

# 시뮬레이터를 활용한 K-City Map 기반 자율주행 알고리즘 개발

프로젝트 지향 자율주행차 전문인력 양성과정

# 목차

---

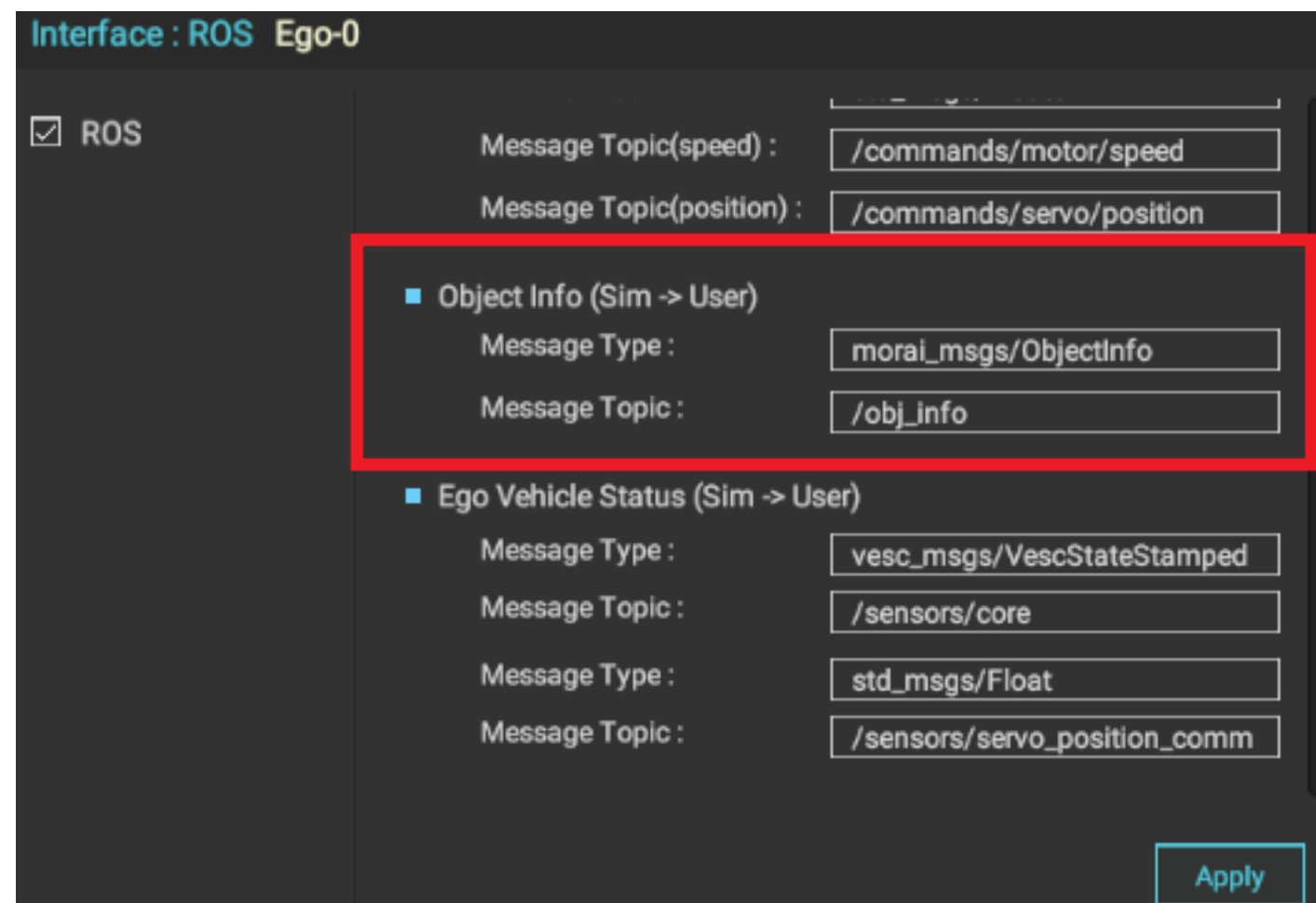
1. 장애물 데이터 받기
2. 정지선 위치 저장
3. 유효 장애물 판단

# **1. 장애물 데이터 받기**

# 장애물 데이터 받기

- 장애물 데이터

- 시뮬레이터에서 장애물의 절대 위치를 알려줌
- 인지 알고리즘 없이도, 판단, 제어 알고리즘에 장애물 데이터를 활용 가능
- Network에서 Object info를 통해 타입, 토픽을 확인 가능



