Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-3 (2023) 277–282

Towards Sim2Real Transfer of Autonomy Algorithms using AutoDRIVE Ecosystem

Chinmay Samak * **Tanmay Samak** * **Venkat Krovi** *

* *Department of Automotive Engineering, Clemson University International Center for Automotive Research, Greenville, SC 29607, USA.* {csamak, tsamak, vkrovi}@clemson.edu

Abstract: The engineering community currently encounters significant challenges in the development of intelligent transportation algorithms that can be transferred from simulation to reality with minimal effort. This can be achieved by robustifying the algorithms using domain adaptation methods and/or by adopting cutting-edge tools that help support this objective seamlessly. This work presents AutoDRIVE, an openly accessible digital twin ecosystem designed to facilitate synergistic development, simulation and deployment of cyber-physical solutions pertaining to autonomous driving technology; and focuses on bridging the autonomy-oriented simulation-to-reality (sim2real) gap using the proposed ecosystem. In this paper, we extensively explore the modeling and simulation aspects of the ecosystem and substantiate its efficacy by demonstrating the successful transition of two candidate autonomy algorithms from simulation to reality to help support our claims: (i) autonomous parking using probabilistic robotics approach; (ii) behavioral cloning using deep imitation learning. The outcomes of these case studies further strengthen the credibility of AutoDRIVE as an invaluable tool for advancing the state-of-the-art in autonomous driving technology.

Keywords: Autonomous Vehicles; Mobile Robots; Digital Twins; Sim2Real; Real2Sim

1. INTRODUCTION

The progression of connected autonomous vehicles (CAVs) necessitates a dual approach of cutting-edge research and comprehensive education. Research endeavors propel immediate advancements in the field, while education plays a pivotal role in equipping the next generation with the necessary skills and knowledge to propel the field even further in the long term. Recently, majority of researchers, educators and students rely on simulation tools and/or a scaled testbeds to to alleviate the monetary, spatial, temporal and safety constraints associated with rapid-prototyping of CAV solutions. In research settings, these platforms accelerate the process of designing experiments, recording datasets as well as re-iteratively prototyping and validating autonomy solutions. In educational contexts, such platforms facilitate the creation of interactive demonstrations, hands-on assignments, projects, and competitions centered around CAV technology.

However, existing platforms catering to the development and validation of connected autonomy solutions have been observed to impose limitations on throughput. Firstly, a significant portion of these platforms lacks the essential integrity necessary to foster hardware-software co-development. Some platforms solely offer software simulators, while others merely provide scaled vehicles for testing autonomy algorithms. Such isolated platforms not only impede the prototyping phase due to compatibility issues but also hinder the validation phase involving the transition from simulation to real-world. Secondly, a majority of these platforms concentrate exclusively on vehicles, neglecting the holistic integration of an intelligent transportation ecosystem encompassing infrastructure, traffic elements, and peer agents. Consequently, their applications remain limited in scope. Thirdly, certain platforms are confined to specific domains or applications, featuring restricted sensing modalities, stringent design requirements, and fixed development frameworks. Such constraints further inhibit the versatility and adaptability of these platforms for broader use cases in connected autonomy research and education.

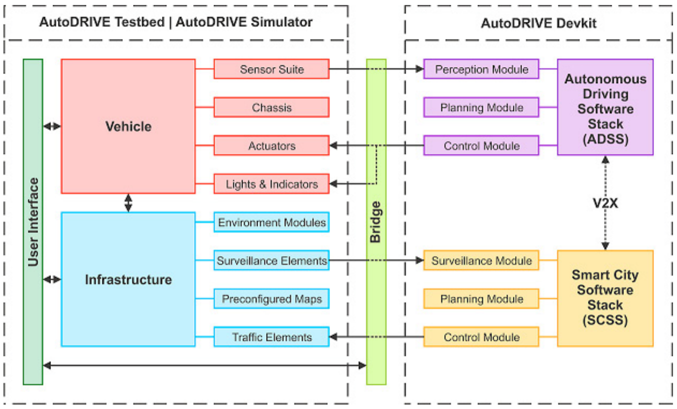


Fig. 1. AutoDRIVE Ecosystem offers of a tightly integrated trio for designing, simulating and deploying autonomy solutions using a unified workflow.

