Mixed Reality Autonomous Vehicle Simulation: Implementation of a Hardware-In-the-Loop Architecture at a Miniature Scale

Robin Baruffa
Univ. de Technologie de Belfort-Montbéliard
Belfort, France
email: robin.baruffa@utbm.fr

Jacques Pereira
Univ. de Technologie de Belfort-Montbéliard
Belfort, France
email: jacques.pereira@utbm.fr

Pierre Romet
CIAD (UMR 7533)
Univ. Bourgogne Franche-Comte, UTBM
Belfort, France
email: pierre.romet@utbm.fr

Franck Gechter
CIAD (UMR 7533)
Univ. Bourgogne Franche-Comte, UTBM
Belfort, France
email: franck.gechter@utbm.fr
Mosel Loria (UMR CNRS 7503)
Université de Lorraine
Vandoeuvre-lès-Nancy-54506, France

Tobias Weiss
CIAD (UMR 7533)
Univ. Bourgogne Franche-Comte, UTBM
Belfort, France
Institute of Energy Efficient Mobility
Univ. of Applied Science
Hochschule, Karlsruhe, Germany
email: tobias.weiss@hs-karlsruhe.de

***Abstract*—Validation of autonomous vehicles is a resource intensive and time-consuming endeavour because of their *safety critical* nature. *X-In-the-Loop* proposes a development method that uses a simulated environment to overcome real-world constraints to test and improve autonomous vehicle capabilities. Development time can be reduced as each iteration on the control software does not necessarily require real-world testing. This paper focuses on studying the influence of switching from simulated to real-world camera data on control algorithms execution time in the context of a preliminary work consisting in implementing a *Hardware-In-the-Loop* architecture on a *miniature car*.**

***Keywords*—Hardware-In-the-Loop; autonomous vehicles; miniature car.**

I. Introduction

The project detailed in this paper is part of a new alternative freight transportation system in cities and in the countryside called SURATRAM (Système Urbain et Rural Autonome de TRansport de Marchandises). It aims to revitalize small businesses struggling to meet the level of flexibility and competitiveness that e-commerce companies can offer. The proposed solution is to deploy an autonomous fleet of cargo vans in various areas to facilitate the transportation of goods, lowering transportation fees by optimizing the path of each package and lessening the need for maintaining extensive stocks; and improving logistic chains' flexibility and speed. The increase in traffic caused by the additional number of vehicles among cities would be mitigated by restricting these autonomous vans to follow existing human-operated public transport vehicles such as tramway or buses in a *platoon* formation. Thus, the next logical step in the development of this transportation system is using a real vehicle as a testbed in order to assess the state of the art platoon algorithms [1]. This *System-under-Test* (SuT) must demonstrate its reliability in various extreme scenarios such as an imminent collision with a pedestrian or an emergency stop performed by any vehicle from the convoy. Extensive testing of hardware in real driving conditions can be very costly as it requires having access to special facilities, and in our case, renting a bus with a driver during the whole test campaign. A solution to this problem is to develop a simulated replica of the real-world system dynamic and its sensors, allowing for thorough validation of the SuT by breaking free from physical and time constraints. With this method, known in the literature as *Hardware-In-the-Loop* (HIL) [2], the real hardware is used to process the simulated sensor data and outputs actions that are fed back in the simulation.

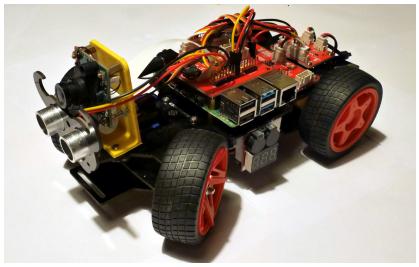

Figure 1. Image of the miniature car.

In order to pave the way for the future design of a complete HIL implementation of a full-sized car, a *miniature car* has been designed to verify the relevance and performance of the chosen HIL architecture (Figure 1).

This preliminary work aims at highlighting the difference in the control algorithm execution time when fed with real-world or with simulated camera data. It is an important metric to check as if the gap is too wide, the control algorithm will not behave as expected when tested in real conditions.

