Ev-ADA: A Simulation-driven Evaluation Architecture for Advanced Driving-Assistance Systems

Assia Belbachir, Jean-Christophe Smal and Jean-Marc Blossevillle
Laboratoire de Mesure de la Mobilité Coopérative (LEMCO)
IFSTTAR
25 allée des marronniers
78000 Versailles, France
Email: firstname.name@ifsttar.fr

Sébastien Glaser
Laboratoire sur les Interactions Véhicules Infrastructure Conducteurs (LIVIC)
IFSTTAR
14, route de la Minière
78000 Versailles, France
Email: firstname.name@ifsttar.fr

***Abstract*—This paper reports the architecture of a simulator which is able to evaluate sensors, path planners and controllers of the advanced driving-assistance systems (ADAS). The outstanding feature of this simulator is that it is able to evaluate algorithms by giving scores. The implementation of the algorithms requires several tools such as Pro-SiVIC™. To have a good evaluation of the developed algorithms, we give a list in this paper of the requirements for an ADAS simulator. The simulator architecture and the developed algorithms are tested in several ADAS scenarios. Using Pro-SiVIC™ as a simulator, we are now able to evaluate different algorithms for ADAS.**

***Keywords*-Simulation architecture; Pro-SiVIC™; Evaluation; ADAS.**

I. Introduction

Advanced driving-assistance systems (ADAS) received an increasing attention from the car industry recently. To attract industrial attention, pieces of hardwares and softwares are developed. However, the software developments cannot work from the first time and can make costly damage. This is why, there is a strong need to ease the development and the validation process of different parts of hardware and software components. In this sense, using computational simulation techniques can be a candidate solution to this problem since it is cheaper in terms of time, money and human resource needed. By generating different types of vehicles, a simulator should be able to evaluate the vehicle's behavior. Up until now, several simulators are developed. They can be logically divided into two main groups. The first group focuses on simulating only one specific behavior, such as the camera perception or the path planning [1]. The second group simulates all the system's components behaviour at the same time especially for ADAS [2], [3]. The proposed simulator in this paper belongs to the second group, since the overall aim is to simulate and evaluate different sensors, path planning and control algorithms for ADAS. We use Pro-SiVIC™ [4] and ^{RT}Maps [5]. The former is able to generate the design (hardware) of different vehicles (e.g., the wheel's dimension, the environment, etc.) and the latter is used to implement different perception, path planning and control algorithms. In our case, we want to simulate and evaluate algorithms for ADAS. However, there are two questions rising: "What do we need to simulate?" and "How can we evaluate all the behaviors?" As a brief answer to the first question is that we are trying to simulate the perception, the path planning and the control part (see Figure 1). For the second question, the evaluation tests need to be realistic. We define realistic tests such as:

- Definition of different scenarios: the defined scenarios should simulate different road traffic cases, several kinds of road (e.g., motor-way), etc.

- Evaluation requirements: we need to define some criteria to evaluate algorithms. For that we define eight simulator

