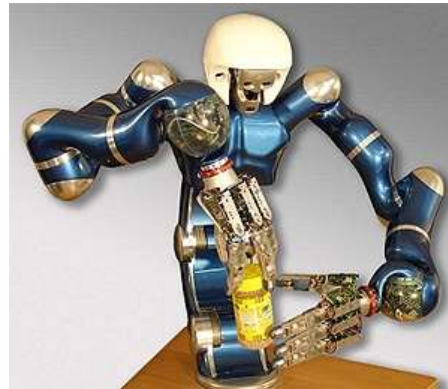


# Control System Design for Automated Driving

## Lecture 11



# Automated Driving Toolbox

- 자율주행 시스템 설계, 시뮬레이션 및 테스트를 위한 알고리즘 및 Tool 을 제공
  - Ground Truth Labeling
  - Tracking and Sensor Fusion
  - Planning Mapping, and Control
  - Cuboid Driving Scenario Simulation
  - 3D Simulation using Unreal Engine
- This lecture note is based on the "Automated Driving Toolbox User Guide" from Mathworks.

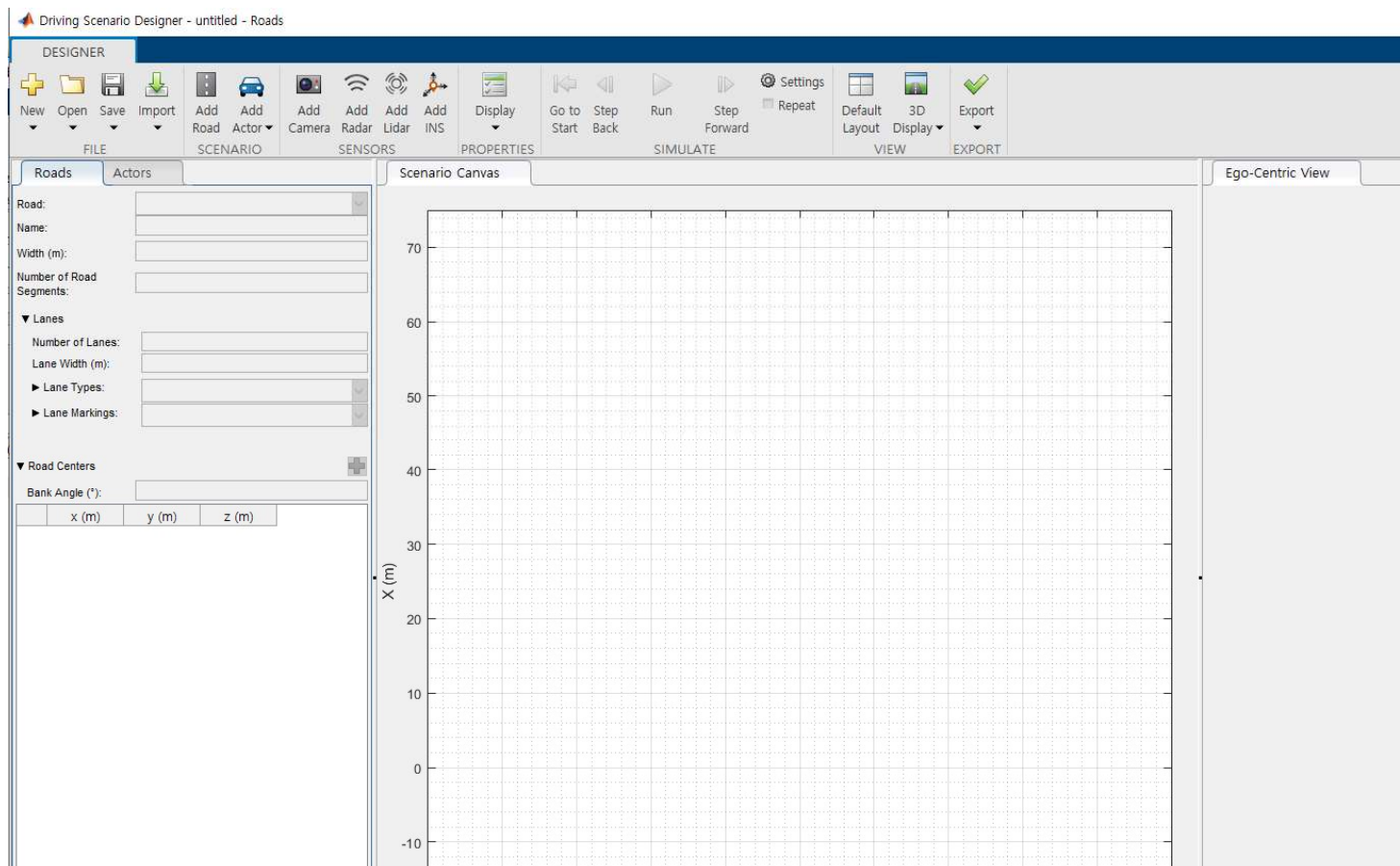
# Cuboid Driving Scenario Simulation

# Driving Scenario Designer

- 자율주행 시뮬레이션을 위한 시나리오 작성을 가능하게 해주는 Automated Driving Toolbox 기능
- 자율주행 시뮬레이션을 위한 도로, 타겟 차량, 보행자, 교차로 등을 간편하게 제작할 수 있는 사용자 인터페이스 제공
- Ego Vehicle에서 카메라, 라이다, 레이더 센서 추가 가능
- 제작된 '시나리오'와 '센서 데이터'를 시물링크로 전달할 수 있는 Scenario Reader Block 및 센서블럭 생성 지원

# Create Driving Scenario

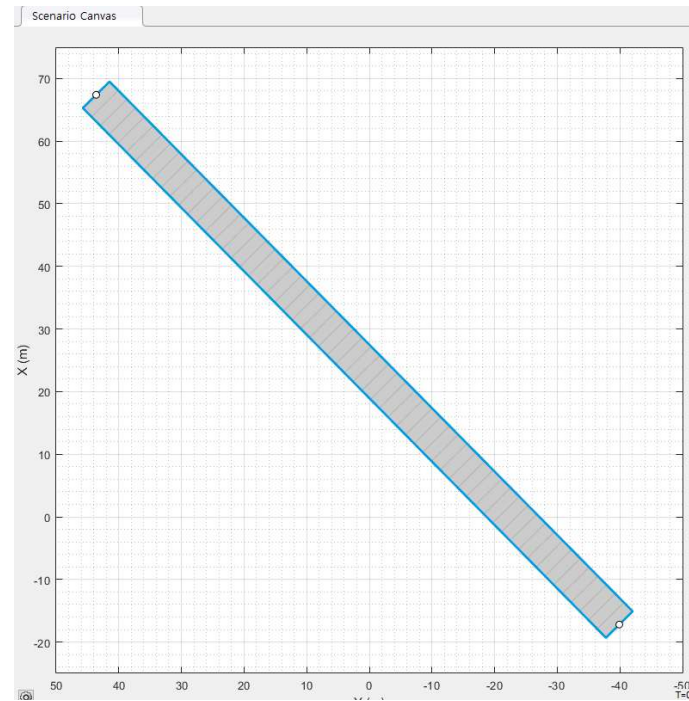
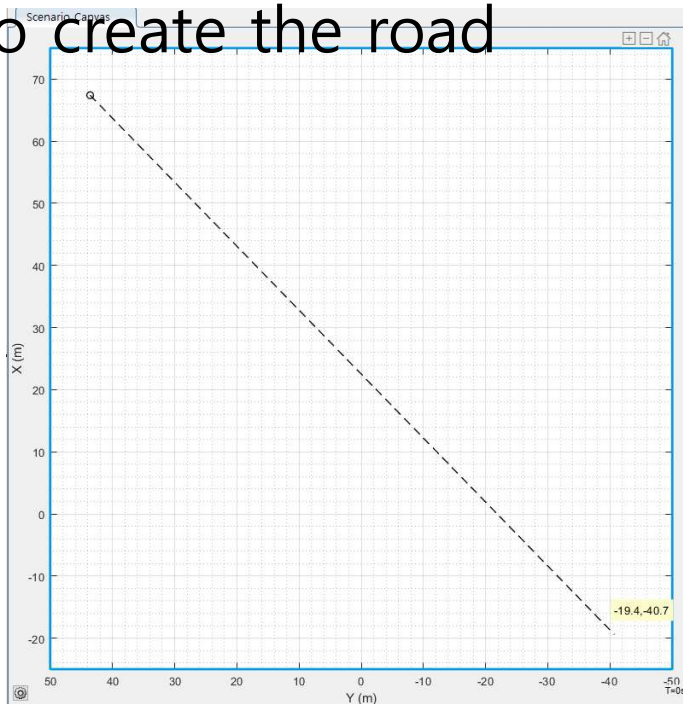
- To open the app, at the MATLAB command prompt, enter 'drivingScenarioDesigner'



# Add a Road



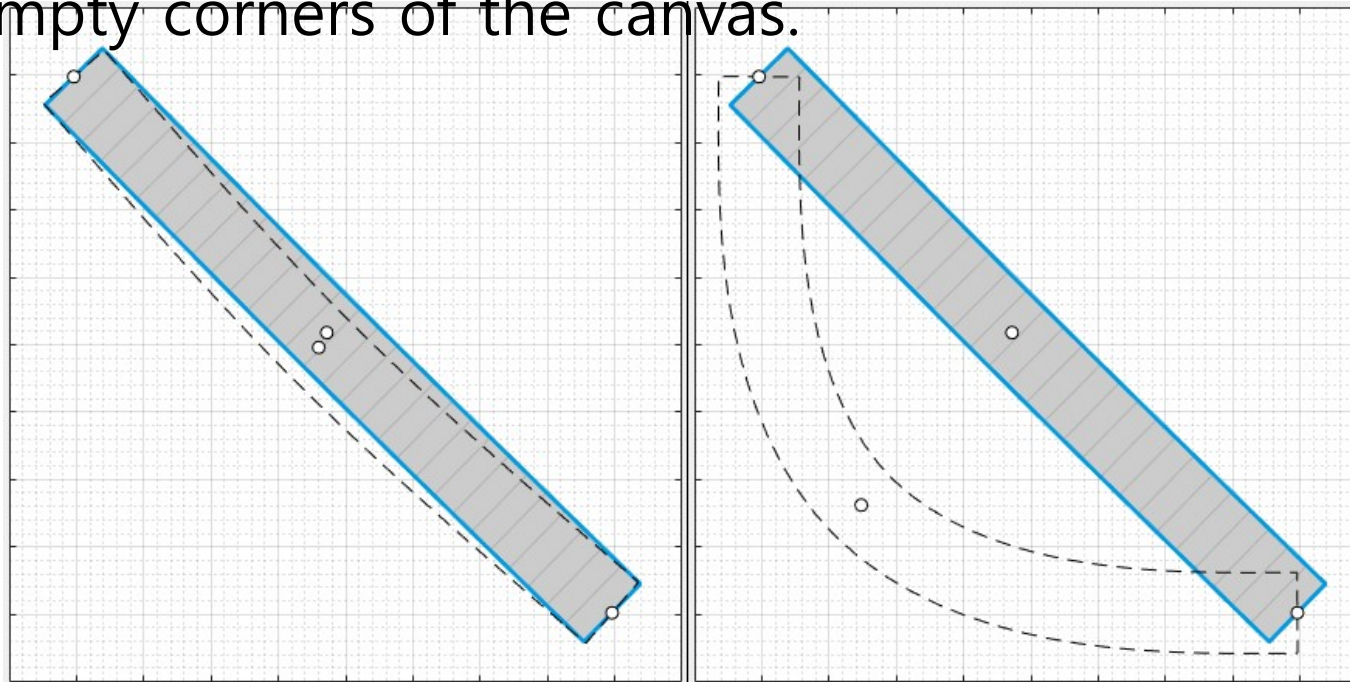
- Add a curved road to the scenario canvas. On the app toolstrip, click Add Road.
- Then click one corner of the canvas, extend the road to the opposite corner, and double-click the canvas to create the road



