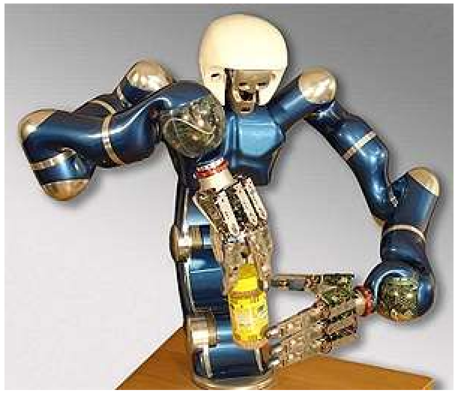
Control System Design for Automated Driving

Lecture 12

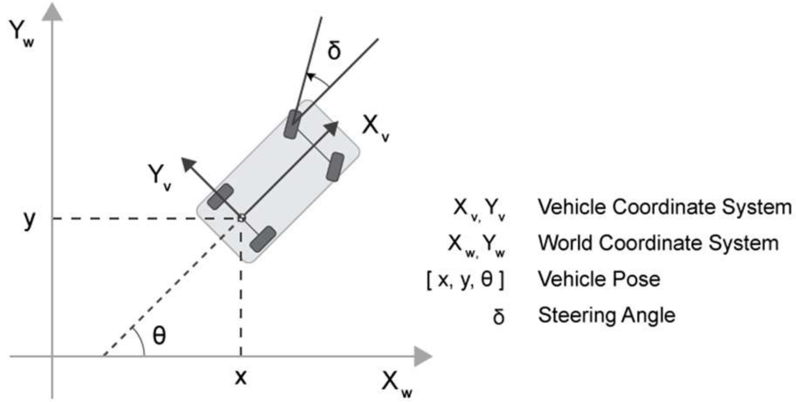

Create Closed-Loop ADAS Algorithm Using Driving Scenario with Simulink

- In a closed-loop ADAS algorithm, the ego vehicle is controlled in response to the changes in its scenario environment as the simulation advances.
- To test the scenario, you use a driving scenario that was saved from the Driving Scenario Designer app. In this model, you read in a scenario using a Scenario Reader block, and then visually verify the performance of the algorithm.
- Although the scenario includes a predefined ego vehicle, the Scenario Reader block is configured to ignore this ego vehicle definition. Instead, the ego vehicle is defined in the model and specified as an input to the Scenario Reader block.

Scenario Reader Block

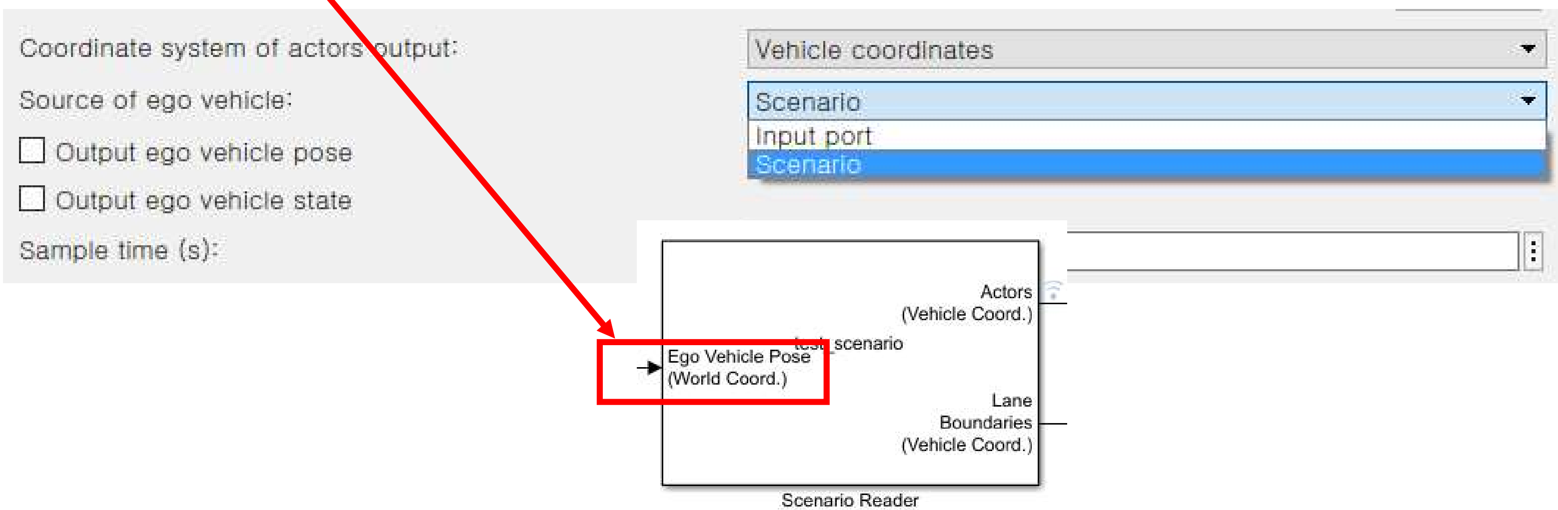
- To import the generated scenario into a Simulink model, use a Scenario Reader block.
- This block reads the roads and non-ego actors from either scenario file saved from the drivingScenario app.
- The block outputs the poses of non-actors in either the ego vehicle coordinate system or the world coordinates of the scenario.
- The ego vehicle is passed into the block through an input port.

Scenario Reader Block



- Block Parameters
  - ➢Scenario Selection

  - ➢Source of ego vehicle can be either scenario or input port.



