

Co-Simulation of Human and Machine Generated Drivers on a Single Test Track

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science at George Mason University

By

Raghavendra Somepalli
Bachelor of Technology
Anurag Group of Institutions, 2017

Director: Dr. Duminda Wijesekera, Professor
Department of Computer Science

Spring Semester 2019
George Mason University
Fairfax, VA

Copyright © 2019 by Raghavendra Somepalli
All Rights Reserved

Dedication

This is dedicated to my family.

Acknowledgments

I would like to thank my family and friends who made this happen. My advisor, Duminda Wijesekera, guided me at every step. My colleagues, Aarti Modani and Chaitanya Yavvari, helped me whenever I needed any assistance. My friend, Jai Vora, was of invaluable support. Finally, thanks to Dr. Zuric and Dr. Kan for being immensely encouraging.

Table of Contents

	Page
List of Tables	vi
List of Figures	vii
Abstract	viii
1 Introduction	1
1.1 Microscopic Simulators	1
2 Related Work	2
3 SUMO	3
3.1 Creating a Scenario	3
3.1.1 OSM file	4
3.1.2 Generating Traffic Network from OSM	4
3.1.3 Generating Machine Controlled Traffic in SUMO	5
3.1.4 Generating SUMOCFG	5
3.1.5 Simulation in SUMO	6
4 Implementation	7
4.1 Basic Environment	8
4.2 Creating Lanes	8
4.3 Creating Vehicles	9
4.4 Creating the Human Driven Vehicle	9
5 Future Work	11
6 Conclusion	12
REFERENCES	13

List of Tables

Table	Page
3.1 SUMO reporting options	3

List of Figures

Figure	Page
3.1 Exporting from OpenStreetMap	4
3.2 Network in SUMO	6
4.1 TraCI Protocol	7
4.2 The Simulation in Unity	10
4.3 Autonomous Traffic	10

Abstract

CO-SIMULATION OF HUMAN AND MACHINE GENERATED DRIVERS ON A SINGLE TEST TRACK

Raghavendra Somepalli, M.S.

George Mason University, 2019

Thesis Director: Dr. Duminda Wijesekera

With autonomous vehicles in the early phases of road testing, we can surely expect them on the road in a few years. But the existence of a mixed traffic with human-driven vehicles will be inevitable for a long time after the introduction of autonomous vehicles and the ramifications of such a traffic are hard to foresee. But by creating a simulation which involves both kinds of vehicles, we can study their behavior and analyze the effect the autonomous vehicles has on the human-driven vehicular traffic. Using SUMO, which is a microscopic traffic simulator, the traffic of autonomous vehicles can be replicated by using machine generated vehicles whose speed and position in each simulation step is controlled by SUMO. This thesis tries to interface SUMO and Unity, a powerful cross-platform game engine, by using the Traffic Control Interface (TraCI) protocol and in doing so, a car was generated in Unity which can be human controlled and driven in the same track as the machine generated vehicles. This paper reports the details of this co-simulation using a real life test track which facilitates a comprehensive analysis of interactions between the vehicles in the real world.

Chapter 1: Introduction

With the rapid increase in the volume of traffic and the regular changes in the road network, traffic simulation is crucial since the alternative includes real-life setup & experimentation of relevant solutions. This can involve many unexpected risks & can incur unnecessary expenditure. Consequently, simulation is the best solution to study traffic before deployment which allows us to tweak many variables in the environment all the while providing reliable data.

Based on the number of parameters involved in the simulation, we have three different models: Macroscopic models, Mesoscopic models and Microscopic models. Macroscopic models have the least number of parameters of the three models as they inspect the relation between aspects like speed, volume, density, etc. Mesoscopic models illustrate specific vehicles but do not describe multiple other exact details like in a microscopic simulation. In this thesis, for the purpose of traffic simulation, a microscopic simulator called SUMO was used and the following subsection talks about microscopic simulators.

1.1 Microscopic Simulators

Microscopic simulation can be used to analyze specific elements in a transportation system such as changes in the individual vehicles and behavior of individual passengers. These changes can be tracked on a step-by-step basis in the simulation.

The modeling to focus on an individualistic elements means we require a large amount of computation power. This is because each of the elements' behavior is to be tracked in each step every second or sub-second. But this might have been a complication in the earlier days when simulation was just being developed. With the computing power we have in the current time, it is easy to achieve such a microscopic simulation even for large areas.

