Repeatable & Scalable Multi-Vehicle Simulation with Offloaded Dynamics using Federated Modeling

Rahul Bhadani[1] and Jonathan Sprinkle[2]

[1] *Department of Electrical and Computer Engineering, The University of Arizona, Tucson, Arizona, USA*[*]

[2] *Department of Computer Science, Vanderbilt University, Nashville, TN, USA*[†]

In this paper, we present a method to perform multi-vehicle simulation of autonomous systems that improves the repeatability of robotics simulations and can improve the scale of such simulations for dynamically complex devices such as autonomous vehicles (AV). Current approaches to simulation of multi-component AV typically infer the kinematics or dynamics through the rigid-body motion that uses joint angles and shapes. Such simulations encounter challenges for simulated AV, as the methods to discretize the behavior are prone to error accumulated over time and are computationally intensive – frequently resulting in chaotic behavior. The accumulated error results in a lack of repeatability of the simulation results. Further, when simulating multiple AVs, simulations typically fail to scale to tens of vehicles, even when slowed to permit more accurate results as the state evolves. This paper provides an architecture for improving the repeatability of simulations using federated modeling and state synchronization through a director. The method consists of replacing the inverse kinematic vehicle models with computational models of their dynamics, offloading dynamics from the physics engine for state evolution, synchronizing vehicle updates using a director, and performing the simulation at slower than real-time if needed. Our method reduces the error of trajectory deviation during repeated simulations by at least threefold. An implementation of the results of the work is presented through a Robot Operating System (ROS) package.

I. INTRODUCTION

In robotics and autonomous cyber-physical systems (CPS), simulation provides cost-effective means to uncover corner cases and validate a vehicle controller without logistic bottlenecks of field experiments. Modeling autonomous robotic vehicles requires an understanding of highly nonlinear models of physical systems. There are significant works on the understanding of nonlinear systems [1–3] and its approximation [4–7]. However, nonlinearity causes the system to exhibit sensitivity around the initial condition and the trajectory evolution diverges with time. Thus, we may seek a simpler model that leads to a predictable outcome within permissible accuracy. The fragility (lack of repeatability) of such a system makes it harder to conduct simulations with repeatable results. For societal-scale CPS, a reliable and repeatable[1] simulations are desirable. Repeatability allows peers to reproduce research, and to assess experimental results that can support proof of concepts; such simulations are reasonable tools to study any bias that has been introduced either due to bad experiment design, less-useful system model[2] or incorrect inertial tensor [10] and take necessary measure to discover and eliminate biases. As autonomous vehicle CPS are intrinsically heterogeneous in nature and involve complex interaction among computational, physical, and human components, we think that taking the approach of federated modeling may be one way to reduce the complexity of simulation and increase its repeatability.

Here we present a workaround and an associated software tool to achieve repeatable simulation for Connected and Autonomous Vehicles (CAV) using an existing 3D simulation framework. Our method uses offloading dynamics from the physics engine to federated models, director for time synchronization, and conducting simulation at slower than real-time for achieving repeatable results. Our method was able to reduce the root-mean-square error of trajectory deviation over multiple repeated simulations by at least threefold. We emphasize that our work is about creating methods to provide repeatable simulations with popular tools such as ROS-Gazebo with minimal or no changes to an existing software tool. The main results of the paper are highlighted in Table I.

A. Contribution

We present a workaround to mitigate the fragility and associated results. Our contribution is as follows.

(i) We develop an architecture for multi-vehicle simulation of autonomous vehicles to facilitate repeatable outcomes

(ii) We propose a software tool that works in conjunction with existing ROS-based simulators to enhance

[*] rahulbhadani@email.arizona.edu

[†] jonathan.sprinkle@vanderbilt.edu

[1] Repeatability is the property of an experiment that yields the same outcome from several trials, performed at different times and in different places [8]. Another definition of repeatability comes from [9] that defines repeatability in robotics as a measure of the ability of the robot (vehicle) to move back to the same position and orientation over and over again.

[2] Less useful captures the sentiment captured by George E. P. Box — “All models are wrong, but some are useful”.

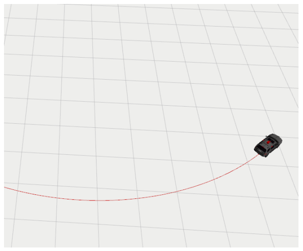

FIG. 1. Open-loop control of the simulated vehicle with constant velocity $v = 3$ m/s and steering angle $\delta = 0.07065$ rad as per Equation (1). Vehicle simulation in ROS/Gazebo of the CAT Vehicle Testbed 2.0.2.

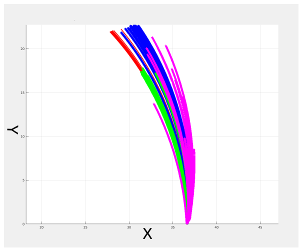

FIG. 2. Repeated simulation of the identical input and identical initial conditions, recorded on multiple machines. Each curve denotes the trajectory from a different simulation. Colors correspond to simulations on different machines. Each simulation ran for different amount of time ranging from 10 s to 120 s approximately. Videos demonstrating the lack of repeatability can be found on our public GitHub repository https://github.com/jmscslgroup/repeatability-analysis.

the repeatability of simulations.

