An Integrated Framework for Mobile-Based ADAS Simulation

João S.V. Gonçalves, João Jacob, Rosaldo J.F. Rossetti, António Coelho and Rui Rodrigues

Abstract The increasing number of vehicles and mobile users has led to a huge increase in the development of Advanced Driver Assistance Systems (ADAS). In this paper we propose a multi-agent-based driving simulator which integrates a test-bed that allows ADAS developers to compress testing time and carry out tests in a controlled environment while using a low-cost setup. We use the SUMO microscopic simulator and a serious-game-based driving simulator which has geodata provided from standard open sources. This simulator connects to an Android device and sends data such as the current GPS coordinates and transportation network data. One important feature of this application is that it allows ADAS validation without the need of field testing. Also important is the suitability of our architecture to serve as an appropriate means to conduct behaviour elicitation through peer-designed agents, as well as to collect performance measures related to drivers' interaction with ADAS solutions.

Keywords Mobile ADAS · Driving simulators · Serious games · SUMO

J.S.V. Gonçalves · R.J.F. Rossetti (✉)
Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Laboratório de Inteligência Artificial e Ciência de Computadores, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
e-mail: rossetti@fe.up.pt

J.S.V. Gonçalves
e-mail: joao.sa.vinhas@fe.up.pt

J. Jacob · A. Coelho · R. Rodrigues
Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, INESC TEC, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
e-mail: joao.jacob@fe.up.pt

A. Coelho
e-mail: acoelho@fe.up.pt

R. Rodrigues
e-mail: ruirodrig@fe.up.pt

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*, Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_10

1 Introduction

The technological advances in both the mobile and the transportation industries are remarkable. This has made the development of ADAS an interesting topic [1]. However, even though most high-end cars nowadays ship with built-in embedded systems, most of the older cars do not have such devices. This brings about an interesting research opportunity, which is to develop and test ADAS that run on low-cost devices, such as an Android tablet or smartphone.

The main goal of this paper is to describe the methodology of our Multi Agent System (MAS) based driving simulator, integrating SUMO microscopic simulator with driving simulators. We also intend to describe our implementation of a test-bed to easily develop ADAS using the system, simulating their use in a low-cost and controlled environment.

There are several benefits to testing ADAS in a simulated environment rather than in real-world scenarios. As the tests are not conducted in a real physical location, they are not subjected to travel times, traffic or other adverse conditions which could render them mute. This, as well as being able to deploy the simulator in low-cost computers, and therefore reaching more test subjects, leads to time compression of the tests. Noticeably, cost reduction is another significant benefit as the electrical cost of running a simulator is dismissive when compared to fuel costs of real-world testing. Besides preventing the safety risks inherent to driving, simulation allows us to control the test environment and manipulate it according to the specificities of the ADAS being tested.

The objective of our work was to develop the MAS and include a test-bed that was easy to implement and replicate in low-cost environments. We aimed to combine SUMO microscopic simulator with IC-DEEP, which is a driving simulator developed at LIACC [2], with the Geostream framework developed at SI and CG, as well as to enhance them with logs of simulated GPS positions in a mobile device. To achieve so, a mobile application/service was developed in order to receive this communication from the simulator and override the default GPS sensor of the device. We wanted to make it easy to extend the communication between the simulator and a mobile device, providing the latter with more information such as the current speed limit, semaphoric information, or other data from the network.

This work aims to contribute with a novel multi-faceted methodology to simulate and research multiple human factors in Intelligent Transportation Systems (ITS) and, particularly, with a novel approach to test ADAS that will enable developers to validate and test their applications more easily and efficiently while reducing costs.

In the following sections we describe the development and results of our implementation. In Sect. 3 we introduce some related state-of-the-art works on the subject of ITS, focusing on simulation, on the integration of different scope simulators and also on the topic of serious games. We then describe our approach, architecture and development details. Finally we present our preliminary verification as well as their analysis in Sect. 5. We finish this paper with a set of conclusions and interesting future work.

2 Background and Related Work

The Artificial Transportation Systems (ATS) [3, 4] concept has been one of the main research topics in the IEEE ITS Society [5]. A typical approach to ATS modeling and development is the MAS metaphor. Another potentially concomitant approach is the HLA concept, which is based on the idea of distributed simulation so as to meet the requirements of all usages and users rather than of a single simulation model and analysis perspective [6].

In [7], authors propose to integrate a driving simulator and a traffic microsimulator, in an attempt to tackle the mutual-dependence between the driver's behavior and traffic conditions.

Combining SUMO microscopic traffic simulation [8], using MAS capabilities, with other simulators has also been researched [9]. Authors in [6] have studied an HLA-based approach to simulate electric vehicles in Simulink and SUMO. Driver-centric simulation has been researched by authors in [10], where they have developed a simulation tool that provides feedback to the network based on the driver's behaviour.

Driving simulators are no doubt an important tool when researching ATS, specially so when studying the influence of human factors in driving faults [2]. These faults often occur in direct consequence of performing secondary tasks while driving [11]. In [12] authors introduce a game-engine-based modeling and computing platform for ATS. They describe the artificial population both in their macroscopic and microscopic aspects.

