Real-is-Sim: Bridging the Sim-to-Real Gap with a Dynamic Digital Twin

Jad Abou-Chakra[1], Lingfeng Sun[1], Krishan Rana[2], Brandon May[1],
Karl Schmeckpeper[1], Niko Suenderhauf[2], Maria Vittoria Minniti[1], Laura Herlant[1]
[1]Robotics and AI Institute, [2]Queensland University of Technology

***Abstract*— We introduce real-is-sim, a new approach to integrating simulation into behavior cloning pipelines. In contrast to real-only methods, which lack the ability to safely test policies before deployment, and sim-to-real methods, which require complex adaptation to cross the sim-to-real gap, our framework allows policies to seamlessly switch between running on real hardware and running in parallelized virtual environments. At the center of real-is-sim is a dynamic digital twin, powered by the Embodied Gaussian simulator, that synchronizes with the real world at 60Hz. This twin acts as a mediator between the behavior cloning policy and the real robot. Policies are trained using representations derived from *simulator* states and always act on the *simulated* robot, never the real one. During deployment, the real robot simply follows the simulated robot's joint states, and the simulation is continuously corrected with real world measurements. This setup, where the simulator drives all policy execution and maintains real-time synchronization with the physical world, shifts the responsibility of crossing the sim-to-real gap to the digital twin's synchronization mechanisms, instead of the policy itself. We demonstrate real-is-sim on a long-horizon manipulation task (PushT), showing that virtual evaluations are consistent with real-world results. We further show how real-world data can be augmented with virtual rollouts and compare to policies trained on different representations derived from the simulator state including object poses and rendered images from both static and robot-mounted cameras. Our results highlight the flexibility of the real-is-sim framework across training, evaluation, and deployment stages. Videos available at https://real-is-sim.github.io.**

I. INTRODUCTION

Simulation environments are a powerful tool for developing robotic manipulation policies. They offer advantages such as continuous performance monitoring, large-scale parallel rollouts, full access to state information, and precise control over resets and environmental variation. These features make policy learning in simulation significantly more practical to do efficiently. However, translating these advantages to the real world is challenging. The main difficulty lies in ensuring that a policy behaves the same way in both simulation and reality. This requires two things: aligning the input distributions the policy sees in both domains, and ensuring that both the simulated and real robots respond similarly to the same policy outputs. Achieving both reliably is difficult.

To overcome this limitation, we introduce real-is-sim, a new paradigm in which a simulator that continuously synchronizes with the real world is the sole interface to a behavior cloning policy and is ever-present; even during real-world execution. This paradigm is powered by Embodied Gaussians [1], the first simulator to achieve real-time synchronization with physical environments in a generalizable way, through the integration of Gaussian splatting [2] and particle-based physics [3]. This simulator enables the creation of a dynamic digital twin of the physical system that can be updated in real time via sensor feedback. Policies are trained on representations derived from the simulator's state, effectively reframing real-world interactions as simulated experiences and unifying training and deployment.

