Augmented Reality on LiDAR data: Going beyond Vehicle-in-the-Loop for Automotive Software Validation

Thomas Genevois, Jean-Baptiste Horel, Alessandro Renzaglia, Christian Laugier

Augmented Reality on LiDAR data: Going beyond Vehicle-in-the-Loop for Automotive Software Validation

Thomas Genevois[1], Jean-Baptiste Horel[1], Alessandro Renzaglia[2] and Christian Laugier[1]

***Abstract*— Testing and validating advanced automotive software is of paramount importance to guarantee safety and quality. While real-world testing is highly demanding and simulation testing is not reliable, we propose a new augmented reality framework that takes advantage of both environments. This new testing methodology is intended to be a bridge between Vehicle-in-the-Loop and real-world testing. It enables to easily and safely place the whole vehicle and all its software, from perception to control, in realistic test conditions. This framework provides a flexible way to introduce any virtual element in the outputs of the sensors of the vehicle under test. For each modality of sensing, the framework requires a real time augmentation function that preserves real sensor data and enhances them with virtual data. The LiDAR data augmentation function is presented together with its implementation details. Relying on both qualitative and quantitative analysis of experimental results, the representability of tests scenes generated by the augmented reality framework is finally proven.**

I. INTRODUCTION

Autonomous drive and advanced driver assistance software have shown an outstanding development in the last decade. The performance of this software is nowadays reaching and maybe going beyond the level of human drivers' skills. This is bringing a drastic change in the automotive industry. Despite this outstanding progress, few of the recent research developments have been applied to prototype vehicles and extremely few have been transferred to commercial applications. This is mainly due to the lack of convenient testing tools and validation procedures.

The automotive industry commonly uses the V model development where every step in the conception of a software has to be tested and validated. This industry also adopts well defined tests and validation procedures such as the ones of Euro NCAP [1]. This guarantees standards of safety, reliability and quality. However, the complexity of recent AI-based algorithms poses new challenges in terms of validation. Moreover, critical scenarios, where it is more relevant to test the systems, involve potentially dangerous situations, such as unexpected behaviors of other road actors (cars, pedestrians, bikes, etc.) and may lead to collisions. Testing advanced automotive software in these scenarios is time-consuming and costly, involving prototypes and dedicated tests sites with complex infrastructures, such as actuated dummies [2]. Realistic simulation frameworks are a commonly adopted solution to overcome these problems and easily test many different scenarios without taking any risk. However, the representability of tests in simulation to validate real systems is not proven and pure simulations cannot be the only answer. As a result, nowadays the most common validation procedure currently used in industry is to drive a fleet of prototype vehicles for millions of kilometers and the main metric to assess the performance remains the average distance run without an error or a disengagement. Yet new approaches [3] emerge. Automotive software validation still needs more convenient testing tools to enable a proper validation procedure and quality assessment.

[1]Univ. Grenoble Alpes, INRIA, 38000 Grenoble, France.
[2]Univ Lyon, Inria, INSA Lyon, CITI, F-69621 Villeurbanne, France.
e-mail: `firstname.lastname@inria.fr`

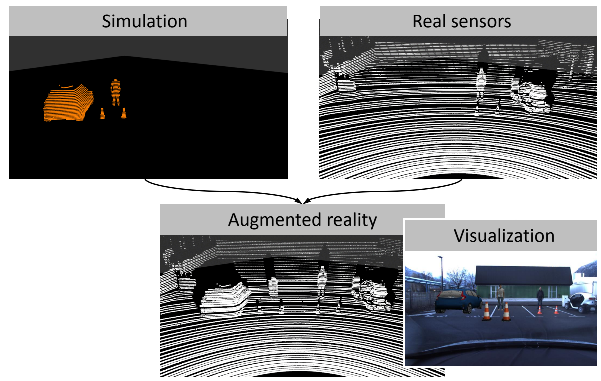

Fig. 1. Example of LiDAR point cloud augmentation. The introduced virtual point cloud and initial sensor point cloud are shown on the top. The technique that we present, deployed on a vehicle, generated the fused point cloud and the visualization of the augmented scene on the bottom.