In Section II, we first present the current state of the art in the autonomous vehicle development field. Section III then describes our implementation of the HIL framework on a miniature car. Section IV provides a description of the experimental protocol and the preliminary results regarding

 ISBN: 978-1-61208-831-0

the gap between car and its cyber-twin. Finally, in Section V, we close this paper with a summary and an outlook on the future improvements needed to validate the simulation.

II. State of the art

In the literature, the general framework *X-In-the-Loop* is the subject of numerous works [3][4], especially in the robotics and autonomous vehicle field where the SuT can be a *safety-critical system* [5].

The classical approach is to use *Model-In-the-Loop* [6] as a first step to create a mathematical representation of the system's dynamics.

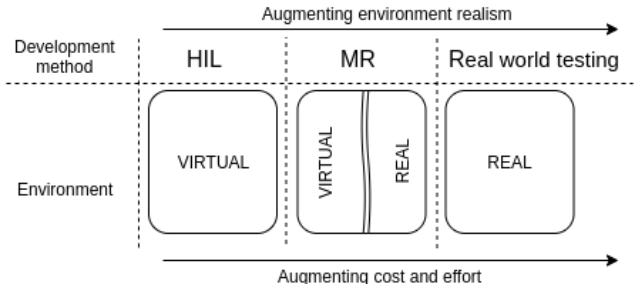

Figure 2. Environment in HIL architecture to MR [7].

In some cases, limitations can occur both in simulation (e.g., in terms of fidelity) and in the real world (e.g., in terms of price and safety), therefore a hybrid framework like *Mixed Reality* (MR) [8] can be used to mitigate this. Its purpose is to create hybrid observations by mixing simulated and real observations in order to add entities such as pedestrians, cars or obstacles [9][10]. It can provide an intermediary step between the relatively easy and inexpensive HIL, and costly real-world testing [7]. Figure 2 shows how much time and financial investment is required for each development method.

As it is critical to make sure that the SuT will perform similarly in real world testing and in its virtual world, numerous works [11][12] focus on quantifying the difference in dynamics between the real world model and the simulated model. We think software execution time differences deserve to be studied in more detail as they can influence the overall behaviour of the SuT.

III. Proposition

The short-term goal of this project is to implement HIL on a miniature car and its cyber-twin to have it *follow a simulated bus,* both in the simulation and in the real world. Our architecture allows for testing the control algorithm with sensor data coming from the simulation or from the real car interchangeably, which is a key component of HIL methods. Additionally, when using only simulated observations, it is possible to execute the actions in an *open-loop configuration* on the real-world car, with the purpose of verifying the fidelity between the simulated vehicle dynamics and its real counterpart.

Related work focuses first on developing the control algorithm in simulation, then implementing it in a test platform. A key advantage of our miniaturisation method is its simplicity and low cost. After a description of the global architecture, this section will go into details about the subsystems architecture.

A. Global architecture

On the one hand, a computer runs the simulation, which updates the car's position and velocity when receiving an action, then sends the simulated sensors readings over the network. On the other hand, the raw data is fed to a decision making unit running on the real car's hardware, which outputs an action composed of throttle level and steering angle. This action will be transmitted to both the simulator and the *Electronic Control Unit* (ECU), controlling the real vehicle.

Figure 3 shows the overall architecture. The top simulation part is executed on a computer because of its computationally intensive nature, while the bottom part is executed on the *miniature car*'s embedded hardware.

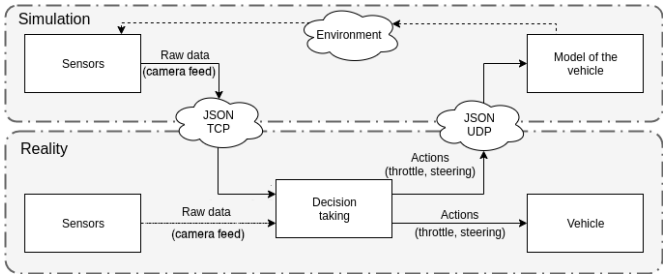

Figure 3. Global architecture of our HIL implementation.

B. Simulator architecture

Simulators have different goals and capabilities. The requirements are that they must offer both a realistic and customizable graphical environment, along with realistic physics. In literature, the use of a robotic simulator like *Gazebo* [7][9][10][13] or game-based engines such as *Unity* [8][14] or *Unreal-Engine* [15] are standard.