# Add a Road



- To make the road curve, add a road center around which to curve it. Right-click the middle of the road and select Add Road Center.
- Then drag the added road center to one of the empty corners of the canvas.

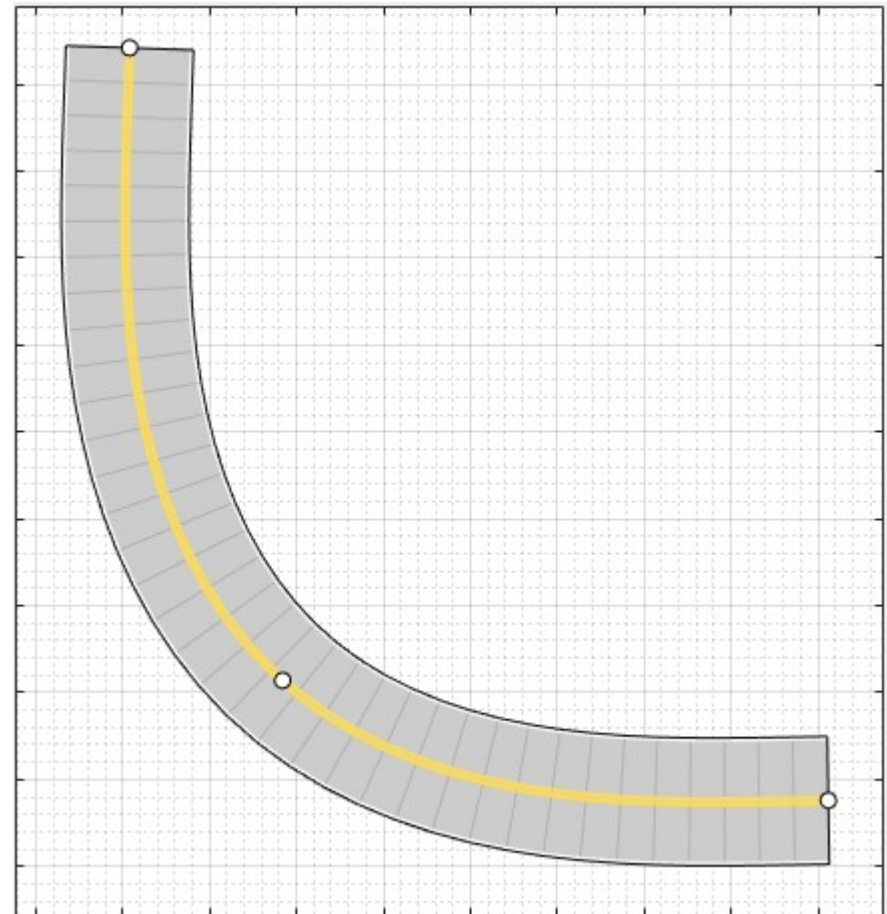




# Add Lanes

▼ Lanes

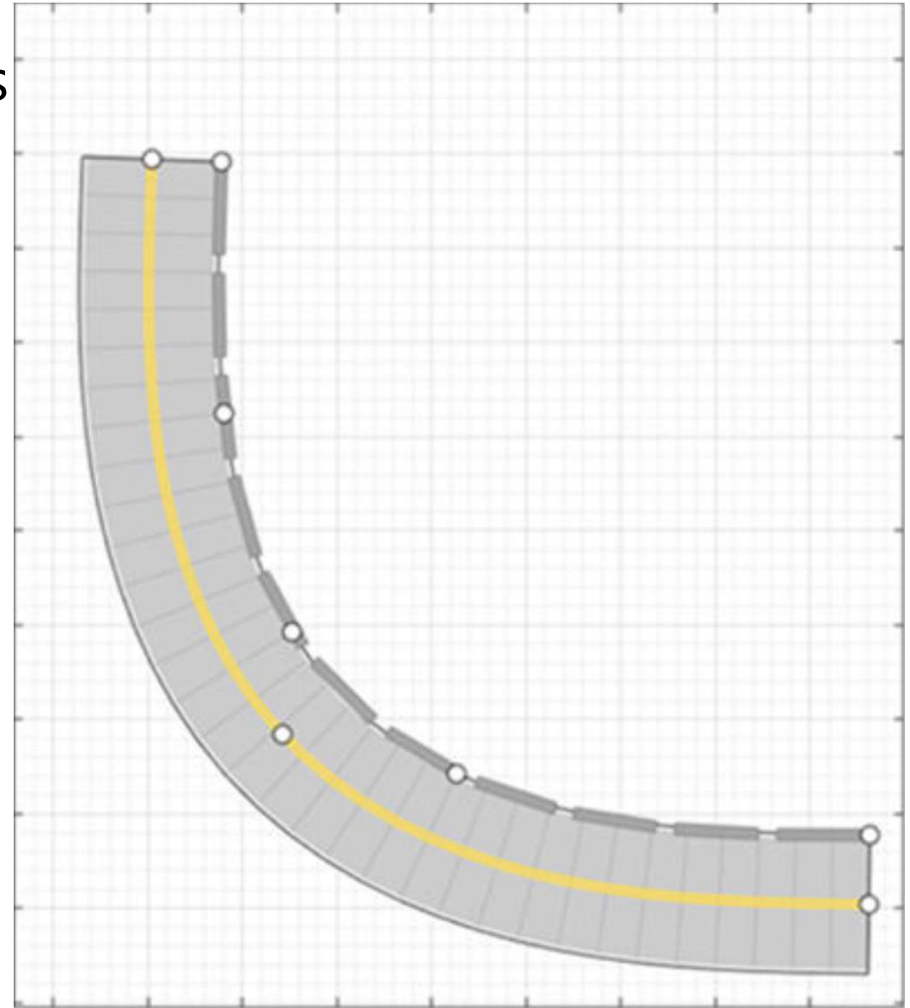
- In the left pane, on the Roads tab, expand the Lanes section. Set the Number of lanes to [1 1].





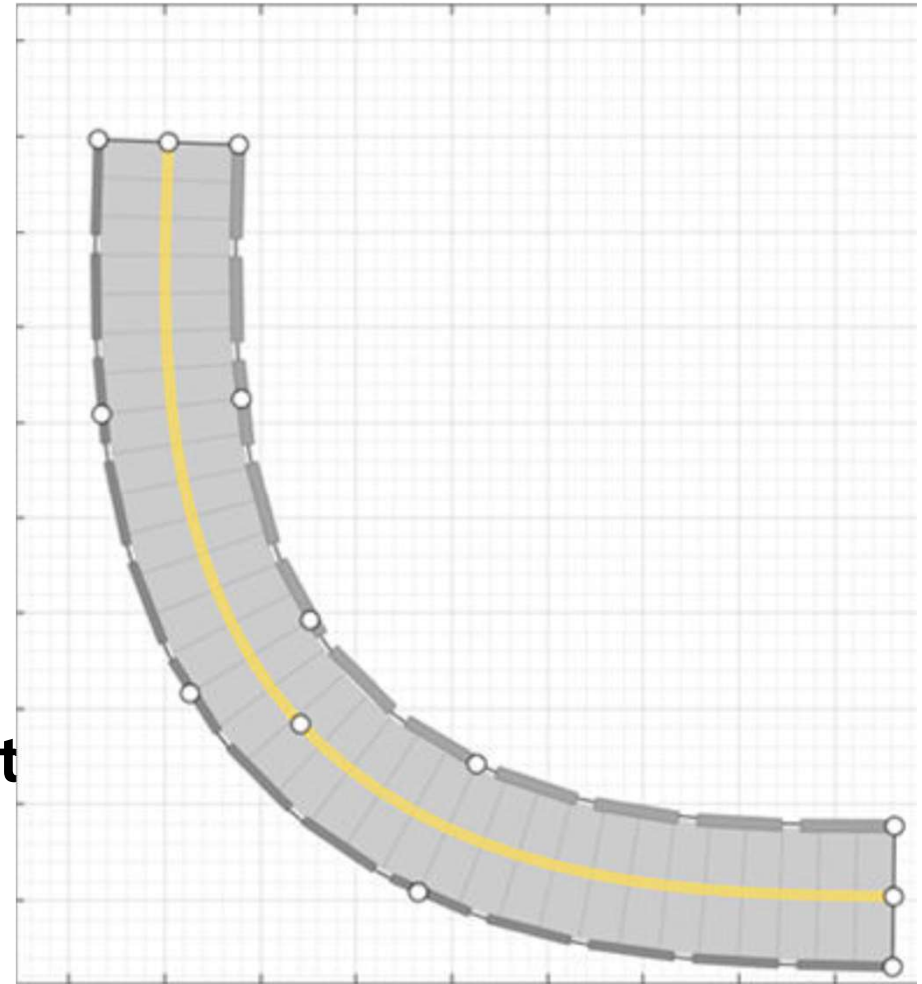
# Add Barriers

- To add barriers along the edges of the curved road,
- On the app toolstrip, click **Add Actor > Jersey Barrier**.
- Move the cursor to the right edge of the road and click to add a barrier along it.
- To add a gap of 1m between the barrier segments change the value of the **Segment Gap (m)** property in the **Barriers** tab to 1.