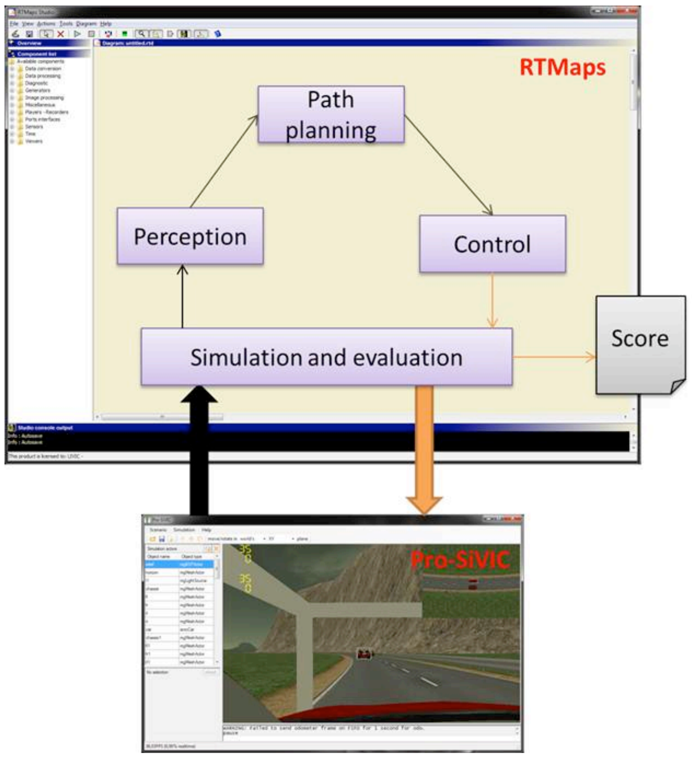

Figure 1. The general architecture to simulate the perception, path planning and control algorithms.

requirements for the ADAS (see Figure 2) and discuss them in Section III.

The originality of this paper is that, while simulating, we are able to *evaluate* algorithms. We applied our work to a project called: ABV (Automatisation de la conduite à Basse Vitesse sur des itinéraires sécurisés : low speed automation, on a safe trip). This project aims to automatize the driving in low speed (less than 50km/h). The system should be able to advice or take decisions for the safety of the driver and pedestrians/cars on the road. As safety is a main matter, we must evaluate each function of the system, choosing the best option available to realize them.

Our paper is organized as follow: first of all we explain the background and discuss briefly the used tools mainly Pro-SiVIC™. Section III describes our simulator requirements to evaluate different algorithms. Section IV shows the deployed architecture Ev-ADA and some experiments to show the system versatility. We conclude the paper by a discussion and future work.

II. Background

Several simulators have already been developed [6] such as MORSE [7], Player/Stage [8] and Gazebo [9]. In general, these simulators are used to imitate the behavior of a robotic system. However, these simulators do not evaluate algorithms by giving scores. Our aim is first of all to simulate the system and secondly to assess how the system is running. We defined several criterion to assess the ADAS system running. We used the simulator Pro-SiVIC™, that is a platform for prototyping sensors. We used ᴿᵀMaps to be able to implement the loop of perception–path planning–control by using different algorithms. The coupling between Pro-SiVIC™ and ᴿᵀMaps brings to ᴿᵀMaps the ability to observe simulated data from Pro-SiVIC™. As follow we explain how can Pro-SiVIC™ and ᴿᵀMaps works together.

A. Simulation using Pro-SiVIC™

Pro-SiVIC™ is developed in order to be independent of applications type. To be realistic, Pro-SiVIC™ integrates all functionalities allowing the most realistic possible graphical in the environment. *mg Engine* is the graphical 3D engine used. To reduce the computing board process, *mg Engine* uses a tree of binary positioning (BSP) (for more details ses [10]). To ensure its portability under numerous operating systems, this application is developed in C^{++} under LGPL with OpenGL and SDL libraries. In general several functionalities can be developed such as:

1. Simulated sensors*:* Several sensors can be simulated such as camera, inertial platform, odometer, telemeter, etc.

Camera (module sivicCamera): It simulates different sets of camera configured by using the Pro-SiVIC™ parameters or by using the parameters related to OpenGL.

Inertial Navigation System (module sivicInertial): this module simulates the inertial sensor.

