Vehicle-Pedestrian Interaction in SUMO and Unity3D

Leyre Artal-Villa[1] and Cristina Olaverri-Monreal[2*[0000-0002-5211-3598]]

[1] Public University of Navarra, Dept. Electrical and Electronic Engineering,
Campus de Arrosadía s/n 31006 Pamplona, Spain.
leireartalvilla@gmail.com
[2] Johannes Kepler University Linz, Austria, Chair for Sustainable Transport Logistics 4.0
Altenberger Straße 69, 4040 Linz, Austria
cristina.olaverri-monreal@jku.at

Abstract. Road fatalities that involve Vulnerable Road Users (VRU) outnumber in some countries and regions ones that involve vehicular drivers and passengers. As most vulnerable road user fatalities happen in urban areas, where the traffic conditions are more demanding and an increased pedestrian interaction can result in unpredictable scenarios, it is imperative to study solutions to reduce the high rate of accidents in which pedestrians are involved. To this end, we present in this paper a simulation framework that provides a framework to generate a variety of pedestrian demands to simulate vehicle-pedestrian interaction and vice versa. A Transmission Control Protocol (TCP) connection combines the game engine Unity 3D with the Simulation of Urban Mobility (SUMO) open source traffic simulator. After creating the 2D scenario SUMO was connected with Unity 3D by using the Traffic Control Interface (TraCI) Protocol and TraCI as a Service (TraaS) library. The motion in Unity took place after instantiating the pedestrians retrieved from SUMO. The system was evaluated by detecting and visualizing pedestrians and vehicles that were within a specific range.

Keywords: VRU, P2V, V2P, SUMO, Unity 3D

1 Introduction

According to the World Health Organization, 85 000 people die annually from road traffic injuries in the European Region [1]. Urban areas have more complex intersections and a higher number of pedestrians and cyclists that might be overseen by drivers [2], particularly if they are involved in non-driving tasks. Due to the increased complexity in urban road and traffic scenarios, vulnerable road users (VRUs) are involved in more fatalities.

Pedestrians are especially vulnerable as road users because an armored vehicle does not protect them, nor do they wear any protective helmets [3]. Addressing the risk of death in road traffic is fundamental to achieve the Sustainable Development Goals

* Corresponding author

This is a pre-copyedited version of a contribution published in New Knowledge in Information Systems and Technologies. WorldCIST'19 2019. Advances in Intelligent Systems and Computing, vol 931, Rocha Á., Adeli H., Reis L., Costanzo S. (eds) published by Springer, Cham. The definitive authenticated version is available online via https://doi.org/10.1007/978-3-030-16184-2_20

(SDGs), by targeting efficient transportation services that do not affect health security. Therefore, it is essential to reduce mortality and injuries derived from road crashes [4]. In this context, P2V (Pedestrian-to-Vehicle) and V2P (Vehicle-to-Pedestrian) communication technologies have become crucial.

We present a platform to evaluate P2V and a V2P communication in a simulated 3D environment that includes the microscopic modeling of vehicles and pedestrians relying on the recent release of the last version of Simulation of Urban Mobility (SUMO) and TraaS (TraCI as a Service).

Next section outlines related literature in the field. Section 3 delineates the proposed work and describes the pedestrian modeling, creation of the scenario in SUMO and the connection of SUMO with Unity 3D. Section 4 describes the 3D simulation in Unity 3D and Section 5 presents the process to evaluate the implemented system. Finally, Section 6 concludes the work.

2 Related Literature

Communication targeting VRU protection addresses the use of smart devices to send Personal Safety Messages (PSM), over a wireless communication channel. This approach is also known as General Packet Radio Service (GPRS)-based, since the smart devices use the latitude and longitude coordinates, which are then transformed into local coordinates to estimate the relative position of the communicating pedestrians and vehicles [5]. Vehicle-to-Pedestrian (V2P) and Pedestrian-to-Vehicle (P2V) communication or a combination of them rely on GPRS.

