

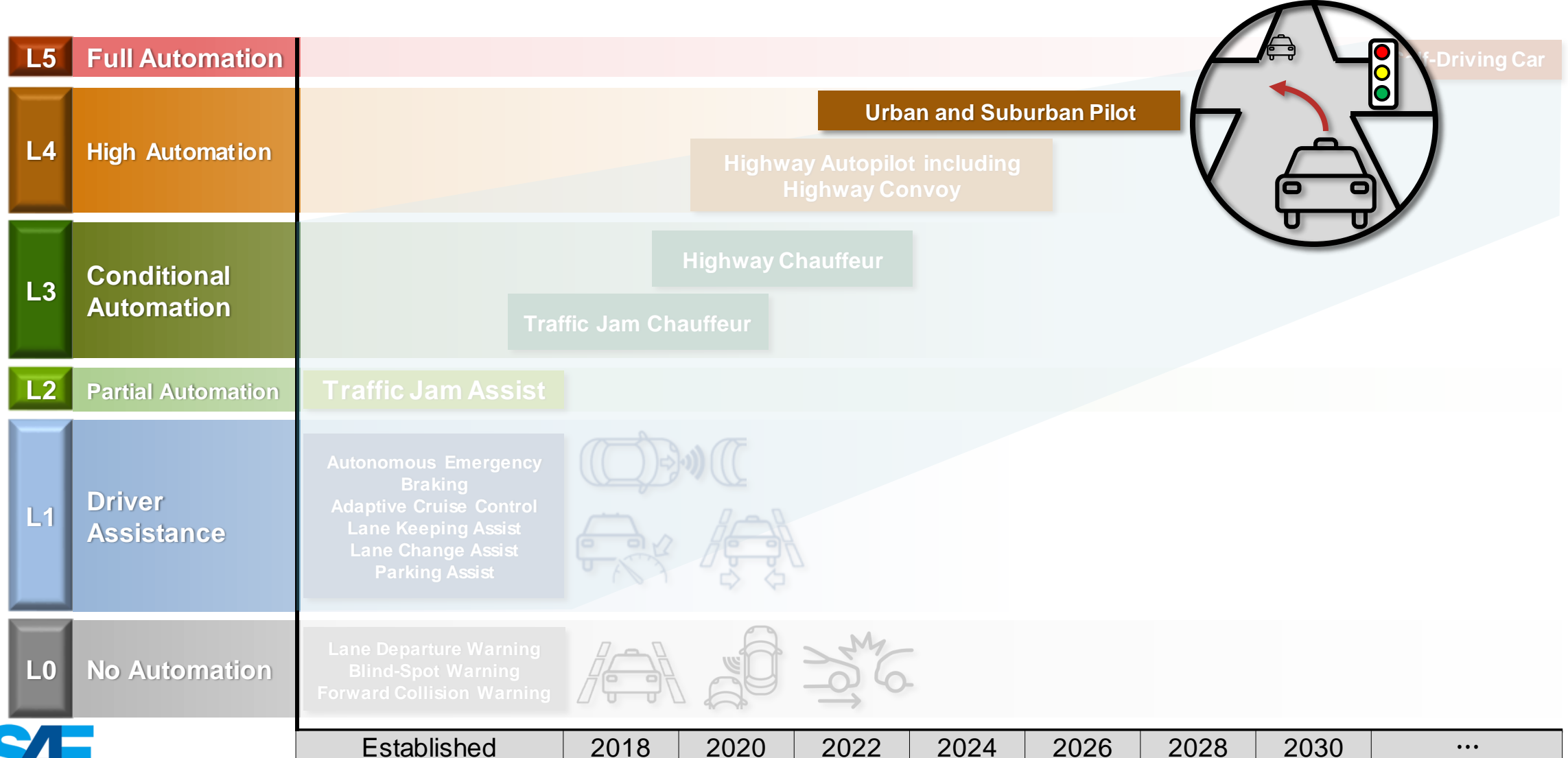
MathWorks
**AUTOMOTIVE
CONFERENCE 2024**
North America

Automated Driving in the Urban Environment with RoadRunner Scenario

Seo-Wook Park, MathWorks

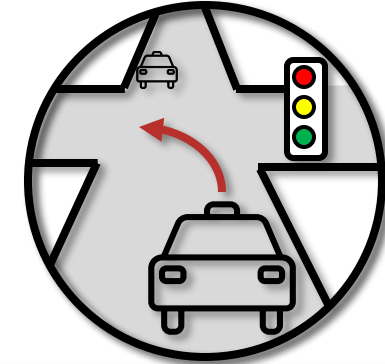


Evolution of Automated driving technology

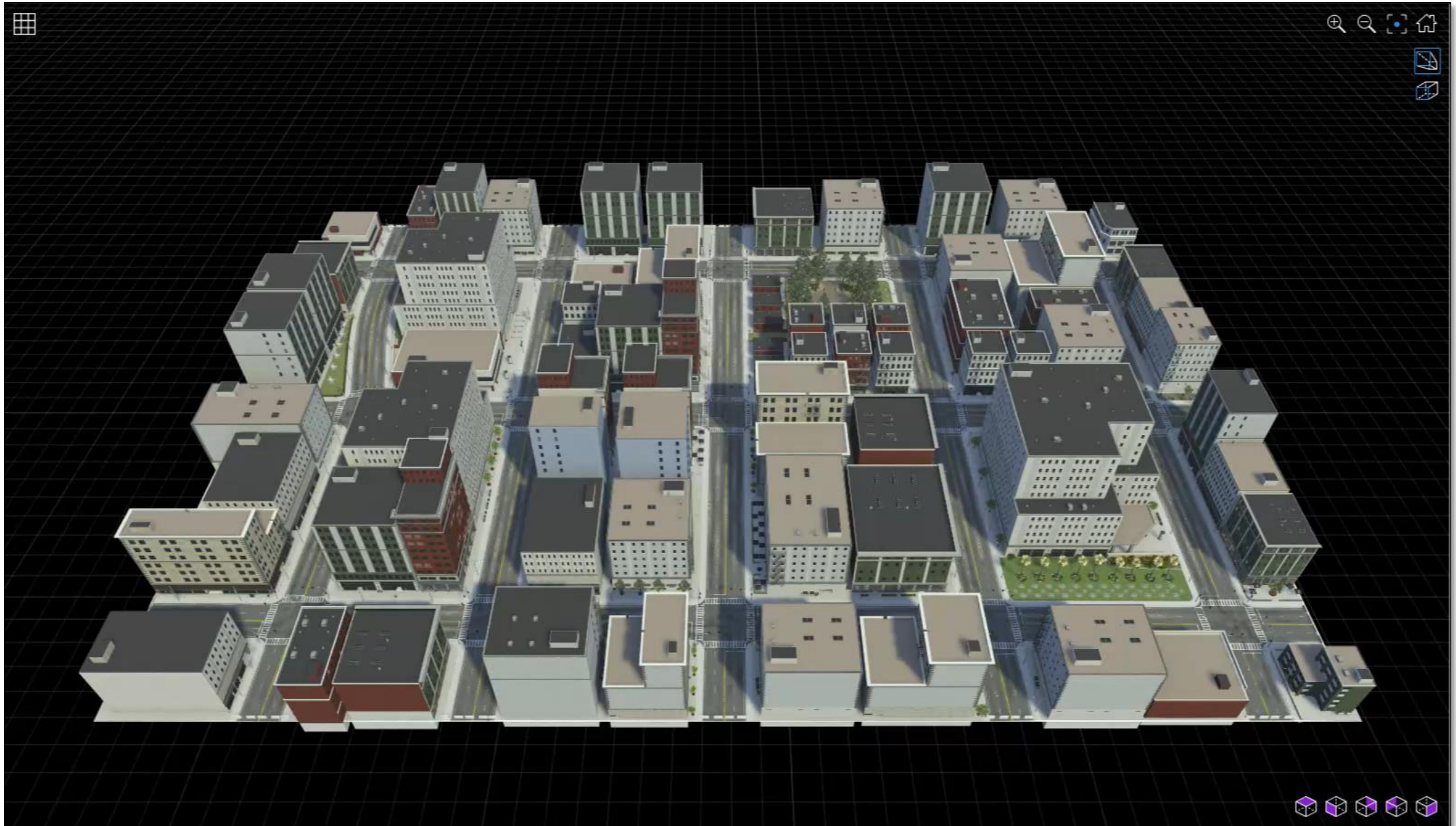


Challenges for Automated Driving in the Urban Environment

- Complex road geometry and maneuvering in the intersections with traffic signals
- Sensor and perception limitations
 - Limited sensor FoV, Sensor occlusion
- Communication and connectivity issues
 - GPS signal loss, V2X communication
- Infrastructure, regulatory, and standard considerations
 - V2X standards SAE J2735, ETSI TS 101 539, T/CSAE 53-2020
 - DSRC, Cellular-V2X, LTE C-V2X, 5G C-V2X
 - AV regulations



Traffic light follower at urban intersections



Workflow



Note) V2X: Vehicle-To-Everything, SPaT: Signal Phase and Timing

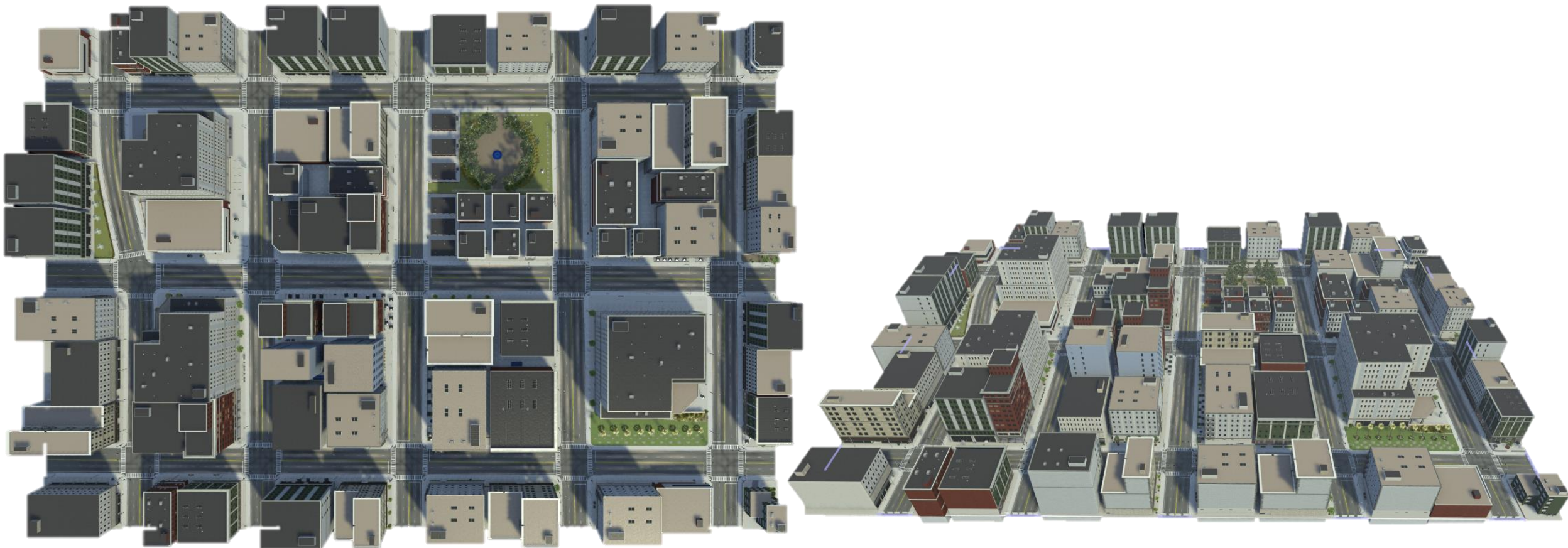
Workflow





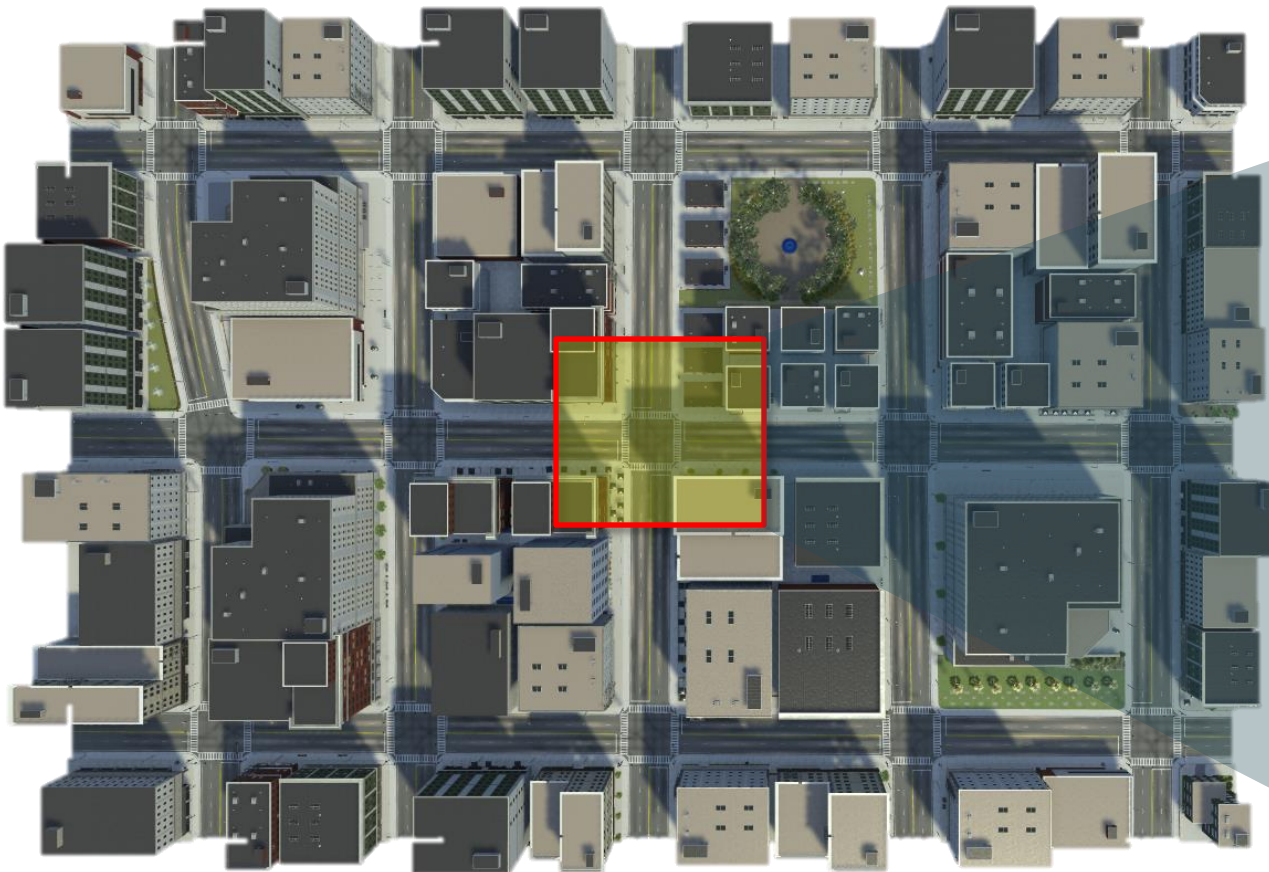
Create a complex urban scene

- 3D environment of a [US City Block](#) containing 15 intersections with traffic lights.
- All roads in the scene are two-way roads with four lanes.

