

Mixed-Reality Digital Twins: Leveraging the Physical and Virtual Worlds for Hybrid Sim2Real Transition of Multi-Agent Reinforcement Learning Policies

Chinmay Samak^{1b}, Graduate Student Member, IEEE, Tanmay Samak^{1b}, Graduate Student Member, IEEE, and Venkat Krovi^{1b}, Senior Member, IEEE

Abstract—Multi-agent reinforcement learning (MRL) for cyber-physical vehicle systems usually requires a significantly long training time due to their inherent complexity. Furthermore, deploying the trained policies in the real world demands a feature-rich environment along with multiple physical embodied agents, which may not be feasible due to monetary, physical, energy, or safety constraints. This work seeks to address these pain points by presenting a mixed-reality (MR) digital twin (DT) framework capable of: (i) boosting training speeds by selectively scaling parallelized simulation workloads on-demand, and (ii) immersing the MRL policies across hybrid simulation-to-reality (sim2real) experiments. The viability and performance of the proposed framework are highlighted through two representative use cases, which cover cooperative as well as competitive classes of MRL problems. We study the effect of: (i) agent and environment parallelization on training time, and (ii) systematic domain randomization on zero-shot sim2real transfer, across both case studies. Results indicate up to 76.3% reduction in training time with the proposed parallelization scheme and sim2real gap as low as 2.9% using the proposed deployment method.

Index Terms—Autonomous agents, reinforcement learning (RL), multi-robot systems, simulation and animation.

I. INTRODUCTION

CONNECTED autonomous vehicles are exemplars of cyber-physical systems co-existing with other agents, whose deployment presents a significant challenge due to the increased complexity of multi-agent interactions. 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 (MRL) offers the tantalizing potential of learning

through self-exploration, which can potentially capture intricate cooperative/competitive multi-agent interactions.

Cooperative MRL [1], [2], [3], [4] 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 MRL include coordinating vehicle actions to minimize congestion, maintaining safety margins, and ensuring smooth interactions with peers.

On the other hand, competitive MRL [5], [6], [7], [8] 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 MRL encompass strategic decision-making, opponent modeling, and adapting to aggressive driving behaviors while preserving safety.

In either case, one of the key challenges in MRL training is longer wall-clock times, which can be accelerated [9], [10] by (a) developing sample-efficient RL algorithms [11], or (b) accelerating simulations to improve data generation rate [12]. This work focuses on the latter aspect, which in turn can be addressed by (i) training in low-fidelity simulations that can run faster than real-time [1], [2], [3], (ii) parallelizing simulations to accelerate data collection [7], [8], or (iii) both [6]. Here, adopting low-fidelity simulations usually leads to a heightened sim2real gap, as marked in the literature through simulation-only deployments or explicit remarks for real-world experiments [6]. Contrarily, adopting brute-force parallelization usually requires extensive computational resources [7], [8], [13], which may not be sustainable – this is because existing frameworks cannot “*smartly*” parallelize replicas of multi-agent systems by selectively isolating collision, interaction, and perception between the agents/environments.

Another challenge arises when transitioning MRL policies from simulation to reality, either for training/fine-tuning or inference/evaluation. This is due to the requirement of multiple vehicles operating in a physical test environment, which may not always be feasible due to monetary, spatial, energy, or safety constraints. Consequently, there is a need to develop novel sim2real transfer techniques for MRL based on domain

Received 6 March 2025; accepted 15 July 2025. Date of publication 23 July 2025; date of current version 30 July 2025. This article was recommended for publication by Associate Editor A. Maria Metelli and Editor J. Kober upon evaluation of the reviewers’ comments. This work was supported by U.S. National Science Foundation under Grant NSF IIS-1925500, and Grant NSF CNS-1939058. (Chinmay Samak and Tanmay Samak contributed equally to this work.) (Corresponding author: Chinmay Samak.)

The authors are with the Department of Automotive Engineering, Clemson University International Center for Automotive Research (CU-ICAR), Greenville, SC 29607 USA (e-mail: csamak@clemson.edu; tsamak@clemson.edu; vkrovi@clemson.edu).

This article has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2025.3592085>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3592085

2377-3766 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See <https://www.ieee.org/publications/rights/index.html> for more information.