Pedestrian detection has been the focus of research in many works. For example, by using AdaBoost and support vector machine algorithms [6] or by detecting and tracking pedestrians using cameras, (i.e. from a moving vehicle using both Histogram of Oriented Gradients (HOG) and Kalman filter [7] or by using the back-camera of a mobile device using image-processing techniques [8, 9]).

By using both P2V and V2P communication, the authors in [10] developed an application based on a collision-prediction algorithm. The proposed application broadcast the device's position to the vehicles nearby, and reciprocally broadcasts the vehicular position to the pedestrians nearby.

As far as simulation tools are concerned, PARAMICS, VISSIM, AIMSUM and SUMO stand among the most recognized simulators, which use microscopic models and allow the inclusion of pedestrian flow in the simulation [11]. For example, PARAMICS has its own software to simulate pedestrian behavior in real word environments. VISSIM allows making a 3D simulation with pedestrians, but it fails in the calibration process of certain parameters and it is a difficult pro-gram to handle [12]. AIMSUM enables pedestrian-vehicle interactions at uncontrolled, actuated-controlled or fixed-controlled intersections, but it is not convenient for navigating between different periods and it does not provide background maps [13].

SUMO is an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks [18]. The recent release from December 2017 of the last version (0.32.0) together with the last version of TraaS

from August 2017 reveal the possibility of performing a remote-controlled realistic 3D traffic scenario, which includes pedestrians. The aforementioned version of SUMO implements Traffic Control Interface (TraCI) which provides the necessary commands for both remotely retrieving and changing the state of pedestrian objects.

By means of TraaS library and the above mentioned methods, authors in [14] retrieved the information (vertices, length, width, type) of lanes and cars (speed, position, angle) from SUMO and instantiated them as game objects in Unity 3D. Authors in [15, 16] reused the translated library and connected the traffic light system data from SUMO into Unity 3D.

As the integration of pedestrians in current implemented simulators add realism to the scenarios and use cases, we focus on developing a 3D visualization of traffic, which includes both pedestrians and vehicles from a driver centric perspective simultaneously.

3 Framework Implementation

Relying on the work presented in [10] we implemented a scenario in order to establish connections between vehicles and pedestrians and vice versa as part of the SUMO OSM environment, an option in the CoAutoSim3D simulation platform [17]. The CoAutoSim3D simulator is independent from the operating system and always up-to-date with the latest version of Unity 3D. When running 3DCoAutoSim the user can choose among several environments. One of them is the SUMO OSM environment: a dynamic and configurable environment by the user. It allows for one to take advantage of the benefits provided by a microscopic 2D traffic-modeling tool (SUMO) and a powerful game-engine (Unity 3D). In fact, due to the flexibility provided by this option, the user can perform a 3D realistic simulation of any area of interest. Furthermore, the 3D reconstructed map presented in [14] was used to maintain the high quality visualization provided by the implemented environment.

When the user selects the SUMO OSM environment, the simulator accesses a SUMO configuration file (`.cfg or .SUMOcfg`) that contains all the necessary information to conduct the simulation in SUMO and performs the Transmission Control Protocol (TCP) connection to SUMO. SUMO acts as a server and Unity 3D as a client. The simulation can be performed by loading the configuration file in Unity 3D. To implement the 3D simulation with the pedestrian-detection system we proceeded as follows.

3.1 Pedestrians Modeling in SUMO

Pedestrians in SUMO need dedicated lanes and areas, which differ from the rest of the elements of a scenario in terms of attributes that determine their behavior.

For example, when walking along an edge, pedestrians use sidewalks if they are available. With respect to zebra crossings, there are two possible cases:

- If the network contains walking areas, pedestrians may only cross a street whenever there is a pedestrian crossing.

- However, if the network does not include walking areas, pedestrians will move between any two edges that allow pedestrians at an intersection.

Pedestrian crossing behavior does not only depend on the type of roads, but also on the vehicles. SUMO contains rules to mimic ideal traffic scenarios by targeting the avoidance of collisions between vehicles and pedestrians. Therefore, pedestrians will only use a crossing if the whole length of the crossing is free of vehicles for the whole time needed to cross. This behavior is not expected and not occurring in real life. If a vehicle occupies the whole width of the lane and gets too close to a pedestrian, the pedestrian may briefly move to the side of the lane in order to let the vehicle pass.

