SimNet: Learning Reactive Self-driving Simulations from Real-world Observations

Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen,
Chih Hu, Luca Del Pero, Błażej Osiński, Hugo Grimmett and Peter Ondruska*

***Abstract*— In this work we present an end-to-end trainable machine learning system capable of realistically simulating driving experiences. This can be used for verification of self-driving system performance without relying on expensive and time-consuming road testing. In particular, we frame the simulation problem as a Hidden Markov Process, leveraging deep neural networks to model both state distribution and dynamics. These are trainable directly from the existing raw observations without the need of any handcrafting in the form of plant or kinematic models. All that is needed is a dataset of historical traffic episodes. Our formulation allows the system to construct never seen scenes, imagine a range of possible futures, and react to the self-driving car's behaviour. We train our system directly from 1000 hours of driving logs and demonstrate significantly higher performance compared to the baseline methods in terms of realism and reactivity. To the best of our knowledge, this is the first work that directly merges highly realistic data-driven simulations with a closed loop evaluation for self-driving vehicles. We make the data, code, and pre-trained models publicly available to further stimulate simulation development.**

I. INTRODUCTION

Self-Driving Vehicles (SDVs) have the potential to radically transform society in the form of safe and efficient transportation. Modern machine learning methods have enabled much of the recent advances in self-driving perception [1], [2], [3], [4], [5], prediction [6], [7], [8], [9] and planning [10], [11]. The availability of large datasets unlocked significantly higher performance compared to older, hand-engineered systems.

However, the problem of validating SDV performance remains still largely unsolved. Most industry players validate empirically by deploying their self driving systems to a fleet of vehicles accompanied by safety drivers. In the case of unusual or failure behaviours, the safety driver takes over. Observed issues serve as feedback to improve the system. However, this process is both expensive and time-consuming, requiring the collection of thousands or even millions of miles, depending on the system's maturity. It is also hard to replicate or directly compare the performance of different system versions, as it is impossible to experience exactly the same driving situation twice.

A common approach to mitigate some of these issues is *log replay*, where the movement of other traffic participants is replayed around the SDV in simulation as it happened when the log was collected. However, if the SDV's new actions differ from those when the log was collected, the traffic participants don't react to it, and thus the simulation becomes unrealistic and ineffective for validation. For example, even a slight braking during the log replay can result in an unrealistic collision with the trailing car due to non-reactivity. These unrealistic outcomes are a result of what is called *simulation drift*.

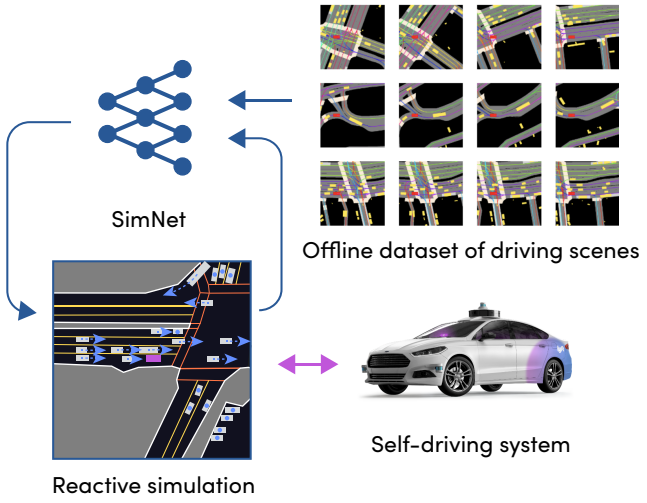

Fig. 1. The proposed trainable simulation system. We frame the simulation problem as reactive episode synthesis that can be used to validate the performance of a self-driving driving system.

One way to implement simulation reactivity is by scripting traffic participant behaviour to follow certain rules. However, this is time consuming and still lacks the realism and fidelity of road testing, thus undermining the validation effort.

In this paper, we aim to create realistic simulated driving experiences just like those that an SDV would encounter in the real world. For example, when the SDV decides to slow down, the simulated vehicle behind it should react by either slowing or overtaking, just as it would in a road test. Additionally, agent behaviour should capture the stochastic multi-modality that we observe on the road.