Scenario Reader Block

- Scenario reader block outputs are generated as Simulink Bus containing Matlab Structure as follows.
  - “Actors” structure

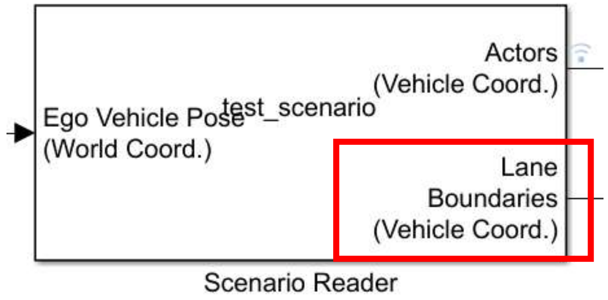
Field	Description	Type
NumActors	Number of actors (“except” ego vehicle)	Nonnegative integer
Time	Current simulation time	Real-valued scalar
Actors	Actor poses	NumActors-length array of actor pose structures

Scenario Reader Block

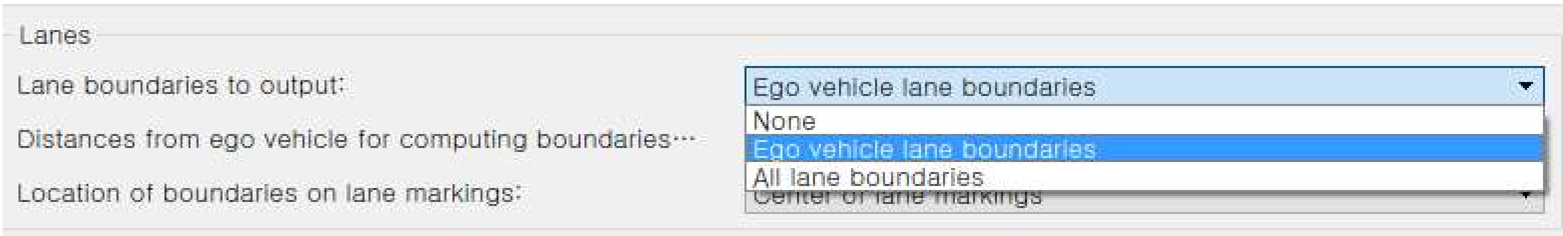
➢ "Actors.Actors" structure

Field	Description
`ActorID`	Scenario-defined actor identifier, specified as a positive integer.
`Position`	Position of actor, specified as a real-valued vector of the form $[x\ y\ z]$. Units are in meters.
`Velocity`	Velocity (v) of actor in the x- y-, and z-directions, specified as a real-valued vector of the form $[v_x\ v_y\ v_z]$. Units are in meters per second.
`Roll`	Roll angle of actor, specified as a real-valued scalar. Units are in degrees.
`Pitch`	Pitch angle of actor, specified as a real-valued scalar. Units are in degrees.
`Yaw`	Yaw angle of actor, specified as a real-valued scalar. Units are in degrees.
`AngularVelocity`	Angular velocity (ω) of actor in the x-, y-, and z-directions, specified as a real-valued vector of the form $[\omega_x\ \omega_y\ \omega_z]$. Units are in degrees per second.

Scenario Reader Block



- Scenario lane boundaries can be returned as a Simulink bus containing a Matlab Structure as follows



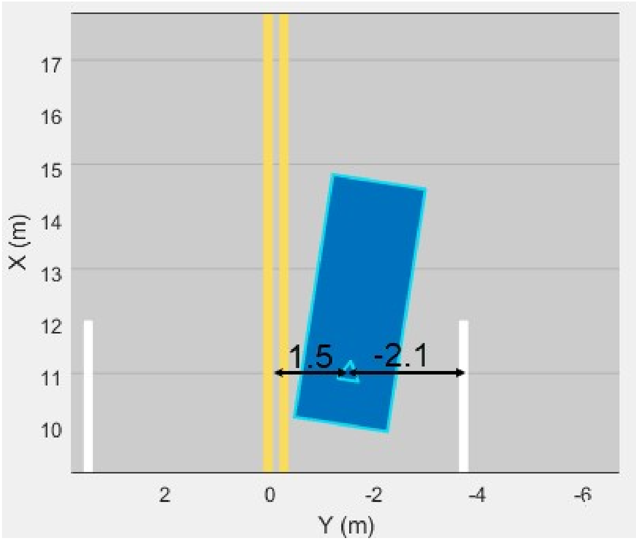
➢Lane boundaries structure

Field	Description	Type
`NumLaneBoundaries`	Number of lane boundaries	Nonnegative integer
`Time`	Current simulation time	Real scalar
`LaneBoundaries`	Lane boundaries	`NumLaneBoundaries`-length array of lane boundary structures

Scenario Reader Block

➢ "LaneBoundaries.LaneBoundaries" Structure Fields

- Coordinates
- Curvature
- CurvatureDerivatives
- HeadingAngle
- LateralOffset
- BoundaryType
- Strength
- Width and etc.



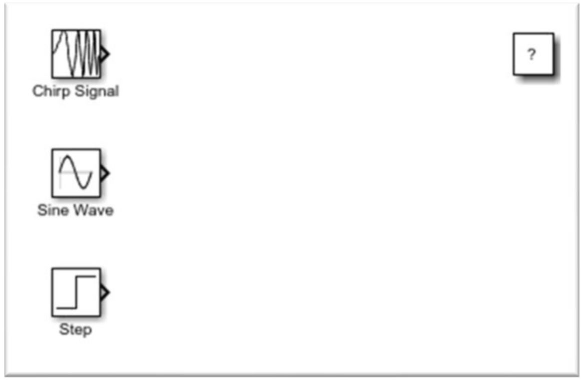
An offset to the right of the ego vehicle is negative. Units are in meters. In this image, the ego vehicle is offset 1.5 meters from the left lane and 2.1 meters from the right lane.