For the rest of the discussion, we will use the physics engine and Gazebo interchangeably for simplicity. However, we should remember that Gazebo is a simulator that uses physics engines such as Open Dynamics Engine, Bullet, and Dart along with a rendering engine.

Section II provides background on the problem discussed in this paper. Section III formulates the problem statement. Section IV provides some examples to demonstrate fragility in simulation using ROS and proposes methods to mitigate those weaknesses. Section V demonstrates success in achieving repeatable & scalable simulations. We compare the state-of-the-art (SOTA) method of simulation in ROS with our proposed method, called the offloaded dynamics. We conclude the article with future directions and plan the extension of the current work.

II. BACKGROUND

Advanced problems in CAV research involve vehicle-to-infrastructure and vehicle-to-vehicle communication, and vehicle control that requires reliable communication. In such systems, we are increasingly moving from a static environment to a dynamic one. Such complex scenarios require a scalable simulation with a common model of computation, typically through ROS [11] and a physics engine simulator (e.g. Gazebo) for implementing any applications either for simulation or for field experiments. Examples of the use of ROS in physical system experiments for CAVs include ring road experiments for the dampening of traffic waves with a single automated vehicle [12–14], and demonstration that commercially available adaptive cruise controllers are not string-stable [15] using the CAT Vehicle Testbed [16].

Preparations for the experiments in [12] were unable to rely on simulation prior to the full-scale field experiment due to weaknesses in repeatability and inability to scale to even tens of vehicles in the ring (the experiment required over 20 vehicles). Scaling simulation to more than two vehicles resulted in stochastic disruptions, including an abrupt drop of real-time factor (RTF), loss of messages intended for delivery to respective agents, and trajectory deviation. While doing evaluations of velocity controllers, we further found evidence of instability and lack of repeatable results over several simulations in the same computer as well as across different computers. A similar study on simulation fragility seen in CARLA simulator was reported in [17] where the authors didn't demonstrate any solution or workaround strategy to overcome the fragility of the simulation.

We present an example of fragility with ROS and Gazebo simulators. Consider the trajectories of repeated simulations of an autonomous vehicle (AV) under the open-loop control with fixed speed $v = 3$ m/s and a fixed steering angle of $\delta = 0.07065$ rad, that ran for 10 s to 120 s, as shown in Figure 1, and Figure 2. The simulation was performed on four machines with different configurations (varying number of processor cores, RAM, etc). From the traces of simulation on these machines, we determined that the outcome of the simulation differed between machines, as well as on the same machine. For a feedback control algorithm, deviation in simulation results has potential issues in terms of repeatability of the results and could artificially alter the outcome.

Limitations of the existing simulation method motivate the following questions: What are the factors contributing to the deviation of simulation results across different runs? What are the metrics that define such deviations? What are the ways to achieve repeatability in such cases without altering software and hardware? We envision that solving these problems to produce a repeatable & scalable simulation is key to developing and testing societal-scale CPS applications such as CAVS, with respect to the safe operation of AVs and their interaction with the environment. Such simulations are a powerful tool for developing, and verifying CPS applications and enabling transfer learning from simulation to hardware, as these simulations can simulate sensors, delay, and interaction with agents in the 3D world – which is not possible with microscopic simulators such as Sumo [18], and Aimsun [19].

A. Plausible Reasons for Simulation Fragility

State-of-the-art simulation software may not exhibit repeatability issues for a small number of vehicles (depending on the availability of RAM, GPU, etc.), but all simulations begin to degrade when the system load grows. In other words, those simulations fail to repeat in attempting to scale. Control of an AV simulation requires the movement of rigid bodies and interaction among bodies in 3D space. Physics engines use differential equations solvers to compute contact forces in both linear and angular dimensions and determine state evolution over time. The overall process is computationally expensive. A second challenge is that the simulation of each of the joints and masses that make up the vehicle is prone to errors that accumulate over time.

As the vehicle models are made more complicated and refined, say, by increasing the number of triangular mesh or actuators, and joints or by spawning more vehicles in the simulated world, computation slows down to provide a trade-off between accuracy and performance. One such trade-off is discussed in [20] where an iterative process is used to solve a differential equation. When the system load is high, the computation needs more time to return the result and small deviations in error may accumulate over time. Meanwhile, the simulation engine chooses to advance the simulation by some extrapolation techniques. Over time, error accumulates and the decision-making ability of the controller gets affected non-deterministically which varies across simulation runs as agents are making decisions based on data that differ between different simulation runs. For multi-vehicle simulations, as the number of vehicles increases, the amount of messages increases, and beyond a given threshold, data packets start dropping. This packet-drop is entirely system-load dependent and hence affects the control decision made by vehicles that are not reproducible.

