**Using Python to create custom Car Following scenarios on OpenDS Driving Simulator with controlled average speed and speed variation**

**Junhao Wang**

Student

University of Michigan 2901 Baxter Road

Ann Arbor, MI 48109-2150 USA

[junhao@umich.edu](mailto:PAGreen@umich.edu)

# Abstract

This paper describes the design and usage of a Python program that creates custom Car Following scenarios on OpenDS Driving Simulator with controlled average speed and speed variation. This work is built upon existing car following scenario – HighTraffic.1, with multiple cars on a straight one-lane road that has a starting and end point. In brief, a Python program is written to create controlled pseudo-random speed profile for both lead vehicle and side vehicle, by adding up adding up multiple sin waves with different amplitude, frequency and phase shift. In addition, redundant part of the .XML files of HighTraffic.1 was commented out. Finally, the speed profiles of both lead vehicle and side vehicle are plotted to PNG files for inspection. This work can be used by IOE437 students to create Car Following scenarios with interesting combinations of speed profiles of lead and side vehicles. Because the speed of both vehicles are pseudo-random, the driving simulation experience could approximate real-driving experience in Car Following situations.

*Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.*

*Copyright is held by the owner/author(s).*

# INTRODUCTION

IOE 437, taught by Dr. Paul Green at Department of Industrial and Operations Engineering, is an introductory class to various aspects of automobile ergonomics. One of the final homework is to use OpenDS, an open-source driving simulation software, to conduct experiment on driver distraction. Most students, however, encountered the following three problems when using OpenDS: (1) settings.xml file needs different configuration settings for different hardware in order for computer to control simulation through Logitech G27 driver interface (steering wheel and foot controls). (2) scenario.xml file needs highly technical knowledge to configure in order to create custom car following scenarios for experiment. (3) data analysis is difficult with vague descriptors of data columns, and lead vehicle distance is missing from the data analysis output. This work aims at solving the second problem by automating the configuration of scenario.xml file in order to create realistic custom car following scenarios.

# AN OVERVIEW OF OPENDS SCENARIO FILE STRUCTURE

# A general introduction to OpenDS is omitted in this paper, because the introductory materials are either taught in class, or online (<https://www.opends.eu/)>. Rather this section will go through the complex file structure that constructs an OpenDS driving scenario. Scenario folders are located at assets/DrivingTasks/Projects of the downloaded software. Within each scenario foldee are the following XML files: (1) a named XML file that links all other parts together, for example, highTraffic.xml. (2) settings.xml that configures the controller such as keyboard, joystick and driver name, etc. (3) scene.xml that contains all objects in the simulation, including road, car, road signs, etc. (4) interaction.xml that creates custom activities that can be triggered by conditions (5) scenario.xml that creates custom scenarios by specifying physical attributes of vehicles at different times (speed, acceleration, etc.). These XML files are read by the java program at run time and used to create simulations.

**2. PREPARE EXISTING SCENARIO FOLDER**

Because the XML files are hard to be created from scratch due to their complexity, this project takes the approach to modify existing files from HighTraffic.1, a scenario that has multiple vehicles on a straight one-lane road. The following steps are taken to clean up the folder in order to have a clean lead vehicle / side vehicle scenario: (1) task.xml is commented out and related task tag is commented out from the HighTraffic.xml file. (2) in scenario.xml, redundant cars are commented out, leaving only a lead vehicle and side vehicle. (3) unrelated XML files are deleted , leaving only highTraffic.xml, interaction.xml, scenario.xml, scene.xml, settings.xml task.xml.

Additional reasons to use HighTraffic.1 scenario is that: (1) it uses a straight road, which makes it easier to calculate steering wheel movement variance, lateral movement standard deviation, etc. (2) it already has models of lead and side vehicles, therefore saves the time to build new car models (3) it has analytics recording for a controlled length of drive, having a starting and end point.

# 3. CREATE PSEUDO-RANDOM SPEED PROFILE

In order to create a realistic yet controlled driving experience for test subject, the lead and side vehicles should be driving within a range of speed that varies randomly, as in real-life driving. The technique of combining multiple sin waves with different amplitude, frequency and phase shift was used to create a pseudo-random speed profile with an average speed and amplitude of deviation from the average speed. The complexity of the speed profile can be changed through adding more and less sin waves together, but for the purpose of simple usage, default number and parameters of sin waves are used: 10 sin waves with 0.1 radian angular frequency as the basis. Each one of the sin wave applies a multiplication of a random floating number between 0 and 1 on the base angular frequency, and the phase shift is the new angular frequency times the product of a random floating number between 0 and 1 and the number of sin waves, which is 10. The amplitude is first updated by dividing a quarter of the number of sin waves and then updated by multiplying a random floating number between 0 and 1. The first step is determined by inspection of the plot of speed profiles, and diving a quarter of the number of sin waves gives a good approximation of the randomness in real-life driving. After adding all the sin waves up, the value is divided by the number of sin waves. This approach, while containing certain degrees of qualitative inspection, gives a realistic yet controlled range of speed. This approach also ensures that the mean speed is not affected by the complexity of the speed profile.