Regarding driving simulators with ADAS testing capabilities, authors in [13] propose a reconfigurable driving simulator with several components to accommodate different ADAS testing or training. A framework for ADAS assessment and benchmarking has been developed in [14], with configurable scenarios and 3D scenes and multiple sensors input.

Authors in [15] developed a Full Speed Range Adaptive Cruise Control with their platform for ADAS prototyping and evaluation, SiVIC. The platform is capable of reproducing vehicle and sensor behaviors in a realistic fashion, according to the configured environment in the simulator. The developed platform also simulates noised and imperfect data.

A system comprising a large scale driving simulator, built in a 360° full dome with 3D scenes from real city area has been developed in [16]. The system contains a multitude of features such as real-time hardware-in-the-loop, wireless communication devices and bio signal analysis and is used to develop and test ADAS as well as Advanced Safety Vehicle, ITS infrastructure and others.

3 Methodological Approach

The proposed system architecture is as described in Fig. 1. The main module of the system is the SUMO simulator, which is responsible for the network's multi-agent microscopic simulation, and has multiple driving agents. This module provides an overview of the whole MAS and can be manipulated directly.

The SUMO module also acts as a "central server", providing all the essential information for both IC-DEEP and the High Fidelity Simulator. This information consists of the network infrastructure and the agents in the system, whereas terrain morphology and road or building geometry are provided by the Geostream framework.

Both of the driving simulators have a local representation of the whole MAS and are capable of controlling any driving agent. The simulators are also able to connect to an Android device and pass along all the information deemed necessary, such as the GPS coordinates of the current driving agent being controlled. The Android device is running a service that receives the incoming connections from the simulator and also the ADAS being tested. The dotted area in Fig. 1 corresponds to the developed components as of the writing of this paper.

3.1 Simulators and Framework

The proposed system architecture has two simulators; however, as of the writing of this paper, only the IC-DEEP simulator has been enhanced and integrated into the framework. The latter is implemented in Unity3D and has the Geostream framework embedded directly. The Geostream framework connects with Open Street Maps, Google Geolocation API, Google Altitude API and other data providers in order to fetch the required geographical information of a given location and

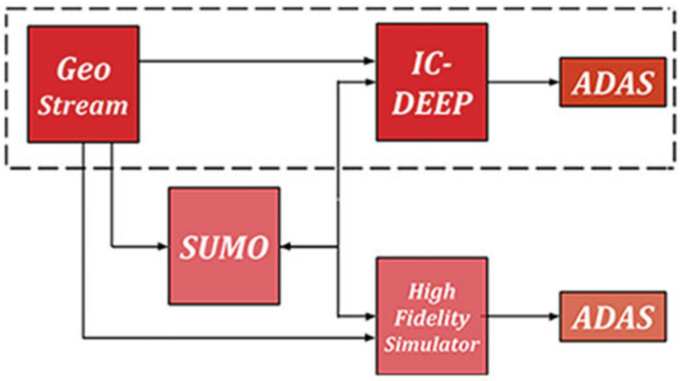


Fig. 1 Overview of the system's architecture

remodel it in a fashion which can be interpreted by all the simulators in a coherent and consistent way. This is especially important to the SUMO microscopic simulator as the raw network data imported from Open Street Maps, typically, generates unrealistic ways and intersections. The information generated by Geostream framework is then parsed into both simulators to generate a 3D scene that is representative of the chosen test location. The Geostream framework is explained in further detail in the following section.

3.2 The Geostream Framework

The Geostream Framework was originally designed to aggregate location-based data from multiple sources and recreate urban and rural environments for use in location-based games. One of the issues in location-based games is the need to accurately determine the user's current context. As such, the Geostream provides a foundation on top of which location-based games can be developed. However, since the information is of use to other areas (simulation, procedural modeling), the framework was further altered so that it can be used in non-game development.

The framework accesses the following web services, using them as providers for the respective context-sensitive data (Table 1):

The data retrieved by these sources is then combined as a means to recreate the current location context of the player (or in this case, the user). Certain sources may be optionally not used if their data adds little to the problem at hand. After loading data from the above sources, the scene graph looks similar to that of Fig. 2.

As Fig. 2 depicts, the typical scene graph of a Geostream based application consists of a World Map gameobject, and other several Geo-gameobjects (objects not based on the external sources of data, but that are placed in the game world, such as a player, enemies, cars, etc.). The World Map game object consists of

Table 1 Geostream currently used services and their data

Service	Data requested
Open street maps	Human structures information and meta-data (buildings, roads, bridges, traffic lights…) of a given area
Microsoft bing maps	Aerial photos and traffic information of given area
Google places API	Information about POIs (Points of Interest) in given area
Google elevation API	Altitude of given coordinates
Google geocoding API	Coordinates of a given address
Open weather	Detailed, current weather information of given location
Map quest	Aerial photos of a given area

