2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

SynChrono: A Scalable, Physics-Based Simulation Platform For Testing Groups of Autonomous Vehicles and/or Robots

Jay Taves, Asher Elmquist, Aaron Young, Radu Serban and Dan Negrut[1]

***Abstract*— This contribution is concerned with the topic of using simulation to understand the behavior of groups of mutually interacting autonomous vehicles (AVs) or robots engaged in traffic/maneuvers that involve coordinated operation. We outline the structure of a multi-agent simulator called SYNCHRONO and provide results pertaining to its scalability and ability to run real-time scenarios with humans in the loop. SYNCHRONO is a scalable multi-agent, high-fidelity environment whose purpose is that of testing AV and robot control strategies. Four main components make up the core of the simulation platform: a physics-based dynamics engine that can simulate rigid and compliant systems, fluid-solid interactions, and deformable terrains; a module that provides sensing simulation; an agent-to-agent communication server; dynamic virtual worlds, which host the interacting agents operating in a coordinated scenario. The platform provides a virtual proving ground that can be used to answer questions such as "what will an AV do when it skids on a patch of ice and moves one way while facing the other way?"; "is a new agent-control strategy robust enough to handle unforeseen circumstances?"; and "what is the effect of a loss of communication between agents engaged in a coordinated maneuver?". Full videos based on work in the paper are available at https://tinyurl.com/ChronoIROS2020 and additional descriptions on the particular version of software used is available at https://github.com/uwsbel/publications-data/tree/master/2020/IROS.**

I. INTRODUCTION AND RELATED WORK

Given the recent surge of interest in autonomous vehicles (AVs) and robotics, the tasks of testing and proving these agents are both important and daunting. Sending out unproven agents in the real-world can lead to deaths, injury, or destruction of property [1]. However, comprehensive physical testing is an expensive and time consuming process. Against this backdrop, being able to carry out testing of agent-control strategies in simulation before moving to real world testing is very desirable. SYNCHRONO is one of several platforms that seek to address this issue, two prominent examples being Carla and Gazebo [2], [3]. What sets SYNCHRONO apart are its physics-based dynamics engine and sensing simulation support. Specifically, it draws on a multi-physics simulation engine with support for rigid and flexible bodies, fluid-solid interaction, deformable terrains, etc.; it is scalable via distributed-memory parallel computing as enabled by the Message Passing Interface; it is faster than real time for multi-agent scenarios; can tap into a sensor simulation capability; and, it is open source. SYNCHRONO has a modular design built off the infrastructure provided by the CHRONO open-source simulation platform [4], [5]. Specifically, it uses the CHRONO::Vehicle [6] and CHRONO::Sensor modules, which provide the ability to test scenarios that: are highly transient; include multiple agents that share a dynamic environment; and could embed the human in the loop. The contribution is organized as follows. In the remainder of this section we outline earlier contributions in the field and comment on their scaling attributes. We then present an overview of the technologies that come together to make SYNCHRONO. Finally, we present two case studies that analyze the scaling performance of SYNCHRONO in representative scenarios.

[1]Department of Mechanical Engineering, University of Wisconsin–Madison, Madison WI, USA {jtaves, amelmquist, aryoung5, serban, negrut}@wisc.edu

