Submicroscopic and Physics Simulation of Autonomous and Intelligent Vehicles in Virtual Reality

Olivier LAMOTTE, Stéphane GALLAND
Multiagent Simulation in Virtual Cities
System and Transportation Laboratory
University of Technology of Belfort-Montbéliard
90010 Belfort cedex, France
Email: {olivier.lamotte,stephane.galland}@utbm.fr

Jean-Michel CONTET, Franck GECHTER
Multiagent Systems for Intelligent Vehicle
System and Transportation Laboratory
University of Technology of Belfort-Montbéliard
90010 Belfort cedex, France
Email: {jean-michel.contet,franck.gechter}@utbm.fr

***Abstract*—Simulation, which creates abstractions of the system is an appropriate approach for studying complex systems that are inaccessible through direct observation and measurement. Many simulators aimed at studying vehicles dynamics are existing. Most of them are focusing on mechanical simulation of the vehicle with a special focus on tyre/road contact. The main drawback of these is the requirement of real vehicle (this can be a simple prototype) to build a dynamical model. Another drawback is the difficulty to integrate virtual sensors and onboard artificial intelligence abilities. The aim of Virtual Intelligent Vehicle Urban Simulator is thus to overcome the general drawbacks of classical solutions by providing the possibility of designing vehicle virtual prototype with well simulated embedded sensors. This paper presents the global architecture of the simulator and draws some comparisons of simulations with real experiments.**

***Keywords*-Multiagent-Based Simulation; Physics Simulation; Autonomous Vehicle.**

I. INTRODUCTION

Many simulators aimed at studying vehicles dynamics are existing. Among the most popular, we can cite Callas and Prosper [**?**] and widely used in automotive industry. Most of them are focusing on mechanical simulation of the vehicle with a special focus on tyre/road contact, also known as submicroscopic simulation of vehicle [**?**]. The main drawback of these is the requirement of real vehicle (this can be a simple prototype) to build a dynamical model. Generally, the model is built by testing the real vehicle with embedded sensors, generally accelerometers, in a set a possible dynamical conditions (wet road, snow, curves, etc.). Another drawback is the difficulty to integrate virtual sensors and onboard artificial intelligence abilities. Taking virtual camera as example, they are generally reduced to a simple pinhole model without distortion simulation for instance. In this context, System and Transportation Laboratory decided to develop a simulation/prototyping tool, named Virtual Intelligent Vehicle Urban Simulator (VIVUS), aimed at simulating vehicles and sensors, taking into account their physical properties and prototyping artificial intelligence algorithms such as platoon solutions [1] or obstacle avoidance devices [2]. The aim of VIVUS is thus to overcome the general drawbacks of classical solutions by providing the possibility of designing vehicle virtual prototype with well simulated embedded sensors. The main interests of this solution are:

- Prototyping artificial intelligence algorithms before vehicle first prototype construction. In this case, artificial intelligence systems can be developed, tested and tuned with a virtual prototype of real vehicle.
- Testing critical and/or banned use cases, i.e., cases that implies partial or total destruction of vehicles, to draw the limits of the retained solutions.
- Testing and comparing several algorithms/solutions for embedded functionalities with a low development cost. It can thus help to choose future embedded devices related to retained solutions requirements (processing power needs, connectivity, etc.).
- Testing and comparing sensors solutions before integrating them into the vehicle.
- Integrating tests and evaluations results into vehicle design process.
- Using informed and documented virtual reality to access attribute and state values of the vehicle, its components, and environmental objects around.

The participation of the System and Transportation Laboratory to CRISTAL project [**?**] allowed to test the developed tools under specific constraints.

During the first part of this project, tests were made in simulation in order to compare platoon solutions under several criterium (inter-vehicle distance measurement, stability proofs, lateral error between two following vehicles, sensors limitations, etc.). For this step, the vehicle model used was based on laboratory prototype models that don't have the same physical properties as final vehicles. During the second phase, it is planned to go into further tests including vehicle final models (physical model, energetic model, retained sensors models, etc.). This paper presents the developed simulator architecture and draws results obtain during the first phase of the project. Theses results are then compared with real experiments performed with laboratory prototype electrical vehicles.
The paper is structured as follow: Section 2 explains VIVUS simulator architecture, and details vehicle and sensors models. Then, Section 3 draws some experimental simulated results compared with experiments with real vehicles. Finally, Section 4 concludes the paper by giving

some future works.

