An Integrated Architecture for Autonomous Vehicles Simulation

José L. F. Pereira, Rosaldo J. F. Rossetti
Artificial Intelligence and Computer Science Laboratory
Department of Informatics Engineering
Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, S/N • 4200-465 Porto • Portugal
T: (+351) 225081566 F: (+351) 225574103
{jlpereira, rossetti}@fe.up.pt

ABSTRACT

Modeling and simulation tools are being increasingly acclaimed in the research field of autonomous vehicles systems, as they provide suitable test beds for the development and evaluation of such complex systems. However, these tools still do not account for some integration capabilities amongst several state-of-the-art Intelligent Transportation Systems, e.g. to study autonomous driving behaviors in human-steered urban traffic scenarios, which are crucial to the Future Urban Transport paradigm.

In this paper we describe the modeling and implementation of an integration architecture of two types of simulators, namely a robotics and a traffic simulator. This integration should enable autonomous vehicles to be deployed in a rather realistic traffic flow as an agent entity (on the traffic simulator), at the same time it simulates all its sensors and actuators (on the robotics counterpart). Also, the statistical tools available in the traffic simulator will allow practitioners to infer what kind of advantages such a novel technology will bring to our everyday's lives. Furthermore, an architecture for the integration of the aforementioned simulators is proposed and implemented in the light of the most desired features of such software environments.

To assess the usefulness of the platform architecture towards the expected realistic simulation facility, a comprehensive system evaluation is performed and critically reviewed, leveraging the feasibility of the integration. Further developments and future perspectives are also suggested.

Keywords

Autonomous Driving, Autonomous Vehicles Simulation, Robotics Simulation, Microscopic Traffic Simulation, Agent-based Simulation

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

1. INTRODUCTION

The autonomous vehicle paradigm has been around in tinkerers minds for some years now, as it envisions to radically change the concept of mobility. With the *technology boom*, researchers are now more focused on the software challenges that such a complex system requires, as hardware itself is becoming more affordable. Also, some authors agree that the autonomous vehicle paradigm is rather a software problem ([4], [11] or [15]). To accomplish such a goal, current traffic modeling and simulation tools need to be adapted having in mind its requirements, as they already account for a multitude of features related to traffic research and engineering, which can be great helpers when applied to the development of driverless vehicles.

A careful analysis throughout the recent Defense Advanced Research Projects Agency (DARPA) challenge competition demonstrates the embrionary state of such tools. Often the teams used robotics simulators with no cooperation capabilities, which implemented simplified models for vehicle and object collision, not offering any kind of realistic behaviour or traffic analysis tools [6, 7, 8, 16].

In this paper we intend to contribute to the autonomous vehicles simulation tools introducing an integrated architecture, seamlessly coupling both a traffic and a robotics simulator. Such a tool would allow practitioners to have in consideration the robotic nature of an autonomous vehicle – which uses a set of sensors/actuators to perceive the real world, making intelligent actions on navigation and planning – at the same time it simulates it within a virtual urban traffic network.

The remainder of the paper is organized as follows. We first discuss on the details of the proposed architecture, emphasizing on integration aspects. The following section is dedicated to presenting the developed prototype, where some technological decisions are made. We illustrate the approach with the implementation of a reactive agent and conclude with suggesting further developments.

2. THE PROPOSED ARCHITECTURE

Depicted in Figure 1, a software architecture for the autonomous vehicle simulation in a traffic environment is proposed. This architecture has a distributed nature, as each simulator must use the maximum possible resources. It consists of four major modules, briefly described below:

Microscopic Traffic Simulator Simulates almost all ve-

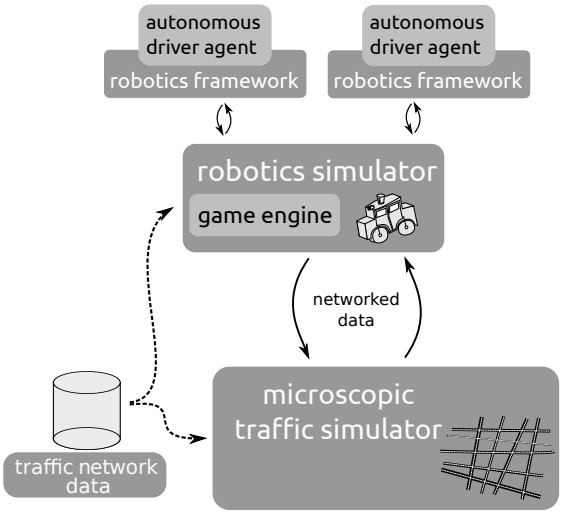


Figure 1: The proposed architecture for autonomous vehicle simulation in a traffic environment

hicles with a resemblance to the macroscopic simulation of real traffic streams. It also maintains all infrastructure systems, such as induction loops or traffic light plans. Given the extensibility of the simulator, a higher level statistical framework may be coupled in order to study traffic behavior patterns of individual and cooperative strategies for autonomous vehicles;

Robotics Simulator Performs the simulation of all autonomous vehicles in the environment, along with all its sensors and actuators, and mirrors every surrounding object. It also features a game engine for immersive 3D animation through both powerful physics and visualization modules;

Coherent Network Data Represents the traffic network topology model, as well as its realistic 3D environment data;

Autonomous vehicle interface and control Manages the brain of the vehicle. Typically an external software driver can be deployed to perform the autonomous vehicle high-level tasks using an agent-based methodology. A Hardware Abstraction Layer (HAL) is underneath for seamless real/virtual world development, and sensor/actuator permutation.

We can acknowledge the bidirectional communication between each simulator, whereas the autonomous vehicle provides its kinematic variables to the traffic simulator, and the traffic simulator calculates the world state and return back its surrounding data. All of these transactions should occur in the same time step.

3. PROTOTYPE DEVELOPMENT

3.1 Simulator Selection

As pointed out earlier, using two types of simulators may be a feasible approach when simulating autonomous vehicles on an urban environment. Such an idea was already introduced in [5], although the selected software architecture and frameworks did not provide satisfactory results. In the former paper, the chosen traffic and robotics simulation software did not have the sufficient maturity level, which led to various implementation difficulties and, consequently, to an inefficient platform. However, its findings have provided some lights on the usefulness of the integration and difficulties to be overcome.

In [9], the authors present a comprehensive review to the simulation of autonomous vehicles, having in mind both robotics and traffic simulators. This review was crucial to the decision of using both SUMO and USARSim traffic and robotics simulator to this project. Although their maturity level is inferior to their commercial counterparts, they have a strong support from the open-source community and would allow for full core access which should be favorable for low level optimizations.

SUMO [14] is a highly portable, microscopic road traffic simulation package designed to handle large road networks and has a strong commitment with the academia and research community. It is mainly developed at Institute of Transportation Systems at the German Aerospace Center and it is written is C++.

USARSim [3] is an open-source high-fidelity robotics simulator based on the Unreal Tournament game engine. Due to Unreal license restrictions, USARSim is written in Unreal Script, the official scripting language which offers interface with the engine core. This particular issue can be critical as there is no direct access to all the features of the engine. However, USARSim's current development state, high quality sensor simulation and physics rendering makes it the best choice to the project. Another major drawback could be its dependency on Windows operating system, a constraint by Unreal Engine 3. However, as there are no other dependencies to the remaining software, all the produced software is multi-platform.

