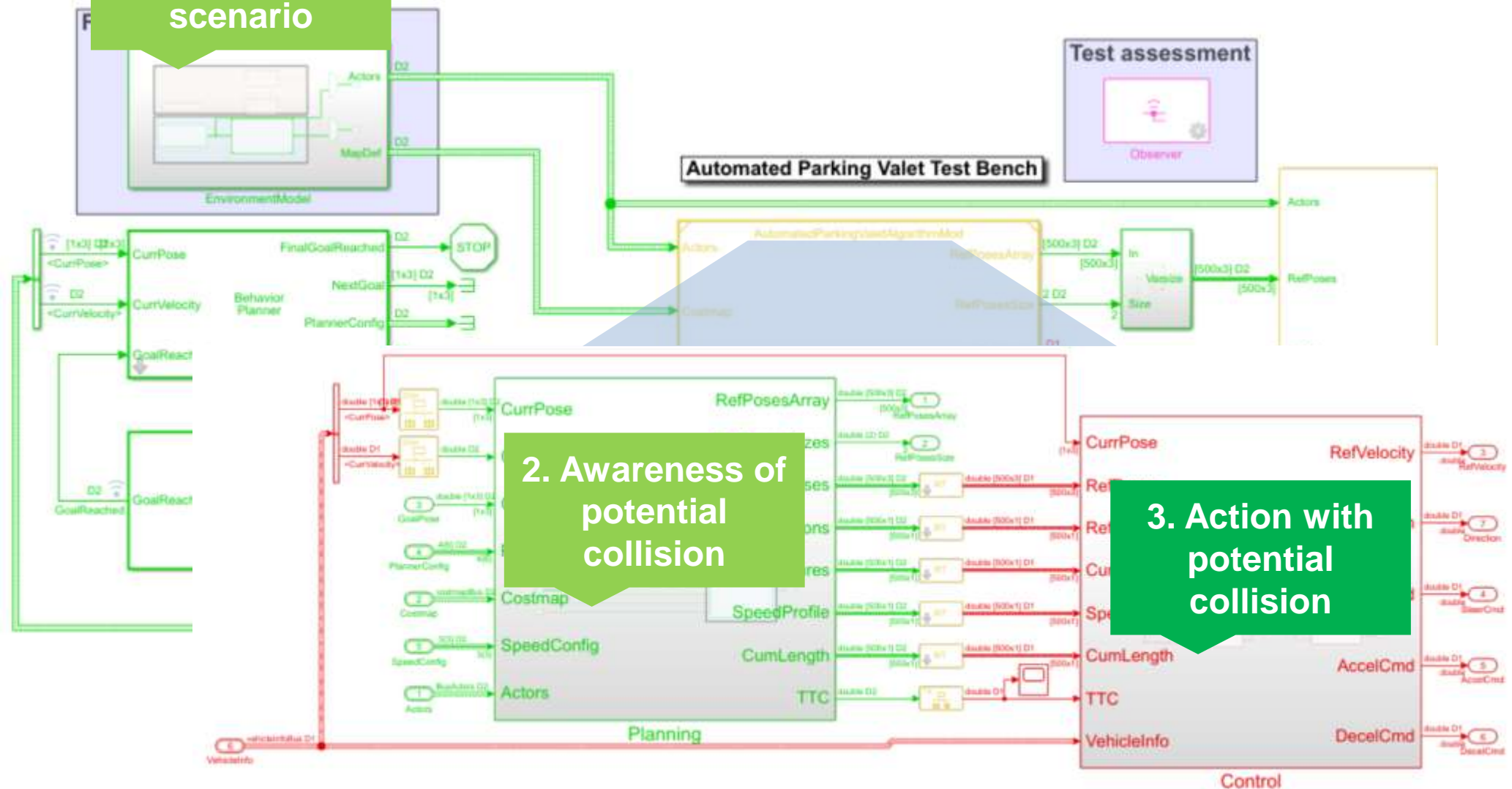


Time-to-collision with known path



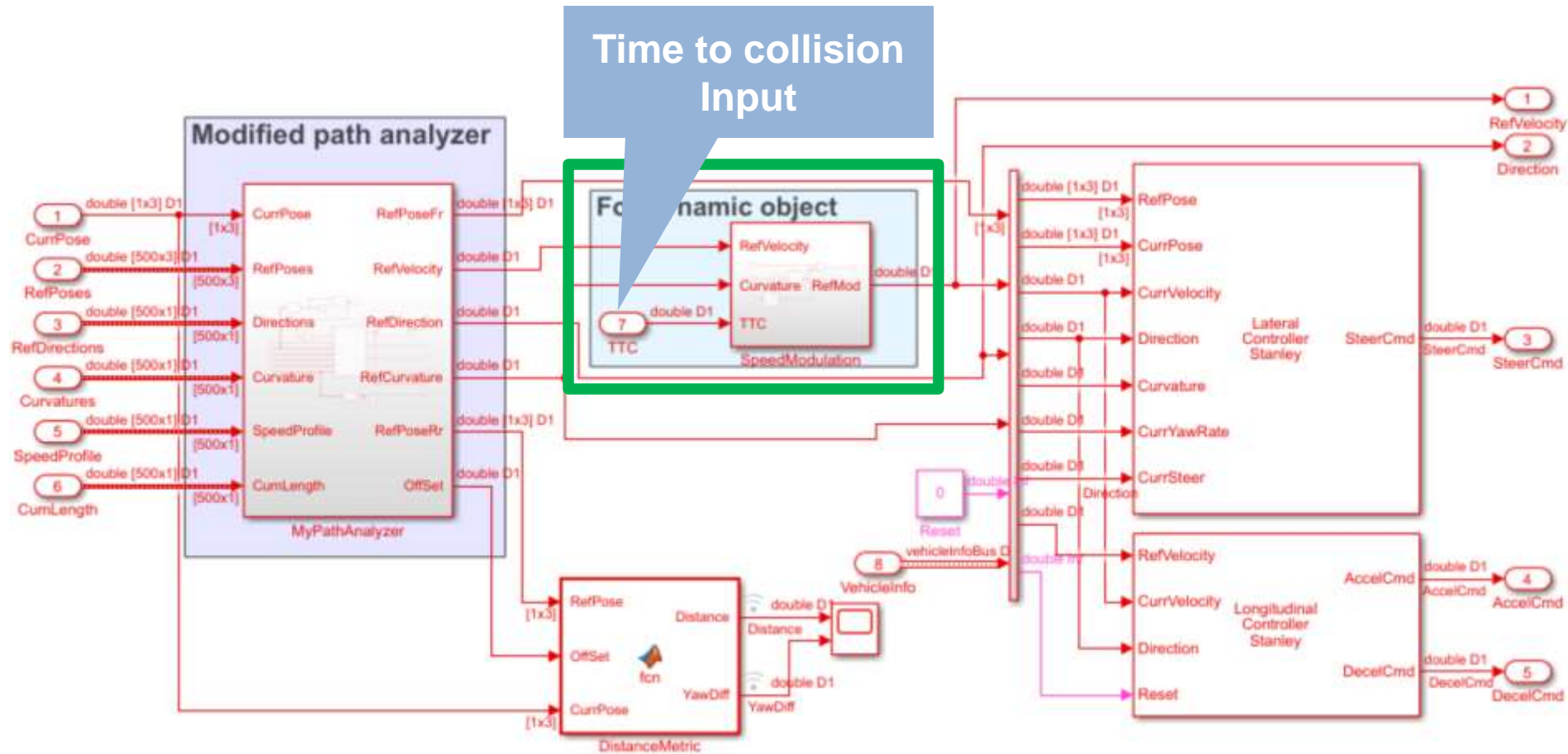
Identify changes to react to pedestrian

1. Update map based on scenario



Action with potential collision

- Reduce speed when TTC is low



Test in dynamic environment

The screenshot displays the Simulink environment for testing an Automated Parking Valet. The main window shows a 2D plot with X and Y axes ranging from 0 to 50. A blue trajectory starts at approximately (18, 12) and moves towards a target area. A black vehicle icon is positioned at the end of the trajectory. The plot is titled "Automated Parking Valet".

On the left, the Model Browser shows the project structure for "AutomatedParkingValetTestBenchMod". Key components include:

- Control
 - Lateral Control
 - Longitudinal Control
 - MyPathAnalyzer
 - SpeedModulation
 - Subsystem
- Planning
 - Varsize Rows Rate T
 - Varsize Rows Rate T
 - Varsize Rows Rate T
 - Varsize Rows Rate T
 - Varsize Rows Rate T
- EnvironmentModel
 - Fixed To Variable Size
 - PlannerConfig
 - Subsystem1
 - Vehicle Model
 - Visualization

The right side of the interface shows a block diagram of the system architecture. Key blocks and their connections include:

- EnvironmentModel**: Contains sub-blocks for "Actors" and "MapDef".
- Behavior Planner**: Receives "CurrPose" and "CurrVelocity" as inputs and outputs "GoalReached".
- Helper Goal Checker**: Receives "GoalReached" and "VehicleInfo" as inputs and outputs "NextGoal".
- Vehicle Model**: Receives "VehicleInfo" and "D1" as inputs and outputs "VehicleInfo".
- Visualization**: Receives "VehicleInfo" and "D1" as inputs.
- Assessment**: Contains an "Observer" block.
- Varsize Size**: Outputs "[500x3] D2" to "RefPoses" and "[500x3]" to "RefVel".

