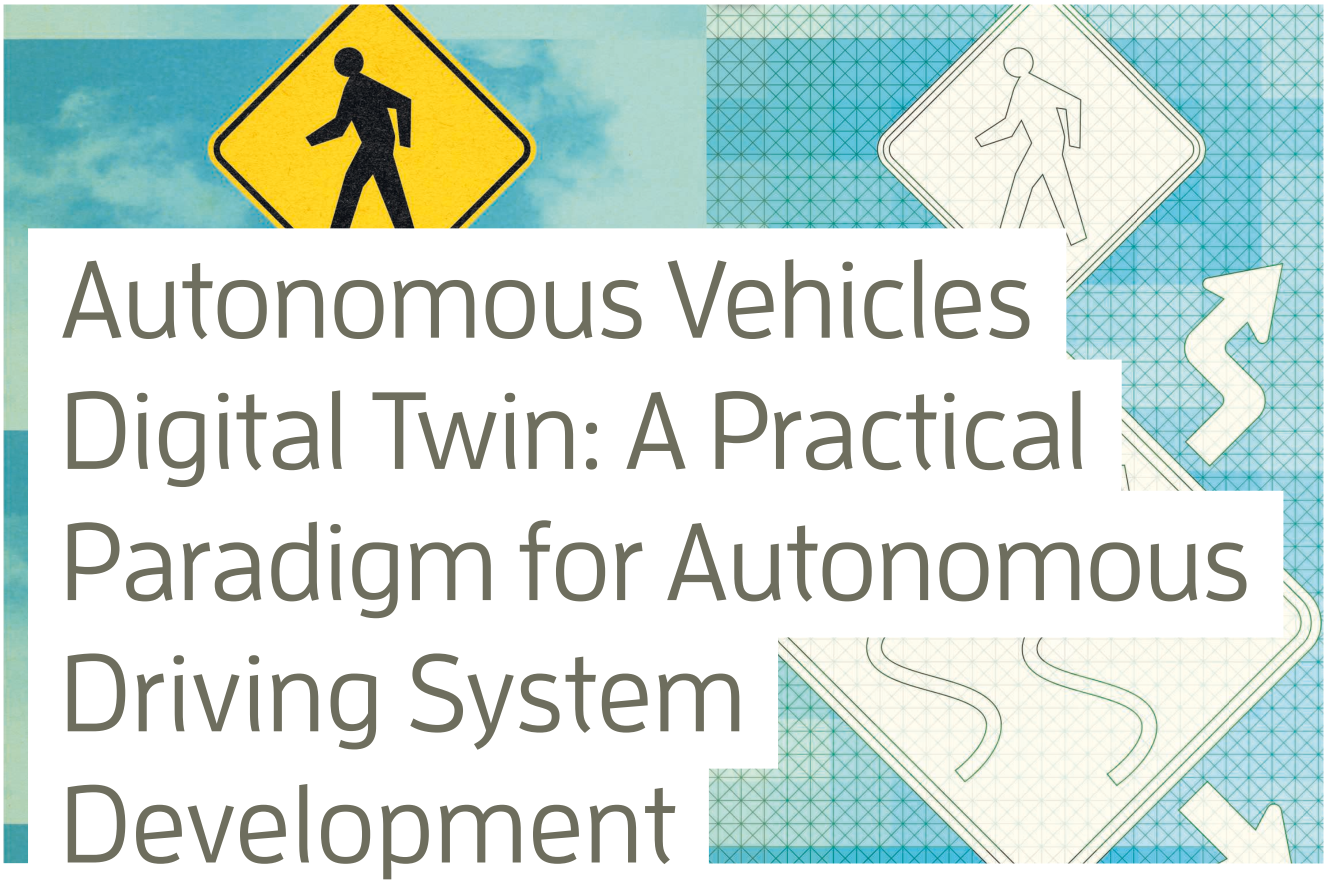

Autonomous Vehicles Digital Twin: A Practical Paradigm for Autonomous Driving System Development

Bo Yu and Chongyu Chen, PerceptIn
Jie Tang, South China University of Technology
Shaoshan Liu, PerceptIn
Jean-Luc Gaudiot, University of California, Irvine

In this article, we share our real-world experiences of digital twin, a practical autonomous driving system development paradigm, which generates an integral, comprehensive, precise, and reliable representation of the physical environment to minimize the need for physical testing.

