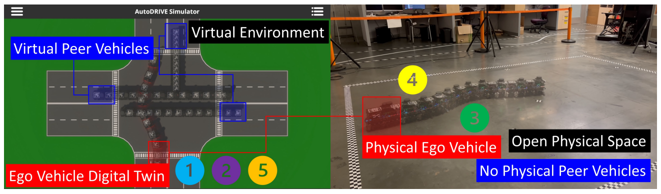
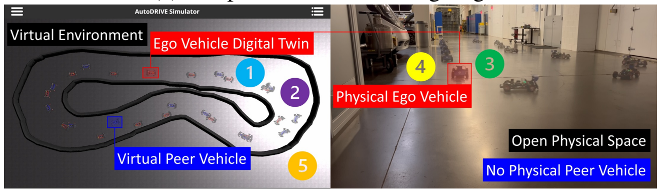
A Scalable and Parallelizable Digital Twin Framework for Sustainable Sim2Real Transition of Multi-Agent Reinforcement Learning Systems

Chinmay V. Samak*, Tanmay V. Samak* and Venkat N. Krovi

***Abstract*— Multi-agent reinforcement learning (MARL) systems usually require significantly long training times due to their inherent complexity. Furthermore, deploying them in the real world demands a feature-rich environment along with multiple embodied agents, which may not be feasible due to budget or space limitations, not to mention energy consumption and safety issues. This work tries to address these pain points by presenting a sustainable digital twin framework capable of accelerating MARL training by selectively scaling parallelized workloads on-demand, and transferring the trained policies from simulation to reality using minimal hardware resources. The applicability of the proposed digital twin framework is highlighted through two representative use cases, which cover cooperative as well as competitive classes of MARL problems. We study the effect of agent and environment parallelization on training time and that of systematic domain randomization on zero-shot sim2real transfer across both the case studies. Results indicate up to 76.3% reduction in training time with the proposed parallelization scheme and as low as 2.9% sim2real gap using the suggested deployment method.**



(a) Cooperative MARL using Nigel.



(b) Competitive MARL using F1TENTH.

Fig. 1: Resource-aware digital twin framework for sim2real transfer of multi-agent deep reinforcement learning systems: (1) observe, (2) decide, (3) act, (4) estimate, and (5) update.

I. Introduction

In the rapidly evolving landscape of connected and autonomous vehicles (CAVs), the pursuit of intelligent and adaptive driving systems has emerged as a formidable challenge. In such a milieu, multi-agent learning stands out as a promising avenue for developing autonomous vehicles capable of navigating complex and dynamic environments while considering the nature of interactions with their peers. Particularly, multi-agent reinforcement learning (MARL) offers the tantalizing potential of learning through self-exploration, which can potentially capture intricate cooperative/competitive multi-agent interactions.

Cooperative MARL [1]–[7] fosters an environment where autonomous vehicles collaborate and share information to accomplish collective objectives such as optimizing traffic flow, enhancing safety, and efficiently navigating road networks. It mirrors traffic situations where vehicles must work together, such as intersection management or platooning scenarios. Challenges in cooperative MARL include coordinating vehicle actions to minimize congestion, maintaining safety margins, and ensuring smooth interactions with peers.

On the other hand, competitive MARL [8]–[12] introduces elements of privacy and rivalry in autonomous driving, simulating scenarios such as overtaking, merging in congested traffic, or racing. In this paradigm, agents strive to outperform their counterparts, prioritizing individual success over coordination. Challenges in competitive MARL encompass strategic decision-making, opponent modeling, and adapting to aggressive driving behaviors while preserving safety.

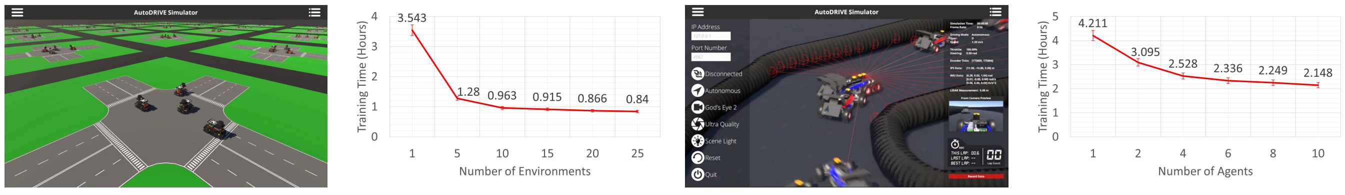
Irrespective of the problem formulation, one of the key challenges in training MARL policies is the sample efficiency of the environment, which translates to longer training times. This is often addressed by (a) training in low-fidelity simulations that can run faster than real-time [2]–[6], (b) parallelizing simulations to accelerate data collection [9], [10], or (c) both [12]. Adopting the first approach usually leads to a heightened sim2real gap, as marked in the literature through simulation-only deployments or explicit remarks for real-world experiments [12]. Adopting the second approach, on the other hand, usually requires extensive computational resources [9], [10], [13], which may not be sustainable. Additionally, none of the prior works addresses the challenge of selectively isolating collision, interaction, and perception between parallelized replicas of MARL systems.