©2026 IEEE

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on February 21, 2026 at 01:43:20 UTC from IEEE Xplore. Restrictions apply.

adaptation [14], identification [15], or augmentation [16] methods, which have been explored in the context of single-agent RL. For example, [17] proposes an interesting approach of using pre-recorded data to approximate ego viewpoint with a synthetic vehicle rendered as an obstacle during training. However, they use a simple kinematic model for agent simulation, and require a full-scale physical vehicle during testing, which can be prohibitively expensive, difficult to set up, and potentially unsafe. More recently, [4], [6] have tried to address this gap, albeit partially, in the context of MARL. Particularly, [4] randomizes agent dynamics to mitigate the sim2real gap, but misses other potential domain randomization aspects like environment dynamics or agent observations/actions. Additionally, they do not consider simulation parallelization, and wait till the end of an episode to perform domain randomization. [6] proposes system identification and domain randomization to mitigate the sim2real gap. They parallelize low-fidelity simulation instances for faster training, but without considering selective isolation of perception, collisions, and interactions of the agents, and consequently, they too have to wait till the end of an episode to perform domain randomization. While these approaches are stepping stones toward MARL sim2real transfer, they both simplified the problem by adopting extensive observation spaces with ground-truth information about the environment and employing mocap for sim2real transfer, which made the experimental setup complex and expensive, with the trained policies heavily dependent on it. Additionally, both of them (typical of others in literature) used multiple physical vehicles, albeit scaled, in a synthetically constructed physical test environment for real-world deployments, which may not scale well.

Our work tries to address these MARL pain points by proposing and offering a unified set of open-source¹ tools for multi-agent robotics problems that were previously scattered and underexplored in related fields. The key contributions of this work can be summarized as follows:

- *Smart Parallelization*: We contribute a modular simulation parallelization framework, which allows selectively isolating the exteroceptive perception, collision, and interaction within or among MARL system(s).
- *Digital Twinning*: We introduce a bi-directional mixed-reality digital twinning framework to immerse a limited number of physical agents within a digital environment running virtual peer(s) to train/evaluate 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 case studies are benchmarked against a comparable set of baseline algorithms to assess their efficacy. We numerically evaluate the sim2real gap to analyze the effect of systematic domain randomization introduced in this work.

¹GitHub: <https://github.com/AutoDRIVE-Ecosystem/MRDT-MARL>

II. CASE STUDIES

A. Cooperative Multi-Agent Scenario

We formulated 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. This scenario is representative of a standard uncontrolled traffic intersection. Here, 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) *Observation Space*: Each agent, i ($0 < i < N$), employed an appropriate subset of its sensor suite to collect observations: $o_t^i = [g_t^i, \tilde{p}_t^i, \tilde{\psi}_t^i, \tilde{v}_t^i] \in \mathbb{R}^{2+4(N-1)}$. 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 $g_t^i = [g_x^i - p_x^i, g_y^i - p_y^i]_t \in \mathbb{R}^2$ was the ego agent's goal location relative to itself, $\tilde{p}_t^i = [p_x^j - p_x^i, p_y^j - p_y^i]_t \in \mathbb{R}^{2(N-1)}$ 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 throttle and steering commands: $a_t^i = [\tau_t^i, \delta_t^i] \in \mathbb{R}^2$. From a collision avoidance perspective, the throttle command allowed agents to speed up/down, and the steering command allowed agents to dodge their peers.

3) *Reward Function*: Extrinsic reward was formulated as:

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

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 $k_p * \|g_t^i\|_2$, 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 ($k_r = 0.01$) and penalty ($k_p = 0.425$) coefficients were tuned based on g_t^i to ensure that the agents incurred a unit positive or negative reinforcement (i.e., $\approx \pm 1$) at the end of an average episode, to ensure stable training and avoid reward hacking.

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

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

B. Competitive Multi-Agent Scenario

We formulated 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. This scenario is representative of a standard FITENTH (a.k.a. RoboRacer) autonomous racing competition [18]. Here, 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.