This research presents AutoDRIVE[1] Samak et al. (2023), a publicly accessible ecosystem specifically designed to facilitate the integrated development, simulation, and de-

[1] https://autodrive-ecosystem.github.io

10.1016/j.ifacol.2023.12.037

ployment of cyber-physical solutions pertaining to autonomous driving technology. This seamless workflow is made possible by a tightly integrated trio, consisting of an algorithm development kit for designing autonomy solutions, a software simulator for virtual prototyping and testing them under a variety of conditions and edge-cases, and a hardware testbed for physical deployment and validation (refer Fig. 1). The synergy among these three sub-systems not only enhances the hardware-software co-development of autonomy solutions but also effectively bridges the gap between simulation and reality. This work places particular emphasis on the challenges associated with bridging the sim2real gap for autonomy-oriented applications using the proposed ecosystem. In this context, we delve into the percepto-dynamics modeling and simulation aspects of a scaled vehicle and infrastructure using the proposed ecosystem. Furthermore, we substantiate our claims by showcasing the successful transition of two candidate autonomy algorithms from simulation to reality to help support our claims: (i) autonomous parking using probabilistic robotics approach for mapping, localization, path planning and control; (ii) behavioral cloning using computer vision and end-to-end deep imitation learning.

2. STATE OF THE ART

2.1 Software Simulators

Over the years, open-source community has contributed several simulators for autonomous driving applications. Gazebo Koenig and Howard (2004), natively integrated with Robot Operating System (ROS) Quigley et al. (2009), is commonly used for scaled autonomous robots. TORCS Wymann et al. (2021) has been an early focus in the self-driving community, particularly for manual and autonomous racing. Other notable examples include CARLA Dosovitskiy et al. (2017), AirSim Shah et al. (2018), and Deepdrive Voyage (2021), developed using the Unreal game engine, as well as Apollo GameSim Baidu Inc. (2021), LGSVL Simulator Rong et al. (2020) and AutoRACE Simulator Samak et al. (2021a), created using the Unity game engine.

However, these simulators pose 3 major limitations:

- Firstly, some simulation tools prioritize photorealism over physical accuracy, while others prioritize physical accuracy over graphics quality. In contrast, AutoDRIVE Simulator strikes a balance between physics and graphics, providing a range of configurations to accommodate varying compute capabilities.
- Secondly, the dynamics and perception systems of scaled vehicles and environments differ significantly from their full-scale counterparts. Existing simulation tools do not adequately support scaled ecosystems to their full capacity. Consequently, transitioning from full-scale simulation to scaled real-world deployment necessitates substantial additional effort to re-tune the autonomy algorithms.
- Thirdly, the existing simulators may lack precise real-world counterparts, rendering them unsuitable for "digital twin" applications, involving synthetic data generation, variability testing, reinforcement learning, real2sim and sim2real transfer, among others.

2.2 Hardware Testbeds

In recent times, numerous educational institutions have embarked on the development of scaled autonomous vehicles. Popular examples include the MIT Racecar Karaman et al. (2017), F1TENTH O'Kelly et al. (2019), and AutoRally Goldfain et al. (2019). Additionally, community-driven platforms like HyphaROS RaceCar HyphaROS Workshop (2021) and Donkey Car Donkey Community (2021) have emerged, tailored to specific applications like map-based navigation and vision-aided imitation learning, respectively. Duckietown Paull et al. (2017) is another platform, which employs differential-drive robots instead of kinodynamically constrained car-like vehicles, thereby falling short of the community's requirements. Nevertheless, it remains a popular platform for teaching autonomy fundamentals, much like TurtleBot3 Robotis Inc. (2021).

However, these platforms pose 3 major limitations:

- Firstly, some of these platforms lack diverse sensing modalities, sufficient computational power, constrained actuation mechanisms, and active or passive infrastructural elements.
- Secondly, most platforms utilize commercial radio-controlled (RC) cars as their base-chassis, which are expensive, may not be readily available worldwide, and limit exploration in the realm of mechatronics engineering. Additionally, many platforms are confined to specific software frameworks like ROS, creating a skill-set dependency for end-users.
- Thirdly, some platforms lack any form of simulation support, some support simulation using RViz Hershberger et al. (2021) or Gazebo, while others provide task-specific Gym Brockman et al. (2016) environments for machine learning; none of which is ideal.