To describe the proposed framework a prototype was developed using SUMO and USARSim. Its methodological approach is many-fold:

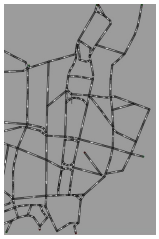
Modify SUMO simulator SUMO is a vehicle- and edge-based microscopic simulator. Therefore, it is not prepared to allow lane-independent vehicles to circulate. Also, a spawning mechanism must be implemented to carefully place and identify an autonomous vehicle in the network. Finally, a communication class must be implemented for it to efficiently communicate with USARSim;

Modify USARSim simulator USARSim is also not fully prepared to accommodate a urban autonomous vehicle. The latest version (based on Unreal Engine 3) does not provide a LIDAR range scanner, and there are no 3D models of a four wheeled conventional vehicle and realistic terrain. Similarly to the previous step, a communication class needs to be implemented;

Integrate simulators Apply the required modifications to integrate both simulators;

Implement a control agent To successfully validate the proposed architecture and prototype in its essence, a simple agent was devised with a simple dashboard showing real-time sensor data from the autonomous vehicle. The control algorithm moves the vehicle through a white line, obeying the car-following rule, and possibly deviating from random objects.

(a) Aliados in OpenStreetMaps



(b) Aliados in netEditor

(c) Aliados in a 3D model

Figure 2: Aliados in several representations.

3.2 Reference Network

The reference network for which the autonomous vehicles were simulated on is part of the Aliados area in Porto, Portugal. Its model has to be represented in two forms, as each one will fit the requirements of the two simulators. Such models are denominated a topological model and a 3D model for SUMO and USARSim respectively.

In the SUMO case, the netEditor [10], a pluginable traffic network editor, was used to import the required network from OpenStreetMaps (OSM), implement an O-D Matrix for starting nodes, and select all allowed directions on each road. Figure 2a represents the OSM Aliados topology, and Figure 2b the imported network from OSM.

Regarding the USARSim simulator, the 3D model from Aliados was gathered using a software designed and developed in [12]. This software uses Procedural Modeling and geographically referenced information to render a realistic scenario, tipically used on computer games. Giving that this work intended to make the simulation framework as realistic as possible, a rapid prototyping tool for 3D scenarios was imperative. Figure 2c depicts Aliados 3D model rendered in the Autodesk 3ds Max modeling tool [1].

3.3 Surrounding Vehicles Integration

We are ready to manage all surrounding vehicle information from SUMO to USARSim. SUMO was accordingly modified to calculate the surrounding vehicles around an autonomous vehicle and therefore, an algorithm to handle such vehicles was devised. It uses a hash-table comprehending vehicles already created on the scene, and calculates their next movements. Its pseudo-code is represented by Algorithm 1. *AbsMove(veh)*, *Create(veh)* and *Kill(veh)* corresponds to the *WorldController* USARSim command definitions [2].

3.4 Network Coherence

Having successfully imported the Aliados road network into SUMO using the netEditor and to USARSim using the procedural modeling application, we implemented a network calibration method between each simulator scenario model.

As both networks for the two simulators rely on different sources, albeit they both resemble the same physical zone of Aliados, in Porto city, their coordinate references are not the same. To overcome this, an Euclidean transformation method was used to map a point from one coordinate system to the other. Also, to calculate the transformation parameters (A matrix and b_1, b_2), the practitioner would only need to know the location of two points in both simulators

Algorithm 1 Surrounding vehicles control algorithm

```
surrVehicles = myVehicle.getSurrVehicles()
for curVeh = surrVehicles.first() → surrVehicles.last() do
  if vehHashMap.exists(curVeh) then
    AbsMove(curVeh)
  else
    Create(curVeh)
    AbsMove(curVeh)
    vehHashMap.insert(curVeh)
  end if
  curVeh ← curVeh + 1
end for
for curVeh = lastSurroundingVehicles.first() → lastSurroundingVehicles.last() do
  if !vehHashMap.exists(curVeh) then
    Kill(curVeh)
  end if
  curVeh ← curVeh + 1
end for
lastSurroundingVehicles ← surrVehicles
```

coordinate system.

We must note that despite the 3D nature of the USARSim simulator, as the 3D model contains plain roads, the z value will always remain constant, therefore this transformation is $T : \mathbb{R}^2 \to \mathbb{R}^2$.

This coordinate transformations were implemented on SUMO simulator given its C++ native implementation and a standard library containing the trigonometric functions.