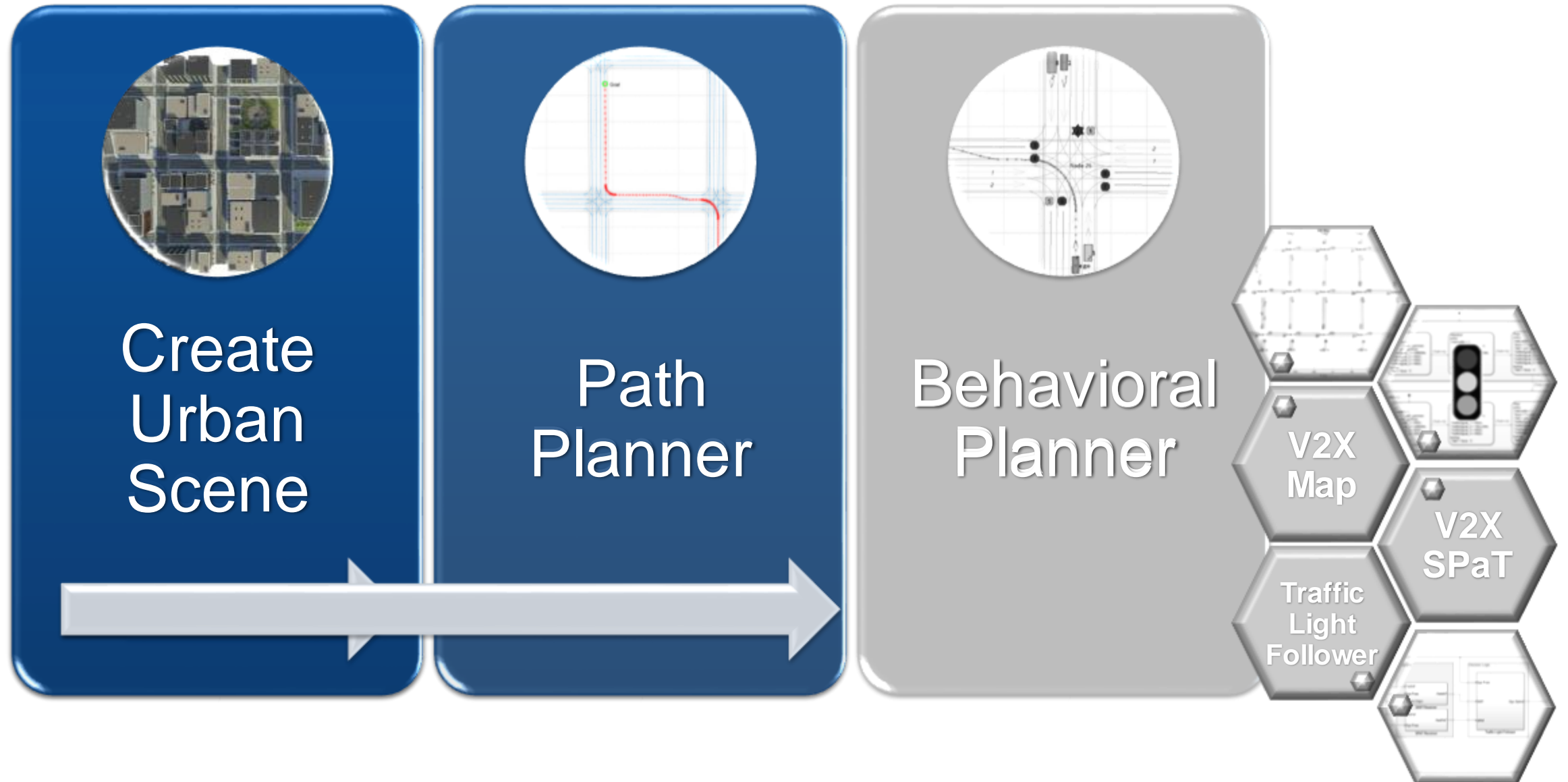


Create a complex urban scene

- 3D environment of a [US City Block](#) containing 15 intersections with traffic lights.
- All roads in the scene are two-way roads with four lanes.



Workflow



Path Planner

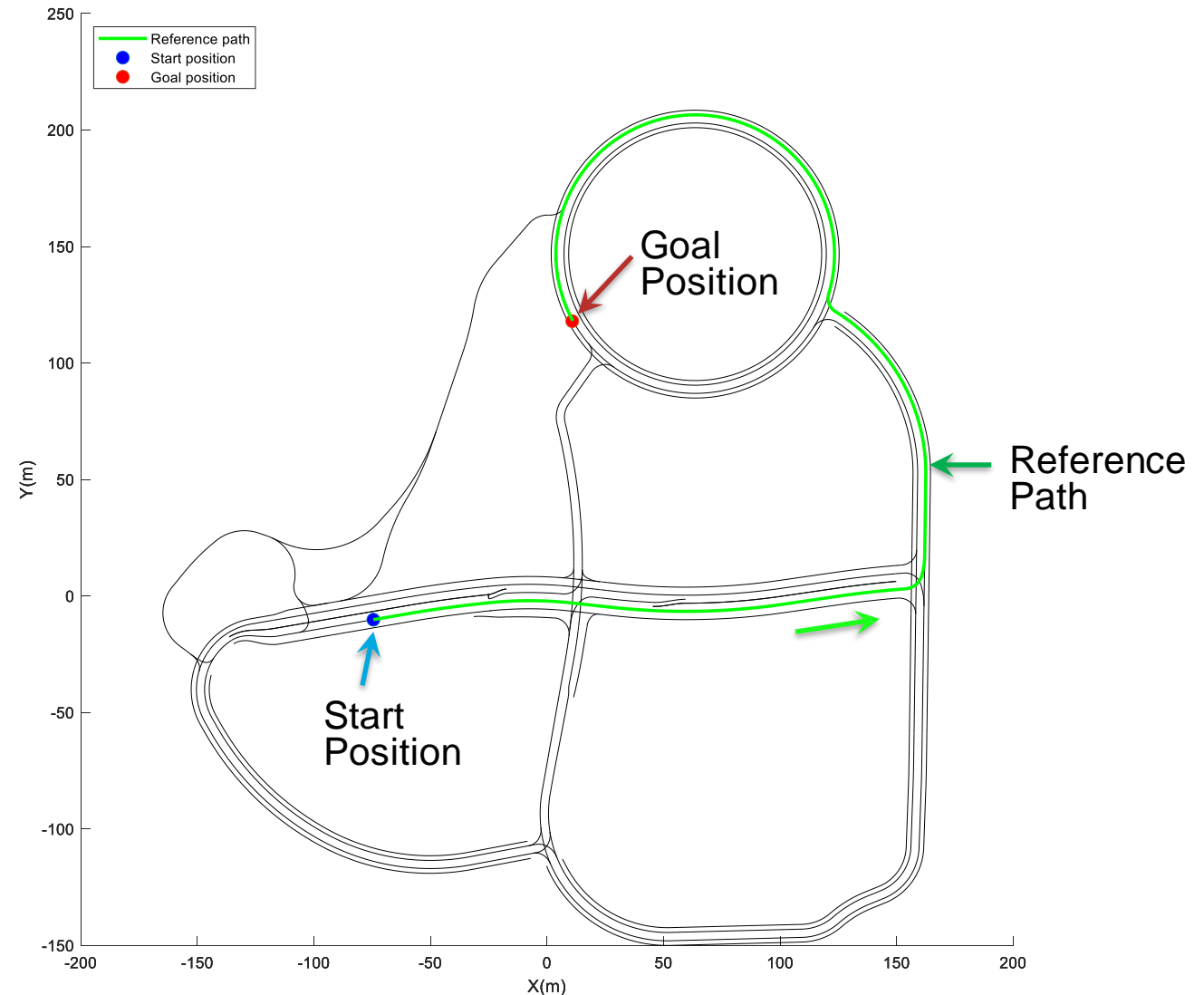


Lane-Level Path Planning with RoadRunner Scenario

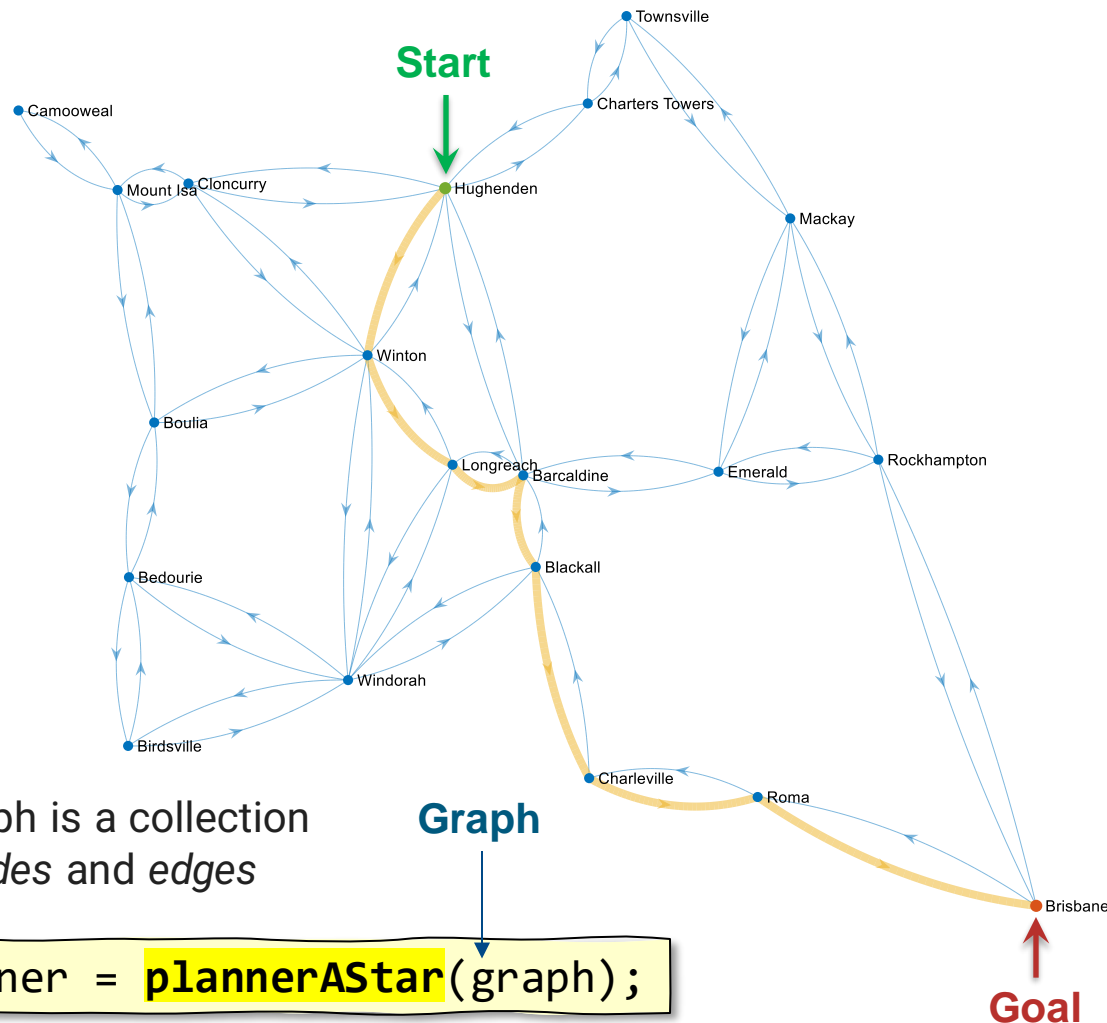
Design a lane-level path planner in MATLAB[®] and cosimulate with RoadRunner Scenario.

- *Automated Driving Toolbox*
- *RoadRunner Scenario*
- *Navigation Toolbox*

- Finds the shortest path between the start position and the goal position.

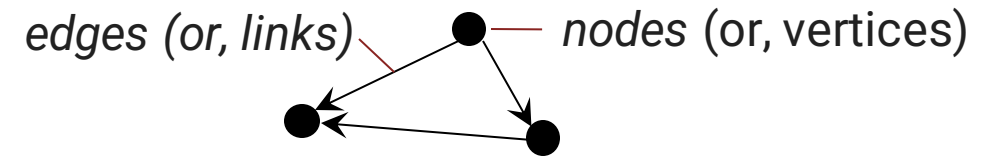


Path Planner



- Finds the shortest path between the start position and the goal position.