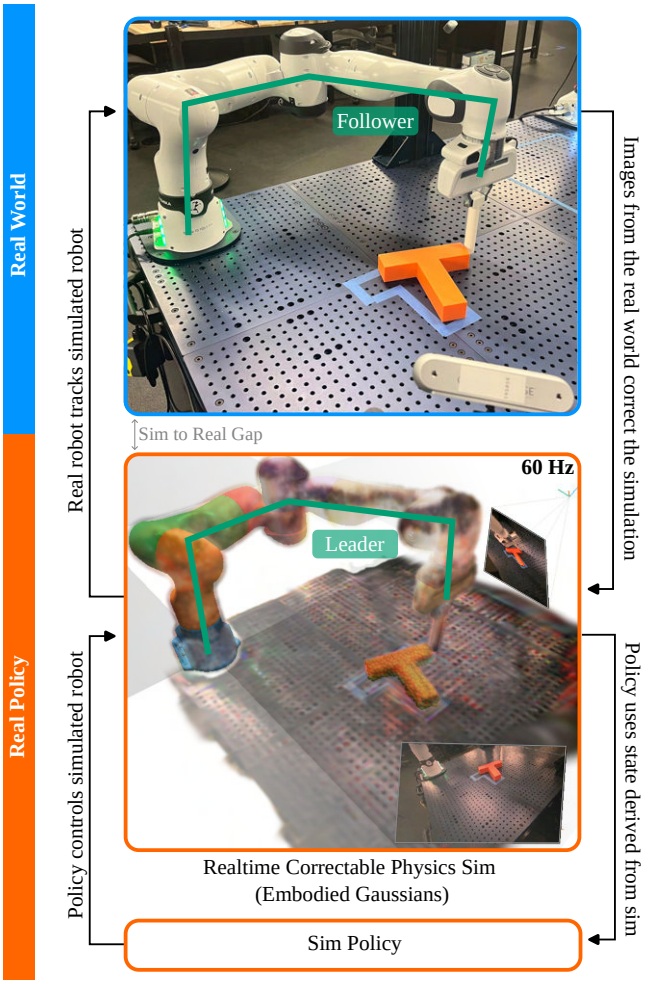

Fig. 1: The real-is-sim framework, illustrating the information flow between its components. A policy trained in a physics simulator that can be synchronized with the real world [1] controls a simulated robot. Real-world observations continuously update the simulator, maintaining its state close to ground truth. The real robot then mirrors the simulated robot's joint positions. This approach shifts the sim-to-real gap challenge from the policy to the physics simulator.

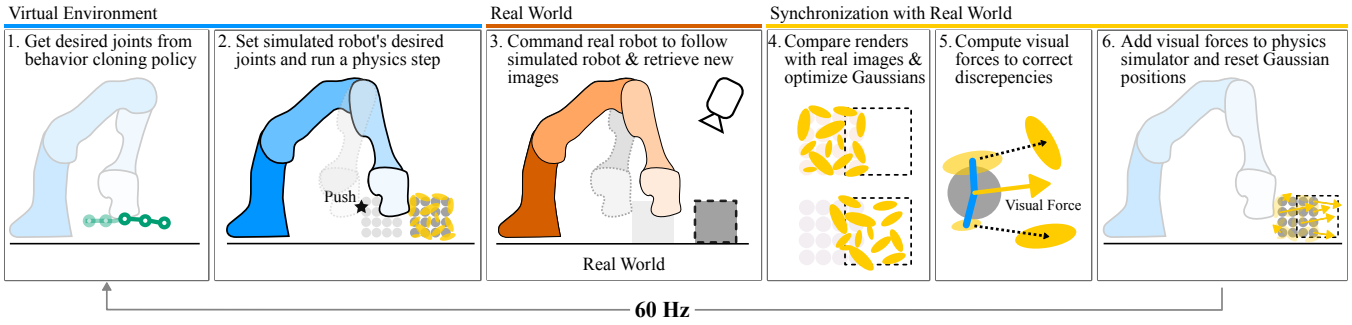

Fig. 2: Figure adapted from [1]. It illustrates how Embodied Gaussians represents the world using physical particles and visual Gaussians. The Gaussians are corrected using real RGB images, and in turn exert fictitious visual forces on the particles to align the simulation with the real world. The adaptation highlights how this mechanism integrates into the real-is-sim framework.

Real-is-sim functions by collecting synchronized simulation states during real-world interactions, which serve as demonstration trajectories for policy training. We avoid sim-only demonstrations to ensure all trajectories reflect motions the real robot can reliably execute whilst keeping the simulation synchronized with the real world. These policies operate on simulator-derived representations and produce joint-space actions for the simulated robot. This process is consistent in the two main operating modes: (i) when deployed on a real robot, and (ii) when evaluating in virtual environments. When connected to a real robot, the physical robot mirrors the simulated robot's configuration, while sensor data continuously updates the state of the objects in the simulator. This architecture decouples the policy from the physical hardware, ensuring the policy always operates on simulator states drawn from a similar distribution as the training data. The real-is-sim paradigm enables seamless switching between real-world deployment and virtual evaluation, as the only distinction between the two modes is the activation or deactivation of the real robot's follower behavior.

By maintaining a real-time digital twin, the real-is-sim framework unifies simulation and reality, enabling consistent policy execution, scalable offline evaluation, and flexible representation learning. Our experiments on the PushT task demonstrate three key applications of real-is-sim: (i) enabling rapid policy evaluation in a virtual environment that correlates strongly with real-world performance, (ii) supporting data augmentation by intermixing real and simulated data to improve performance, and (iii) identifying better simulation-derived representations to increase task effectiveness.