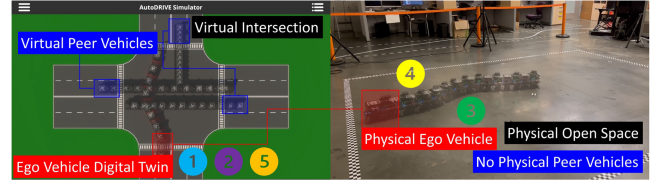
1) *Observation Space*: Each agent collected a vectorized observation: $o_t^i = [v_t^i, m_t^i] \in \mathbb{R}^{28}$. Here, $v_t^i \in \mathbb{R}^1$ represents the estimated forward velocity of i -th agent, and $m_t^i = [m_t^{i,1}, m_t^{i,2}, \dots, m_t^{i,27}] \in \mathbb{R}^{27}$ is the LIDAR range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

2) *Action Space*: The action space to control Ackermann-steered vehicles was $a_t^i = [\tau_t^i, \delta_t^i] \in \mathbb{R}^2$. Here, the throttle command allowed the agents to optimize their speed profile, and the steering command allowed them to optimize their race line, overtake their peers, and avoid collisions.

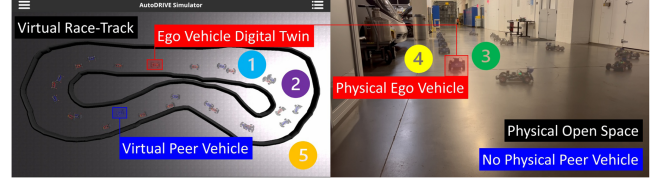
3) *Reward Function*: Following signals guided the agents:

- *Behavioral Cloning*: The behavioral cloning (BC) [19] 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 [20] $^g r_t$ ensured that the agent optimized its actions safely (mimicking safe demos) and ethically (avoiding proactive reward hacking) by promoting the closeness of new observation-action pairs to those from the recorded demonstrations.
- *Curiosity Reward*: The curiosity reward [21] $^c r_t$ promoted exploration by rewarding proportional to the difference in predicted and actual encoded observations.
- *Extrinsic Reward*: The objective of lap time reduction and incorporation of motion constraints was 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 .

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



(a) Cooperative MARL using Nigel.



(b) Competitive MARL using FITENTH.

Fig. 1. Proposed mixed-reality digital twin framework for hybrid sim2real transfer of MARL systems: (1) observe, (2) decide, (3) act, (4) estimate, and (5) update.

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:

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

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 [22] to model, simulate, train, and deploy two MARL case studies. This choice was driven based on the comparative analysis presented in [22], which satisfied all the requirements of this study. From a digital twinning perspective, data-driven system identification and calibration were used to customize models of Nigel [23] and FITENTH [24] vehicles from real-world data to ensure reliable simulation.

A. Simulation Parallelization

We leveraged the open-source nature of AutoDRIVE Simulator to implement a selectively scalable agent/environment parallelization framework. 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 instances of the simulator application 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