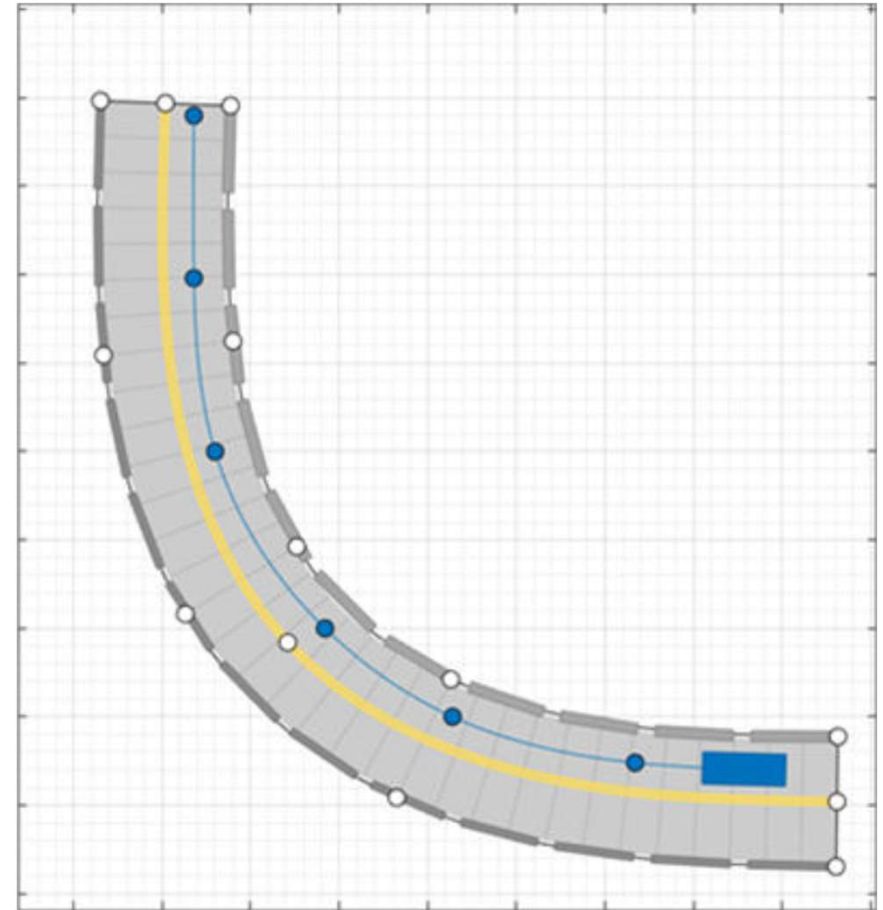
# Add Guardrails

- To add a guardrail barrier to the left edge of the road,
- Right click on the road and select **Add Guardrail > Left edge**.
- Specify a 1m gap between the barrier segments for the
- guardrail, using the **Segment Gap (m)** property in the **Barriers** tab



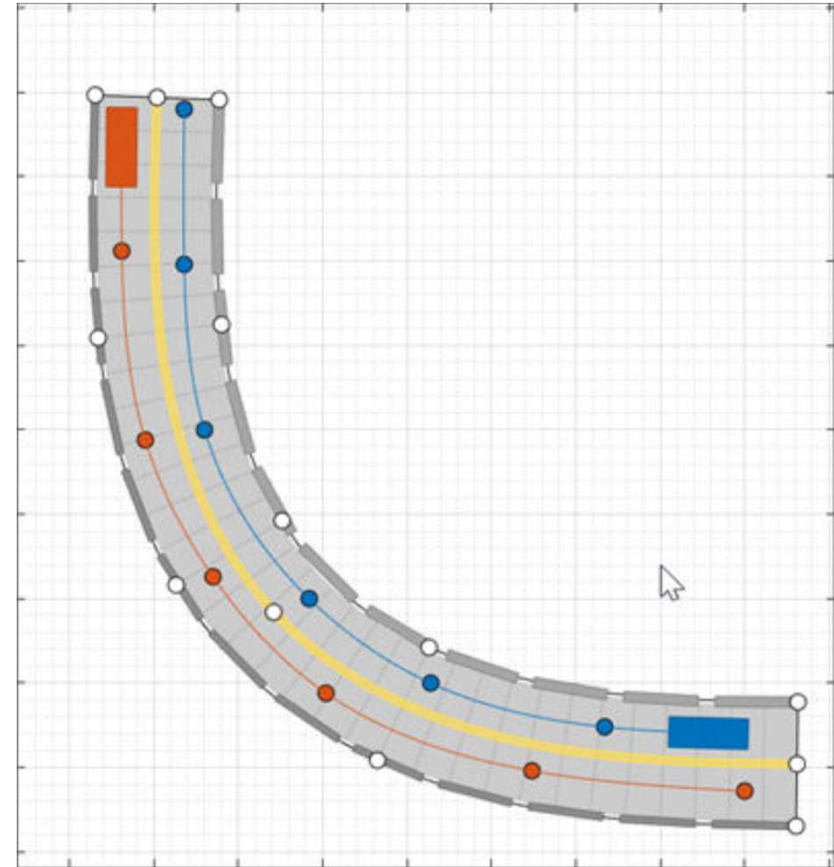
# Add Ego Vehicle

- To add the ego vehicle, right click one end of the road, and select **Add Car**.
- To specify the trajectory of the car, right-click the car, select **Add Waypoints** and add way point along the road.
- After you add the last waypoint along the road, press **Enter**.
- To adjust the speed of the car, set constant speed to 15 m/s on the Actors tab in the left pane.



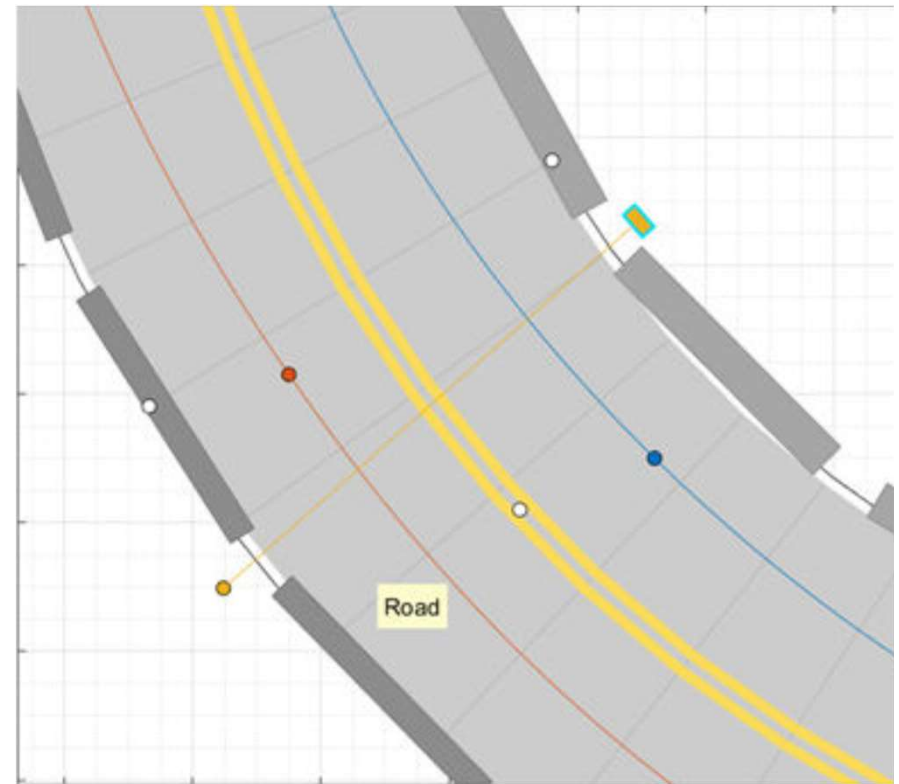
## Add Second Car

- Add a vehicle for the ego vehicle sensors to detect.
- On the app toolstrip, click **Add Actor** and select **Car**.
- Add the second car with waypoints, driving in the lane apposite from the ego vehicle.



# Add Pedestrian

- Add to the scenario, a pedestrian crossing the road.
- Zoom in on the middle of the road, right-click one side of the road, and click **Add Pedestrian**.
- Then, to set the path of the pedestrian, add a waypoint on the other side of the road.
- To test the speed of the cars, and the pedestrian, run the simulation. Adjust actor speeds as needed.



# 자율주행 자동차의 환경인식

- 카메라 센서



<차선 이탈 방지 시스템>



<MobileEye, 보행자 감지 시스템>



# 자율주행 자동차의 환경인식

- 이미지 센서 정보의 불확실성
  - 야간, 역광 등 일광 조건
  - 눈, 안개, 비 등의 기상 조건
  - 공사중이거나 차선이 희미한 도로
  - 주변의 혼동되는 이미지 등