A. Related Work

While there are several vehicle and robotics simulation platforms, SYNCHRONO occupies a niche in terms of realistic physics, scalability, sensing simulation, and the fact that it is open-source. Carla and USARSim [7], compromise on physics accuracy by using video game engines [8] to drive their simulations. While this may certainly be a reasonable and effective compromise for many scenarios of interest, SYNCHRONO has focused on fidelity in physics, driven by CHRONO::Vehicle, allowing it to be applicable to scenarios such as off-road mobility and limiting conditions where accurate physics are crucial [9]. An additional compromise in the simulation physics is found in the choices made by swarm-simulation platforms such as Stage [10]. In order to scale up to the massive amount of robots needed in a swarm scenario, platforms tailored towards supporting swarms often go the route of either 2D simulations, or 3D simulations with very simple robotics that can be scaled up to large numbers of agents. This is the case for the recently detailed Titan simulation library, where the GPU was used to simulate 100s of relatively simple soft robots [11]; Titan is physics-based and scalable, but it compromises by focusing in a relatively simple physics-subset. Finally, the simulation platforms that do provide realistic physics simulation fail to scale. In [12], it is shown that Gazebo displays quadratic scalability beyond a handful of robots. Additionally, the Gazebo-based simulator ParaGazebo showed good scaling results in up to a point, after which the simulator slowed down when the number of agents was increased as the cost of spawning additional threads got very high. The authors of ARGoS confirm this finding, stating that both Webots and Gazebo fail to scale meaningfully beyond a small number of robots [13]. ARGoS itself presents impressive scalability as a simulation management framework (not physics engine). It uses multiple threads to manage various simulation engines in parallel, showing scaling of up to 10^4 swarm robots.

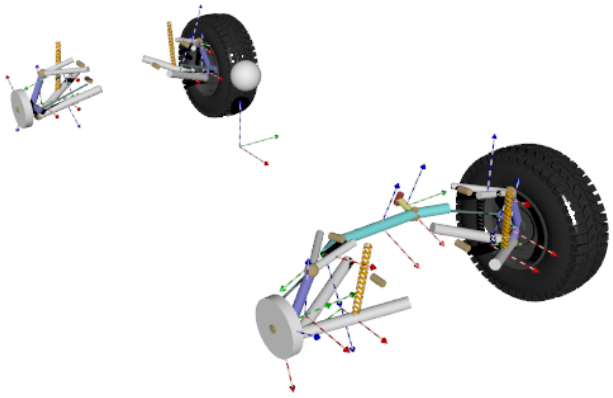

Fig. 1. CHRONO::Vehicle is used to create a vehicle model via subsystems rather than requiring construction of each body and constraint.

II. SOLUTION OVERVIEW

A. Multi-physics support

The vast majority of the robot/autonomous vehicle simulation platforms are built off video gaming environments, e.g., Unity, Unreal Engine, PhysX [14], [8], [15], thus emphasizing photorealism and benefiting from simplified creation of virtual worlds. While eye-pleasing, these platforms rely on simplified physics that can compromise the simulation-to-real-world transferability of any candidate design.

For SYNCHRONO, accurately simulating the agent dynamics represents a priority; i.e., capturing the effect of vehicle suspension, the interaction between the tires and the road, or tracks and deformable terrain, etc. The dynamics simulation engine builds on the CHRONO infrastructure, particularly on the CHRONO::Vehicle submodule, to accurately simulate agent dynamics and vehicle-ground interaction [6]. CHRONO allows the user to construct vehicle models from subsystem templates such as suspension type, tire model, drive line, power train and steering mechanism. An illustration of a vehicle generated with these subsystems is shown in Fig. 1.

The simulation platform captures the motion of multibody systems, when the time evolution (dynamics) of the systems is governed by a set of differential algebraic equations [16]

$$\dot{\mathbf{q}} = \mathbf{L}(\mathbf{q})\mathbf{v} \tag{1a}$$

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{f}\left(t, \mathbf{q}, \mathbf{v}\right) - \mathbf{g}_q^T(\mathbf{q}, t)\lambda \tag{1b}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{q}, t)\ . \tag{1c}$$

Above, Eq. (1b) simply states Newton's second law, with $\mathbf{f}$ the set of applied forces; $\mathbf{M}(\mathbf{q})$ the generalized mass; $\mathbf{q}$ the set of generalized positions; $\mathbf{v}$ the set of generalized positions, related to $\dot{\mathbf{q}}$ as in Eq. (1a); t is time; λ is the set of Lagrange multipliers associated with the kinematic constraint equations formulated in Eq. (1c). In the case of fluid-solid interaction problems, this set of equations is augmented with the momentum (Navier-Stokes) and mass balance equations, which are solved in a coupled fashion with the Newton-Euler equations of motion in Eq. (1) [17]. CHRONO is able to capture deformable terrain either by modeling it as a granular bed [18] or through the soil-contact model (SCM) [19].