At the bottom, the text "Copyright 2018 The MathWorks, Inc." is visible.

Improve design to handle pedestrians

The screenshot displays a Simulink model titled "Automated Parking Valet" with several interconnected components:

- EnvironmentModel:** A block labeled "For dynamic object" containing "Actors" and "MapDef".
- Behavior Planner:** A central block receiving "CurrPose" (1x3, 0x3) and "CurrVelocity" (D2, 0x3) as input. It outputs "FinalGoalReached", "NextGoal", "PlannerConfig", and "SpeedConfig".
- GoalReached:** A block receiving "GoalReached" and "D2" as input, outputting "GoalReached".
- Helper Goal Checker:** A block receiving "GoalReached" and "D2" as input, outputting "NextGoal", "SpeedConfig", and "VehicleInfo".
- AutomatedParkingValetAlgorithm:** A block at the bottom that receives "D2" and "VehicleInfo" as input and outputs "D2".
- Vehicle Model:** A block receiving "Cmd" and "VehicleInfo" as input, outputting "VehicleInfo".
- Observer:** A block receiving "VehicleInfo" as input, outputting "Observer".
- Visualization:** A block receiving "VehicleInfo" as input, outputting "Visualization".
- Varsize:** A block receiving "In" and "Size" as input, outputting "RefPoses" (500x3, D2) and "RefVel" (500x3).

The plot shows a blue car starting at approximately (18, 12) and moving towards a red dot at (0, 7). A black rectangle representing a pedestrian is located at (0, 37). The plot axes are X (0 to 50) and Y (-25 to 25).

Copyright 2018 The MathWorks, Inc.

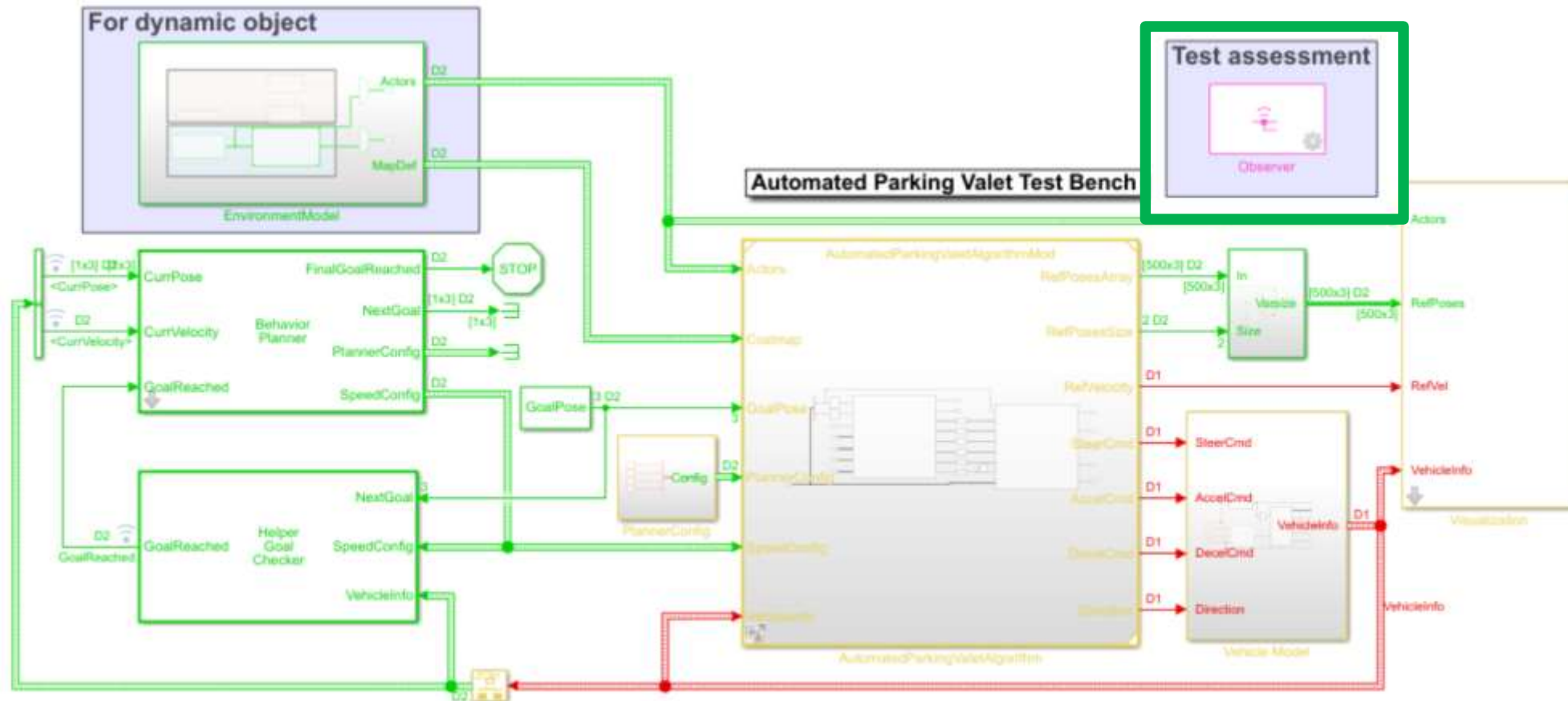
Evaluate Path Planner and Controller for Automated Parking