To achieve this, we frame simulation as an ML problem, in which we generate driving episodes that need to be both *realistic* and *reactive* to SDV behaviour. We then present a system leveraging high-capacity ML models trained on large amounts of historical data, Figure 1. Our approach significantly narrows the gap between road testing and offline simulation. To the best of our knowledge, this is the first work that connects simulation and planning with large-scale

* Authors are with Lyft Level 5 self-driving division. Contact: pondruska@lyft.com.

datasets in a realistic self-driving setting.

Our contributions are four-fold:

1) The formulation of the self-driving simulation problem as an ML problem which seeks to generate driving episodes that are both realistic and reactive to SDV behaviour;
2) A fully machine-learned simulation system that can sample these episodes based on historical driving data;
3) Qualitative and quantitative evaluations against baseline methods using 1000 hours of real-world data;
4) The code and the pretrained models of the experiments to further stimulate development in the community.

II. RELATED WORKS

Our work is situated in the broader context of trajectory prediction and simulation. Much research has focused on the accurate prediction of trajectories, as it is a crucial component of Model-Predictive Control (MPC). MPC, which is a commonly used framework in robotics and SDV, relies on a model to predict future states and choose the optimal one. Below is an overview of prediction models.

Classical methods to predict the behaviour of traffic participants include dynamic models [12], [13], [14], kinematic models [15], [16], [17], [18], [19], [20], Kalman filter-based systems [21], [22], Monte Carlo sampling [23], [24], [25] and trajectory prototypes [26], [27], [28], [29]. Today, deep learning methods for trajectory predictions are widely adopted. Notable examples include [6], [30], and [31]. Authors of [6] leverage bird's-eye view (BEV) rasters and a fixed set of future trajectory anchors. During training, the model learns displacement coefficients from those anchors along with uncertainties. TPNet [30] is a two stage network, where during the first stage the final future waypoint of the trajectory is predicted, starting from BEV semantic rasters. Then, proposals are generated to link past observations with this final waypoint and points near to it. Authors of [31] introduce the "LaneConv" operator to perform fast convolutions over the learned lane graph. LaneConv extends the traditional graph convolution operator by including relational information to all lanes in the frame which further increases performance.

Another line of work tackles trajectory prediction as a sequence modeling task. In this sense, Social LSTM [32] is a precursor in applying RNNs to model past and future trajectories of agents while considering inter-agent interaction. The authors employ a social pooling operator to share information between neighbouring sequences in the scene, which are modeled using different LSTMs. Several works stem from this approach, including Social GAN [33] and DESIRE [7], where the latter relies on a conditional Variational Autoencoders to model future uncertainty. In MATF [34] a recurrent encoder-decoder structure models both time and agent interactions. Scene context information is added to the encoder structure, which models past trajectories. Similarly, in [9] an encoder-decoder is employed in a probabilistic framework to learn future trajectories of agents. The model uses attention to exploit future and past agents' interactions. Attention between agents and lanes is also exploited in SAMMP [35]. In MANTRA [36], an associative memory is employed to store temporal embeddings. The memory is built during training using a recurrent encoder-decoder structure. Finally, [37] uses data about the agents' goals in order to improve predictions. Overal, these methods' crucial limitation is evaluation in a single time step and not forward-simulating into the future, which does not expose them to the challenge of error accumulation.

Our work intends to provide a closed-loop evaluation of SDV performance. To this end, it has to tackle the problem of error accumulation, which is currently under-explored in the SDV literature. One possible solution is to use an advanced driving simulator with agents controlled by hand-crafted rules. A notable example of such driving simulator is CARLA [38]. A disadvantage of this solution is that hand-coded actors tend to be unrealistic and rarely present a wide enough variety of behaviours. Another important work [39] deals with accumulating errors and presents a method trained entirely from data. A limitation of this model is that it directly predicts the visual output, which results in unnecessary errors, such as a car changing shape in consecutive frames. Our method avoids this problem by employing a fast, simple, and high-level feature representation. Crucially, [39] deals with a more limited task of predicting highway traffic and not a full range of driving situations, unlike our method.

III. SELF-DRIVING SIMULATION AS A LEARNING PROBLEM

In this section, we formulate the simulation problem as a machine learning problem. Specifically, we aim to sample realistic driving experiences that the SDV would encounter in the real world, that also react to the SDV behaviour given by its control policy f. To help model this realism, we have access to historical driving scenes D that the capture observed behaviour of other traffic participants on top of a semantic map $\mathcal{M}$ in a variety of diverse driving scenarios.