B. Sensing support via Chrono::Sensor

For simulating perception, SYNCHRONO relies on the CHRONO::Sensor submodule which provides simulation of camera, lidar, GPS and IMU. The driving goal is to decrease the gap between what is sensed in the real world and what we can generate through sensor simulation. When vehicle sensors are noisy and distorting, the digital twin should recreate these same distortions and noise. As an example, GPS data is hindered by buildings and environmental factors, causing imperfect data in high density urban areas. The simulation should take into account these factors and geometries so that an algorithm that is dependent on GPS data is evaluated in a realistic manner [20]. The simulated sensor data can provide training data for machine learned object and lane recognition algorithms, and allow them to be tested in a reproducible virtual environment.

C. Synchronized multi-agent environments

A simulation platform is multi-agent if it enables a time and space coherent simulation of groups of AVs operating at the same time in a shared physical space. A common feature of multi-agent problems is that the dynamics of the agents are not strongly coupled. Indeed, the agents operate in a shared environment, but their evolution is coupled only through sensing (unless the agents collide). An example of strongly coupled dynamics is a fording operation, when the fluid changes the dynamics of the agent while the agent influences the dynamics of the fluid [17]. Although CHRONO handles strongly coupled dynamics, the interest in SYNCHRONO is in loosely coupled dynamics. Then, it is possible to run each agent's dynamics separately as one standalone process that advances its dynamics at a small time step Δt, and update other agents as to what has happened at a much slower $N : \Delta t$ rate, where, for instance, $N = 10$. Indeed, given that the only way agent B impacts agent A is through sensing, agent A does need to have *some* knowledge of agent B, but this knowledge can be updated at a slower frequency compared to the frequency at which the dynamics solution updates the state of agent A. This is the origin of the name SYNCHRONO: multiple CHRONO systems are run in parallel and only occasionally (relative to the CHRONO integration step) synchronized and updated as to what has happened in the separate CHRONO worlds. This allows the simulation as a whole to run more quickly and for each CHRONO system to be computationally separated, potentially spread across many nodes of a supercomputer.

SYNCHRONO uses the MPI framework [21] to achieve inter-agent synchronization. Each MPI rank manages a particular agent, that we refer to as the ego agent for that rank. The other agents in simulation (ego agents for a different rank) exist as "zombie agents" within the ego agent's simulation. Each MPI rank runs CHRONO dynamics for the ego agent every CHRONO timestep and receives information about the zombie agents at a slower frequency called the heartbeat. Rather than re-create the zombie agents as fully-fledged agents, the rank places them in the environment of the ego agent based on information communicated via MPI from the

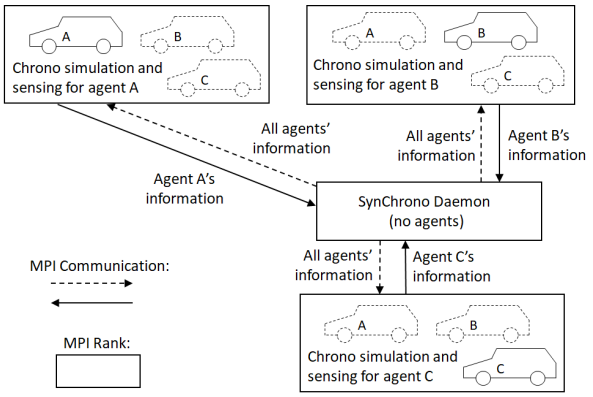

Fig. 2. Schematic of SYNCHRONO Inter-rank Synchronization. Each rank's CHRONO system only simulates one agent, all other agents are represented as static entities based on communicated state data.