For this work, it was decided to choose Unity3D (which uses the *PhysX 4.1* physical engine), as it comes free of charge for non-commercial use and has good rendering capabilities which are needed for realistic camera sensors.

A generic 3D model of a car and a bus are used (Figure 5). Their dynamical parameters are set arbitrarily, as having an accurate physics model is currently beyond the scope of this project, but it will be investigated in future works. The bus can either be user-controlled or driven by a predefined spline. A virtual camera and a tachometer measuring the rotation speed of the wheels are used as sensors.

C. Software architecture

The simulation layer is managed by Unity3D. A script parses the JSON (JavaScript Object Notation) output from the simulator (camera feed and odometry), then forwards it through the local network to the real car. The latter sends the received observation information to be processed over the *Robotic Operation System* (ROS) network. An output action

 ISBN: 978-1-61208-831-0

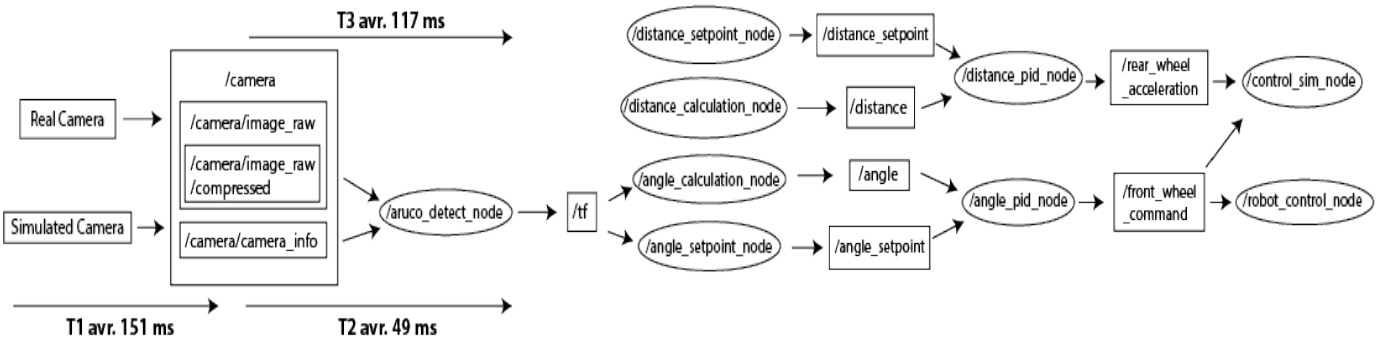

Figure 4. ROS computation graph along with measured computation time.

is then forwarded simultaneously to the ECU and back to the simulation. The ROS middleware is very convenient and often used in the literature [7][9] as it integrates many sensors drivers, natively supports multi-machine communication, integrates various visualization packages and offers the possibility to record and replay every incoming observation and outgoing actions in a deterministic manner.

Since the scope of this project is not about the sensor processing nor about the control algorithm, both have been simplified in the following way :

1) Camera processing: An *ArUco marker* (Figure 5) is used on the back of the bus. These markers are widely used for robotic and autonomous vehicles applications [16], as their detection and position estimation is made easier thanks to their simple binary shape. A ROS node called /aruco_detect_node receives the camera stream (from the simulation or the real sensor) and outputs the marker's coordinate system concerning the camera (shown as /tf in Figure 4). A copy of the video feed with the detected axis system overlaid is used for debugging purposes.

2) Control algorithm: The relative distance between the car and the bus is calculated and fed into a PID (Proportional Integral Derivative) controller that outputs a throttle command based on this distance and a user-defined distance setpoint parameter. The car's heading error is defined as the angle between the centre of the camera and the lateral position of the ArUco marker. This heading error is fed to a PID controller along with the desired heading angle (null in this case) and outputs a steering command.

D. Hardware architecture

The model car is a $25cm$ long and $14cm$ wide four-wheel drive with front steering wheels and rear propulsion. Its on-board computer is a *Raspberry PI 4* running the ROS server which communicates with the simulation computer via a Wi-Fi connection. The steering angle and throttle commands are transmitted to an external ECU that controls the motors in an open-loop configuration through GPIO (General Purpose Input Output) pins. The front-mounted camera is set to have a resolution of $640 \times 480px$ at $10Hz$.