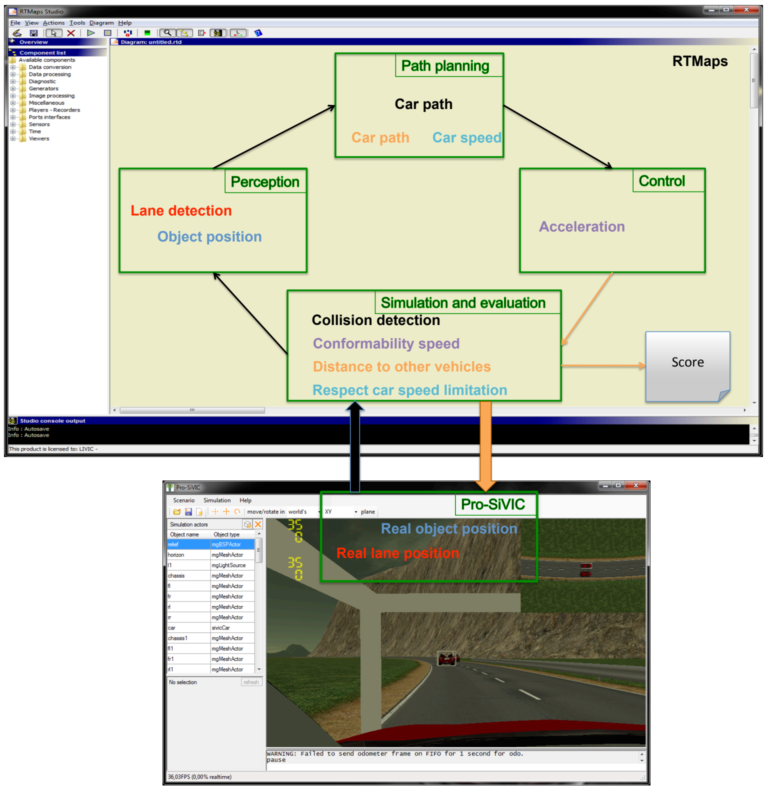

Figure 2. The general architecture to simulate the perception, path planning and control algorithms with different simulator requirements.

Odometer (module sivicOdometer): It provides the distance covered by a vehicle.

Telemetric scanner (module sivicTelemeter): This module simulates a laser scanner. Depending on the type of the telemeter, several methods can be implemented such as ray tracing or others.

2. Vehicle model*:* Three axes are defined : Roll, pitch and head. A generic model is able to reproduce the movement of the vehicle taking into account shock absorbers, viscosity and tie adherence [10]. In Pro-SiVIC™ other car models can be implemented and used from external libraries.

3. Mode changes*:* Several control modes are possible. The vehicle can be internally controlled by Pro-SiVIC™ features or externally controlled as in our case using ᴿᵀMaps.

B. Simulation under ᴿᵀMaps

We implemented sensors, path planner and lateral, longitudinal controllers under ᴿᵀMaps. [11]. The path planner redefines a path when the vehicle trajectory should be changed for example in case of an obstacle in front of the vehicle.

III. Simulator requirements for ADAS

ADAS are systems that assist the driver in his driving process. The main objective of these systems is to increase car safety and road safety. Such ADAS systems are adaptive cruise control (ACC), lane departure avoidance, lane keeping, emergency braking, etc. Every system is specialized in one topic (path, control, etc.). our work has been to define and to group all the required criterion to assess the ADAS simulated components. Eight requirements, explained below, have been selected :

1) *Lane detection error:* During these last years, a lot of algorithms were developed for road lane detection. Different types of sensor are used, such as LIDAR, RADAR, Camera [12], etc. Our simulator should be able to compare the real position of the lane with the perceived position lane. In the next subsection we detail how the error is computed ($\delta Lane$).

2) *Pedestrian detection error:* A lot of algorithms have been designed to detect pedestrians on the road. The objective of this detection process is to avoid collisions with pedestrian. It is hard to sense, process data and avoid the pedestrian when the car is at high speed. Intensive work has been done on this topic [13], however to ensure the correct pedestrian detection implies that the vehicle speed is limited. For assessing this process part, the error between the simulated pedestrian position and the detected pedestrian position has been computed (δPos_i, where i represent the pedestrian object).

3) *Car position detection error:* ADAS perception systems should detect other vehicles or objects in order to avoid collisions. Our assesment process computes the error of the simulated position of the car and the relative estimated position with other vehicles or objects(δPos_i, where i represent the car object).

4) *Car localization error:* Some dedicated process (odometer, GPS like etc.) is used to localize the vehicle on the road. A localization error should be computed to evaluate the localization correctness.