The last version of SUMO provides a framework to generate a variety of pedestrian demands both in an explicit and random fashion, defined as follows:

- Explicit: manually defining the pedestrian movement in an .xml file.
- Random: using the tool `randomTrips.py` with the option `--pedestrian` that supports generating random pedestrian demand.

3.2 Simulation in SUMO: 2D Scenario

The approach taken for explaining the generation of the 2D scenario consists of four steps:

- Generation of a realistic network from OSM data
- Edition of the network with `netedit` and `netconvert`
- Generation of the vehicle routes and the pedestrian demand
- Executing a simulation with pedestrians in SUMO-GUI

In order to have a realistic scenario, a network was imported from Open Street Map (OSM). Since the conversion process included some imperfections that made the simulation differ from reality, the network was edited with `netedit`, which is a Graphical User Interface (GUI) application for editing traffic networks. It can be used to create networks from scratch and to modify all aspects of existing networks. Fig. 1 shows the `.net` file obtained from the OSM WebWizard before editing. Fig. 2 represents an example of pedestrian topology in `netedit` before and after edition.

`Netconvert` is a network generator that provides SUMO-format networks. After obtaining the network file (`.net` file) and generating both the vehicle routes and the pedestrian demand, the simulation was executed in SUMO-GUI, which is the same application as SUMO, just extended by a graphical user interface.

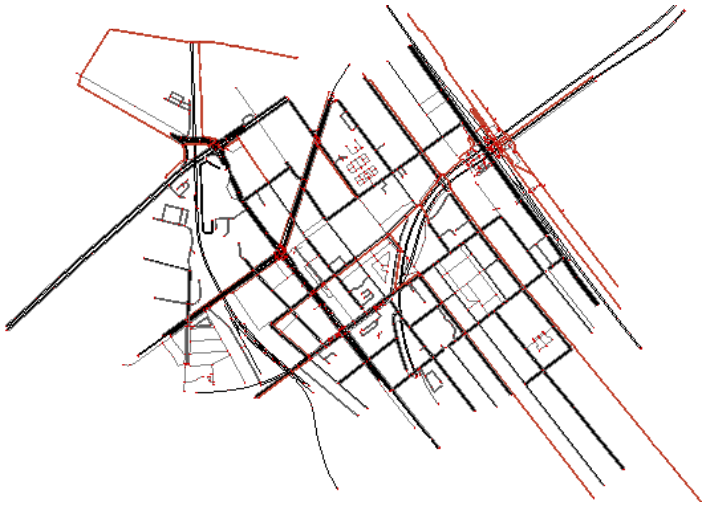
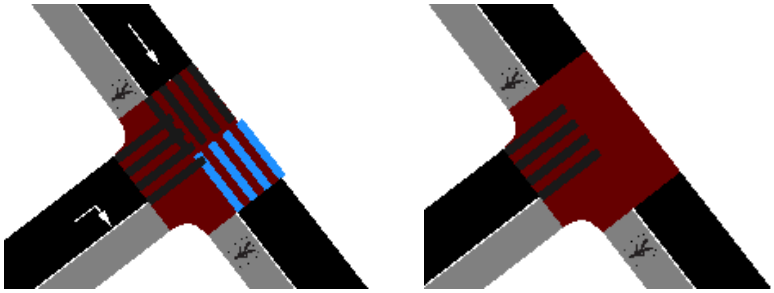


Fig. 1. .net file obtained from OSM WebWizard before editing.



A) B)

Fig. 2. Pedestrian topology in `netedit` before (A) and after (B) edition.

3.3 Connecting SUMO with Unity 3D: TraCI Protocol and TraaS library

As previously mentioned 3DCoAutoSim connects both a microscopic traffic simulator, SUMO, and a powerful 3D Graphic Engine, Unity 3D, by setting a TCP connection between them. TraCI is the Application Programming Interface (API) used for that purpose. It is based on a client/server architecture and it enables the retrieval of values from simulated objects from the server (SUMO) to the client (Unity 3D) and their behavior manipulation on-line, as well as to execute the simulation from the client side.