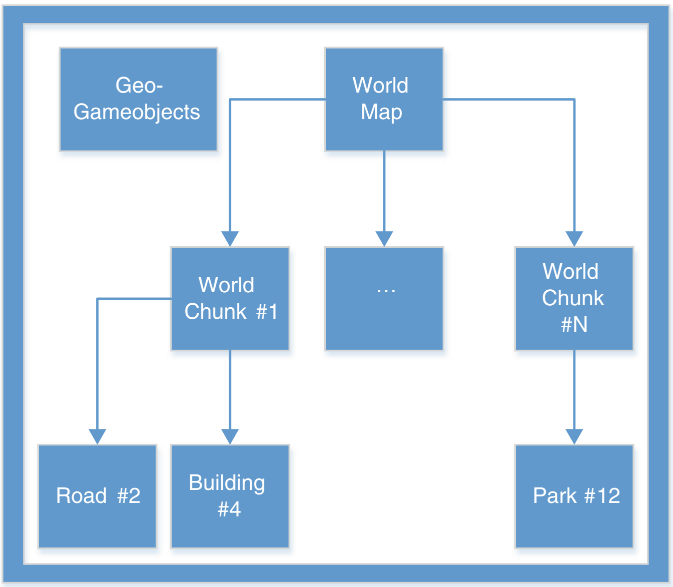


Fig. 2 Example of a Geostream scene graph

several Map chunks. Each Map chunk holds the information of a specific part of a location, such as the geometry of the terrain of that part (its elevation) and its aspect (the corresponding part of an aerial photo).

As Fig. 3 shows, a World Map is comprised of several World Chunks. In the above image, the grid mesh that is drawn with a blue color, represents the limits of a given World Chunk. In this case, the World Map consists of 16 World Chunks, each responsible for computing the information that belongs to itself (not only its geometry or texture, but those of the structures that belong to it: POIs, roads, buildings, natural structures, etc.). The behaviour of each World Chunk is summarized in the fluxogram of Fig. 4.

When a World Map is created, using a certain coordinate or address as a center reference, it will procede to create NxN World Chunks, each with a predefined width and height of fractions of degrees. Then, each World Chunk will proceed to look at the information it needs to reconstruct itself. Additionally, a World Map can have a Geo-Gameobject to “follow”. This means that as the referenced Geo-Gameobject moves around, new World Chunks may be dynamically loaded, while others are unloaded to preserve memory.

New sources of information can be added to a World Chunk for it to make use of. Additionally, it is possible to change, in design time, how certain types of meta data are processed. For instance, in Unity, in the editor view, one can visualize what current tags of Open Street Maps or Google Places are being processed and how (as it can be seen in Fig. 5).

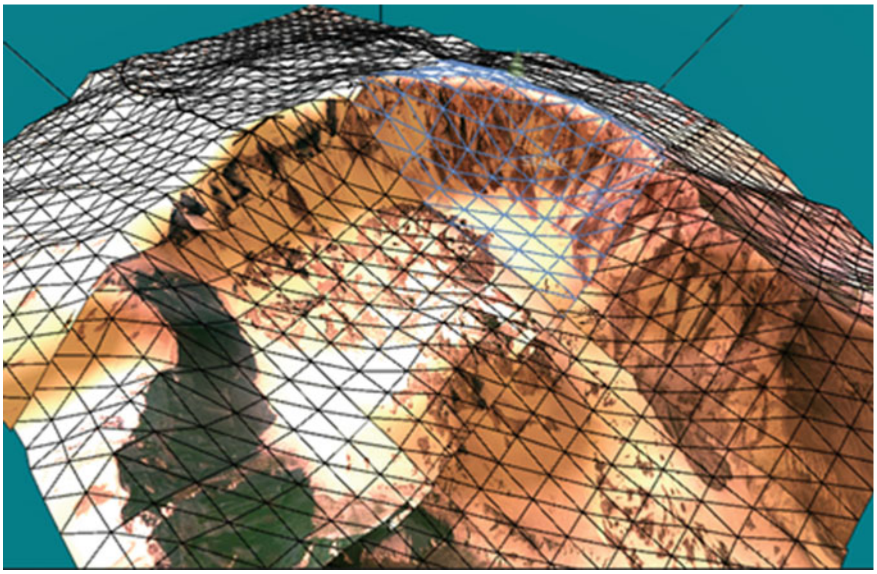


Fig. 3 Screenshot of the reconstruction of the Mount St. Helens crater

As seen in Fig. 5, a dictionary is created in design time, pairing tags with the script classes responsible for treating information related to a specific tag. When the world object is created, it will instantiate, through C# Reflection feature, the necessary scripts (all derived from the Structure Script super-class). So, whenever a structure's information with a certain tag is compiled after consulting the multiple sources, it is passed on to the respective script for treatment. In Fig. 5, we can see that "residential", "motorway", "bridge" and "highway" are all treated by the Road Script, a script responsible for generating roads with different features. This allows for further expanding the possibilities or procedurally generating other real-world objects based on the information compiled from external sources. For instance, one could add the tag "traffic light", present in the information gathered via Open Street Maps, coupled with a Traffic Light Script that would be capable of treating that specific data. It could, for example, instantiate a semaphore prefab in that structure's position. Same thing could be done with trees, lamps, and other simple and repeatable structures.

The biggest drawback of Geostream Framework, is that it is limited by the availability of the context-related sources, and the quality (or existence) of information. Performance wise, as most operations are threaded, loading of World Chunks is made in a smooth manner, albeit for areas with many structure-related information available, it can take some time to generate all the needed geometry (even so, the FPS of the simulation rarely becomes affected). In locations with abundant information and with the needed scripts performing procedural generation of structures, the results are both visually appealing and accurate (note that generated structures fit with the structures captured by the aerial photos).