# 장애물 데이터 받기

---

- 장애물 데이터
  - 시나리오 탭에서 Object를 눌러서 다양한 Object를 놓을 수 있음
  - 오브젝트를 더블클릭 눌러서 이동, 회전, 크기를 변경 가능

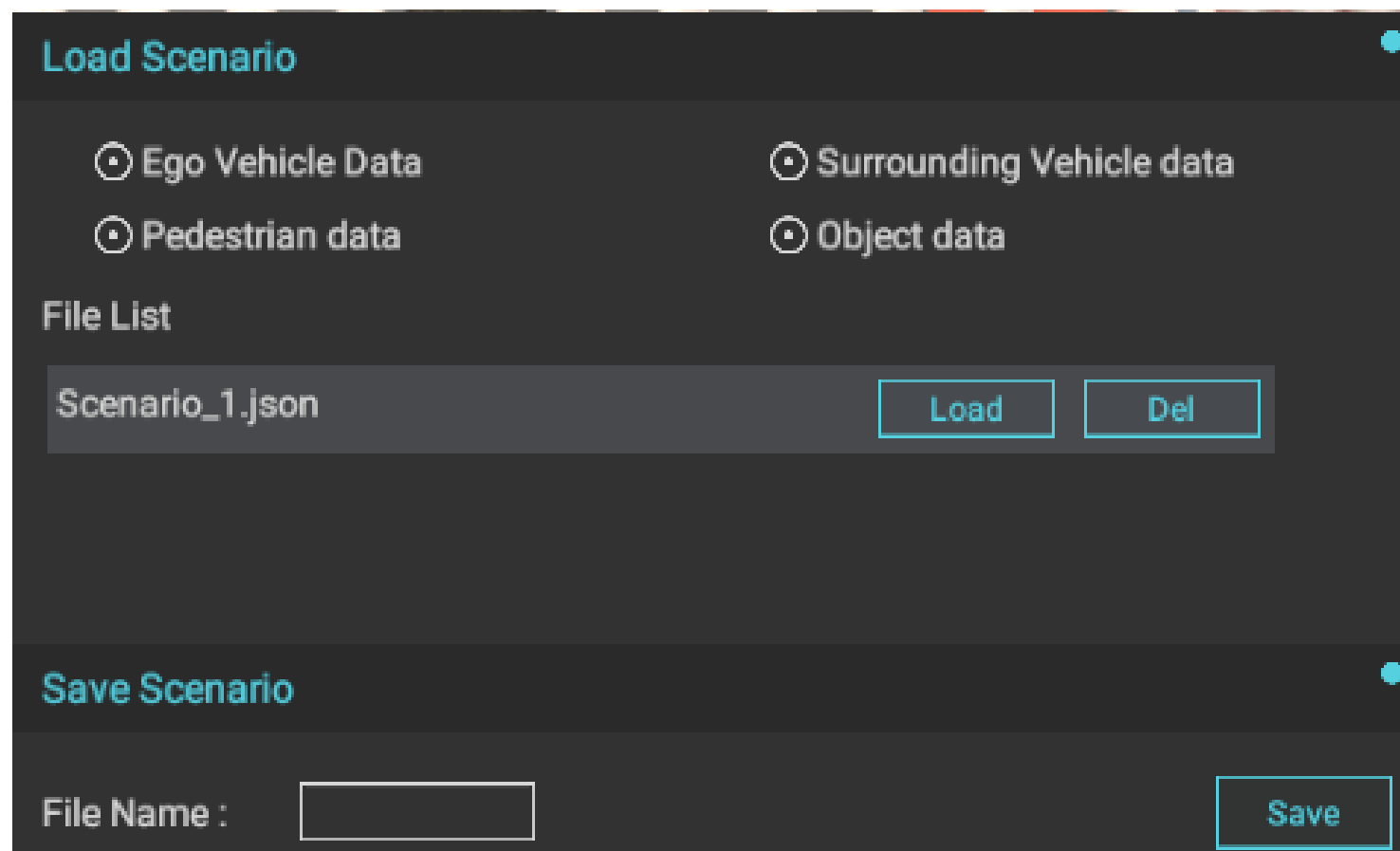


# 장애물 데이터 받기

---

- 장애물 데이터

- 시나리오 Save & Load 기능을 이용해 현재 시나리오를 저장 또는 Load 할 수 있다.