Having the coordinate mapping between the two simulators ready, the vehicles from SUMO were correctly translated to USARSim and the autonomous vehicle was also shown on SUMO. However, this improvements have shown a network disparity between the two models, as the road lengths and positions from each other were not exactly the same thus mirroring the vehicles in a slight different position from where they should have been drawn. To address this issue, the Aliados SUMO network description was edited in netEditor to approximate the two network positions, however the lack of a proper tool that could render both networks at the same time prevented us to carry out this task on the most approximate way.

3.5 Multi-agent communication

The USARSim platform implements a *Wireless Communications Server* to act as a middle man between robots, simulating message and connection dropping when its distance is not realistically feasible which can be used for Multi-Agent Systems based coordination methodologies. In the following section an example implementation of a reactive agent to control an autonomous vehicle in USARSim is detailed, as a mean to introduce a practitioner to the development of agent-based vehicle control in the proposed platform. However, for the sake of time this agent does not feature any type of communication abilities with other vehicles. Refer to the USARSim documentation for further elucidation on the *Wireless Communications Server* protocol [2].

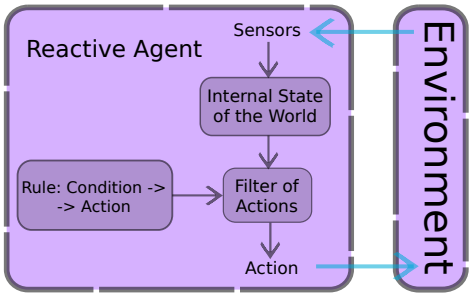


Figure 3: Simple reactive agent diagram (adapted from [13]).

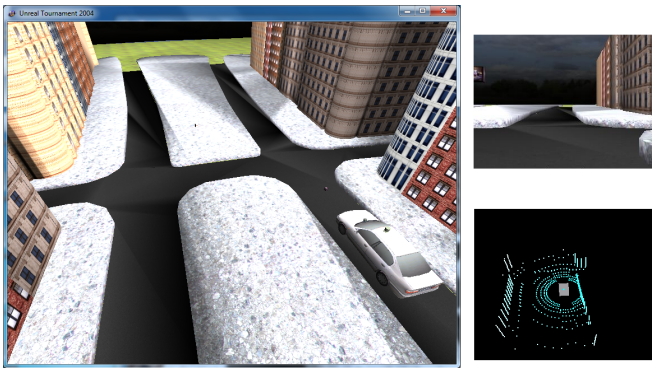


Figure 4: Optical Camera and LIDAR sensor visualization interface on a preliminary simulation in USARSim.

4. SIMPLE REACTIVE AGENT

To validate the proposed framework and its implementation as already discussed earlier, a simple autonomous agent was developed to spawn a vehicle in the simulation, receive its sensors information and calculate a trajectory based on it. It follows a reactive agent methodology, with the model as depicted in Figure 3.

This application was coded in C++ with the Qt4 framework from Nokia, which provides cross-platform Graphical User Interface (GUI) and networking capabilities.

4.1 Agent Architecture

The simple reactive agent software architecture is depicted in Figure 5. It consists of a central class *controller* which sets up the communication *QTcpSocket* with USARSim and instantiates an *usarsim_sensor* for each sensor in the vehicle. It uses an hash-table that maps each sensor id to its correspondent object, while redirecting the received sensor message for parsing. Each sensor also inherits a *QWidget* class, containing sensor information in a graphical format dependent of the sensor type. For example, a LIDAR (*rangeScanner3D*) sensor provides a 3D point cloud visualization in OpenGL as illustrated in Figure 4.