3. AUTODRIVE ECOSYSTEM

This section primarily focuses on digital-twin capabilities of AutoDRIVE Ecosystem, highlighting the strong resemblance between AutoDRIVE Simulator Samak et al. (2021c); Samak and Samak (2022b) and AutoDRIVE Testbed for seamless sim2real transfer of autonomy algorithms developed using AutoDRIVE Devkit.

3.1 Physical Vehicle

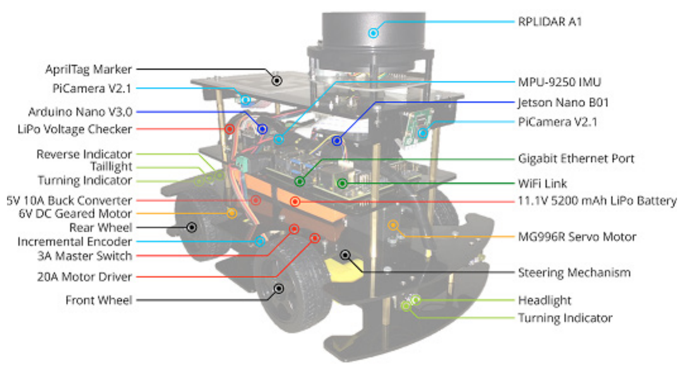


Fig. 2. Native vehicle (Nigel) of AutoDRIVE Ecosystem with its components and sub-systems highlighted.

AutoDRIVE's native vehicle, named Nigel (refer Fig. 2), offers realistic driving and steering actuation, comprehensive sensor suite, high-performance computational resources, and a standard vehicular lighting system.

Chassis Nigel is a 1:14 scale vehicle, which adopts front/rear/all-wheel-drive Ackermann-steered mechanism, thereby resembling car-like kinodynamic constraints.

Power Electronics An 11.1 V 5200 mAh lithium-polymer (LiPo) battery acts as the power-source for the vehicle. Other components such as the buck converter, motor driver, switch and voltage checker help route the raw power to appropriate sub-systems of the vehicle.

Sensor Suite Nigel hosts a comprehensive sensor suite comprising throttle and steering feedbacks, 1920 CPR incremental encoders, 3-axis IPS, 9-axis IMU, two 62.2° FOV cameras with 3.04 mm focal length and a 7-10 Hz, 360° FOV LIDAR with 0.15-12 m range and 1° resolution.

Computation, Communication and Software Nigel relies on Jetson Nano Developer Kit - B01 for high-level computation (autonomy algorithms), communication (V2V and V2I), and software installation (JetPack SDK and AutoDRIVE Devkit). It also utilizes Arduino Nano (running the vehicle firmware) for sensor data acquisition and filtering, and actuators/lights control.

Actuators Nigel is equipped with two 6V 160 RPM rated DC geared motors to drive its rear wheels, and a 9.4 kgf.cm servo motor (saturated at ± 30° w.r.t. zero-steer value) to steer its front wheels. All the actuators are operated at 5V, providing a top speed of ∼0.26 m/s for driving and ∼0.42 rad/s for steering.

Lights and Indicators Nigel's lighting system comprises of dual-mode headlights, triple-mode turning indicators, and automated taillights and reverse indicators.

3.2 Virtual Vehicle

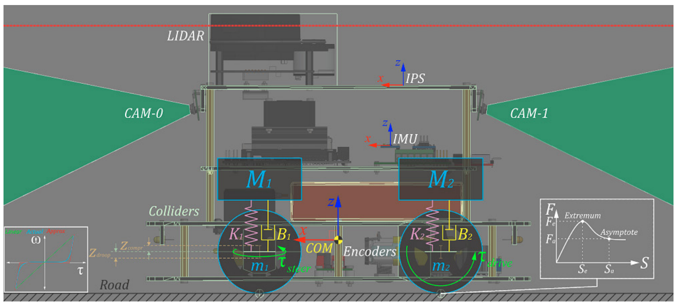


Fig. 3. Simulation of vehicle dynamics, sensors and actuators. The left inset depicts actuator dynamics model and the right inset depicts tire dynamics model.