In this paper, we propose a new framework based on augmented reality for testing advanced automotive software. The key aspect of this method is the design of a merge function allowing a real-time augmentation of LiDAR data with virtual elements (see Fig. 1 for an illustrative example). With this solution, we open new possibilities for testing. A testing site can very easily be populated with many and diverse virtual elements in order to create more complex test scenarios. Virtual pedestrians or cars are easier to operate and offer richer and more active behaviors (e.g. reacting to the ego-vehicle's motion). Furthermore, all elements of the test scenario that may induce a collision risk can be replaced by their virtual counterpart to secure the tests in the early stages of development or to test the system in critical situations. Virtual scenarios are also repeatable and this is a key feature to reproduce experiments. Our AR[1] testing implementation accurately represents the virtual scenes and guarantees a consistent fusion with the real world. So AR tests produce meaningful results that can be used to infer

[1]AR: augmented reality

the behavior of the vehicle in the real world. Finally, as any element can be either real or virtual, AR testing offers a smooth transition from simulation to actual testing. For these reasons, the proposed AR framework can be a fundamental testing solution for the validation of advanced automotive software. The contributions we bring along with this paper are so resumed as follows:

- the design of a new framework enabling AR directly on sensor data
- the data fusion methodology that allows the real-time augmentation of LiDAR sensor data
- both qualitative and quantitative experimental results to validate the applicability of the proposed method.

II. Approach and related works

There are various tools for testing advanced automotive software. Software-in-the-Loop (SiL) consists of an execution of the exact software under test in a virtual environment. Any test scenario can be executed without any risk and with a reduced engineering cost. Hardware-in-the-Loop (HiL) involves the exact software and the computer under test. This is more realistic yet the vehicle itself remains virtual and uses an approximate model. Vehicle-in-the-Loop (ViL) allows much more realistic testing than SiL or HiL. This method involves the whole real vehicle under test in an entirely virtual environment. ViL has been widely used for automotive testing over the last decade. Various implementations have been realized and designed for different purposes. A common structure can be observed [4]. ViL always involves the software under test, the computer and the whole vehicle in motion. The state and actions of the actual vehicle are updated in a virtual environment which is then perceived by emulated sensors that replace the actual sensors. While the test happens entirely in the virtual environment, the actual vehicle may be standing on a test bench [5] or driving on an empty road [6]. As the environment remains purely virtual, the testing is limited by the simulator realism, the emulation of the sensors and the diversity of the virtual scenes. Even though ViL is a highly effective tool for developers, the gap between ViL and actual testing is too large for ViL to replace an extensive actual testing in a validation procedure.

More recently, several works made a significant improvement to go beyond ViL and combine a virtual and real test environment. As these methods augment real test scenes with virtual elements, we call them augmented reality. AR offers safe and efficient testing with virtual elements but also rich, dense and realistic environments. Several approaches introduce AR at object level. They represent the virtual elements by their position, speed, status and main characteristics. There is no realistic sensor emulation. Object level AR does not require heavy computation and allows light implementations. The simple representation of AR elements enables to share them on communications between vehicles and with infrastructure. Thanks to this, object level AR can be synchronized with several vehicles under test. It can also be centralized and computed in the infrastructure. In [7], Chen et al. present a unified fusion data format to represent the AR scene at object level. Also, an implementation of object level AR has been realized at the Mcity facility [8], it relies on the V2X communication protocol to introduce and share virtual objects. Another implementation has been developed for the ZalaZONE proving ground, it introduced the concept of Scenario-in-the-Loop [9]. SciL offers a mixed reality framework where virtual objects are introduced on a communication protocol based on 5G. Also some sensors are emulated on the vehicle under test with a simple range sensor model. However perception is a critical part of advanced automotive software and it can not be challenged by object level AR. A realistic sensor emulation is needed. AR has to be introduced directly in the sensor detection, in the messages as they come out of the sensor drivers, and only then perception can be tested in AR. Moreover scenes with partial perception, i.e. when something occludes part of the scene, cannot be recreated with object level AR but can be accurately generated with sensor level AR. Also, object level AR cannot integrate virtual background elements such as for example infrastructure, trees or walls because they cannot be represented by objects in the main communication protocols. In the meanwhile, any foreground or background element can be real or virtual with sensor level AR. But sensor level AR requires rather heavy computations to be executed in real time, this is a hard constraint on implementation. There are very few examples of AR implementations at sensor level. Hildebrandt et al. introduced a sensor level AR with the concept of World-in-the-Loop [10]. WiL has been implemented for the monocular camera of a drone. However WiL runs a decoupled simulation of the vehicle under test which only exchanges perception with the real one. It requires to filter and adjust the perception to compensate the differences between the virtual and real environment.

