Synthetic Driving Scenario Generator

M. Cossu[1], R. Berta[1], L. Forneris[1], F. Bellotti[1]

[1]Department of Naval, Electrical, Electronic, Telecommunications Engineering (DITEN), University of Genoa, Genoa, Italy
marianna.cossu@edu.unige.it

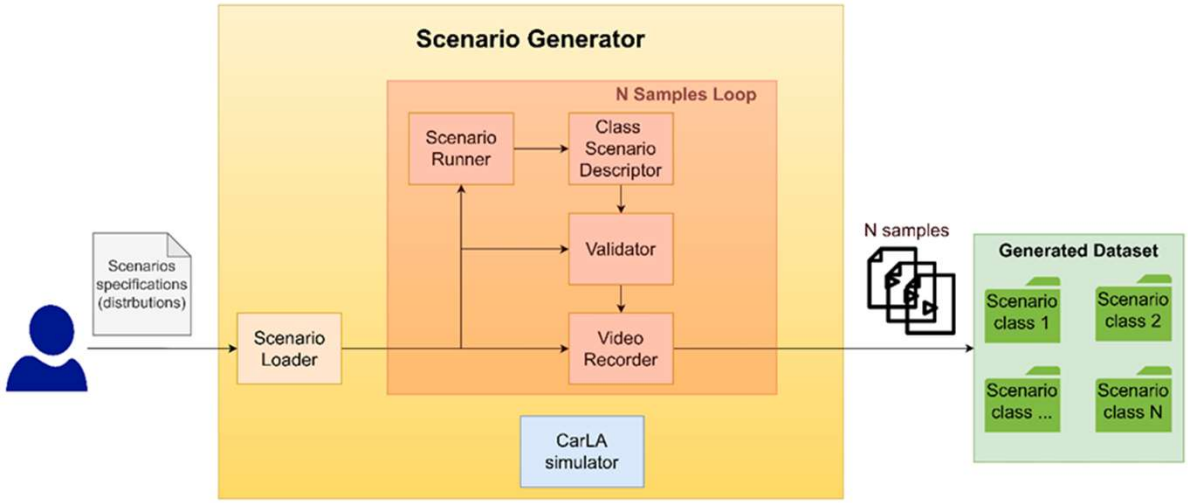

Fig. 1 Overall system architecture

Introduction

Traffic scenario detection is important to guarantee functioning of automated driving functions in different Operational Design Domain (ODD). Implementing scenario detection through machine learning requires datasets. Since there is a lack of public datasets available in this field, we thus decided to develop a system for synthetic scenario generation (Fig. 1), featuring:

• Ability to generate a considerable number of samples as varied as possible (based on real-world statistics)

• Ability to include environmental variability

• Usability (e.g., effectiveness, efficiency (parametric generation system) and ease of use)

For the simulation phase we use Car Learning to Act (CarLA), an open-source simulator for automated driving research.

Parameter distributions

One of the most crucial aspects in generating realistic synthetic driving scenarios is the creation of samples consisting of real-world-like actions. In order to do that it is required a significant amount of high-quality data.

Our tool used multidimensional real-world parameters distributions obtained from the processing of the ADScene database that contains more than 1 M km of drive recordings.

Thanks to scenario-specific multidimensional distributions, we understand, extract, and parameterize all the key characteristics of each part of the scenarios (e.g., initial ego speed, initial opponent speed, action duration, etc.).

In addition to the parametric variability, an environmental variability was introduced as well (14 different weather and light conditions, initial location, different traffic intensities, different models of vehicles each characterized by different set of colors, etc.)

Sunny | Heavy rain | Sunset + paddles

Fig. 2 Examples of different weather and light conditions

Scenario Loader and Runner

Scenario Loader and **Runner** have the aim to:

- define which scenario the user wants to simulate running the desired class scenario descriptor and the simulator
- load the multidimensional distributions of the scenario that the user wants to reproduce

Class Scenario Descriptor

Class Scenario Descriptor defines and implements the desired scenario's samples as follows:

1. extracts a combination of parameters from the real-world multidimensional distributions and defines the common parameters.
2. defines the step-by-step behavior of each actor in the scenario:
    1. Protagonists and, if exists, adversary
    2. non-protagonists, such as traffic component
3. initializes and implements the validator and the features that allow the tool to define when a scenario is a failure or not (E.g., detect crashes, etc.)
4. informs the scenario runner to starts the simulation and the recording

We implemented 10 different classes: (1, 2) cut-in executed in front and behind the ego vehicle, (3, 4) cut-out executed in front and behind the ego vehicle, (5) ego vehicle lane change, (6) leading vehicle brake, (7) free ride recorded in front of the ego vehicle, (8) free ride recorded behind the ego vehicle, (9) following ego vehicle and (10) following leading vehicle. In Fig. 3 are shown some examples of possible scenarios.

Brake | Cut in behind right/left | Cut in front right/left

Fig. 3 Examples of three different scenarios

Validator and Recorder

Validator performs the following steps:

1. Executes a real-time analysis of the simulated samples
2. Checks for erroneous situations such as crash, car leaving their roadway, etc.
3. If an abnormal behavior is detected, it informs the system to stop the simulation and delete the sample

The validator helps to save generation time and avoids the need for:

- a manual human verification
- a validation of the correct creation of the dataset

Recorder starts the recording of each sample using different type of sensors (camera, lidar radar, etc.)

Results

As an initial experiment, we generated a dataset with +800 samples for the ego lane change, cut in front, cut in behind, cut out front and cut out behind scenarios, obtaining:

• a generation rate of 160 samples/h.

• an error sample rate 10 %

Different networks have been trained and tested with the generated dataset reaching high values of test accuracy > 90%

elios
Università degli Studi di Genova

Università di Genova | DITEN DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA, ELETTRONICA E DELLE TELECOMUNICAZIONI