The vehicle is jointly modelled as a rigid-body and a collection of sprung masses with inherent damping (refer Fig. 3). The "sprung-mass" representation computes the suspension forces, which, aggregated with the tire forces, are applied to the "rigid-body" representation that exactly mimics the mass, center of mass and moment of inertia of the "sprung-mass" representation. Needless to say, the two representations are related through the rigid-body center of mass equation: $X_{COM} = \frac{\sum {}^iM * {}^iX}{\sum {}^iM}$; where, X_{COM} is the rigid-body center of mass offset, ${}^iM$ are the sprung masses such that $M = \sum {}^iM$ is the total mass of rigid-body, and ${}^iX$ are the sprung mass coordinates w.r.t. ${}^iM$.

Suspension Dynamics The vehicle is modeled with a rather stiff suspension system to simulate the selective passive compliance between wheel mounts and vehicle chassis to account for losses at these interfaces due to vibration, friction, wear, damping, loosening, deformation, fatigue and fretting. The suspension force acting on each of the sprung masses can be then computed using a second-order dynamic model: ${}^iM * {}^i\ddot{Z} + {}^iB * ({}^i\dot{Z} - {}^i\dot{z}) + {}^iK * ({}^iZ - {}^iz)$; where, ${}^iB$ and ${}^iK$ are the damping and spring coefficients of i-th suspension, respectively. The computed suspension forces jointly affect the rigid-body dynamics of the vehicle as well as the tire forces being computed at that time instant (since it affects the load bearing down on the tires).

Wheel Dynamics The wheels of the vehicle are also modelled as rigid bodies of mass m, acted upon by gravitational and suspension forces: ${}^im * {}^i\ddot{z} + {}^iB * ({}^i\dot{z} - {}^i\dot{Z}) + {}^iK * ({}^iz - {}^iZ)$; and the wheel rotations are damped to mimic rotational losses due to rolling resistance.

Tire Dynamics The tire forces are broken down into longitudinal F_{t_x} and lateral F_{t_y} components, and are computed based on the respective friction curve for each tire: $\begin{cases} {}^iF_{t_x} = F({}^iS_x) \\ {}^iF_{t_y} = F({}^iS_y) \end{cases}$; where, ${}^iS_x$ and ${}^iS_y$ are the longitudinal and lateral slips of i-th tire, respectively. Here, the friction curve is approximated as a two-piece spline $F(S)$; one from zero (S_0, F_0) to extremum point (S_e, F_e), and other from extremum point (S_e, F_e) to asymptote point (S_a, F_a) (refer right inset in Fig. 3): $F(S) = \begin{cases} f_0(S); & S_0 \leq S < S_e \\ f_1(S); & S_e \leq S < S_a \end{cases}$; where, $f_k(S) = a_k * S^3 + b_k * S^2 + c_k * S + d_k$ is a cubic polynomial function, and $F(S)$ is saturated after the asymptote point (S_a, F_a).

Now, slip is in-turn dependent on the various factors like tire stiffness, steering angle, wheel speeds, suspension forces and rigid-body momentum. Longitudinal slip S_x is determined based on the difference between the longitudinal components of surface velocity of the wheel compared to the angular velocity of the wheel: ${}^iS_x = \frac{{}^ir * {}^i\omega - v_x}{v_x}$; where, v_x is the longitudinal linear velocity of the vehicle (i.e., surface velocity of the wheel), ${}^ir$ is the radius of i-th wheel, and ${}^i\omega$ is the angular velocity of i-th wheel. Lateral slip S_y is determined by the angle (commonly denoted as α) between the direction the tire is moving in and the direction the tire is pointing in: ${}^iS_y = \frac{v_y}{|v_x|}$; where, v_x is the longitudinal linear velocity of the vehicle, and v_y is the lateral linear velocity of the vehicle (a.k.a. sideslip velocity).

Sensor Simulation As described earlier, the vehicle is provided with an abundance of sensing modalities, all of which are modeled and simulated close to their real-world counterparts.