Refer to https://kr.mathworks.com/help/driving/ref/scenarioreader.html

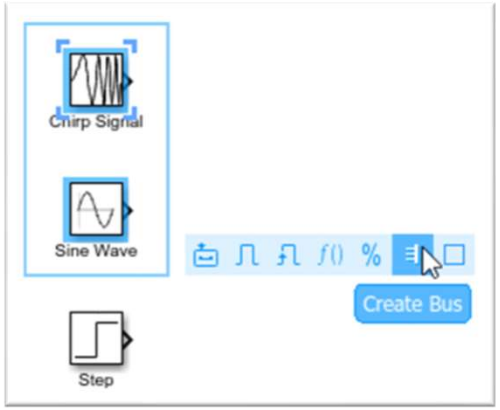
Simulink Bus

- Bus Creator : Group signal lines within a component

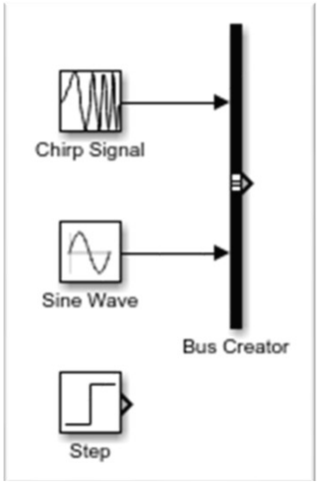
①



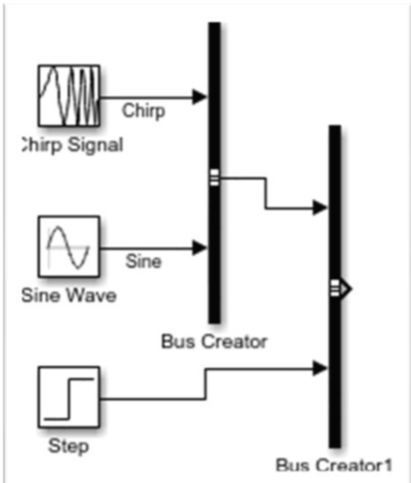
②



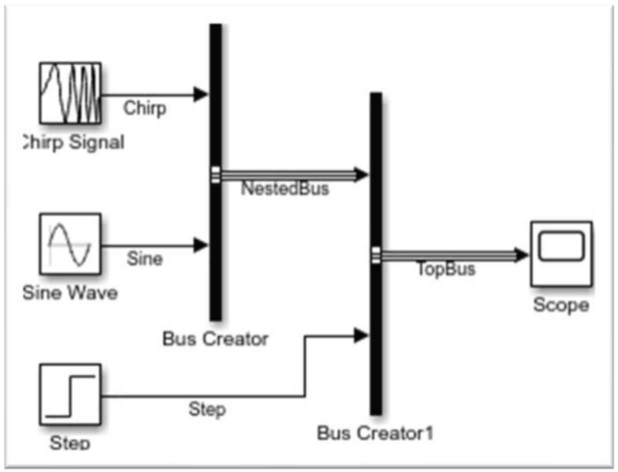
③