Another important challenge arises during the real-world deployment of MARL policies. While generic RL has studied the sim2real transfer of trained policies using techniques such as domain adaptation [14]–[16], identification [17]–[19], or augmentation [19]–[22], the sim2real transfer of MARL systems has been typically under-explored. While recent works such as [7] and [12] filled this gap, they simpli-

*These authors contributed equally.

This work was supported in part by the U.S. National Science Foundation under NSF IIS-1925500 and NSF CNS-1939058.

C. V. Samak, T. V. Samak, and V. N. Krovi are with the Department of Automotive Engineering, Clemson University International Center for Automotive Research (CU-ICAR), Greenville, SC 29607, USA. Email: {csamak, tsamak, vkrovi}@clemson.edu



(a) (b) (c) (d)

Fig. 2: Simulation parallelization: (a) depicts a snapshot of 25 cooperative MARL environments training in parallel, and (b) denotes the training time for different levels of environment parallelization. Similarly, (c) depicts a snapshot of 10×2 competitive MARL agents training in parallel, and (d) denotes the training time for different levels of agent parallelization.

fied problem formulation by adopting extensive observation spaces with ground truth information about the environment and employed benchmark equipment such as mocap for real-time feedback during sim2real transfer. Additionally, these works used multiple physical vehicles, albeit scaled, within a synthetically constructed physical test environment for real-world deployments, which may not be sustainable.

This work tries to address these two MARL pain points through the following open-source[1] contributions:

- **Parallel Training:** This work contributes a modular simulation parallelization framework, which allows selectively isolating the exteroceptive perception, collision, and interaction within or among MARL system(s).
- **Sim2Real Transfer:** We introduce a bi-directional digital twinning framework to immerse a limited number of physical agents within a digital environment running virtual peer(s) to evaluate the MARL policies.
- **Case Studies:** This work presents a cooperative non-zero-sum use-case of intersection traversal and a competitive zero-sum use-case of head-to-head autonomous racing. The agents are provided with realistically sparse observation spaces and employ onboard state estimation for real-time feedback during sim2real transfer.
- **MARL Analysis:** The proposed MARL formulations are benchmarked against a decentralized reactive planning algorithm to assess their efficacy. We numerically evaluate the sim2real gap to analyze the effect of systematic domain randomization introduced in this work.

II. Formulation

A. Cooperative Multi-Agent Scenario

This section formulates a decentralized 4-agent collaborative scenario, wherein each agent's objective was to traverse a 2+2 lane, 4-way intersection without colliding or overstepping lane bounds. Each agent perceived its intrinsic states and received limited information from its peers (via V2V communication); no external sensing modalities were employed. Each agent was reset independently, resulting in highly stochastic initial conditions. The exact structure/map of the environment was not known to any agent. Consequently, this problem was framed as a partially observable Markov decision process (POMDP), which captured hidden state information through limited observations.

[1]GitHub: `https://github.com/autodrive-ecosystem`

1) Observation Space: Each agent, i $(0 < i < N)$, employed an appropriate subset of its sensor suite to collect observations: $o_t^i = \left[g^i, \tilde{p}^i, \tilde{\psi}^i, \tilde{v}^i\right]_t \in \mathbb{R}^{2+4(N-1)}$. This included IPS for positional coordinates $[p_x, p_y]_t \in \mathbb{R}^2$, IMU for yaw $\psi_t \in \mathbb{R}^1$, and incremental encoders for estimating vehicle velocity $v_t \in \mathbb{R}^1$. This follows that $g_t^i = \left[g_x^i - p_x^i, g_y^i - p_y^i\right]_t \in \mathbb{R}^2$ was the ego agent's goal location relative to itself, $\tilde{p}_t^i = \left[p_x^j - p_x^i, p_y^j - p_y^i\right]_t \in \mathbb{R}^{2(N-1)}$ was the position of every peer agent relative to the ego agent, $\tilde{\psi}_t^i = \psi_t^j - \psi_t^i \in \mathbb{R}^{N-1}$ was the yaw of every peer agent relative to the ego agent, and $\tilde{v}_t^i = v_t^j \in \mathbb{R}^{N-1}$ was the velocity of every peer agent. Here, i represents the ego agent and $j \in [0, N-1]$ represents every other (peer) agent.

2) Action Space: The Ackermann-steered vehicles were controlled using discrete throttle and steering commands: $a_t^i = \left[\tau_t^i, \delta_t^i\right] \in \mathbb{R}^2$. Here, the throttle command $\tau_t \in \{0.5, 1.0\}$ allowed the agents to slow down if necessary, and steering command $\delta_t \in \{-1, 0, 1\}$ allowed the agents to steer left, straight, or right for collision avoidance.