Each driving episode of length T can be described as a sequence of observed states $s_1, s_2, ..., s_T$ with each state capturing the position, rotation, size and speed of all nearby traffic participants z:

$$s_t = \{z_t^1, z_t^2, ..., z_t^k\}. \tag{1}$$

This representation corresponds exactly to the output of the perception system, which turns raw sensor measurements into the vectorised detections of traffic participants and can then be fed to the SDV's control algorithm.

The SDV itself is modelled simply as one of these participants z^{SDV}, but unlike other traffic participants its dynamics are controlled by a known function f implementing the self-driving algorithm:

$$z_{t+1}^{\text{SDV}} = f(z_t^{\text{SDV}}, s_t). \tag{2}$$

Generating new driving experiences can then be described as sampling from the joint distribution of the stochastic behaviour of other traffic participants and the deterministic SDV behaviour.

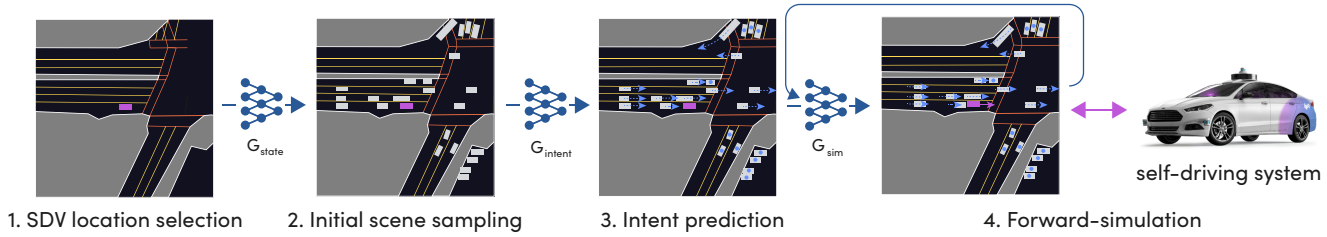

Fig. 2. Overview of the proposed simulation sampling process. To generate a new driving episode we first pick and sample an initial state capturing the positions of all traffic participants. Next, the latent intent of all participants is predicted capturing their long-term goals. Finally, the state is forward-simulated with the traffic participants controlled by a neural network and the behaviour of SDV controlled by a self-driving control loop.

Note that this joint distribution is inherently non-deterministic. Given an initial state s_1 there are many possible ways that the future can unfold. This is due to the hidden intents of traffic participants as well as process noise. At each moment in time, traffic participants must act and react to new information, such as attempts to merge, nudge, slow down, resulting in a complex set of driving behaviours. In the following section, we describe a method leveraging deep learning that can realistically sample such reactive episodes.

IV. GENERATING REALISTIC AND REACTIVE DRIVING EPISODES

In this section we describe an effective way to draw realistic and reactive driving episodes as defined in the previous section.

In particular, we assume that in a driving episode $s_1, ..., s_T$ each traffic participant z has a latent intent τ that captures both its destination and a path to get there, but without further specifics e.g. speed profile. For example, a driver's intent might specify that in order to reach home, they will change lanes and make a turn before pulling over. This intent affects their behaviour but is known only to the traffic participant themself, and is not directly observable by others.

The sampling of driving episodes can then be formalised as a Hidden Markov Model assuming this latent intent. The resulting probability distribution factorises as:

$$p(s_1, s_2, ..., s_T) = p(s_1) \prod_{i=1}^{K} p(\tau_i|s_1) \prod_{t=2}^{T} p(s_t|s_{t-1}, \tau_{1:K}). \tag{3}$$

Furthermore, we assume the actions are locally independent for each traffic participant:

$$p(s_t|s_{t-1}, \tau_{1:K}) = \prod_{k=1}^{K} p(z_t^k|s_{t-1}, \tau_k). \tag{4}$$

This follows the intuition of real-world driving where participants act independently, each controlling their own behaviour pursuing their own intents, observing others and reacting to new information without knowing the intents of others.

Each probability term in the initial state distribution $p(s_1)$, intent probability $p(\tau_i|s_1)$, and participant policy $p(z_t^k|s_{t-1}, \tau_k)$ is modelled by a separate neural network G_{state}, G_{intent} and G_{sim}. Using high-capacity neural network models results in accurate modelling of the state and transition probabilities necessary for a realistic simulation.