IV. Experiments

This section will present two different experiments. The first one consists in controlling the bus manually in the simulation while the car will follow it from a safe distance. Then it is visually confirmed if the miniature car movements are both synchronized and coherent with the simulation.

The second experiment will aim to demonstrate that our HIL infrastructure can operate in real testing conditions: a manually controlled three-wheeled robot will represent the bus in the real world. The model car will run the same software and its performance will be visually assessed. In the two experiments, measurements are performed to quantify both delays in sensor data transmission from the simulation to the car's local network and the camera processing time resulting in usable information for the control algorithm. Comparing response time in HIL and real-world configuration permits to assess the coherence between both systems, which is critical since any delay discrepancy in the control loop can lead to different behaviours.

In Figure 4, T_1 represents the delay between the rendering of the virtual camera and its reception on the ROS network, and T_2 and T_3 are the times taken by the ArUco detection node to compute the marker coordinate from simulated and real camera data respectively. Delays measurements of the robot using simulated observations were performed on 308 samples and 343 samples when using real-world observations. The camera framerate was set to 5 Hz, which is the maximum frequency the ArUco detection package would handle on our hardware before starting to drop measurements.

Figure 4 shows a significant difference in delay between the two experiments, even though it is performed by the same code running on the same hardware. The only difference is the content of the video stream, as it has been setup to use the same compression format, encoding, framerate and resolution.

A. Real car guided by the simulated one

The leading virtual bus is placed at a distance from the virtual car, which will have to accelerate then slow down to get to its distance setpoint from the bus and maintain it while the simulated bus follows a predefined spline trajectory. The real miniature car will perform the same actions as its virtual twin.

 ISBN: 978-1-61208-831-0

The laptop running the simulation and the embedded computer were synchronized beforehand to avoid timing issues.

In the current absence of additional sensors allowing for precise location, we visually confirm that the miniature car follows the same trajectory as its cyber-twin.

B. Real hardware in real world condition

The purpose of this specific experiment is to demonstrate that the same software can be used in real conditions with real sensor data. Since the physical car is at a smaller scale, a robotic platform will act as the leading bus and will be user-controlled. This robot also uses ROS, which facilitated its implementation. The distance setpoint and the ArUco marker width were edited to take into account the scale difference.

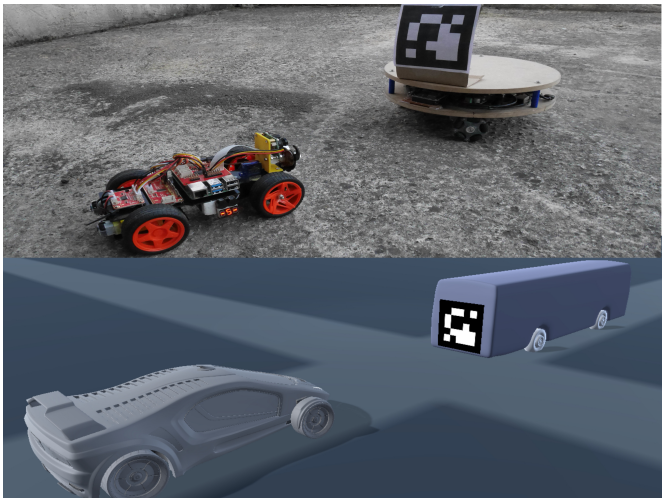

Figure 5. Miniature car in real-world test condition.

An overview of the experimental apparatus can be seen in Figure 5. Visual observation confirms that the car follows the robot from a safe distance, without significant change in behaviour compared to Subsection IV-A.

The latency increase in real-world conditions is likely to be caused by the increased complexity of the video stream. The execution time of image processing algorithms with an adaptive number of iterations, such as the one used for detecting ArUco markers, can be shorter if the input image is sharp and contains little noise. Thus, adding noise to simulated observations is not only needed for ensuring that the sensor processing algorithm will be robust in non-ideal condition, but it can also be a source of discrepancy in the behaviour of the same algorithm given different sensor data.

V. Conclusion and Future Work

This paper has presented a use case of HIL for autonomous vehicle implementation at a smaller scale. It has proven the effectiveness of the chosen combination of simulation, software and hardware. The global architecture proved to yield similar dynamics behaviour when sensory information comes from a simulated or a real-world environment, but a difference in computation time has been observed. The specific scope of use will dictate whether this delay discrepancy is acceptable.