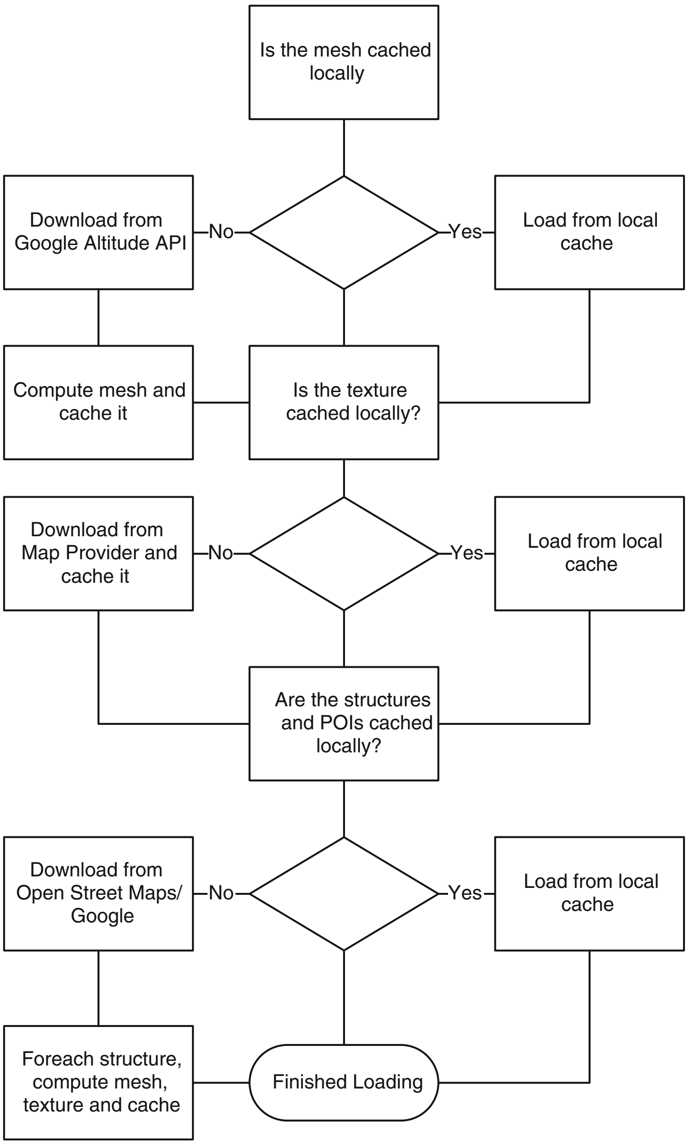
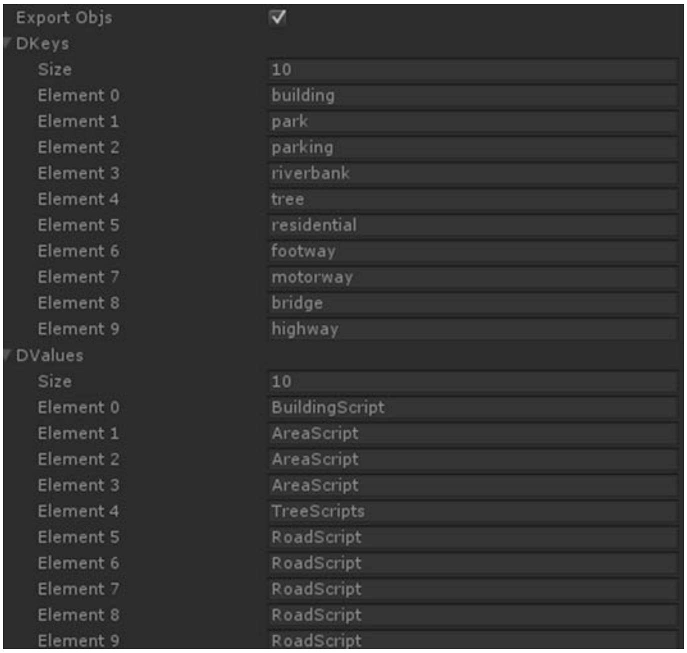

Is the mesh cached locally
Download from Google Altitude API
No
Yes
Load from local cache
Compute mesh and cache it
Is the texture cached locally?
Download from Map Provider and cache it
No
Yes
Load from local cache
Are the structures and POIs cached locally?
Download from Open Street Maps/ Google
No
Yes
Load from local cache
Foreach structure, compute mesh, texture and cache
Finished Loading

Fig. 4 Depiction of the loading behavior of a World Chunk

Notice how in Fig. 6, some buildings are red, while some are not. Red buildings are those that have all their geometry's height defined at the source, while textured buildings have their height generated based on the height of neighboring buildings and those whose heights are known from the source. Buildings with extra detail can also be computed by further developing the Building Script, associated with the creation of those structures.