Chapter 2: Related Work

With car manufacturers, among others, in the race to develop autonomous vehicles have led to the creation of several tools aimed at microscopic simulation. The following part of this section mentions some related tools.

monoDrive[1] consists of a Ultra High Fidelity (UHF) simulator which can help analyze hard to predict edge cases. This can also simulate virtual vehicles for long distances in virtually created maps. This simulator does not provide a human controlled vehicle and also uses virtual environments.

LG Silicon Valley Lab created the LGSVL Simulator[2] which is based on Unity and can be integrated with Apollo[3], Autoware[4], DuckieTown[5]. It includes an ego vehicle which can autonomously drive in a simulated real life environment along with other machine controlled vehicles.

AirSim[6] was developed for AI research with an objective to gather data for deep learning, computer vision and reinforced learning algorithms for autonomous vehicles. It is built on both Unreal Engine and Unity, with the Unity release still in the experimental phase. It allows the extraction of training data while driving around a drone or a car in a virtual environment.

Along with the aforementioned tools, other Advanced Driver Assisted Systems (ADAS) exist such as SimDriver[7], ANSYS[8], Opal-RT's RT-Lab[9], cognata[10]. Some of these include human controlled vehicles but they include ADAS, while this research aims to co-simulate human driven vehicles and autonomous vehicles.

Chapter 3: SUMO

This thesis used SUMO, short for Simulation of Urban MObility, which is an open-source microscopic simulator that was introduced in 2001 and being developed by the employees of the Institute of Transportation Systems in the German Aerospace Center. The presence of Traffic Control Interface (TraCI) protocol in SUMO was the imperative reason this particular simulator was chosen. This enables us to make use of the TCP server/client architecture to exchange data with SUMO through external applications. SUMO provides the following basic command line options:

Table 3.1: SUMO reporting options

Option	Description
-v	Verbose Output
--print-options	Print options before processing
-help	Prints the help screen
-V	Prints the current version
-X	et schema validation scheme of XML inputs
-W	Disables output of warnings
-l	Writes all messages to FILE
--message-log	Writes all non-error messages to FILE
--error-log	Writes all warnings and errors to FILE

3.1 Creating a Scenario

To obtain a realistic environment, the geographical data was obtained from OpenStreetMap (OSM)[11] which is an open-source collaborative project that contains free geographical data of the world. The following steps explain the creation of a scenario for the data extracted from OSM.

3.1.1 OSM file

An area of the map is selected and exported from the OpenStreetMap website. The area selected is shown in Figure 3.1 and the extracted data is in a `.osm` format. The OSM data was saved under the filename `map.osm`.

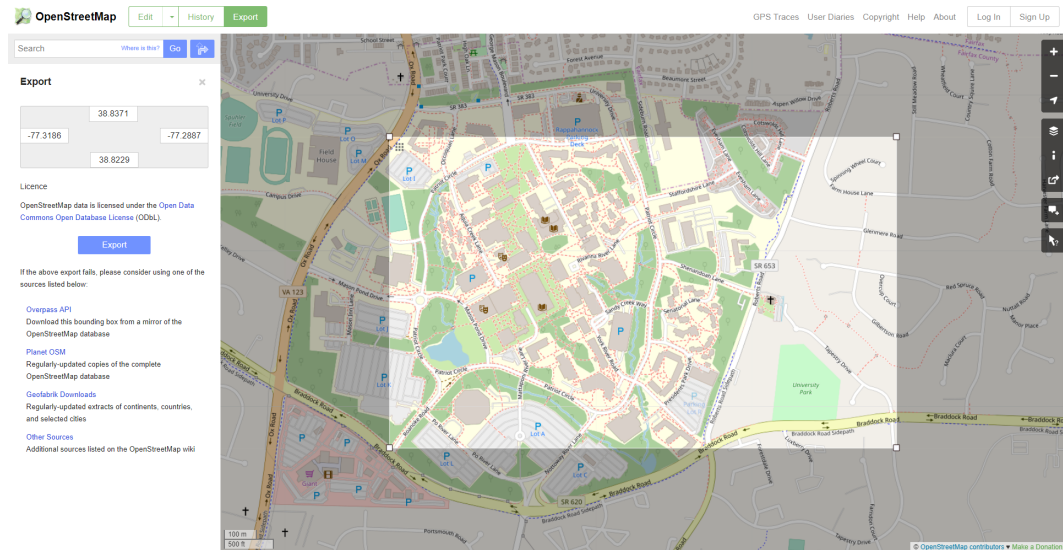


Figure 3.1: Exporting from OpenStreetMap

3.1.2 Generating Traffic Network from OSM

A SUMO network file has the `.net.xml` format and can be generated by using the NET-CONVERT tool that is part of the SUMO package. This file consists of the network which has (0,0) as origin of the system. The network can be specified by using XML to create the whole structure which is a long and unnecessary process. But since we are using OSM data which also involves creating a significantly large scenario for our simulation, it is tiresome to create the whole network from scratch. The following command was used to convert the downloaded OSM data to a `.net.xml` format.

```
netconvert --osm-files map.osm -o map.net.xml
```