For a physics engine, floating-point precision is an important aspect. If system dynamics happen to be chaotic in nature, the hardware floating precision affects the outcome as the system evolves. ROS as well as the various implementation of the physics engine are multi-threaded and hundreds of processes run in parallel. In such a case, if an operating system scheduler interrupts jobs being executed by ROS or physics engine in favor of another higher priority job, we may end up with non-repeatable results of the simulation.

III. PROBLEM STATEMENT: FRAGILITY IN AUTONOMOUS VEHICLE SIMULATION

Consider a plant function f that is used to simulate a car in a physics engine. We limit the scope of this paper to velocity control for controlling the positions of vehicles:

$$\mathbf{x} = f(u) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state of the vehicle consisting of position, heading, and tire angle, and $u \in \mathbb{R}^m$ is the control input consisting of velocity and desired steering angle, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$; n is the size of state vector and m is the size of input vector. Let $\hat{f}$ be the discretized f:

$$\hat{\mathbf{x}}_k = \hat{f}(u_k) \tag{2}$$

For examples presented in this article, we use CAT Vehicle simulator [16] implemented in ROS/Gazebo where a car is controlled by a velocity controller: $u_k \equiv [v_k, \delta_k]$ where v_k and δ_k are input velocity and desired steering angle respectively at a discrete-time with time index k. In the simulator, f is a kinematic model of the car that uses ROS for control. Rigid body dynamics is provided by Gazebo. Further, let $\hat{g}$ be a discretized plant function representing the physics engine:

$$\xi_k = \hat{g}(\mathbf{W}, \hat{\mathbf{f}}, h, \rho, \psi, \mathbf{J}) \tag{3}$$

where ξ_k is the state vector representing the overall simulation. In general, ξ_k is a vector consisting of linear, and angular positions, linear, and angular velocities of agents, sensors, and non-actors in the world frame of reference, $\mathbf{W}$ is a vector of applied/external forces and torque on agents in the simulation. The plant function houses agents denoted by plant models $\hat{\mathbf{f}} \equiv \{\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_n\}$. h is the simulation time-step, and ρ is the maximum update rate of Gazebo. The RTF is a product $h \times \rho$. ψ denotes simulated sensors in the plant function. $\mathbf{J}$ is the full specification of the AV model with actuators, joints, and links, and other parameters such as inertial tensor, frictional coefficients, etc. that are required for simulating vehicles by physics engine and for calculating contact forces and collision for rigid body dynamics.

For each pair of simulation runs (either on same computer or different ones), error terms to denote fragility in vehicle simulation can be calculated as

$$e(t) = \sqrt{(x(t)_{\text{sim}_1} - x(t)_{\text{sim}_2})^2 + (y(t)_{\text{sim}_1} - y(t)_{\text{sim}_2})^2} \tag{4}$$

where $(x(t), y(t))$ denotes coordinates of the vehicle's trajectory, and subscripts denote different simulation runs. From Equation (4), we can calculate root mean square error as overall error for the deviation between trajectories

$$E = \sqrt{\frac{\sum_{i=1}^{N} e_i^2(t)}{N}} \tag{5}$$

where N is the number of coordinate points in the trajectory data. For a repeatable system, the error E across the simulation should be bounded, i.e., $E \pm \epsilon$ where ϵ is some uncertainty threshold. However, existing multi-vehicle simulators lack such bounds.

The original paper in Gazebo [21] mentions the simulator's limitations in terms of scaling the distributed computation. Simulation using complex models in Gazebo fail to provide reliable result in real-time. As a result, the RTF in Gazebo slows down, data packets are dropped, and impacts the reliability of the simulation. Reduction in the RTF depends on the current processor load, available RAM, and network load, thereby affecting the repeatability of the simulation. In such a scenario, our problem statement is as follows:

1. Design an architecture for multi-vehicle simulation that can provide repeatable outcomes within $E = \pm\epsilon$.
2. Find a non-intrusive method to achieve repeatability that does not require changes to the simulators, software architecture, or operating system.
3. Demonstrate an integration strategy for ROS and Gazebo to perform repeatable simulation using a software tool.

Our findings demonstrate that control of simulated vehicles by federated models and employing a director for state synchronization provides an elegant solution to achieve more repeatable results for trajectory simulations of vehicles.

IV. METHODS: MITIGATING THE FRAGILITY OF SIMULATION

In ROS-Gazebo simulation, the input control command actuates joints of the AV model in Gazebo (or any other physics simulator). Joints through equations of physics change the state of various links constituting the vehicle upon actuation. Concurrently, contact forces between various surfaces are also calculated. This overall approach is computationally intensive and may lead to packet drops and bandwidth bottlenecks, which are undesirable for real-time control applications. Figure 3 demonstrates the real-time factor of the SOTA method in ROS for simulated AV driving with constant u_k in a circle for a number of simulations. The RTF deteriorates as the number of vehicles increases until the RTF is no longer governed by the product $h \times \rho$. Figure 4 shows error $e(t)$ as a function of time for a circular trajectory of a vehicle obtained from a pair of simulations run on two different machines.