Export Objs	✓
DKeys	
Size	10
Element 0	building
Element 1	park
Element 2	parking
Element 3	riverbank
Element 4	tree
Element 5	residential
Element 6	footway
Element 7	motorway
Element 8	bridge
Element 9	highway
DValues	
Size	10
Element 0	BuildingScript
Element 1	AreaScript
Element 2	AreaScript
Element 3	AreaScript
Element 4	TreeScripts
Element 5	RoadScript
Element 6	RoadScript
Element 7	RoadScript
Element 8	RoadScript
Element 9	RoadScript

Fig. 5 Subsection of the World Map behaviour script

3.3 Mobile Device

The Android service sets the current GPS location using *Mock Location* API to override the default location provider. All applications running on the mobile device that use or perceive the current location will also be affected by the running service.

The new GPS coordinates are sent from the simulator every second; however, this value is a parameter of the simulator and so can be adjusted to the specific needs of each scenario. The developed service can be run as a standalone application, and thus testing the ADAS mobile applications independently, as shown in the left side of Figs. 1 and 2. There is also the option to use the service as a library in any Android application, as long as it matches API level 19, as shown in the right side of Fig. 7.

3.4 Interaction of Driving Simulators and Android

A typical interaction between the simulators and the mobile devices is shown in Fig. 8. The modules are connected via TCP-IP sockets due to implementation simplicity. The communication messages are formatted in JSON and therefore the message contents can be easily changed to add different kind of data.

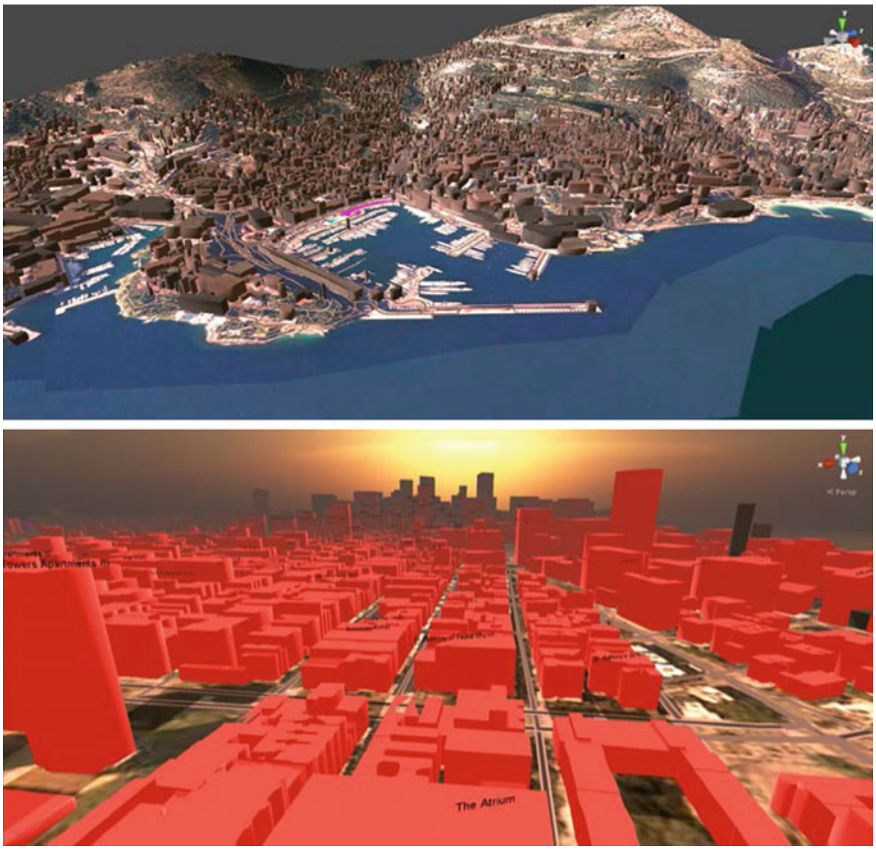


Fig. 6 Geostream procedural generation of Monaco (*above*) and New York (*below*)

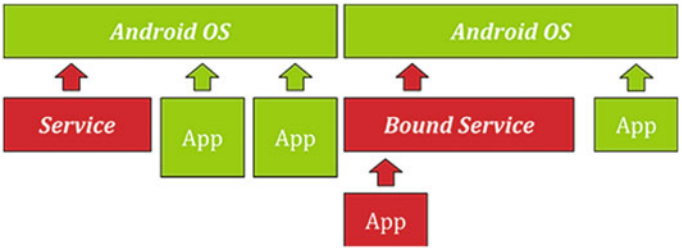


Fig. 7 Standalone mobile application (*left*) and mobile application with included library (*right*)

The basic message template contains two compulsory fields, which are *latitude* and *longitude*. Other optional fields are the current *speed*, the GPS *accuracy*, the message *timestamp* or even the *speed limit* from the current location. A specific instantiation of this interaction is discussed in the next section.