3.1.3 Generating Machine Controlled Traffic in SUMO

We can use the python script `randomtrips.py` provided in the SUMO package to generate the traffic of machine controlled vehicles. It does this by choosing a random source edge and a destination edge and creating a route between them. A file generated like this has the `.rou.xml` extension.

```
python randomTrips.py -n map.net.xml -r gmu.rou.xml -e 100 -l
```

3.1.4 Generating SUMOCFG

By using the network file `map.net.xml` and the route file `map.rou.xml` generated earlier, we can manually create the `map.sumocfg` by just indicating the network and route files in the `.sumocfg` file. It's structure is as shown below.

```
<configuration>
  <input>
    <net-file value="gmu.net.xml"/>
    <route-files value="gmu.rou.xml"/>
  </input>
  <time>
    <begin value="0"/>
    <end value="300"/>
  </time>
</configuration>
```

3.1.5 Simulation in SUMO

The `.sumocfg` generated in the earlier steps can be opened in SUMO and a simulation can be started. The traffic network and the simulation in SUMO can be seen in the Figure 3.2.



Figure 3.2: Network in SUMO

Chapter 4: Implementation

This implementation was achieved with interfacing SUMO which is a traffic simulator and Unity which is a powerful game engine through TraCI. If using TraCI, SUMO is started with the `--remote-port` option which starts it in a server mode and looks for any connections. An external application, viz. Unity in this case, can connect to the port specified with the `--remote-port` option and gather simulation data or control the simulation[12]. The following figure shows the interaction between SUMO which works like a server and a client through TraCI.

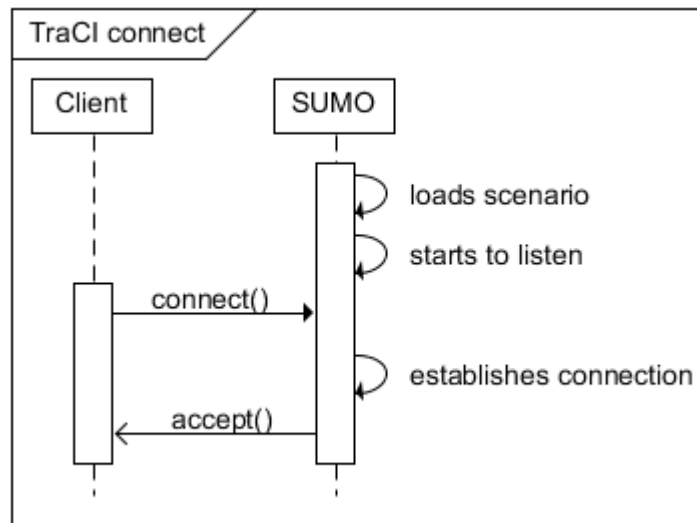


Figure 4.1: TraCI Protocol

TraCI protocol was implemented as a TraCI as a Service (TraaS) library and was provided as a Java Archive (JAR) file with the SUMO package. As there was no implementation of TraCI in C#, a tool called IKVM.NET[13] was used to generate a .NET library in the

form of a Dynamic-link Library (`.dll`) file from the given `.jar` file. After downloading `traci.jar`, the following command was used to convert it into a `traci.dll` file.

```
ikvmc traci.jar
```

This file is placed in the Assets directory of the Unity project along with the other files from the IKVM.NET package. A scenario was created in Unity by using the data retrieved from SUMO through TraCI which is described in the following sections.

4.1 Basic Environment

The simulation starts in Unity by launching SUMO using the `map.sumocfg` file generated earlier with `--remote-port` option. The paths to the SUMO executable and the `map.sumocfg` file can be specified in Unity. Unity tries to connect to SUMO on the same port that SUMO was started on through Unity and since SUMO is waiting for any incoming connections, when Unity sends a request, SUMO connects through TraCI. Once the connection is established, we can initialize the GameObjects in SUMO for the vehicles and lanes using the prespecified prefabs.