④

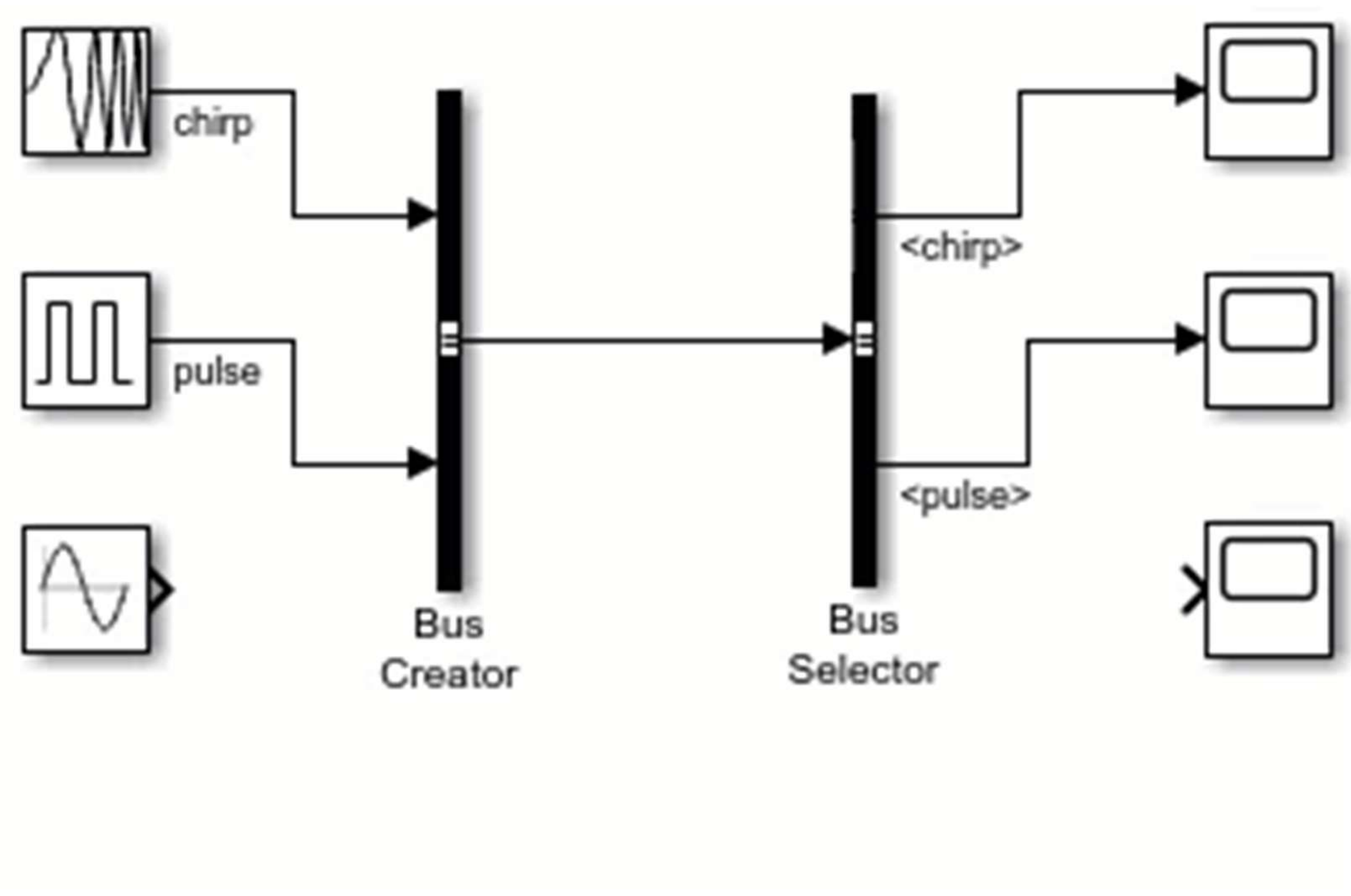


⑤



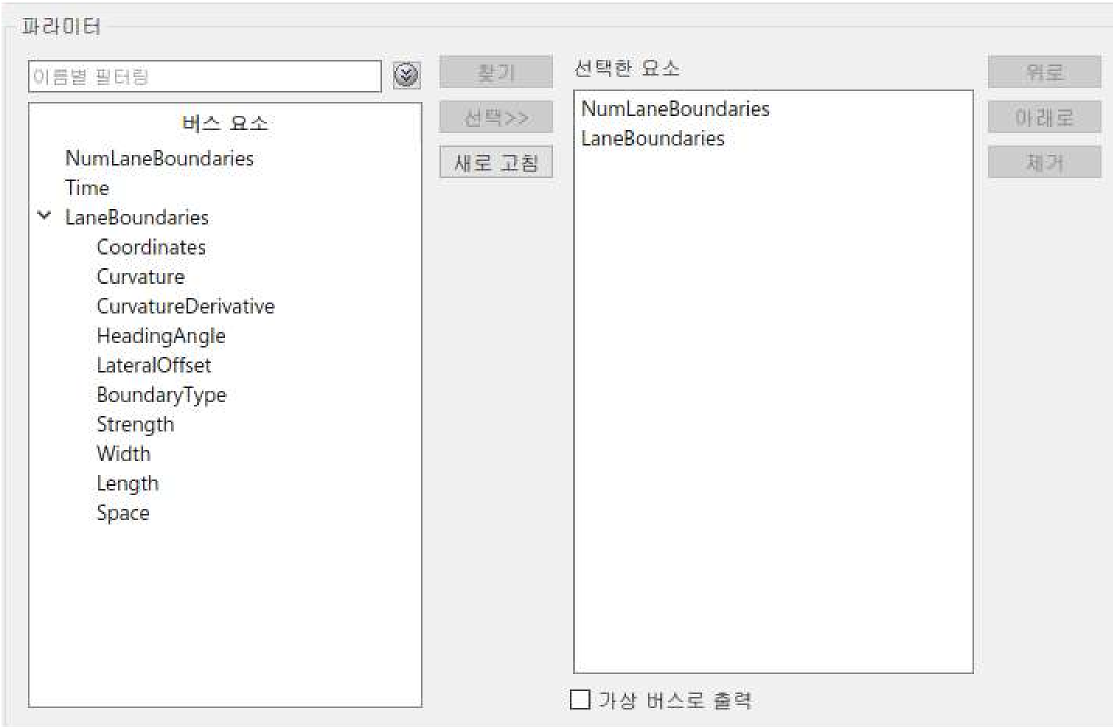
Simulink Bus

- Bus Selector



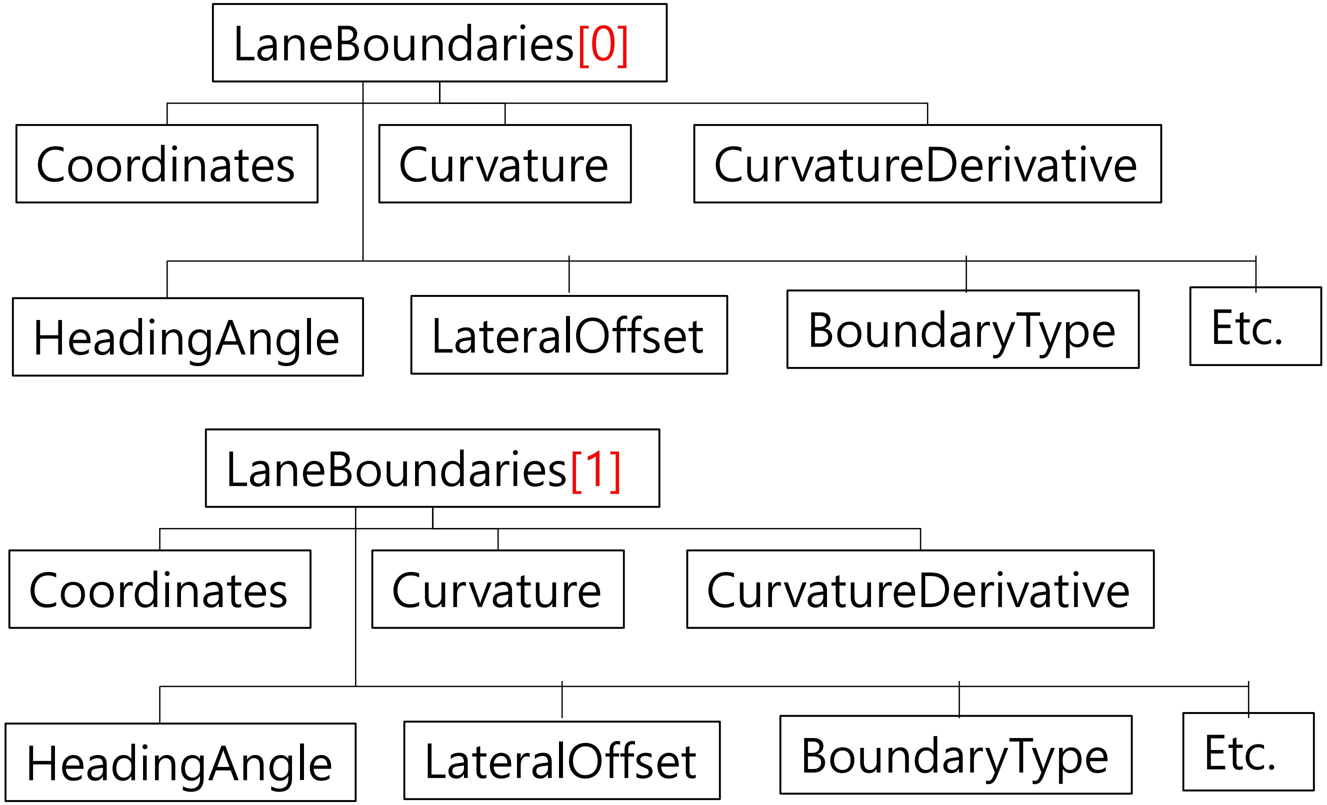
Scenario Reader Block

- ‘LaneBoundaries bus contain left and right lane information.
- There are two buses (left and right lane) in the ‘LaneBoundaries’