A TraaS library with a class for pedestrians that contained the necessary TraCI methods and which could be imported in Unity 3D was required for retrieving and changing information about pedestrian objects between SUMO and Unity 3D without changing the internal logics and structure of the already implemented and evaluated 3DCoAutoSim simulator. The implemented class made it possible to use the available commands to retrieve the values of each person from TraCI.

4 Simulation in Unity

The 3D simulation contains buildings created with the 3D city editing and visualization tool CityEngine. In addition it consist of the following objects that were retrieved from SUMO: car lanes, vehicles, sidewalks, pedestrian crossings, pedestrians, manually placed traffic lights and road markings, a user-controlled vehicle, a bidirectional communication system between cars and pedestrians using P2V and V2P and as many GPS as the number of pedestrians in the simulation. The motion of the pedestrians in the simulation occurred after their instantiation as follows.

The SUMO-retrieved-objects are instantiated in Unity 3D as Game-Objects. In order to have each SUMO pedestrian correctly classified in Unity, a class Pedestrian is created, which has the following attributes: position, id, speed and angle. The class `Pedestrian` also contains some `getters` that allow to get both X and Z coordinates of the position of each pedestrian (`getX()` and `getZ()`), the angle (`getAngle()`), the speed (`getSpeed()`) and the ID (`getId()`).

When the connection between both programs has been established and all the lanes retrieved from SUMO have been read, the function `ReadPedestrians()` is called (every 5 frames in Unity in order to match the SUMO timestamp), which is in charge of creating a list of pedestrians that will be further printed and moved in the environment (in `PrintPedestrians()`). The code below depicts which parameters are received from SUMO and how information of every pedestrian in the Unity list is updated each time-step.

Thus, every time there is a change in SUMO, the list is updated and filled with the pedestrians existing in that time-step. The position, the ID, the speed and the angle for every pedestrian is retrieved from SUMO using TraCi commands contained in the so-called “person value retrieval”.
Then, a pedestrian-object is created in Unity 3D for each simulated pedestrian in SUMO and its attributes position, id, speed and angle are set to the SUMO 2D position, the SUMO id, the SUMO speed and the SUMO 2D angle (in degrees, with respect to the Y axes) of every simulated pedestrian respectively. The created pedestrian object is added to the list of pedestrians in the last position. It should be noted that until now, nothing has been displayed in Unity 3D, as the information about the simulated pedestrians is in Unity, but it cannot be seen in the game view or in the scene view until all the pedestrians from the list are printed in Unity and moved in `PrintPedestrians()`.

```
Function ReadPedestrians() SUMOmanager.cs
//-Reading the retrieved pedestrians from SUMO-
public void ReadPedestrians()
{
//Start with an empty list every SUMO time-step
pedlist.Clear();
//Read the ID-s of the pedestrian in the current time-
step
```

```
pedIDs = (SumoStringList)sumoTraciConnection.do_job_get(
Person.getIDList());
//For every pedestrian in the list
foreach (string perid in pedIDs)
{
//--Retrieving information from SUMO-
//SUMO 2D position of this pedestrian
positionperson =
(SumoPosition2D)sumoTraciConnection.do_job_get(Per-
son.getPosition(perid));
//SUMO speed of this pedestrian
speedperson=
((java.lang.Double)sumoTraciConnection.do_job_get(
Person.getSpeed(perid))).doubleValue();
//SUMO angle of this pedestrian
angleperson =
((java.lang.Double)sumoTraciConnection.do_job_get(
Person.getAngle(perid))).doubleValue();
//Create the pedestrian object with the SUMO information
of this pedestrian
pers = new Pedestrian(positionperson, perid, speedperson,
angleperson);
//add this pedestrian object to the list
pedlist.Add(pers);
}
//Create the pedestrian game-object in Unity 3D
// if it has not been previously created or update
// the game-object if it has been previously created
PrintPedestrians();
//Perform another time-step in SUMO
sumoTraciConnection.do_timestep();
}
```

5 System Evaluation Procedure

To evaluate the framework we implemented a bidirectional communication between vehicles and pedestrians and detected whether they were within the specific range of 40m.