The reactive agent rules set is represented by the *rule_set* class, to which the *controller* feeds with received sensor data, and retrieves the proper driving speed an steer angle to apply onto the robot vehicle. In this reactive agent example, the camera is used to process the lane position and tries to maintain its aim at it, whereas the LIDAR sensor prevents the vehicle to collide with close objects. Figure 6 depicts

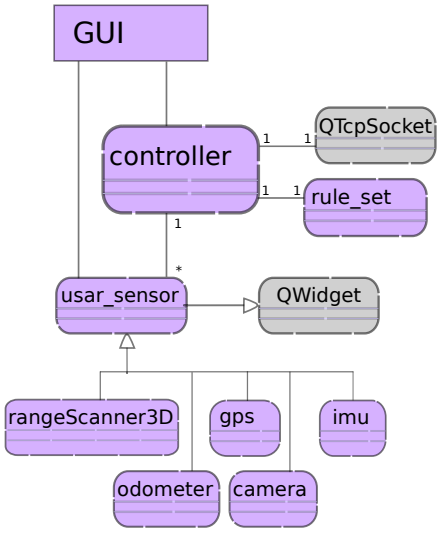


Figure 5: Simple reactive agent architecture.

a sequence diagram with the most important transactions between the simulators and the reactive agent.

4.2 Navigation

This section provides more detailed description on the *Drive* and *Steer* commands sent to USARSim, using LIDAR and camera sensors respectively, as an example of a simple autonomous control for an urban vehicle. The reader must note that these methods are not validated to be used on robust autonomous vehicle control, but only to model a very simple navigation algorithm for the sake of demonstration.

To drive the vehicle, only the front side point data from the LIDAR sensor is considered to make it aware of its front vehicles. Furthermore, we select a square window with horizontal and vertical angles (typically 5 to 10 degrees), and apply the mean to it. Afterwards, the *Drive* speed is calculated using the following formula (Equation 1):

$$drive\_speed = max\_speed \cdot (1 - e^{-m/k_1}) \cdot (1 - e^{-\alpha/k_2}) \quad (1)$$

The expressions in parenthesis attenuate the speed depending on the measured mean m to make the vehicle slow down in other vehicles' presence, and on the steering angle α, to prevent it to steer in high speeds.

The vision-based method to support vehicle steer decisions is processed in the following steps:

Image binarization with a predefined threshold to filter lanes only (in white);

Calculate white pixel density of both left (ρ_{left}) and right (ρ_{right}) side of the image;

Implement a proportional control with the density difference, given by Equation 2.

$$\alpha = K \cdot (\rho_{right} - \rho_{left}) \quad (2)$$

Therefore if the left pixel density is higher, the steering angle α will be negative and the vehicle will move towards

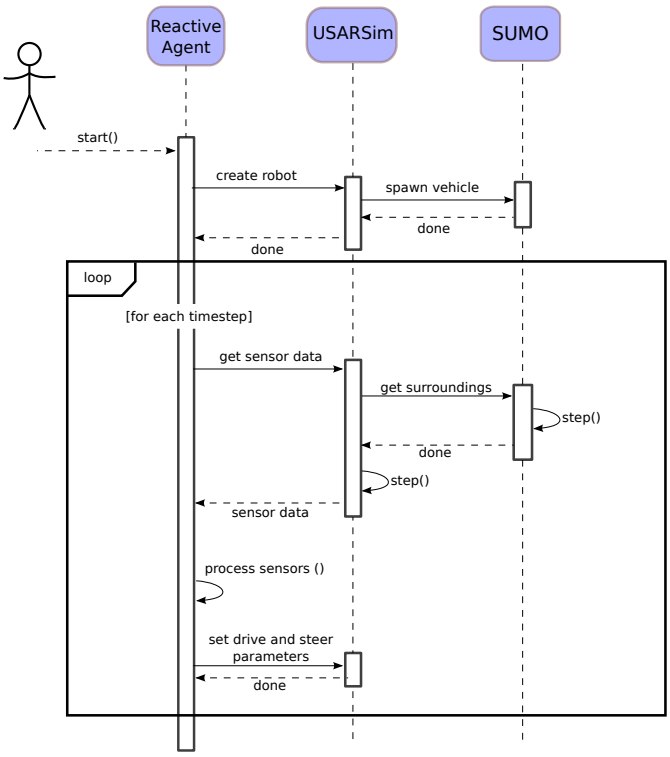


Figure 6: Sequence diagram stating the interaction between simulators and the reactive agent.