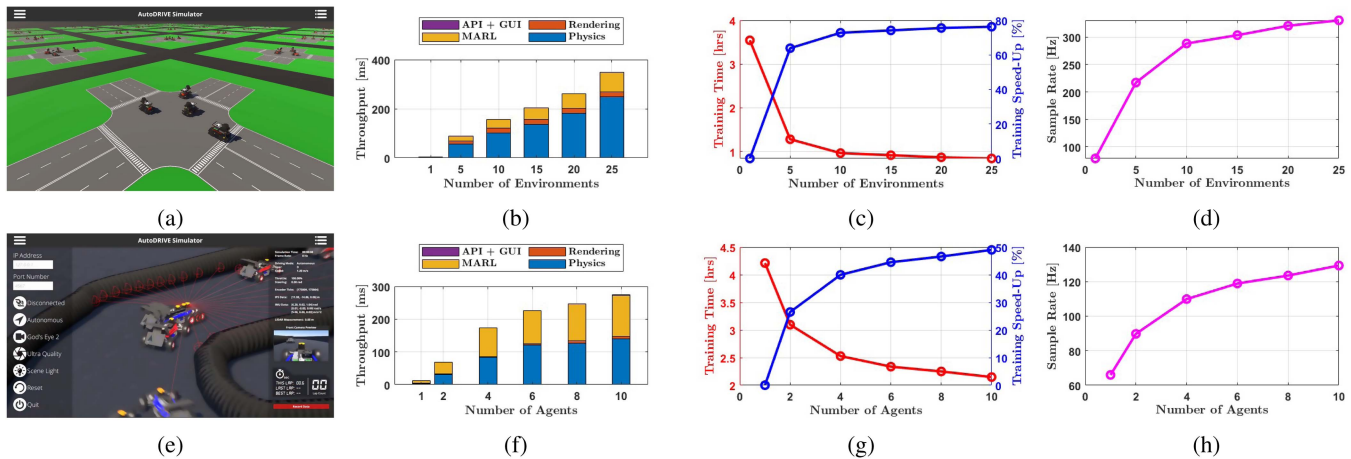


Fig. 2. Simulation parallelization: (a) 25×4 cooperative MARL agents analyzed for (b) computational throughput, (c) training time, and (d) sample rate across different levels of environment parallelization; (e) 10×2 competitive MARL agents analyzed for (f) computational throughput, (g) training time, and (h) sample rate across different levels of agent parallelization.

instance (refer Fig. 2(a)). 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 (refer Fig. 2(e)). 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.

It should be noted that parallelization beyond a certain point can hurt (i.e., point of diminishing return), wherein the increased simulation workload may slow down the training so much that parallel policy optimization can no longer accelerate it. This “saturation point” is dependent on the hardware/software configuration, and is subject to change.

B. Learning Architecture

While there is no limitation on RL algorithms to be employed within our framework, we leverage proximal policy optimization (PPO) [25] for our MARL case-studies. PPO is an on-policy method, which is empirically equally effective as its off-policy counterparts [26]. Moreover, 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 criterion to limit overall policy change.

In terms of policy updates, cooperative MARL uses the collective experience of all agents to update a common policy, a.k.a. centralized training and decentralized execution (CTDE) or multi-agent PPO (MAPPO). Contrarily, competitive MARL uses the independent experience of each agent to update its own policy, a.k.a. decentralized learning or independent PPO (IPPO), since the agents are in pure competition and not allowed to share any observations or experiences. 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

TABLE I
TRAINING CONFIGURATIONS

PARAMETER DESCRIPTION	COOPERATIVE MARL	COMPETITIVE MARL
Hyperparameters		
Neural network architecture*	3-layer FCNN \times {128, Swish [27]}	
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 (γ)	–	9.9e-1
Strength	–	1e-2
Encoding size	–	128
Learning rate (α)	–	3e-4
Curiosity Reward		
Discount factor (γ)	–	9.9e-1
Strength	–	2e-2
Encoding size	–	256
Learning rate (α)	–	3e-4
Extrinsic Reward		
Discount factor (γ)	9.9e-1	9.9e-1
Strength	1.0	1.0

*There are 2 identical networks: one acts as actor (policy) and the other as critic (value function).

(refer Fig. 2(d) and (h)), thereby increasing their correlation with the true state-action distribution, and stabilizing the training.

Table I hosts the detailed training configurations adopted for the cooperative as well as competitive MARL scenarios. The noted parameter values were arrived at by analyzing the agent(s)’ behaviors to satisfy the intended objectives qualitatively, while also ensuring a stable learning process. MARL training was carried out on a single laptop PC with 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 [16] across k agent/environment replicas. Particularly, each parallelized agent/environment was simulated with a different set of dynamical parameters during each episode (more diverse domain randomization in less time), and since the parallelized

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_τ^k)	$\xi \cdot [0: 4.17e-4: 1e-2]$ s	—

agents/environments were mutually isolated from each other, this did not interrupt numerical integration between simulation time steps (i.e., maintained solver consistency). This is a slightly different strategy compared to those explored in the literature: (i) per-interval domain randomization (e.g., [28]), which may violate solver consistency, or (ii) per-episode domain randomization (e.g., [4], [6], [29]), which may limit diversity of parameters, even if parallelized.