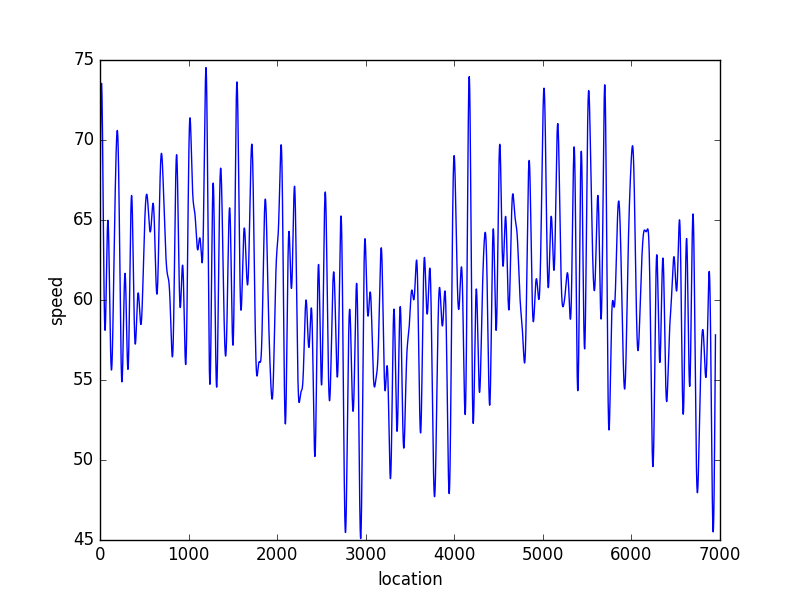
Figure 1 and Figure 2 are sample speed profiles of lead vehicle. 

Figure 1. Sample Speed Profile A with average speed 60km/h and amplitude of change 10km/h

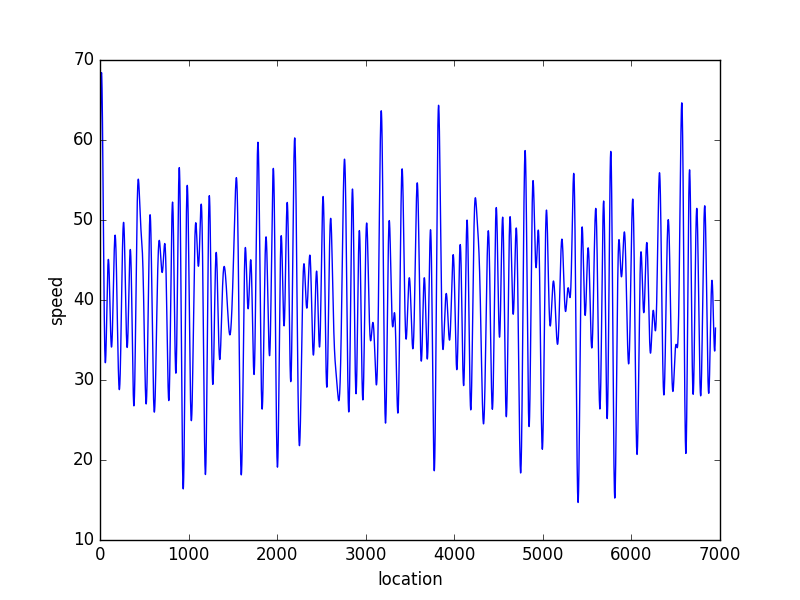


Figure 2. Sample Speed Profile B with average speed 40km/h and amplitude of change 20km/h

As observed from these Figures, the speed profile, while seemingly random with spikes, follows the average speed and amplitude of speed changes.

# 4. INSERT SPEED CONTROL POINTS INTO SCENARIO.XML FILE

After creating the speed profile, a famous Python package - BeautifulSoup is used to modify existing XML file. This package can search through the XML file and insert/modify/delete content and tags. The following steps were taken to modify scenario.xml: (1) delete all current speed control points, tagged as wayPoint. Each wayPoint contains 3-D coordinate of the point and related vehicle speed. Each vehicle has an array of wayPoints, defining its speed and route on the map. The vehicle is therefore automatically following the wayPoints at a speed no greater than the specified speed. (2) insert custom created wayPoint using pseudo-random speed profile or 0 km/h that means not moving. The resulting scenario.xml file has updated speed for both side and lead vehicles.

# 5. CONCLUSIONS AND FUTURE WORK

# This work aims at solving one of the three problems students of IOE 437 encountered during the driving simulator homework project, namely: automatically configure scenario.xml file in order to create realistic custom car following scenarios. Three things are learned in the process of coding, experimenting with different parameters, researching best methods to manipulate XML files, debugging and documentation: (1) have a clear definition of the project goal before anything. (2) play with the problem long enough to have a fun and intuitive solution. (3) do not arbitrarily expand the scope of the project because that would easily lead to burnout.

# The future direction of this work can take the following three routes: (1) develop more custom control for car following scenarios, such as frequently occurring event while driving, lane changing, etc. (2) port the Python code into openDS-based GUI, for the convenience of students who do not have a basic understanding of computers. (3) more analytics such as steering entropy built into the simulation, visualization of minimal time to collision, etc.

# 6. SOFTWARE DOWNLOAD LINK

All software with instruction on how to use it can be acquired from <https://github.com/JunhaoWang/simulation_project>