3) Reward Function: Extrinsic reward was formulated as:

$$r_t^i = \begin{cases} r_{goal} & \text{if safe traversal} \\ -k_p * \left\|g_t^i\right\|_2 & \text{if traffic violation} \\ k_r * \left(0.001 + \left\|g_t^i\right\|_2\right)^{-1} & \text{otherwise} \end{cases} \quad (1)$$

This function rewarded each agent with $r_{goal} = 1$ for successfully traversing the intersection and penalized them proportional to their distance from the goal, represented as $k_p * \left\|g_t^i\right\|_2$, for collisions or lane-boundary violations. The agents were also continuously rewarded inversely proportional to their distance-to-goal, to negotiate the sparse-reward problem. The reward (k_r) and penalty (k_p) constants were set to 0.01 and 0.425 respectively.

4) Optimization Problem: The extrinsic reward function motivated each agent to maximize its expected future discounted reward by learning an optimal policy $\pi_\theta^*\left(a_t|o_t\right)$.

$$\underset{\pi_\theta(a_t|o_t)}{\operatorname{argmax}} \; \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \quad (2)$$

B. Competitive Multi-Agent Scenario

This section formulates a 2-agent autonomous racing scenario, wherein each agent's objective was to minimize its lap time without colliding with the track or its opponent. Each agent collected its observations; no information was shared among the agents. The exact map of the environment was not known to any agent. Consequently, this problem was also framed as a POMDP. However, contrary to the cooperative MARL, this problem adopted a hybrid imitation-reinforcement learning architecture to guide the agents' exploration, thereby reducing training time. To this end, we recorded 5 laps worth of single-agent demonstrations for each agent by manually driving it in sub-optimal trajectories.

1) Observation Space: Each agent collected a vectorized observation: $o_t^i = \left[v_t^i, m_t^i\right] \in \mathbb{R}^{28}$. Here, $v_t^i \in \mathbb{R}^1$ represents the estimated forward velocity of i-th agent, and $m_t^i = \left[{}^1m_t^i, {}^2m_t^i, \cdots, {}^{27}m_t^i\right] \in \mathbb{R}^{27}$ is the LIDAR range array with 27 ranging measurements uniformly spaced 270° around each side of the heading vector, within a 10 m radius.

2) Action Space: The action space to control Ackermann-steered vehicles was $a_t^i = \left[\tau_t^i, \delta_t^i\right] \in \mathbb{R}^2$. Here, $\tau_t^i \in \{0.1, 0.5, 1.0\}$ represents the discrete throttle commands for torque-limited (85.6 N-m) actuation, and $\delta_t^i \in \{-1, 0, 1\}$ represents the left, straight, and right steering commands.

3) Reward Function: Following signals guided the agents:

- **Behavioral Cloning:** The behavioral cloning (BC) [23] algorithm updated the policy in a supervised fashion with respect to the recorded demonstrations, mutually exclusive of the reinforcement learning update.
- **GAIL Reward:** The generative adversarial imitation learning (GAIL) reward [24] ${}^g r_t$ ensured that the agent optimized its actions safely and ethically by rewarding proportional to the closeness of new observation-action pairs to those from the recorded demonstrations.
- **Curiosity Reward:** The curiosity reward [25] ${}^c r_t$ promoted exploration by rewarding proportional to the difference in predicted and actual encoded observations.
- **Extrinsic Reward:** The objectives of lap time reduction and motion constraints were handled using an extrinsic reward function ${}^e r_t$. The agents received a reward of $r_{checkpoint} = 0.01$ for passing each of the 19 checkpoints c_i on the race track, $r_{lap} = 0.1$ upon completing a lap, $r_{best\ lap} = 0.7$ upon achieving a new best lap time, and a penalty of $r_{collision} = -1$ for colliding with the track bounds or peer agent (in which case both agents were penalized equally). Additionally, a continuous reward promoted higher velocities v_t.

$$ {}^e r_t^i = \begin{cases} r_{collision} & \text{if collision} \\ r_{checkpoint} & \text{if checkpoint passed} \\ r_{lap} & \text{if lap completed} \\ r_{best\ lap} & \text{if best lap time} \\ 0.01 * v_t^i & \text{otherwise} \end{cases} \tag{3} $$

4) Optimization Problem: The multi-objective problem of maximizing the expected future discounted reward while minimizing the behavioral cloning loss $\mathcal{L}_{BC}$ is defined as:

$$ \underset{\pi_\theta^i(a_t|o_t)}{\operatorname{argmax}} \quad \eta \left(\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i \right] \right) - (1-\eta)\mathcal{L}_{BC} \tag{4} $$

where η weighs the degree of imitation and reinforcement learning updates, and $r_t^i = {}^g r_t^i + {}^c r_t^i + {}^e r_t^i$.