Our method for achieving repeatability and scalability is broken down into three parts: (A) offloading the vehicle dynamics to a federated model, (B) using a director model for message handling, processing, and synchronization (C) simulating at an RTF suitable for scalability.

A. Federated Modeling

Instead of simulating vehicles using rigid body dynamics offered by the Gazebo physics engine, we simulate vehicle dynamics through a separate ROS node. Hence, the car plant $\hat{f}$ is replaced by $\hat{\mathbf{x}}_k^* = \hat{f}^*(u_k)$ which is decoupled from Gazebo $\hat{g}$. Further, we can throttle down the publish rate $1/\Delta T$ (the rate at which the state of the system is published) such that $1/\Delta T < 1/h$. Throttling down avoids overwhelming the data network on ROS when simulating multiple vehicles and sensors. Thus, instead of actuating the vehicle model in the physics engine, a federated model updates the model's states relative to the world frame.

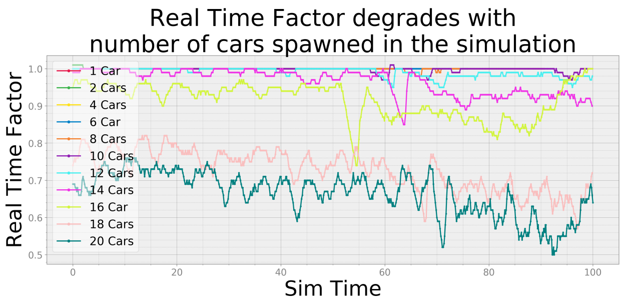

FIG. 3. RTF vs the number of vehicles in the simulation with the SOTA approach. As more number of vehicles are spawned in the simulation, the RTF increasingly deteriorates. Here, our desired RTF was 1.0.

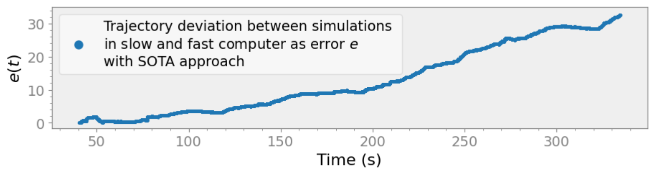

FIG. 4. Trajectory deviation as $e(t)$ between two simulation runs performed on two different computers with same simulation setup using SOTA

We use a bicycle model in Equation (6) as a federated model instead of 3D rigid body dynamics as used in the original CAT Vehicle simulator to model a vehicle $\hat{f}^*$. x_1, x_2 are the coordinates of the vehicle's position. x_3 is tire angle, and x_4 is heading angle. Thus, $\mathbf{x} \equiv [x_1, x_2, x_3, x_4]$.

$$\begin{aligned}
x_{1k} &= x_{1k} + \Delta T v_k \cos(x_{3k}) \cos(x_{4k}) \\
x_{2k} &= x_{2k} + \Delta T v_k \cos(x_{3k}) \sin(x_{4k}) \\
x_{3k} &= x_{3k} + \Delta T \delta_k \\
x_{4k} &= x_{4k} + \Delta T v_k \cdot \sin(x_{3k}) \cdot (1/L)
\end{aligned} \tag{6}$$

where L is the vehicle's wheelbase. The evolution of the system is then governed by Equation (6). The model feeds back its system velocity, position, and orientation to determine its position in the specified frame of reference. We use ROS for the implementation of Equation (6). The ROS node subscribes to the input velocity and steering angle and publishes the updated state vector $\hat{\mathbf{x}_k}$. $\hat{\mathbf{x}_k}$ is used to update the position of the vehicle in the Gazebo for visualization and interaction with other vehicle models and simulated sensors. In practice, any arbitrary $\hat{f}^*$ can be used to govern the vehicle's state evolution, replacing Equation (6), potentially including hybrid models that could speed up execution without compromising accuracy [22].

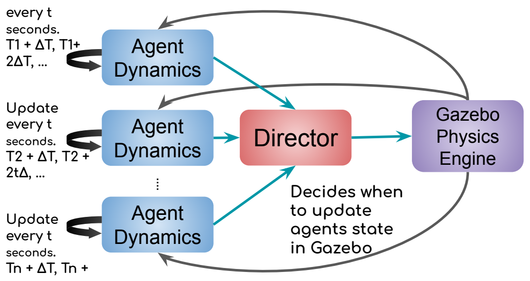
every t seconds. T1 + ΔT, T1+ 2ΔT, ...
Agent Dynamics
Update every t seconds. T2 + ΔT, T2 + 2tΔ, ...
Agent Dynamics
Director
Gazebo Physics Engine
Update every t seconds. Tn + ΔT, Tn +
Agent Dynamics
Decides when to update agents state in Gazebo

FIG. 5. A director synchronizes the state update for each agent for predictability and correct action by agents.