II. Simulation of Virtual Vehicle

One of the major goals of VIVUS is the validation in a virtual universe of automatic control algorithms of simulated vehicles. This set of algorithms includes platooning algorithms. Consequently it is essential to recreate an environment close to vehicles and be able to simulate the vehicles themselves, their sensors, and the events happening in vehicle's environments. VIVUS may be run under real time constraints. Indeed because algorithms embedded in virtual vehicles and real vehicles are the same, the virtual simulator may provide performances as close as possible to reality.

Simulation of virtual vehicles is, on one hand, to simulate the physical behaviors of the vehicles. In the other hand, simulation may compute perception data for vehicle sensors and may control the vehicle board. Physical simulation is handled by a physics simulation software, which allows to apply forces to the different vehicle components (dampers, motors, feign, etc.). All these components are 3D objects on, which physic-based relationship are apply.

The rest of this section focuses on the models of VIVUS, the virtual autonomous vehicle, and sensors.

A. Simulator Architecture

As previous expressed, VIVUS is a 3D-based simulator, which supports physics simulation and realistic 3D rendering. According to game and serious game literature, virtual simulators are mainly composed by the following modules and related data structures: physics simulator, artificial intelligence simulator, and 3D rendering engine. Unfortunately theses modules cannot use the same data structures for efficiency concerns [3]. VIVUS simulator model follows this way and has one module for the physics engine, one for 3D rendering and one for control algorithms. Figure 1 illustrates the overall architecture.

Control algorithms are external software, which retreive sensor information from VIVUS and send back control orders. These orders are received and applied by the physics engine. Platooning algorithms [1] are successfully applied conjointly with VIVUS platform.

Physics 3D Model is based on the PhysX engine. This engine is one of the best considering the accuracy and the realistic behavior obtained. PhysX is an engine simulating physical laws in 3D. The physics model is defined by $\langle E, L\rangle$ where E and L are respectively the sets of simulated objects and physics laws. Objects supported by the physics simulation engine are defined by $\langle G, p, o, m, S_l, S_a, A_l, A_a\rangle$; m is the mass of the object; S_l (resp. S_a) and A_l (resp. A_a) are the current linear (resp. angular) velocity and acceleration of the object. p and o are the current position and orientation of the object. Finally G is the geometrical shape associated to the simulated object (basically a box, a cylinder or a sphere). Equation 1 is the transformation applied by the physics simulator at each simulation step t to obtain the model state at step $t+1$. Let δ_t and f be respectively the current state of the simulated

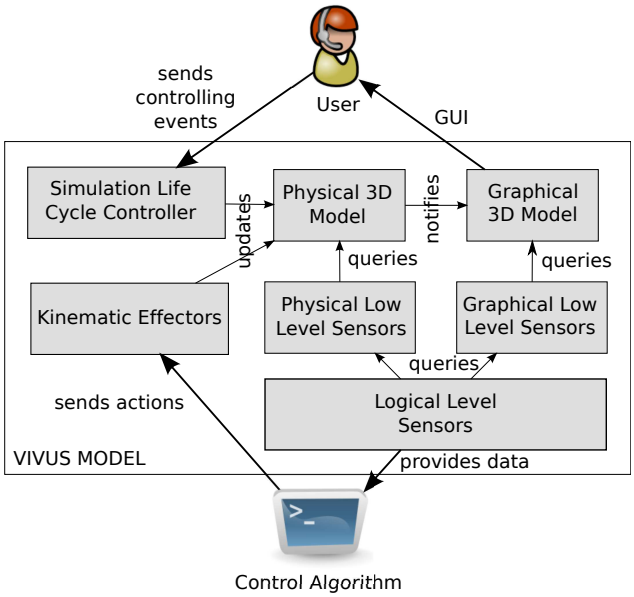

Figure 1. VIVUS global architecture

world and the function that is mapping a simulated object to a motion request. Operator $\prod_\omega$ permits to detect and solve conflicts between motion requests from all ω objects, according to physics laws. Operator $\oplus$ computes a new world state from a given one and a set of motion actions.