III. Methodology

In this work, we adopted and adapted the AutoDRIVE Ecosystem [26], [27] to model, simulate, train, and deploy two MARL case studies. Particularly, we created analytical models of Nigel [28] and F1TENTH [29] complemented with data-driven system identification and calibration. Further details about vehicle, sensor, and environment modeling within AutoDRIVE Simulator [30], [31] are available in [32]. That being said, this section describes certain novel enhancements to AutoDRIVE Ecosystem introduced in this work.

A. Simulation Parallelization

From a computing perspective, AutoDRIVE Simulator was developed modularly using object-oriented programming (OOP) constructs. We built on this fact to implement a selectively scalable agent/environment parallelization architecture. The simulator was configured to take advantage of CPU multi-threading as well as GPU instancing (only if available) to efficiently parallelize various simulation objects and processes while maintaining cross-platform support. Following is an overview of the simulation parallelization schemes:

- **Parallel Instances:** Multiple simulation instances can be spun up to train families of multi-agent systems, each isolated within its own simulation instance. This is a brute-force parallelization technique, which can cause unnecessary computational overhead.
- **Parallel Environments:** Isolated agents can learn the same task in parallel environments, within the same simulation instance. This method can help train single/multiple agents in different environmental conditions, with slight variations in each environment.
- **Parallel Agents:** Parallel agents can learn the same task in the same environment, within the same simulation instance. The parallel agents may collide/perceive/interact with selective peers/opponents. Additionally, the parallel agents may or may not be exactly identical, thereby robustifying them against minor parametric variations.

B. Learning Architecture

We leverage the proximal policy optimization (PPO) algorithm [33] to train both the multi-agent systems described in this work; justification follows. PPO is an on-policy method, which is empirically equally competitive to its off-policy counterparts [34]. Additionally, PPO promotes stable and efficient learning by imposing 2 complementary constraints: (a) a clipped surrogate objective to control each action probability update, and (b) a KL divergence early stopping criteria to limit overall policy change.

In terms of policy updates, cooperative MARL uses the collective experience of all agents to update a common policy. Contrarily, competitive MARL uses the independent experience of each agent to update its individual policy. Nevertheless, in both cases, the parallelized agents contribute their experiences to update their respective herd's policy. This results in distributed sampling, which improves data collection speed and diversity, thereby increasing its correlation with the true state-action distribution and stabilizing training.

TABLE I: Training Configurations

PARAMETER DESCRIPTION	COOPERATIVE MARL	COMPETITIVE MARL
Hyperparameters		
Neural network architecture	3-layer FCNN $\times$ {128, Swish [35]}	
Batch size	64	64
Buffer size	1024	1024
Learning rate (α)	3e-4	3e-4
Learning rate schedule	Linear	Linear
Entropy regularization (β)	1e-3	1e-3
Policy update (ϵ)	2e-1	2e-1
Regularization parameter (λ)	9.8e-1	9.8e-1
Epochs	3	3
Maximum steps (n_{max})	1e6	1e6
Behavioral Cloning		
Strength (η)	–	5e-1
GAIL Reward		
Discount factor (${}^{g}\gamma$)	–	9.9e-1
Strength	–	1e-2
Encoding size	–	128
Learning rate (${}^{g}\alpha$)	–	3e-4
Curiosity Reward		
Discount factor (${}^{c}\gamma$)	–	9.9e-1
Strength	–	2e-2
Encoding size	–	256
Learning rate (${}^{c}\alpha$)	–	3e-4
Extrinsic Reward		
Discount factor (${}^{e}\gamma$)	–	9.9e-1
Strength	–	1.0

TABLE II: Domain Randomization

PARAMETER DESCRIPTION	COOPERATIVE MARL	COMPETITIVE MARL
Observation Noise		
Position (w_x^k, w_y^k)	$\xi \cdot N$(0,1e-4) m	–
Orientation (w_ψ^k)	$\xi \cdot N$(0,3.0625e-4) rad	–
Velocity (w_v^k)	$\xi \cdot N$(0,1e-4) m/s	$\xi \cdot N$(0,1e-4) m/s
LIDAR Scan (w_m^k)	–	$\xi \cdot N$(0,1e-6) m
Action Noise		
Throttle (w_τ^k)	$\xi \cdot N$(0,2.5e-3) norm%	$\xi \cdot N$(0,2.5e-3) norm%
Steering (w_δ^k)	$\xi \cdot N$(0,2.5e-3) norm%	$\xi \cdot N$(0,2.5e-3) norm%
Agent Dynamics		
Center of Mass ($w_{x_{cg}}^k$, $w_{y_{cg}}^k$, $w_{z_{cg}}^k$)	–	$\xi\cdot$[-5e-2:1.11e-2:5e-2] m
Suspension Stiffness (w_K^k)	–	$\xi\cdot$[-100:22.22:100] N/m
Tire Stiffness ($w_{c_\alpha}^k$)	–	$\xi\cdot$[-2.5:5.6e-1:2.5] N/rad
Environment Dynamics		
Surface Friction (w_μ^k)	$\xi\cdot$[-1e-1:8.33e-3:1e-1]	–
Communication Delay (w_d^k)	$\xi\cdot$[0:4.17e-4:1e-2] s	–