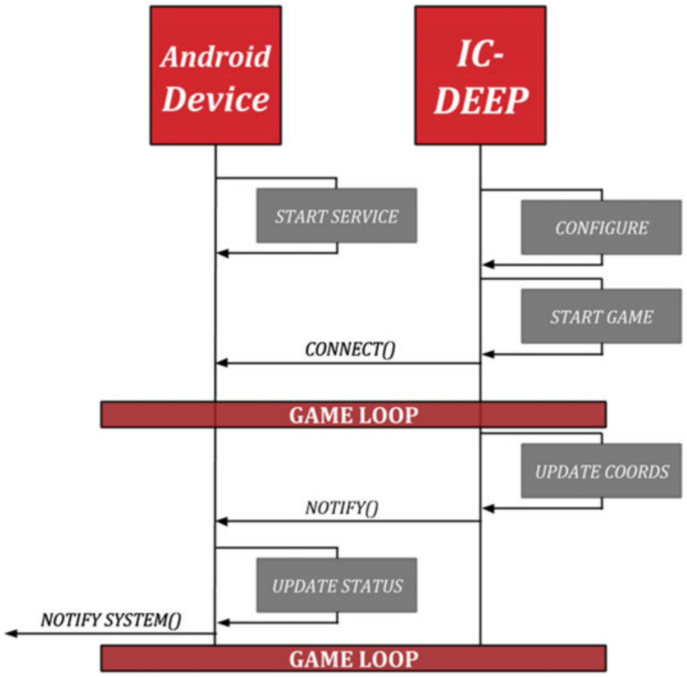


Fig. 8 Typical interaction between IC-DEEP simulator and an ADAS

The coupling of SUMO microscopic simulator with the driving simulators, namely with IC-Deep, uses the same methodology as implemented and described elsewhere [17]. However, this raises some issues regarding the communication channel, as the SUMO TraCI protocol uses sockets and currently does not support more than one active socket. This is obviously a bottleneck when controlling multiple driving agents and a possible SUMO extension to support parallelism in terms of communication is in study.

4 Preliminary Verification

The preliminary verification to assess the proof of concept and also the efficiency of the developed architecture focused on the modules in the dotted area of Fig. 1, the remainder of the system will be developed later on, as mentioned in the next section. We have divided the verification into two independent tests. Both of the tests were performed in the same geographical location, which was downtown Porto, on *Avenida dos Aliados*, as seen in Fig. 9.

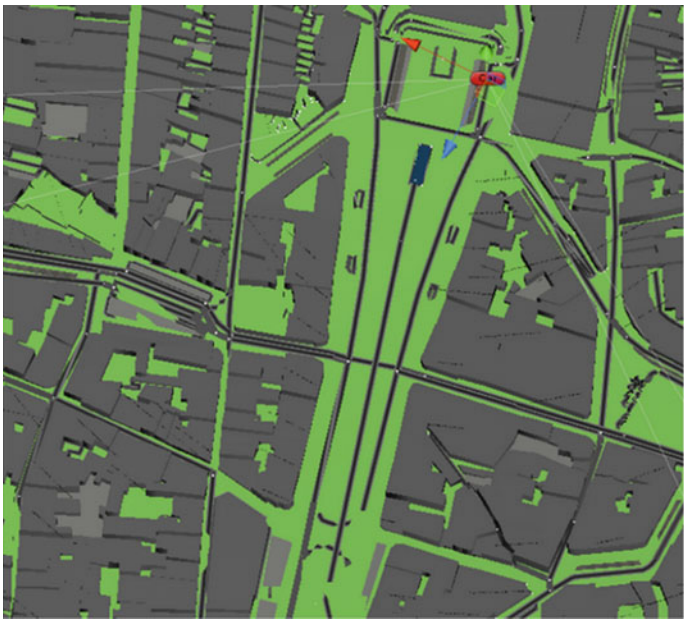


Fig. 9 Generated 3D Scene ortographic view

In the first experiment we test the simulator accuracy to represent real-world scenarios using the Geostream framework. The other test aims to emulate the GPS signal on the mobile device.

4.1 Simulator Accuracy

To test the simulator accuracy we have collected multiple GPS trace logs while driving a real car in the selected geographical location. We have then overlayered a visual representation of the obtained traces on the simulator and on Google Earth, both results can be seen in Fig. 10. The results show that the generated 3D scene is highly representative of the real-world location. In one of the trace logs we have noticed an error and highlighted it in Fig. 10 (see labels 1 and 2), this error happens due to the data being collected as raw, untreated GPS, where the *road matching* algorithm [18] has not been applied. This is also an interesting result, as the error can been seen in both the simulator and Google Earth alike.

Apart from testing the fidelity of the simulation with GPS trace logs, it is also noticeable that the orthographic view of the generated 3D scene very much resembles the satellite image of the same location, as it can be seen in Fig. 11.