- Explore system robustness with simulation
 - Remove intermedia points
 - Specify different parking maps and spots
 - Identify design flaws and improve the design.

- Improve design to handle moving pedestrian
 - Add moving pedestrian
 - Create costmap from ground truth
 - Reduce speed based on time-to-collision

- Test automation for regression tests

Automate regression testing

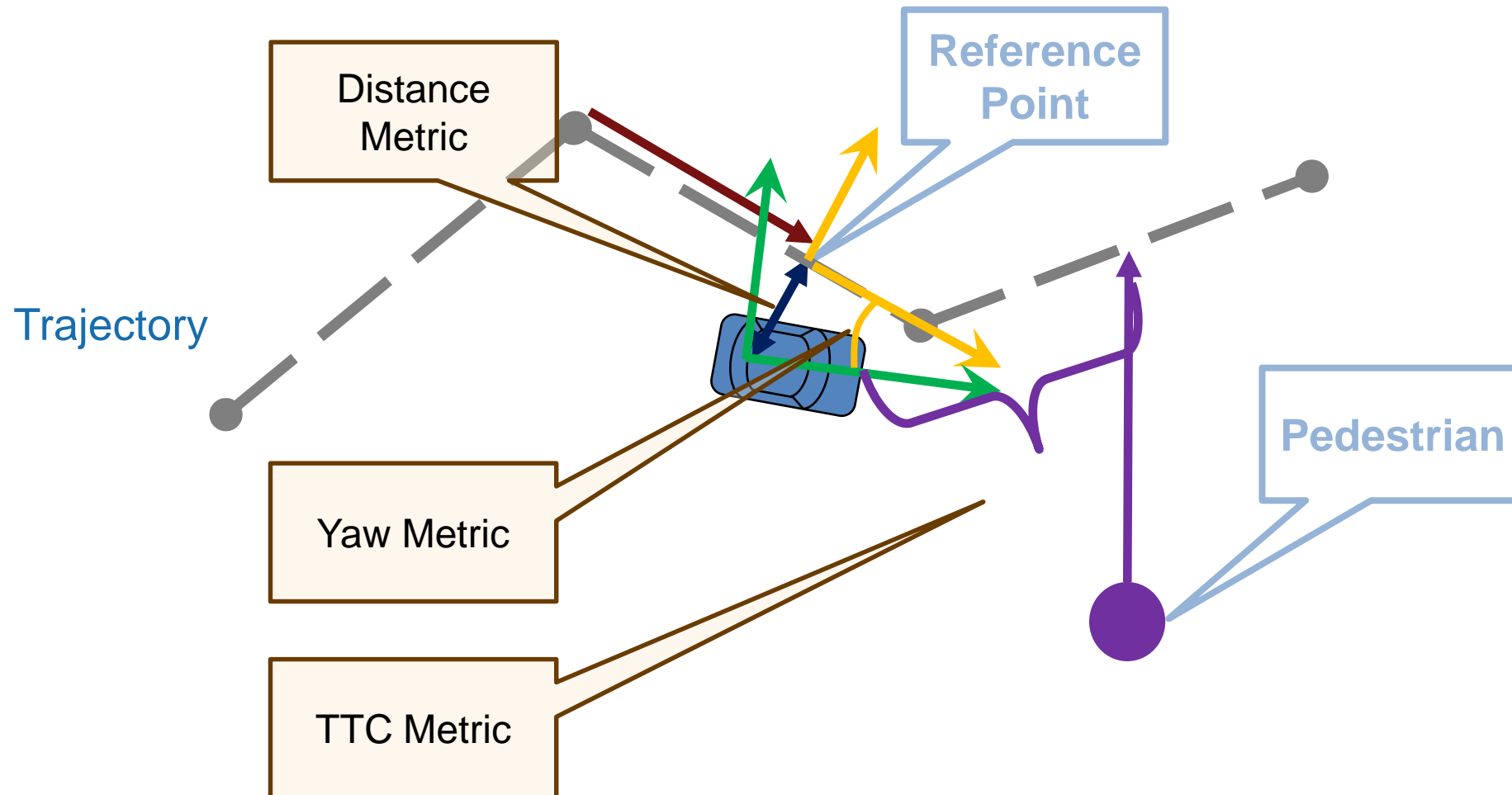


Test assessment metrics

AutomatedParkingValetTestBenchMod_Observer1/Test Assessment - Test Sequence Editor

The screenshot displays the Test Sequence Editor interface. At the top is a toolbar with icons for file operations (save, undo, redo, up arrow), test execution (test, list, stop), and navigation (back, play, forward, next, previous). Below the toolbar is a 'Symbols' panel on the left, which is divided into 'Input', 'Output', 'Local', and 'Constant' sections. The 'Input' section contains three items: 1. DistanceMetricAccepta, 2. YawDiffMetricAcceptab, and 3. IsTrajValid. The main workspace on the right is titled 'Step' and contains a single step named 'VerifyMetrics'. This step is expanded to show the code: `VerifyActualVSRefMetrics` followed by three verification statements: `verify(duration(DistanceMetricAcceptable==false)<0.5)`, `verify(duration(YawDiffMetricAcceptable==false)<0.5)`, and `verify(IsTrajValid==true)`.