In this paper, we present a new framework to realize AR at a sensor level. Unlike WiL, it bonds the virtual vehicle to the actual one's status. This strong connection enables realistic sensor emulation, without filtering. Moreover, as the structure is similar to ViL and as any element can be either real or virtual, it provides a unique smooth transition from ViL to AR and from AR to real testing. This offers new opportunities for testing and validation.

Several recent research works present LiDAR data processing techniques. The solutions developed focus on offline augmentation of training datasets for machine learning. In [11], the authors use real LiDAR data recording as background to add annotated and simulated point clouds of virtual objects. In [12], the authors simulate adversarial weather conditions by adding impacts on simulated water sprays to real LiDAR point clouds. Our AR solution differs in that it allows implementing LiDAR data augmentation in real time.

III. Methodology

A. Structure of the system

In this section, we present the proposed framework to introduce AR in the actual sensing of the vehicle. Even if our AR system does much more than ViL, it has the same architecture and several modules in common with most ViL systems. Reusing the ViL topology which is described in [13], our system consists of the four following modules:

- a virtual environment which contains a twin of the experimental vehicle
- a synchronization module which updates the position and state of the virtual twin
- a sensor emulation which generates outputs from the virtual sensors and integrates them in the actual sensors' outputs
- a visualization which helps testers to understand the AR scene.

Fig. 2 proposes a schematic representation of the software framework. The periodic messages of the sensors of the real vehicle give rhythm to the virtual world. So all modules must run in real time, their execution duration must be short compared to the period of the sensors. This is a heavy constraint on the design and implementation of the solution.

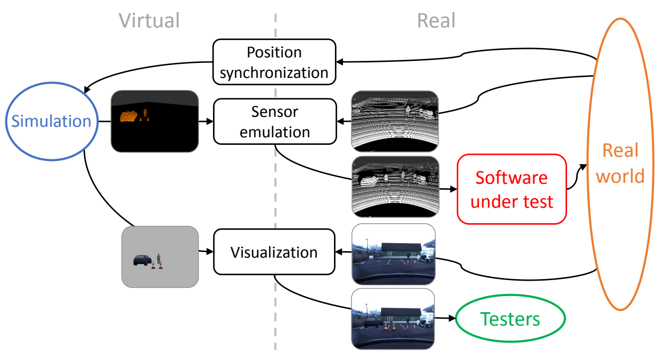

Fig. 2. Structure of our AR framework.

1) Virtual environment: We generate a virtual environment which is anchored to a real world position with a reference in GPS coordinates. Then, the virtual environment contains only a virtual twin of the vehicle under test and the virtual elements that we want to add in augmented reality. The virtual twin must be an accurate representation of the real vehicle. There is no restriction on the virtual elements of the test scene. The scene can be as complex as required by the test and include any type of object, the only limits are the ones of the simulator. Apart of the virtual vehicle and the test elements, the virtual environment is empty. Our method does not need a background, a ground plane and any representation of the actual test site. This makes this method easy to deploy in a new place.

2) Synchronization: The absolute position of the vehicle under test must be constantly estimated by an accurate localization system. The estimated position is used to set the position of the virtual twin of the vehicle under test in the virtual environment. This straightforward synchronization gives a great flexibility. The AR system can be deployed without any installation.