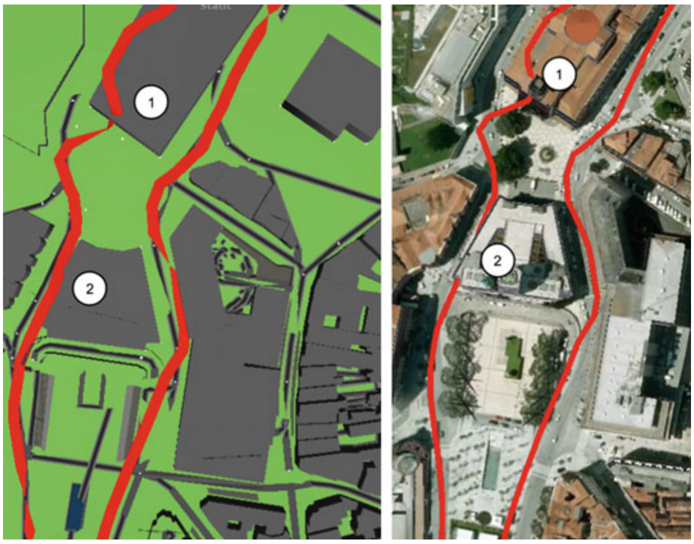


Fig. 10 GPS traces on the simulator (*left*) and Google Earth (*right*)

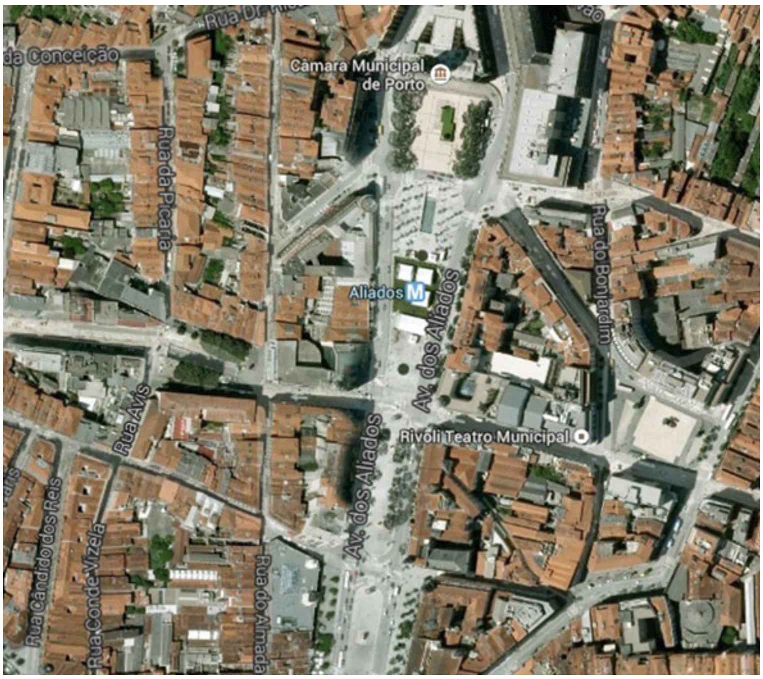


Fig. 11 Satellite image of Av. dos Aliados

4.2 Mobile Device ADAS

To test the communication and emulation in the mobile device the setup consisted of a basic usage scenario, using a simple *ADAS* that shows the user his current and average speed, the total kilometers traveled, and, most importantly, warns him when he exceeds the speed limit of the current location as shown in Fig. 12.

The interaction between the simulator and the mobile application followed that of Fig. 8. To run the simulation the mobile application must be started and the device's IP address, which is shown in the application initial screen, must be entered in the simulator configuration screen. After this the simulation can start and the simulator internally updates the geographical coordinates as the driver traverses the network. These coordinates are passed on to the mobile device as described above, every second and via a JSON formatted message over a TCP-IP socket.

In this particular simulation the information sent to the mobile device consists of the current GPS coordinates and the speed limit of the current location. The heading of the vehicle is calculated internally by the mobile application using a simple algorithm that computes the bearing with the last two known locations and thus, even though this can be done easily, there is no need to pass on an orientation variable from the simulator.

The main goal of this experimentation was to understand whether or not the mobile device simulated GPS position and calculated speed matched those of the simulator. We have used the driving simulator and Google Maps application to compare the marked position while driving. To compare the driving speed we have used the developed ADAS.

Even though the results from the simulator and the mobile device were not recorded, any inaccuracies were not noticed when testing. The only possible minor differences would be due to the fact that the Google Location API on the Android device automatically adjusts the current position to the nearest road, which is, as mentioned above, a technique called *road matching*.

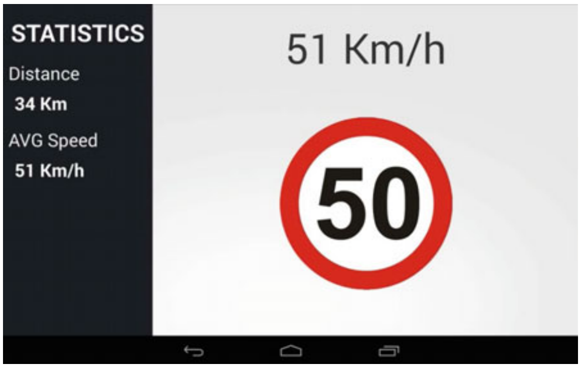


Fig. 12 Developed ADAS

5 Conclusion

In this paper we presented a multi-faceted MAS-based driving simulator methodology. The presented framework can be used to simulate and test multiple aspects in human factors in ITS, generally. Among others we identify some that we consider more expressive of the system's spread, such as supporting a Serious Game [19] to test driving behaviors and ergonomics, simulating driver's idiosyncrasies effects on the ATS with peer-designed agents, and also prototyping and validating Advanced Driver Assistance Systems.