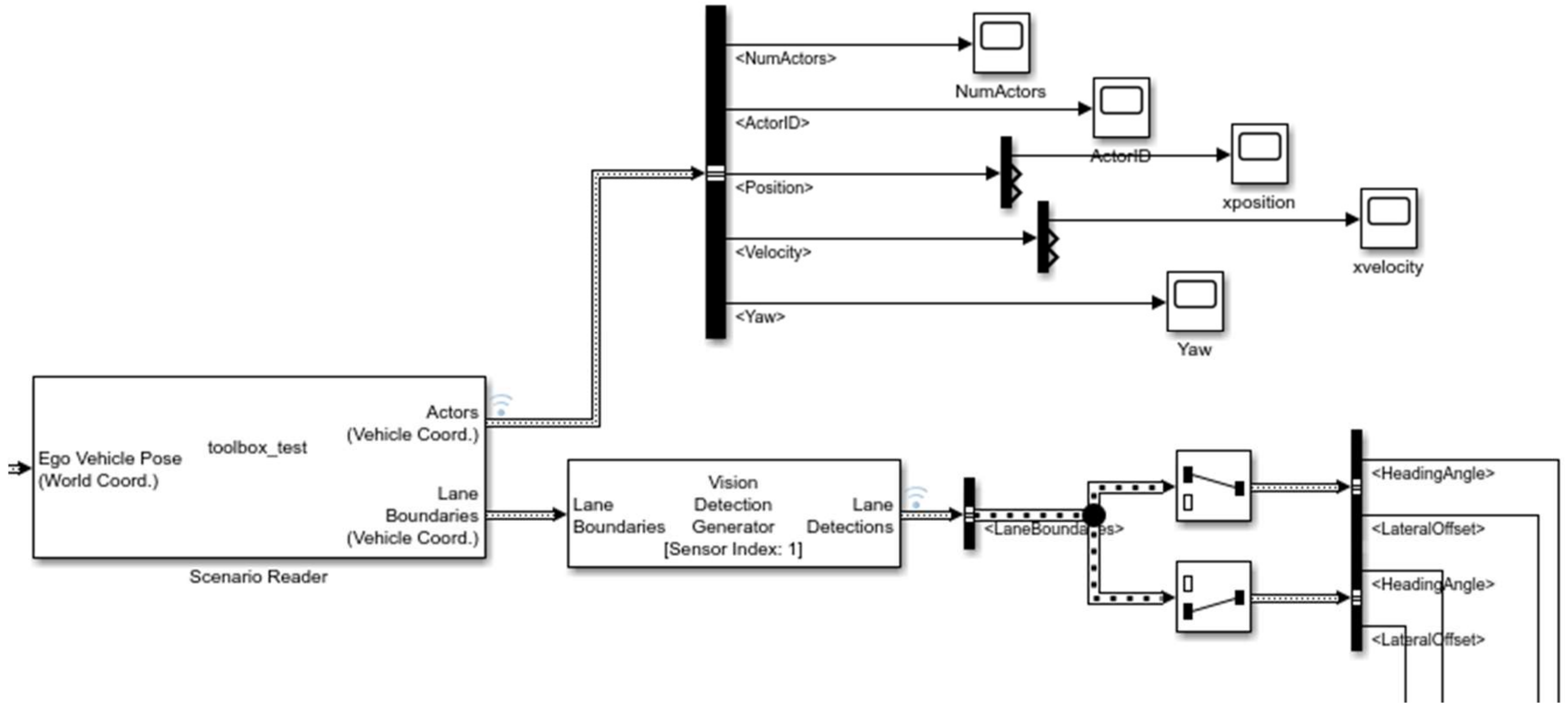
Scenario Reader Block

- Array of Structure in the LaneBoundaries Bus



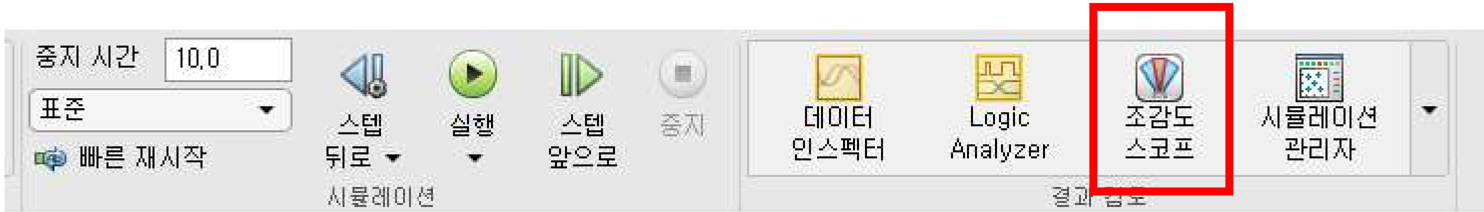
Scenario Reader Block

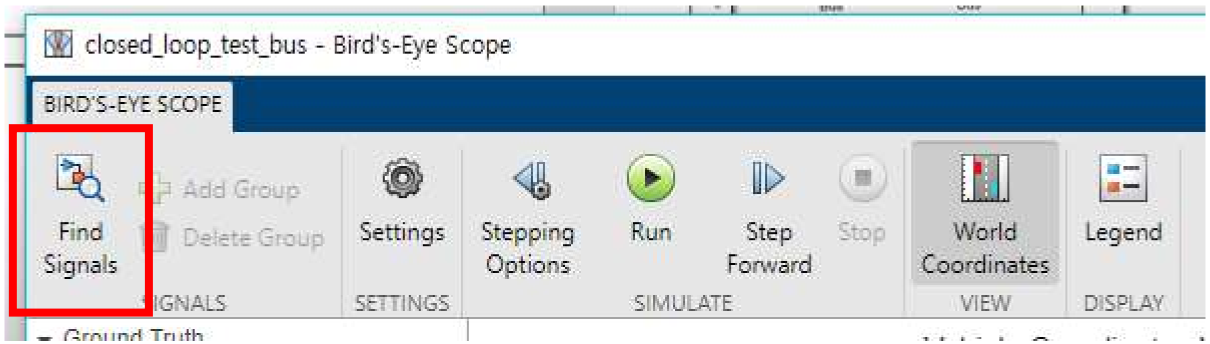
- In order to ungroup the buses, use 'Select' block first, and then use 'Bus Selector' to extract each lane info.
- The same technique will be necessary when there are multiple actors in the scenario.