- 매번 장애물을 새로 놓을 필요가 없다.



# 장애물 데이터 받기

---

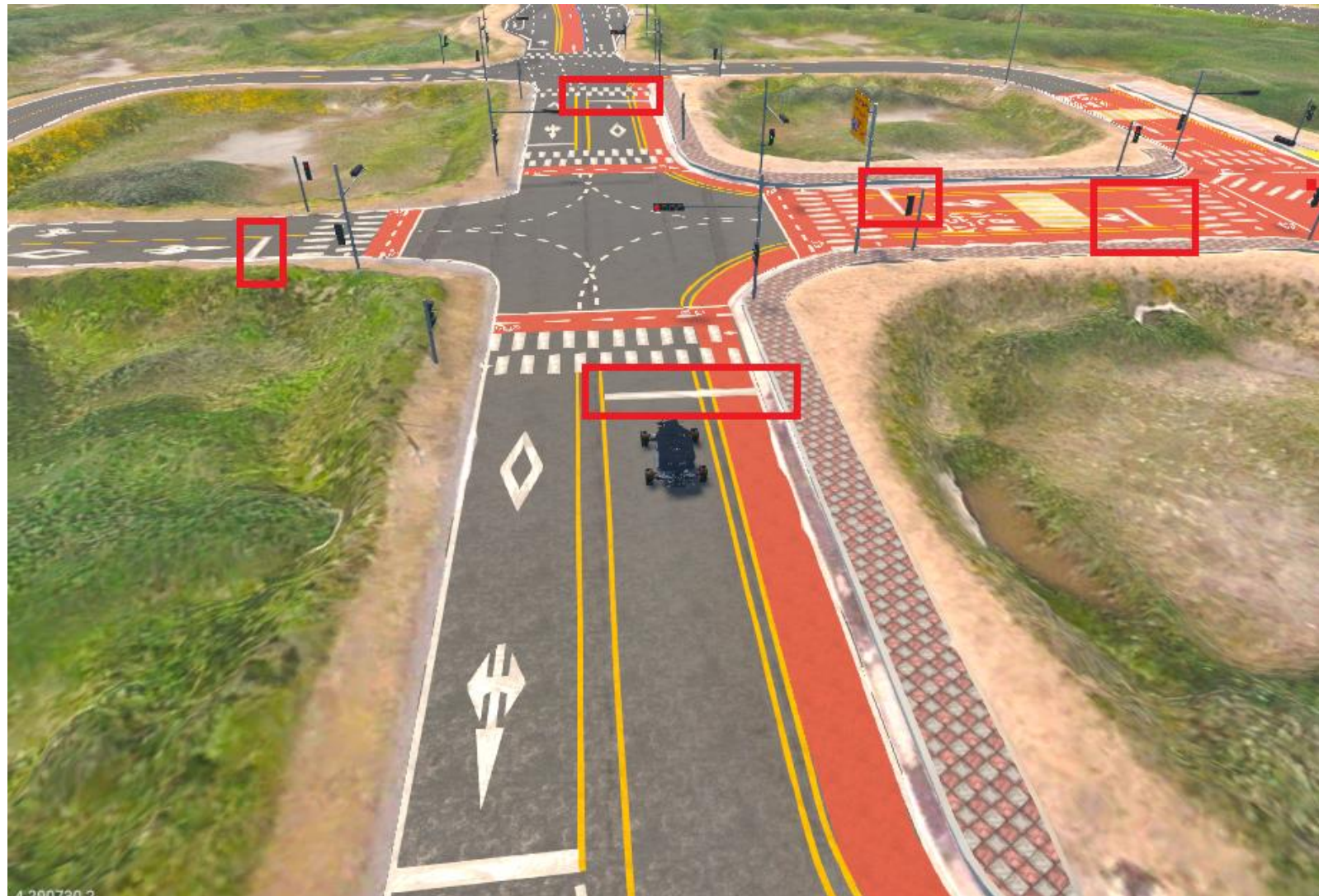
- 장애물 데이터
  - 시뮬레이터에 있는 모든 장애물이 Object\_info를 통해 들어온다.
  - 차량 좌표계 기준으로 앞쪽(x축)에 있는 장애물만 고려한다.
  - 현재 Local path 기준으로 경로 위에 있는 장애물만 고려한다.