역광 조건



기상 조건



공사중인 도로



희미한 차선

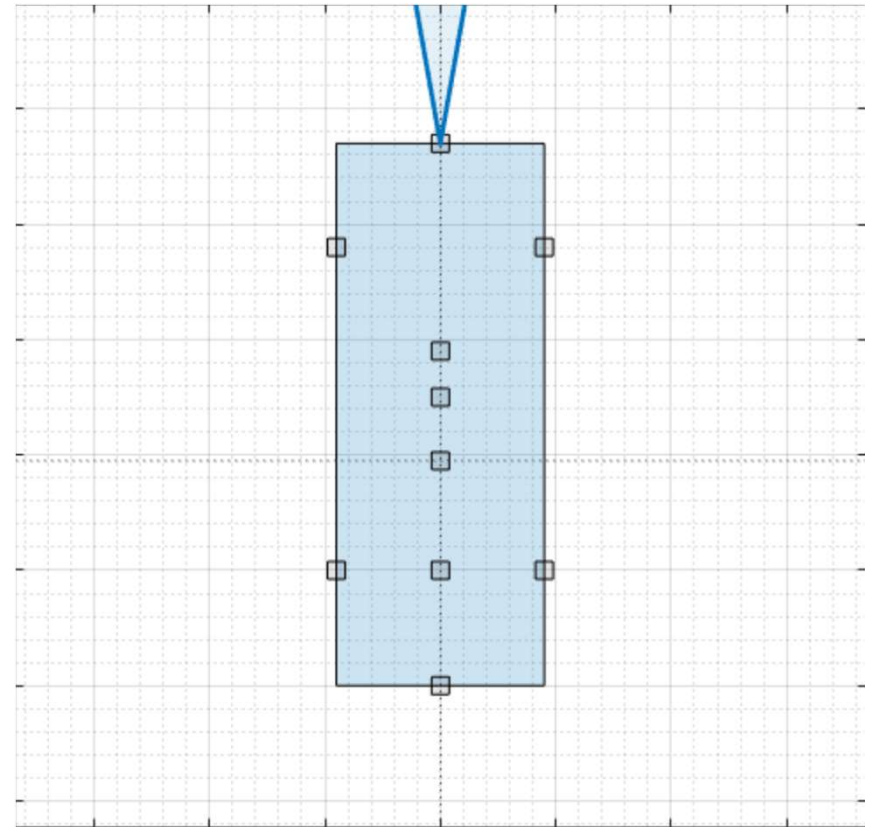


트럭에 반사된 이미지



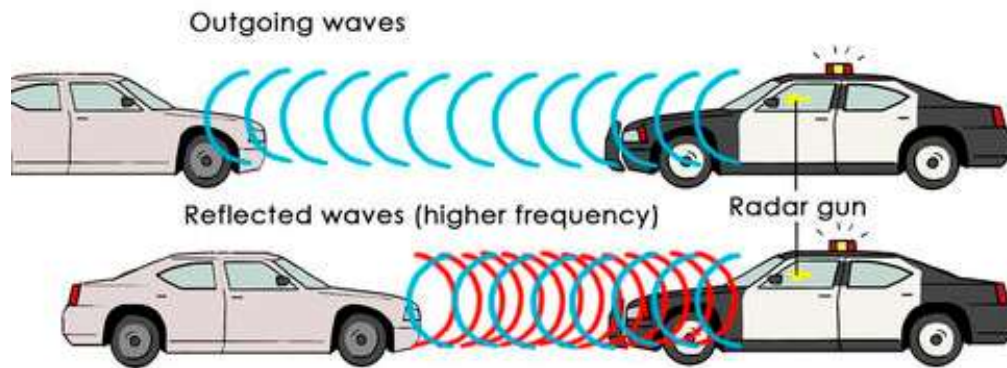
# Add Camera

- On the app toolbar, click **Add Camera**.
- The sensor canvas shows standard locations at which to place sensors.
- Click the frontmost predefined sensor location to add a camera sensor to the front bumper of the ego vehicle.
- By default, the camera detects only actors and not lanes. To enable lane detections, on the sensors tab in the left pane, expand the **Sensor Parameters**, and set **Detection Type** to *Objects & Lanes*.

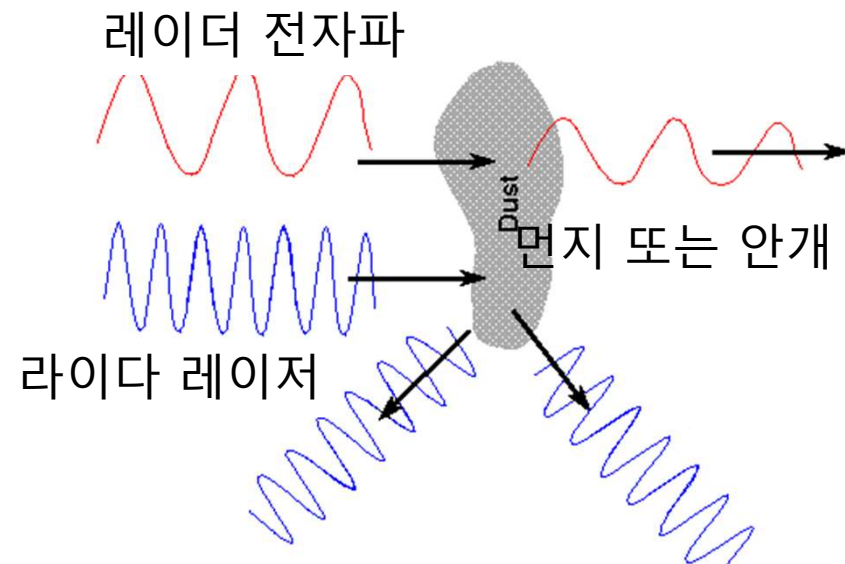


# 자율주행 자동차의 환경인식

- 레이더 센서



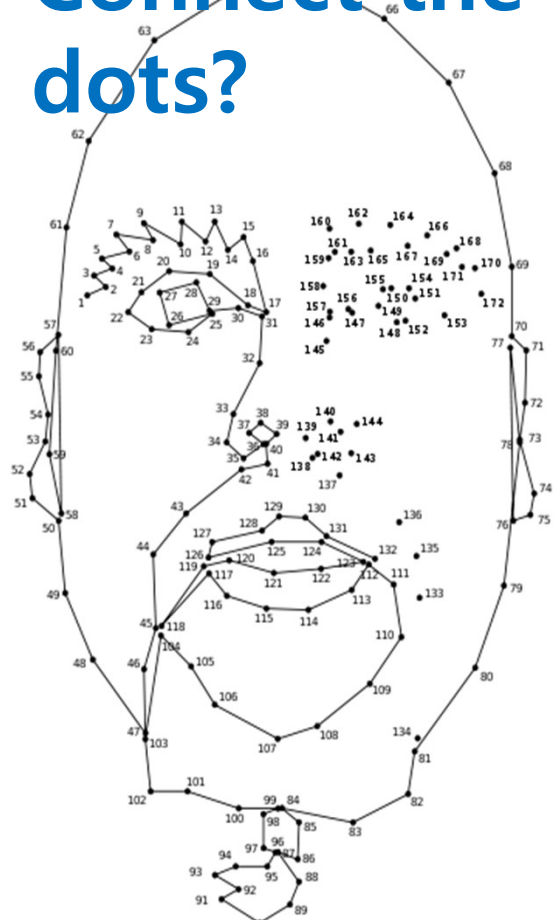
<도플러 효과>



# 자율주행 자동차의 환경인식

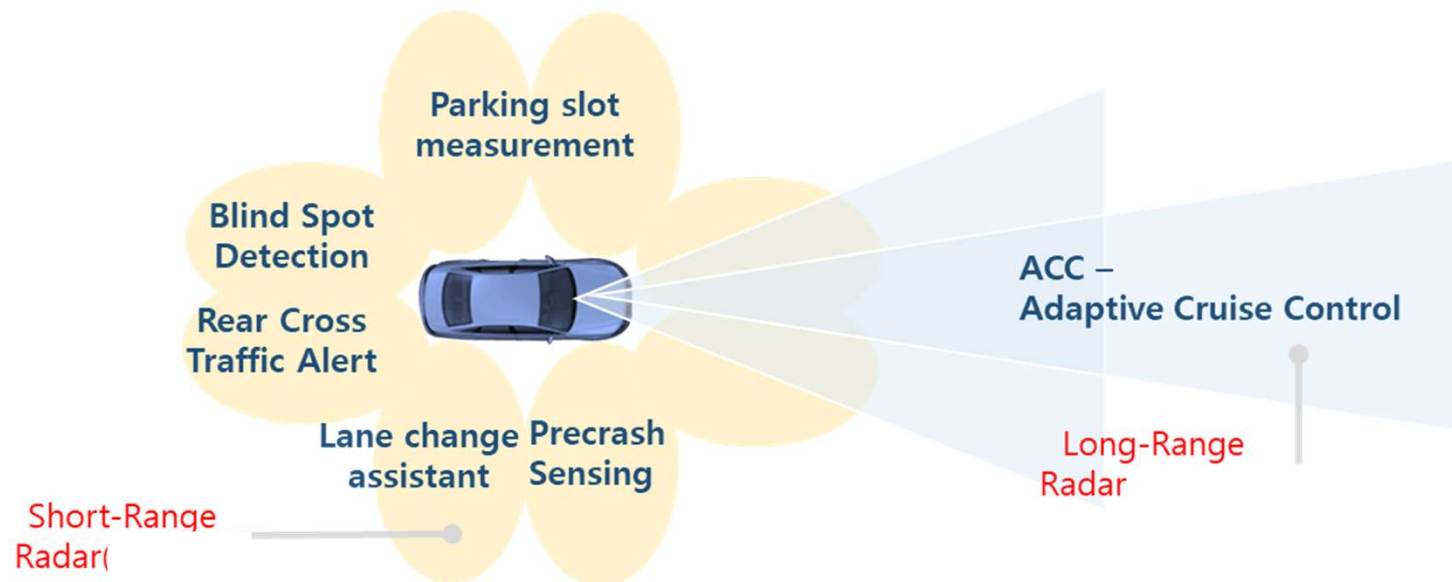
- 레이더 센서

Connect the dots?



# 자율주행 자동차의 환경인식

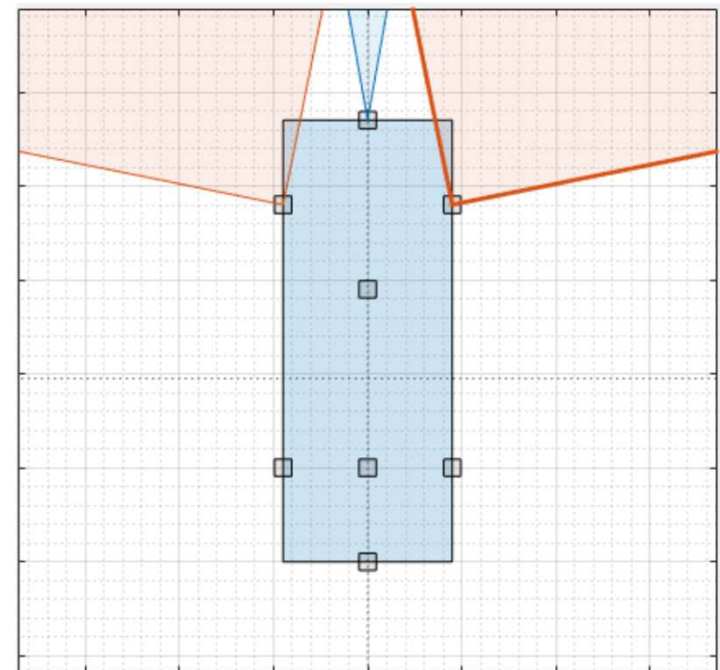
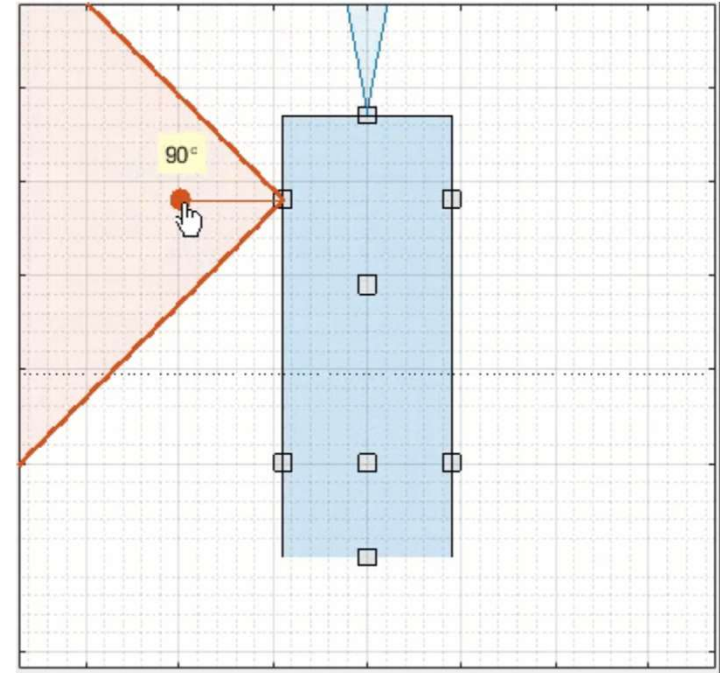
- 레이더 센서