To visualize that the V2P communication had been achieved in the simulated 3D environment, the pedestrians were color-coded, pedestrians' detection (P2V communication) by the vehicle was visualized through an object-bounding box. Fig. 3 shows an example for the pedestrians' detection visualization.

The list of all the simulated pedestrians was iterated and an array of renderers was created for every person-object. A renderer is what makes an object appear on the

screen. In this case, renderers were used to access and modify the person-objects. The array stored all renderers of each person-object as a component in its children elements. Then, the difference between the position of each pedestrian and the position of the user-controlled car was obtained. After that, the list of all the simulated cars was iterated.

For each pedestrian i, a subtraction was calculated and vice versa between the position of that pedestrian and the position of each simulated car. At this point, if the distance of pedestrian i and any simulated car or the distance of the pedestrian i and the user-controlled car was less than 40m, the color of the pedestrian object i, which represents the pedestrian with `id=i` in SUMO, was changed in the 3D environment from the initial color (white) to red. If the distance was more than 40m and the i_{th} pedestrian had been previously red colored, the pedestrian object was reversed to white. This procedure helped to represent the pedestrians that were aware of the presence of vehicles in the surroundings.

To establish a P2V communication, a pedestrian list was iterated. Since the function `OnGUI()` is called every frame, the variable `pedlist` contained all the pedestrians that existed at a given time in SUMO and therefore all the pedestrians that had been instantiated as Game-Objects in Unity. For every pedestrian the difference between its position and the position of the user-controlled vehicle was obtained by subtracting both 3D positions and computing the norm of the three dimensional vector obtained because of the mentioned subtraction.

The above describe strategy has been proved to be effective to establish the bidirectional communication between the pedestrians and the vehicles in the surroundings.

Fig. 3. Example of pedestrians' detection by using an object-bounding box.

6 Conclusion and Future Work

We proposed in this work a system to detect VRU and vehicles in the vicinity as a means to enhance road safety. The system has the potential to assist both drivers and

pedestrians in a variety of situations in which technology used by both parties is evaluated. The implementation of a framework that relies on the linking through a TCP connection of the game engine (Unity 3D) to access data from the SUMO open source traffic simulator has been proved to be capable of simulating vehicle-pedestrian and pedestrian-vehicle interaction. Future work will be pursued by validating the simulation framework in a variety of use cases.

Acknowledgments. This work was supported by the BMVIT endowed Professorship Sustainable Transport Logistics 4.0 and the Erasmus Program, code A WIEN 20.