Over the last decade, autonomous driving (AD) has evolved into a thriving sector, which begins to reshape the transportation system in the aspects of safety and efficiency.[1] With an increasing number of autonomous vehicles (AVs) operating on the road, safety and reliability undoubtedly arise to be the most important concerns of the AV system development. Nonetheless, while the latest literature indicates that AVs have the potential to significantly improve vehicle safety,[2] this safety improvement cannot be achieved until billions of kilometers of accumulative testing under all-weather conditions has been taken by a fleet of AVs.[3] With existing physical testings by running a fleet of AVs and development infrastructures that mainly exploit physical testing data, it will take decades and tens of billions of dollars of investment to achieve the safety goals for AVs.

Digital Object Identifier 10.1109/MC.2022.3159500
Date of current version: 29 August 2022

Fortunately, there is a simulation-based method, dubbed *digital twin*, to significantly accelerate the verification process of AVs while effectively reducing the development costs in a high-fidelity virtual environment. Digital twin requires a development paradigm shift. Instead of relying heavily on physical testing data, the digital twin base approach creates a virtual but precise representation of the physical environment that can simulate a wide spectrum of weather and traffic conditions. Testing AVs in virtual and controllable environments could lead to orders of magnitudes of improvement on development efficiency in terms of time and cost. For instance, to test how AVs can handle heavy snow, we may have to wait for months until snow arrives and then collect physical testing data on the road. With digital twin, we can construct a road and generate a heavy snow scenario, and consequently produce various high quality testing data as needed. Encouragingly, our initial digital twin system deployment that combines both physical testing and digital twin testing has led to significant advantages in cost reduction and efficiency improvement,[4] which still leaves tremendous headroom for further improvement.

In summary, this article makes the following contributions:

- We pinpoint the problems of existing AD simulation methods and how digital twin can effectively address these problems.
- We first generalize three design principles of the AD digital twin system, drawing from our real-world development experience.
- We present an architecture of the AD digital twin system, and the technical details of the building blocks, including ingestion of real-world mapping data, sensor data simulation, as well as synthesis of various traffic participants.

AD SIMULATION

Simulation is not new to the automobile industry, for example, vehicle dynamic simulators[5] have been widely used in the development process, such as steering system development. As for AD software development, simulators have been used heavily to test and verify the decision making module and the path planning module under various conditions by feeding perception data (for example, position and moving states of the ego vehicle and other traffic participants).[6] This approach is lightweight and scalable but lacks a high-fidelity representation of the world, hence leading to two problems: 1) simulation tests are not equivalent to the physical tests that exercise the end-to-end AV software pipeline, which includes perception, localization, decision making, path planning, and vehicle control[7]; and 2) cases related to physical conditions, such as a wide spectrum weather and lighting conditions, cannot be examined.

[TESTING AVs IN VIRTUAL AND CONTROLLABLE ENVIRONMENTS COULD LEAD TO ORDERS OF MAGNITUDES OF IMPROVEMENT ON DEVELOPMENT EFFICIENCY IN TERMS OF TIME AND COST.]

Recently, high-fidelity simulators based on game engines have been developed for testing end-to-end AD software, such as Carla[8] and LGLVS,[9] which use computer graphic (CG) models, rendering algorithms and physical models to produce a high-fidelity environment including background scenes, sensor data, and traffic participants. Unfortunately, the gap between virtual reality and actual reality has not been closed by this effort for several reasons.

First, these simulators only provide maps of virtual cities, in which geographic and physical features of the environment are different from those of real-world road tests. Without a digital twin map that precisely reconstructs the structural of the real-world environment, AD functions related to road geometry and traffic rules (for example, exit or enter highway ramps) can hardly be assessed in the simulators.

Second, the behaviors of moving objects, such as vehicles and pedestrians, are hard-coded and controlled by predetermined animations embedded in the simulators, which is not able to represent the wide spectrum of behaviors and interaction of real traffics. Therefore, challenging scenarios, such as driving at intersections and aggressive driving behaviors (for example, vehicle cut-ins), cannot be faithfully evaluated.

Last, the fidelity of sensor data are low in these simulators, and thus not able to verify the functionalities and reliability of the AV software under test. For instance, when simulating lidar sensors in these simulators, the generated point clouds are approximated by depth maps or ray casting on 3D objects, which do not take reflection and diffusion into account, and hence the distribution of point clouds in simulation is far from the physical counterpart.