$$\begin{aligned} \langle E, L\rangle &\mapsto \quad \langle E, L\rangle \times E \mapsto \mathbb{R}^3 \\ \delta_{t+1} &\to \quad \delta_t, f \\ &\to \quad \delta_t \bigoplus \prod_{e \in \mathrm{dom}\, f} f(e) \end{aligned} \tag{1}$$

During each simulation step, the physical 3D model notifies the graphical 3D model about each change. This last model is applying the newly received position and orientation on the graphical representations of the physical objects. According to our model separation assumption, the graphical model data structure is based on classical 3D scenegraph.

Figure 2 illustrates the sequence of actions run during one simulation step. Kernel has role to schedule all the components of VIVUS. Kernel runs control algorithms, registers influences given by these algorithms and runs PhysX, in order. Control Algorithm may obtain information from the simulated objects. In this way it senses the world model via a set of high level — logical — sensors. In the given sequence diagram example, only graphical sensors are represented. Details on the differences between high level and low level sensors are explained in Section II-C. From the sensed data, control algorithm is able to decide an action, which is given to the Kernel. The control algorithm stage is repeated for each available algorithm. Then Kernel is able to launch the physics simulation, which will retreive algorithm's influences, solve them and apply resulting reactions [3][4]. For each moved object inside the physics model, a notification is sent to the 3D rendering engine to update its internal data structures.

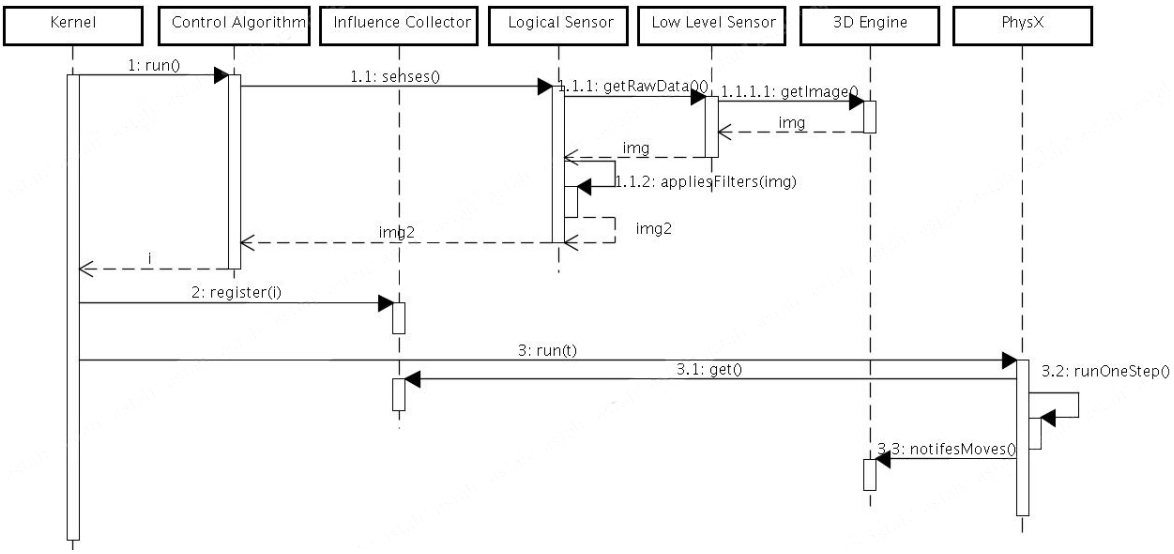

Figure 2. Sequence Diagram for one Simulation Step

B. Physics Model of Vehicles

In order to obtain simulation results as near as possible from the reality, a complete physical model of vehicles has been made. This section presents the dynamical model of SeT laboratory vehicle platform RobuCab4 illustrated by Figure 3.

Figure 3. Systems and Transportations Laboratory vehicles : RobuCAB4 (top), GEM SET CAR (bottom)

This model has been designed to suit PHYSX engine requirements. Models designed for PHYSX are based on composition of PHYSX elementary objects. New elementary components can also be defined using PHYSX requirements. Vehicle dynamical modelisation is a common task in simulation. The RobuCAB vehicle is then considered as a rectangular chassis with 4 engine/wheel components.