Sampling from this process consists of executing four steps as summarised in Figure 2, and outlined in detail in the next subsections:

1) Initial SDV location l is chosen from all permissible locations on the map;
2) Initial state s_1 is drawn from the distribution of all feasible states. This state captures the total number and initial poses of all traffic participants;
3) Intent trajectory τ_i is generated for each traffic participant z_i employing multi-modal prediction. This path describes the agent's intended destinations while allowing some flexibility of how to get there (i.e. acceleration), arising from interactions with other traffic participants and SDV;
4) Final driving episode $s_2, ...s_T$ is generated via step-by-step forward simulation employing the participant's policy $p(z_t^k|s_{t-1}, \tau_k)$, taking into consideration the individual intents and self-driving control system f.

This formulation offers a high degree of flexibility, allowing one to tailor the properties of the resulting simulation:

- **Full simulation:** Executing all above steps results in generating new, never-experienced driving episodes from all locations.
- **Journey simulation:** By keeping the initial SDV location l fixed, we can synthesise many different initial conditions and driving episodes starting at that position.
- **Scenario simulation:** By using an existing historical state of interest as s_1, we can generate many resulting possible futures.
- **Behaviour simulation:** We can replace intent prediction τ of particular agents by hard-coding a specific path for them to follow. This forces a particular high-level behaviour but still leaves flexibility in execution. This is useful for simulating SDV behaviour in specific situations, e.g. being cut-off by another car.

A. Initial state sampling

To represent the state s around the self-driving vehicle, we leverage a bird's-eye view representation rendering positions of nearby traffic participants on top of a semantic map $\mathcal{M}$. This representation has proven to be an effective representation in recent motion prediction and planning works [40], [10]. One advantage is that it effectively captures both local context and a variable amount of traffic participants in the form of a single image I_s.

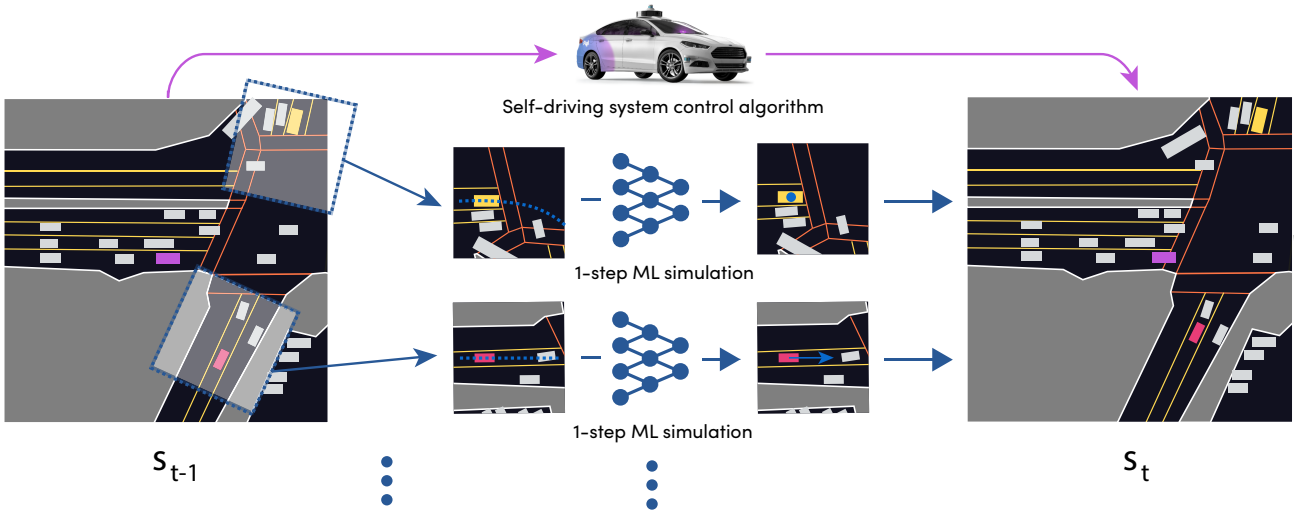

Fig. 3. Detail of the interactive state unroll. For all agents in the state s_{t-1} we independently run 1-step prediction to advance along their intent trajectory τ_i. The self-driving car is controlled by control algorithm f. The new positions then form a new state s_t and the process repeats.