Visualize Simulation

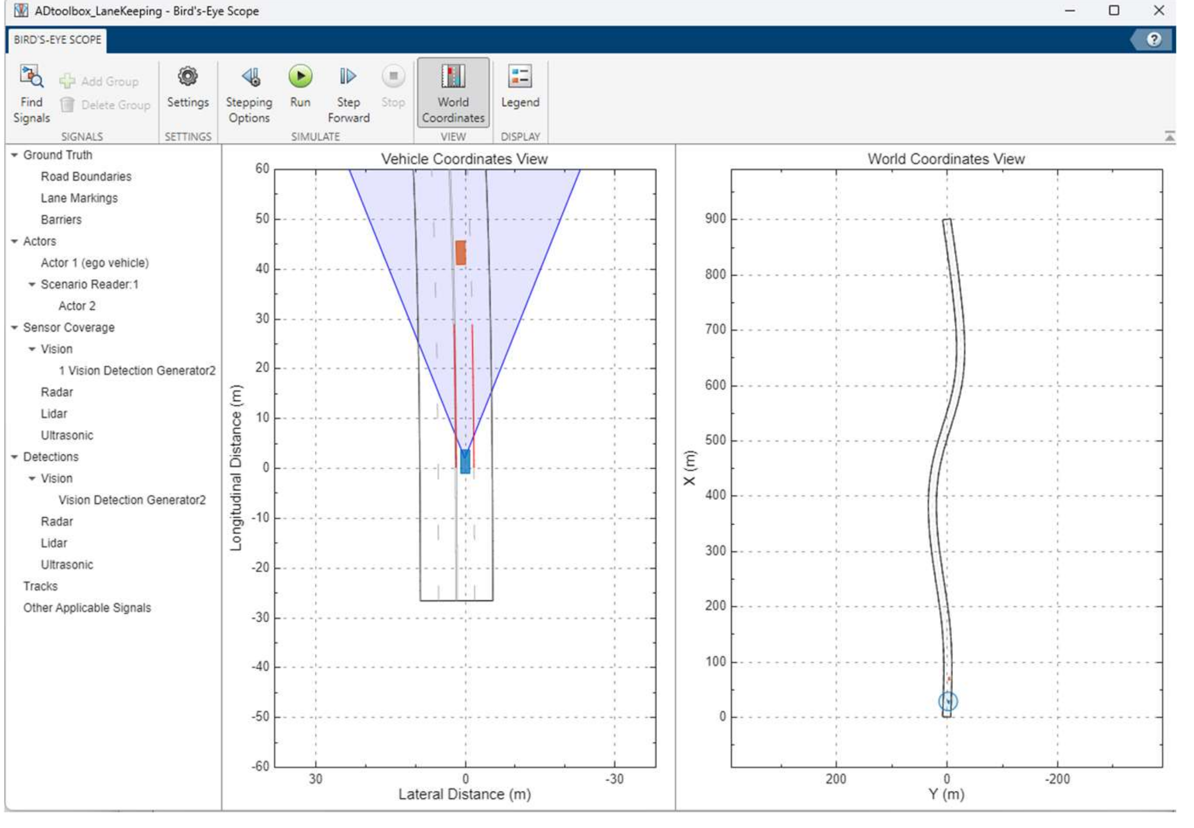
- To visualize the scenario, use the Bird's-Eye Scope
- From the Simulink toolstrip, under Review Results, click Bird's-Eye Scope(조감도 스코프). Then, in the scope, click Find Signals and run the simulation.