B. Synchronization using Director

Due to the way ROS works, physics engines cannot receive updated states to change the poses of each vehicle in the 3D world simultaneously. Consider $(T_1, T_2, T_3, \cdots)$ be the spawn time of the each vehicle node in ROS. If a throttled publish rate for vehicles is $1/\Delta T$, then each vehicle can produce its update state at $T_i + k\Delta T, \quad i \in [1, n]$. We introduce a director node that receives the updated state from $\hat{\mathbf{f}}^* \equiv \{\hat{f^*}_1, \hat{f^*}_2, \cdots, \hat{f^*}_n\}$ and decides when to send these states to Gazebo. The director maintains a stack of updated states in each time window ΔT and propagates updated states once the stack is filled, emptying the stack in the process. If the stack is not filled within the ΔT time window, then the stack is discarded and the process repeats. As soon as the stack is discarded or emptied, the director starts processing the next time step. The state of the vehicle in Gazebo doesn't change until it is directed to do so by the director and Gazebo stays paused. Without synchronization, even if a vehicle's state advances, a follower vehicle in the platoon will not be able to estimate its leader vehicle using simulated sensors in the 3D world correctly and may affect the decision-making ability of the deployed controller. We provide a schematic of the director in Figure 5.

C. Operating at slower than real-time

Reducing the RTF slows down the simulated clock and hence the number of messages produced per wall-clock time unit is reduced. This allows a physics engine to handle messages reliably without significant latency or message loss. By specifying product $h \times \rho < 1$, we can execute nodes in slower than real-time. For example $h = 0.01$ s, $\rho = 100$ Hz is real-time while $h = 0.01$ s, $\rho = 50$ Hz is half the real-time. As of this work, we choose appropriate RTF by trial-and-error. Thus a Gazebo plant model with the offloaded dynamics can be abstracted as

$$\xi_k = \hat{g^*}(\hat{\mathbf{x}}^*_k, h, \rho, \psi, \mathbf{J}') \tag{7}$$

where no external/applied force $\mathbf{W}$ is provided since the vehicle state updates are offloaded. $\hat{\mathbf{x}}^*_k$ are received from director and $h \times \rho$ may be less than 1.0. Unlike Equation (3), $\mathbf{J}'$ doesn't need to be full vehicle-specification but rather a solid body for which inertial tensor is simpler to calculate with greater precision. For comparison, a pictorial representation of SOTA and offloaded dynamics methods is provided in Figure 6 and Figure 7.

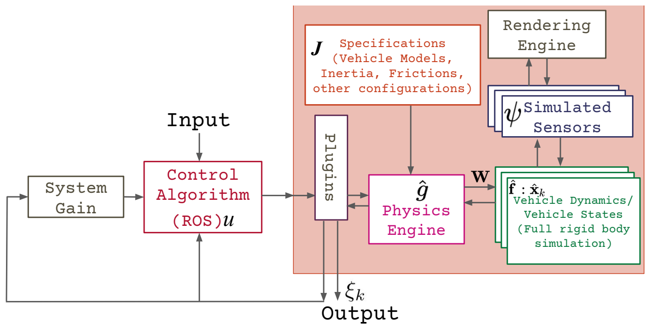
J Specifications (Vehicle Models, Inertia, Frictions, other configurations)
Rendering Engine
ψ Simulated Sensors
Input
System Gain
Control Algorithm (ROS) u
Plugins
$\hat{g}$ Physics Engine
$\mathbf{W}$
$\hat{\mathbf{f}} : \hat{\mathbf{x}}_k$ Vehicle Dynamics/ Vehicle States (Full rigid body simulation)
ξ_k
Output

FIG. 6. SOTA for AV simulation using ROS-Gazebo control. Vehicle specification consists of various actuators, links, and joints.

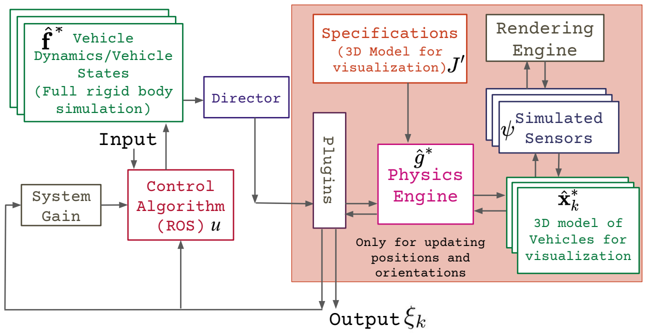
$\hat{\mathbf{f}}^*$ Vehicle Dynamics/Vehicle States (Full rigid body simulation)
Director
Specifications (3D Model for visualization) J'
Rendering Engine
ψ Simulated Sensors
Input
System Gain
Control Algorithm (ROS) u
Plugins
$\hat{g}^*$ Physics Engine
$\hat{\mathbf{x}}^*_k$ 3D model of Vehicles for visualization
Only for updating positions and orientations
Output ξ_k

FIG. 7. Modified approach using offloaded dynamics. Vehicle specification consists of a solid body devoid of separate links and joints. $\hat{f}^*$ computes vehicle dynamics for the state evolution independently. A director synchronizes state updates from multiple vehicle nodes. The new odometry information is consumed by Gazebo $\hat{g}^*$ to update the vehicle's state in the simulated world. The updated vehicle state and sensor information are fed back for the next time-step. Plugins are user-written programs to use physics engine APIs.