3) Sensor emulation: The virtual twin of the vehicle is equipped with a set of sensors that mimics the sensors of the actual vehicle. An accurate, realistic and real-time emulation of the sensors is needed. Although the framework is generic, we focus here on LiDAR sensors. The emulated LiDARs must return the detection of the virtual objects under a point cloud format. The point clouds are then merged with those returned by each corresponding actual sensor. The merge process is a key component of the proposed AR framework: it must be real time despite the amount of data to process; it must consider a realistic sensor model; it must reproduce all occlusions between real and virtual world. For each sensor, the merge produces a new point cloud that represents the AR perception. It can then be sent to the software of the vehicle under test in place of the actual sensor's point cloud. Thanks to this, the use of AR is seamless for the software under test.

4) Visualization: The virtual twin of the vehicle is also equipped with a set of cameras that mimics the ones of the actual vehicle. Thanks to the simulator, the virtual cameras return images of the virtual objects. These images are then merged with those of each corresponding camera. For each camera, this produces a new image that represents the AR perception. It provides the testers with a convenient insight of the AR scene. If using a photo-realistic simulator and a realistic image merge function, this visualization can be used as AR for perception with cameras. However, a simulator with approximate graphics and a simple merge procedure suffice for the purpose of visualization.

B. Integration in LiDAR perception

The integration of the outputs of the virtual LiDARs in the perception of the actual LiDARs is one key component of our contribution. It is what makes the difference between ViL and AR. Let us consider one of the actual LiDARs and its corresponding virtual sensor. Both periodically produce point clouds of their detection but they are generally not synchronized. The virtual sensor has the same position on the vehicle that its corresponding actual sensor. The position of the virtual vehicle is updated at a high frequency thanks to the localization system. Therefore we can assume that the position of the actual and virtual sensor match each other. Most simulators may not be able to achieve the emulation of high resolution 3D LiDARs in real time. Then it may be needed to use a virtual LiDAR with a different resolution, not as good as the actual sensor. Therefore the requirements for the merge function are that it should merge two point clouds, sensed from the same position but that are not synchronized and have different resolutions. There is a hard real time constraint on the merge function, its execution duration must be short compared to the period of sensor. Point clouds consist of large amount of data, about a million points to process every second for a single dense 3D sensor. For sake of real time, our merge function is designed to be executed in CUDA programming [14] and parallelized over the numerous points with as few memory transfer as possible. Let $\mathscr{P}_s$, $\mathscr{P}_v$ and $\mathscr{P}_f$ be the actual, virtual and fused point clouds. Let p denote a point whose Cartesian coordinates are (x,y,z) and spherical coordinates are (r,θ,ϕ). For every point cloud provided by the virtual sensor, the system computes $\mathscr{R}_v$ the array of equivalent spherical coordinates.

$$\mathscr{R}_v = \{(r_v, \theta_v, \phi_v), \forall p_v \in \mathscr{P}_v\} \tag{1}$$

Let R be the function that returns the range coordinate of a point. Then we can define the function R_v that returns the range of the point in $\mathscr{R}_v$ which is the closest to the given angular coordinates (θ, ϕ):

$$R_v(\theta,\phi) = R\left(\underset{(r_v,\theta_v,\phi_v)\in\mathscr{R}_v}{\arg\min}\left\|\begin{bmatrix}\theta_v-\theta\\ \phi_v-\phi\end{bmatrix}\right\|\right) \tag{2}$$

The frequency of the virtual sensor must be set such that $\mathscr{R}_v$ is updated faster than the actual sensor's frequency. Then, for every point cloud provided by the actual sensor, we compute the fused point cloud, first in spherical coordinates, $\mathscr{R}_f$:

$$\mathscr{R}_f = \{(\min(r, R_v(\theta,\phi)), \theta, \phi), \forall p \in \mathscr{P}_s\} \tag{3}$$

(3) defines a fused point cloud that considers all occlusions between real and virtual objects. This point cloud represents what would have been detected by a LiDAR sensor in the AR scene. (3) enables virtual objects to correctly occlude actual objects even though the point cloud does not have the same density. It also preserves all points from the actual point cloud, as long as they are not occluded. This is an exact model to represent all occlusions, real elements occluding virtual ones and virtual elements occluding real ones. Also actual weather conditions that generate occlusions in the point cloud (like fog, rain or snow) will affect equally virtual and real elements. Furthermore the model guarantees that real data are unchanged. It also maintains the exact format of the real sensor data, number of points and resolution. Finally $\mathscr{R}_f$ is converted in Cartesian coordinates to get $\mathscr{P}_f$ and $\mathscr{P}_f$ is sent to the software of the vehicle under test instead of $\mathscr{P}_s$. Then a simple rerouting of the sensor message is enough to make AR happen, it is seamless for the software under test.