In addition, different from the simulation for AV software development, many recent works on vehicle hardware development utilize the term *digital twin*[10] for physical simulation[11] based on physical modeling tools, such as MATLAB and Modelica,[12] to accelerate the development and integration of vehicle hardware.

In our context, the digital twin paradigm should be able to generate an integral, comprehensive, precise, and reliable virtual environment that closely approximate the physical environment, so as to allow AD companies to quickly test and verify their software. To begin with, this demands a virtual but high-fidelity representation of complex systems, including but not limited to, high-fidelity 3D environments through the integration of real-world high-definition (HD) maps, high-fidelity sensor modeling that is able to approximate the behaviors of real-world sensors, and synthesis of traffic participants that are able to interact with sensors and with each other.

PRINCIPLES OF AD DIGITAL TWIN DESIGN

We regard digital twin as a paradigm, rather than a group of techniques, for the AD technology development process, which should include design principles specific to AD applications, and a group of building blocks to compose a high-fidelity AD digital twin. Drawing from our real-world deployment experiences, in this section, we summarize the principles of AD digital twin paradigm:

- *Principle 1: structural twin*: The digital twin should include precise 3D models of the environments, which render the same geographical and geometrical properties as the physical counterparts. Based on the structural twin, physical processes of AVs can be faithfully modeled. For example, with the geometric information of a target environment, realistic lidar point clouds and camera images could be synthesized; with physical road structures and geometric information, navigation and control algorithms can be faithfully verified in the simulator.
- *Principle 2: physical twin*: Physical processes of object movements, collisions, and sensing should be faithfully modeled in the AV simulator. Physical twin properties should correspond to real-world objects properties to support physical simulation. For example, a vehicle dynamics model requires a set of parameters to precisely emulate the vehicle's maneuvers; mass and mesh of objects are necessary for modeling collision; texture and color of objects are for rendering camera images and lidar point clouds.
- *Principle 3: logical twin*: Simulated traffic participants (for example, vehicles and pedestrians) should have a behavior similar to their physical counterparts when interacting with the AV and other objects. This principle is critical for testing the planning and decision module of an AV, especially in scenes of heavy traffic.

ARCHITECTURAL DESIGN OF AD DIGITAL TWIN

Overall architecture

To represent structural, physical, and behavior information in a virtual environment, we have developed our digital twin system based on a game engine,[13] which provides graphics and physics engines for 3D modeling, image rendering, and physical simulation. On top of the game engine, three building blocks are designed to implement the digital twin properties: 1) the 3D digital twin map corresponds to the structural twin, 2) sensor models correspond to the physical twin, and 3) the traffic controller corresponds to the logical twin.

The architecture of our AD digital twin is illustrated in Figure 1. To preserve the structural information, the digital twin map is constructed offline based on the AV's HD maps.[14] In the digital twin map, CG models with physical properties (for example, material, and color) are used to represent 3D objects. The traffic controller updates the states of traffic participants (for example, cars and trucks) in each simulation cycle, including kinematic states, location, and so on. To construct realistic traffic flows, the digital twin system uses real data recorded by AVs to compose high-fidelity scenarios, a snippet of recorded data containing events or

AV behaviors we want to test, such as turning left at an intersection. To react to any updates in AD software, the traffic controller uses kinematic model-based intelligent agents to update traffic states rather than simply replaying recorded data, which could avoid unrealistic interactions, such as a rear-end crash with acceleration into the AV. Based on the CG model and the states of traffics, the sensor model uses graphic rendering techniques to emulate the physical sensing process of cameras and lidars, which generates sensor frames in line with physical counterparts in terms of data structure and throughput for the AD software. The AD software generates control commands that drive the AV to move according to the vehicle dynamic model in the simulator.

Digital twin map design

The digital twin map maintains the structural and physical properties of the real scene. To be compatible with rendering pipeline and physical simulation, we use CG models to represent the digital twin map.