V. SIMULATION EXPERIMENT AND RESULTS

In this section, we compare our offloaded dynamics method from Section IV with SOTA. The SOTA simulation was done using the CAT Vehicle Testbed simulator.

		RMS Error with the SOTA Approach				RMS Error with Offloaded Dynamics			
		Slow Machine		Fast Machine		Slow Machine		Fast Machine	
		Sim 1	Sim 2	Sim 1	Sim 2	Sim 1	Sim 2	Sim 1	Sim 2
Slow Machine	Sim 1		0.953	14.746	16.40		0.396	0.362	0.387
	Sim 2	0.953		15.830	14.110	0.396		0.384	0.369
Fast Machine	Sim 1	14.746	15.830			0.362	0.384		0.384
	Sim 2	16.40	14.110	3.338		0.387	0.369	0.384	

TABLE I. A comparison of RMS error E across simulations for the SOTA approach and modified approach with offloaded dynamics.

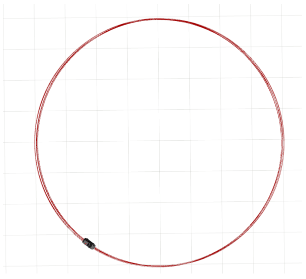

FIG. 8. A snapshot of the trajectory of the vehicle under fixed velocity and steering angle control with the SOTA approach. The simulation was run for 300s. The circumference of the circular trajectory was 230 m.

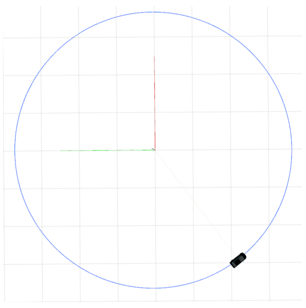

FIG. 9. A snapshot of the trajectory of the vehicle under fixed velocity and steering angle control with offloaded dynamics. The simulation was run for 300s. The circumference of the circular trajectory was 230 m.

Next, we conducted a set of simulations with offloaded vehicle dynamics $\hat{f}^*$ using Equation (6) and control input u without feedback. Gazebo takes updated $\hat{\mathbf{x}}_k^*$ and sets the position of each vehicle in the world frame of reference. The simulation experiment was conducted using a high-level python API that we wrote to automate the overall simulation. At the API level, the simulation expects the circumference of trajectory, the desired number of vehicles n to simulate, vehicle model $\hat{f}$ or $\hat{f}^*$ and controller, h, ρ, ΔT for throttling controller output, log-time, and u_k. The API invokes appropriate ROS nodes, Gazebo, rosbag record for logging data, and controller. Upon termination of the simulation, we can analyze the log traces from rosbag files for repeatability.

A. Simulation Results: Assessing the Repeatability

In the first set of simulations, we set $u = [8.0\ \text{m/s}, 0.071\ \text{rad}]$ to control a single-vehicle $\hat{f}$ using SOTA approach and f^* using offloaded dynamics. We performed this simulation on two different computers, let us call them slow (4 GB RAM) and fast computers (64 GB RAM). We had a total of 8 simulations: two simulations for the SOTA approach using $\hat{f}$ and $\hat{g}$ and two for offloaded dynamics using $\hat{f}^*$ and $\hat{g}^*$ on each of the slow and fast computers. A snapshot of the trajectory for SOTA and offloaded dynamics captured on a slow computer is shown in Figure 8, and Figure 9. From the snapshot, it is evident that the trajectories of the vehicle do not overlap on several rotations with the SOTA approach (see also Figure 4) while with the newer approach, it overlaps to greater precision, as evident from the error plot in Figure 10. For a single vehicle simulation, we didn't observe any reduction in RTF in the simulation.

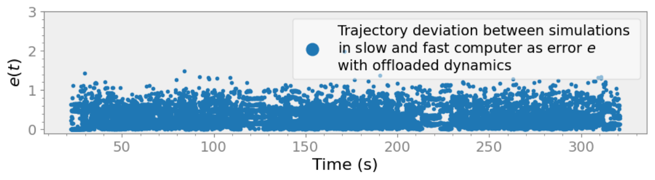

FIG. 10. Trajectory deviation as $e(t)$ between two simulation runs performed on two different computers with same simulation setup with offloaded dynamics. Compared to Figure 4, we see the error has dropped almost by 10 fold.

We computed error E as defined in Equation (5) to quantify how much deviations occur between different runs within a single machine and between different machines. Error E is tabulated in Table I for SOTA, and offloaded dynamics.