Table I hosts the detailed training configurations adopted for the cooperative as well as competitive MARL scenarios. The parameter values mentioned were arrived at by analyzing the agent(s)' behaviors to satisfy the intended objectives qualitatively, while also ensuring a stable learning process. All the trainings were carried out on a single laptop PC having 12th Gen Intel Core i9-12900H 2.50 GHz CPU, NVIDIA GeForce RTX 3080 Ti GPU, and 32.0 GB RAM.

C. Domain Randomization

We leveraged the simulation parallelization architecture to introduce systematic domain randomization [20] across k agent/environment replicas. This allowed us to maintain the solver consistency across simulation time steps while introducing dynamical perturbations, which is not explored in the literature. Table II hosts the detailed domain randomization parameters for the cooperative as well as competitive MARL scenarios. Particularly, since the cooperative MARL scenario was environment-parallelized, we vary the dynamics of each environment replica in this case. Contrarily, since the competitive MARL scenario was agent-parallelized, we vary the dynamics of each agent replica in this case. Additionally, in both cases, we also introduce noise in the agents' observations and actions at each time step. Here, the parameter ξ denotes the degree of domain randomization. In this work, we analyze the effect of no domain randomization (NDR), i.e. $\xi = 0$, low domain randomization (LDR), i.e. $\xi = 1$, and high domain randomization (HDR), i.e. $\xi = 2$.

D. Sustainable Sim2Real Transfer

Our work proposes a sustainable method of transferring the trained MARL policies from simulation to reality. The term *"sustainability"*, here specifically alludes to resource-awareness, where we aim to minimize the physical resources in deploying and validating MARL systems.

To this end, we present a bi-directional digital twinning framework, which establishes a real-time synchronization between the physical and virtual realms. Fig. 1 (captured at 1 Hz) depicts the sim2real transfer of MARL policies discussed in this work using the proposed digital twinning framework. Here, we deploy a single physical agent in an open space and connect it with its digital twin, which operates in a virtual environment along with other virtual agents to generate actions in the digital space. These action sequences are then relayed back to the physical agent to be executed in the real world. This way, we can exploit the real-world characteristics of vehicle dynamics and tire-road interactions while being resource-altruistic by augmenting environmental element(s) and peer agent(s) in the digital space.

IV. Results

We use the follow-the-gap method (FGM) [36] as a benchmark for the MARL policies discussed in this work. The choice of FGM was justified since (a) it is capable of negotiating static and dynamic obstacles, which is essential for multi-agent systems, (b) it can work with non-holonomic Ackermann-steered vehicles adopted in this work, and (c) it is a decentralized reactive algorithm making a solid case for *"apples-to-apples"* comparison with the MARL policies.

A. Cooperative Multi-Agent Scenario

1) Training and Simulation Parallelization: Fig. 3 depicts the key performance indicators (KPIs) used to analyze the cooperative MARL training without any domain randomization (i.e., NDR). It was observed that the agents took over 600k steps to understand the collective objective of safe intersection traversal. This is marked by a sustainable increase in the cumulative reward (from $\sim$3 to $\sim$8) as well as episode length (from $\sim$470 to $\sim$600 steps). This is also when the policy entropy (i.e., randomness) fluctuated significantly, signifying that the agents were still learning. After this initial exploration, the agents tried reward hacking by choosing to take a longer time to traverse the intersection. This is marked by an increase in the episode length (from $\sim$600 to $\sim$700

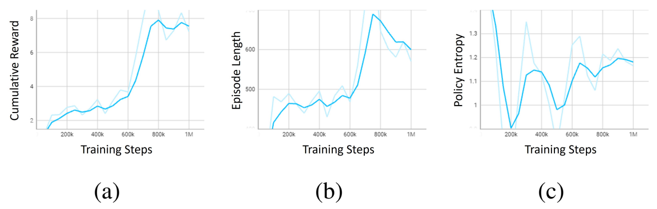


Fig. 3: Training results for cooperative MARL: (a) denotes cumulative reward, (b) denotes episode length, and (c) denotes policy entropy w.r.t. training steps.