To sample an initial state s_1 similar to our training distribution, we leverage conditional generative adversarial networks (cGANs) [41] conditioned on empty scenes capturing only the semantic map $I_{\mathcal{M}}$. This network is trained on pairs of $\{I_{\mathcal{M}}, I_s\}$ constituting the training dataset. Specifically, the generator network is trained to convert $I_{\mathcal{M}}$ into I_s and the discriminator to distinguish the synthetic states I_s from real ones. Upon convergence, the generator can create unlimited amounts of new synthesised states s_1 for any map location, indistinguishable from real ones, to seed the simulation.

To extract the final numerical positions and rotations of the vehicles $z_1, z_2, ...z_K \in s$, we use a connected components algorithm. For each connected component we compute the centroid and a minimum bounding box capturing its size.

B. Intent prediction

Given the initial state s_1, the next step is predicting the desired path τ_k for each vehicle z_k. This is the path the vehicle will follow in the next few seconds of the simulation and captures the vehicle intent.

For example, in the case of an intersection, we need to decide whether the vehicle wants to take a turn or continue straight. This depends on the vehicle's environment - i.e. only some lanes permit taking turns. Simultaneously, some paths are more likely than others.

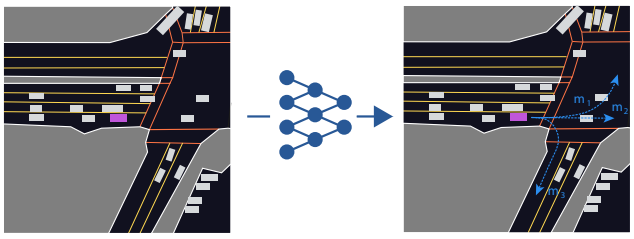

Fig. 4. An example of multi-modal prediction of the intent of a traffic participant. The simulated path is chosen according to the mode probabilities c_i.

We use multi-modal trajectory prediction for each vehicle, conditioned on the surrounding state to model this task:

$$p(\tau_k|s) = \sum_{m=1}^{M} c_m p(\tau_k|\mu_m), \tag{5}$$

where c_m is the mode probability and μ_m is the trajectory for that mode. We employ a model described in [40] leveraging a CNN architecture that takes bird's-eye-view rasterised states s_t around the traffic participants z_i as input, and produces M trajectories μ_m consisting of poses $x_1, y_1, x_2, y_2, ..., x_t, y_t$ and M mode probabilities c_m. The model is illustrated in Figure 4. The final path τ_k is then sampled from μ_m for each vehicle independently, according to the mode probabilities c_m. As different future intents can be sampled for the same vehicle, this results in differences in the simulated states.

We train the model on past agent trajectories. In particular, we take all traffic participants from the training dataset D with observation history longer than 3s and train the model to predict their individual future trajectory for 5s matching simulation episode length.

C. Forward simulation

Finally, with initial state s_1 and desired intents of all agents $\tau_{1:K}$, we generate the full sequence $s_2, s_3, ..., s_T$. This generation happens one step at a time, executing policy $p(z_t^k|s_{t-1}, \tau_k)$ for each traffic participant and SDV control policy f for the self-driving vehicle.

The traffic participant policy is similar to the intent prediction policy and uses a CNN network conditioned on the bird's-eye view of the state around the agent and trained on historical observations. There is, however, one important difference: instead of regressing over multi-modal distributions, the task is to predict only the velocity along the given intended path.

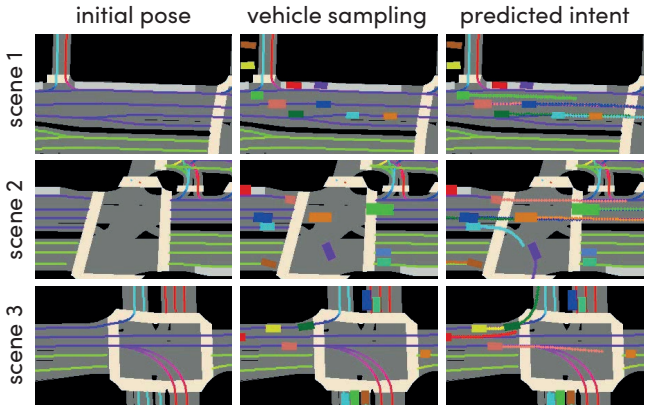

Fig. 5. An example of initial state and intended traffic participant trajectories. Each row shows a separate exemplar episode. From left to right: the initial location of ego, the sampled traffic participants' positions, and the predicted intent of each participant.