B. Scaling the Simualtion

We wanted to assess how many vehicles can be simulated at the same time without an automatic reduction in the RTF of the Gazebo. We saw the RTF deterioration on scaling in Figure 3 when executing on the fast machine with the SOTA approach. Simulating more than 4 vehicles in the slow machine results in the reduction of RTF and simulating anything beyond 12 vehicles renders the simulation unresponsive. Simulation of more than 14 vehicles results in an *out-of-memory* error. Specifying the RTF to 0.1 by adjusting the max-update rate and time-step in Gazebo also didn't help in simulating more cars on the slow machine with repeatable results.

To scale the simulation with repeatable results using offloaded dynamics, we first simulated in real-time with $h = 0.01$ s, $\rho = 100$ Hz. We found out that although the chaotic nature of simulation was mitigated to large extent (Figure 9), RTF would still fluctuate up to 0.7 even when specifying it to 1.0 for as many as 20 cars. However, resultant RTF and trajectories were consistent for all cars when we specified $h = 0.01$ s, $\rho = 10$ Hz (ten-times slower) and throttled the publish rate of each vehicle at $1/\Delta T = 20$ Hz. RTF remained at 0.1 for the entirety even after spawning as many as 20 vehicles. At this point, the scalability of the simulation is only limited by resources such as RAM and GPU.

C. Trajectory replication from the real-world data

The last simulation experiment we did was to use velocity data gathered from a real-world field experiment. The real-world data to be used came from the Arizona ring-road experiment where we deployed Followerstopper [13] to dampen phantom traffic waves. In the simulation, we spawned 21 vehicles $\hat{\mathbf{f}^*} \equiv \{\hat{f^*}_i\},\ i \in [1, 21]$ with $u_i = [v_i, 0.065 \text{ rad}]$, v_i were read from data files obtained during the field experiment [23]. We replicated the experimental condition by placing 21 vehicles equidistant on a 260 m circular track (Figure 11). We conducted a set of simulations to verify if simulated vehicles with given dynamics provide scalability and repeatability in terms of vehicle trajectories. As stated in Section I, the SOTA was not suitable enough to scale the simulation for such a task. Attempting to control vehicles with velocities obtained from the field experiment to vehicles in SOTA simulation didn't result in expected behavior and vehicles trajectories were unpredictable, some vehicles crashed into each other, some vehicles didn't receive commands to move at all. With offloaded dynamics, we were able to conduct simulation after setting $h = 0.01$ s, $\rho = 10$ Hz and $1/\Delta T = 30$ Hz. The resulting time-space diagram of vehicles from running the simulation twice is drawn in Figure 12. The red line in Figure 12 denotes the position of the leader vehicle over time while black lines denote the positions of all follower vehicles over time. With offloaded dynamics approach, we were able to achieve repeatable simulation with an error deviation of $E = 7.7$ m on an average per vehicle.

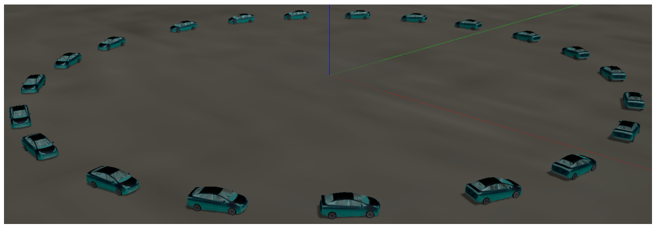

FIG. 11. Replicating Arizona ring-road experiment with offloaded dynamics in simulation

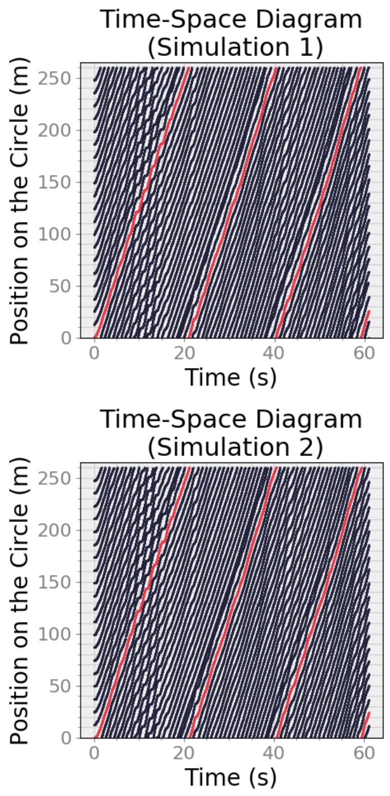

FIG. 12. Time-space diagram of all 21 vehicles obtained from simulation with experimentally-obtained velocity profiles from field data. We simulated the trajectory of vehicles in ROS-Gazebo using velocity profiles from Arizona Ring-road experiment data. Red line indicates position of leader vehicle in the platoon over time and black lines denote position of all other vehicles following the leader vehicle. We ran the simulation twice to assess repeatability. We used method of offloaded dynamics as described in this article with $h = 0.01$ s, $\rho = 10$ Hz and $1/\Delta T = 30$ Hz for throttling agent's state update. Each vehicle on an average had an error deviation of $E = 7.7$ m between two simulations.