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. Mixed-Reality Digital Twinning

We propose a hybrid method for transferring the MARL policies from simulation to reality. The term “*hybrid*” specifically alludes to the mixed-reality digital twin framework, which establishes a real-time bi-directional synchronization between the physical and virtual worlds. The intention is to minimize the number of physical agent(s) and environmental element(s) while deploying and validating MARL systems in the real world. Fig. 1 (captured at 1 Hz) depicts the sim2real transfer of the trained MARL policies discussed in this work using the proposed framework while Fig. 3 (captured at 5 Hz) depicts the possibility of optionally training/fine-tuning MARL policies (e.g., if there is a significant modification in the real-world setup such as the deliberately introduced turf mat in our case) within the same framework (thereby minimizing the experimental setup while enjoying the benefits of real-world data for policy update). Here, we deploy a single physical agent in an open space and connect it with its digital twin. The “ego” digital twin operates in a virtual environment with virtual peers, collects observations,

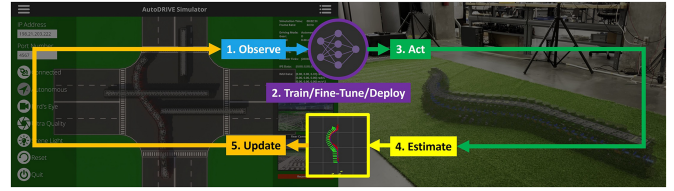


Fig. 3. Immersing MARL policies in the mixed-reality digital twin framework for training, fine-tuning, or deployment.

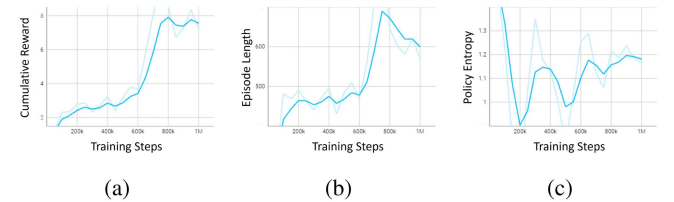


Fig. 4. Cooperative MARL training: (a) cumulative reward, (b) episode length, and (c) policy entropy w.r.t. training steps.

optimizes (optionally, during training/fine-tuning) and/or uses (during testing/inference) the MARL policy to plan actions in the digital space. The planned action sequences are relayed back to the physical twin to be executed in the real world, which updates its state in reality. Finally, the ego digital twin is updated based on real-time state estimates of its physical twin (estimated on board) to close the loop. This process is repeated recursively until the experiment is completed. 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. This also alleviates the safety concern of the experimental vehicles colliding with each other or the environmental element(s), especially as operational scales and number of agents increase.

IV. RESULTS

We benchmark MARL policies discussed in this work against 3 baselines. First, we choose follow-the-gap method (FGM) [30] as a common benchmark for both cooperative and competitive tasks. Additionally, we benchmark the cooperative MARL policies against artificial potential field (APF) method [31] and timed-elastic-band (TEB) planner [32], which are common approaches for dynamic obstacle avoidance. Similarly, we also benchmark the competitive MARL policies against disparity-extender algorithm (DEA) [33] and pure behavioral cloning (PBC) [19], which are popular approaches in FITENTH autonomous races. Finally, we also benchmark the performance of the best cooperative MARL policy before (base) and after fine-tuning (FT) in the real world to adapt to the deliberately introduced turf mat.

A. Cooperative Multi-Agent Scenario

1) *Training and Simulation Parallelization*: Fig. 4 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 600 k steps to understand the collective objective of safe intersection traversal. This is marked by a sustainable increase in the cumulative reward (from ~ 3 to ~ 8) as well as episode length (from ~ 470 to ~ 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 ~ 600 to ~ 700 steps) between 700 k and 750 k 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 1 M steps, the policy converged at a stable cumulative reward (~ 8) and episode length (~ 600 steps), while settling at a policy entropy of ~ 1.2 . For LDR and HDR, the KPIs followed a similar trend but with increasing fluctuations. Finally, for FT, we observed an initial performance degradation (owing to the large and sudden domain gap), which eventually recovered and performed well ($>60\%$ success rate).

From a computing perspective, we analyzed the effect of parallelizing the intersection-traversal environment from 1 replica (4 agents) up to 25 replicas (100 agents). We observed that the throughput decreased linearly up to 347.8 ms as the replicas increased and that the computational workload was highest for handling physics, followed by MARL, rendering, and least (negligible) for the API/GUI calls (refer Fig. 2(b)). As shown in Fig. 2(c), reduction in training time (up to 76.3%) was quite non-linear, with a saturating point approaching after 15-20 parallel environments. This resulted in a boost in MARL sample rate from 78.4 Hz for a single replica to 330.7 Hz for 25 parallel environment replicas (refer Fig. 2(d)).