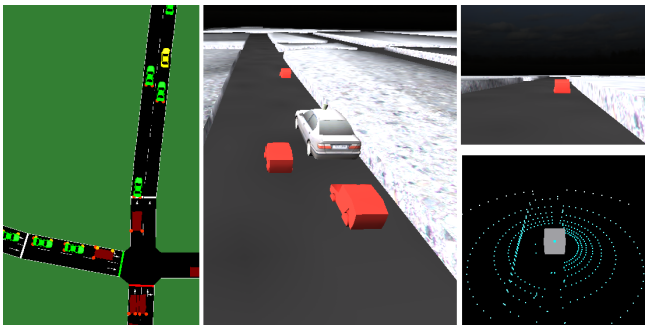


Figure 7: Framework synchronization aspect. On the first row, SUMO view is presented (autonomous vehicle yellow painted) followed by USARSim's. The camera and LIDAR sensors from the reflective agent are depicted in last.

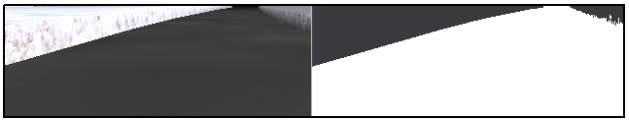


Figure 8: The binarization process of the vehicle front camera image.

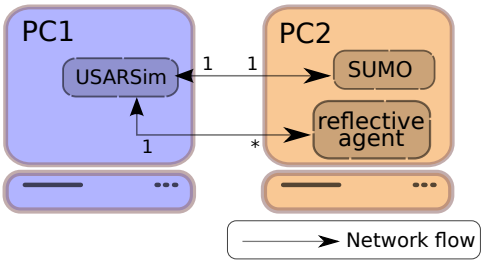


Figure 9: Computer-application arrangement to perform evaluation tests.

left. If the right pixel density is higher, α will be positive and the vehicle will move towards right.

From Figure 7, we can state the successfulness of the test. On the first row we can visualize the autonomous vehicle as the yellow car, which is surrounded by three other cars; one at front, one at the back and another on the left side. From the USARSim view we can see the same environment in a 3D model which confirms the synchronization of the two simulators. A careful look at the network topology yields a slight offset amongst the vehicles position in each simulator, as a consequence of the calibration issues already discussed in Section 3.4. The last row on the figure depicts the camera image, which is seeing the front vehicle on the top, whereas a LIDAR point cloud data clearly detecting the road is depicted below. Given the limited resolution of the range sensor it is not evident the presence of a car just using its data.

5. RESULTS AND DISCUSSION

To properly validate the integration architecture and implementation devised for the project, several metrics were analyzed to measure its effectiveness and therefore give a more consistent critic about each part of the system. This way, we can assess the selected choices of the traffic and the robotics simulators, SUMO and USARSim respectively, which seemed the most appropriate to be used in this project.

Table 1: Two computer set-ups used to evaluate the integrated platform.

PC	PC1	PC2
CPU	Intel C2D E8500	Intel C2D E6550
Clock speed	3.16GHz	2.33GHz
RAM size	3.71GB	3.00GB
Graphics Card	Intel Q45 Xpress	Nvidia GF 8600GT
Operating System	Windows 7	Ubuntu 11.04

This evaluation tests were performed on two relatively recent desktop computers available in our department denominated PC1 and PC2. Table 1 illustrates the computers specifications, which should be considered in the critical assessment of the result values. Also, Figure 9 shows which application was executed on which computer.

To evaluate the platform performance to be eventually scrutinized against other frameworks, the CPU usage of both simulators were assessed as well as the network bandwidth in several test runs.

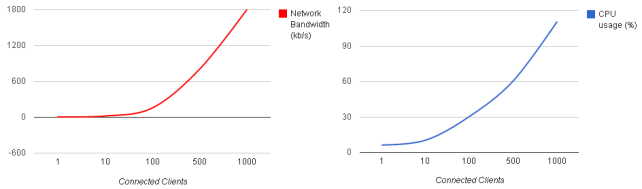


Figure 10: Performance Test Run 1 - SUMO simulator performance with several connected clients.