5) *Path planning error:* A huge number of algorithms have been developed for path planning, originally for robotics applications. These algorithms have as main criterion to avoid collision with existent objects and to reduce the computation time. Our objective is to evaluate the capability of the algorithms to avoid collisions with other objects, at any time ($\delta Collision$).

6) *Control/command error:* Algorithms of control allow to control the path execution. In general the speed and the direction of the vehicle are controlled (e.g., Longitudinal and lateral ref. to *e-value* project). ($\delta Speed$)

7) *Driver safety estimation:* It is the main requirement that should be taken into account in all ADAS systems. ADAS should warn the driver in case of high risk or should take the control to prevent accidents. For example, while driving too close to the preceding car, a sound signal can be used to prevent the driver or braking can be triggered($\delta Dist\_secur$).

8) *Driver comfort estimation:* Even if the comfort cannot be fully evaluated, some criterion should be respected, related to speed changes for example. Next section explains how each requirement is computed and let the simulator evaluates the perception–path planning–control/command loop regarding this comfort criteria ($Accel_{confort}$).

All these requirements are used to evaluate any ADAS system. In the next section, we explain how we can merge these requirements to define a final score.

IV. A SIMULATION DRIVEN EVALUATION ARCHITECTURE FOR ADAS

To satisfy the aforementioned requirements, developers need a tool that support sensor, path planner and control command specification and development. For this purpose, we used Pro-SiVIC™ with ^{RT}Maps that are fully able to support the algorithm specification and development tasks.

Our simulator is composed by Pro-SiVIC™ and ^{RT}Maps, where we have added a component that assess algorithms by giving scores. The detailed computed scoring process is explained in the following subsections.

A. Pro-SiVIC™ components and the link with ^{RT}Maps

Modeling cars, under Pro-SiVIC™ needs several components. Car description uses observers and sensors. Each car is also described with parameters, the wheel's dimension, the weight, etc. that are estimated from real car measurements. This ability to implement different vehicle shapes make our simulator versatile. The road shape is generated by using PathEdit. The latter is used to generate the vehicle path in a specific road. According to the road description, PathEdit generates a trajectory as a set of position coordinates and speed set points on the road.

The implementation of the environment in Pro-SiVIC™ is easy, the vehicle path being as well easily loaded. The dynamic model of a car is taken into account and can be modified under Pro-SiVIC™. Observers have been implemented to allow ^{RT}Maps to take the vehicle position in the simulated time.

B. Scores

We developed a component called *ABVsim* under ^{RT}Maps from observations (e.g., CarObserver). This component pro-

Structure: Ego structure.

```
1: struct Ego {
   int $id$; $index\_lane$;$number\_ego$;$gear$;
   Vehicle_Type $type$;
   double $position\_x$;$position\_y$;$heading\_xy$;
   $position\_x\_standard\_deviation$;
   $position\_y\_standard\_deviation$;
   $heading\_xy\_standard\_deviation$;
   $yaw\_rate$;$speed\_x$;$speed\_y$;
   $acceleration\_x$;$acceleration\_y$;
   $yaw\_rate\_standard\_deviation$;
   $speed\_x\_standard\_deviation$;
   $speed\_y\_standard\_deviation$;
   $acceleration\_x\_standard\_deviation$;
   $acceleration\_y\_standard\_deviation$;
   $steering\_angle$;$timestamp$;
   Indicators: $indicators$;
   float $revolutions\_wheel\_rear\_right$;$revolutions\_wheel\_rear\_left$;
   $revolutions\_wheel\_front\_right$;$revolutions\_wheel\_front\_left$;
   $revolutions\_motor$;$weight\_empty$;
   $position\_x\_rear$;$position\_x\_front$;
   $position\_y\_right$;$position\_y\_left$;
   $radius\_wheel\_rear$;$radius\_wheel\_front$;
   $speed\_x\_minimum$;$speed\_x\_Max$;
   $acceleration\_x\_minimum$;$acceleration\_x\_Max$;
   $curvature\_Max$; $ratio\_steering\_wheel\_on\_front\_wheel$;
   Clutch $clutch$;
   Charge $charge$;
   Status $status$;
   };
```