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 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, APF, TEB, and FT. 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. 5. 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 ($<30\%$), also reflected across the reward (<5 points) as well as duration (<500 steps) metrics. APF followed with $\sim 40\%$ success rate, <5 reward points, and <600 steps. TEB averaged with

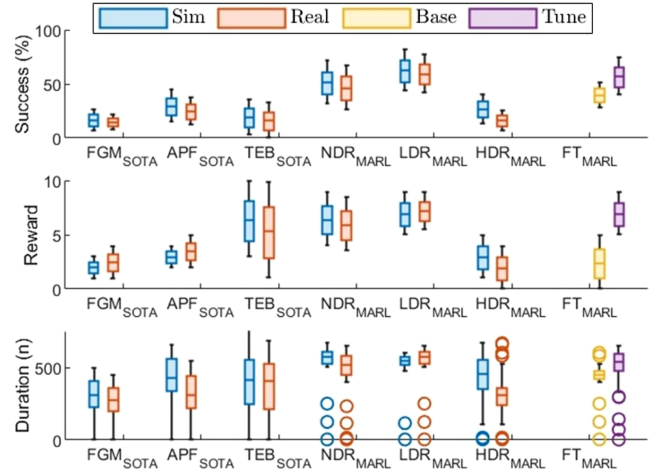


Fig. 5. Deployment and benchmarking of intersection traversal policies with 4 cooperative agents (A1-A4).

$\sim 30\%$ success rate <8 reward points, and <600 steps, but its performance varied greatly since it returned infeasible solution at times. NDR 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 was the most consistent with success rates reaching as high as 80%, rewards as high as 9 points, and episode durations ranging between 500 and 600 steps. HDR performed poorer than other MARL configurations, where it could only achieve up to 40% success rate. Finally, the performance of base LDR degraded when deployed on turf mat ($<50\%$ success), which was improved upon fine-tuning ($\sim 75\%$ success). It is worth mentioning that the closeness between sim and real metrics estimates the sim2real gap, which was least for LDR (4.12%), followed by NDR (9.38%), TEB (13.32%), FGM (16.01%), APF (19.41%), and HDR (33.85%).

B. Competitive Multi-Agent Scenario

1) *Training and Simulation Parallelization*: Fig. 6 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 ~ 200 k 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 ~ 200 k and ~ 800 k steps) guided the agents toward completing multiple laps around the race track. This is marked by an exponential reduction in the BC loss (from ~ 0.1 to ~ 0.025) and a progressive increase in the extrinsic (from ~ 50 to ~ 57), curiosity (from ~ 0.8 to ~ 1.1), and GAIL (from ~ 9 to ~ 11) rewards as well as the episode length (from ~ 1400 to ~ 1600 steps). It was interestingly observed that the red agent (Agent 1) dominated the blue one (Agent 2) till about 500 k

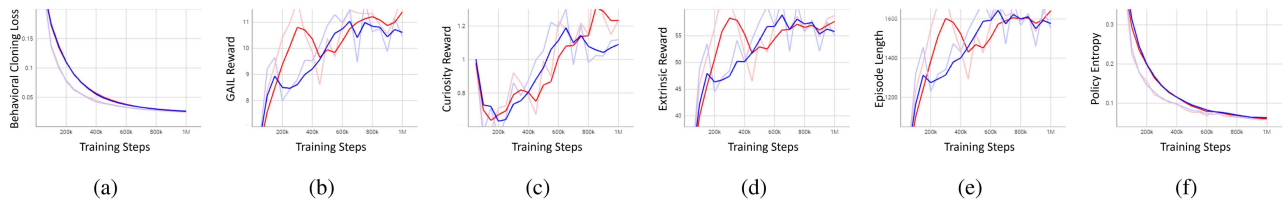


Fig. 6. Competitive MARL training: (a) BC loss, (b) GAIL reward, (c) curiosity reward, (d) extrinsic reward, (e) episode length, and (f) policy entropy w.r.t. training steps.

steps, after which the latter learned the “*competitive spirit*” and bridged the performance gap. Towards the end of 1 M 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 a similar trend for LDR and HDR, but with higher fluctuations (especially in policy entropy), owing to the randomized parameters.