# Add Radar

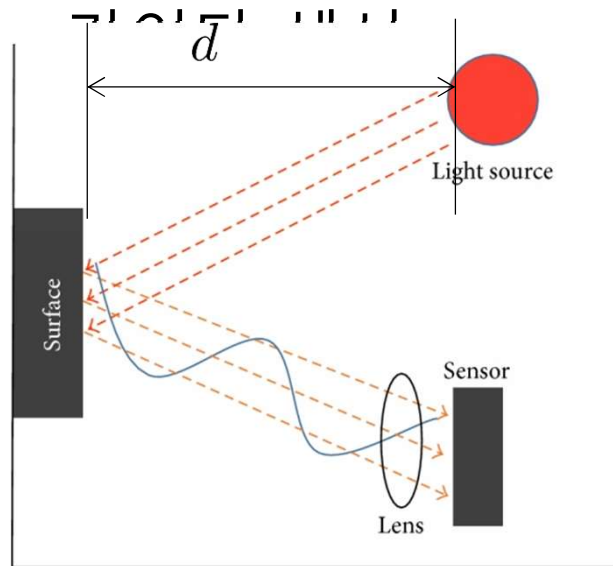
- Snap a radar sensor to the front-left wheel. Right-click the predefined sensor location for the wheel and select **Add Radar**.
- Tilt the radar sensor toward the front of the car. Move your cursor over the coverage area, then click and drag the angle marking.
- Add an identical radar sensor to the front-right wheel. Right-click the sensor on the front-left wheel and click **Copy**.
- Then right-click on the front-right wheel and click **Paste**.



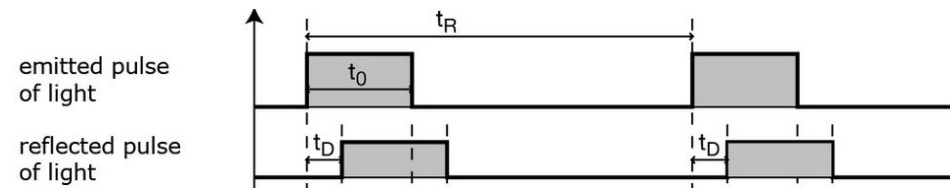
# 자율주행 자동차의 환경인식 센서



<라이다 센서>

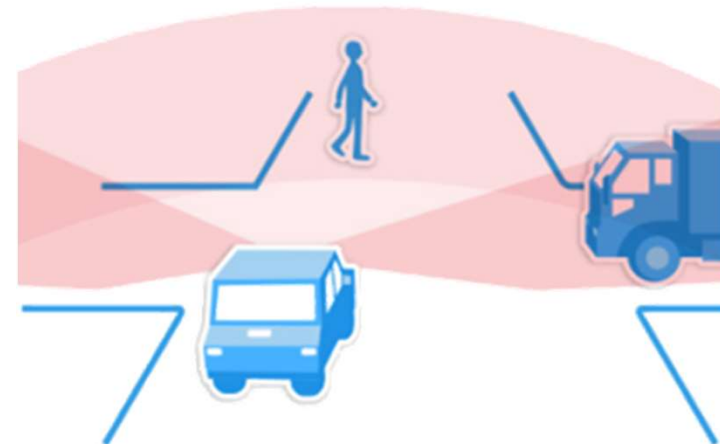


<라이다 센서를 거리측정 원리>



$$d = \frac{1}{2} c \times t_d$$

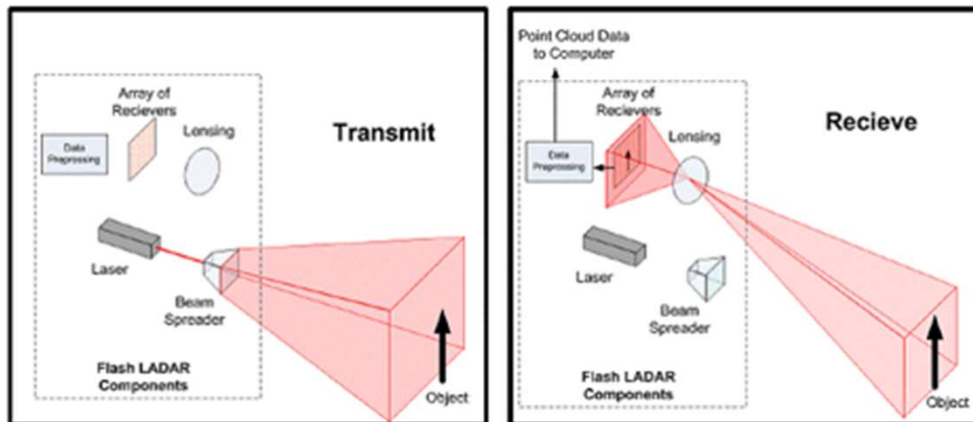
c : 빛의 속도  
 $t_d$  : 시간 지연  
 d : 물체까지의 거리



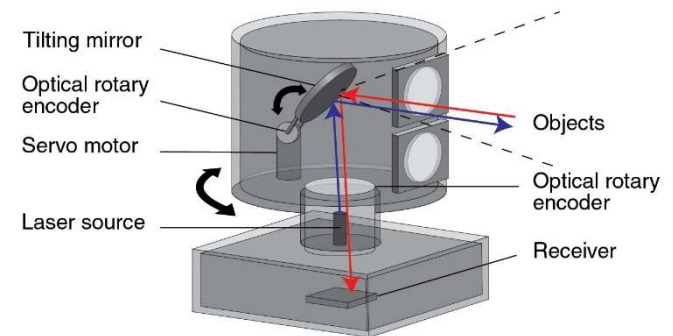
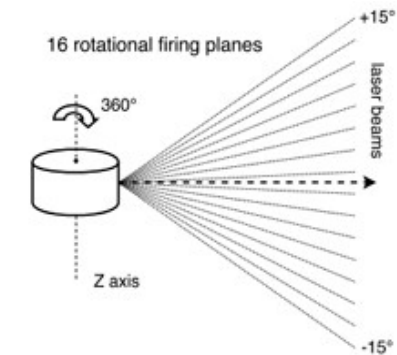
<라이다 센서를 이용한 주변 환경 측정>

# 자율주행 자동차의 환경인식 센서

- 라이다 센서



<Flash lidar>

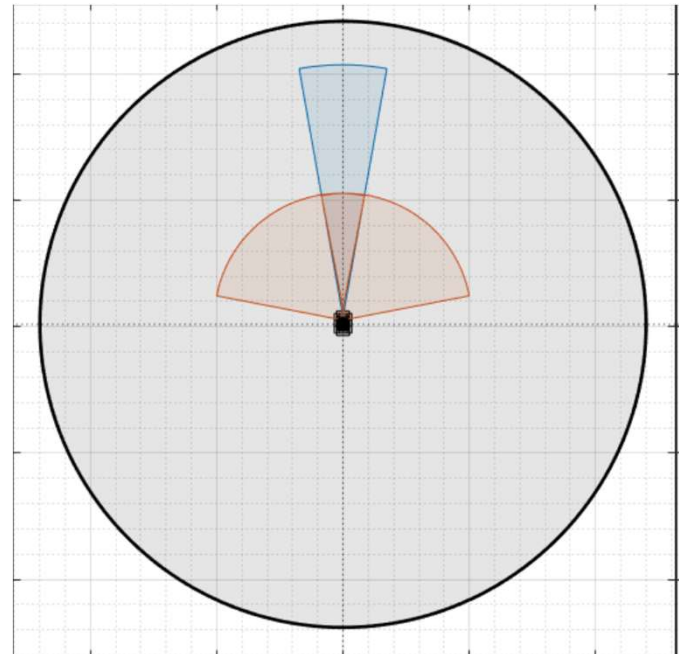
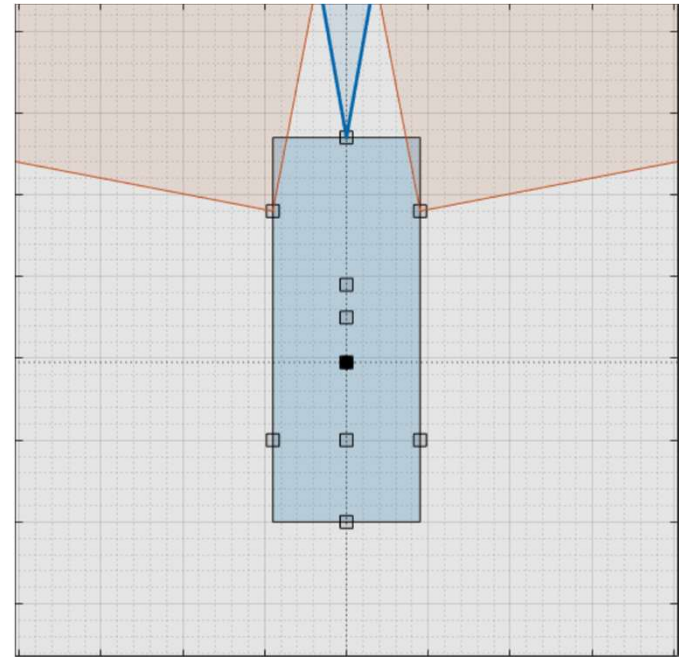


<Rotating mirror lidar>



# Add Lidar

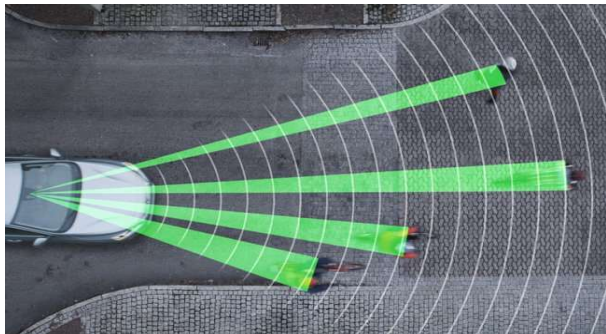
- Snap a lidar sensor to the center of the roof of the vehicle. Right-click the predefined sensor location for the roof center and select **Add Lidar**.
- The lidar sensor appears in black. The gray surrounding the vehicle is the coverage area of the lidar sensor. Zoom out to see the full view of the coverage areas.
- (**Warning!**) Adding Lidar sensor will significantly slow down your simulation!!!