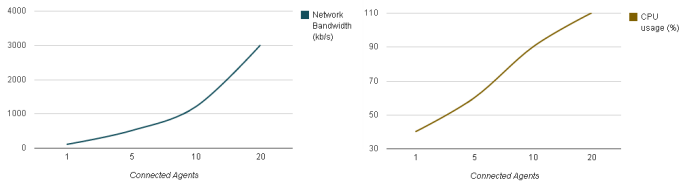


Figure 11: Performance Test Run 2 - USARSim simulator performance with several reactive agents.

These test runs use the bundled operating system profiling tools to measure the aforementioned metrics. Thus, on Windows 7 (the PC1 machine) the metrics are accessed using the "Performance" tool on Administrative Tools > Performance. In Ubuntu Linux, the *top* and *iftop* commands provided the CPU and network usage respectively.

The first approach was to simulate a SUMO network connecting several autonomous vehicle entities and evaluating the CPU usage and network bandwidth used by them. As one of the main goals of the framework is to retain a real-time refresh-rate, i.e. a minimum 30Hz, when the network latency is superior to 33ms the test-run is stopped. The steps to be taken into account in this SUMO performance test are:

1. Start SUMO loaded with Aliados network in PC2;
2. Start the number of instances of the client sample as required in PC2;
3. Wait until the simulation reaches 10s simulation time;
4. Record both CPU usage and network bandwidth on PC2.

Figure 10 depicts the achieved results from the test run. It really looks promising as we can simulate over 1000 autonomous vehicles without significant loss. However, we have to note that when the simulation grows with more vehicles, the consequent number of vehicles surrounding the autonomous vehicles will increase, and so will the network flow and CPU usage.

The next test run depicts the CPU usage and network bandwidth of USARSim connected from one to several autonomous vehicle agents, with all the sensors from the implemented autonomous vehicle. Therefore, the steps to reproduce the test run are:

1. Start USARSim loaded with Aliados network in PC1;
2. Start the number of instances of the reflective agent as required in PC2;
3. Wait until the simulation reaches 10s simulation time;
4. Record both CPU usage and network bandwidth on PC1.

Figure 11 plots the results acquired during this second test run. Despite being run on PC1, the fastest machine, USARSim consumes a lot of CPU power given its realistical simulation complexity. Furthermore, the lack of a dedicated Physics Processing Unit (PPU) leaves all physics calculations to the CPU. The required network bandwidth is also significantly higher than in its SUMO counterpart, as sensor data is quite large (5 sensors) and the network protocol is not serialized thus wasting higher bandwidth (typically it needs 50 to 100 bytes per each message whereas serialized buffer only require 5 to 20 or even less).

From the practical results we can infer that the required network bandwidth for the framework application can represent a bottleneck if certain conditions are met. Therefore, researchers who intend to adapt this architecture are encouraged to comprehensively study the message passing efficacy when selecting the simulation tools.

6. CONCLUSIONS

This paper has presented the methodological and technical challenges of a practical solution for the coupling of two simulators, namely a traffic simulator and a robotics simulator to increase the reliability of simulation with several self-driving vehicles, seamlessly integrated within a common traffic environment. An integration architecture was devised pinpointing the main aspects towards an efficient communication among simulators, which have been addressed to some extend. This flexible approach was not bound to any specific simulation software architecture thus opening up new opportunities to the development of other platforms. Also, a prototype to demonstrate the integration was also devised and effectively implemented using both SUMO and USARSim, which were accordingly patched to meet the integration requirements. To couple these two simulators, a method for the management of seamless vehicle mirroring was implemented using efficient data types and networking. The robotics simulator provided its position to the traffic one, whereas the latter calculates all microsimulation vehicles, feeding back their position to the robotics simulator in the same time step.