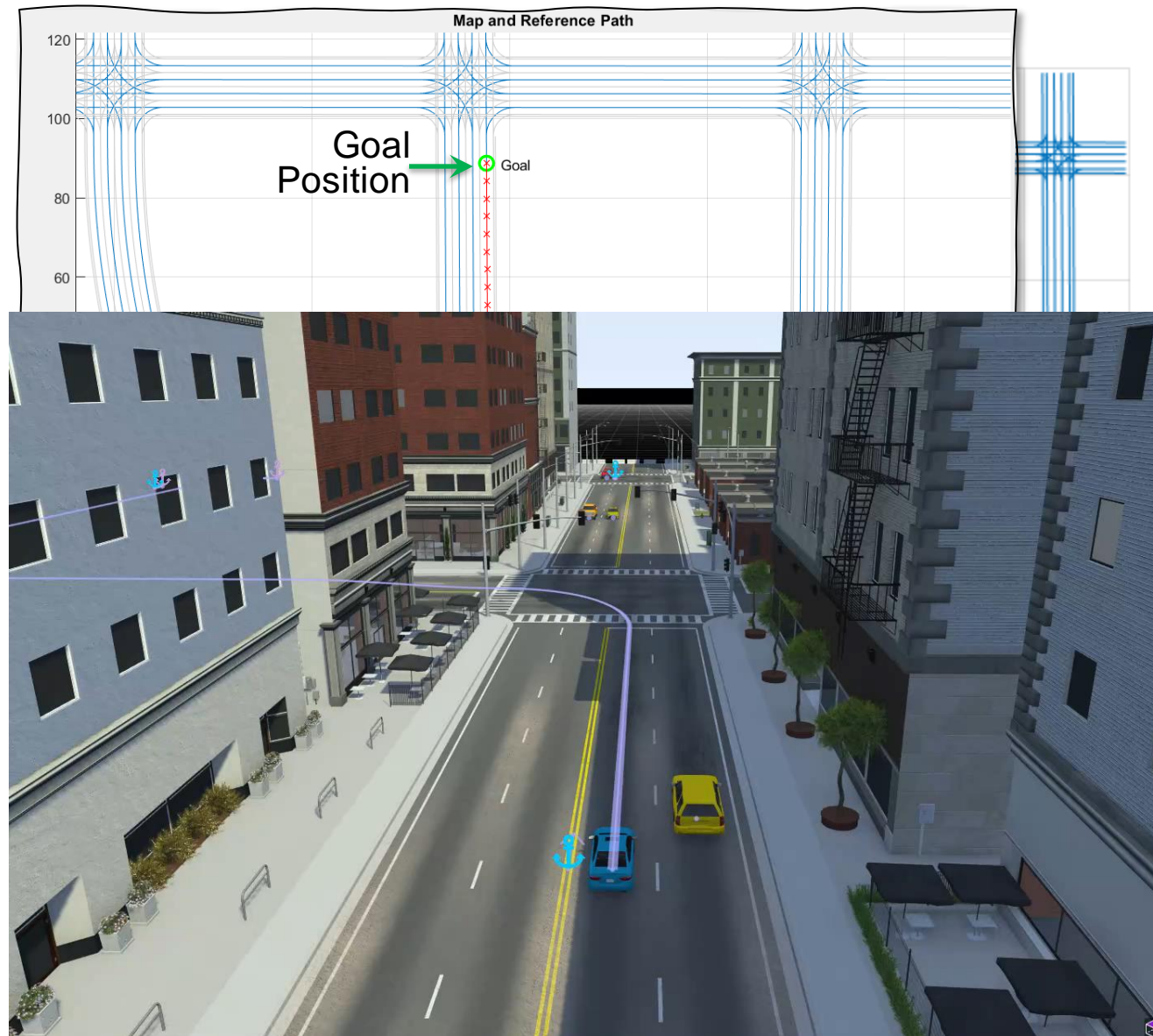
1. Create the nodes and edges table from RoadRunnerHD map



2. Create a graph data structure using [navGraph](#) from *Navigation Toolbox*
3. Create an [A* planner](#) from the navGraph object
4. Use the planPath method to find the shortest path between the start position and the goal position

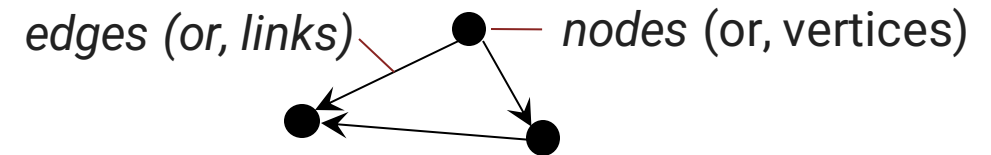
```
refPath = planPath(planner, graph, Start, Goal);
```

Path Planner



- Finds the shortest path between the start position and the goal position.

1. Create the nodes and edges table from RoadRunnerHD map



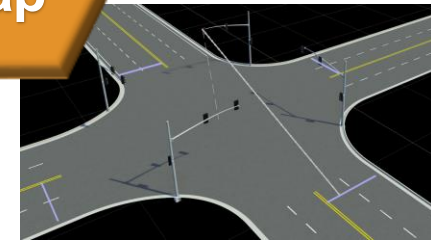
2. Create a graph data structure using [navGraph](#)
3. Create an [A* planner](#) from the navGraph object
4. Use the planPath method to find the shortest path between the start position and the goal position

```
refPath = planPath(planner, graph, Start, Goal);
```

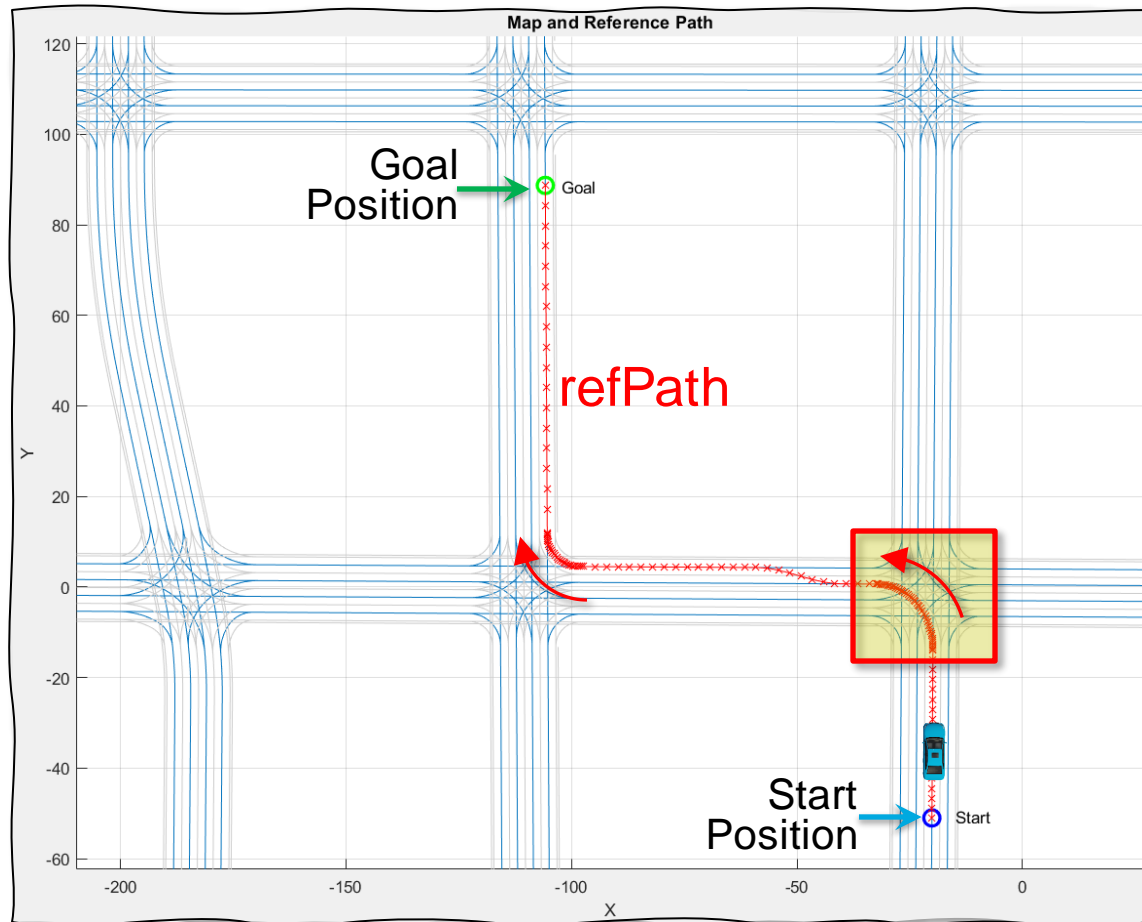
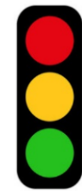
Workflow



Behavioral Planer at intersections



- How to detect an approaching intersection?
- How to get current state and timing of the associated traffic light?
- How to identify a maneuver at the intersection to follow the reference path – proceed straight, turn left, or turn right?
- How to decide to “go” or “stop” at the intersection, based on the traffic light’s state and timing?

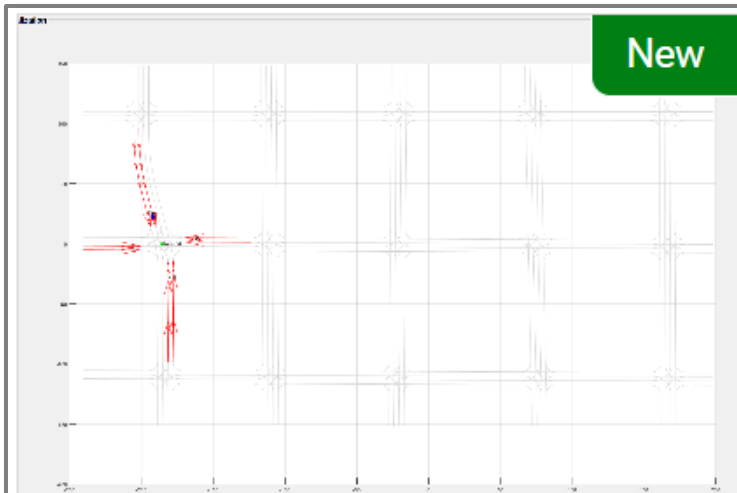


Workflow



Note) V2X: Vehicle-To-Everything, SPaT: Signal Phase and Timing

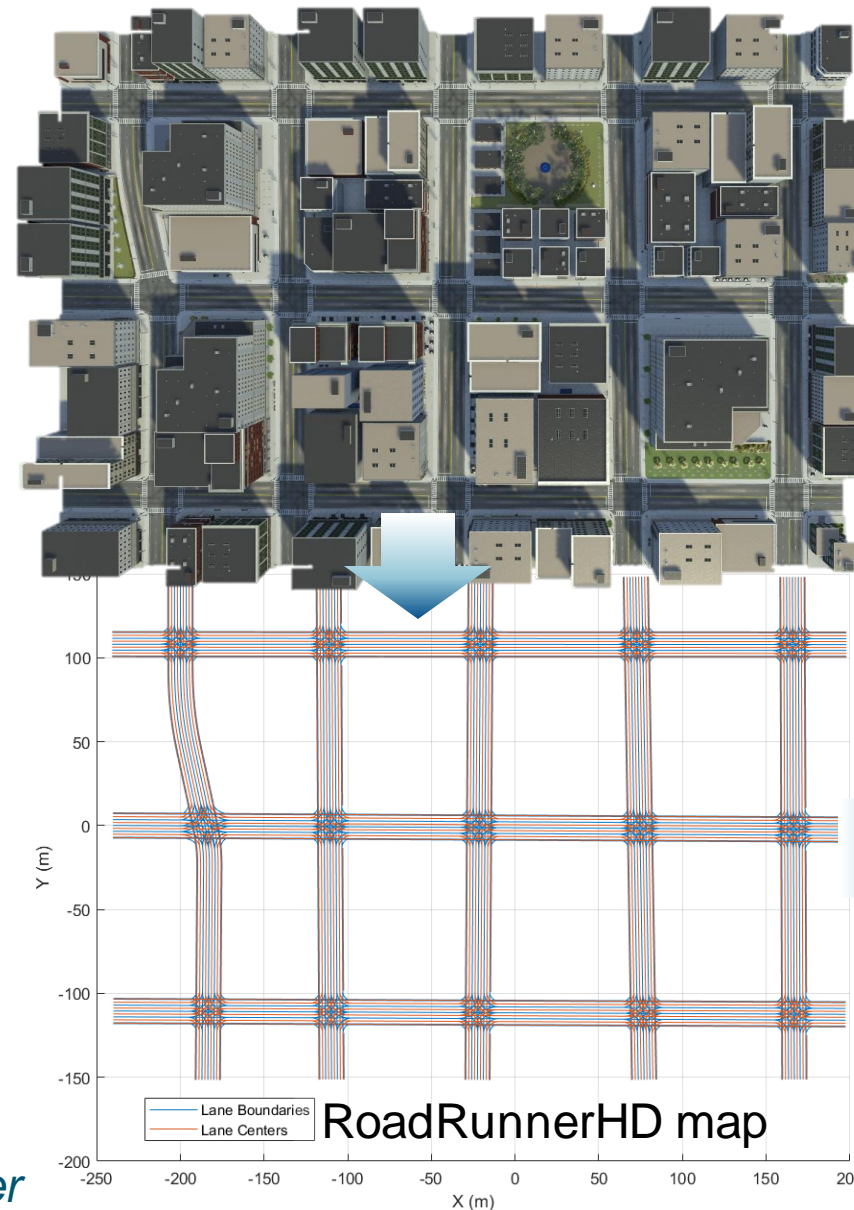
Generate V2X MAP from RoadRunner



Generate V2X MAP Message from RoadRunner

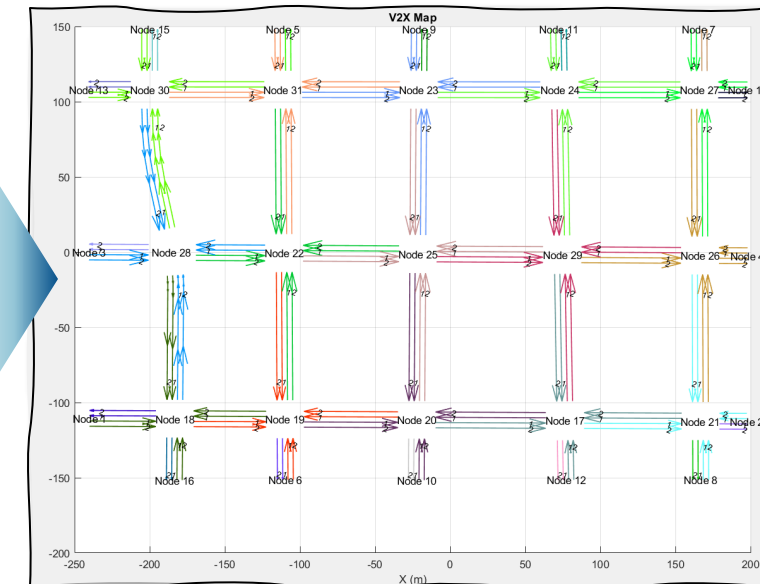
Generate MAP message and model road side unit for vehicle-to-everything (V2X) communication.