$$\mathscr{P}_f = \{(x,y,z), \forall p \in \mathscr{R}_f\} \tag{4}$$

Fig. 3 illustrates the principle of sampling and occlusion introduced by (2) and (3). (1) and (2) are implemented as two sequential operations in a single CUDA kernel which is executed at every point cloud received from the virtual LiDAR. $\mathscr{P}_s$ is stored as a sorted array so that R_v is straightforward to compute. (3) and (4) are implemented together in a second CUDA kernel which is executed at every point cloud received from the actual LiDAR. The table of ranges corresponding to $R_v(\theta,\phi)$ is the only element stored in memory. There is no other memory transfer. This efficient implementation in CUDA makes this application feasible in real time.

IV. Implementation and Experimental Validation

In this section, we present the implementation of our AR framework and several experimental results. The main goal is to prove that our AR generated scenes correctly represent real test scenarios. To do so, we provide an experimental analysis with 3 criteria of representability. First, a real-time execution of the data augmentation is needed to prevent any delay in the perception. Second, the point cloud augmentation must preserve the integrity of the objects' shapes. Third, the position synchronization together with the point cloud augmentation must guarantee the consistent motion of the virtual objects with respect to the vehicle under test.

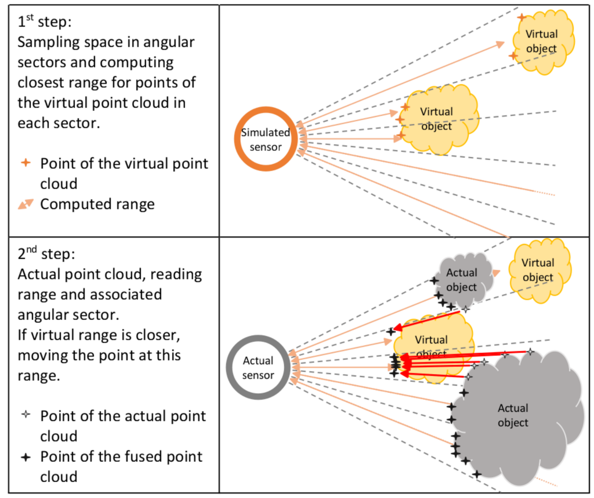

Fig. 3. Illustration of the principle of point clouds fusion used in this work

A. Implementation

1) Experimental set-up: The augmented reality system described in this paper has been deployed on a Renault Zoe. The vehicle is equipped with a Velodyne HDL64 on its roof, 3 Ibeo Lux LiDARs on the front and one on the back, a monocular camera in front, a Xsens IMU and a 2 antennas RTK GPS Spectra SP90. This vehicle is also equipped with a drive-by-wire system such that it can be programmed for autonomous drive or any type of driver assistance. All the software used for this work have been implemented on a computer on board of the car, equipped with a NVIDIA Titan X GPU. In our test scenarios, we used several elements including pedestrians, cars and construction cones that have been brought either to the real environment or to the virtual one. All LiDARs have been configured for AR perception. The monocular camera of the vehicle is used to generate the visualization of the AR scene. We use Gazebo simulator [15] to generate the virtual environment.

2) Real time constraint: The duration of the data augmentation must be short compared to the sensor measurement period such that the introduction of AR in the sensor messages does not cause any noticeable delay. Experiments showed that an iteration of the point cloud augmentation is executed in average in 10ms for the dense point cloud of the Velodyne HDL64 (more than 150000 points) while the sensor period is 100ms. A point cloud of an Ibeo Lux is processed in only 0.6ms while the period of this sensor is 40ms. Thus the real time constraint is satisfied.