This choice can be considered to be realist, the chassis being made as a rectangular un-deformable shape. As for the engine/wheel components, RobuCAB4 platform owns 4 wheel drive each of them being directly linked to one electrical engine. The following parts describe all the parameters determined and/or computed for the vehicle.

1) Chassis model: Chassis is modeled as a rectangular shape with a size denoted C, a mass denoted M_c and a gravity center G_c. Considering vehicle design, the following propositions can be exposed:

$$C = \begin{pmatrix} L_c \\ l_c \\ h_c \end{pmatrix} \quad G_c = \begin{pmatrix} 0 \\ 0 \\ h_{zs} \end{pmatrix} \tag{2}$$

Gravity center position G_c of the chassis has been computed taking into account gravity center of the body, gravity center of each component of the chassis, i.e., battery, embedded electronic card and components, etc. Wheel/engine components are not included in this computation.

2) Tyre and shock absorber models: Wheels are modeled as dynamical objects. Each wheel is considered to have diameter R and a mass M_{wheel}. Dynamical object being defined, they can be set at specific position.

Tyre grip computation function takes tyre sliding as an input. Lateral and longitudinal tyre sliding are computed separately. The output of this function is the tyre grip. This value can then be interpreted depending on the tyre model used.

PHYSX tyre model computes tyre friction constraints from a Hermit spline.

Model parameters have been defined from standard data for 130/70-10 Michelin tyre. Then extremum A, asymptote B and rigidity coefficient, P_x for longitudinal and P_y for lateral, have been defined thanks to several experiments made on the real vehicle.

Shock absorber model is defined by PHYSX with several parameters such as damping constant A_a, stiffness A_r

and free length A_l. All these constants were defined from real vehicle experiments.

3) Engine model: Engine model proposed by PHYSX corresponds to a standard engine with a starting torque C_d and a braking torque C_f. Real vehicle engines are electrical engines with permanent magnet allowing it to make both acceleration and braking. After experiments with a standalone engine, C_d and C_f values have been determined.

4) Summary of the vehicle physical model:

- $L_c = 1.95\ m$ chassis length
- $l_c = 1.195\ m$ chassis width
- $h_c = 2.3\ m$ chassis height
- $M_c = 350\ kg$ suspending mass
- $h_{zs} = 52\ cm$ gravity center height
- $R = 42\ cm$ wheel diameter
- $M_{wheel} = 7\ kg$ wheel mass
- $L = 1.2\ m$ semi-length
- $e = 1.1\ m$ semi-width
- $h_{center\ wheel} = 19\ cm$ height of wheel axis
- $A_r = 1000$ stiffness constant
- $A_a = 330$ damping constant
- $A_l = 12\ cm$ shock absorber free length
- $C_d = 500\ MKG$ starting torque
- $C_f = 500\ MKG$ braking torque
- $A = (1.0; 0.02)$ Extremum coordinates
- $B = (2.0; 0.01)$ Asymptote coordinates
- $P_x = 15$ tyre longitudinal sliding rigidity
- $P_y = 15$ tyre lateral sliding rigidity

C. Sensor Architecture

Real vehicles are equipped with a set of sensors for immediate environment perception: laser rangefinders, sonars, GPS, monoscopic and stereoscopic video cameras. One of the first steps to design VIVUS is to identify the different types of sensors to simulate. They are classified into different categories according to the type of data they produce, an:

- *image sensor* produces a bitmap;
- *video sensor* produces a sequence of bitmaps;
- *ray-based sensor* provides collisions on a predefined set of rays;
- *communication sensor* provides information from an external source (WiFi network, etc.);
- *location sensor* gives the vehicle position and orientation; and
- *state sensor* provides the state of the vehicle or one of its components.

Sensor algorithms are directly used by control algorithms. Because control algorithms may be the same in virtual vehicle and real vehicle, sensor algorithms must provide the same output data as the real sensors: same data format, frequency and data quality. VIVUS does not reproduce the internal physics of sensors. Physical and accurate simulation of such sensors is already supported by commercial offers (CIVIC, etc.). Sensor algorithms implement simple models, which have the same output properties as the real sensors. This choice of simplicity is due to real-time constraint required by the rest of the VIVUS simulator.