From a computing perspective, we analyzed the effect of parallelizing 2-agent adversarial racing family up to 10 such families (20 agents) training in parallel, within the same environment. We observed that the throughput decreased quite non-linearly up to 274.2 ms with a dropping rate of change as the replicas increased (indicating higher computational efficiency compared to environment parallelization scheme) and that the computational workload was highest for handling physics, followed by MARL, rendering, and least for the API/GUI calls (refer Fig. 2(f)). As observed from Fig. 2(g), reduction in training time (up to 49%) was less dramatic in this case, with a saturating point approaching beyond 10×2 parallel agents. This resulted in a boost in MARL sample rate from 65.9Hz for a single replica to 120.3Hz for 10 parallel agent replicas (refer Fig. 2(h)).

2) *Deployment and Sim2Real Transfer*: The trained policies were first deployed and verified in simulation, where we observed interesting adversarial behaviors (e.g., blocking, baiting, overtaking, etc.). 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, DEA, and PBC. 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. 7. 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 PBC and HDR, 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

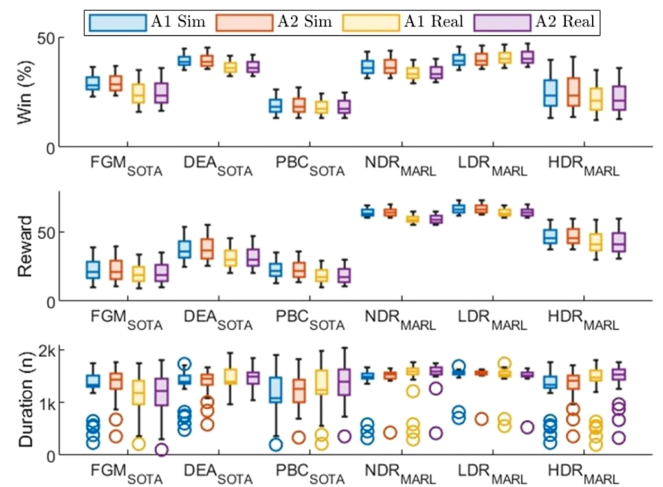


Fig. 7. Deployment and benchmarking of autonomous racing policies with 2 adversarial agents (A1 and A2).

metric. NDR provided higher ($\sim 30\text{--}45\%$) winning consistency for either agent, but could not surpass that of LDR ($\sim 35\text{--}47\%$), which was similar to DEA. The same was reflected by the reward metric, wherein LDR cashed in slightly more reward ($\sim 60\text{--}75$ points) than NDR ($\sim 55\text{--}70$ points). Both performed equally well on the duration metric ($\sim 1400\text{--}1700$ steps), however, LDR was slightly more consistent with lower variance. DEA could collect most reward amongst non-MARL baselines ($\sim 35\text{--}45$ points) and also performed more consistently. Finally, the sim2real gap was least for LDR (2.88%), followed by NDR (6.88%), HDR (8.98%), DEA (10.82%), PBC (11.43%) and FGM (13.48%).

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 mixed-reality sim2real transfer of the policies (for training/testing) using a

single physical vehicle, which was immersed within the proposed digital twin framework to interact with its virtual peers in a virtual environment.

Future avenues of research include analyzing the effect of different communication frameworks and protocols on digital twinning, formulation of physics-guided MARL problems, and scaling the deployments in terms of the number and size of the agents.