II. PRELIMINARIES

a) Digital Twin: The real-is-sim framework is built around a simulator that stays synchronized with the real world. This is achieved using *Embodied Gaussians* [1], a simulator that continuously corrects itself using RGB-based feedback from real-world observations. The system represents the environment using two tightly coupled components:

1) **Oriented particles**, which capture physical properties such as position, rotation, mass, and radius. These are simulated using a particle-based physics system [3].
2) **3D Gaussians**, which model visual appearance through parameters such as position, rotation, scale, color, and opacity and are rendered using Gaussian splatting [2].

Each particle can be rigidly linked to multiple Gaussians, allowing the physical and visual representations to move together. As the physics system updates the particles, the attached Gaussians follow accordingly. The key innovation in Embodied Gaussians is its correction mechanism based on photometric error. By comparing real camera images with rendered views, the system computes fictitious visual forces that act on the particles. These forces continuously adjust the simulation to stay aligned with reality. This correction loop runs at 60Hz, ensuring the simulator remains both physically consistent and visually accurate. An illustration of the corrective mechanism is provided in Figure 2. Additional details are available in [1].

b) Behaviour Cloning (BC): In BC, the goal is to learn a policy π_θ parameterized by θ that maps an observation o_t at time step t to a sequence of future actions $A_t = [a_t, a_{t+1}, \ldots, a_{t+H-1}]$ over a horizon H. Given a dataset of N expert demonstrations $\mathcal{D} = \{(o^i, A^i)\}_{i=1}^N$, where $o^i = \{o_1^i, \ldots, o_T^i\}$ is a sequence of observations and $A^i = \{a_1^i, \ldots, a_T^i\}$ is the corresponding sequence of actions, the policy is trained to model the conditional distribution $p(A_t \mid o_t)$. This is done by minimizing the BC loss:

$$\mathcal{L}_{\text{BC}}(\pi_\theta) = \mathbb{E}_{(o,A)\sim\mathcal{D}}\left[\|\pi_\theta(o) - A\|^2\right], \tag{1}$$

where the policy's predicted action sequence is penalized based on its deviation from the expert-provided actions.

Modeling temporally extended action sequences rather than single-step actions enables smoother and more consistent control [4], [5]. Our approach adopts *Conditional Flow Matching* (CFM) [6], a generative modeling method that learns a continuous trajectory distribution over actions. CFM models a velocity field $v^\theta(A_t^\tau \mid o_t)$ conditioned on o_t, predicting intermediate velocities along a path A_t^τ between a zero trajectory and the ground-truth A_t. The training