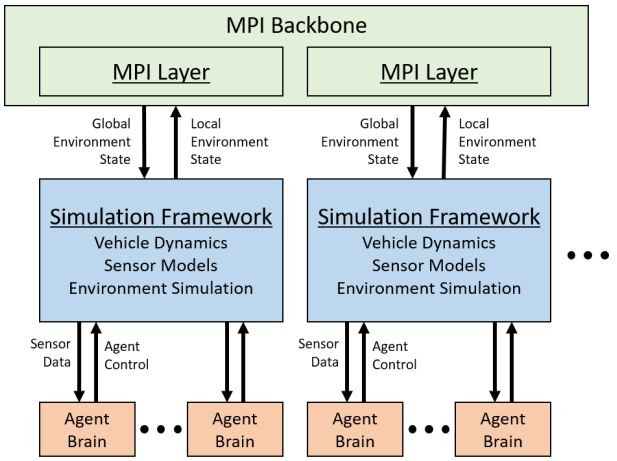

Fig. 3. Overview of SYNCHRONO MPI framework, each agent has a "brain" responsible for making decision and a connection to the MPI layer, for communicating its state to other agents.

rank that had that zombie agent as its ego agent. A schematic of this arrangement is shown in Fig. 2.

As an example, a sedan agent is fully-defined by the fact that it is a sedan agent, and by the states of its four wheels (their positions and rotation angles). When a sedan agent is implemented in SYNCHRONO code, it requires the specification of how to generate state data and how to re-create a zombie sedan from a set of state data. To generate state data, the sedan agent just has to query the underlying CHRONO system for the position and angle values that it requires. SYNCHRONO uses the Google Flatbuffers serialization library to communicate MPI state data between ranks. The main benefit of Flatbuffers is that it does not require any additional memory allocation, and access to the buffer doesn't require parsing a complicated schema.

Currently, SYNCHRONO uses a single centralized hub to synchronize all agent states. Rank 0, which runs the simulation manager, collects messages with state data from each agent and then broadcasts the state messages to all other agents. The simulation manager knows the type of each agent so that it knows what length of state message to expect from each agent. Knowledge of the length of state is the only necessary component for MPI message-passing, and the length of state is constant for a particular type of agent. Articulated buses, for instance, may have twice as much state data to communicate as a regular vehicle. This is all handled by the definition of the articulated bus agent, which specifies the length that its state messages will be.

Having received state messages from every agent, the central hub then broadcasts these state messages out to every other rank. Again, upon receiving state messages, each rank must know what type of agent the message was from, so that it can be properly de-serialized. So in addition to defining what state data must be passed via message, the definition of an agent also requires the definition of how a zombie agent should be constructed. For a vehicle this is as simple as placing static mesh elements in the world at the locations specified for the wheels and the chassis as dictated by the state message. A schematic of the MPI-based software architecture is shown in Fig. 3. The lowest level of the simulation is the control algorithm or brain of the agent which tells the agent what to do at each CHRONO timestep. This decision is fed into CHRONO::Vehicle and CHRONO::Sensor to determine what actually happens based on the agent's decision. The resulting state is passed to the MPI synchronization layer which propagates this updated state to all other ranks at the heartbeat frequency.

An additional concern for this synchronization is the tuning of the "heartbeat" parameter; the frequency at which each agent receives updates about the agents run by other MPI ranks. It is similar to the timestep chosen for physics simulation in that the shorter it is set to, the greater the realism of the simulation, which comes at a higher communication overhead. Since the heartbeat frequency is the rate at which messages are passed between ranks, it also determines the overhead incurred from using MPI. If the heartbeat step were incredibly large and agents never passed messages between each other, there would be no performance hit to the system and no matter how many agents were in the system, it would only take as long to compute as the time to compute the dynamics for a single CHRONO system. However, in this case agents would not be synchronized (time and space coherency would be lost) and they wouldn't sense each other. On the other hand, as the heartbeat step drops lower and lower, proportionally more time is spent passing state messages between the agents compared to the amount of time each node spends computing dynamics.

D. Generic interface control

SYNCHRONO's goal is to provide a testing environment for AV control policies, from car-following schemes to complete control pipelines. Each control algorithm is independent of SYNCHRONO, allowing the programmer to determine the action of the agent. SYNCHRONO has been developed with the goal of creating a flexible framework that will allow for users to easily test their own algorithms.