## 2. 정지선 위치 저장



# 정지선 위치 저장

- 정지선 위치 저장
  - 정지선은 정지한 장애물과 동일하게 간주할 수 있음.
  - 인지 알고리즘으로 부터 신호를 인식하고, 초록불이면 정지선이 없다고 간주하면 된다.
  - 정지선의 절대적인 위치는 변하지 않음



## 정지선 위치 저장

- 정지선 위치 저장
  - 정지선은 정지한 장애물과 동일하게 간주할 수 있음.
  - 인지 알고리즘으로 부터 신호를 인식하고, 초록불이면 정지선이 없다고 간주하면 된다.
  - 정지선의 절대적인 위치는 변하지 않음

The screenshot shows the ROS Topic Monitor window. The left pane lists topics under the 'Topic' column. The right pane displays details for the selected '/odom' topic.

Topic	Type	Bandwidth	Hz	Value
/tf	tf2_msgs/TFMessage			not monitored
/sensors/servo_position_command	std_msgs/Float64			not monitored
/sensors/core	vesc_msgs/VescStateStamped			not monitored
/rosout_agg	roscpp_msgs/Log			not monitored
/rosout	roscpp_msgs/Log			not monitored
<input checked="" type="checkbox"/> /odom	nav_msgs/Odometry	6.99KB/s	9.74	
twist	geometry_msgs/TwistWithCovariance			
pose	geometry_msgs/PoseWithCovariance			
position	geometry_msgs/Point			
z	float64			0.0
y	float64			1179.9388417145237
x	float64			58.71405416197376
orientation	geometry_msgs/Quaternion			
covariance	float64[36]			(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0....)
header	std_msgs/Header			
child_frame_id	string			'base_link'
<input checked="" type="checkbox"/> /obj_info	morai_msgs/ObjectInfo	2.07KB/s	36.51	
velocity	float64[]			()
size_z	float64[]			()
size_y	float64[]			()
size_x	float64[]			()
pose_z	float64[]			()

### **3. 유효 장애물 판단**

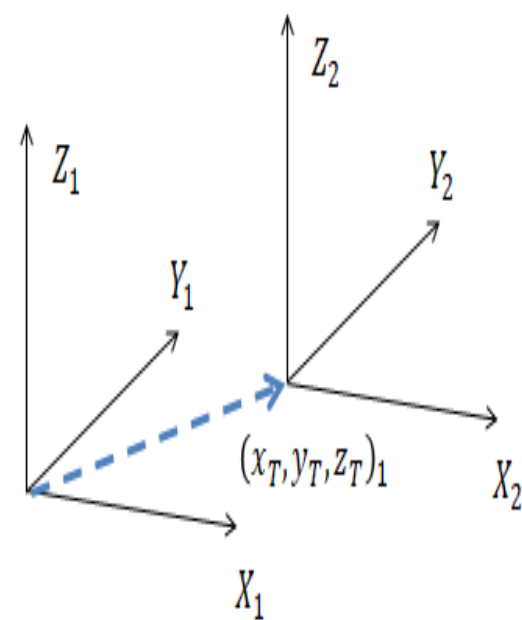


# 유효 장애물 판단

---

- 좌표 변환

- 장애물 좌표는 Global 좌표로 받아진다.
- 차량 기준 좌표계로 변환해준다.
- Translation & Rotation Transformation Matrix 이용