References

1. Jackish, J., Sethi, D., Mitis, M., Szymañski, T., Arra, I. European facts and the Global status report on road safety 2015. Copenhagen: WHO Regional Office for Europe; 2015 http://www.euro.who.int/__data/assets/pdf_file/0006/293082/European-facts-Global-Status-Report-road-safety-en.pdf?ua=1, last accessed 13 October 2018.
2. BrainonBoard.ca Vulnerable Road Users: Pedestrians and Cyclists [Online]. Available: http://brainonboard.ca/program_resources/VulnerableRoadUsersPedestriansandCyclists_Fact_Sheet_Eng_4.pdf last accessed 13 Oct 2018.
3. Olaverri-Monreal, C., Pichler, M., Krizek, G. C., Naumann, S. Shadow as Route Quality Parameter in a Pedestrian-Tailored Mobile Application, In IEEE Intelligent Transportation Systems Magazine. Volume: 8, Issue: 4, pp. 15-27 (2016)
4. World Health Organization Europe Road Safety: Fact sheets on sustainable development goals: health targets, http://www.euro.who.int/__data/assets/pdf_file/0003/351444/3.6-Fact-sheet-SDG-Road-safety-FINAL-10-10-2017.pdf?ua=1, last accessed 2 Oct 2018.
5. Rostami, A., Cheng, B., Lu, H., Gruteser, M., Kenney, J .B. Reducing Unnecessary Pedestrian-to-Vehicle Transmissions Using a Contextual Policy, In: Proc. 2nd ACM Int. Work. Smart, Auton. Connect. Veh. Syst. Serv. - CarSys '17, pp. 3–10 (2017).
6. Guo, L., Ge, P. S. , Zhang, M. H., Li, L. H., Zhao, Y. B. Pedestrian detection for intelligent transportation systems combining AdaBoost algorithm and support vector machine, In: Expert Syst. Appl., vol. 39, no. 4, pp. 4274–4286 (2012).
7. Nkosi, M.P., Hancke, G.P., dos Santos, R.M.A. Autonomous pedestrian detection. In: AFRICON IEEE 10.1109/AFRCON.2015.7332014. (2015)
8. Allamehzadeh, A., Olaverri-Monreal, C. Automatic and manual driving paradigms: Cost-efficient mobile application for the assessment of driver inattentiveness and detection of road conditions. In: IEEE Intell. Veh. Symp. Proc., pp. 26–31 (2016).
9. Allamehzadeh, A., Urdiales de la Parra, J., Garcia, F., Hussein, A. and Olaverri-Monreal, C. Cost-Efficient Driver State and Road Conditions Monitoring System for Conditional Automation, In: Proceedings IEEE Intelligent Vehicles Symposium, Los Angeles, USA, pp. 1497-1502 (2017).
10. Hussein, A., García, F., Armingol, J. M., Olaverri-Monreal, C. P2V and V2P communication for pedestrian warning on the basis of autonomous vehicles. In: IEEE Proceedings Intelligent Transportation Systems Conference (ITSC), pp. 2034–2039 (2016).
11. Kokkinogenis, Z., Sanchez Passos, L., Rossetti, R., Gabriel, J. Towards the next-generation traffic simulation tools: a first evaluation, 6th Iber. Conf. Inf. Syst. Technol., pp. 15–18 (2011).

12. Koh S.Y. Doina., Chin H.C, Traffic Simulation Modelling: VISSIM, https://docplayer.net/9916680-Traffic-simulation-modeling-vissim-koh-s-y-doina-1-and-chin-h-c-2.html, last accessed 14 Nov. 2018.
13. Salgado, D., Jolovic, D., Martin, P. T., Aldrete, R. M. Traffic Microsimulation Models Assessment - A Case Study of International Land Port of Entry, In: Procedia Comput. Sci., vol. 83, no. Ant, pp. 441–448 (2016).
14. Biurrun-Quel, C., Serrano-Arriezu, L., Olaverri-Monreal, C. Microscopic Driver-Centric Simulator: Linking Unity 3D and SUMO, In: Rocha Á., Correia A., Adeli H., Reis L., Costanzo S. (eds) Recent Advances in Information Systems and Technologies. AISC, volume 569, pp. 851-860, Springer, Cham (2017).
15. Olaverri-Monreal, C., Errea-Moreno, J., Díaz-Álvarez, A. Implementation and Evaluation of a Traffic Light Assistance System in a Simulation Framework based on V2I Communication, Journal of Advanced Transportation, vol. 2018, Article ID 3785957, 11 pages, 2018. https://doi.org/10.1155/2018/3785957.
16. Olaverri-Monreal, C., Errea-Moreno, J., Díaz-Álvarez, A., Biurrun-Quel, C., Serrano-Arriezu, L., Kuba, M. Connection of the SUMO Microscopic Traffic Simulator and the Unity 3D Graphic Engine to Evaluate V2X Communication-Based Systems. Sensors Journal, vol. 18, number 12, pages=439, doi:10.3390/s18124399, Multidisciplinary Digital Publishing Institute (2018)
17. Hussein, A., Diaz-Alvarez, A., Armingol, J. M., Olaverri-Monreal, C. 3DCoAutoSim: Simulator for Cooperative ADAS and Automated Vehicles. In: Proceedings 21st International IEEE Conference on Intelligent Transportation Systems, ITSC2018, Hawaii, pp. 3014-3019, (2018).
18. SUMO - Simulation of Urban Mobility http://sumo.dlr.de/index.html, last accessed 2 Dec 2018.