The effectiveness of the framework has culminated in the need for modeling and developing the reactive agent, which comprehends a simple navigation and control algorithm as a means to exemplify and measure the framework's promising potentiality and efficiency.

Giving the complexity of the simulation frameworks used in this project, the next step towards its solidification would be to fix some occasional bugs that might still persist, particularly on SUMO as it was subjected to several modifications and the simulation of real complex traffic networks.

Ultimately, some coordinate mapping techniques could be assessed to study the possibility of using three dimensional topologies of traffic roads in the robotic simulator side, while using the same plain roads in the traffic simulator. The network model calibration method should be evaluated as the result of importing from two different data sources.

Another long-term goal would be to foster the integration of this platform with even one more simulator, namely a pedestrian simulator, whereby it can push further the real

implementation of an even more realistic Artificial Transportation System.

As we are all aware, the advent of autonomous vehicles will boost the need for modern R&D tools for such complex systems to be evaluated. Furthermore, there is a recognized potential on this proposed framework, as it should be able to approximate the modern robotics research methodologies to Future Urban Transport Systems.

7. REFERENCES

[1] Autodesk autocad 2011. `http://www.autodesk.com/`, 2011.

[2] Usarsim reference documentation. `http://iweb.dl.sourceforge.net/sourceforge/usarsim/usarsim-manual_3.1.3.pdf`, 2011.

[3] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. USARSim: a robot simulator for research and education. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1400–1405, Apr. 2007.

[4] E. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen. The seeing passenger car 'vamors-p'. In *Intelligent Vehicles '94 Symposium, Proceedings of the*, pages 68 – 73, oct. 1994.

[5] M. C. Figueiredo. An Approach to Simulation of Autonomous Vehicles in Intense Traffic Scenarios. *MSc. Thesis*, (June), 2009.

[6] G. Franken. DARPA Urban Challenge Princeton University Technical Paper. *Access*, 2005.

[7] T. Leader, C. Reinholtz, R. Hall, M. Code, V. Tech, T. Alberi, J. Farmer, S. Frash, C. Terwelp, and A. Wicks. DARPA Urban Challenge Technical Paper. *Defense*, 2007.

[8] U. Ozguner, K. Redmill, S. Biddlestone, A. Yazici, and C. Toth. Simulation and testing environments for the DARPA Urban Challenge. *2008 IEEE International Conference on Vehicular Electronics and Safety*, pages 222–226, Sept. 2008.

[9] J. L. F. Pereira and R. J. F. Rossetti. Autonomous vehicles simulation: A comprehensive review. In *Proceedings of the 25th European Simulation and Modelling Conference*, ESM'2011, October 2011.

[10] J. L. F. Pereira, R. J. F. Rossetti, and E. Oliveira. Towards a cooperative traffic network editor. In Y. Luo, editor, *Cooperative Design, Visualization, and Engineering*, volume 5738 of *Lecture Notes in Computer Science*, pages 236–239. Springer Berlin / Heidelberg, 2009.

[11] D. Pomerleau and T. Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert*, 11(2):19 –27, apr 1996.

[12] R. Rodrigues, A. Coelho, and L. Reis. Data model for procedural modelling from textual descriptions. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, july 2010.

[13] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[14] C. Sommer, Z. Yao, R. German, and F. Dressler. Simulating the influence of ivc on road traffic using bidirectionally coupled simulators. In *INFOCOM Workshops 2008, IEEE*, pages 1 –6, april 2008.

[15] B. Ulmer. VITA-an autonomous road vehicle (ARV) for collision avoidance in traffic. In *Intelligent Vehicles' 92 Symposium., Proceedings of the*, pages 36–41. IEEE, 1992.

[16] M. Xie, H. Chen, X. F. Zhang, X. Guo, and Z. P. Yu. Development of Navigation System for Autonomous Vehicle to Meet the DARPA Urban Grand Challenge. *2007 IEEE Intelligent Transportation Systems Conference*, pages 767–772, Sept. 2007.