As illustrated in Figure 3, to compute a new state s_t the policy is invoked for every observed traffic participant z_k^t in the current state independently. Simultaneously, vehicle controls are triggered to obtain the SDV's new position:

$$z_t^{\text{SDV}} = f(s_{t-1}, z_{t-1}^{\text{SDV}}). \tag{6}$$

This then constitutes a new state s_t, and the process repeats until the entire episode is generated or simulation is interrupted, i.e. due to a simulated SDV collision.

V. EXPERIMENTS

Here we provide qualitative and quantitative evaluation of the proposed simulation system. In particular, we are interested in the system's ability to synthesise realistic initial states, forward-simulate full driving episodes, and to react to the SDV's behaviour. We capture these using two metrics:

Simulation realism: An average L2 distance of agents compared to their ground-truth positions at different time steps into the future (displacement). For this experiment we initialise the state from a real-world log, and the SDV follows exactly the same path as it did in that log. A perfect simulation system should be able to replicate the behaviour of other agents as it happens in the log.

Simulation reactivity: We measure collision rate in a simple scenario where the simulated SDV travels at half the speed as it did in the original log, or stays stopped when it should start moving. This slow driving behaviour should not cause collisions in reality, and requires trailing cars to react by slowing down or overtaking. Therefore, a perfect system should cause no such collisions.

A. Baseline methods

We compare our system to three simpler baseline methods:

- **Log replay:** A replay of the real-world episode replicating the exact behaviour of all traffic participants. This method achieves perfect simulation realism but due to its inability to react, achieves poor simulation reactivity.
- **Single-shot prediction:** A trajectory comprised of intent and velocity profile is predicted jointly. After the initial trajectory is predicted for each agent independently, this trajectory is followed exactly without taking any new information into account during the unroll. Unlike log replay, this method does not require knowledge of the future to work, but is not capable of reactive behaviour in response to SDV actions. This baseline evaluates efficiency of related motion forecasting models [7], [33], [40] for the purpose of simulation.
- **Iterative prediction:** Similarly as before we use single-shot future motion prediction for each agent, but this is done in each step of the simulation. Unlike our method, this is not conditioned on the initial predicted intent τ_i, but a new prediction is triggered in every step of simulation.

	Displacement [m]				
Method	0.5s	1s	5s	ADE	**Reactivity**
Log replay	-	-	-	-	✗
Single-shot	0.67	1.35	**5.93**	**2.65**	✗
Iterative	0.67	1.44	14.7	5.60	✓
Ours	**0.66**	**1.34**	6.16	2.72	✓

TABLE I

THE PROPERTIES OF THE DIFFERENT METHODS, MEASURING DISPLACEMENT AT DIFFERENT TIMESTEPS AND AVERAGED ACROSS ALL (LOWER IS BETTER), AS WELL AS REACTIVITY.

B. Dataset

We train and test our approach on the recently released Lyft Motion Prediction Dataset [42]. The dataset consists of more than 1,000 hours of dense traffic episodes captured from 20 self-driving vehicles. We follow the proposed train / validation / test split. We rasterise the high-definition semantic map included in the dataset to create bird's-eye view representations of the state, centered around each agent of interest to predict its future trajectories. This representation includes lanes, crosswalks and traffic light information. At crossings, lanes are not rendered if the controlling traffic light is red. We use rasters of size 224x224 in all our experiments. The model is trained on the train split of the dataset for 100k iterations, using a batch size of 64 and the Adam [43] optimizer. Additionally, we transform the predicted trajectories into a common space and increase the resolution for visualisation purposes.

C. Initial state sampling

A qualitative example of sampling the initial state and intended trajectory for each traffic participant is shown in Figure 5. We specifically focus on intersections as these situations give opportunity to traffic participants to make a variety of decisions. As one can see, the generated states and intents look realistic. Furthermore, learned trajectories effectively capture the variety of possible participant behaviours.