A possible use case for SYNCHRONO is to study highway dynamics, where flow rates and/or traffic shock waves are

analyzed. In this scenario, the control algorithm is not necessarily being studied, so the processing of sophisticated sensor data is not important. Instead, exact positions and velocities based on empirical models can be given directly to the agent. Such an approach is realistic when the behavior of a group of agents is of interest instead of that of a single vehicle. In the platoon example, the repercussions of a single vehicle lane change in a crowded highway scenario of multiple autonomous vehicles can be analyzed.

SYNCHRONO has been designed to enable human-in-the-loop (HIL) simulation. The most rudimentary form of human interaction is enabled via a keyboard or a simple steering wheel and pedals; the most sophisticated form is full-sized driving simulators such as at the National Advanced Driving Simulator [22]. With HIL, SYNCHRONO has to run in real-time to provide a realistic experience for the human doing the driving.

For SYNCHRONO to be a viable simulation framework for testing AV control policies, it is important that going from simulation to reality is seamless. For situations where the developed pipeline is meant to be retrofitted on a real-life vehicle, this feature is imperative. To this end, an interface has been developed which promotes a "drag-and-drop" style transition. Using TCP as the means of communication, SYNCHRONO can send information about the simulated environment to a receiver outside of the MPI network. In this case, a controller can be used in a framework like Robot Operating System (ROS). With the use of this interface, a complete control pipeline independent of the SYNCHRONO structure can be run with inputs replicating those from reality, such as sensor data or V2X communication.

III. DEMONSTRATION OF TECHNOLOGY

A. Multi-agent scenarios

As discussed in section II-C, a distributed simulation framework has the potential to reduce computation time since each computing node or core only has a single CHRONO vehicle that it simulates. If agents are added at the same rate as computational resources, then the only increase in runtime should be due to message-passing overhead as more messages move via MPI through the system. The scalability of SYNCHRONO is a main distinguishing factor, as it is capable of running multiple agents in parallel with little additional overhead. To quantify this, we compared the performance of a simulation of many vehicles in a single CHRONO system with the same simulation within SYNCHRONO using as many ranks as there were agents in the simulation.

While scenarios where agents are aggregating and acting upon large amounts of sensor data each time-step before making a decision are certainly important, our focus was on the computational overhead of the actual dynamics simulation itself in comparison with the MPI synchronization costs. We did not give the vehicles in either scenario any algorithm to run or sensor data to process, rather the simulation was simply tasked with simulating their agent's dynamics, and in the case of SYNCHRONO, synchronizing these states across ranks.

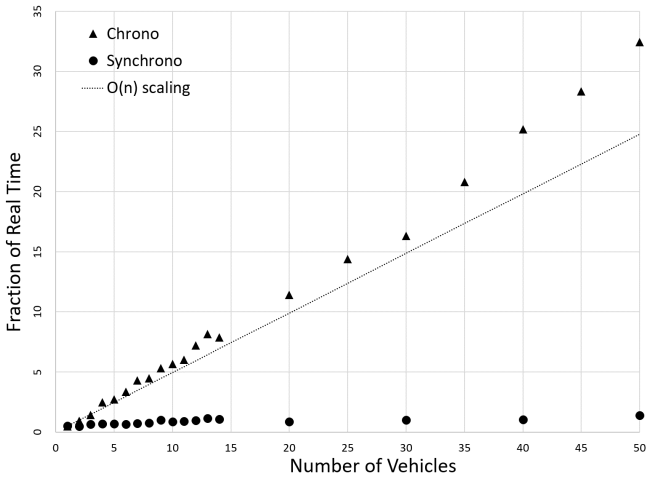

Fig. 4. Comparison between scalability of CHRONO and SYNCHRONO. For systems whose dynamics are not coupled, SYNCHRONO shows very low overhead for adding additional vehicles.

The scaling comparison is shown in Fig. 4 with the fraction of real time $\left(\frac{\text{Real-world runtime}}{\text{Amount of time simulated}}\right)$ plotted against the number of vehicles in simulation. While SYNCHRONO and CHRONO took the same amount of time to run a system with a single vehicle (as is expected since the system and resources are equal in both cases), SYNCHRONO scales much more efficiently given the task is a large decoupled system.