Creating a large-scale CG-based digital twin map is challenging, as manually creating 3D CG models is the prevalent practice in the modern game development process, but the cost of creating a large-scale 3D scene is prohibitively high. Fortunately, AV operations heavily rely on HD maps, which are comprehensive and precise representations of the road environments. Generally, a HD map consists of two layers: 1) a traffic related marking map, which is typically used for autonomous navigation, contains detailed positional and category information of traffic objects, such as lanes, road boundaries, traffic signs, and so on; and 2) a point cloud map used for localization, containing the precise 3D structures of the environment. Although the point cloud map provides structural information, reconstructing the mesh and texture information from cloud point is still an open problem.[15]

We use a hybrid approach that combines the merits of HD map and

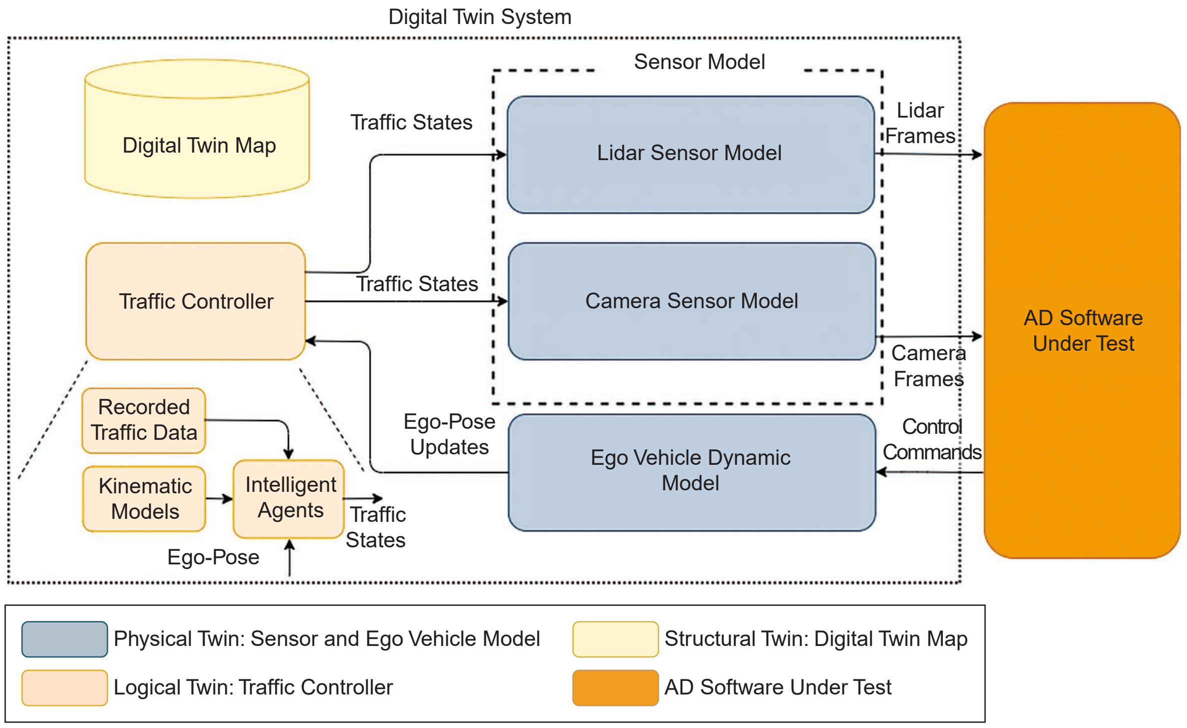

FIGURE 1. The architecture of the digital twin simulator for AD. The digital twin map reflects the structural properties of static environment. The sensor model generates high-fidelity sensor streams. The traffic controller reflects realistic behavior of moving objects. The ego vehicle dynamic model emulates the behavior of an AV under control commands.

CG models to construct the 3D twin map. First, we integrate the HD maps, from which objects' geometrical and structural information are extracted. Then we develop a domain-specific object library containing CG model of traffic facilities and markings. As the number of types of traffic facilities and markings is limited, the cost of building and maintaining the domain-specific object library is manageable. Specifically, the domain-specific object library contains three kinds of CG models: 1) traffic related markings, including road surface, lane markings, crosswalks, and so on; 2) traffic related facilities, such as traffic signs and traffic lights; and 3) other roadside objects, such as poles, trees, walls, and buildings.

We have developed and verified an effective method to generate 3D twin maps, as shown in Figure 2, which consists of two stages: in the first stage we construct the 3D road surface models from the traffic marking map mentioned earlier; in the second stage, we map point clouds of various objects above the road surface to the corresponding 3D models from the domain-specific object library.

In detail, we use the positions of lanes and road boundaries from the traffic marking map to fit planes for representing road surfaces. Mesh structure of the road surfaces can then be obtained by the traditional triangulation on the plane method.[16] Then textures and materials of road surfaces and road markings (for example, lanes and crosswalks) are attached to the generated meshes.