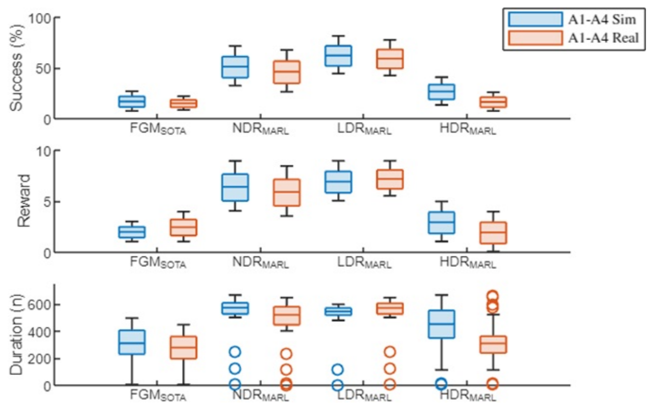


Fig. 4: Deployment and benchmarking of intersection traversal policies with 4 cooperative agents (A1-A4).

steps) between 700k and 750k steps. We anticipate this to be an effect of the last reward term, which continuously rewarded the agents inversely proportional to their distance from the goal. However, this phase was quickly overcome, since the probability of collision or lane boundary violations increased and the resulting reward was comparatively insignificant. By now, the policy entropy was starting to settle but was still fluctuating a little. Towards the end of 1M steps, the policy converged at a stable cumulative reward ($\sim$8) and episode length ($\sim$600 steps), while settling at a policy entropy of $\sim$1.2. For low- and high-domain randomization (i.e., LDR and HDR), the KPIs followed a similar trend but with increasing fluctuations (especially in policy entropy), owing to the randomized parameters.

From a computing perspective, we analyzed the effect of parallelizing the intersection-traversal environment from a single instance (4 agents) up to 25 instances (100 agents). As depicted in Fig. 2(a)-(b) the reduction in training time (up to 76.3%) was quite non-linear, with a saturating point approaching after 15-20 parallel environments.

2) Deployment and Sim2Real Transfer: The trained policies were first deployed and verified in simulation, where we observed interesting emergent behaviors among the agents. The agents strategically slowed down and/or steered away from each other to avoid collision.

Next, we quantitatively analyzed the policies trained with different grades of domain randomization (i.e., NDR, LDR, and HDR) and benchmarked them against FGM. The design of experiments followed 16 simulation runs and 16 real-world deployments, where the performance was assessed across 3 KPIs, viz. success rate, cumulative reward, and episode duration aggregated across all the agents (since this was a cooperative scenario) as depicted in Fig. 4. For real-world deployments, our experiments cycled across all the agents, such that each agent was physically deployed in the loop with the simulated environment comprising its virtual peers (refer Section III-D for implementation details).

It was observed that FGM was least successful ($<$30%) across simulation as well as real-world experiments. The same fact was reflected across the reward ($<$5 points) as well as duration ($<$500 steps) metrics. NDR-MARL performed better with mean success rates above 45%, which allowed it to cash in over 6 reward points on average. Here, the episode duration was between 400 and 650 steps in most cases, with outliers depicting early collisions. LDR-MARL was the most consistent with success rates reaching as high as 80%, rewards reaching as high as 9 points, and episode durations ranging between 500 and 600 steps in most of the cases. Lastly, it was observed that HDR-MARL performed poorer than other MARL configurations, where it could only achieve up to 40% success rate.

Finally, it is worth mentioning that the closeness between sim and real metrics effectively measures the sim2real gap, which was least for LDR-MARL (4.12%) followed by NDR-MARL (9.38%), FGM (16.01%), and HDR-MARL (33.85%) respectively.

B. Competitive Multi-Agent Scenario

1) Training and Simulation Parallelization: Fig. 5 depicts the KPIs used to analyze the competitive MARL training without any domain randomization (i.e., NDR). It was observed that the agents initially (until $\sim$200k steps) tried aggressive maneuvers, which mostly resulted in collisions. This is marked by the low extrinsic rewards ($<$50) and episode lengths ($<$1400 steps) in this phase. This can also be attributed to the higher BC loss ($>$0.1) as well as lower curiosity ($<$0.8) and GAIL ($<$9) rewards, indicating that the agents had not even started imitating the demonstrations correctly. However, the pre-recorded demonstrations soon (between $\sim$200k and $\sim$800k steps) guided the agents toward completing multiple laps around the race track. This is marked by an exponential reduction in the BC loss (from $\sim$0.1 to $\sim$0.025) and a progressive increase in the extrinsic (from $\sim$50 to $\sim$57), curiosity (from $\sim$0.8 to $\sim$1.1), and GAIL (from $\sim$9 to $\sim$11) rewards as well as the episode length (from $\sim$1400 to $\sim$1600 steps). It was interestingly observed that the red agent (Agent 1) dominated the blue one (Agent 2) till about 500 steps, after which the latter learned the "*competitive spirit*" and bridged the performance gap. Towards the end of 1M steps, both the policies converged at stable reward values and episode length, while gradually reducing the policy entropy from $>$0.3 to $<$0.05. Here, the non-zero offset in BC loss indicates that the agents did not over-fit the demonstrations; rather, they explored the state space quite well to maximize the extrinsic reward by adopting aggressive *"racing"* behaviors. The KPIs followed