Also shown is a trend line that plots the expected time taken for a CHRONO system if its scaling were purely proportional to the number of vehicles. Note that the CHRONO simulation deviates from this line, indicating that its scaling performance is close, but slightly worse than $\mathcal{O}(n)$; i.e., linear increase in time relative to the number of vehicles in the experiment. While the SYNCHRONO simulation time does increase with the number of vehicles, this is due to the message passing overhead rather than the increase in system size. This resulted in only a small decrease in performance, remaining close to real time even for 50 vehicles. Also of note is that these simulations were not strictly equivalent comparisons: the CHRONO system ran on a box mesh whereas the SYNCHRONO version ran on the large Park St. mesh, a difference that handicaps the SYNCHRONO version, but will not affect the order of scaling analysis for SYNCHRONO.

B. Urban vs. Highway Scenarios

To both demonstrate the capabilities of SYNCHRONO and to quantify its performance characteristics, we focused on two main scenarios that we considered representative of the type of problems likely to be investigated with SYNCHRONO. The first was a highway platoon scenario, which is representative of simulations where group dynamics are of interest and the sensors and algorithms for any particular vehicle can be simple. The second is a more complex urban scenario, shown in Fig. 5, representative of simulations where individual dynamics are of interest and the sensors, algorithms and environment are more complex.

Fig. 5. Second scenario; Vehicles navigating the Park St. intersection by communicating with a traffic light agent that controls the state of the intersection. This represents an urban environment where individual vehicle dynamics are more important.

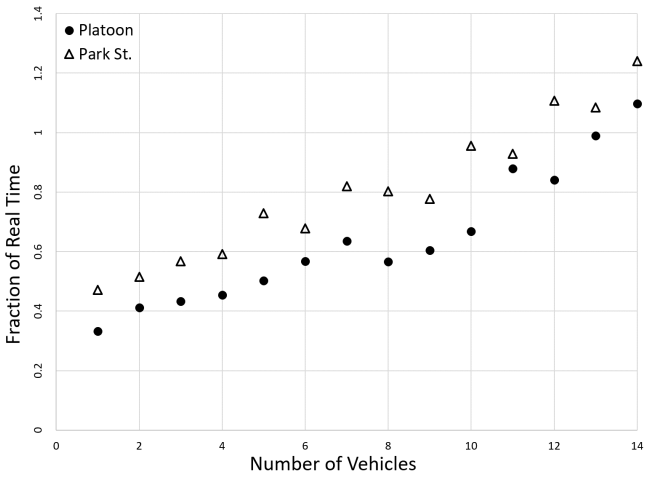

Fig. 6. Scaling Comparison Between Platoon and Park St. Scenarios. Both had one camera but the larger mesh in the Park St. scenario made it somewhat slower.

The platoon simulation was run on a uniform flat plane, with no texture or variation of any sort in the surface. Vehicles start in rows, three vehicles per row, meant to mimic a three-lane highway where the length is much longer than the width and variation in the surface is not the primary interest.

The second simulation was run on a mesh of an intersection near the UW–Madison campus, called the Park St. mesh. The mesh itself is rather large, taking up over 2GB of memory on the GPU. Vehicles start in one of five different lanes around the intersection and follow curves of GPS points through the city. They communicate with a single traffic light agent to determine the color of the light and if they are in a position to stop. Such a scenario represents a starting point for research into more sophisticated algorithms for a single ego agent in a world of well-behaved vehicles.

Both of these scenarios are much more scalable than their purely CHRONO counterparts, but the Park St. scenario in general takes longer to run than the platoon scenario due to the complexity of the mesh and the more sophisticated controls algorithms that the vehicles run, see Fig. 6. While the timing difference is not large, this gap will only widen for scenarios involving more complex sensors and more complex control algorithms.