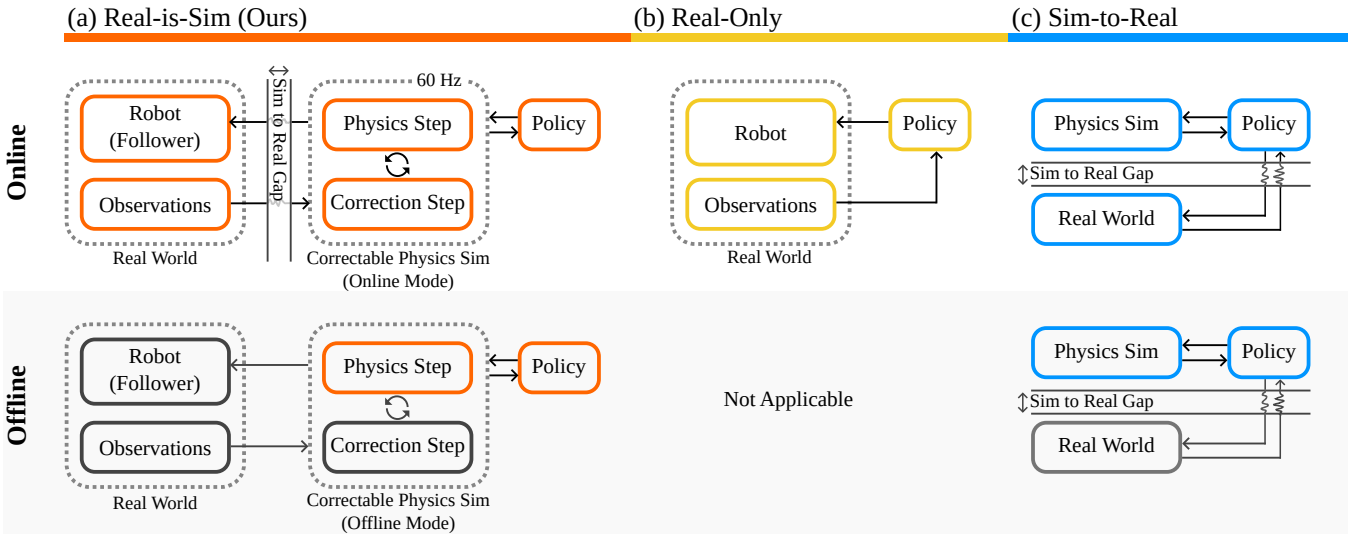

Fig. 3: Comparison of three paradigms: (a) Real-is-Sim, (b) Real-Only, and (c) Sim-to-Real, highlighting their online (real-world interaction) and offline (simulation-based) capabilities. Real-is-Sim offers a unified framework where deploying in a virtual environment is a simplified version of deploying in the real-world, lacking only real-time correction. This ensures seamless transferability. Real-Only is confined to real-world execution. Sim-to-Real struggles with distribution mismatch and dynamics discrepancies, making successful transfer uncertain.

objective minimizes the conditional flow-matching loss:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\tau,\, p(A_t|o_t),\, q_r(A_t^\tau|A_t)} \left\| u(A_t^\tau \mid A_t) - v^\theta(A_t^\tau \mid o_t) \right\|^2, \tag{2}$$

where $\tau \sim \mathcal{U}(0,1)$ is a uniformly sampled interpolation timestep, $q_r(A_t^\tau \mid A_t)$ defines a reference trajectory (e.g., a Gaussian perturbation around a linear interpolation), and $u(\cdot)$ denotes the corresponding ground-truth velocity.

III. METHOD

Real-is-sim is a **framework** that uses a simulator as a mediator between the real world and a learnt policy. For this approach to work, the simulator must continuously synchronize with the physical world by updating its state based on real-world observations. In our work, we use Embodied Gaussians as the simulator, as it provides the necessary mechanisms for real-time alignment. This section first formalizes the framework, then details the complete pipeline: creating a scene, collecting demonstrations, training and deploying a policy, and evaluating the resulting policy.

A. Framework

Our framework consists of three subsystems: (i) the real-world system, (ii) the simulator, and (iii) the learned policy. Real-is-sim distinguishes between two modes: **online**, where the simulator is connected to and synchronized with the real-world system, and **offline**, where the system runs entirely in simulation and can be parallelized. We illustrate the connections in Figure 3a and Figure 4.

The **real-world system** is composed of a set of cameras that output RGB images (o_{rgb}) and a high-impedance joint controller that receives as input a desired joint configuration $c \in \mathbb{R}^d$ for a robot with d degrees of freedom.

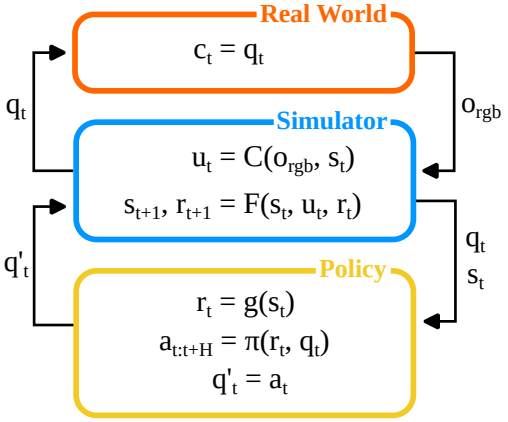

Fig. 4: Information flow between subsystems.

The **simulator** maintains an environment state s_t, a simulated robot state r_t, an environment corrective input u_t, and a simulated robot control input q'_t at each timestep t:

$$s_t = \{p_t^i, R_t^i, v_t^i, \omega_t^i\}_{i=1}^{N_{\text{obj}}}, \quad u_t = \{f_t^i, \tau_t^i\}_{i=1}^{N_{\text{obj}}}, \quad r_t = \{q_t, \dot{q}_t\}.$$

For each simulated object $i \in 1, \ldots, N_{\text{obj}}$, the state is described by the position $p \in \mathbb{R}^3$, orientation $R \in SO(3)$, linear velocity $v \in \mathbb{R}^3$, and angular velocity $\omega \in \mathbb{R}^3$. The corrective control inputs, which align the simulated state with the real world, are computed as virtual forces $f \in \mathbb{R}^3$ and torques $\tau \in \mathbb{R}^3$ applied to each object. For the simulated robot, the state includes the joint positions $q \in \mathbb{R}^d$, joint velocities $\dot{q} \in \mathbb{R}^d$, and the desired joint configuration $q' \in \mathbb{R}^d$. The next states s_{t+1} and r_{t+1} are computed through the simulator's physics step: $s_{t+1}, r_{t+1} = F(s_t, u_t, r_t)$. If the real-world system is connected, the robot's desired joint configuration c is set to q_t at each timestep and the corrective