# 자율주행 환경인식 센서의 장단점

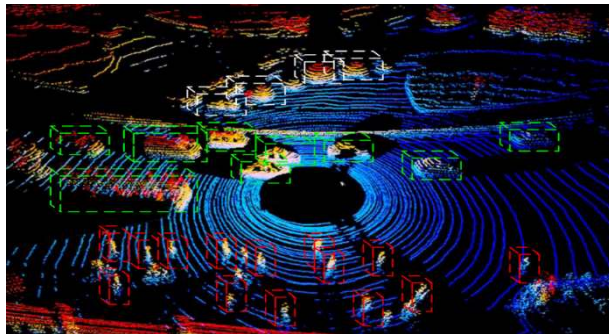
## 레이다

전방 차량 인식 및  
차량 상태(위치, 속도)



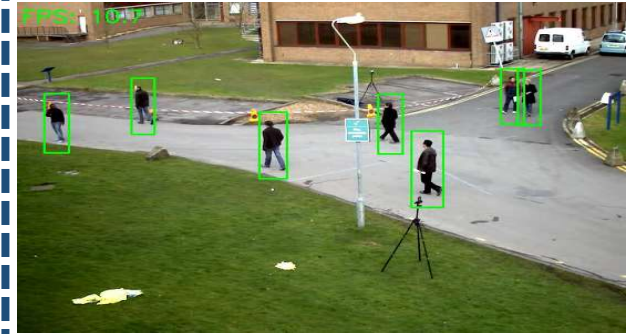
## 라이다

주변환경 인지 기술  
환경 지도구축



## 카메라

차량 및 보행자 검출  
대상의 속성 추정

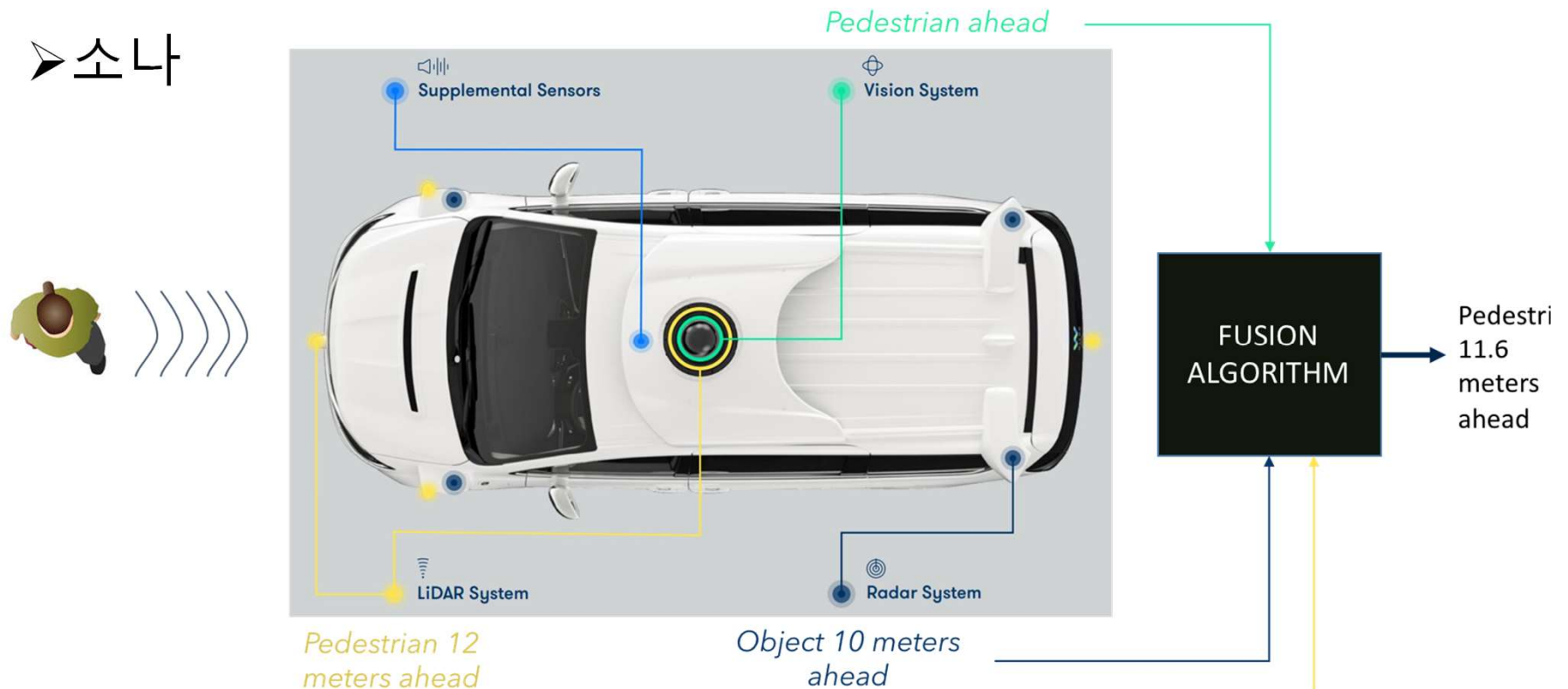


구 분	장 점	단 점
레이다	<ul style="list-style-type: none"> <li>거리 및 상대속도 측정성능 (종방향)</li> <li>외부환경에 강건</li> </ul>	<ul style="list-style-type: none"> <li>클러터 노이즈</li> <li>횡방향 정밀도 부족</li> </ul>
라이다	<ul style="list-style-type: none"> <li>외곽선 추정</li> <li>종/횡방향 해상도가 높음</li> </ul>	<ul style="list-style-type: none"> <li>날씨에 취약, 높은 가격</li> <li>처리할 데이터 양이 많음</li> </ul>
카메라	<ul style="list-style-type: none"> <li>대상 크기 및 속성 추정</li> <li>횡방향 위치 추정 성능</li> </ul>	<ul style="list-style-type: none"> <li>외부환경에 취약</li> <li>종방향 정밀도 부족, 데이터 처리 양</li> </ul>

# 자율주행 인지 기술

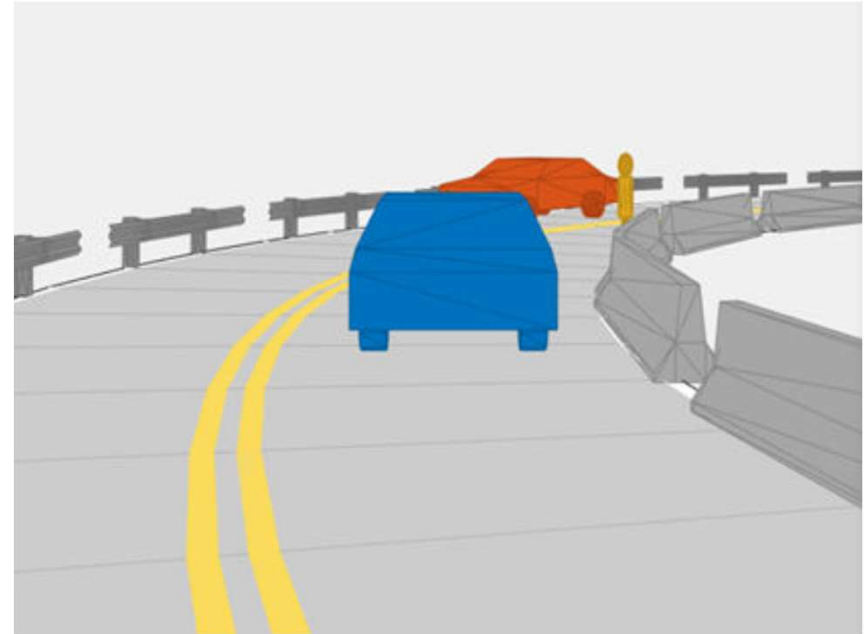
- 센서 융합

- 카메라 센서
- 레이다 센서
- 라이다 센서
- 소나



# Generate Synthetic Sensor Data

- To generate data from the sensors, click Run. As the scenario runs, the Bird's-Eye Plot displays the detections and point cloud data.
- The Ego-Centric View displays the scenario from the perspective of the ego vehicle.
- To export sensor data to the MATLAB workspace, on the app toolstrip, select Export>Export Sensor Data. Name the workspace variable and click **OK**.
- To export a MATLAB function that generates the scenario and its sensor data, select **Export>Export Matlab Function**. This function returns the sensor data as a structure, the scenario as a drivingScenario objects, and the sensor models as System objects.