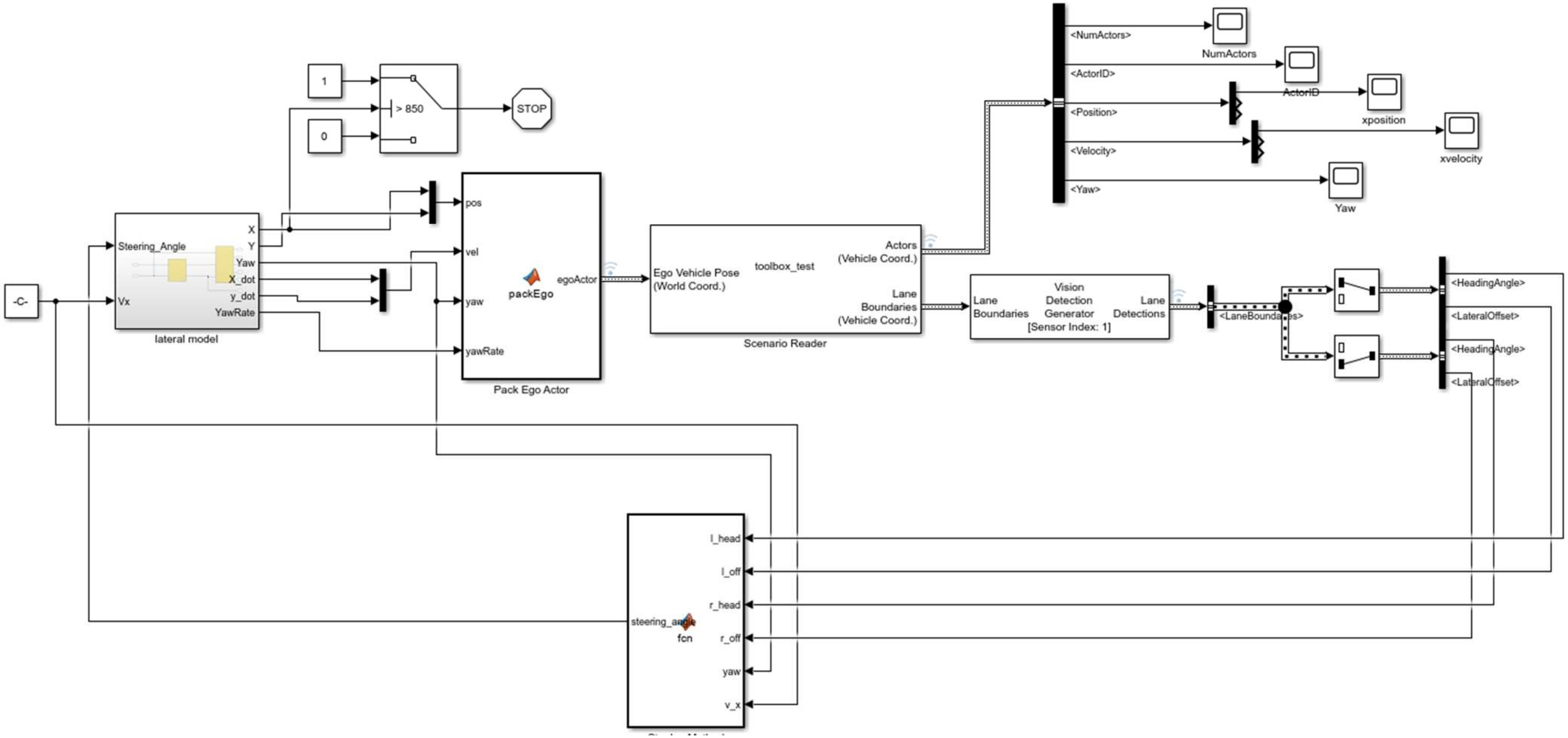
Visualize Simulation

- Click run and watch the simulation in both “vehicle coordinates view” and “world coordinates view”. (This will slow down you computer…)



Test Closed-loop ADAS Algorithm Using Driving Scenario

- Sample Code
  - ➢ADToolbox_LaneKeeping.slx



Test Closed-loop ADAS Algorithm Using Driving Scenario

- Sample Code "ADtoolbox_LaneKeeping.slx"
  - This is an open loop code, but you can easily implement your control algorithm for LKAS or ACC.
  - Run vehicle_lateral_model_setup.m file for the parameter setup.

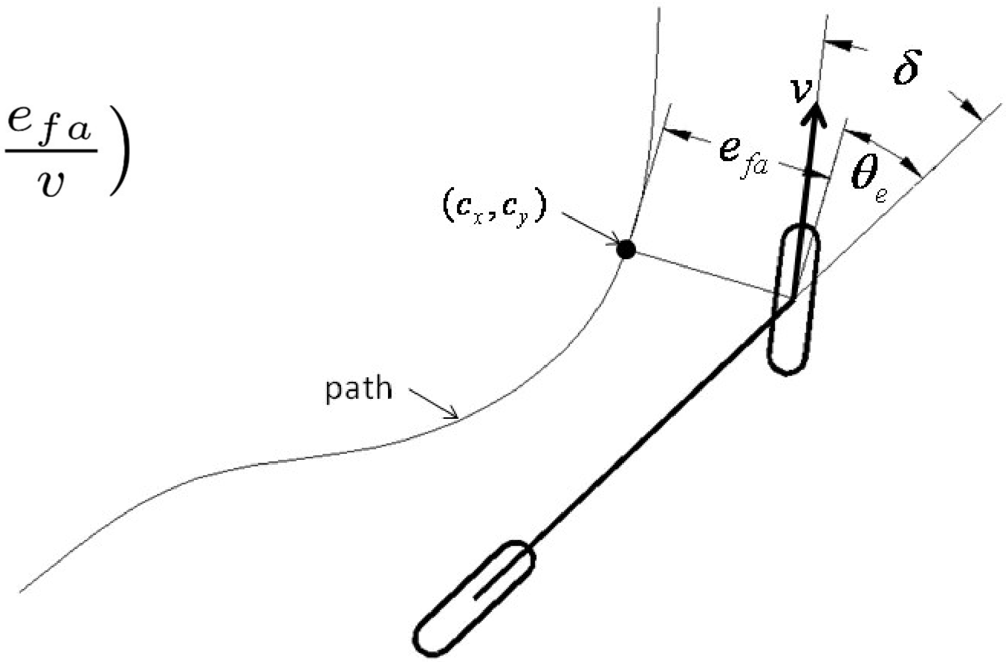
Test Closed-loop ADAS Algorithm Using Driving Scenario

- Employed Stanley Method for Lane Keeping Controller

  - ➢Steering Input

$$\delta = \theta_e + \tan^{-1}\left(K\frac{e_{fa}}{v}\right)$$

- K : control gain



Test Closed-loop ADAS Algorithm Using Driving Scenario

- Pack Ego Actor Block
  - ➢Pack ego vehicle information into a single ego actor bus
  - ➢This will be given to the scenario reader block as a ego vehicle trajectory

Test Closed-loop ADAS Algorithm Using Driving Scenario

- If the following Error occurs, then reselect the scenario file in the scenario reader block and press “적용”.