Since R2024a

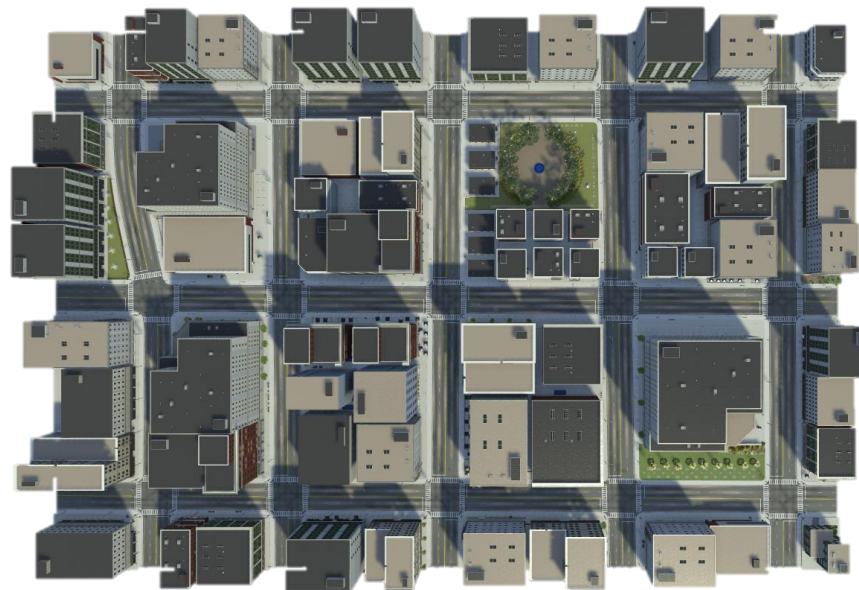


RoadRunner
Scene & Scenario

V2X MAP



Get RoadRunnerHD map for the scene used in scenario simulation



USCityBlockBidirectional.rrscene
GenerateMapMessage.rrscenario

get("Map")



Road data model for representing high-definition (HD) map data in a RoadRunner scene.

Property ^	Value
rrHDMap x	
1x1 roadrunnerHDMap	
Author	""
GeoReference	[0,0]
GeographicBoundary	[-240.8043,-152.9707,-6.8775;198.9754,148.3578,54.5400]
Lanes	2350x1 Lane
LaneBoundaries	2947x1 LaneBoundary
LaneGroups	538x1 LaneGroup
LaneMarkings	3x1 LaneMarking
Junctions	15x1 Junction
BarrierTypes	5x1 BarrierType
Barriers	56x1 Barrier
SignTypes	0x1 SignType
Signs	0x1 Sign
StaticObjectTypes	69x1 StaticObjectType
StaticObjects	2888x1 StaticObject

```
% Open Scene and Scenario
rrApp = roadrunner(rrProjectPath);

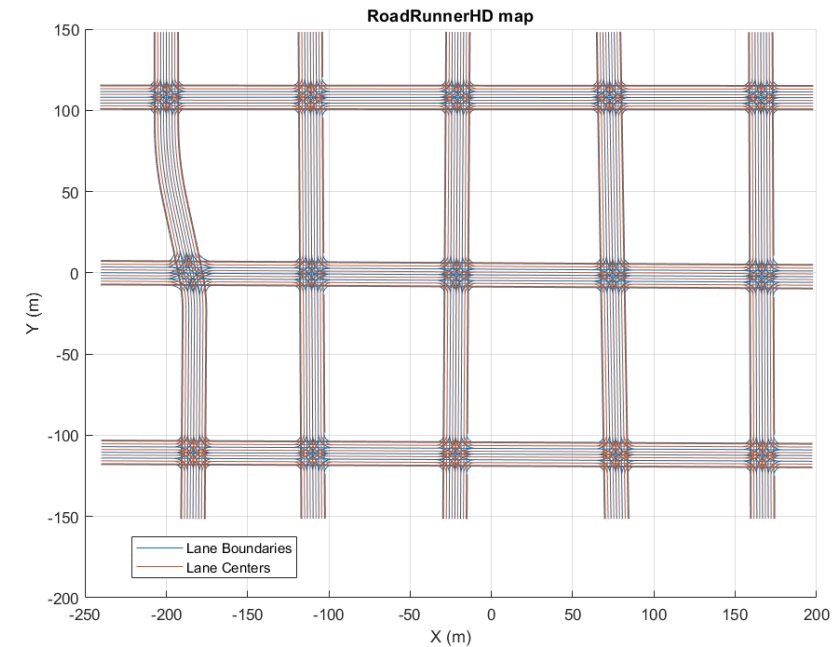
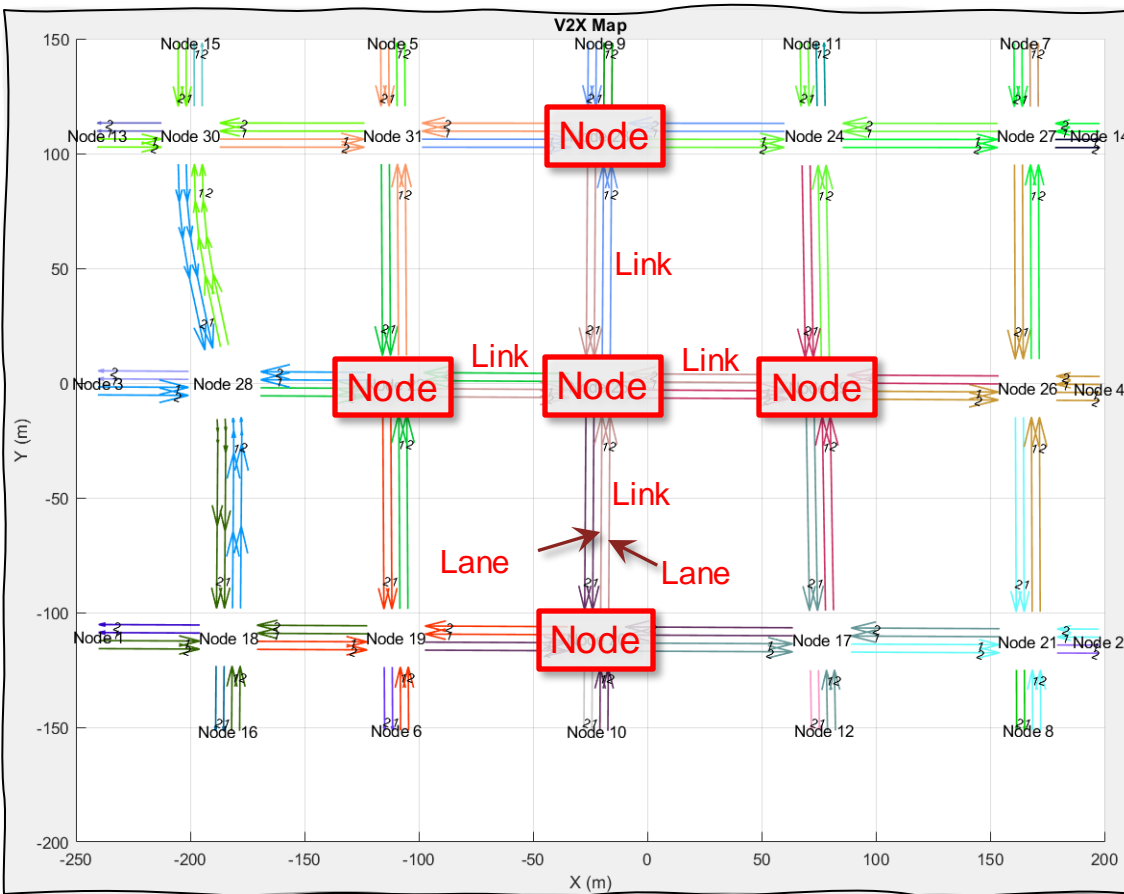
% Open Scene
openScene(rrApp, "USCityBlockBidirectional.rrscene");
% Open the scenario
openScenario(rrApp, "GenerateMapMessage.rrscenario");
```

```
% Create Simulation object
rrSim = createSimulation(rrApp);

% Get RoadRunnerHD map for the scene used
in scenario simulation.
rrHDMap = get(rrSim, "Map");
```

```
% Plot RoadRunner HD Map
plot(rrHDMap, 'ShowLineMarkers', false);
```

Generate V2X map message from RoadRunnerHD map data



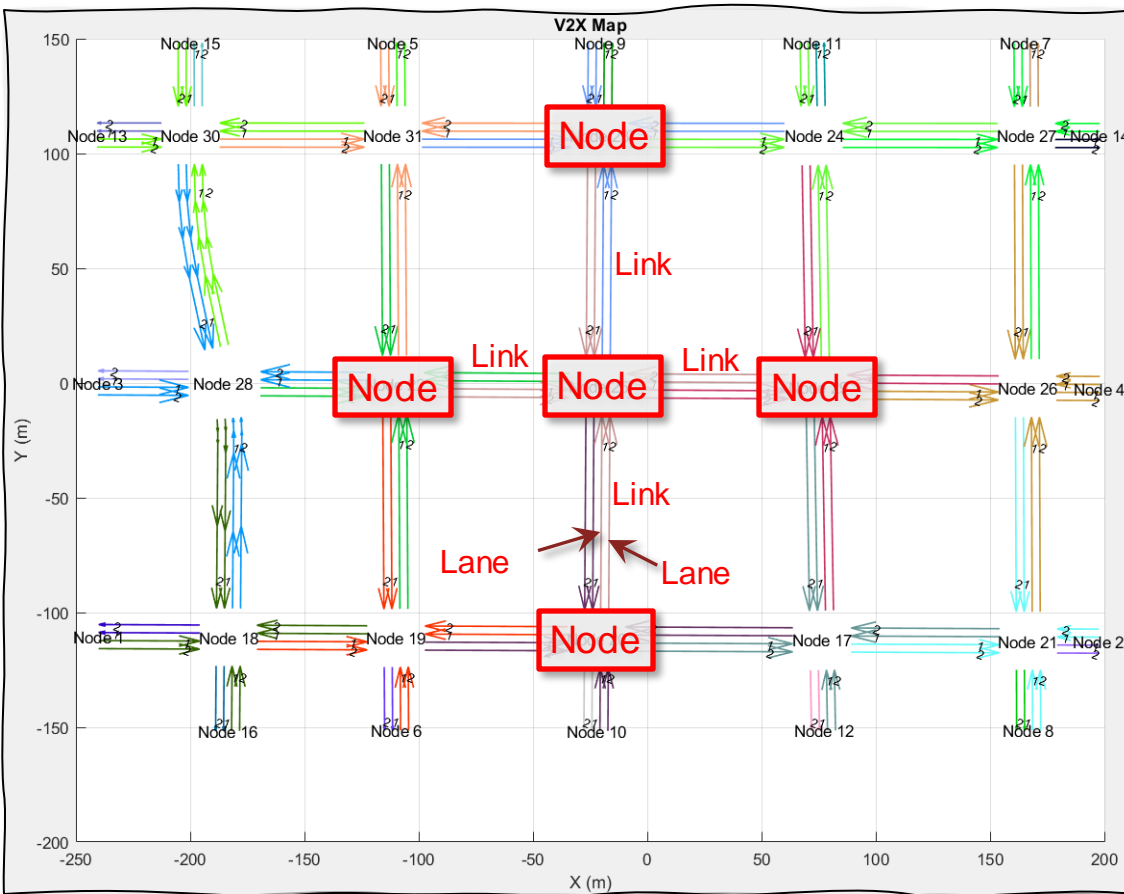
- 1) Finds all **nodes** (intersections or junctions)
- 2) Find **links** connecting all nodes.
- 3) Find connections between all **lanes**.

```

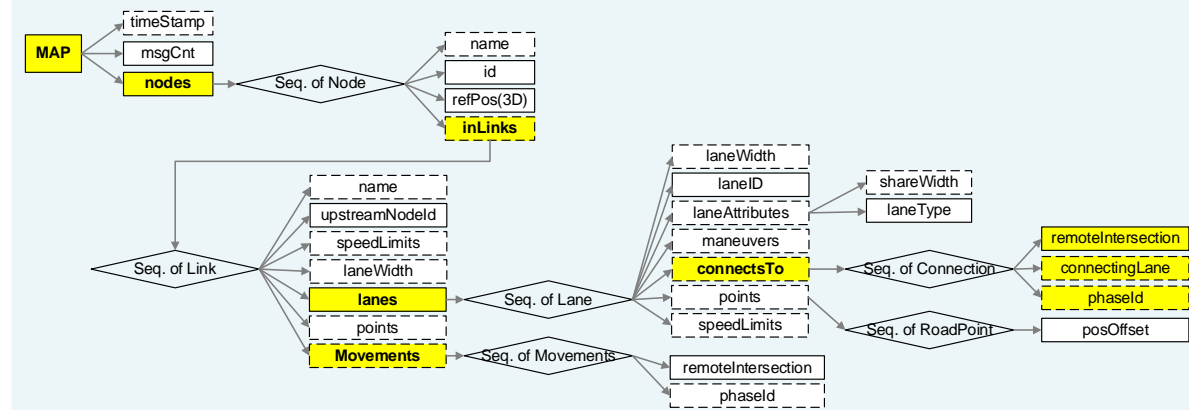
% Generate MAP message
sceneOrigin = [42.3648, -71.0214, 10.0];
v2xMapMsg = helperGenerateV2XMap(rrHDMMap, sceneOrigin);

% Visualize MAP message
helperPlotV2XMap(v2xMapMsg);
  