C. Synchronization parameters

The heartbeat of the simulation's synchronization is a key parameter that can have a large impact on both the accuracy and speed of the simulation. First we'll focus on the sensing error introduced by choosing a heartbeat size that is too large. In the current implementation of SYNCHRONO, zombie vehicles remain stationary in the ego agent's world during the time between consecutive heartbeats. This will cause lag in the sensed data each time the simulation collects sensor data between heartbeats which will almost certainly happen since sensors will collect data at relatively arbitrary frequencies.

In general over a period between heartbeats, H_t, a vehicle will travel a distance of no more than $D_H = |a_{\max}|\, H_t^2/2 + |v_{\max}|\, H_t$ where $|a_{\max}|$ and $|v_{\max}|$ are respectively the maximum accelerations and velocities of the vehicle. D_H is also the maximum positional error of sensed data over the time period, if the sensor collects data immediately before the heartbeat updates. In "normal" operation, vehicles tend to travel at a constant velocity so the acceleration term can be neglected. This leads to a typical error that is directly proportional to the heartbeat frequency of the simulation, for vehicles traveling around $30\,\mathrm{m/s}$, with a heartbeat step of $0.01\,\mathrm{s}$, the error will be around $0.3\,\mathrm{m}$, which may or may not be acceptable, depending on the particular sensors that are included in the simulation and the particular scenario under study.

While having too infrequent a heartbeat frequency will introduce sensing error as zombie agents lag within the ego agent's world, the concern with having the heartbeat too frequently is that it could cause unnecessary overhead, decreasing performance. While this concern is relevant, preliminary tests have shown that a heartbeat frequency near the frequency of the simulation is still negligible compared to the CHRONO computation step for the simulations demonstrated in this paper. Further work will be done to explore this heartbeat frequency trade-off.

D. Real-Time performance

As a whole, CHRONO focuses on high-fidelity models for off-line simulation. For this reason, while performance is an important factor, accuracy of results trumps it. In SYNCHRONO on the other hand, running a simulation at real time is important as it allows for human-in-the-loop simulations where a human controls a conventional vehicle within a world of autonomous vehicles. Throttling a simulation that runs faster than real time is not a challenge, but speeding up simulations certainly is and it is critical to understand which types of simulations can be run in real-time and which cannot. In doing so, the answer is heavily dependent on the hardware running the SYNCHRONO simulations. While one

vehicle running on undeformable terrain is very likely to run in real-time in CHRONO, in SYNCHRONO the quality of the interconnect between the nodes running the CHRONO instance is paramount. The amount of data moved from the daemon to the vehicles is actually very low (hundreds of bytes, for undeformable terrain). Therefore, the latency of the interconnect is crucial in determining what can and cannot run in real time.

Currently, the real-time barrier lies around 10 vehicles (see Fig. 6), with the specific number depending on the hardware and configuration used. For some scenarios, such as those with a simple terrain and no sensors, tens of vehicles may co-exist in a simulation running in real-time, but simulations with more complex terrain/world and numerous sensors will need to be run slower than real-time and cannot have a human in the loop with more than a couple vehicles. In this context, SYNCHRONO experiments that use deformable terrain run at a real-time factor of approximately 100, see [9]. Future work will aim to increase the number of vehicles that can be run in simulation in real-time.

IV. CONCLUSIONS AND FUTURE WORK

This contribution introduced the multi-agent simulation framework SYNCHRONO, highlighted the features that distinguish it from other existing simulation solutions, and outlined preliminary scaling results that quantify its performance. SYNCHRONO builds off accurate physics-based modeling of vehicle dynamics and sensor data simulation through the CHRONO::Vehicle and CHRONO::Sensor modules, respectively. SYNCHRONO uses MPI to synchronize state data between various vehicle ranks, allowing the simulation to leverage multiple compute nodes. The additional overhead of message passing is relatively low with the overall simulation significantly more efficient than monolithic scaling in a shared CHRONO system.

There are several directions for future investigation. First, there are many enhancements that can be made to SYNCHRONO as a software platform, to make it more accessible to the user and more flexible in its configuration. Second, since the ultimate goal is to position SYNCHRONO as a control policy testing environment for multi-agent scenarios, we hope to validate the transferability of solutions developed and tested in SYNCHRONO to real-world scenarios; i.e., address the so-called sim-to-real gap.