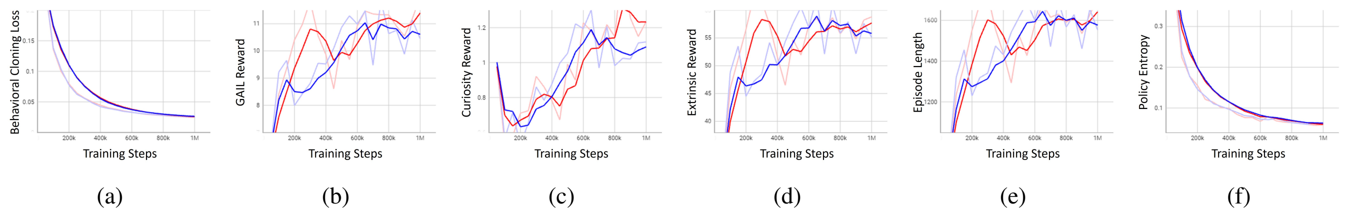


Fig. 5: Training results for competitive MARL: (a) denotes BC loss, (b) denotes GAIL reward, (c) denotes curiosity reward, (d) denotes extrinsic reward, (e) denotes episode length, and (f) denotes policy entropy w.r.t. training steps.

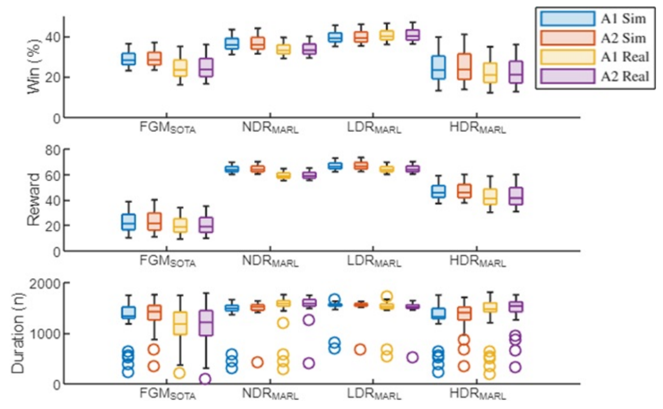


Fig. 6: Deployment and benchmarking of autonomous racing policies with 2 adversarial agents (A1 and A2).

a similar trend for LDR and HDR variations but with higher fluctuations (especially in policy entropy), owing to the randomized parameters.

From a computing perspective, we analyzed the effect of parallelizing the 2-agent adversarial racing family from a single instance (2 agents) up to 10 such families (20 agents) training in parallel, within the same environment. As observed from Fig. 2(c)-(d) the reduction in training time (up to 49%) was less dramatic in this case, with a saturating point approaching after 10×2 parallel agents.

2) Deployment and Sim2Real Transfer: The trained policies were first deployed and verified in simulation, where we observed interesting adversarial behaviors among the agents. One such behavior was a *block-block-overtake* sequence, and another was a *let-pass-and-overtake* sequence. These behaviors conveyed that the agents were explicitly competing, while implicitly coordinating to avoid collisions.

Next, we quantitatively analyzed the policies trained with different grades of domain randomization (i.e., NDR, LDR, and HDR) and benchmarked them against FGM. The design of experiments followed 16 simulation runs and 16 real-world deployments, where the performance was assessed across 3 KPIs, viz. win rate, cumulative reward, and episode duration separately for each agent (since this was a competitive scenario) as depicted in Fig. 6. For real-world deployments, our experiments cycled across all the agents, such that each agent was physically deployed in the loop with the simulated environment comprising its virtual peers (refer Section III-D for implementation details).

It was observed that both agents had the lowest average winning rate (<25%) with HDR-MARL, although they secured the least mean reward (<25 points) with FGM. This highlighted the difference between losing fairly and losing due to collision, which was also corroborated by the outliers in the duration metric. NDR-MARL provided higher (∼30-45%) winning consistency for either agent, but could not surpass that of LDR-MARL (∼35-47%). The same was reflected by the reward metric, wherein LDR-MARL cashed in slightly more reward (∼60-75 points) than NDR-MARL (∼55-70 points). Both performed equally well on the duration metric (∼1400-1700 steps), however, LDR-MARL was slightly more consistent with lower variance.

Finally, the sim2real gap, which was measured by averaging the difference between sim and real performance across all KPIs, was least for LDR-MARL (2.88%) followed by NDR-MARL (6.88%), HDR-MARL (8.98%), and FGM (13.48%) respectively.