```

Generate V2X map message from RoadRunnerHD map data



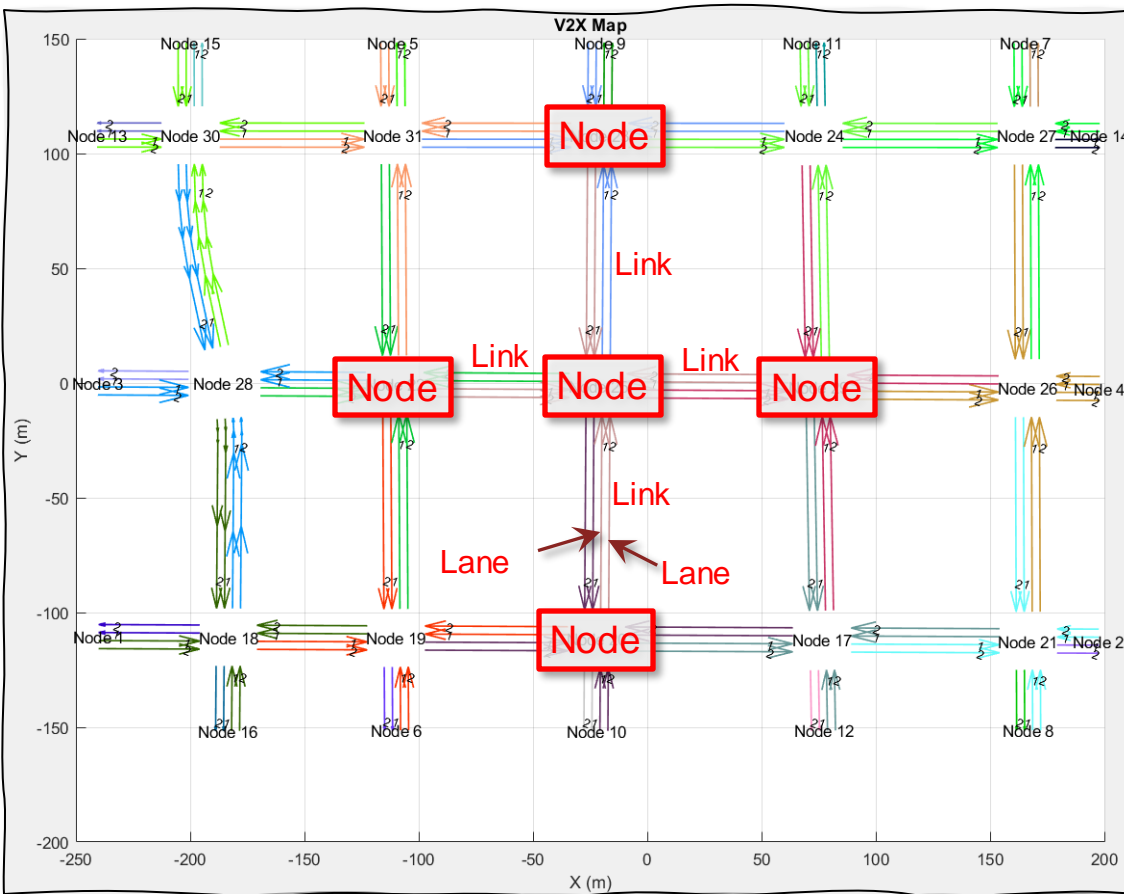
V2X Map message



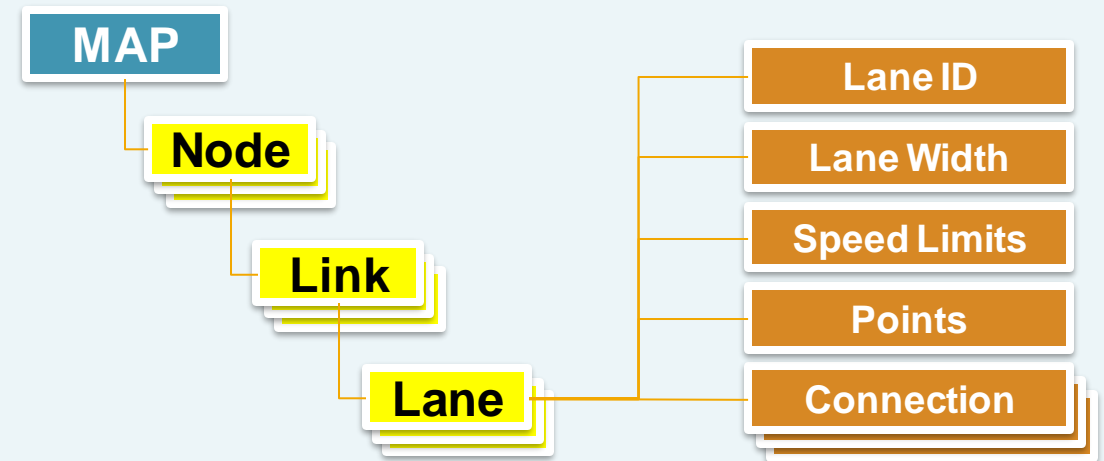
- 1) Finds all **nodes** (intersections or junctions)
- 2) Find **links** connecting all nodes.
- 3) Find connections between all **lanes**.
- 4) Encapsulate the nodes, links, and lane connections within **V2X map messages**.

- T/CSAE 53-2020, *Cooperative Intelligent Transportation System— Vehicular Communication Application Layer Specification and Data Exchange Standard (Phase I)*. China Society of Automotive Engineers, 2020.
- ~ SAE J2735, V2X Communications Message Set Dictionary

Generate V2X map message from RoadRunnerHD map data



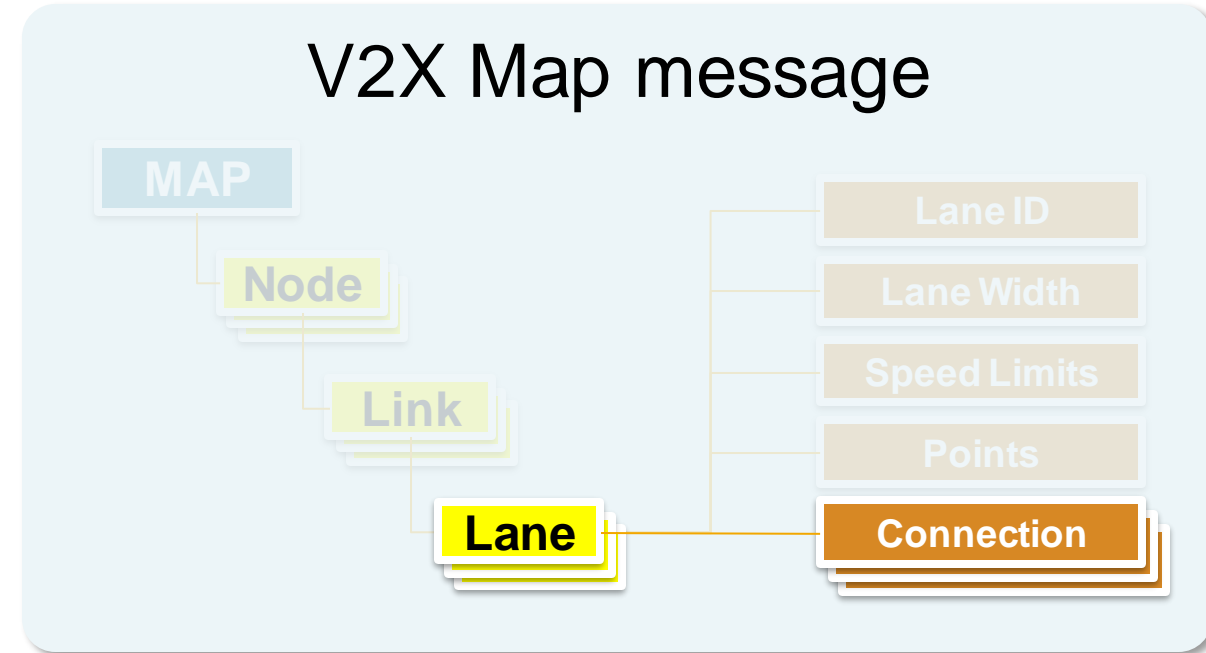
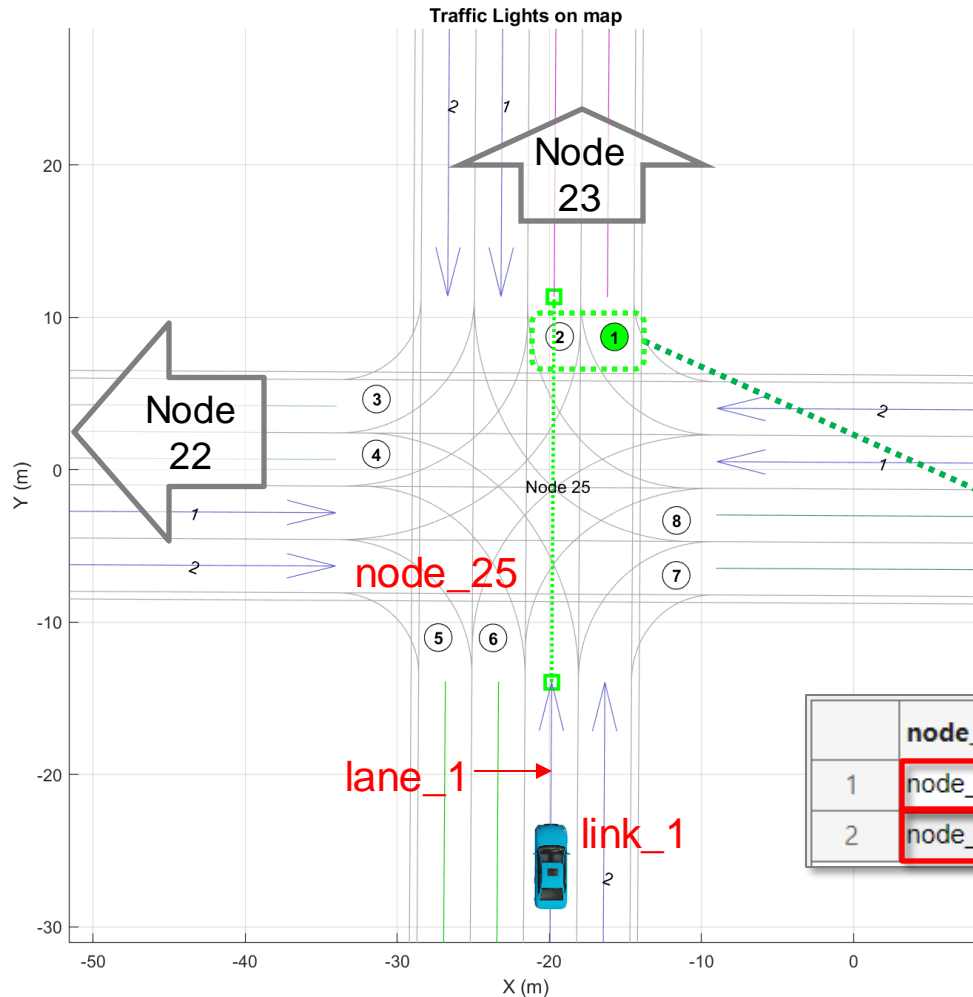
V2X Map message



- 1) Finds all **nodes** (intersections or junctions)
- 2) Find **links** connecting all nodes.
- 3) Find connections between all **lanes**.
- 4) Pack the nodes, links, and lane connections with **V2X map messages**.

- T/CSAE 53-2020, *Cooperative Intelligent Transportation System— Vehicular Communication Application Layer Specification and Data Exchange Standard (Phase I)*. China Society of Automotive Engineers, 2020.
- ~ SAE J2735, V2X Communications Message Set Dictionary