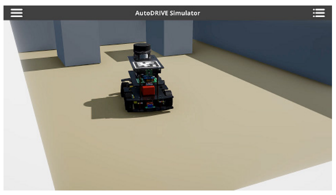
- *Throttle Feedback:* Instantaneous feedback of throttle command $[-1, 1]$, where positive values indicate driving forward and negative values indicate driving reverse.
- *Steering Feedback:* Instantaneous feedback of steering command $[-1, 1]$, where positive values indicate left turns and negative values indicate right turns.
- *Incremental Encoders:* These are simulated by measuring the rotation of each of the rear wheels (based on their rigid-body transform update) and factoring in the resolution of the encoders.
- *IPS:* Position of the vehicle is measured based on its rigid-body transform update. The values are converted from Unity's left-handed coordinate system to the right-handed coordinate system widely adopted for robotics applications. This mimics the AprilTag-based fiducial localization system on the physical vehicle.
- *IMU:* Orientation of the vehicle is measured based on its rigid-body transform update. Additionally, the linear acceleration and angular velocity of the vehicle are measured based on temporally-coherent rigid-body transformations, using rigid-body equations of motion. This mimics the MPU-9250 on the physical vehicle.
- *LIDAR:* Planar laser scan is recorded by 360° iterative ray-casting at 1° resolution and 7 Hz update rate. The raycast hits are recorded between the minimum (0.15 m) and maximum (12.0 m) ranges of the LIDAR, respectively. This mimics the RPLIDAR A1 on the physical vehicle.
- *Cameras:* Physical cameras are simulated based on their focal length (3.04 mm), field of view (62.2°), sensor size (4.6 mm) and target resolution (720p). Additionally, lens and film effects are simulated by post-processing the raw frames. This mimics the two PiCamera V2.1 modules on the physical vehicle.

Actuator Simulation As described earlier, the vehicle has driving and steering actuators, the response delays and saturation limits of which are matched with their real-world counterparts by tuning their torque profiles and actuation limits, respectively (refer left inset in Fig. 3).

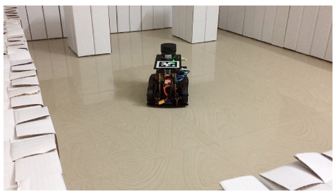
- *Driving Actuators:* Each of the rear wheels is driven using a rotary motor, which applies a torque to it: ${}^{i}\tau_{drive} = {}^{i}I_w * {}^{i}\dot{\omega}_w$; where, ${}^{i}I_w = \frac{1}{2} * {}^{i}m_w * {}^{i}{r_w}^2$ is the moment of inertia of i-th wheel, and ${}^{i}\dot{\omega}_w$ is the angular acceleration of i-th wheel. Additionally, the holding torque of the driving actuators is simulated by applying an idle motor torque equivalent to the braking torque: ${}^{i}\tau_{idle} = {}^{i}\tau_{brake}$.
- *Steering Actuators:* The front wheels are steered using a steering actuator coupled to the steering mechanism of the vehicle. The steering actuator also produces a torque proportional to the moment of inertia of the steering mechanism: $\tau_{steer} = I_{steer} * \dot{\omega}_{steer}$. The individual turning angles, δ_l and δ_r, for left and right wheels, respectively, are calculated based on the commanded steering angle δ, using the Ackermann steering geometry defined by wheelbase l and track width w, as follows: $\begin{cases} \delta_l = \tan^{-1}\left(\frac{2*l*\tan(\delta)}{2*l+w*\tan(\delta)}\right) \\ \delta_r = \tan^{-1}\left(\frac{2*l*\tan(\delta)}{2*l-w*\tan(\delta)}\right) \end{cases}$

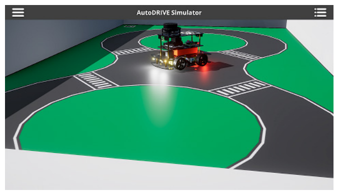
3.3 Physical Infrastructure

AutoDRIVE provides a modular and reconfigurable infrastructure development kit, enabling swift design and construction of customized driving scenarios.



(a) Parking School in simulation.


(b) Parking School in reality.



(c) Driving School in simulation.

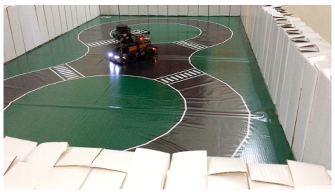

(d) Driving School in reality.

Fig. 4. Infrastructure setup in simulation and reality. Note the degree of dimensional and visual similarity between real and virtual worlds.