We extract the category and pose of objects above the road surface from the point cloud maps, with which corresponding 3D models from the domain-specific object library can be placed onto their locations on the road surfaces. To establish correspondence between point clouds and CG models in the domain-specific object library, we first apply a semantic

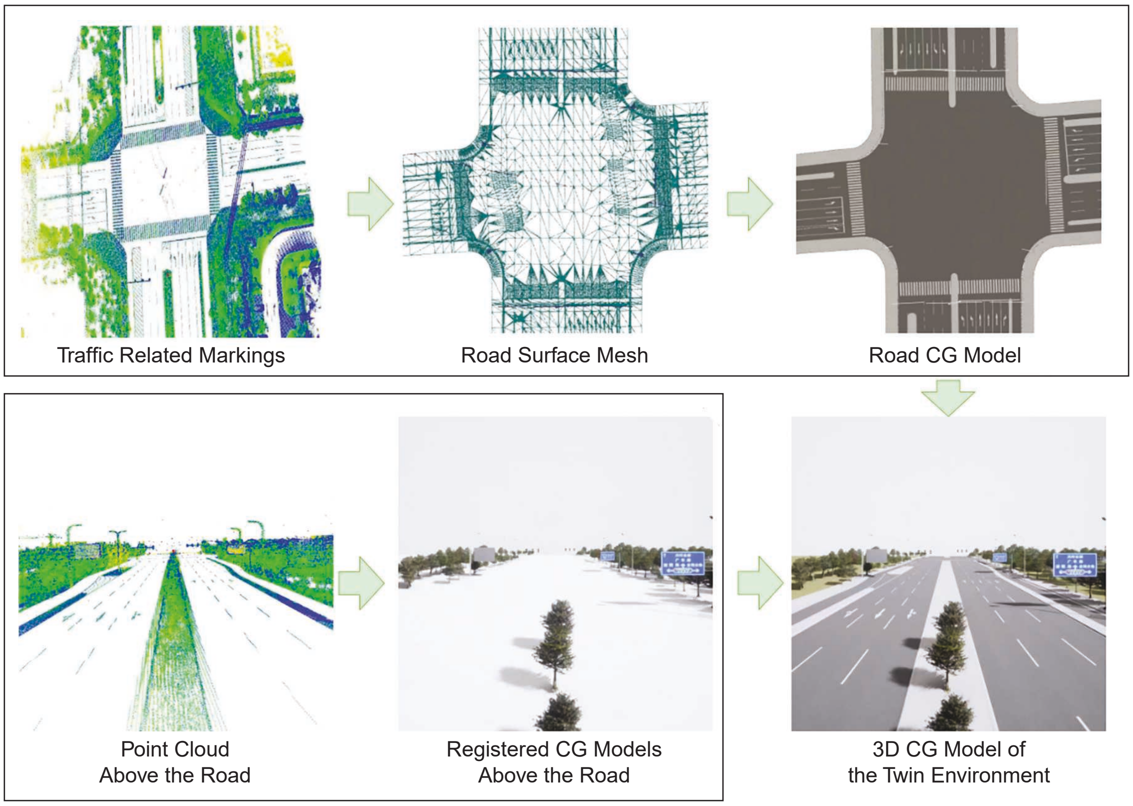
Traffic Related Markings
Road Surface Mesh
Road CG Model
Point Cloud Above the Road
Registered CG Models Above the Road
3D CG Model of the Twin Environment

FIGURE 2. The process of generating 3D twin maps from traffic markings and point cloud maps.

segmentation neural network on the point cloud map to classify point clouds. Then, we identify and remove the point clouds of the road surface using the progressive morphological filter.[17] After that, we apply the Euclidean clustering algorithm of point cloud library (PCL)[18] on the point clouds to obtain the point clouds of each instance. To register CG models in the digital twin map, we first generate the point clouds of the 3D CG models by sampling points on the mesh of CG models using Open3D.[19] Then, we apply the iterative closest point algorithm to align the point clouds of the CG model with the point cloud of each instance,[18] which completes the registration of the CG models in the 3D twin map.

The 3D digital twin map is in the standard format of 3D CG models, which can be easily imported into game engines. With 3D CG models, red, green, blue (RGB) images can be rendered by using the rendering pipeline provided in the game engine.[20] Furthermore, with the mesh and material properties of the CG model, lidar point clouds can be generated with ray tracing–based rendering techniques.[21]

Simulating physical sensors

Cameras and lidars are the AV's main sensors for perception and localization. As our simulation system is based on CG models and rendering pipeline, synthesizing the RGB images is straightforward. However, rendering realistic lidar point clouds is challenging, since point clouds contain not only structural information but also material-related properties, that is, intensity.