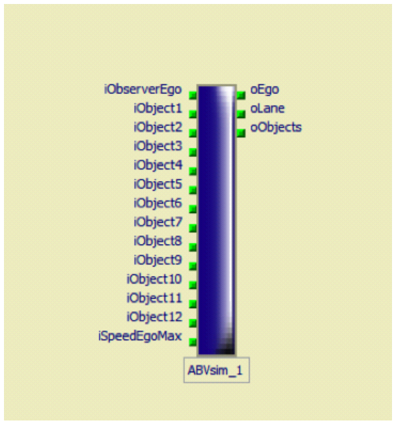

Figure 3. Illustration of different input and output in the ABVsim componant.

vides a data structure such as required in the **ABV** project (see Figure 3).

Fifteenth entries are developed described as follow:

- *iObserverEgo* is an entry that observes from Pro-SiVIC™ the position of the car.
- *iObject1,.., iObject12* are entries to observe other object's position such as pedestrian or cars.
- *iSpeedEgoMax* is the maximal allowed speed of the vehicle.

Three outputs are developed such as:

- *oEgo* is the output of the vehicle position, speed, etc.
- *oLane* is the output of the lane detection using the appropriate sensor. This structure contains also the error of the lane detection.
- *oObject* is related to the existing objects in the environment such as pedestrian or cars.

Different scores are computed for each set of sensors, path planning, control and safety/comfort algorithms as follow:

$$Score_{sensor} = \delta Lane + (\delta \sum_{i=1}^{12} Pos_i) \quad (1)$$

$$Score_{planning} = \delta Collision \quad (2)$$

$$Score_{control} = \delta Speed + \delta Direction \quad (3)$$

$$\begin{aligned} Score_{comfort/security} &= Accel_{confort} \\ &+ \delta Dist\_secur \end{aligned} \quad (4)$$

The sensing score ($Score_{sensor}$) is associated to the lane error detection ($\delta Lane$) and the detection error of other object positions ($\delta \sum_{i=1}^{12} Pos_i$). $\delta Lane$ is a normalized distance between the real lane position and the estimated lane position. The normalized value is between 0 and 1.

The path planning score is related to the collision criterion. If, while running path planning algorithms, the car collides with another object, the $\delta Collision$ is equal to zero.

The controller score $Score_{control}$ represents the compliance with the maximal speed and the direction to be followed. When the vehicle exceed the maximal speed the value $\delta Speed$ is equal to zero. When the car does not follow the road, the $\delta Direction$ value is equal to zero.

The $Score_{comfort/security}$ is the main objective of ADAS systems. $Accel_{confort}$ represents a score between the maximal acceleration allowed for a vehicle and the actual vehicle acceleration. $\delta Dist\_secur$ is related to the distance between the vehicle and other vehicles. In general, this distance should corresponds to a car interval of 2 seconds.

All these scores are normalized between 0 and 1. The higher the normalized score value, the better the score is. The normalization procedure for each value is as follows. $\delta Lane$ is normalized by dividing the result by the traffic lane width. Pos_i is normalized by the car/pedestrian dimension. $\delta Speed$ is normalized by the maximal allowed speed. $Accel_{confort}$ is normalized by the maximal acceleration that the vehicle can drive. $\delta dist_{secur}$ is normalized by the whole driven distance.

V. Simulation results

In our simulation, we are using Windows 7 Professional under Intel(R) Core(TM) i7 CPU 950 @ 3.07GHz, 64 bits.