Environment Modules Environment modules include static terrain and road layouts as well as obstruction objects for rapidly designing custom scenarios. Apart from these, experts may also choose to design scaled real-world or imaginary scenarios using third-party tools, and import them into AutoDRIVE Ecosystem.

Traffic Elements Traffic signs and lights define traffic laws within a particular driving scenario, thereby governing the traffic flow. These modules support IoT and V2I communication technologies, and can be therefore integrated with AutoDRIVE Smart City Manager (SCM).

Surveillance Elements AutoDRIVE features a surveillance element called AutoDRIVE Eye to view the scene from a bird's eye view. The said element is also integrated with AutoDRIVE SCM, and is capable of estimating vehicle's 2D pose within the map by detecting and tracking the AprilTag markers attached to each of them.

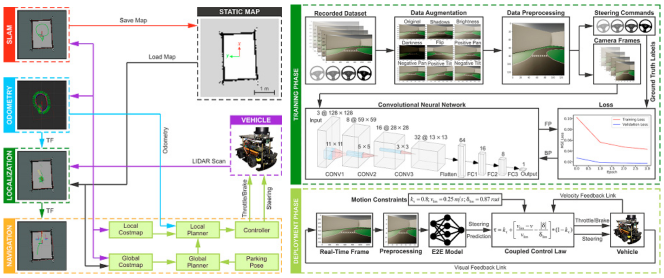
Preconfigured Maps This work focuses on two of the several preconfigured maps offered by AutoDRIVE Ecosystem. Parking School (refer Fig. 4a, 4b) is designed specifically for autonomous parking applications, wherein construction boxes define static obstacles and all the available free-space is drivable. On the other hand, Driving School (refer Fig. 4c, 4d) covers driving over straight roads, curved roads and crossing an intersection.

3.4 Virtual Infrastructure

At every time step, the simulator performs mesh-mesh collision/interference detection and accordingly computes the contact forces, frictional forces and momentum transfer, along with linear and angular drag acting over each of the rigid bodies (vehicle/infrastructure) present in the scene. This closely emulates the interactions among vehicle(s), infrastructure elements and the environment.

4. CASE STUDIES

This work showcases sim2real capability of AutoDRIVE Ecosystem through two different case-studies. Although this paper cannot furnish exhaustive details pertaining to either case study, we recommend interested readers to peruse this technical report Samak and Samak (2022a).

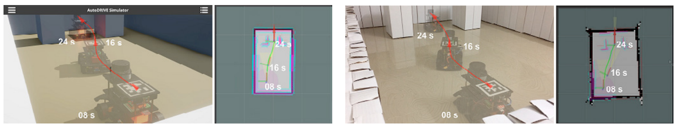


(a) Autonomous parking. (b) Behavioral cloning.

Fig. 5. Architectures of the two presented case-studies.

4.1 Autonomous Parking

This case study was implemented using the probabilistic robotics approach comprising 5 different stages. Firstly, the vehicle mapped its surroundings using the Hector SLAM algorithm Kohlbrecher et al. (2011). It then localized itself against this known static map using range-flow-based odometry Jaimez et al. (2016) and an adaptive particle filter algorithm Fox (2001). For autonomous navigation, the vehicle planned a feasible global path from its current pose to the parking pose using the A* algorithm Hart et al. (1968). Simultaneously, it re-planned its local trajectory for dynamic collision avoidance, leveraging the timed-elastic-band approach Rösmann et al. (2017). A proportional controller generated driving (throttle/brake) and steering commands to enable the vehicle to follow the local trajectory accurately.



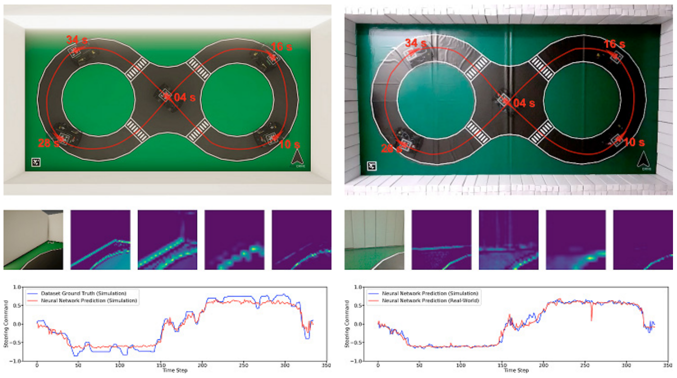
(a) Virtual-world deployment. (b) Real-world deployment.