VI. CONCLUSION AND FUTURE WORKS

In this work, we presented methods for achieving repeatable and scalable simulation for autonomous vehicles in ROS/Gazebo by offloading the state dynamics from the physics engine, synchronizing the agent's state update by a director, and specifying the simulation to run at a slower than real-time. Such a simulation system can provide a reliable platform for developing and testing CPS applications within the simulation, cut down the cost of logistics required for field tests, and provide safety assurance. During the simulation experiment, we didn't establish any formal methods for choosing ΔT or an appropriate RTF. We are looking to extend our proposed work to provide an algorithm to choose appropriate ΔT to optimize the RMS error and message loss. Further, in the upcoming work, we will present our use cases of feedback control, sensor-assisted driving, and scene interaction with offloaded dynamics.

Code and Data

The dataset used for the simulation, relevant code, the ROS package, python API used to implement the proposed method, and python notebooks with a pipeline of simulation experiments is listed at [24].

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant Numbers NSF CNS-1446435, 1446690, 1446702, 1446715 (joint with B. Piccoli, D. Work, B. Seibold), CNS-1253334 and CNS-1544395 (joint with R. Sanfelice).

[1] H. K. Khalil and J. W. Grizzle, *Nonlinear systems*, Vol. 3 (Prentice hall Upper Saddle River, NJ, 2002).

[2] M. Vidyasagar, *Nonlinear systems analysis*, Vol. 42 (Siam, 2002).

[3] L. Perko, "Nonlinear systems: Local theory," in *Differential Equations and Dynamical Systems* (Springer US, New York, NY, 1996) pp. 65–178.

[4] A. V. Kamyad, H. H. Mehne, and A. H. Borzabadi, Applied Mathematics and Computation **167**, 1041 (2005).

[5] R. J. Schilling, J. J. Carroll, and A. F. Al-Ajlouni, IEEE Transactions on neural networks **12**, 1 (2001).

[6] G. Palm, Biological Cybernetics **31**, 119 (1978).

[7] J. Bouvrie and B. Hamzi, SIAM Journal on Control and Optimization **55**, 2460 (2017).

[8] F. Amigoni, V. Schiaffonati, and M. Verdicchio, in *Methods and experimental techniques in computer engineering* (Springer, 2014) pp. 37–53.

[9] P. Shiakolas, K. Conrad, and T. Yih, International journal of modelling and simulation **22**, 245 (2002).

[10] D. Cumin, C. Chen, and A. F. Merry, Simulation in Healthcare **10**, 336 (2015).

[11] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, in *ICRA workshop on open source software*, Vol. 3 (Kobe, Japan, 2009) p. 5.

[12] R. E. Stern, S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, *et al.*, Transportation Research Part C **89**, 205 (2017).

[13] R. Bhadani, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work, in *57th IEEE Conference on Decision and Control*, Vol. 57 (IEEE, 2018).

[14] M. L. Delle Monache, T. Liard, A. Rat, R. Stern, R. Bhadani, B. Seibold, J. Sprinkle, D. B. Work, and B. Piccoli, in *Computational Intelligence and Optimization Methods for Control Engineering* (Springer, Cham, 2019) pp. 275–299.

[15] G. Gunter, D. Gloudemans, R. E. Stern, S. McQuade, R. Bhadani, M. Bunting, M. L. D. Monache, B. Seibold, J. Sprinkle, B. Piccoli, and D. B. Work, IEEE Transactions on Intelligent Transportation Systems , 12 pages (2020).

[16] R. Bhadani, J. Sprinkle, and M. Bunting, in *Proceedings 2nd International Workshop on Safe Control of Autonomous Vehicles (SCAV 2018), Porto, Portugal, 10th April 2018, Electronic Proceedings in Theoretical Computer Science 269*, Vol. 269 (2018) pp. 32–47.

[17] G. Chance, A. Ghobrial, K. McAreavey, S. Lemaignan, T. Pipe, and K. Eder, arXiv preprint arXiv:2104.06262 (2021).

[18] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2018) pp. 2575–2582.

[19] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, in *Fundamentals of traffic simulation* (Springer, 2010) pp. 173–232.

[20] E. A. Lee, M. Niknami, T. S. Nouidui, and M. Wetter, in *2015 International Conference on Embedded Software (EMSOFT)* (IEEE, 2015) pp. 115–124.

[21] N. Koenig and A. Howard, in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3 (IEEE, 2004) pp. 2149–2154.

[22] K. Zhang, J. Sprinkle, and R. G. Sanfelice, IEEE Transactions on Automation Science and Engineering **13**, 479 (2016).

[23] F. Wu, R. E. Stern, S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, F. Wu, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work, "The arizona ring experiments dataset (ared)," (2018).

[24] Bhadani, Rahul and Sprinkle, Jonathan, *Sparkle: Enabling multi-vehicle simulation with scalability and repeatability.*, Department of Electrical & Computer Engineering, The University of Arizona (2022), 0.1.