To meet the different types of sensors, the simulation environment has been divided into 3 levels: physical low level, graphical low level, and logical level. This decomposition is the basis of the overall architecture implementation for the design of the tool.

Physical low level sensors depend on the physical attributes of simulated vehicles (shock, acceleration, braking, etc.) on one hand. In the other hand they also depend on calculations based on the 3D geometry of the environment. Given laser rangefinders for example, low level sensor computes intersections between the 3D objects in the universe and line segments, which are the representing the lasers. The graphical low level sensors use graphic images from the 3D projections computed from significant points of view with specific resolution and frequency. For a stereoscopic camera, the sensor generates two bitmaps from the points of view of the two cameras. They are generated by the 3D graphical engine. But, for performance concern, they are not displayed but directly sent to high level sensors. Finally, logical high level sensor take informations from one or more low level sensors, format them, and apply filters and modifiers. For example, GPS sensor takes global position and orientation of the vehicle in PHYSX object, and translate these 3D coordinates into longitude-latitude pairs.

With this general architecture, we have considerable freedom as to the accuracy of data from sensors and transmitted to the control algorithms. Indeed, virtual sensors are composed of two parts: the low level sensor to collect data from the virtual universe, and the high level sensor to organize data from the lower level so that they meet the exact specifications of the real sensor. It is easy, in this second part, to integrate noise on data to see even these disruptions (real time or after a pre-established scenario). It is almost important that the simulator may enable specific circumstances, failure of a sensor, for example, or a loss of GPS accuracy. Moreover, sensor simulation in virtual universe is often criticized for its too high level of perfection. Indeed, let us consider the example of virtual cameras, 3D rendering engine is able to produce very precise bitmaps without visual default. But real cameras are not perfect at all. VIVUS is thus able to reproduce these defaults. Once again, organization of sensors allows to apply different filters on sensor informations to match them with whose from real sensors.

III. EXPERIMENTS

This section show a comparison between a vehicle simulation and a real vehicle experimentation thanks to a non trivial application: vehicle platoon system.

A. Application presentation

Linear platoon configurations, i.e., virtual trains of vehicles are a promising approach to new transportation systems [5], with innovative capabilities. Platoon systems, when applied to civil vehicles, have been mainly studied as a way to increase track density in highways. More recently, linear platoons have been studied as the basic technology to implement new passenger transportation services in urban environments, with a high adaptability to user needs and safety improvement thanks to automated

or semi-automated driving assistance (obstacle detection and avoidance, automatic car parking, etc.).

A basic problem in platoon systems is the control of the vectorial inter-vehicle distance. Some of the more followed approaches are based on automatic control. In this frame, the control of global platoon geometry has been decomposed in different sub-problems: longitudinal control (distance regulation), lateral control (angle regulation), integrated lateral and longitudinal control. Most of the lateral or longitudinal control proposals are based on PID (Proportional, Integral, Derivative) control [6][7].

A reactive, autonomous longitudinal and lateral control approach, intended for urban area transportation, with stringent conditions: small curve radius and constrained merge and split operations has been developed [1].

In this approach, vehicle's behavior and interactions are specified from a physics inspired model, as described next. For the sake of a simpler presentation, we abstract from lateral control and focus the presentation on rectilinear train's displacement.

A train is composed of n vehicles $V_0, \cdots, V_{n-1}$. The first one, V_0 is assigned the functions of navigation. Vehicle V_i ($i > 0$) measures the distance vector to vehicle V_{i-1} (the preceding one) and calculates an interaction force based on the mechanical laws of a virtual spring damper place between V_i and V_{i-1}. The interaction forces intervene in the calculation of an acceleration vector to be applied to the vehicle. The virtual spring damper model bases on stiffness *k*, damping factor *h* and spring's un stretched length l_0. The forces involved are : spring force $\vec{F}_s$ and the damping force $\vec{F}_d$. Each vehicle V_i is represented by its position $\vec{X}_i$ = [x_i,y_i]. The mass of the vehicle is denoted by *m* (we assume that each vehicle has the same mass). The distance between vehicles is $d = \|\vec{X}_{n+1} - \vec{X}_n\|$. Newton law of motion allows to calculate the vehicle acceleration in the preceding vehicle's reference:

Spring force	$\vec{F}_s = -k(\|\vec{X}_{n+1} - \vec{X}_n\| - l_0)\vec{u_{n+1\ n}}$
Damping force	$\vec{F}_d = -h(\dot{\vec{X}}_{n+1} - \dot{\vec{X}}_n)$

$$m * \vec{\gamma} = -k(\|\vec{X}_{n+1} - \vec{X}_n\| - l_0)\vec{u_{n+1\ n}} - h(\dot{\vec{X}}_{n+1} - \dot{\vec{X}}_n)$$

By integration, speed and vehicle's state (position and orientation) can be determined and the command law (vehicle's direction and speed) can be computed.

B. Simulation and experimental protocol

Based on this algorithm, simulation and experimentation scenarios are designed and performed to check the platoon evolution particularly during lateral displacement situations and a set of safety platoon conditions.

Simulation are realized with the simulator (cf. Figure 4, top) presented in this paper. The second platform is composed of RobuCAB and GEM electrical vehicles modified by the Systems and Transportations Laboratory (cf. Figure 3). These vehicles have been automated and can be controlled by a onboard system.

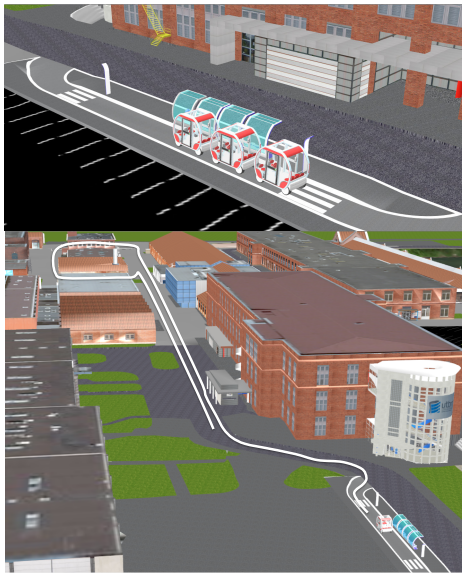

Figure 4. Simulation of a platoon vehicle Station (top) and Simulation and experimentation path (bottom)

Experiments were conducted on the Belfort's Technopark site. The simulations were performed on a 3D geolocalized model of the same site built from Geographical Information Sources and topological data.

Figure 4 (bottom) shows the path (white curve) used for the simulation and experimentation. This path allows to move the train on a long distance in an urban environment using a trajectory with different curve radius.

To compare the simulation and experimentation results, parameters were the same on simulation and real experiments. Thus, the perception of each vehicle is made by a simulated laser range finder having the same characteristics (range, angle, error rate, etc.) as the vehicle real sensor. The distance and angle between vehicles are computed thanks to this sensor.

The algorithm used is the same for both simulations and real experiments. Moreover, the program runs on the same computer. Indeed a great attention has been paid on the fact that simulated vehicles should have the same communication interface as the real ones. Thus, passing from simulation to real vehicle relies only on unplugging artificial intelligence computer from simulator and plugging it on real vehicle. However, experiments were performed with a more important regular distance in order to avoid collision that can lead to irreversible damage for vehicles. The regular distance has been established to 4 m and the safety distance to 1.5 m.

C. Comparison between experimentation and simulation

This subsection presents tests performed both in simulation and with real vehicles to assess the quality of platooning. The following cases were discussed:

- Evaluation of inter-vehicle distance: measuring the distance between two following vehicles, compared to the desired regular inter-vehicle distance during platoon evolution.

- Evaluation of lateral deviation: measuring the distance between the trajectories of the geometric center of a vehicle relative to the same path of his predecessor. For the measurement, points on the first vehicle trajectory were selected. Then, the normal trace of these points is drawn and a measure of the distance between the selected point and the point of intersection with the trajectory of its predecessor is made.

Figure 5 (top) shows the distance variations between vehicles in relation to quick changes of first vehicle speed.