V2X map message: Lane Connection to downstream nodes and traffic signal id



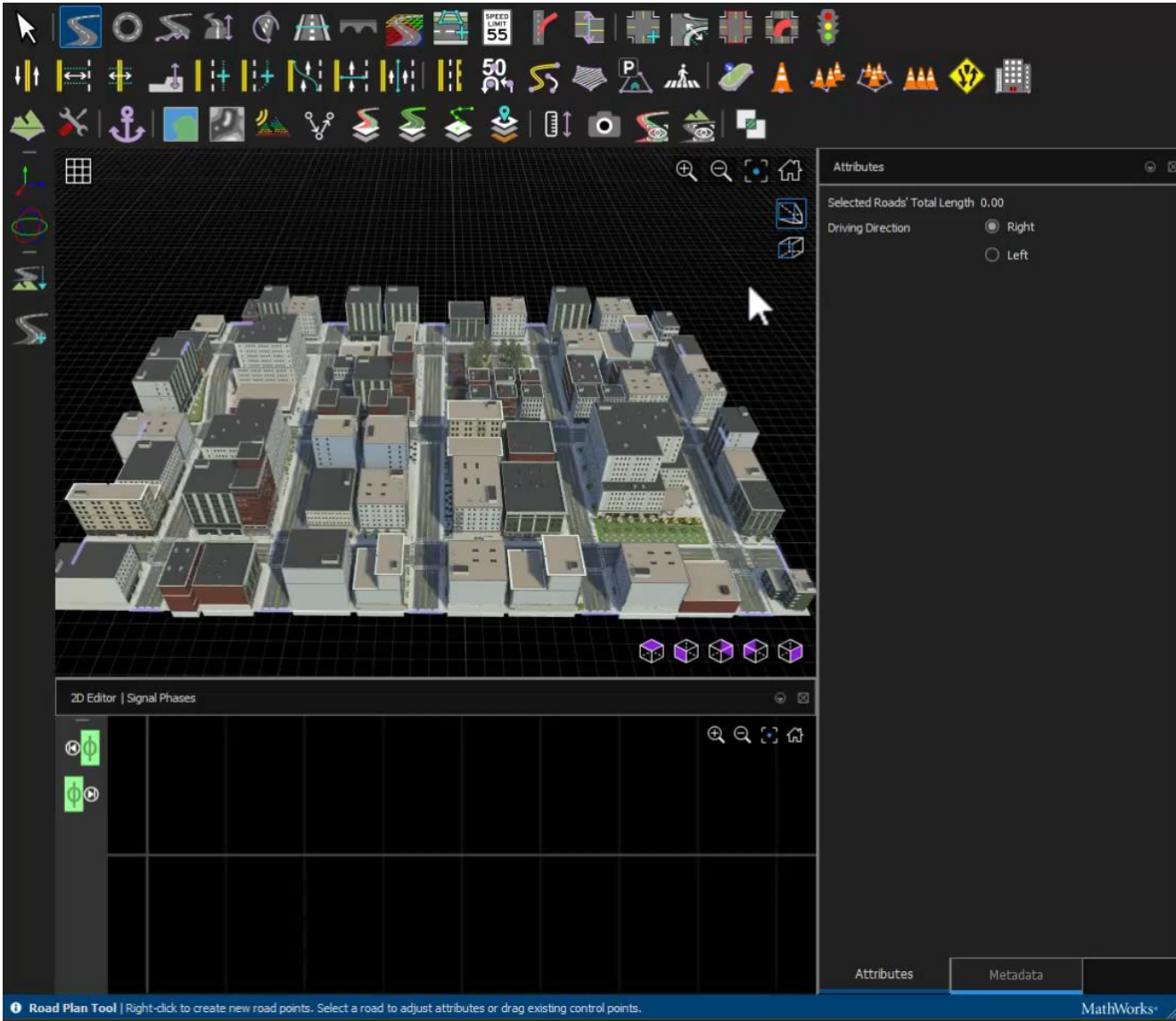
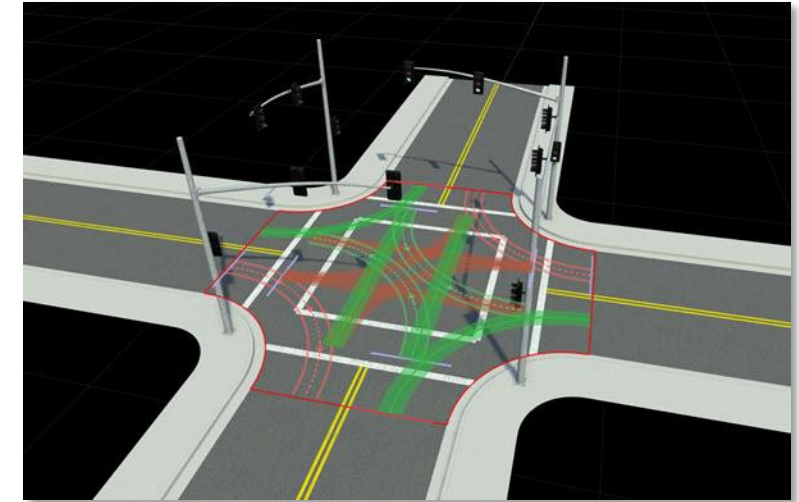
	node_id	link_id	lane_id	signal_id	remote_node	connecting_lane	maneuver	description
1	node_25	link_1	1	2	22	1	0002	Turn Left
2	node_25	link_1	1	1	23	1	0001	Move Straight

Workflow

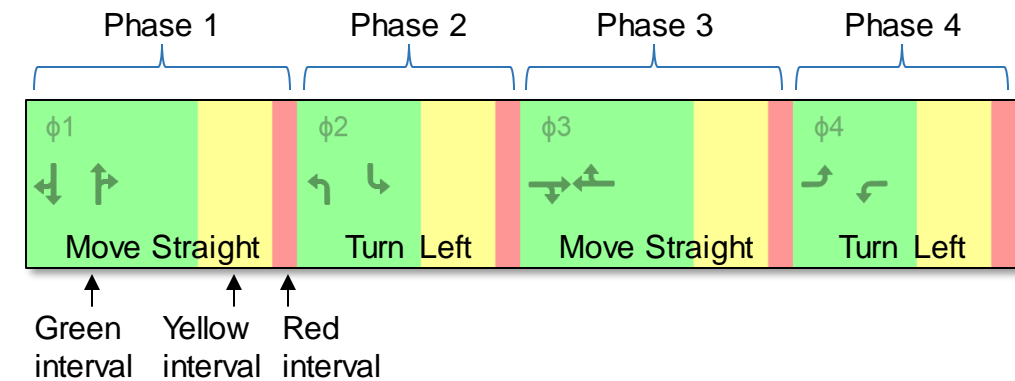


Note) V2X: Vehicle-To-Everything, SPaT: Signal Phase and Timing

“Signal Tool” of RoadRunner



- The Signal Tool is used to configure
 - Junction Signalization
 - Signal Traffic Phases

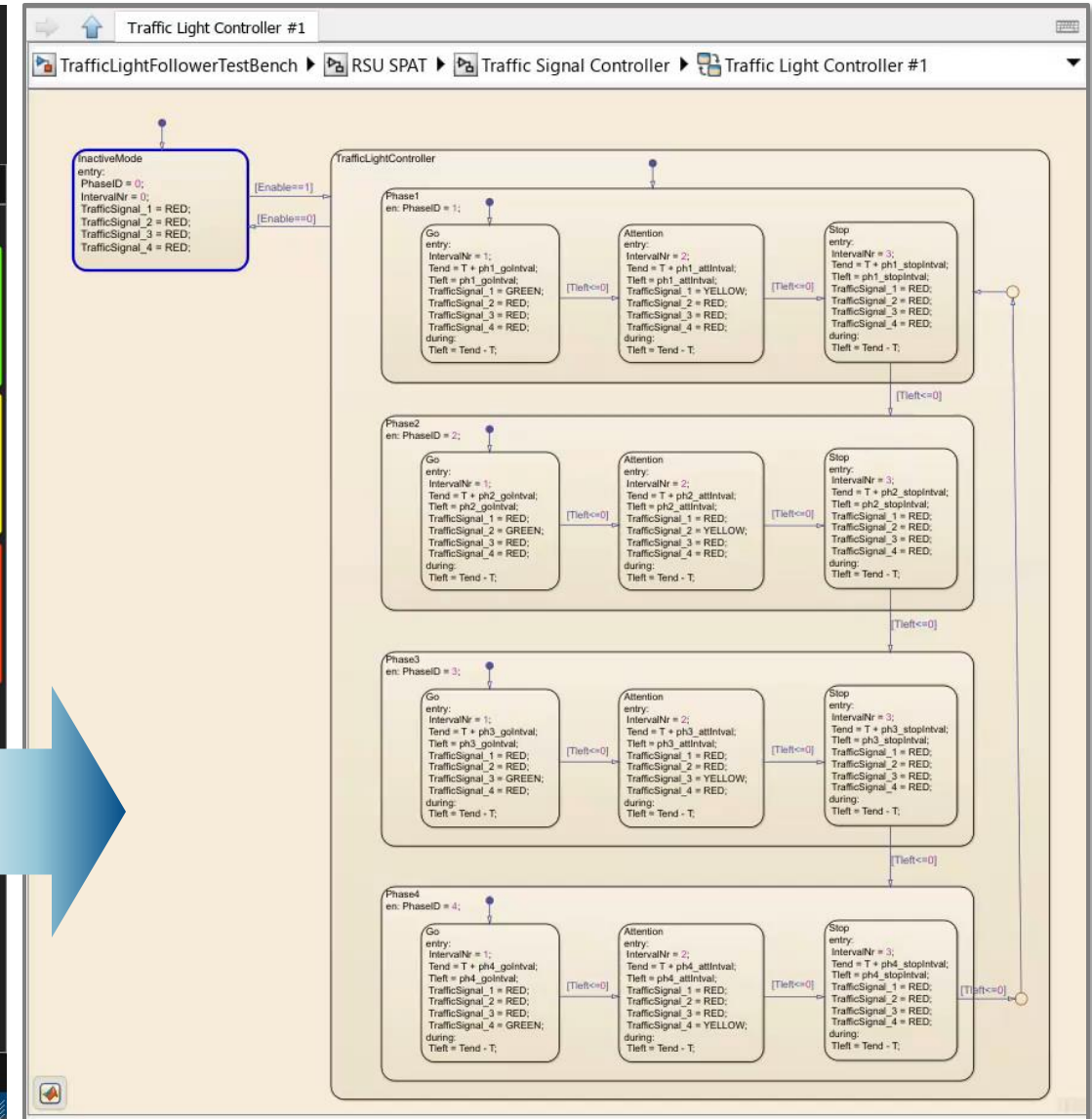


Implement traffic light controller using Stateflow[®]

Interval Name	Interval Type	Interval Time
Interval 1	Green	5.00
Interval 2	Yellow	5.00
Interval 3	Red	1.00

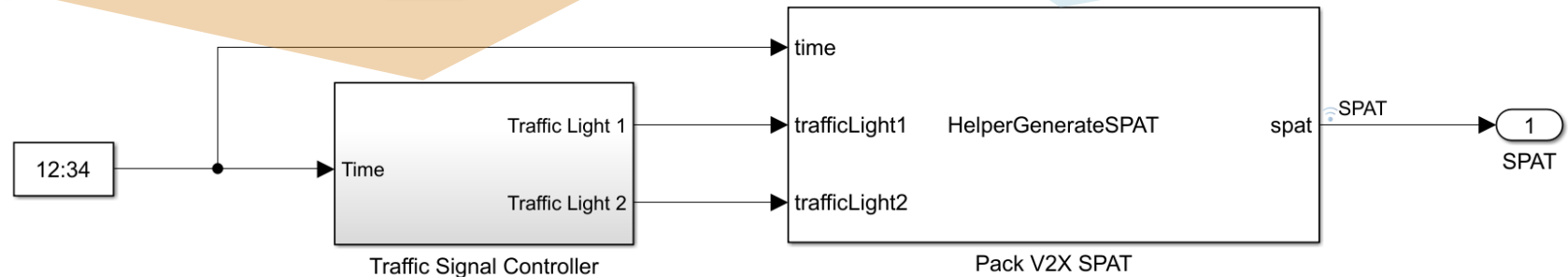
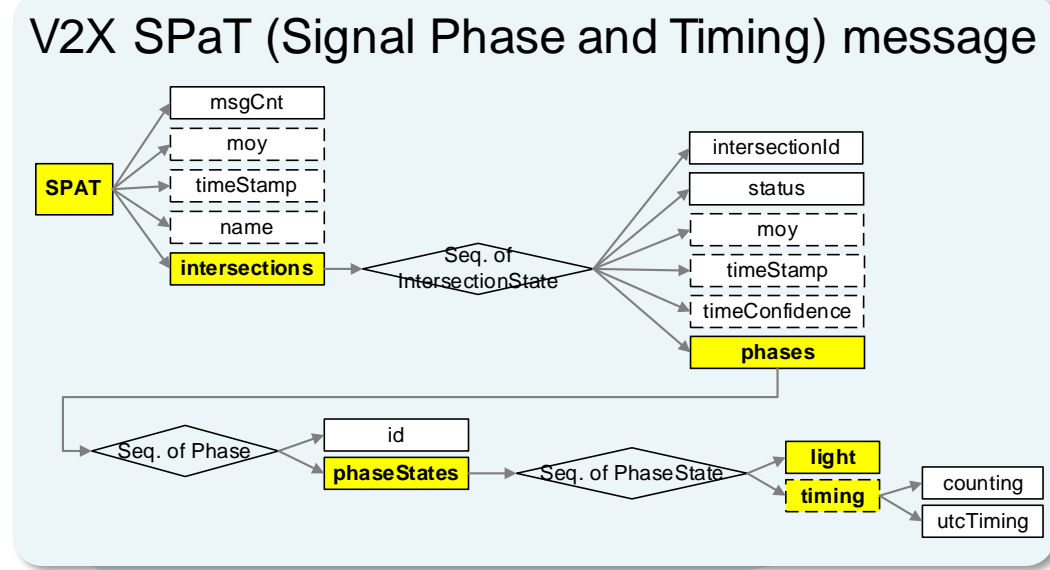
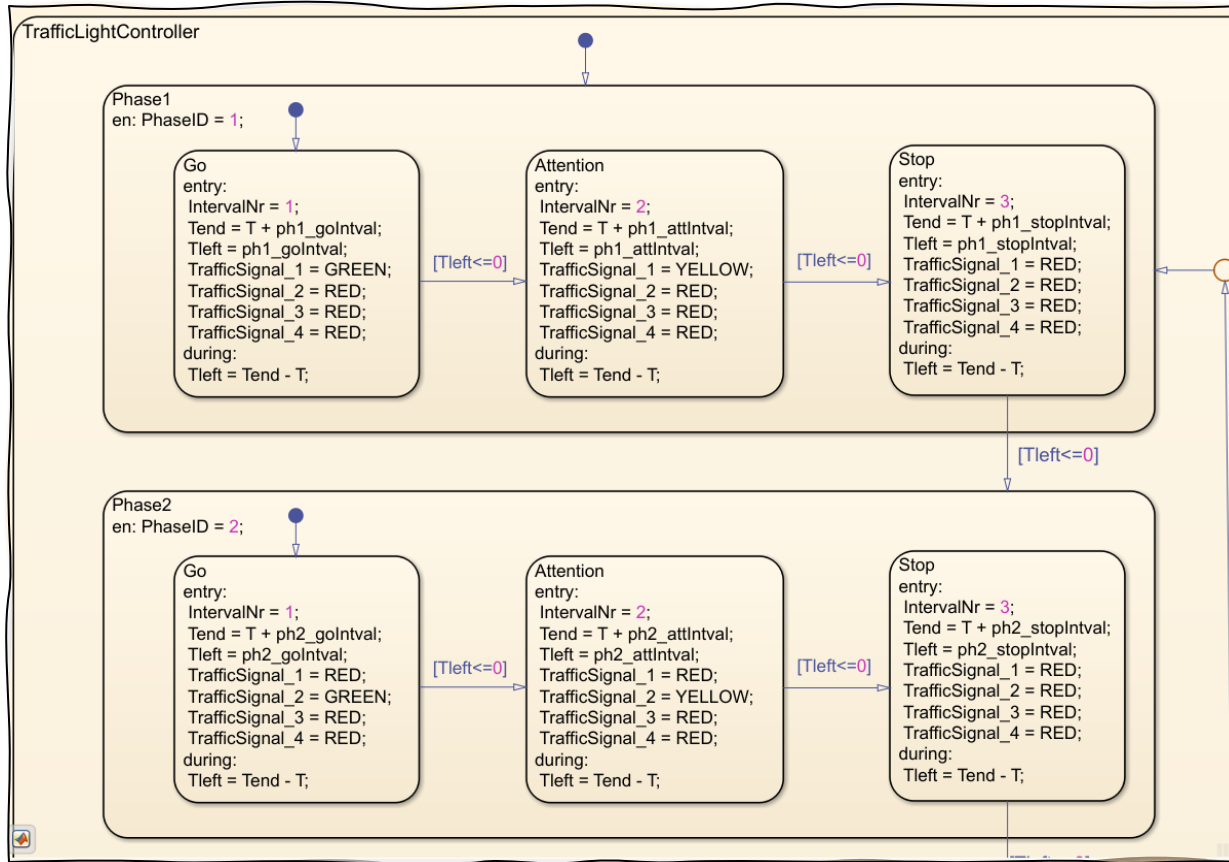
Signal Phasing Table:

Phase	Signal 1	Signal 2	Signal 3	Signal 4
φ1	Green	Red	Red	Red
φ2	Red	Green	Red	Red
φ3	Red	Red	Green	Red
φ4	Red	Red	Red	Green



Generate V2X SPaT (Signal Phase and Timing) message

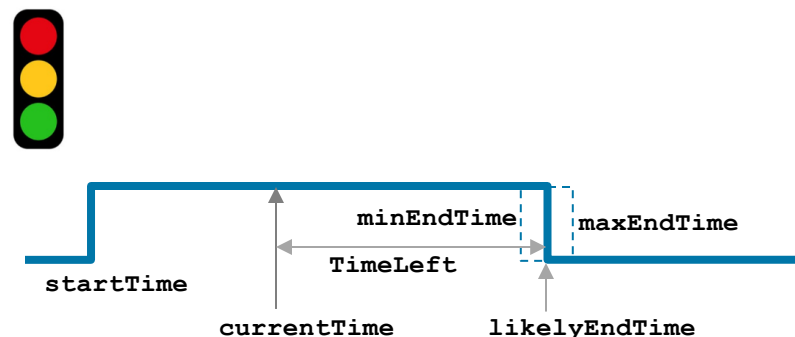
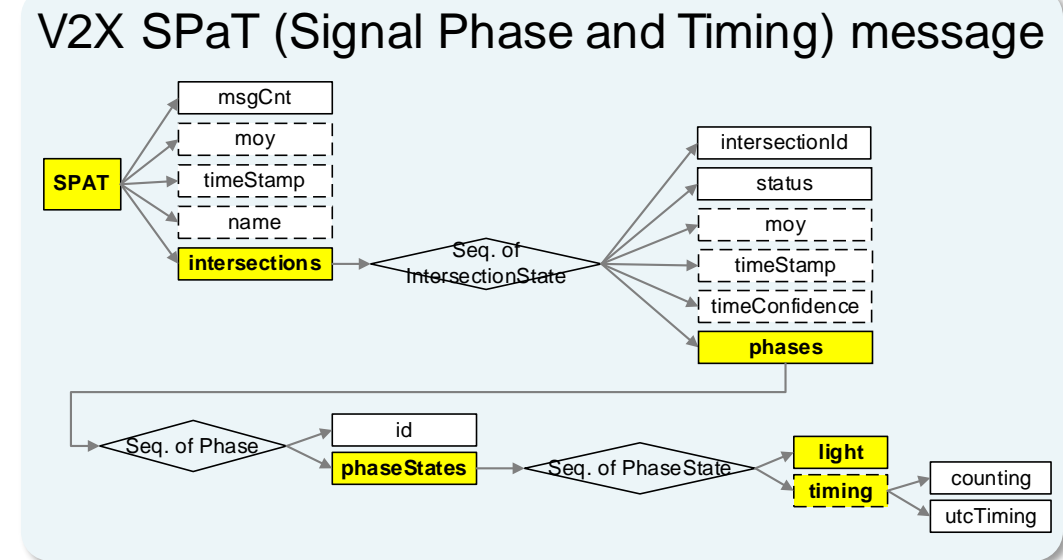
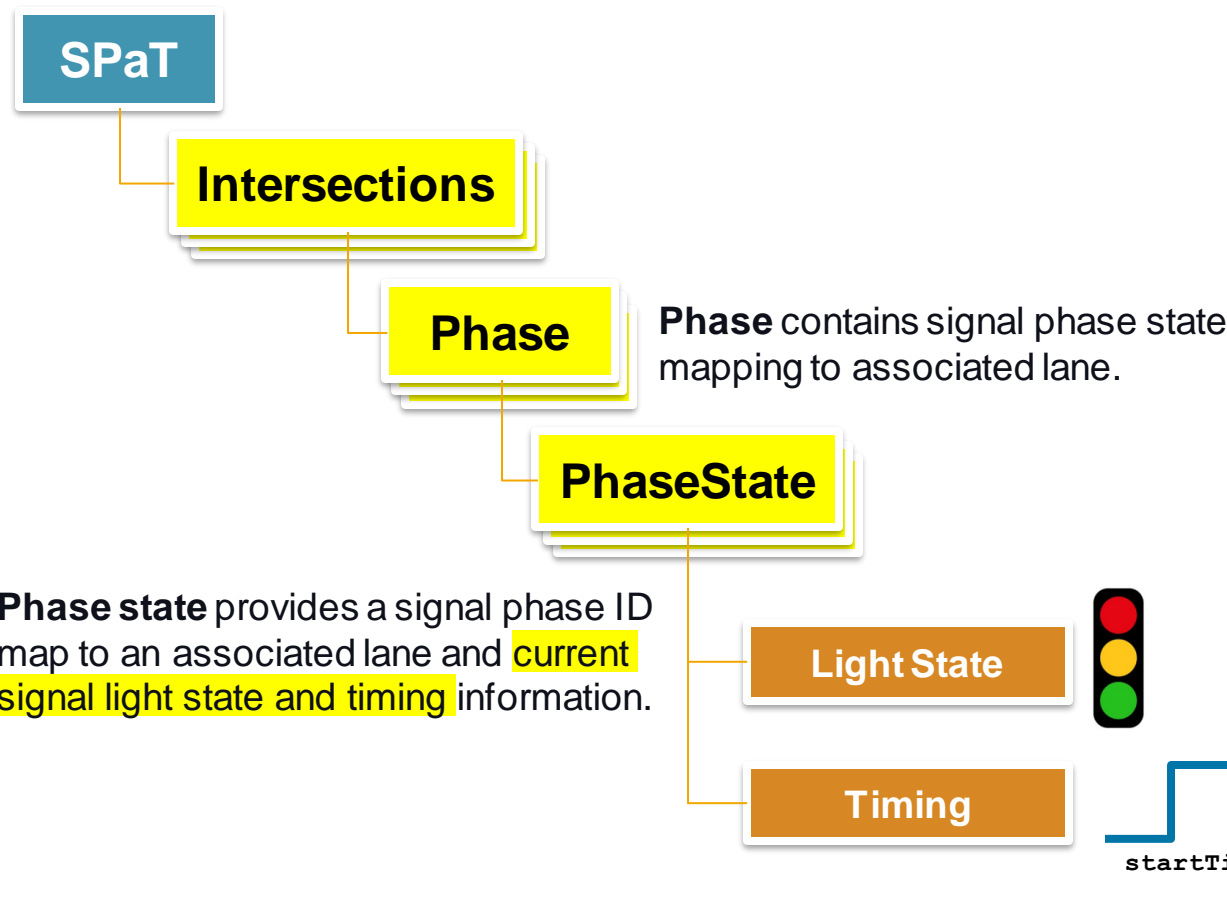
T/CSAE 53-2020



Generate V2X SPaT (Signal Phase and Timing) message

T/CSAE 53-2020

- SPaT message describes the current state of a signal system and its phases and relates this to the specific lanes in the intersection.



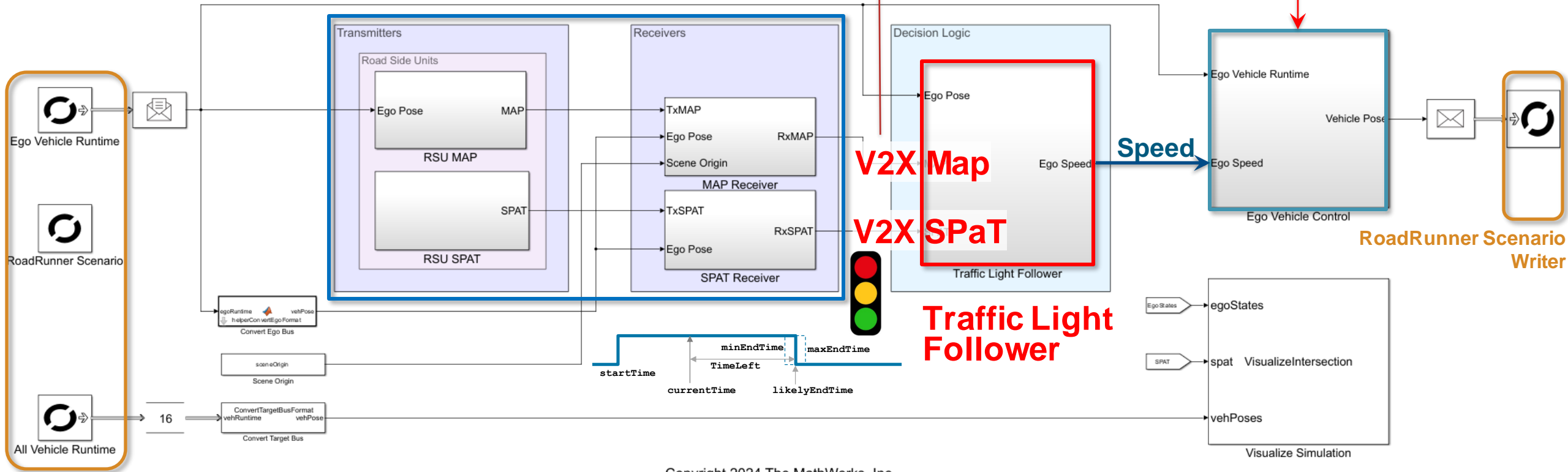
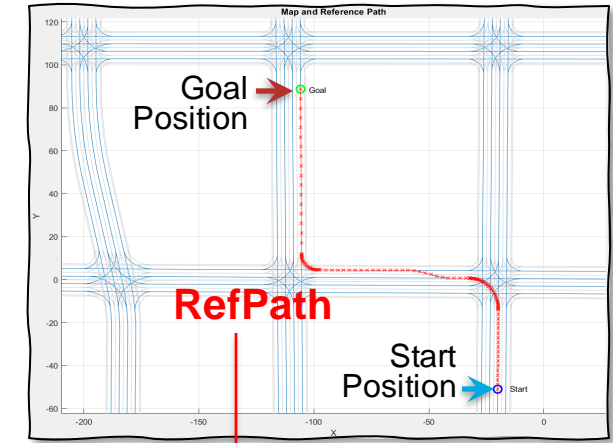
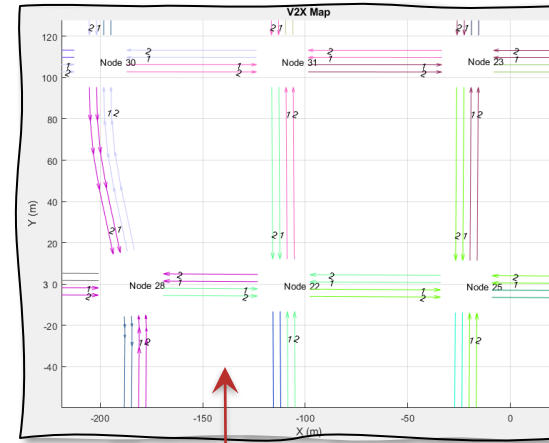
Workflow



Note) V2X: Vehicle-To-Everything, SPaT: Signal Phase and Timing

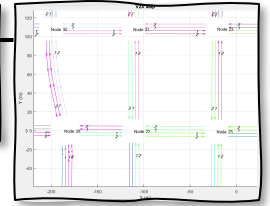
Traffic Light Follower

Road Side Unit for V2X map & SPaT



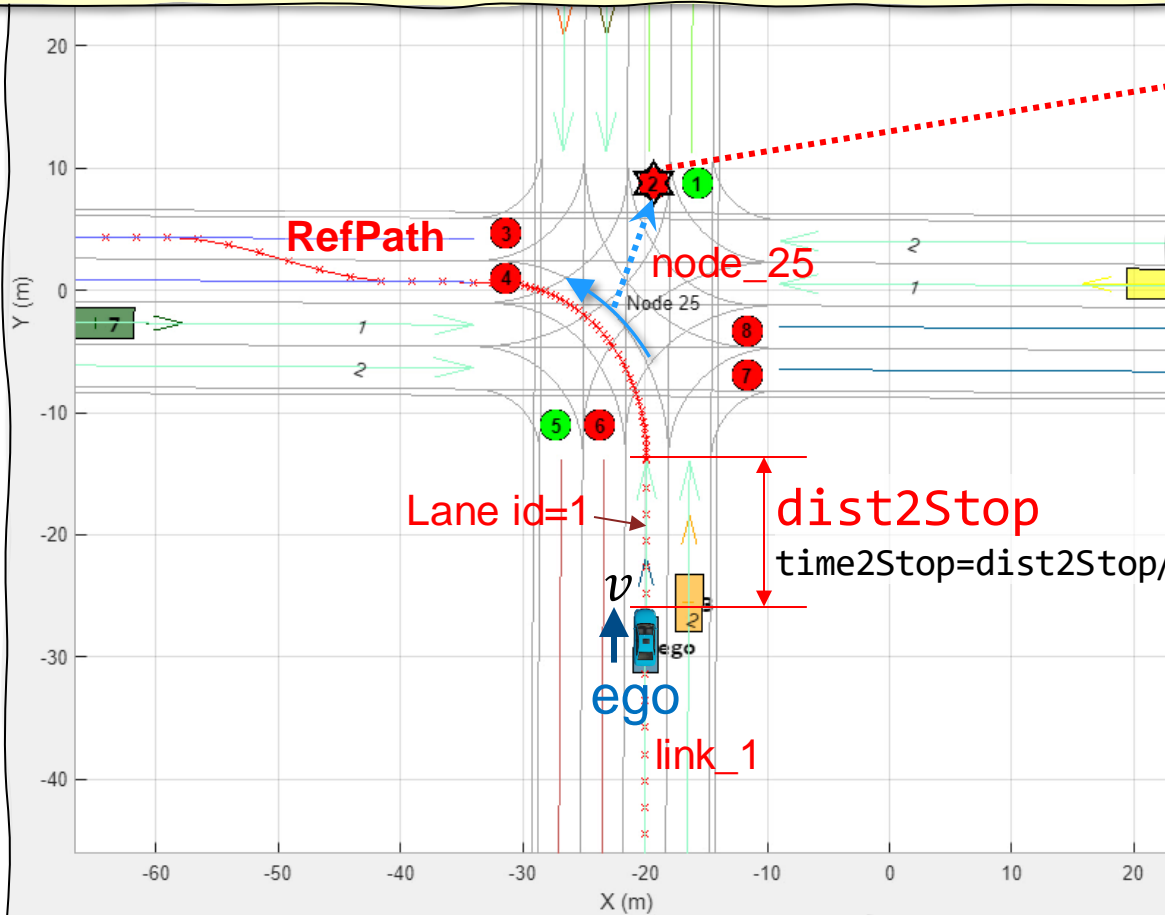
Query V2X Map & SPaT

```
% v2xMapQuery: get map query API object
obj.MapQuery = v2xMapQuery(obj.V2XMap,obj.origin);
```