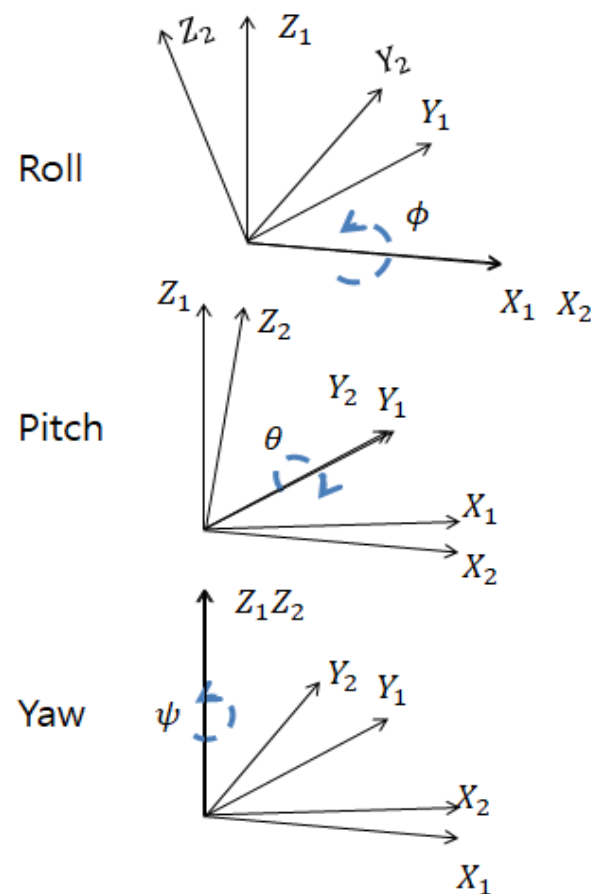


$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_T \\ 0 & 1 & 0 & y_T \\ 0 & 0 & 1 & z_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

# 유효 장애물 판단

- 좌표 변환

- 장애물 좌표는 Global 좌표로 받아진다.
- 차량 기준 좌표계로 변환해준다.
- Translation & Rotation Transformation Matrix 이용



$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

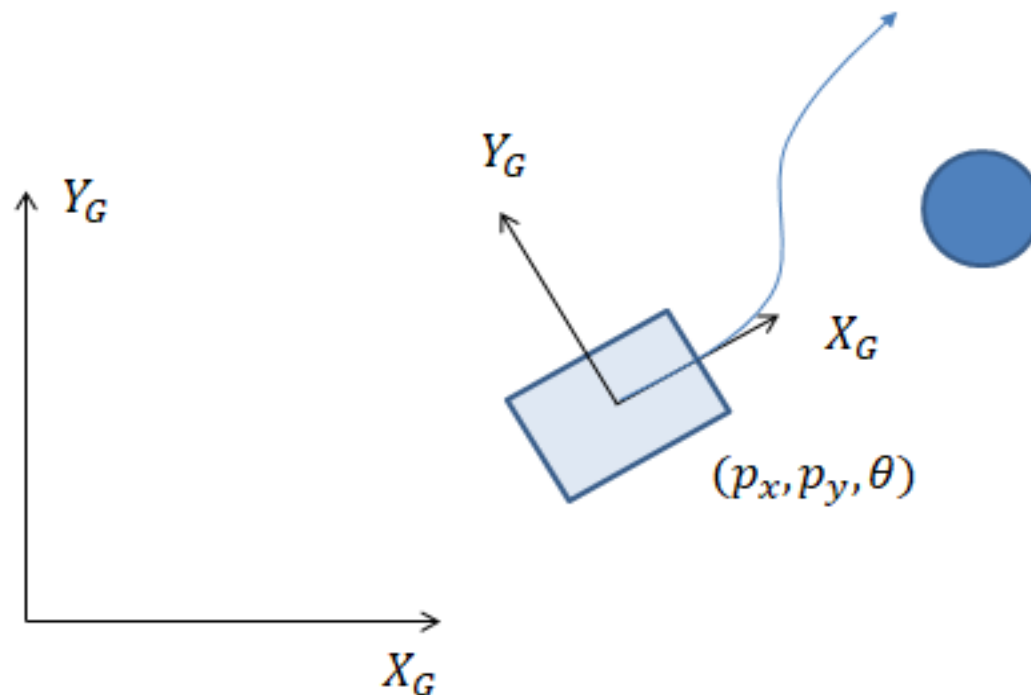
$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