D. Forward simulation

In Figure 6 we qualitatively compare the realism of example episodes sampled by each method. Here, each method starts from an identical initial state, and we evaluate whether the final episode state is realistic or not, denoted by a red

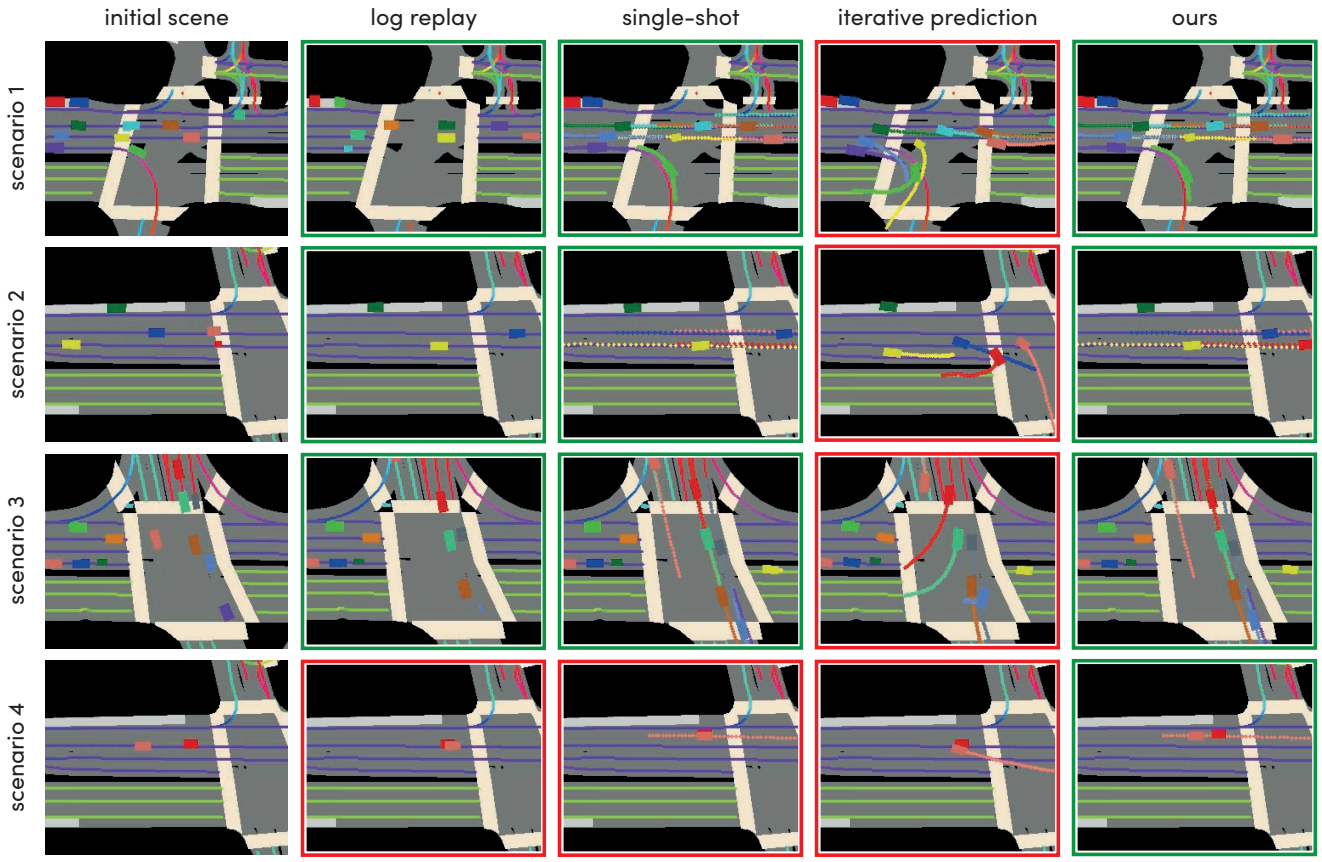

Fig. 6. Example scenario simulations of the proposed method and three baseline methods. We show the common initial state together with state of the simulation after 4 sec for each method. The colour of each final state indicates whether the outcome is plausible (green) or unrealistic (red). In scenario 4, we modify SDV behaviour to keep waiting at an intersection with a green traffic light. Only our method demonstrates realistic reactivity with trailing car slowing down and waiting without crashing into the SDV.

or green border. It can be seen how our proposed method, log replay, and the single-shot prediction can synthesise realistic states. This is also confirmed by Table I, where our method and the single-shot method outperform log replay and the iterative method in terms of realism score. This is due to single-shot directly optimising realism while sacrificing reactivity. On the other hand, unrolling the same model using iterative prediction results in states with agents diverging and going off-road. This is because every decision moves the vehicles slightly further from the training distribution and towards undefined behaviours. This is a well-known limitation of behavioural cloning and explains why the current methods for motion prediction in SDV model predictive control are unsuitable for simulation [44]. This effect is mitigated in our method as the vehicle is conditioned on the initial intent, which keeps it within the proximity of the training distribution during unrolling.