In this scenario, we run two vehicles. One vehicle is a Mini Cooper and the second vehicle is a Megan Renault. All the algorithms are implemented in the Mini Cooper car

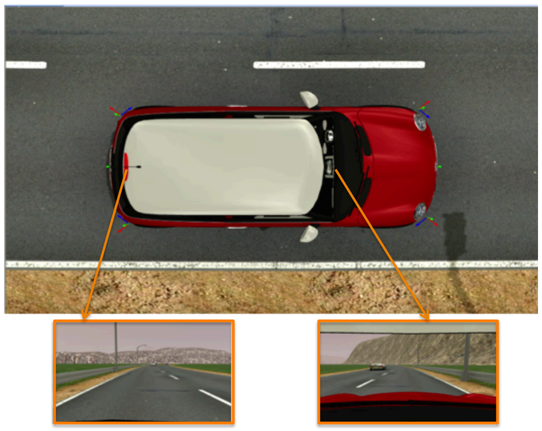

Figure 4. Illustration of different sensors in Ego car.

called Ego. This one follows the second car (Megan Renault) called Car1.

Our path planning algorithm allows the vehicle to follow another vehicle keeping a minimal safety distance with the preceding vehicle (1) and respecting a maximal speed (2).

Ego structure is represented in Figure 4, where two cameras are implemented in the upper front of Ego. A path-planning algorithm is implemented based on Camera1. This camera detects the road surface marking. Ego should follow Car1 at any time and at the same time do not exceed the maximal predefined speed. Several tests are implemented, where we vary the maximal speed and safety distance between Ego and Car1.

All the tests are evaluated in a horse-ring circuit represented in Figure 5.

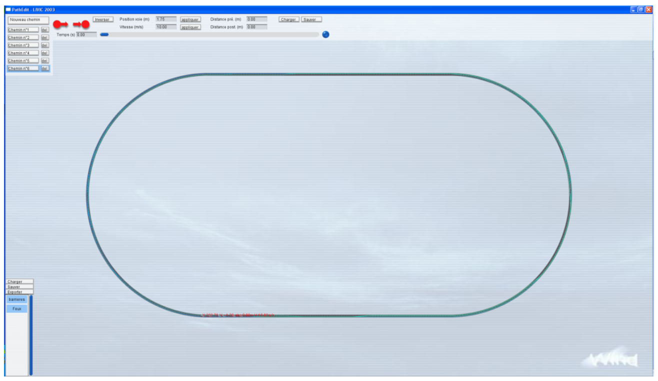

Figure 5. Illustration of a horse-ring circuit for both Ego and Car1.

Different strategies can be used to compute a score. An example of total score utilization is shown as follow:

$$\begin{aligned} Score &= \frac{1}{\gamma+\alpha+\beta+\Gamma} \\ &* \ [\gamma Score_{sensor} \\ &+ \ \alpha Score_{planning} \\ &+ \ \beta Score_{control} \\ &+ \ \Gamma Score_{confort/security}] \end{aligned} \tag{5}$$

γ, α, β and Γ are coefficients. Depending on the coefficient values, some related parameters can be more important than others.

In our case, the control and security/comfortability are the main part that the system should respect, this is why :

$$\beta+\Gamma>\alpha+\gamma \tag{6}$$

We use a weight of β=1, Γ= 2 and a weight of α=γ=1.

Case studies of our score		
Speed Ego < Speed Car1	0s	(0.89 + 0.0 + 0.7 + 0.5) /4 = 0.52
	1s	(0.89 + 1.0 + 0.7 + 0.5) /4 = 0.77
	2s	(0.89 + 1.0 + 0.7 + 0.5) /4 = 0.77
	3s	(0.89 + 1.0 + 0.7 + 0.5) /4 = 0.77
	4s	(0.89 + 1.0 + 0.7 + 0.5) /4 = 0.77
Speed Ego > Speed Car1	0s	(0.89 + 0.0 + 0.7 + 0.5) /4 = 0.52
	1s	(0.89 + 0.0 + 0.7+ 0.5) /4 = 0.52
	2s	(0.89 + 0.0 + 0.7+ 0.6) /4 = 0.54
	3s	(0.89 + 1.0 + 0.7+ 0.7) /4 = 0.82
	4s	(0.89 + 1.0 + 0.7+ 0.7) /4 = 0.82

Figure 6. Illustration of the obtained score using the equation 5

Figure 6 represents different case studies of our score. When the maximal Ego speed is less than Car1 ones, the higher score is 0.77. When the maximal Ego speed is greater than Car1 ones, the higher score is 0.82. Due to the low Ego speed, this one can not follow Car1. This difference of score is only related to the speed divergence. This sceanrio shows that our platform is able to evaluate different implemented algorithms on a simulation mode.