4.2 Creating Lanes

We start this phase by reading information related to each lane from SUMO and subsequently create the respective lanes in Unity. The lane IDs are obtained using the `do_job_get()` method and the data is saved in a list as individual entries. These entries can be parsed and the coordinates can be obtained using which we can build the lanes in Unity. The positions of the lanes obtained follow the (x, y) coordinate system that is in two dimension which needs to be converted to a three dimension system with the (x, y, z) coordinates. Using the `Vector3` structure provided by Unity, we can correlate x and y coordinates to x and z coordinates while using a null value for the y coordinate. By using this edge-to-edge data in the 3D environment, we can repeat the lane prefab to create a lane.

4.3 Creating Vehicles

This phase involves reading data regarding the machine controlled traffic generated and controlled in SUMO and creating the respective vehicles in Unity. Each vehicle ID is saved in a list as separate entries by using the `do_job_get` method. Once the IDs are in a list, it can be traversed and each vehicle data such as position, speed, etc. can be obtained. The vehicle positions are in the form of 2d vertices which can be converted to 3D by using a similar methodology to the one used to create lanes in Unity. Once the vehicles are initialized using the `Start()` method in Unity, we can regularly update the positions of vehicles by obtaining the required data in each simulation step from SUMO.

4.4 Creating the Human Driven Vehicle

Using the prespecified prefab, a `GameObject` was created in Unity and initialized at a random position in the scenario. This vehicle can be human controlled and driven around in the environment. By using TraCI, we can add a vehicle to the corresponding simulation in Unity and update its position to reflect it in SUMO.



Figure 4.2: The Simulation in Unity



Figure 4.3: Autonomous traffic

Chapter 5: Future Work

This simulator could include many other kinds of traffic like pedestrians or other vehicular traffic like bicycles, emergency vehicles, etc. This would bring it even closer to a real-life simulation.

The TraaS library used to interface SUMO and TraCI could be replaced with a C# port or C# implementation of TraCI. This would reduce some cumbersome errors and might improve the simulation speed.

Chapter 6: Conclusion

The aim of this thesis was to create an environment in which both machine controlled vehicles and human driven vehicles could be co-simulated in the same track. This was achieved by using SUMO which is a traffic simulator and Unity which is a game engine that were interfaced using the TraCI protocol. The TraCI protocol which was available as a Java library was converted to a .Net library by using IKVM.NET so that it can be used in Unity. Along with describing the means of achieving this simulation, previous work has been described where most of them involve using virtually created environments for the simulation. And some of them, do not involve a human driven vehicle in the simulation. Hence by using the geographical data retrieved from OSM, it was possible to create a real-life environment in Unity through TraCI.

REFERENCES

- [1] “monodrive: Autonomous vehicle simulator.” [Online]. Available: <https://www.monodrive.io/>
- [2] L. Simulator, “Lgsvl simulator.” [Online]. Available: <https://www.lgsvlsimulator.com/>
- [3] LGSVL, “Apollo,” Apr 2019. [Online]. Available: <https://github.com/lgsvl/apollo>
- [4] “Autoware,” Mar 2019. [Online]. Available: <https://github.com/lgsvl/Autoware>
- [5] “Duckietown,” Sep 2018. [Online]. Available: <https://github.com/lgsvl/duckietown2>
- [6] Microsoft, “Airsim,” Apr 2019. [Online]. Available: <https://github.com/Microsoft/AirSim>
- [7] “Adas autonomous vehicle simulation — realtime technologies.” [Online]. Available: <https://www.faac.com/realtime-technologies/solutions/research-simulation-adas-autonomous-vehicles/>
- [8] “Autonomous vehicle radar: Improving radar performance with simulation.” [Online]. Available: <https://www.ansys.com/about-ansys/advantage-magazine/volume-xii-issue-1-2018/autonomous-vehicle-radar>
- [9] “Autonomous vehicle intelligent vehicle simulation software — co-simulation.” [Online]. Available: <https://www.opal-rt.com/autonomous-vehicle/>
- [10] “Autonomous and adas simulation platform.” [Online]. Available: <https://www.cognata.com/>
- [11] “Openstreetmap.” [Online]. Available: <https://www.openstreetmap.org/export>
- [12] “Traci.” [Online]. Available: <https://sumo.dlr.de/wiki/TraCI>
- [13] “Ikvm.net home page.” [Online]. Available: <https://www.ikvm.net/>

Biography

Raghavendra Somepalli received his Bachelor of Technology from Anurag Group of Institutions, India in 2017.