V. Conclusion

This work identified two pain points in training and deploying MARL systems, and attempted to address them by proposing a scalable and parallelizable digital twin framework. Two representative case studies were formulated to support the claims: a 4-agent collaborative intersection traversal problem and a 2-agent adversarial head-to-head racing problem. The two problems were deliberately formulated with distinct observation spaces and reward functions, but more importantly, also the learning architecture (vanilla MARL vs. demonstration-guided MARL). We analyzed the training metrics in each case and also noted the non-linear effect of agent/environment parallelization on the training time, with a hardware/software-specific point of diminishing return. Finally, we presented a sustainable sim2real transfer of the trained policies using a single physical vehicle, which was immersed within the proposed digital twin framework to interact with its virtual peers in a virtual environment.

One of the bottlenecks of the proposed framework is its dependence on the state estimation and communication protocols used to update the digital twins. High estimation errors or communication delays can affect the MARL policies to not yield the right actions at the right time. We intend to investigate this in the future. Other avenues of research include the formulation of physics-guided MARL problems and scaling the deployments in terms of the number and size of the agents.

References

[1] S. Aradi, "Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2022.

[2] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent Motion Planning for Dense and Dynamic Environments via Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[3] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.

[4] D. Wang, H. Deng, and Z. Pan, "MRCDRL: Multi-Robot Coordination with Deep Reinforcement Learning," *Neurocomputing*, vol. 406, pp. 68–76, 2020.

[5] X. Zhou, P. Wu, H. Zhang, W. Guo, and Y. Liu, "Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning," *IEEE Access*, vol. 7, pp. 165 262–165 278, 2019.

[6] K. Sivanathan, B. K. Vinayagam, T. Samak, and C. Samak, "Decentralized Motion Planning for Multi-Robot Navigation using Deep Reinforcement Learning," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 2020, pp. 709–716.

[7] E. Candela, L. Parada, L. Marques, T.-A. Georgescu, Y. Demiris, and P. Angeloudis, "Transferring Multi-Agent Reinforcement Learning Policies for Autonomous Driving using Sim-to-Real," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 8814–8820.

[8] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[9] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.

[10] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, "Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9403–9409.

[11] C. V. Samak, T. V. Samak, and S. Kandhasamy, "Autonomous Racing using a Hybrid Imitation-Reinforcement Learning Architecture," 2021.

[12] P. Werner, T. Seyde, P. Drews, T. M. Balch, I. Gilitschenski, W. Schwarting, G. Rosman, S. Karaman, and D. Rus, "Dynamic Multi-Team Racing: Competitive Driving on 1/10-th Scale Vehicles via Learning in Simulation," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 1667–1685.

[13] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 91–100.

[14] J. Truong, S. Chernova, and D. Batra, "Bi-Directional Domain Adaptation for Sim2Real Transfer of Embodied Navigation Agents," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2634–2641, 2021.

[15] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[16] X. Wang, Y. Chen, and W. Zhu, "A Survey on Curriculum Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4555–4576, 2022.

[17] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[18] M. Kaspar, J. D. Muñoz Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4383–4388.

[19] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," 2018.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[21] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810.

[22] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep Drone Racing: From Simulation to Reality With Domain Randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2020.

[23] M. Bain and C. Sammut, "A Framework for Behavioural Cloning," in *Machine Intelligence 15*, 1995.

[24] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 4572–4580.

[25] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-Driven Exploration by Self-Supervised Prediction," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 2778–2787.

[26] T. Samak, C. Samak, S. Kandhasamy, V. Krovi, and M. Xie, "AutoDRIVE: A Comprehensive, Flexible and Integrated Digital Twin Ecosystem for Autonomous Driving Research & Education," *Robotics*, vol. 12, no. 3, p. 77, May 2023.

[27] T. V. Samak and C. V. Samak, "AutoDRIVE - Technical Report," 2022.

[28] C. V. Samak, T. V. Samak, J. M. Velni, and V. N. Krovi, "Nigel—Mechatronic Design and Robust Sim2Real Control of an Overactuated Autonomous Vehicle," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 4, pp. 2785–2793, 2024.

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

[30] T. V. Samak, C. V. Samak, and M. Xie, "AutoDRIVE Simulator: A Simulator for Scaled Autonomous Vehicle Research and Education," in *2021 2nd International Conference on Control, Robotics and Intelligent System*, ser. CCRIS'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–5.

[31] T. V. Samak and C. V. Samak, "AutoDRIVE Simulator - Technical Report," 2022.

[32] T. V. Samak, C. V. Samak, and V. N. Krovi, "Towards Validation of Autonomous Vehicles Across Scales using an Integrated Digital Twin Framework," in *2024 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2024, pp. 1068–1075.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017.

[34] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. WU, "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 24 611–24 624.

[35] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," 2017.

[36] V. Sezer and M. Gokasan, "A Novel Obstacle Avoidance Algorithm: "Follow the Gap Method"," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.