Test assessment metrics

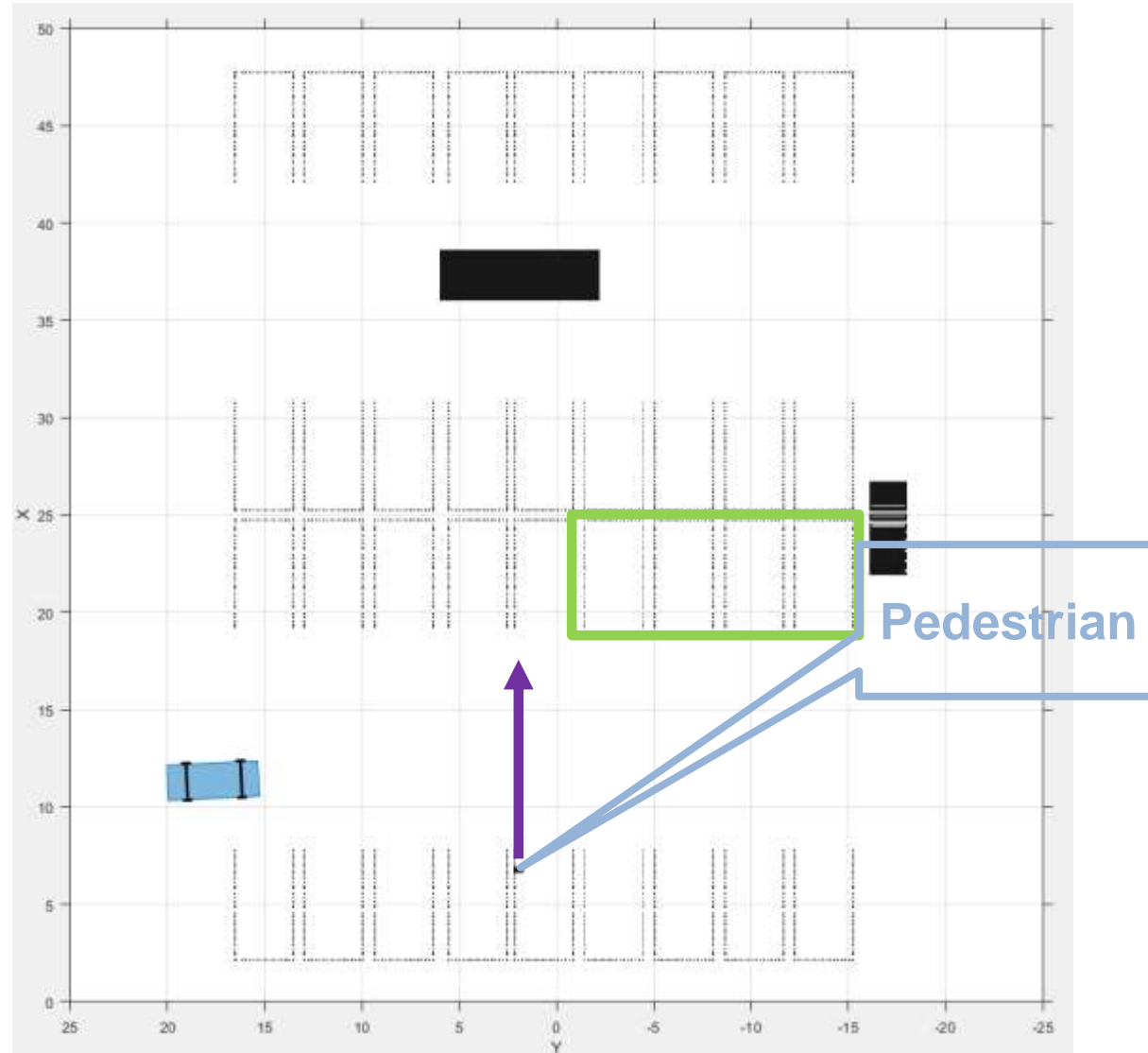


Test iteration

- Tests definition and test management

The screenshot displays the MathWorks Test Browser interface. At the top, a ribbon menu is organized into five sections: FILE (New, Open, Save), EDIT (Cut, Copy, Paste, Delete), RUN (Run, Stop, Debug, Parallel), RESULTS (Report, Visualize, Highlight in Model, Export), and ENVIRONMENT (Import, Preferences). Below the ribbon, the interface is split into two main panes. The left pane, titled 'Test Browser', contains a search filter 'Filter tests by name or tags, e.g. tags: test' and a tree view of test cases. The tree view shows a folder 'TestPathPlanningMAC' containing two sub-folders: 'Dynamic Test