# Save Scenario

- After you generate the detections, click Save to save the scenario file.
- You can also save the sensor models as separate files and save the road and actor models as a separate scenario file.
- You can reopen the scenario by using the exported drivingScenario object. By using the syntax,  

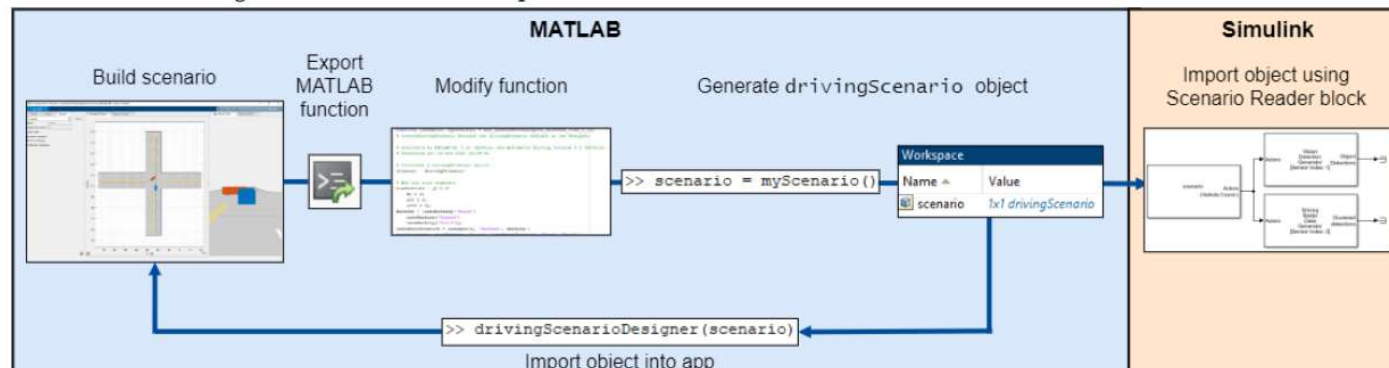
```
>> drivingScenarioDesigner(scenario)
```
- If you are developing a driving algorithm in Simulink, you can use a Scenario Reader block to read roads and actors from the scenario file or drivingScenario object into your model.
- To add sensors that you created in the app to a Simulink model, generate a model containing your scenario and sensors by selecting **Export>Export Simulink Model**.
- In the model the generated **Scenario Reader block** reads the scenario and the generated sensor blocks define the sensors.

# Create Driving Scenario Variations Programmatically

- This example shows how to programmatically create variations of a driving scenario that was built using the Driving Scenario Designer app
- Programmatically creating variations of a scenario enables you to rapidly test your driving algorithms under multiple conditions

# Create Driving Scenario Variations Programmatically

- To create programmatic variations of a driving scenario, follow these steps
  1. Interactively build a driving scenario by using the Driving Scenario Designer app.
  2. Export a MATLAB® function that generates the MATLAB code that is equivalent to this scenario.
  3. In the MATLAB Editor, modify the exported function to create variations of the original scenario.
  4. Call the function to generate a drivingScenario object that represents the scenario.
  5. Import the scenario object into the app to simulate the modified scenario or generate additional scenarios.





# Create Driving Scenario Variations Programmatically

- Build Scenario in App
  - Use the Driving Scenario Designer to interactively build a driving scenario on which to test your algorithms
  - Open the scenario file in the app.  
`drivingScenarioDesigner('LeftTurnScenarioNoSensors.mat')`

# Create Driving Scenario Variations Programmatically

- Export MATLAB Function of Scenario
  - From the Driving Scenario Designer app toolstrip, select Export > MATLAB Function.
  - The exported function contains the MATLAB code used to produce the scenario created in the app.

open `LeftTurnScenarioNoSensors.m`

```
function [scenario, egoVehicle] = LeftTurnScenarioNoSensors()  
% createDrivingScenario Returns the drivingScenario defined in the Designer
```

# Create Driving Scenario Variations Programmatically

- Export MATLAB Function of Scenario
  - Calling this function returns these aspects of the driving scenario.
    - scenario — Roads and actors of the scenarios, returned as a drivingScenario object.
    - egoVehicle — Ego vehicle defined in the scenario, returned as a Vehicle object. For details, see the vehicle function.
  - If your scenario contains sensors, then the returned function includes additional code for generating the sensors.

# Create Driving Scenario Variations Programmatically

- Modify Function to Create Scenario Variations
  - In the exported MATLAB function, the speed of the ego vehicle is set to a constant value of 10 meters per second (speed = 10).
  - In this modified function :
    - egoSpeed is included as an input argument
    - speed, the constant variable, is deleted.
    - To compute the ego vehicle trajectory, egoSpeed is used instead of speed.

```
function [scenario, egoVehicle] = LeftTurnScenarioNoSensors()  
% createDrivingScenario Returns the drivingScenario defined in the Designer  
  
...  
...  
...  
  
% Add the ego vehicle  
egoVehicle = vehicle(scenario, ...  
    'ClassID', 1, ...  
    'Position', [56 19 0]);  
waypoints = [56 19 0;  
    135 19 0];  
speed = 10;
```



```
function [scenario, egoVehicle] = LeftTurnScenarioNoSensors(egoSpeed)  
% createDrivingScenario Returns the drivingScenario defined in the Designer  
  
...  
...  
...  
  
% Add the ego vehicle  
egoVehicle = vehicle(scenario, ...  
    'ClassID', 1, ...  
    'Position', [56 19 0]);  
waypoints = [56 19 0;  
    135 19 0];  
speed = egoSpeed;  
smoothTrajectory(egoVehicle, waypoints, egoSpeed)
```

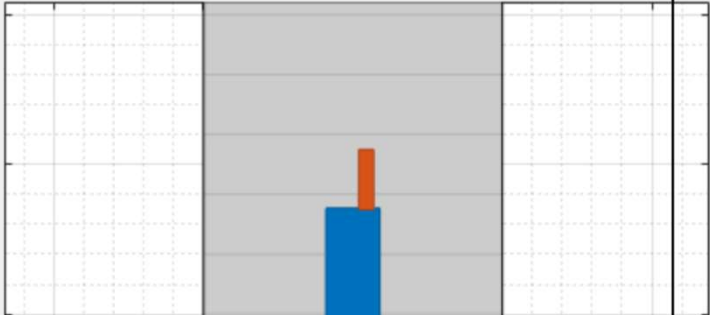
# Prebuilt Driving Scenarios

- The Driving Scenario Designer app provides a library of prebuilt scenarios representing common driving maneuver.
- The app also includes scenarios representing European New Car Assessment Programme (Euro NCAP) test protocols and cuboid versions of the prebuilt scenes.
- In the app, the Euro NCAP scenarios are stored as MAT-files. To open Euro NCAP files, on the app toolbar, select **Open>Prebuilt Scenario**.

# Autonomous Emergency Braking (AEB) Scenarios



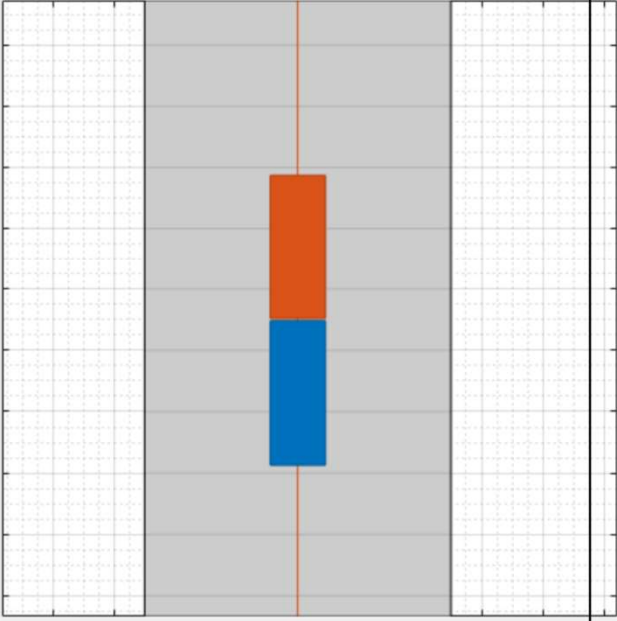
- AEB systems warn drivers of impending collisions and automatically apply brakes to prevent collisions or reduce the impact of collisions.
- Ego vehicle collides with the bicyclist at the front.

File Name	Description
AEB_Bicyclist_Longitudinal_25width.mat	<p>The ego vehicle collides with the bicyclist that is in front of it. Before the collision, the bicyclist and ego vehicle are traveling in the same direction along the longitudinal axis. At collision time, the bicycle is 25% of the way across the width of the ego vehicle.</p> 

# AEB Scenarios

- Car to Car rear braking scenario.



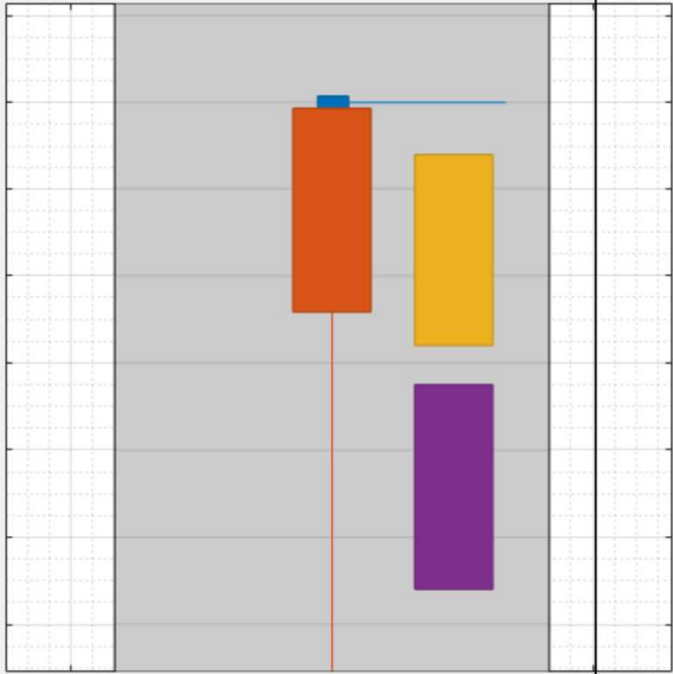
File Name	Description
AEB_CCRb_2_initialGap_12m.mat	<p>A car-to-car rear braking (CCRb) scenario, where the ego vehicle rear-ends a braking vehicle. The braking vehicle begins to decelerate at <math>2 \text{ m/s}^2</math>. The initial gap between the ego vehicle and the braking vehicle is 12 m.</p>  <p>Additional scenarios vary the amount of deceleration and the initial gap between the ego vehicle and braking vehicle.</p>



# AEB Scenarios

- AEB pedestrian Child Scenario.