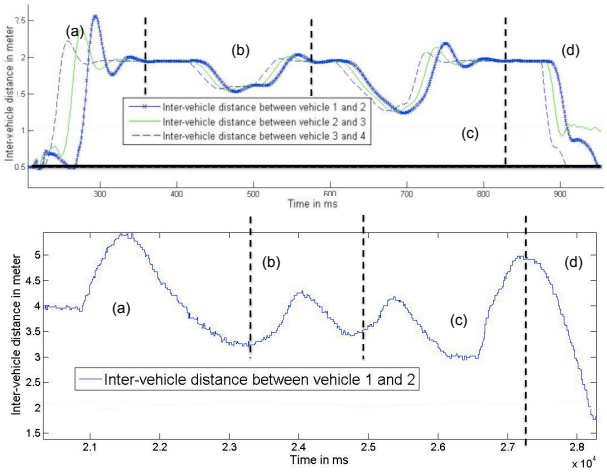

Figure 5. Inter-vehicle distance in simulation (top) and in real experiments (bottom)

Figure 5 (bottom) shows the distance evolution between two electric vehicles during the experiment.

One can observe that despite the very sudden changes in first vehicle speed, this value is above the safety distance and stabilizes rapidly to the regular distance. Figure 5 (top and bottom) present the same inter-vehicle variation.

The distance between the trajectories of the vehicle geometric center relatively to the same path of its predecessor has also been evaluated. Lateral deviation may cause problems in curves such as collision with vehicles in the opposite direction.

Wheel rotation (degree)	Curve	Medium error in simulation	Medium error in experimentation
5.73	18 m	12 cm	30 cm
11.46	9 m	30 cm	40 cm
17.2	6 m	50 cm	46 cm
22.9	4.5 m	55 cm	55 cm
28.65	3.6 m	67 cm	70 cm

Results presented in the table above show that the tracking error of the vehicle simulation is close to the experiment. Indeed, results representing the average error between each car of the train have same order values.

IV. Conclusion

This paper presented VIVUS simulator aimed at testing virtually prototyped vehicles with embedded intelligence and perception ability. Referring to the results obtained with this simulator and the comparison made with equivalent real experiments, VIVUS can be considered as a reliable tool for prototyping new intelligent vehicles and testing dynamical properties in critical and/or forbidden scenarios. Some research are now made to better simulate sensors focusing on video sensors and especially there defects such as lens flare phenomenon, chromatic aberration, climatic condition influence (rain, snow, water projection, etc.). VIVUS will now be used in the second phase of the CRISTAL project to test platoon functionnality with near-real virtual vehicles.

Acknowledgement

The authors would like to thank Professors Abderraffîâa KOUKAM, Yassine RUICHEK, and Pablo GRUER for their constant support and informed contributions.

References

[1] J.-M. Contet, F. Gechter, P. Gruer, and A. Koukam, "Bending virtual spring-damper : a solution to improve local platoon control," *Lecture Notes in Computer Science*, vol. 5544, 2009.

[2] F. Gechter, J.-M. Contet, P. Gruer, and A. Koukam, "Car-driving assistance using organization measurement of reactive multi-agent system," in *International Conference on Computational Science 2010 (ICCS 2010)*, Amsterdam, May 2010.

[3] S. Galland, N. Gaud, J. Demange, and A. Koukam, "Environment Model for Multiagent-Based Simulation of 3D Urban Systems," in *the 7th European Workshop on Multi-Agent Systems (EUMAS09)*, Ayia Napa, Cyprus, Dec. 2009.

[4] F. Michel, "The IRM4S model: the influence/reaction principle for multiagent based simulation," in *Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS07)*. ACM, May 2007.

[5] J. Hedrick, M. Tomizuka, and P. Varaiya, "Control issues in automated highway systems," *IEEE Control Systems Magazine*, vol. 14, no. 6, pp. 21 – 32, 1994.

[6] P. Daviet and M. Parent, "Longitudinal and lateral servoing of vehicles in a platoon," in *IEEE Intelligent Vehicles Symposium*, Tokyo, Jpn, 1996, pp. 41–46.

[7] M. J. Woo and J. W. Choi, "A relative navigation system for vehicle platooning," *SICE 2001. Proceedings of the 40th SICE Annual Conference. International Session Papers (IEEE Cat. No.01TH8603)*, pp. 28 – 31, 2001.