Cases' (highlighted with a yellow box) and 'Static Test Cases'. Under 'Dynamic Test Cases', there are two test cases: 'Iterate on Different Parking Spot Gain 2' and 'Iterate on Different Parking Spot Gain 4'. Under 'Static Test Cases', there is one test case: 'Iterate on Different Parking Spot'. The right pane, titled 'Dynamic Test Cases', shows the breadcrumb 'TestPathPlanningMAC » Dynamic Test Cases', the text 'Test Suite', a dropdown menu for 'Select releases for simulation:' set to 'Current', and three expandable sections: 'TAGS', 'DESCRIPTION', and 'REQUIREMENTS'.

Testing multiple goal poses



Automate regression testing

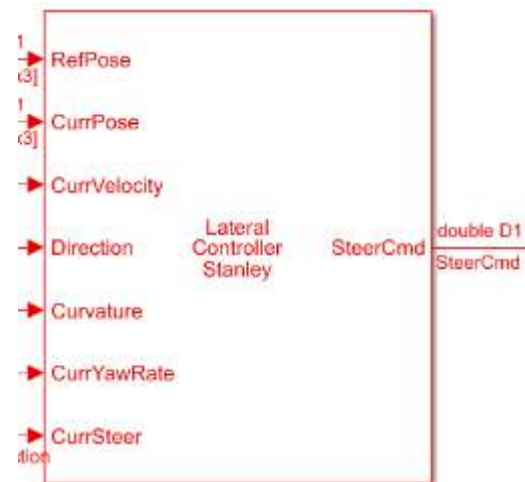
- Use test manager to inspect reason of failed test.