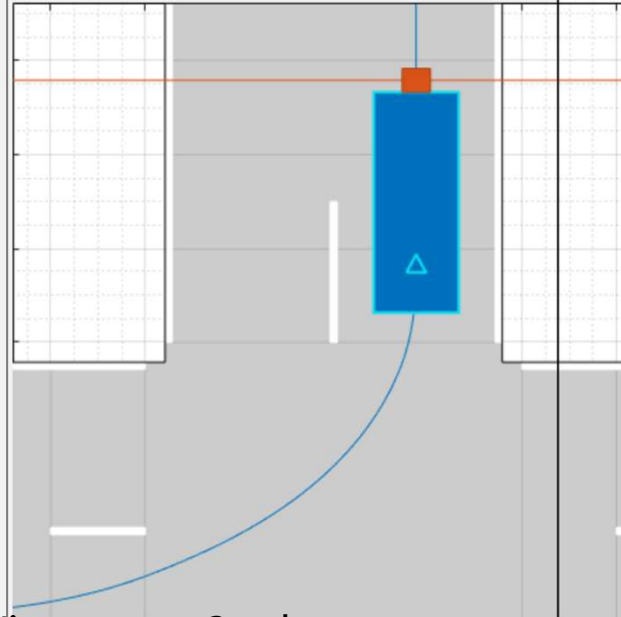
File Name	Description
AEB_PedestrianChild_Nearside_50width.m at	<p>The ego vehicle collides with a pedestrian who is traveling from the right side of the road, which Euro NCAP test protocols refer to as the near side. These protocols assume that vehicles travel on the right side of the road. Therefore, the right side of the road is the side nearest to the ego vehicle. At collision time, the pedestrian is 50% of the way across the width of the ego vehicle.</p>  <p>The diagram illustrates the AEB pedestrian child scenario. It shows a top-down view of a vehicle's width, represented by a gray rectangle. A red vertical line indicates the vehicle's centerline. A blue horizontal line at the top represents the pedestrian's path. An orange rectangle represents the pedestrian's body, and a yellow rectangle represents the pedestrian's head. A purple rectangle represents the pedestrian's legs. The pedestrian is shown crossing the vehicle's width from right to left, with the head and legs positioned on the right side of the vehicle's width at the time of collision.</p>

# AEB Scenarios

- AEB turning far side scenario.



File Name	Description
AEB_PedestrianTurning_Farside_50width.mat	The ego vehicle turns at an intersection and collides with a pedestrian who is traveling parallel with the left side, or far side, of the vehicle at the start of the simulation. At collision time, the pedestrian is 50% of the way across the width of the ego vehicle.

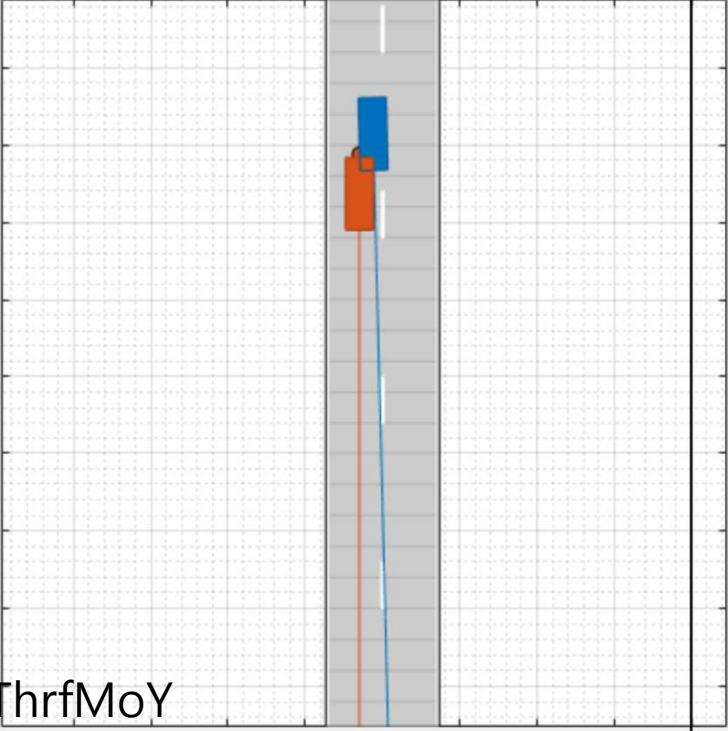


<https://m.post.naver.com/viewer/postView.naver?volumeNo=30595797&memberNo=29951632>

In an additional scenario, the pedestrian is on the other side of the intersection and travels parallel with the right side, or near side, of the vehicle at the start of the simulation.

# Emergency Lane Keeping (ELK) Scenarios

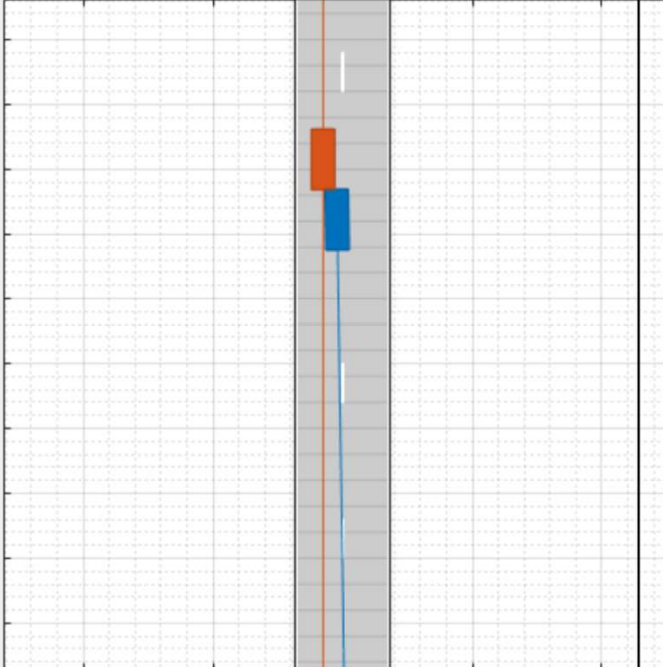
- ELK fast overtaking scenario.

File Name	Description
ELK_FasterOvertakingVeh_Intent_Vlat_0.5.mat	<p>The ego vehicle intentionally changes lanes and collides with a faster, overtaking vehicle that is in the other lane. The ego vehicle travels at a lateral velocity of 0.5 m/s.</p> 

<https://www.youtube.com/watch?v=OicEThrfMoY>

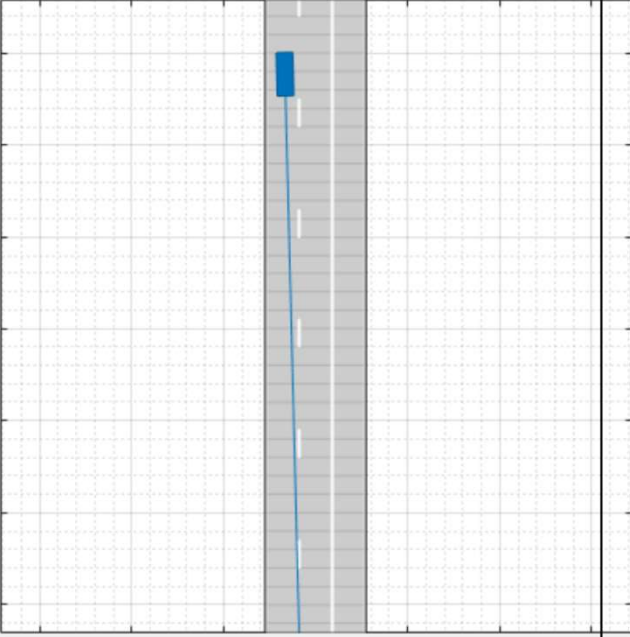
# ELK Scenarios

- ELK on coming vehicle scenario.

File Name	Description
ELK_OncomingVeh_Vlat_0.3.mat	<p>The ego vehicle unintentionally changes lanes and collides with an oncoming vehicle that is in the other lane. The ego vehicle travels at a lateral velocity of 0.3 m/s.</p>  <p>Additional scenarios vary the lateral velocity.</p>

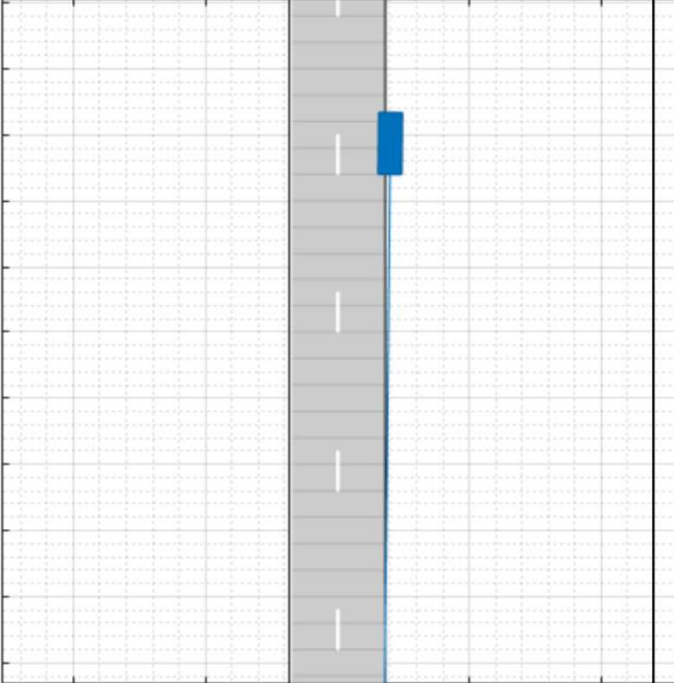
# Lane Keeping Assist (LKA) Scenarios

- LKA ego vehicle depart scenario.

File Names	Description
LKA_DashedLine_Solid_Left_Vlat_0.5.mat	<p data-bbox="1081 547 1688 707">The ego vehicle unintentionally departs from a lane that is dashed on the left and solid on the right. The car departs the lane from the left (dashed) side, traveling at a lateral velocity of 0.5 m/s.</p>  <p data-bbox="1081 1417 1688 1505">Additional scenarios vary the lateral velocity and whether the dashed lane that the vehicle crosses over is on the left or right.</p>

# LKA Scenarios

- LKA ego vehicle road edge scenario.

File Names	Description
<p data-bbox="450 544 1016 579">LKA_RoadEdge_NoBndry_Vlat_0.5.mat</p>	<p data-bbox="1106 544 1767 679">The ego vehicle unintentionally departs from a lane and ends up on the road edge. The road edge has no lane boundary markings. The car travels at a lateral velocity of 0.5 m/s.</p>  <p data-bbox="1106 1441 1693 1476">Additional scenarios vary the lateral velocity.</p>



# Intersection Scenarios

- EgoVehicleGOesStraight\_BicycleFromLeftStraight\_Collision 시나리오
  - 좌측에서 접근하는 자전거와 충돌하는 시나리오
  - 센서를 통해 물체 인식 가능