# 유효 장애물 판단

---

- 좌표 변환

- 장애물 좌표는 Global 좌표로 받아진다.
- 차량 기준 좌표계로 변환해준다.
- Translation & Rotation Transformation Matrix 이용



$$\begin{bmatrix} X_1 \\ Y_1 \\ 1 \end{bmatrix}_G = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ 1 \end{bmatrix}_L$$

# 유효 장애물 판단

- 실습

- 장애물, 정지선을 받아서, 차량 기준 뒤쪽(x축)에 있을 경우 무시

```
54 class vaildObject :
55
56     def __init__(self):
57
58         self.vaild_stoplane_position=[
59             [58.26,1180.09],
60             [85.56,1228.28]
61         ]
62
63     def get_object(self,obj_msg):
64         self.all_object=ObjectInfo()
65         self.all_object=obj_msg
66
67
68     def calc_vaild_obj(self,ego_pose):
69         global_object_info=[]
70         loal_object_info=[]
71         if self.all_object.num_of_objects > 0:
72
73             tmp_theta=ego_pose[2]
74             tmp_translation=[ego_pose[0],ego_pose[1]]
75             tmp_t=np.array([[cos(tmp_theta), -sin(tmp_theta),tmp_translation[0]],
76                             [sin(tmp_theta),cos(tmp_theta),tmp_translation[1]],
77                             [0,0,1]])
78             tmp_det_t=np.array([[tmp_t[0][0],tmp_t[1][0],-(tmp_t[0][0]*tmp_translation[0]+tmp_t[1][0]*tmp_translation[1])],
79                                [tmp_t[0][1],tmp_t[1][1],-(tmp_t[0][1]*tmp_translation[0]+tmp_t[1][1]*tmp_translation[1])],
80                                [0,0,1]])
81
82             for num in range(self.all_object.num_of_objects):
83                 global_result=np.array([[self.all_object.pose_x[num]], [self.all_object.pose_y[num]], [1]])
84                 local_result=tmp_det_t.dot(global_result)
85                 if local_result[0][0]>0 :
86                     global_object_info.append([self.all_object.object_type[num],self.all_object.pose_x[num],self.all_object.pose_y[num],self.all_object.velocity[num]])
87                     loal_object_info.append([self.all_object.object_type[num],local_result[0][0],local_result[1][0],self.all_object.velocity[num]])
88
```