Iterate on Different Parking Spot	2 ✓ 2 ✗
Iteration 1	✓
Iteration 2	✗
Iteration 3	✓
Iteration 4	✗

Iterate on Different Parking Spot	2 ✓ 2 ✗
Iteration 1	✓
Iteration 2	✗
Verify Statements	✗
Test Assessment/.../...	✓
Test Assessment/.../...	✗
Test Assessment/.../...	✓
Assertion	✓
Assertion1	✓
Sim Output (AutomatedPar	
Iteration 3	✓
Iteration 4	✗

Yaw Metric

Automate regression testing



Iterate on Different Parking Spc 4	
Iteration 1	✓
Iteration 2	✓
Iteration 3	✓
Iteration 4	✓

Controller Settings

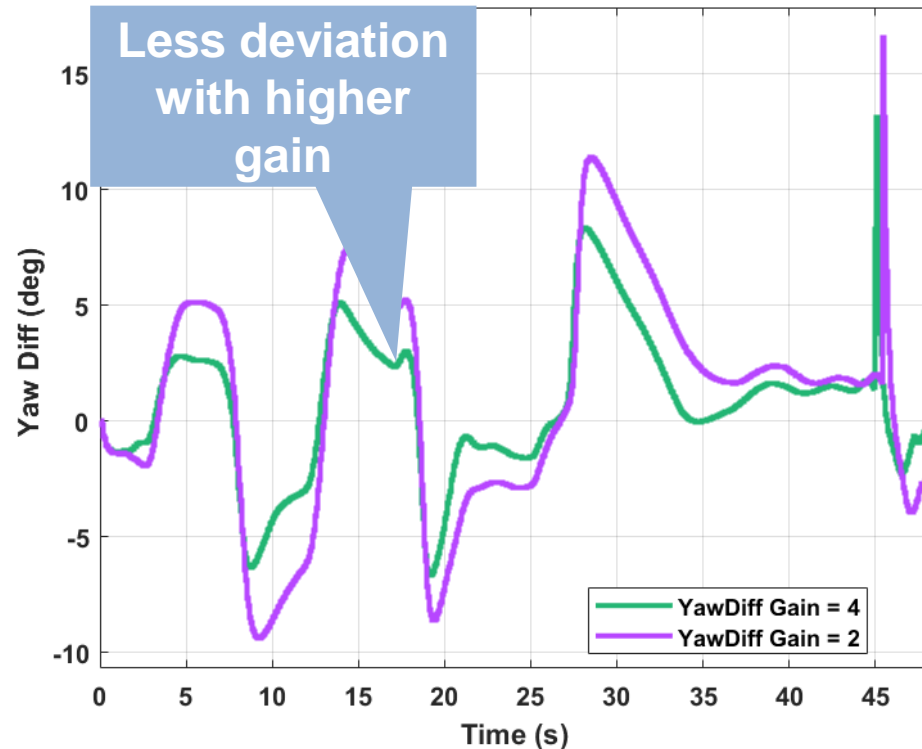
Vehicle model: Dynamic bicycle model

Position gain of forward motion: LateralControllerPositionGainForward

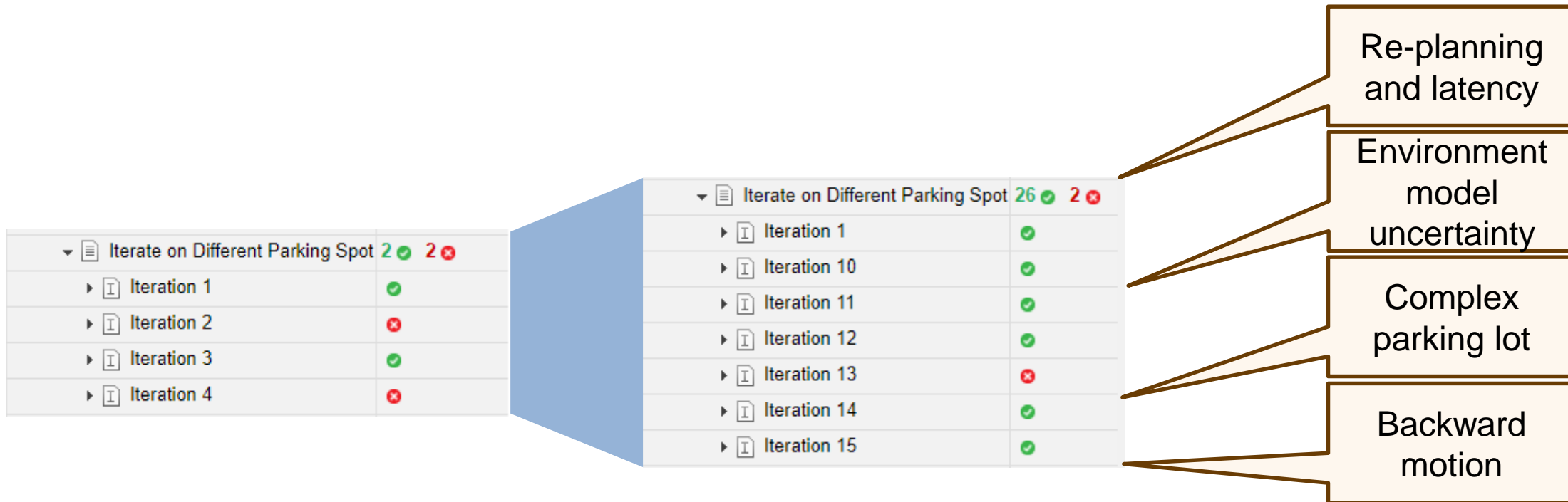