Fig. 6. Sim2real transition of autonomous parking algorithm. Video: `https://youtu.be/piCyvTM2dek`

During simulation-based testing (refer Fig. 6a), parameter variations such as infusion of Gaussian noise in LIDAR measurements [$n_{lidar} \sim N(0, 0.025)$ m] and actuator commands [$n_{drive} \sim N(0, 0.013)$ m/s and $n_{steer} \sim N(0, 0.018)$ rad/s]. Furthermore, perturbation of wall modules' poses [$n_{x,y} \sim N(0, 0.01)$ m and $n_\theta \sim N(0, 0.087)$ rad] as well as introduction of unmapped obstacles were performed. Going forward, the same pipeline was deployed on the real vehicle (refer Fig. 6b) using AutoDRIVE Testbed to validate seamless sim2real transfer. The pipeline performed flawlessly owing to realistic LIDAR and vehicle dynamics models as well as variability analysis during simulation.

4.2 Behavioral Cloning

This case study was based on Samak et al. (2021b), wherein the objective was to utilize a convolutional neural network (CNN) for cloning the end-to-end driving behavior of a human (refer to Fig. 5b). To achieve this, AutoDRIVE Simulator was employed to record 5-laps of temporally-coherent labeled manual driving data. This data was then balanced, augmented, and pre-processed using standard computer vision techniques to train a 6-layer deep CNN model for 4 epochs with a learning rate of 1e-3 using the Adam optimizer Kingma and Ba (2014). Following training, the model was deployed back into AutoDRIVE Simulator to evaluate its performance (refer to Fig. 7a).



(a) Virtual-world deployment. (b) Real-world deployment.

Fig. 7. Sim2real transition of behavioral cloning algorithm. Video: `https://youtu.be/rejpoogaXOE`

Variability testing concerning light intensity and direction as well as vehicle's initial conditions and velocity limit was performed to ensure algorithm robustness. Further, the same CNN model was deployed onto AutoDRIVE Testbed for validating the sim2real transition of this vision-based algorithm (refer Fig. 7b). Despite subtle variations in the environmental conditions, the model generalized well and the vehicle was able to complete several laps along the track without major deviation and/or collision.

5. CONCLUSION

In this paper, we presented AutoDRIVE, a publicly accessible digital twin ecosystem for CAVs developed with an aim of tightly integrating reality and simulation into a unified toolchain, without compromising on the comprehensiveness, flexibility and accessibility required for prototyping and validating autonomy solutions. This work focused on bridging the autonomy-oriented sim2real gap using the proposed ecosystem. Furthermore, we extensively discussed the modeling and simulation aspects of the ecosystem and substantiated its efficacy by demonstrating the successful transition of two candidate autonomy algorithms including autonomous parking and behavioral cloning from simulation to reality to help support our claims. Further research will delve into the investigation of handling real and virtual world uncertainties, formulating qualitative/quantitative evaluation metrics and benchmarks, as well as improving the robustness and generalization of sim2real frameworks for autonomous vehicles.

REFERENCES

Baidu Inc. (2021). Apollo Game Engine Based Simulator.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym.

Donkey Community (2021). An Open-Source DIY Self-Driving Platform for Small-Scale Cars.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. In S. Levine, V. Vanhoucke, and K. Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, 1–16. PMLR.

Fox, D. (2001). KLD-Sampling: Adaptive Particle Filters. In T. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press.

Goldfain, B., Drews, P., You, C., Barulic, M., Velev, O., Tsiotras, P., and Rehg, J.M. (2019). AutoRally: An Open Platform for Aggressive Autonomous Driving. *IEEE Control Systems Magazine*, 39(1), 26–55. doi: 10.1109/MCS.2018.2876958.

Hart, P.E., Nilsson, N.J., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107. doi:10.1109/TSSC.1968.30 0136.

Hershberger, D., Gossow, D., and Faust, J. (2021). RViz: 3D Visualization Tool for ROS.