# 유효 장애물 판단

---

- 실습
  - 장애물, 정지선을 받아서, 차량 기준 뒤쪽(x축)에 있을 경우 무시

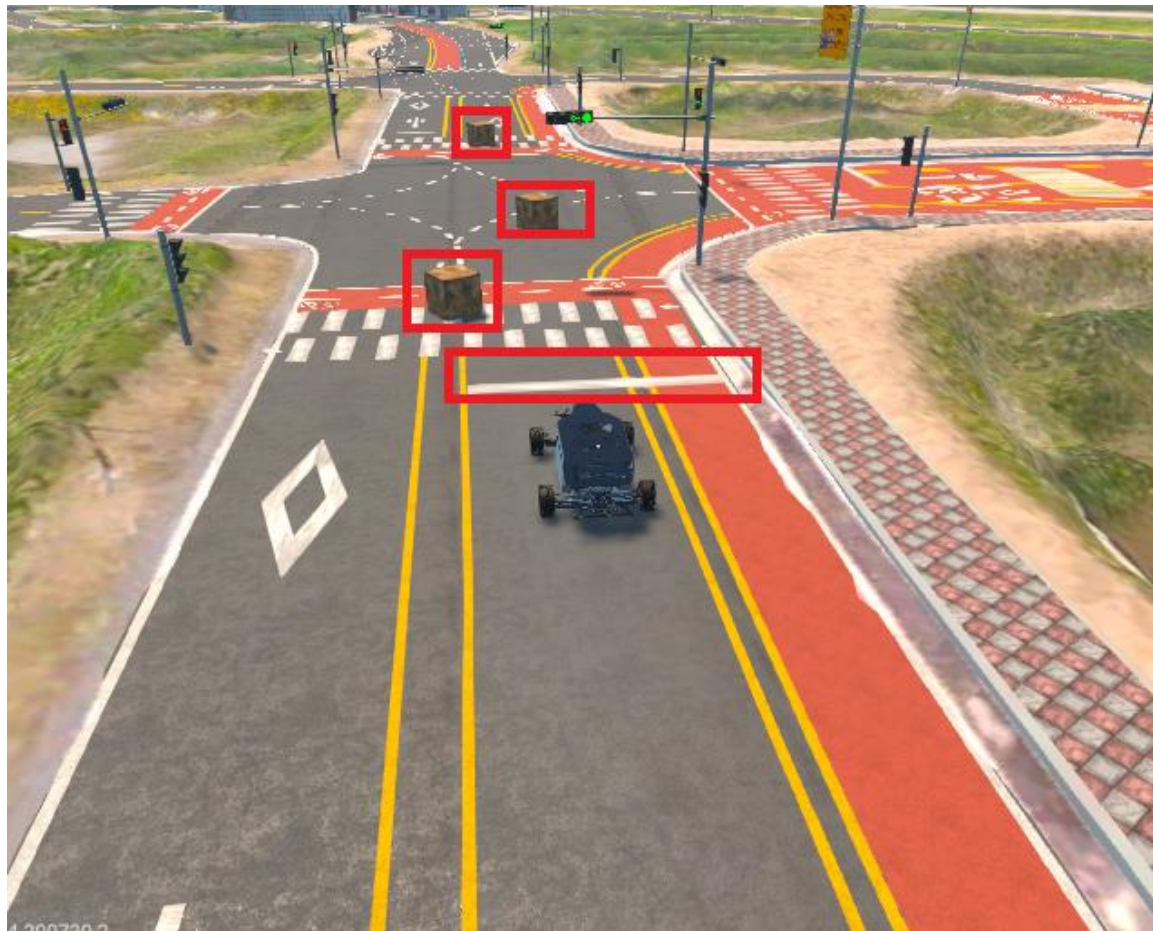
```
87         loal_object_info.append([self.all_object.object_type[num], local_result[0][0], local_result[1][0], self.all_object.velocity[num]])
88
89     for num in range(len(self.vaild_stoplane_position)):
90         global_result=np.array([[self.vaild_stoplane_position[num][0]], [self.vaild_stoplane_position[num][1]], [1]])
91         local_result=tmp_det_t.dot(global_result)
92         if local_result[0][0]>0 :
93             global_object_info.append([1, self.all_object.pose_x[num], self.all_object.pose_y[num], 0])
94             loal_object_info.append([1, local_result[0][0], local_result[1][0], 0])
95
96
97     return global_object_info, loal_object_info
```



# 유효 장애물 판단

- 결과

- 타입 1 : 정지선, 타입 2 : 장애물
- 차량 좌표계 기준으로 x,y 값이 구해짐



```
[2, 11.299852883203812, 2.636856322708809, 0.0]  
[2, 20.721908043944268, -0.6156689227386778, 0.0]  
[2, 40.136054300895694, 1.0218066836041544, 0.0]  
[1, 4.665112199087844, 0.40365922538433097, 0]  
[1, 60.04617095861545, -0.3065736052773218, 0]
```

**END**