Throughout the development of our preliminary platform, we learned the importance of choosing algorithms that take a fixed amount of time to execute, regardless of the input complexity. One of the most challenging tasks was to measure delays at every step of the computation graph without interfering significantly with the normal execution of the code.

Future work will focus on quantifying the dynamical differences between the simulated car and its real counterpart by adding additional sensors, either on the car or in the test environment, to get ”ground truth” measurements. We will also investigate how ROS2, the new version of the ROS middleware, performs to further lower latency. A simple yet effective implementation would be to use additional fixed ArUco markers to estimate the car’s position with good accuracy [16]. The next step towards a practical solution could be to use more advanced computer vision algorithms that do not require modification on the vehicle that needs to be followed.

These incremental improvements will ultimately lead to using a real autonomous vehicle platform that will be used as a testbed for developing novel MR architectures. It would be possible to overlay virtual obstacles on top of the real-world camera feed to improve the control algorithm’s performance and lower development time.

Acknowledgment

This work is carried out as part of our studies at the Université de technologie de Belfort-Montbéliard, in the context of the SURATRAM Project. The SURATRAM project is made possible thank to the support of Région Bourgogne-Franche-Comté.

References

[1] F. Gechter, J.-M. Contet, S. Galland, O. Lamotte, and A. Koukam, “Virtual intelligent vehicle urban simulator: Application to vehicle platoon evaluation,” *Simul. Model. Pract. Theory*, vol. 24, pp. 103–114, 2012.

[2] J. A. Ledin, “Hardware-in-the-loop simulation,” in *Embedded Systems Programming*, vol. 12, 1999, pp. 42–60.

[3] W. Huang, K. Wang, Y. Lv, and F. Zhu, “Autonomous vehicles testing methods review,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 163–168.

[4] J. E. Stellet, M. R. Zofka, J. Schumacher, T. Schamm, F. Niewels, and J. M. Zöllner, “Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 1455–1462.

[5] O. Gietelink, “Design and validation of advanced driver assistance systems,” in *TRAIL Research School*, 2007.

[6] A. R. Plummer, “Model-in-the-loop testing,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 220, no. 3, pp. 183–199, 2006.

[7] M. R. Zofka, M. Essinger, T. Fleck, R. Kohlhaas, and J. M. Zöllner, “The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality lidar stimulation,” in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2018, pp. 151–157.

[8] F. Gechter, B. Dafflon, P. Gruer, and A. Koukam, “Towards a hybrid real/virtual simulation of autonomous vehicles for critical scenarios,” in *In The Sixth International Conference on Advances in System Simulation (SIMUL 2014)*, 10 2014, pp. 14–17.

 ISBN: 978-1-61208-831-0

[9] M. R. Zofka et al., "Traffic participants in the loop: A mixed reality-based interaction testbed for the verification and validation of autonomous vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3583–3590.

[10] I. Y. Chen, B. MacDonald, and B. Wunsche, "Mixed reality simulation for mobile robots," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 232–237.

[11] C. Koehler, A. Mayer, and A. Herkersdorf, "Determining the fidelity of hardware-in-the-loop simulation coupling systems," in *2008 IEEE International Behavioral Modeling and Simulation Workshop*, 2008, pp. 13–18.

[12] N. Seegmiller, F. Rogers-Marcovitz, G. Miller, and A. Kelly, "Vehicle model identification by integrated prediction error minimization," in *The International Journal of Robotics Research*, vol. 32, 07 2013, pp. 912–931.

[13] I. Chen, B. A. MacDonald, and B. Wünsche, "A flexible mixed reality simulation framework for software development in robotics," in *Journal Of Software Engineering In Robotics*, vol. 2, 2011, pp. 40–54.

[14] M. Wu, S.-L. Dai, and C. Yang, "Mixed reality enhanced user interactive path planning for omnidirectional mobile robot," *Applied Sciences*, vol. 10, p. 1135, 2020.

[15] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 33–42.

[16] J. Bacik, F. Durovsky, P. Fedor, and D. Perdukova, "Autonomous flying with quadrocopter using fuzzy control and aruco markers," in *Intelligent Service Robotics*, vol. 10, 2017, pp. 185–194.

 ISBN: 978-1-61208-831-0