Finally, we investigate how well the generated episodes react to SDV behaviour. Scenario 4 of Figure 6 shows a synthesized sequence where the SDV stopped at an intersection instead of moving forward as it did in the log. Our method generates a realistic outcome whereby the trailing vehicle stops behind the SDV. Contrarily, both log replay and single-shot episodes end with collisions with the SDV. The trailing vehicle in the iterative prediction slows down but ultimately diverges as in other examples. In conclusion, only our proposed model can generate both realistic and reactive experiences. The simpler baseline methods sacrifice either reactivity or realism, or both.

VI. CONCLUSIONS

We have presented an end-to-end trainable machine learning system that generates simulations of on-road experiences for self-driving vehicles. The proposed method leverages large volumes of historical driving logs to synthesize new realistic and reactive driving episodes that can be used to validate SDV performance. The results of the evaluation show that the proposed method outperforms the commonly-used log-replay method and also methods based on motion prediction in terms of both realism and reactivity.

We believe this is an exciting step towards significantly decreasing the need of on-road testing in self-driving development, and the democratisation of the field. We hope the release of our system's source code will further stimulate development in ML simulation systems, leading them to become the core element of self-driving system development.

REFERENCES

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[5] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[6] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *Conference on Robot Learning*, 2020.

[7] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[9] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *Advances in Neural Information Processing Systems*, 2019.

[10] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[11] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] M. Brännström, E. Coelingh, and J. Sjöberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, 2010.

[13] C.-F. Lin, A. G. Ulsoy, and D. J. LeBlanc, "Vehicle dynamics and external disturbance estimation for vehicle path prediction," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 3, 2000.

[14] J. Huang and H.-S. Tan, "Vehicle future trajectory prediction with a dgps/ins-based positioning system," in *American Control Conference*, 2006.

[15] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *International Conference on Intelligent Computer Communication and Processing*, 2009.

[16] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 4, 2006.

[17] R. Miller and Q. Huang, "An adaptive peer-to-peer collision warning system," in *IEEE Vehicular Technology Conference*, 2002.

[18] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. Dietmayer, "Imm object tracking for high dynamic driving maneuvers," in *IEEE Intelligent Vehicles Symposium*, 2004.

[19] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *IEEE Intelligent Vehicles Symposium*, 2008.

[20] P. Lytrivis, G. Thomaidis, and A. Amditis, "Cooperative path prediction in vehicular environments," in *International IEEE Conference on Intelligent Transportation Systems*, 2008.

[21] H. Dyckmanns, R. Matthaei, M. Maurer, B. Lichte, J. Effertz, and D. Stüker, "Object tracking in urban intersections based on active use of a priori knowledge: Active interacting multi model filter," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[22] H. Veeraraghavan, N. Papanikolopoulos, and P. Schrater, "Deterministic sampling-based switching kalman filtering for vehicle tracking," in *IEEE Intelligent Transportation Systems Conference*, 2006.

[23] A. Broadhurst, S. Baker, and T. Kanade, "Monte carlo road safety reasoning," in *IEEE Proceedings. Intelligent Vehicles Symposium*, 2005.

[24] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using monte carlo sampling," *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 1, 2008.

[25] M. Althoff and A. Mergel, "Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, 2011.

[26] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *IEEE intelligent vehicles symposium*, 2009.

[27] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 9, 2006.

[28] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE transactions on intelligent transportation systems*, vol. 11, no. 3, 2010.

[29] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[30] L. Fang, Q. Jiang, J. Shi, and B. Zhou, "Tpnet: Trajectory proposal network for motion prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[31] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," *arXiv preprint arXiv:2007.13732*, 2020.

[32] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[33] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[34] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[35] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-head attention for multi-modal joint vehicle motion forecasting," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[36] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo, "Mantra: Memory augmented networks for multiple trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2019.

[38] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on Robot Learning*, 2017.

[39] M. Henaff, A. Canziani, and Y. LeCun, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," in *International Conference on Learning Representations*, 2018.

[40] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[41] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

[42] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," *arXiv preprint arXiv:2006.14480*, 2020.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[44] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.