VI. Conclusion and Future Works

Our contribution aims at defining an architecture and a framework to evaluate various types of advanced driving-assistance systems (ADAS). In our experiments, an ego car is used to follow another car on a horse-ring road. Each algorithm part (perception, path planning, task control) is evaluated using different types of scores. To extend the Pro-SiVIC architecture, an evaluator based on proposed criteria has been implemented. These criteria are: (1) Lane detection error, (2) Pedestrian detection error, (3) Car position detection error, (4) Car localization error, (5) Path planning error, (6) Control/command error, (7) Driver safety estimation (8) Driver comfort estimation.

The evaluation of the tested algorithms related to lane detection (based on camera), path planning, control command and comfort/safety of the driver, gives a satisfied score. Our Ev-ADA simulator is now able to evaluate different types of algorithms working on different types of scenarios.

This work opens perspectives. As future works, we plan to evaluate other algorithms in other case studies, varying not just the speed, but also external parameters such as the weather, the traffic, etc. As a matter of fact, the versatility of Pro-SiVIC allows us to evaluate algorithms in various conditions including raining, cloudy, dark weather associated with different car traffic situations.

We will be also able to compare different algorithms between them in the same reproduced conditions. This work will contribute to obtain the best ADAS systems suitable for drivers, safety criteria included. Nevertheless, even if, working with simulation tools reduces works, time and resources, we should recognize that real experimentations will be necessary to take account driver perceptions.

Acknowledgment

The authors would like to thank all the LEMCO team and Dominic Gruyer for allowing us the use of the simulator. We would like to thank also CIVITEC for their daily help.

References

[1] T. Siméon, J. p. Laumond, and F. Lamiraux, "Move3d: a generic platform for path planning," in *in 4th Int. Symp. on Assembly and Task Planning*, 2001, pp. 25–30.

[2] M. Parent, "Advanced urban transport: Automation is on the way," *IEEE Intelligent Systems*, vol. 22, pp. 9–11, 2007.

[3] "Haveit eur. project [online]. available: http://www.haveit-eu.org/," Last access date 10/2011.

[4] N. Hiblot, D. Gruyer, J.-S. Barreiro, and B. Monnier, "Pro-sivic and roads, a software suite for sensors simulation and virtual prototyping of adas," *DSC2010 Driving Simulation Conference*, 2010.

[5] F. Nashashibi, B. Steux, P. Coulombeau, and C. Laurgeau, ""rtmaps a framework for prototyping automotive multi-sensor applications," *In Proc. of the IEEE Intelligent Vehicles Symposium*, 2000.

[6] S. Petters, D. Thomas, M. Friedmann, and O. Stryk, "Multilevel testing of control software for teams of autonomous mobile robots," in *Proceedings of the 1st International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, ser. SIMPAR '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 183–194.

[7] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular Open Robots Simulation Engine: MORSE," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, 2011.

[8] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *11th International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, 2003. [Online]. Available: citeseer.ist.psu.edu/gerkey03playerstage.html

[9] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2149–2154.

[10] M. S. Glaser, "Modélisation et contrôle d'un véhicule en trajectoire limite : application au développement d'un système d'aide à la conduite," Ph.D. dissertation, Ecole Doctorale Sitevry (Université Evry-Val-D'Esonne), 12 mars 2004.

[11] S. Glaser, V. Benoit, S. Mammar, D. Gruyer, and L. Nouvelière, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *Trans. Intell. Transport. Sys.*, vol. 11, pp. 589–606, September 2010.

[12] J. McCall and M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 20 –37, 2006.

[13] L. Oliveira and U. Nunes, "Context-aware pedestrian detection using lidar," in *In: IEEE Intelligent Vehicles Symposium*, 2010, pp. 773–778.