(a) Real-is-Sim - Real World Deployment (**Online** Mode)

(b) Real-is-Sim - Virtual Deployment (**Offline** Mode)

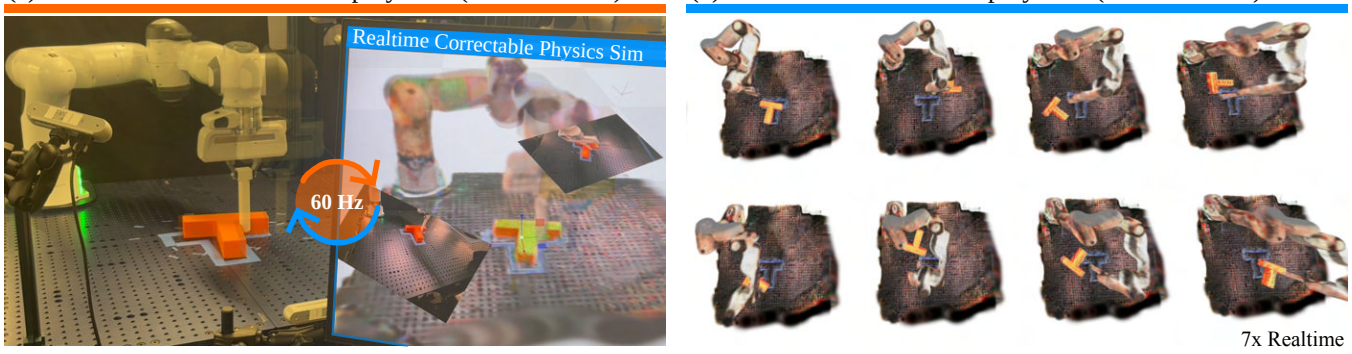

Fig. 5: The two modes of the Real-is-Sim framework: real-world deployment and virtual environment deployment.

inputs u_t are set to $u_t = C(s_t, o_{\text{rgb}})$, where C represents the corrective mechanism in Embodied Gaussians (Fig. 2).

The **policy** is a function $\pi(r_t, q_t)$ that maps a representation $r_t = g(s_t)$ to a trajectory $a_{t:t+H} \in \mathbb{R}^{m\times(d+1)}$, where r_t is derived from the simulator state s_t, m is the number of waypoints in the trajectory (also called the action horizon). Each waypoint a consists of a desired joint configuration $q' \in \mathbb{R}^d$ and the estimated progress towards task completion $p \in [0, 1]$. Once the trajectory is computed, the simulated robot follows it over a time horizon H, updating its desired joint configuration q'_{desired} every H/m seconds to reach the next waypoint in the trajectory a. After the trajectory is completed, a new one is generated. In our implementation, we set $H = 1$ second and $m = 32$.

B. Pipeline

Scene Setup We construct the environment following the Embodied Gaussians procedure. Each object is generated from 4 viewpoints and is represented by a position p, a rotation R, and a set of fixed-radius spheres for collision geometry. Visual appearance is modeled by attaching Gaussians to each body, optimized using Gaussian splatting [2]. The ground and robot model are manually specified.

Demonstration Collection Demonstrations are collected with Embodied Gaussians continuously correcting the simulated state using live camera observations. Collecting with the simulator in-the-loop allows users to monitor synchronization quality and adapt their demonstrations to maintain it. To improve synchronization, during demonstrations we minimize occlusions by keeping the robot body out of camera views, move quickly when not contacting objects, slow down during interactions, and maintain ample clearance around object edges. Throughout each demonstration, we record the full sequence of simulation states at 60Hz. Although the simulation models rich physical properties, only the object poses and robot joint configurations change over time; all other physical parameters are initialized once and remain constant. Similarly, visual states (the Gaussians) are only stored at the initial timestep. A full list of static parameters is provided in [7].