REFERENCES

- [1] S. H. Semnani, H. Liu, M. Everett, A. de Ruyter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3221–3226, Apr. 2020.
- [2] 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 *Proc. 2018 IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.
- [3] K. Sivanathan, B. K. Vinayagam, T. Samak, and C. Samak, "Decentralized motion planning for multi-robot navigation using deep reinforcement learning," in *Proc. 3rd Int. Conf. Intell. Sustain. Syst.*, 2020, pp. 709–716.
- [4] 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 *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 8814–8820.
- [5] J. Betz et al., "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.
- [6] P. Werner et al., "Dynamic multi-team racing: Competitive driving on 1/10-th scale vehicles via learning in simulation," in *Proc. 7th Conf. Robot Learn.*, J. Tan, M. Toussaint, and K. Darvish, Eds., 2023, pp. 1667–1685.
- [7] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dür, "Superhuman performance in Gran turismo sport using deep reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4257–4264, Jul. 2021.
- [8] Y. Song, H. Lin, E. Kaufmann, P. Dür, and D. Scaramuzza, "Autonomous overtaking in Gran turismo sport using curriculum reinforcement learning," in *Proc. 2021 IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9403–9409.
- [9] Z. Liu, X. Xu, P. Qiao, and D. Li, "Acceleration for deep reinforcement learning using parallel and distributed computing: A survey," *ACM Comput. Surv.*, vol. 57, no. 4, pp. 1–35, Dec. 2024.
- [10] A. Stooke and P. Abbeel, "Accelerated methods for deep reinforcement learning," 2019, *arXiv:1803.02811*.
- [11] D. Kane, S. Liu, S. Lovett, and G. Mahajan, "Computational-statistical gap in reinforcement learning," in *Proc. 35th Conf. Learn. Theory*, P.-L. Loh and M. Raginsky, Eds. 2022, pp. 1282–1302.
- [12] T. Kim, M. Jang, and J. Kim, "A survey on simulation environments for reinforcement learning," in *Proc. 18th Int. Conf. Ubiquitous Robots*, 2021, pp. 63–67.
- [13] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. 5th Conf. Robot Learn.*, A. Faust, D. Hsu, and G. Neumann, Eds. 2022, pp. 91–100.
- [14] J. Truong, S. Chernova, and D. Batra, "Bi-directional domain adaptation for Sim2Real transfer of embodied navigation agents," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2634–2641, Apr. 2021.
- [15] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.
- [16] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.
- [17] T.-H. Wang, A. Amini, W. Schwarting, I. Gilitschenski, S. Karaman, and D. Rus, "Learning interactive driving policies via data-driven simulation," in *Proc. 2022 Int. Conf. Robot. Automat.*, 2022, pp. 7745–7752.
- [18] R. Mangharam, V. Krovi, J. Betz, A. Amine, C. Samak, and T. Samak, "24th RoboRacer autonomous racing competition," 2025. [Online]. Available: <https://icra2025-race.roboracer.ai>
- [19] M. Bain and C. Sammut, "A framework for behavioural cloning," *Mach. Intell.*, vol. 15, pp. 103–129, 1995.
- [20] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, 2016, pp. 4572–4580.
- [21] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2778–2787.
- [22] T. Samak, C. Samak, S. Kandhasamy, V. Krovi, and M. Xie, "Auto-DRIVE: A comprehensive, flexible and integrated digital twin ecosystem for autonomous driving research & education," *Robotics*, vol. 12, no. 3, May 2023, Art. no. 77.
- [23] 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 Trans. Mechatron.*, vol. 29, no. 4, pp. 2785–2793, Aug. 2024.
- [24] M. O'Kelly et al., "F1/10: An open-source autonomous cyber-physical platform," 2019, *arXiv:1901.08567*.
- [25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [26] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022, pp. 24611–24624.
- [27] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.
- [28] V. Makoviychuk et al., "ISAAC gym: High performance GPU based physics simulation for robot learning," in *Proc. 35th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021, pp. 406–417.
- [29] E. Chisari, A. Liniger, A. Rupenyan, L. Van Gool, and J. Lygeros, "Learning from simulation, racing in reality," in *Proc. 2021 IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8046–8052.
- [30] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: 'Follow the gap method'," *Robot. Auton. Syst.*, vol. 60, no. 9, pp. 1123–1134, 2012.
- [31] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. 1985 IEEE Int. Conf. Robot. Automat.*, 1985, pp. 500–505.
- [32] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5681–5686.
- [33] N. Otterness, "'The Disparity Extender' Algorithm, and F1/Tenth," 2019. [Online]. Available: <https://www.nathanotterness.com/2019/04/the-disparity-extender-algorithm-and.html>