% **getConnection**: get matching traffic light signal_id and maneuver
% at intersection based on ego current states and reference path

```
[signal_id_ego,maneuver_ego] =
obj.MapQuery.getConnection(lane_ego,obj.RefPath,obj.SearchDist);
```



Ego current state



Ego V2X Map info

Maneuver at Junction	TURN_LEFT
Traffic Signal Runtime associated with ego	
SignalId	2
Ego States	
Position (x,y)	-19.9, -29.0
Speed (m/s)	10.0
Yaw (deg)	89.6
Node Name	
Link Name	link_1
Lane Id	1
Dist2StopLine (m)	12.28
Time2StopLine (sec)	1.51
Action	Apply brake to stop

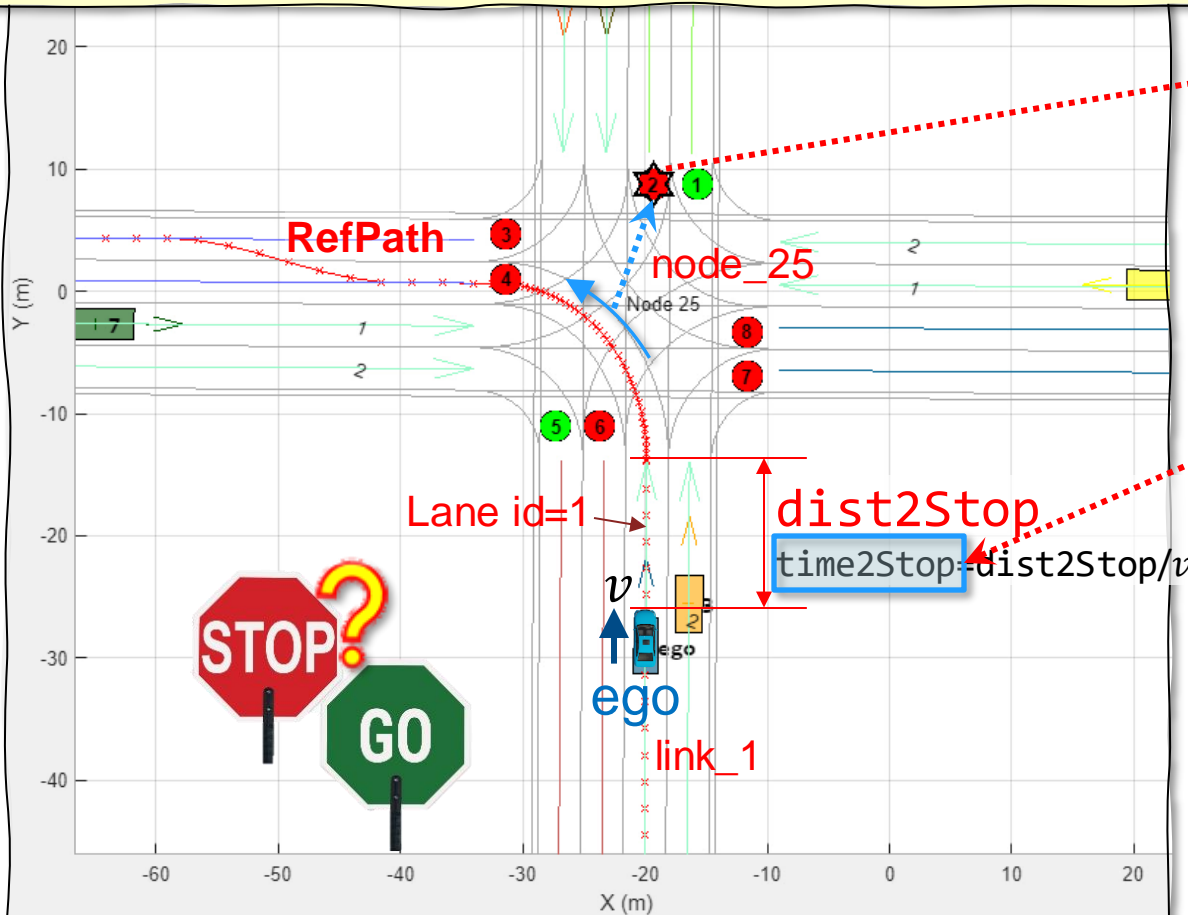
```
% locateVehicleOnMap: get v2x map information
% associated with the current ego position
[node_ego,link_ego,lane_ego,dist2stop,info] =
obj.MapQuery.locateVehicleOnMap(Ego.Position);
```

Stop or Go at the intersection?

% **getConnection**: get matching traffic light signal_id and maneuver
% at intersection based on ego current states and reference path

[signal_id_ego, maneuver_ego] =

obj.MapQuery.getConnection(lane_ego, obj.RefPath, obj.SearchDist);



% **getCurrentSPAT**: get traffic light state and timing

% associated with ego car

[Light, Tleft, Tdur] = **getCurrentSPAT**(NodeId, SignalId, **SPAT**);

Maneuver at Junction **TURN_LEFT**

Traffic Signal Runtime associated with ego

SignalId	BulbColor	TLeft (s)	Duration (s)
2	RED	5.90	25.00

switch **Light** % traffic bulb color associated with ego

→ case **BulbColor.RED**

→ if **Tleft** > **Time2Stop**

% red light stays when ego vehicle reaches the stop line

→ StopAction = true; % then, stop in advance

end

case {**BulbColor.GREEN**, **BulbColor.YELLOW**}

if **Tleft** < **Time2Stop**

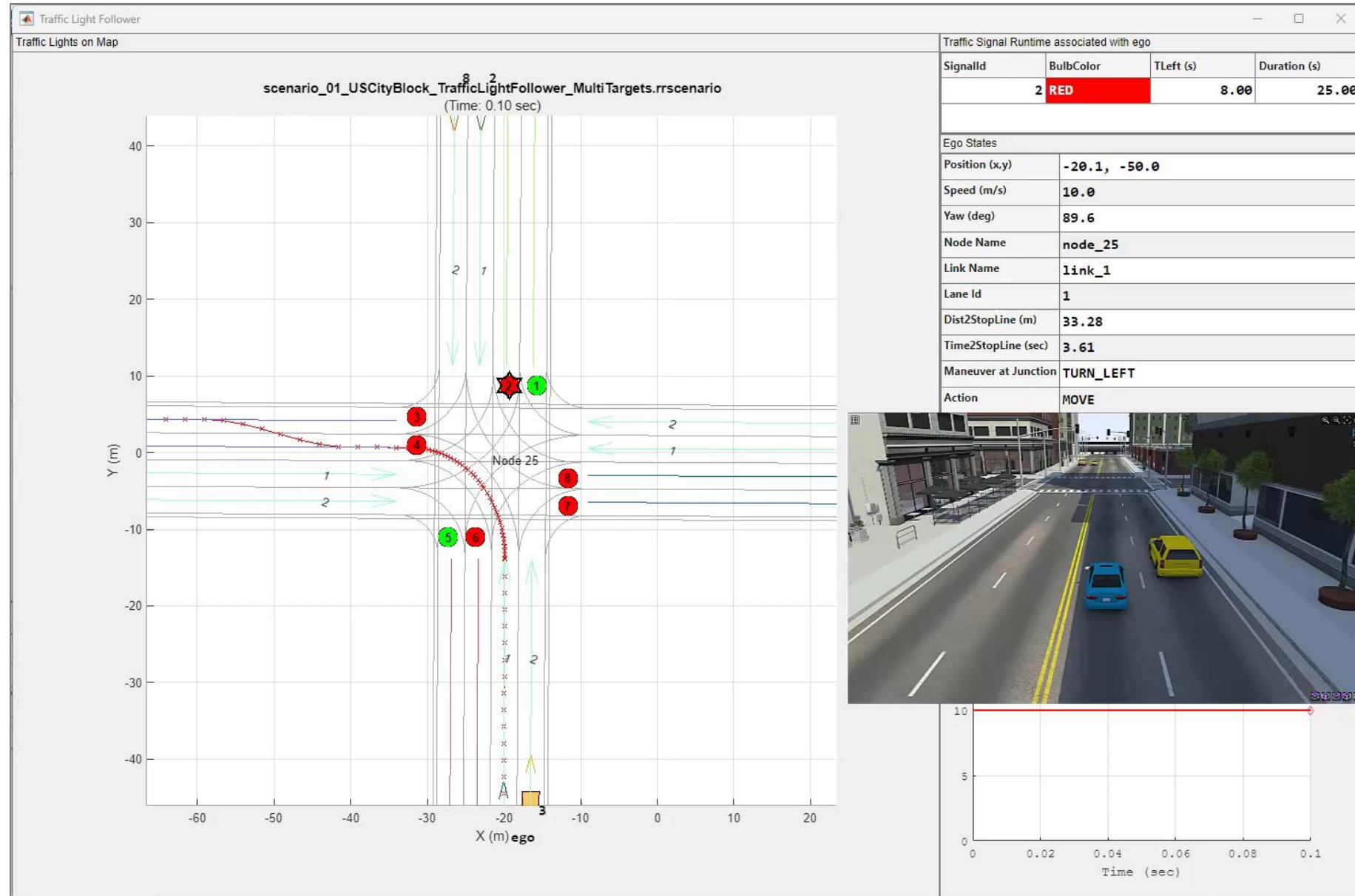
% green or yellow light will be expired when ego vehicle reaches the stop line

StopAction = true; % then, stop in advance

end

end

Traffic light follower with co-simulating RoadRunner Scenario



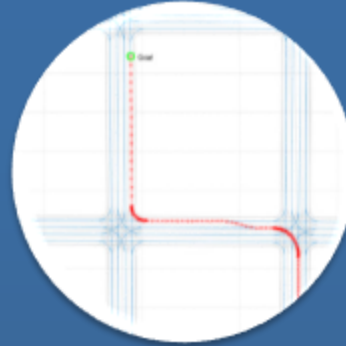
Key Takeaways:

Automated Driving in the Urban Environment with RoadRunner Scenario



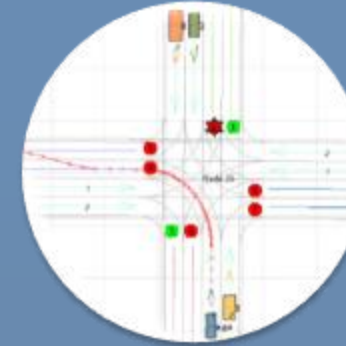
Create a complex **Urban Scene** consisting of Intersections with Traffic Lights.

- *RoadRunner*
- *RoadRunner Asset Library*



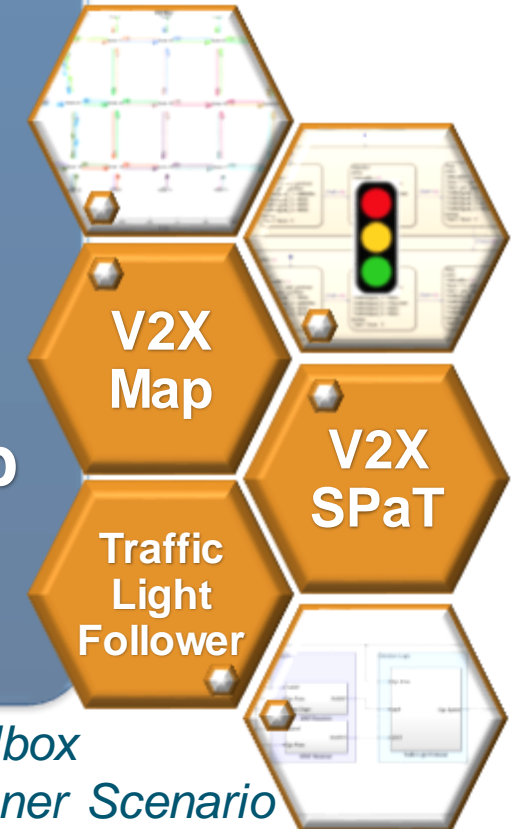
Design **Path Planner** using **A-star Planner**.

- *Navigation Toolbox*



Develop **Behavioral Planner** using **V2X Map and SPaT**.

- *Automated Driving Toolbox*
- *RoadRunner, RoadRunner Scenario*
- *Simulink, Stateflow*



MathWorks
**AUTOMOTIVE
CONFERENCE 2024**
North America

Thank you

Please contact me at spark@mathworks.com
with questions