B. Integrity of objects' shapes

1) Raw point cloud: Fig. 4 shows a point cloud from an AR scene which contains a set of real objects and, next to it, their virtual counterpart. The point cloud fusion process does not alter the perception of real objects and maintains the resolution of the sensors' point clouds. However, for sake of a real time simulation, we had to reduce the resolution of the virtual sensor. As a consequence, the resolution of the angular sampling has been set to 0.5°. Because of this, virtual

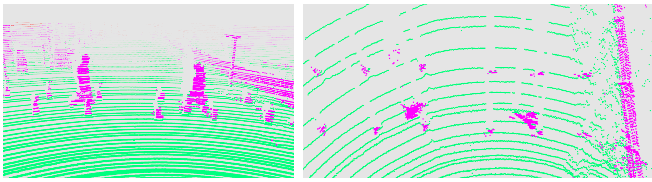

Fig. 4. Example of a point cloud obtained by AR, seen from the vehicle prospective and from above. The scene consists of an actual pedestrian and 6 actual construction cones on the left plus a virtual pedestrian and the same set of cones on the right. Pedestrians are approximately 5m away from the vehicle. The points are displayed in green for those of the ground and magenta for the other points.

Fig. 5. On the left is an example of the visualization of an AR scene, generated from the front camera of the vehicle. On the right is the corresponding occupancy grid, seen from above. The free space is displayed in black and the occupied space in blue.

objects tend to appear like squares at longer range. Still this effect is limited and in our case it can only be observed for objects farther than 10m. Then the square effect produces squares of about 4cm at 5m range and 18cm at 20m range. The observation of the point clouds during several tests with various types of obstacles showed that this is enough to distinguish most of the details of the virtual objects.

2) Perception after filtering: Point clouds are generally used after being processed with various perception filtering and fusion techniques and only then the perception output is provided to the navigation software under another format. In this work, we used together the GEOG-Estimator [16] and the CMCDOT [17] to produce a dynamic occupancy grid. The dynamic occupancy grid is a probabilistic representation of the 2D occupancy by obstacles around the vehicle under test. It also provides the speed of the objects in motion. The dynamic occupancy grid is the format that we use to feed several navigation and driver assistance software. In various AR scenes, we compared the representation of virtual objects and their real equivalent in dynamic occupancy grids to estimate the representability of AR generated objects after filtering. In Fig. 5, we can observe the visualization and the corresponding occupancy grid for a static scene with various objects, real and virtual. There is no noticeable difference between the virtual and real objects in the occupancy grid. This suggests that the irregularities of the shape of the virtual objects, introduced by point cloud sampling, are filtered out. Fig. 6 shows an example of a predicted occupancy grid which is the occupancy grid over time, obtained by propagating occupancy according to the estimated speed. The predicted occupancy grids that we obtained in various experiments showed that there is no significant difference between virtual and real obstacles. This suggests that the shape and the motion of a virtual object seen as a cluster of cells of the dynamic occupancy grid are consistent and representative of a real object.

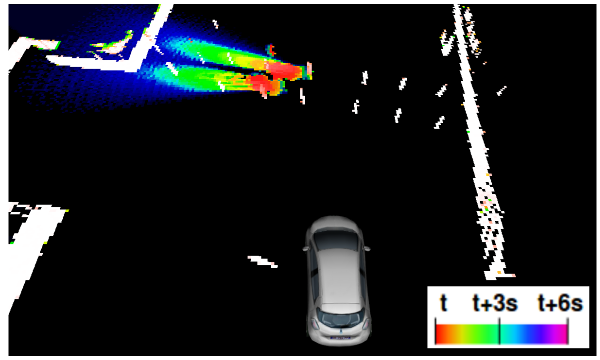

Fig. 6. Example of the predicted occupancy grid for a scene with a virtual and an actual pedestrian walking side by side at the same speed. Several construction cones are placed around the pedestrians, half of them are virtual. The color of each cell of the grid denotes at which time an occupancy is predicted in it. Cells with static detected occupancy are white and empty cells remain black.