Policy Training Our policy outputs an action trajectory $a_{t:t+H}$, where each waypoint consists of a desired joint configuration $q' \in \mathbb{R}^d$ and a scalar progress value $p \in [0, 1]$. The progress value represents the relative completion of the task and is labeled based on the elapsed time in the demonstration, assuming that all demonstrations end at the target position. This heuristic, validated in [8], allows us to evaluate policies without manually engineering rewards. The policy is conditioned on the robot's current joint configuration q_t. For object state-based policies, we additionally condition on the object's pose, represented by a position and quaternion For image-based policies, we encode rendered images into 64-dimensional vectors using a ResNet. Embodied Gaussians enables the rendering of images from arbitrary viewpoints, allowing the creation of virtual cameras. The representations used for conditioning, including object pose and rendered images, are shown in Figure 6.

Policy Deployment During deployment, the simulator continues running alongside the real robot to ensure it remains synchronized with the physical world. The trained policy processes the state of the simulator and outputs the desired joint configurations for the simulated robot. This means the policy is operating with the same type of inputs it was trained on, controlling the simulated robot as it was during demonstration collection. The integration with the real-world system occurs through two key mechanisms: (1) RGB images from the real-world cameras are continuously input into the system, providing real-time corrections to the state of the simulated objects. This feedback enables the simulator to continuously adjust and align the simulated environment with the real world. (2) The real robot's joint controller is configured to follow the simulated robot's joint positions q (and not the desired joints q'). This design choice, where the real robot acts as a follower, allows for a smooth transition between operating on a real robot and operating solely in a virtual environment.

Policy Evaluation By deactivating the real-world coupling and corrective force calculation (u_t), the same setup can be used in a virtual-only environment. This allows for easy parallelization of environments which enables fast policy evaluation. In virtual deployments, we achieve a speedup of 5x to 8x, depending on the chosen representation for the policy, when running 20 environments concurrently.

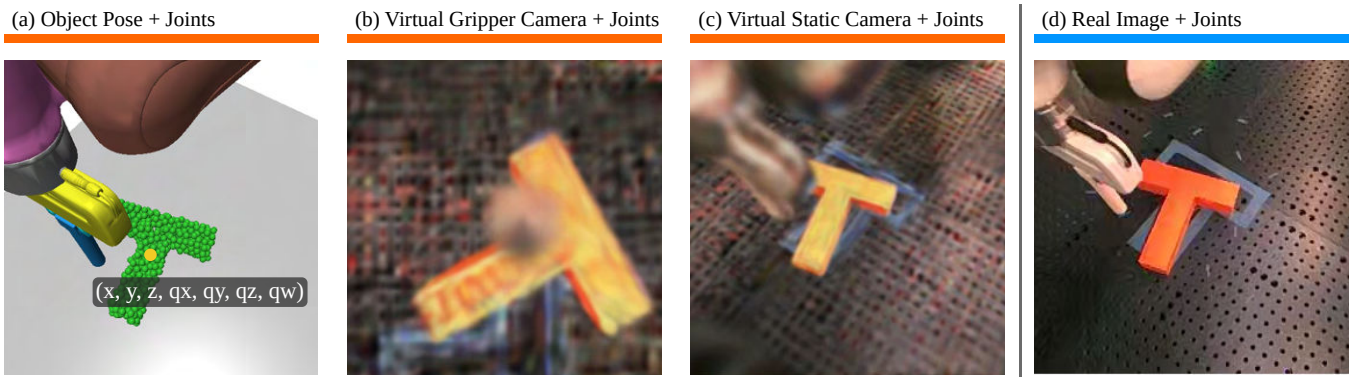

Fig. 6: Comparison of representations from the Real-to-Sim framework versus a real image baseline: (a) State-based representation (object position and orientation), (b) Virtual camera on robot end effector, (c) Virtual camera at real camera position, (d) Real camera image, which is used to correct the simulation, but not as a direct input to the policy.

IV. EXPERIMENTS

Experimental Setup We evaluate our framework on the PushT task: a long horizon task where a robot uses a pusher to position a T-shaped object into a predefined location. The T-Block's geometry (represented as a set of spherical shapes attached to a single body frame) is automatically generated using the method from Embodied Gaussians [1]. We use three RealSense D455 cameras running at 90 fps with a resolution of 420×270. The robot mesh is known, and the ground plane is estimated from the camera-generated pointclouds. Gaussians attached to the robot, table, and T-Block are learned once using the Gaussian Splatting procedure outline in [1] and reused across experiments.