구성요소: Simulink | 범주: Block 경고

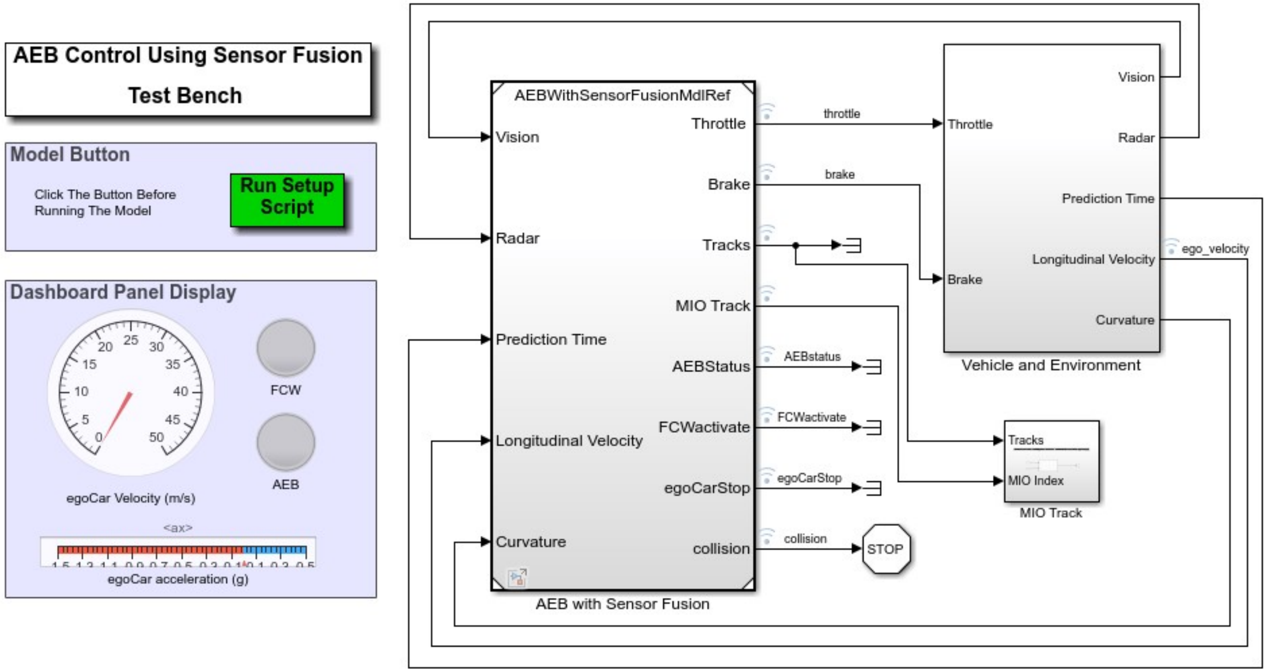
The signal connected to the input port of 'closed loop driving scenario 2021a/Bus Selector' is not a bus signal. The input to the Bus Selector block must be a virtual or nonvirtual bus signal.
A possible cause of this error is the use of a bus-capable block (such as Merge or Unit Delay) that in this current situation is unable to propagate the bus downstream to the block reported in this error. Please see Simulink documentation for further information on composite (i.e. bus) signals and their proper usage.

구성요소: Simulink | 범주: Block 오류

Test Closed-loop ADAS Algorithm Using Driving Scenario

- AEB Testbench Example
  - ➢Open the model by typing

    >> open_system('AEBTestBenchExample')



Test Closed-loop ADAS Algorithm Using Driving Scenario

- AEB Testbench Example
  - ➢This example shows how to model closed-loop control algorithm working with Automated Driving Toolbox.
  - ➢However, “Tracking and Sensor Fusion” block requires deeper understanding about sensor data analysis.
  - ➢In this class, it is recommended to use the output from the “Scenario Reader block” to get the information about the Actors and Lanes.

Test Closed-loop ADAS Algorithm Using Driving Scenario

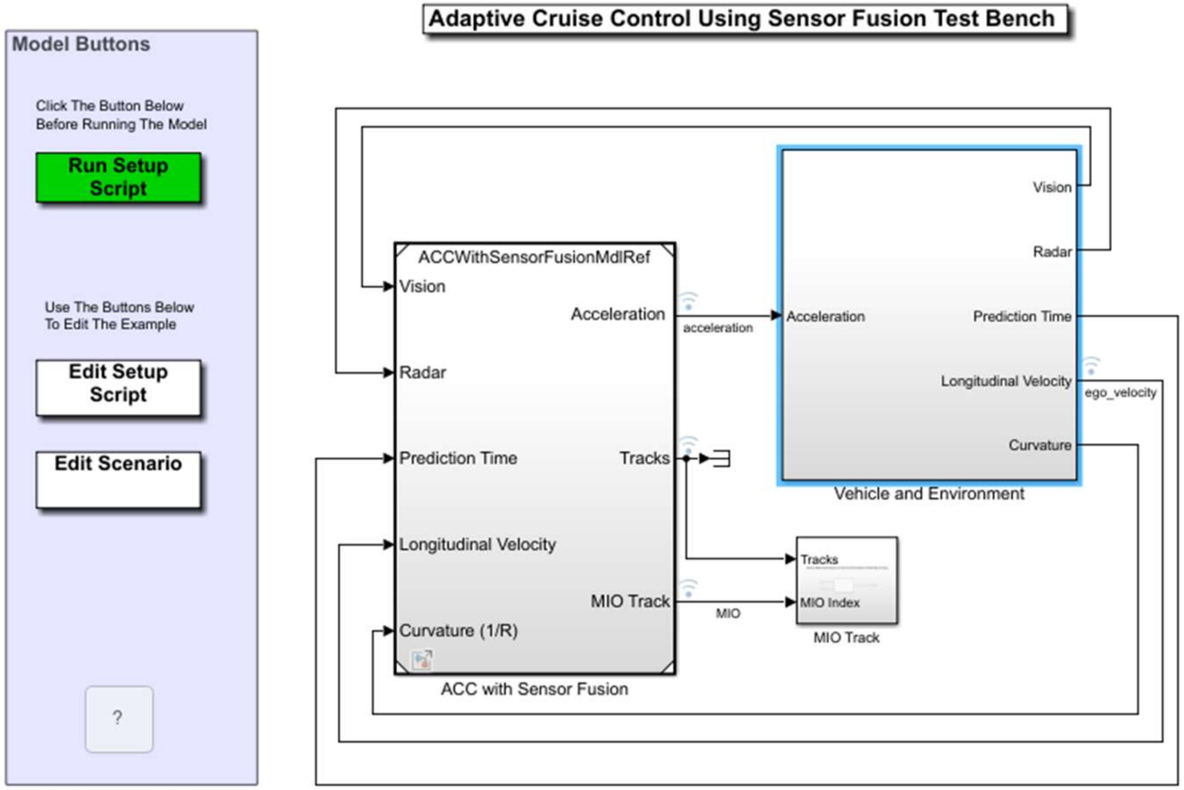
- helperAEBSetUp.m 파일을 이용하여 다양한 조건에서 AEB 시험 가능
- 아래와 같이 85번째 라인에 “+1” 을 추가하여 시뮬레이션 진행
  - SimParams.v_set = norm(egoVehicle.Velocity) +1; % set ego velocity (m/s)

- Ego Vehicle 속도가 높아짐에 따라서 충돌회피 실패

Test Closed-loop ADAS Algorithm Using Driving Scenario

- ACC TestBench Example

  >>open_system('ACCTestBenchExample')



Test Closed-loop ADAS Algorithm Using Driving Scenario

- Lane Following Testbench Example

  `>> open_system('LaneFollowingTestBenchExample')`