High-fidelity AV simulators[9] utilize depth maps or ray casting[8] to synthesize lidar point clouds. With a lidar sensor's location and internal configurations, pixels in 2D images can be transformed to 3D point clouds. Ray casting can model the point clouds as the 3D locations where lidar arrays projected from sensor nodes first hit some objects' surface mesh. Both of the methods essentially model point clouds as the result of intersections between lidar rays and meshes of 3D models, therefore can only produce structural information.

We employ a material-aware approach to synthesize point clouds, which renders intensities of point cloud based on objects' material information by ray tracing.[21] We have developed a rendering model parameterized by material's light-related properties to compute the energy of reflection, diffusion, and transmission of lidar rays given the energy of the incoming ray on the objects' surface. Only the rays with energy above a threshold continue to transmit. Unlike ray casting that stops ray transmission when hitting a surface, ray tracing will recursively apply the rendering model to transmit lidar rays until the energy of rays falls below the threshold. Intensities can be derived based on the energy of received rays. The effectiveness of the proposed method is demonstrated in Figure 3. It shows that the intensity distribution of the point clouds synthesized by our method is very close to that of a real object.

Synthesis of traffic participants

A typical AV generates gigabytes of traffic flow data that contains time series of moving object states and the ego vehicle states (for example, pose, type, contour, and velocity) per day.[22] Our digital twin system can replay the recorded data in the simulation environment by simply placing an object on the digital twin map at each timestamp. Poses of an object in the digital twin map are obtained by transforming the object's pose from the real-world coordinate to the game engine's coordinate. With recorded real data and real scenarios that are extracted from recorded data, the digital twin system is able to closely examine the AV's behavior in real-world traffic environments, which is invaluable for identifying corner cases and debugging AV systems.

However, simply replaying the recorded traffic flows does not provide the necessary flexibility in modeling the behaviors of traffic participants. In the case of a software update to the AV, the AV's behavior will deviate from that in the recordings, which also impact the behaviors of other traffic participants. For example, if the AV changes its behavior from lane

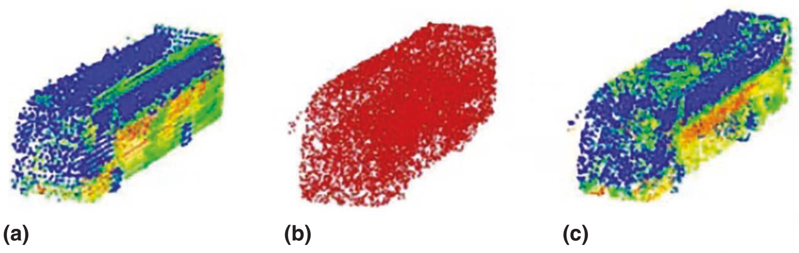

FIGURE 3. The lidar point clouds of a bus. (a) Real data. (b) Synthesized by ray casting. (c) Synthesized by our material-aware approach. Color represents intensity.

keeping to lane changing after a software update in the planning module, the behaviors of other vehicles behind the AV will also be affected by the AV's algorithm change.

To address this problem, instead of simply replaying the recordings, we treat each traffic participant as an intelligent agent in which we can apply suitable kinematic models and behavioral planning algorithms. Hence, each traffic participant can dynamically react to the changing environment. In our design, dubbed intelligent replay, the digital twin system reuses the routing information from the recordings but enables behavioral planning for each traffic participant, allowing each traffic participant to intelligently interact with the AV under test, and thus effectively stress-test the newly developed algorithms.

EVALUATION

We employ a controller-agent architecture to implement the digital twin system in Figure 1, in which the digital twin map is built offline based on the process mentioned earlier; the traffic controller, the sensor model, and the ego vehicle dynamic model form a loop-closed pipeline with the AD system. The traffic controller drives the simulation process. In each simulation cycle, the traffic controller update traffic states according to realistic traffic recordings and intelligent replay mentioned earlier; then the sensor model does the heavy lifting computation, which renders images and lidar frames.