C. Relative position and speed consistency

A fast and accurate localization system is required in order to guarantee the synchronization of the virtual twin of the vehicle. For this, we rely on a 2-antennas DGPS which grants accuracy in position and orientation. This is combined with an IMU through an extended Kalman filter so that the localization is also fast and reactive. This localization system is estimated to be few cm and about 1° accurate at a frequency of 100Hz. Therefore the same accuracy is expected in the relative position of the virtual objects in AR scenes. We now prove it with experimental results. For this, we chose to study the result of a time to collision prediction. The TTC[2] is highly depending on the relative position of the obstacles with respect to the vehicle over time. Therefore we assume the relative position and speed of the obstacle are at least as accurate as the TTC prediction. So we consider the TTC prediction accuracy as an indicator of the consistency of AR objects' motion.

On a straight portion of road, we run a total of 30 experiments that represent the collision with a pedestrian who is either standing in the middle of the road or crossing it. The vehicle under test is manually driven until the collision at various speeds between 10 and 25km/h. For a half of the experiments we use a real object, a dummy hanging on a Tyrolean traverse. For the other half, we use a pedestrian introduced via AR. The TTC is estimated by propagating the ego-motion and the dynamic occupancy grid up to a horizon of 5.5s. Because of experimental constraints, the motion of our dummy is limited to few seconds. So we focus our comparison on the last 3 seconds before collision. Fig. 7 shows the graphs of the TTC predictions. With both the real

[2]TTC: time to collision

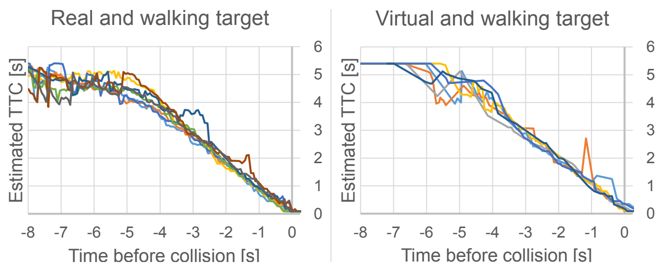

Fig. 7. Estimated time to collision during the few seconds before collision over several experiments. The real target is a dummy hanging on a Tyrolean traverse in order to mimic a pedestrian crossing the street. The virtual target is a pedestrian introduced through AR.

and virtual target, we observe the same expected decrease at a rate of about -1, from the horizon of the computation to the collision. Random experimental variations can be observed but the global behavior remains constant and does not depend of the target being real or virtual. Fig. 8 shows the statistic distribution of TTC predictions over series of experiments, at different instants before collision. The disparities among a series with the real target is wider than the difference with the virtual counterpart. It proves that the random experimental variations have more effect than the use of AR. Then we can state that the use of AR does not have a significant impact on the accuracy of TTC prediction. Therefore we infer that the relative position of the virtual targets with respect to the vehicle is consistent over time and accurate enough so that it does not disturb our perception system.

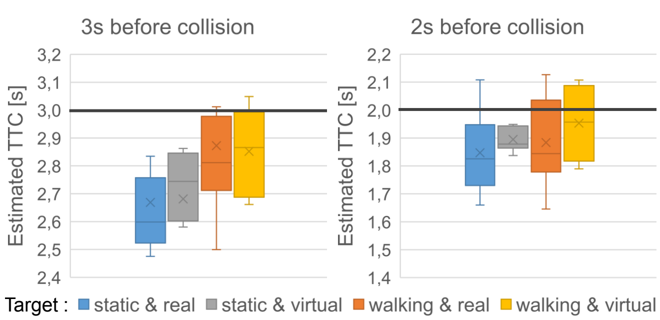

Fig. 8. Distribution of the estimated time to collision, at 3s and 2s before collision. The average, the median, the quartiles and the extrema are displayed for each series.

V. Conclusions

We presented a new augmented reality framework and its key components. AR provides new test opportunities including dense, safe, cheap and reconfigurable test environments but also numerous test actors with rich behaviors that can be collided such as crowds of virtual pedestrians. Relying on experimental results, we showed that this AR method preserves real data and augment it with virtual data that are consistent and representative of real test conditions. Based on these results, we now use this AR implementation to challenge various navigation software. A collision mitigation driver assistant based on the cost map of [18] is being developed and tested in AR on our prototype. Future developments include the ability to augment other sensors, the design of test actors reacting to the vehicle and the contextual automated generation of critical test scenarios [19].