The preliminary verification has illustrated the system efficiency and usability, as well as its ability to accurately represent real-world scenarios without the need of extensive 3D modeling or expensive hardware setups. This ability allows researchers to conduct studies regarding singularities of the different geographical locations.

As a great advantage over other systems we point out the fact that our system is always up to date in terms of real-world mapping, and also that there is no need to waste any time creating a scene when the sole purpose is to test an ADAS or do any other kind of simulation.

In addition to implementing the remaining components of the proposed methodology, there is an ambitious workload of further developments. We would like to point out some that we consider proprietary and more challenging. We believe it would be interesting to support batch simulations, in order to collect significant data and extract more elaborate conclusions. There are also improvements specific to driving simulators that we envisage, such as more detailed scenarios and improved physics.

It would also be interesting to develop cache servers that could store the responses from external services, and thus improve loading times. Another interesting enhancement would be to allow multiple agents to connect to multiple ADAS, simulating distributed ADAS applications while extending SUMO capabilities. There are also refinements to be done in the Geostream framework, especially regarding road generation and also importing models and textures for different buildings.

References

1. Baumann M, Keinath A, Krems JF, Bengler K (2004) Evaluation of in-vehicle HMI using occlusion techniques: experimental results and practical implications. Appl Ergon 35(3):197–205
2. Goncalves J, Rossetti RJF, Olaverri-Monreal C (2012) IC-DEEP: a serious games based application to assess the ergonomics of in-vehicle information systems. In: 2012 15th International IEEE conference on intelligent transportation systems (ITSC), pp 1809–1814
3. Wang F-Y (2003) Integrated intelligent control and management for urban traffic systems. In: Intelligent transportation systems, 2003. Proceedings 2003 IEEE, vol 2, pp 1313–1317

4. Wang F-Y, Tang S (2005) A framework for artificial transportation systems: from computer simulations to computational experiments. In: Intelligent transportation systems, 2005. Proceedings of IEEE, pp 1130–1134
5. Rossetti RJF, Liu R, Tang S (2011) Guest editorial special issue on artificial transportation systems and simulation. IEEE Trans Intell Transp Syst 12(2):309–312
6. Macedo J, Kokkinogenis Z, Soares G, Perrotta D, Rossetti RJF (2013) A HLA-based multi-resolution approach to simulating electric vehicles in simulink and SUMO. In: 2013 16th International IEEE conference on intelligent transportation systems—(ITSC), pp 2367–2372
7. Punzo V, Ciuffo B (2011) Integration of driving and traffic simulation: issues and first solutions. IEEE Trans Intell Transp Syst 12(2):354–363
8. Behrisch M, Bieker L, Erdmann J, Krajzewicz D (2011) Sumo-simulation of urban mobility-an overview. In: The third international conference on advances in system simulation SIMUL 2011, pp 55–60
9. Maia R, Silva M, Araujo R, Nunes U (2011) Electric vehicle simulator for energy consumption studies in electric mobility systems. In: 2011 IEEE forum on integrated and sustainable transportation system (FISTS), pp 227–232
10. Gomes P, Olaverri-Monreal C, Ferreira M, Damas L (2011) Driver-centric VANET simulation. In: Communication technologies for vehicles, Springer, Germany, pp 143–154
11. Kern D, Müller M, Schneegaß S, Wolejko-Wolejszo L, Schmidt A (2008) CARS-Configurable automotive research simulator. In: Mensch and computer workshop band, pp 256–260
12. Miao Q, Zhu F, Lv Y, Cheng C, Chen C, Qiu X (2011) A game-engine-based platform for modeling and computing artificial transportation systems. IEEE Trans Intell Transp Syst 12 (2):343–353
13. Hassan B, Berssenbrugge J, Al Qaisi I, Stocklein J (2013) Reconfigurable driving simulator for testing and training of advanced driver assistance systems. In: 2013 IEEE International symposium on assembly and manufacturing (ISAM), pp 337–339
14. Noth S, Edelbrunner J, Iossifidis I (2012) An integrated architecture for the development and assessment of ADAS. In 2012 15th International IEEE conference on intelligent transportation systems (ITSC), pp 347–354
15. Gruyer D, Pechberti S, Glaser S (2013) Development of full speed range ACC with SiVIC, a virtual platform for ADAS prototyping, test and evaluation. In: 2013 IEEE intelligent vehicles symposium workshops (IV Workshops), pp 93–98
16. Yu S, Lee S-Y, Kim M-S, Lee D-G (2006) Development and evaluation of ITS devices using KAAS(KATECH Advanced Automotive Simulator) system. In: International joint conference SICE-ICASE, 2006, pp 2116–2120
17. Pereira JLF, Rossetti RJF (2012) An integrated architecture for autonomous vehicles simulation. In: Proceedings of the 27th annual ACM symposium on applied computing, pp 286–292
18. El Najjar M, Bonnifait P (2005) A road-matching method for precise vehicle localization using belief theory and kalman filtering. Auton Robots 19(2):173–191
19. Rossetti RJF, Almeida JE, Kokkinogenis Z, Goncalves J (2013) Playing transportation seriously: applications of serious games to artificial transportation systems. Intell Syst IEEE 28 (4):107–112