As digital twin maps are built from HD maps based on the process mentioned earlier, the positional precision of traffic markings and facilities is in line with that of the HD maps, which is with an accuracy of 10 cm. Regarding compute resource allocation, we implement the digital twin system along with AV software on AWS cloud servers, a g5.4×large (16 cores vCPUs and a Nvidia A10 GPU) instance and a g4dn.4×large (16 cores vCPUs and a Nvidia Tesla T4 GPU) instance.[23] The computationally intensive algorithms in the system, that is, the rendering pipeline that synthesizes the lidar and camera data and the AV's perception algorithm, is allocated to the GPU. The rest of the modules are executed using CPU resources. In our evaluation, the digital twin system renders a 64-line lidar and two 1080p cameras. The frame rate of rendering lidar is 10 Hz, which is in line with the physical device. The frame rate of cameras is 13 Hz on the g4dn.4×large and 30 Hz on the g5.4×large. The cost of physical tests is about US$180/hour[4]; the cost of the cloud digital twin–based development system is US$2.2/hour on g4dn.4×large and US$3.2/hour on g5.4×large. In the digital twin virtual environment, the vehicle under simulation can run at the same speed as a physical AV in the real world. With the same operation budget, the cumulative driving miles of an AV in the virtual environment can be two orders of magnitudes more efficient compared to the physical counterpart. Further optimization on the compute system can further improve the cost and efficiency advantages of the digital twin paradigm.

ABOUT THE AUTHORS

BO YU is the CTO of PerceptIn, Fremont, California, 94539, USA. His research interests include algorithm and systems for robotics and autonomous vehicles. Yu received a Ph.D. from Tsinghua University. He is a Senior Member of IEEE. Contact him at bo.yu@perceptin.io.

CHONGYU CHEN is a software engineer at PerceptIn, Fremont, California, 94539, USA. His research interests include computer graphics, robotics, and computer vision. Chen received a master's degree in computer and information technology from the University of Pennsylvania. Contact him at mikeccy@gmail.com.

JIE TANG is an associate professor in the School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510000, China. Her research interests include computing systems for autonomous machines. Tang received a Ph.D. in computer science from the Beijing Institute of Technology. She is the corresponding author of this article and a Senior Member of IEEE. Contact her at cstangjie@scut.edu.cn.

SHAOSHAN LIU is the founder and CTO of PerceptIn, Fremont, California, 94539, USA. His research interests include autonomous driving technologies and robotics. Liu received a Ph.D. in computer engineering from the University of California, Irvine. He is a Senior Member of IEEE. Contact him at shaoshan.liu@perceptin.io.

JEAN-LUC GAUDIOT is a professor in the Department of Electrical Engineering and Computer Science, University of California, Irvine, California, 92697, USA. His research interests include multithreaded architectures, fault-tolerant multiprocessors, and implementation of reconfigurable architectures. Gaudiot received a Ph.D. in computer science from the University of California, Los Angeles. He is a Fellow of IEEE, the American Association for the Advancement of Science, and president of the IEEE Computer Society. Contact him at gaudiot@uci.edu.

Restricted by physical testing constraints, the current AV development process is slow and costly, and it has become the bottleneck of the AV industry.[3] In this article, we introduce the AV digital twin, a practical and effective paradigm for efficiently developing AD system. Unlike traditional AD simulation methods, which mostly focus on one aspect of AV development, the digital twin paradigm generates an integral, comprehensive, precise, and reliable representation of the physical environment, thus delivering a fast development iteration time at a low cost.

Drawing from our real-world deployment experiences, we summarize three design principles of AD digital twin (structural twin, physical twin, and logical twin), and present a digital twin system constructed based on the design principles and delve into the technical details of the core components of the digital twin system, including digital twin maps, physical sensor simulation, and synthesis of traffic participants. The digital twin paradigm allows AV companies to stress-test various AV software components on different roads with a wide spectrum of weather conditions, to generate high-fidelity sensor data for deep learning model training, and to verify the reliability of the end-to-end AV system when interacting with all kinds of traffic participants.