Acknowledgment

This work has been developed within the scope of an ”Institut de Recherche Technologique NanoElec” project, founded by the French program ”Investissement d'Avenirs” ANR-10-AIRT-05. A part of this work has been supported by the PRISSMA project, co-financed by the French Grand Défi on Trustworthy AI for Industry. The authors wish to thank N. Turro and L. Rummelhard for their collaboration and continuous support.

References

[1] R. Schram, “Euro ncap's first step to assess automated driving systems,” in *Euro NCAP Working Group on Automated Driving*, 2019.
[2] C. Grover, M. Avery, and J. Silberling, “3D car target for future vehicle testing,” in *Intern. Technical Conf. on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.
[3] N. Rajabli, F. Flammini, R. Nardone, and V. Vittorini, “Software verification and validation of safe autonomous cars: A systematic literature review,” *IEEE Access*, vol. 9, pp. 4797–4819, 2021.
[4] T. Laverghetta, J. F. Leathrum, and N. D. Gonda, “Integrating virtual and augmented reality based testing into the development of autonomous vehicles,” in *MODSIM 2018*, 2018.
[5] A. Albers and T. Düser, “Implementation of a vehicle-in-the-loop development and validation platform,” in *FISITA 2010 World Automotive Congress*, Budapest, Hungary, 2010.
[6] T. Bokc, M. Maurer, and G. Farber, “Validation of the vehicle in the loop (vil); a milestone for the simulation of driver assistance systems,” in *2007 IEEE Intelligent Vehicles Symposium*, 2007, pp. 612–617.
[7] Y. Chen, S. Chen, T. Xiao, S. Zhang, Q. Hou, and N. Zheng, “Mixed test environment-based vehicle-in-the-loop validation - a new testing approach for autonomous vehicles,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1283–1289.
[8] Y. Feng, C. Yu, S. Xu, H. X. Liu, and H. Peng, “An augmented reality environment for connected and automated vehicle testing and evaluation,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.
[9] Z. Szalay, “Next generation x-in-the-loop validation methodology for automated vehicle systems,” *IEEE Access*, 2021.
[10] C. Hildebrandt and S. Elbaum, “World-in-the-loop simulation for autonomous systems validation,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 10 912–10 919.
[11] J. Fang, D. Zhou, F. Yan, T. Zhao, F. Zhang, Y. Ma, L. Wang, and R. Yang, “Augmented lidar simulator for autonomous driving,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1931–1938, 2020.
[12] J. R. Vargas Rivero, T. Gerbich, B. Buschardt, and J. Chen, “Data augmentation of automotive lidar point clouds under adverse weather situations,” *Sensors*, vol. 21, no. 13, p. 4503, 2021.
[13] C. Park, S. Chung, and H. Lee, “Vehicle-in-the-loop in global coordinates for advanced driver assistance system,” *Applied Sciences*, 2020.
[14] D. Luebke, “Cuda: Scalable parallel programming for high-performance scientific computing,” in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2008, pp. 836–838.
[15] Osrf, “Why gazebo?” [Online]. Available: http://gazebosim.org/
[16] L. Rummelhard, A. Paigwar, A. Nègre, and C. Laugier, “Ground estimation and point cloud segmentation using spatiotemporal conditional random field,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.
[17] L. Rummelhard, A. Nègre, and C. Laugier, “Conditional monte carlo dense occupancy tracker,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2485–2490.
[18] L. A. Serafim Guardini, A. Spalanzani, C. Laugier, P. Martinet, A.-L. Do, and T. Hermitte, “Employing severity of injury to contextualize complex risk mitigation scenarios,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1839–1845.
[19] J.-B. Horel, C. Laugier, L. Marsso, R. Mateescu, L. Muller, A. Paigwar, A. Renzaglia, and W. Serwe, “Using formal conformance testing to generate scenarios for autonomous vehicles,” in *Design, Automation and Test in Europe-Autonomous Systems Design (DATE/ASD)*. IEEE, 2022.