Position gain of reverse motion: 2.5

Yaw rate feedback gain: 0.25

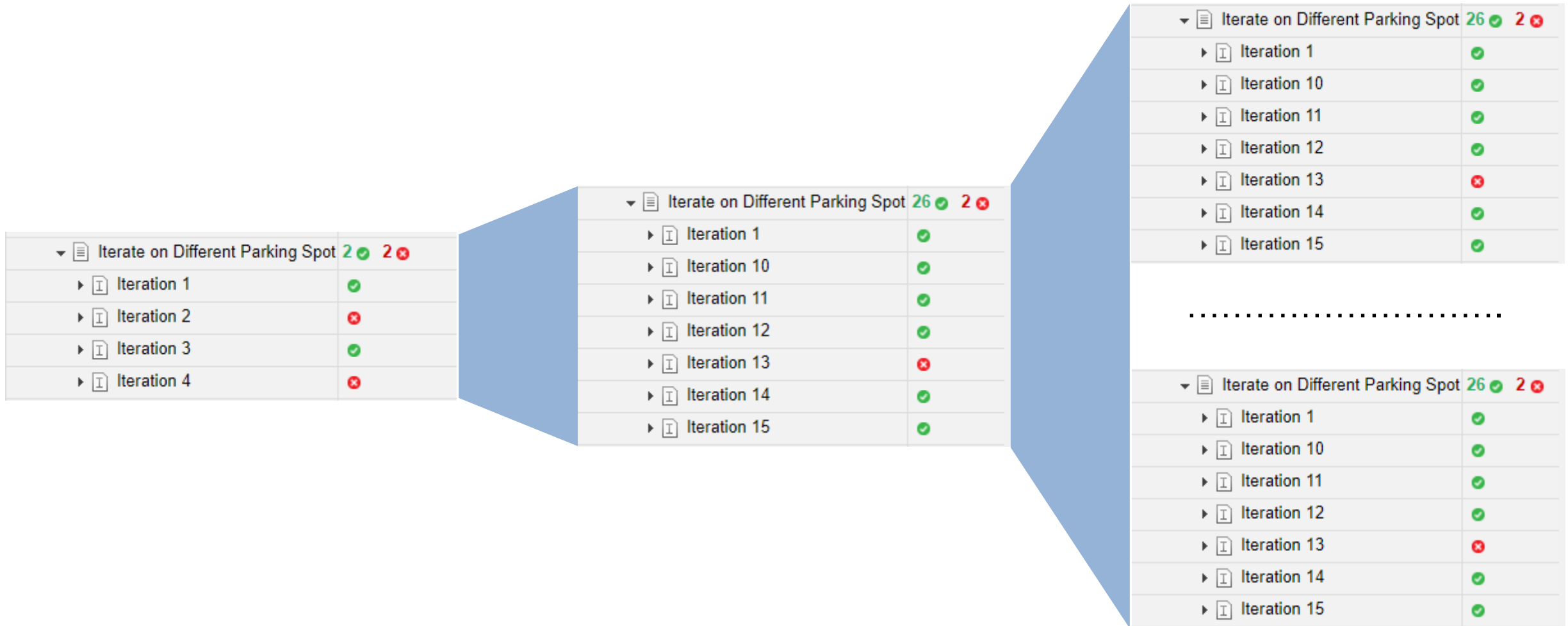
Steering angle feedback gain: 0.2



Automate regression testing



Automate regression testing



Evaluate Path Planner and Controller for Automated Parking

- Explore system robustness with simulation
 - Remove intermediate points
 - Specify different parking maps and spots
 - Identify design flaws and improve the design
- Improve design to handle moving pedestrians
 - Add moving pedestrian
 - Create costmap from ground truth
 - Reduce speed based on time-to-collision
- Test automation for regression tests
 - Add metrics for planner and controller
 - Add test case definition/ management