Extensibility is the imminent next step for the digital twin paradigm. By incorporating different domain-specific simulators into the digital twin system, we can greatly accelerate the development process of different domains within the intelligent transportation ecosystem. To accelerate the development of vehicle to everything (V2X) system in the context of infrastructure-vehicle cooperative AD, we plan to incorporate V2X network simulators to our digital twin system. Similarly, to advance compute system designs for AD, we plan to incorporate design automation tools and compute system simulators to our digital twin system. With the proposed paradigm, the digital twin system will become the core engine that drives the whole AV industry forward. C

REFERENCES

1. S. Liu, L. Li, J. Tang, S. Wu, and J.-L. Gaudiot, "Creating autonomous vehicle systems," *Synthesis Lectures Comput. Sci.*, vol. 8, no. 2, pp. i–216, 2020, doi: 10.2200/S01036ED1V01Y202007CSL012.
2. M. Blanco, J. Atwood, S. M. Russell, T. Trimble, J. A. McClafferty, and M. A. Perez, "Automated vehicle crash rate comparison using naturalistic data," Virginia Tech Transp. Inst., Blacksburg, VA, USA, Tech. Rep., 2016. [Online]. Available: https://vtechworks.lib.vt.edu/handle/10919/64420
3. N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transp. Res. A, Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016, doi: 10.1016/j.tra.2016.09.010.
4. S. Liu, B. Yu, J. Tang, and Q. Zhu, "Invited: Towards fully intelligent transportation through infrastructure-vehicle cooperative autonomous driving: Challenges and opportunities," in *Proc. 58th Design Autom. Conf.*, 2021, pp. 1323–1326, doi: 10.1109/DAC18074.2021.9586317.
5. CARSIM. Accessed: Jan. 21, 2022. [Online]. Available: https://www.carsim.com/products/carsim/
6. D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.
7. B. Yu, W. Hu, L. Xu, J. Tang, S. Liu, and Y. Zhu, "Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations," in *Proc. 2020 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, pp. 1067–1081, doi: 10.1109/MICRO50266.2020.00089.
8. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, PMLR, 2017, pp. 1–16.
9. G. Rong *et al.*, "LGSVL simulator: A high fidelity simulator for autonomous driving," in *Proc. 2020 IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, pp. 1–6, doi: 10.1109/ITSC45102.2020.9294422.
10. S. Boschert and R. Rosen, "Digital twin—the simulation aspect," in *Mechatronic Futures*, P. Hehenberger and D. Bradley, Eds. Cham, Switzerland: Springer-Verlag, 2016, pp. 59–74.
11. A. Rassõlkin, T. Vaimann, A. Kallaste, and V. Kuts, "Digital twin for propulsion drive of autonomous electric vehicle," in *Proc. 2019 IEEE 60th Int. Sci. Conf. Power Elect. Eng. Riga Tech. Univ. (RTUCON)*, pp. 1–4, doi: 10.1109/RTUCON48111.2019.8982326.
12. Modelica. Accessed: Jan. 21, 2022. [Online]. Available: https://www.openmodelica.org/
13. Unreal Engine. Accessed: Jan. 21, 2022. [Online]. Available: https://www.unrealengine.com/en-US/

14. S. Liu, J. Tang, C. Wang, Q. Wang, and J.-L. Gaudiot, "A unified cloud platform for autonomous driving," *Computer*, vol. 50, no. 12, pp. 42–49, 2017, doi: 10.1109/MC.2017.4451224.
15. A. Badki, O. Gallo, J. Kautz, and P. Sen, "Meshlet priors for 3d mesh reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2849–2858.
16. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Berlin, Germany: Springer Science & Business Media, 2012.
17. K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang, "A progressive morphological filter for removing nonground measurements from airborne lidar data," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 4, pp. 872–882, 2003, doi: 10.1109/TGRS.2003.810682.
18. R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. 2011 IEEE Int. Conf. Robot. Autom.*, pp. 1–4, doi: 10.1109/ICRA.2011.5980567.
19. Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv: 1801.09847*.
20. C. Ioannidis and A. Boutsi, "Multi-threaded rendering for cross-platform 3D visualization based on Vulkan Api," *Int. Archives Photogram., Remote Sens. Spatial Inform. Sci.*, vol. XLIV-4/W1-2020, pp. 57–62, Sep. 2020, doi: 10.5194/isprs-archives-XLIV-4-W1-2020-57-2020.
21. M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. San Mateo, CA, USA: Morgan Kaufmann, 2016.
22. S. Liu, B. Yu, J. Tang, Y. Zhu, and X. Liu, "Communication challenges in infrastructure-vehicle cooperative autonomous driving: A field deployment perspective," *IEEE Commun. Mag.*, early access, May 5, 2022, doi: 10.1109/MWC.005.2100539.
23. Aws ec2. Accessed: Jan. 21, 2022. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

Digital Object Identifier 10.1109/MC.2022.3195368