Supplementary Material

Leichen Wang[1,2], Bastian Goldluecke[2], and Carsten Anklam[1]

[1] Research and Development of Radar Sensor, Daimler AG, Sindelfingen, Germany
`leichen.wang, carsten.anklam@daimler.com`
[2] Department of Computer and Information Science, University Konstanz, Germany
`bastian.goldluecke@uni-konstanz.de`

1 Occupancy Grid Mask

Inspired by [39] and [40], we utilize occupancy grid mask in our framework. Since the code of [39] and [40] has not been open sourced, we use Bresenham's line algorithm [41] to generate the corresponding occupancy grid mask from LiDAR point clouds.

Bresenham's line algorithm is one of the most widely extensive algorithms for occupancy grids map by two-dimensional ray casting. Given coordinate of ego position $\mathbf{E}(x_o, y_o)$ and a set of points $\mathbf{P}_i(x_i, y_i)$. The all pixels are initialized to unknown space. The algorithm terminates once the nearest point $\mathbf{P}_i$ is found, all of the pixels between $\mathbf{P}_i$ and $\mathbf{E}$ are considered as free space if the point density is less than threshold. The rest pixels are occupied space. Details of building occupancy grid mask can be found in Alg.1.

Algorithm 1: Bresenham's line algorithm for occupancy grid mask

```
Input: ego position $\mathbf{E}$, points $\mathbf{P}$
Output: occupancy grid mask $\mathbf{M}$
$\mathbf{M}[:] \leftarrow$ UNKNOWN ;
for $\boldsymbol{p}$ in $\boldsymbol{P}$ do
    $\mathbf{C} = BresenhamLine(\mathbf{p}, \mathbf{E})$;
    for $c_j$ in $\boldsymbol{C}$ do
        if $c_j \subset \boldsymbol{M}$ then
            if point density in $c_j$ is larger than threshold then
                $\mathbf{M}[c_j] \leftarrow$ OCCUPIED;
            else
                $\mathbf{M}[c_j] \leftarrow$ FREE;
            end
        end
    end
end
```

Algorithm 2: Bresenham's line algorithm for emergency scenario generator

```
Input: ego position $\mathbf{E}$, original points $\mathbf{OP}$, augmented points $\mathbf{AP}$
Output: merged point $\mathbf{MP}$
$\mathbf{MP} \leftarrow \mathbf{OP}+\mathbf{AP}$ ;
for $ap$ in $AP$ do
    $\mathbf{C} = BresenhamLine(\mathbf{ap}, \mathbf{E})$;
    for $op$ in $OP$ do
        if $op \subset C$ then
            DELETE ap in MP;
        end
    end
end
for $op$ in $OP$ do
    $\mathbf{C} = BresenhanLine(\mathbf{op}, \mathbf{E})$;
    for $ap$ in $AP$ do
        if $ap \subset C$ then
            DELETE op in MP;
        end
    end
end
```

Types of objects	Pedestrian	Cyclist	Traffic cone	Tire	Others
Number	55	30	30	10	75
Types of scenarios	Intersection	Country road	Industrial park	Building site	Others
Number	5	5	2	2	6

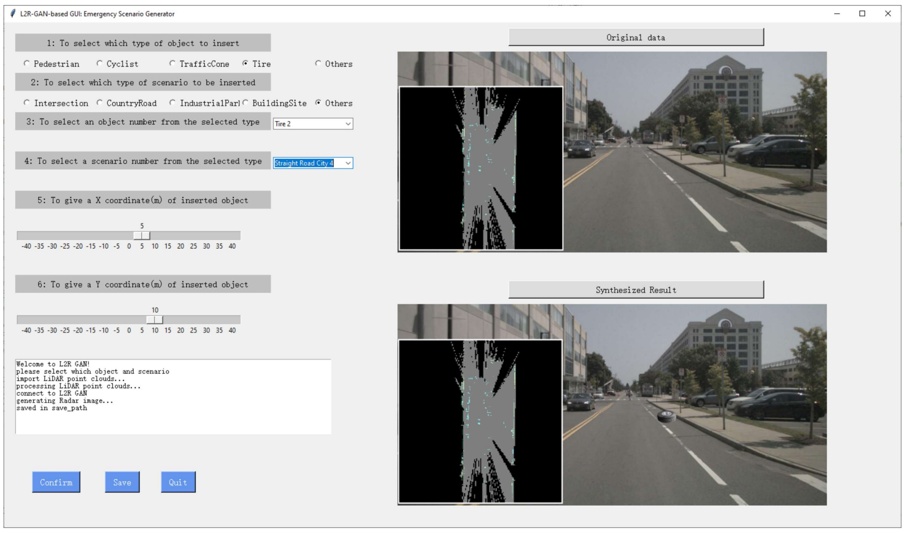
Fig. 1. Number of model and traffic scenarios for different categories. Some categries are included in the "others" type, such as cyclist and bady carriages.

2 Emergency Scenario Generator

To generate emergency scenario for the ADAS application, we collect 200 semantically labeled objects from the nuScenes dataset to create an object database for the user to choose from. We also select 20 different traffic scenarios from nuScenes dataset. The number of each object and traffic scenario has been given in Fig. 1.

In our L2R-GAN-based GUI, the user firstly select an object from the object database, then choose a traffic scenarios from scenario database. After that the user is required to define the position of the inserted object. Finally all occluded points are calculated and removed by raycasting methods. Details of removing occluded points can be found in Alg.2.

The python-based GUI interface is shown in Fig.2.


L2R-GAN-based GUI: Emergency Scenario Generator

1: To select which type of object to insert

Pedestrian [✓ Tire] Cyclist TrafficCone Others

2: To select which type of scenario to be inserted

Intersection CountryRoad IndustrialParl BuildingSite [✓ Others]

3: To select an object number from the selected type: Tire 2

4: To select a scenario number from the selected type: Straight Road City 4

5: To give a X coordinate(m) of inserted object: 5

6: To give a Y coordinate(m) of inserted object: 10

```
Welcome to L2R GAN!
please select which object and scenario
import LiDAR point clouds...
processing LiDAR point clouds...
connect to L2R GAN
generating Radar image...
saved in save_path
```

Fig. 2. L2R-GAN-based GUI can be applied to generate different emergency scenarios. The user can select different objects and scenarios from our database.