ACKNOWLEDGMENTS

This work was partially supported by National Science Foundation grant CPS–1739869. The authors would also like to thank Continental Mapping of Sun Prairie, WI, for providing the virtual world replica of Park Street in Madison, WI. Additional support for the development of SYNCHRONO and CHRONO has been provided by the SAFER-SIM program, which is funded through a grant from the U.S. Department of Transportation's University Transportation Centers Program (69A3551747131).

REFERENCES

[1] The New York Times, "As U.S. Investigates Fatal Tesla Crash, Company Defends Autopilot System." http://www.nytimes.com/2016/07/13/business/tesla-autopilot-fatal-crash-investigation.html. Accessed: 2016-09-09.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[3] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator.," in *IROS*, vol. 4, pp. 2149–2154, Citeseer, 2004.

[4] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering – Lecture Notes in Computer Science* (T. Kozubek, ed.), pp. 19–49, Springer, 2016.

[5] Project Chrono, "Chrono: An Open Source Framework for the Physics-Based Simulation of Dynamic Systems." http://projectchrono.org, 2020. Accessed: 2020-03-03.

[6] R. Serban, M. Taylor, D. Negrut, and A. Tasora, "Chrono::Vehicle Template-Based Ground Vehicle Modeling and Simulation," *Intl. J. Veh. Performance*, vol. 5, no. 1, pp. 18–39, 2019.

[7] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1400–1405, IEEE, 2007.

[8] Epic Games, "Unreal Engine." https://www.unrealengine.com, 2020.

[9] D. Negrut, R. Serban, A. Elmquist, J. Taves, A. Young, A. Tasora, and S. Benatti, "Enabling Artificial Intelligence studies in off-road mobility through physics-based simulation of multi-agent scenarios," in *NDIA Ground Vehicle Systems Engineering and Technology Symposium*, 2020.

[10] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[11] J. Austin, R. Corrales-Fatou, S. Wyetzner, and H. Lipson, "Titan: A parallel asynchronous library for multi-agent and soft-body robotics using nvidia cuda," *arXiv preprint arXiv:1911.10274*, 2019.

[12] H. Yang and X. Wang, "A case study on the performance of gazebo with multi-core cpus," in *International Conference on Intelligent Robotics and Applications*, pp. 671–682, Springer, 2017.

[13] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, *et al.*, "Argos: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[14] Unity3D, "Main Website." https://unity3d.com/, 2016. Accessed: 2016-06-09.

[15] NVIDIA, "PhysX simulation engine." Available online at http://developer.nvidia.com/object/physx.html, 2019.

[16] E. J. Haug, *Computer-Aided Kinematics and Dynamics of Mechanical Systems Volume-I*. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

[17] H. Mazhar, A. Pazouki, M. Rakhsha, P. Jayakumar, and D. Negrut, "A differential variational approach for handling fluid-solid interaction problems via smoothed particle hydrodynamics," *Journal of Computational Physics*, vol. 371, pp. 92–119, 2018.

[18] A. M. Recuero, R. Serban, B. Peterson, H. Sugiyama, P. Jayakumar, and D. Negrut, "A high-fidelity approach for vehicle mobility simulation: Nonlinear finite element tires operating on granular material," *Journal of Terramechanics*, vol. 72, pp. 39 – 54, 2017.

[19] A. Tasora, D. Mangoni, D. Negrut, R. Serban, and P. Jayakumar, "Deformable soil with adaptive level of detail for tracked and wheeled vehicles," *International Journal of Vehicle Performance*, vol. 5, no. 1, pp. 60–76, 2019.

[20] A. Elmquist and D. Negrut, "Methods and models for simulating autonomous vehicle sensors," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[21] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard Version 3.0," 09 2012. Chapter author for Collective Communication, Process Topologies, and One Sided Communications.

[22] T. U. of Iowa, "The national advanced driving simulator."