HyphaROS Workshop (2021). HyphaROS Racecar.

Jaimez, M., Monroy, J.G., and Gonzalez-Jimenez, J. (2016). Planar Odometry from a Radial Laser Scanner. A Range Flow-based Approach. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 4479–4485. doi:10.1109/ICRA.2016.7487647.

Karaman, S., Anders, A., Boulet, M., Connor, J., Gregson, K., Guerra, W., Guldner, O., Mohamoud, M., Plancher, B., Shin, R., and Vivilecchia, J. (2017). Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT. In *2017 IEEE Integrated STEM Education Conference (ISEC)*, 195–203. doi:10.1109/ISECon.2017.7910242.

Kingma, D.P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. doi:10.48550/ARXIV.1412.69 80.

Koenig, N.P. and Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, 2149–2154. doi:10.1109/IR OS.2004.1389727.

Kohlbrecher, S., von Stryk, O., Meyer, J., and Klingauf, U. (2011). A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 155–160. doi:10.1109/SSRR.2011.6106777.

O'Kelly, M., Sukhil, V., Abbas, H., Harkins, J., Kao, C., Pant, Y.V., Mangharam, R., Agarwal, D., Behl, M., Burgio, P., and Bertogna, M. (2019). F1/10: An Open-Source Autonomous Cyber-Physical Platform.

Paull, L., Tani, J., Ahn, H., Alonso-Mora, J., Carlone, L., Cap, M., Chen, Y.F., Choi, C., Dusek, J., Fang, Y., Hoehener, D., Liu, S.Y., Novitzky, M., Okuyama, I.F., Pazis, J., Rosman, G., Varricchio, V., Wang, H.C., Yershov, D., Zhao, H., Benjamin, M., Carr, C., Zuber, M., Karaman, S., Frazzoli, E., Del Vecchio, D., Rus, D., How, J., Leonard, J., and Censi, A. (2017). Duckietown: An Open, Inexpensive and Flexible Platform for Autonomy Education and Research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1497–1504. doi:10.1109/ICRA.2017.7989179.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3.

Rösmann, C., Hoffmann, F., and Bertram, T. (2017). Kinodynamic Trajectory Optimization and Control for Car-Like Robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5681–5686. doi:10.1109/IROS.2017.8206458.

Robotis Inc. (2021). TurtleBot3.

Rong, G., Shin, B.H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., Agafonov, E., Kim, T.H., Sterner, E., Ushiroda, K., Reyes, M., Zelenkovsky, D., and Kim, S. (2020). LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. doi: 10.1109/ITSC45102.2020.9294422.

Samak, C.V., Samak, T.V., and Kandhasamy, S. (2021a). Autonomous Racing using a Hybrid Imitation-Reinforcement Learning Architecture. doi:10.48550/A RXIV.2110.05437.

Samak, T., Samak, C., Kandhasamy, S., Krovi, V., and Xie, M. (2023). AutoDRIVE: A Comprehensive, Flexible and Integrated Digital Twin Ecosystem for Autonomous Driving Research Education. *Robotics*, 12(3). doi:10.3390/robotics12030077.

Samak, T.V. and Samak, C.V. (2022a). AutoDRIVE - Technical Report. doi:10.48550/ARXIV.2211.08475.

Samak, T.V. and Samak, C.V. (2022b). AutoDRIVE Simulator - Technical Report. doi:10.48550/ARXIV .2211.07022.

Samak, T.V., Samak, C.V., and Kandhasamy, S. (2021b). Robust Behavioral Cloning for Autonomous Vehicles Using End-to-End Imitation Learning. *SAE International Journal of Connected and Automated Vehicles*, 4(3), 279–295. doi:https://doi.org/10.4271/12-04-03-0 023.

Samak, T.V., Samak, C.V., and Xie, M. (2021c). AutoDRIVE Simulator: A Simulator for Scaled Autonomous Vehicle Research and Education. In *2021 2nd International Conference on Control, Robotics and Intelligent System*, CCRIS'21, 1–5. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3483845.34 83846.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In M. Hutter and R. Siegwart (eds.), *Field and Service Robotics*, 621–635. Springer International Publishing, Cham.

Voyage (2021). Deepdrive.

Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., and Sumner, A. (2021). TORCS, The Open Racing Car Simulator.