For object state-based policies, we use the pose of the object encoded as a translation vector $\mathbf{p} \in \mathbb{R}^3$ and a unit quaternion $\mathbf{R} \in \mathbb{SO}(3)$. For image-based policies, we use a 64 dimensional feature vector output by a ResNet [5] which was jointly trained with the policy. The images input to the ResNet are rendered from the simulation using Gaussian Splatting from virtual cameras. These cameras can be placed anywhere. In our experiments, we use a static camera placed in the same location as the real camera and a virtual gripper camera which we mount to the simulated robot's end-effector (these are shown in Figure 6).

To evaluate policy success on the PushT task, we define 20 starting poses on the table that are not part of the demonstration set. In real-world testing, success is determined by whether the policy moves the T-Block to the target location from these poses. Each pose is tested three times, resulting in 60 measurements per success rate evaluation. In virtual-only testing, success is determined when the policy outputs a progress value above 0.9, a strategy verified in [8], which serves as a good proxy for success without requiring engineered reward values.

Offline Evaluation In this experiment, we demonstrate how real-is-sim's offline mode facilitates policy training and evaluation. A key challenge in imitation learning lies in selecting optimal checkpoints for deployment. This difficulty arises because the loss used in behavior cloning does not reliably indicate policy performance, and real-world robot evaluations require substantial time and resources. Consequently, researchers often default to using the final training checkpoints despite it being suboptimal.

We demonstrate that real-is-sim's offline mode enables efficient checkpoint selection through rapid evaluation. For a state-based policy trained on 30 demonstrations, we implement a flow-matching policy trained for 50,000 steps (Figure 7). We then compare two evaluation methodologies: (i) evaluation on only the virtual world (without the real robot attached and the corrective mechanisms engaged) and (ii) real-world online evaluations. Our virtual-only approach initializes environments with identical test cases to the real-world evaluations. The virtual-only evaluation calculates success rates through the policy-reported progress. While we use this metric to maintain generality, practitioners could customize reward functions for their specific tasks.

We selected four representative checkpoints (10k, 15k, 25k, 50k) for real-world evaluation and performed more comprehensive analysis using faster offline evaluation. Figure 7 shows that, while absolute performance may differ, the relative ordering of checkpoints is preserved - making offline evaluation a practical tool for selecting high-performing policies in minutes rather than hours. Notably, later checkpoints do not consistently outperform earlier ones, suggesting practitioners could use offline evaluation for either early stopping or optimal checkpoint selection. While this experiment uses identical initial states for both evaluation modes to establish correlation, real applications could benefit from randomized initial states during training-phase evaluations.

Offline Data Collection and Augmentation In this experiment, we demonstrate how real-is-sim's offline mode can be used to to collect additional demonstrations to increase task performance (Figure 8). We train a state-based policy using 30 real-world demonstrations collected in online mode - which achieves a success rate of around 57%. We also augment the real-world demonstrations using the simulator only (offline mode). To collect the additional demonstrations, we deploy this policy in offline mode across multiple parallel environments and identify failure states where the policy falls into a local minima. These failure states are then presented to the user to provide an additional 30 demonstrations specifi-

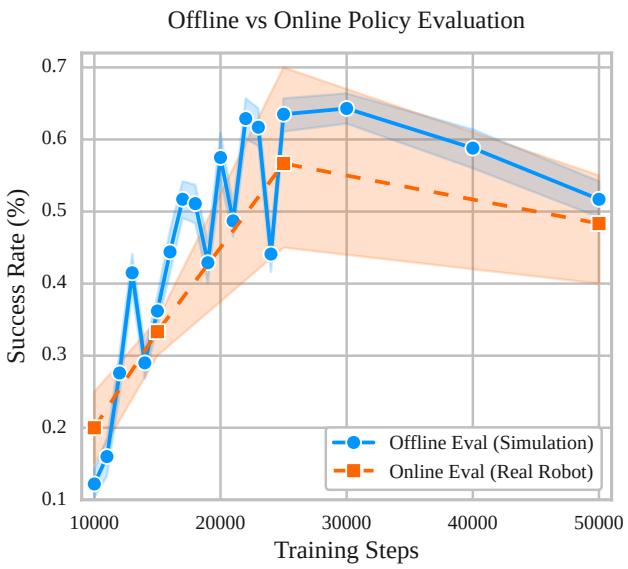

Fig. 7: Success rate of an imitation learning policy vs training steps, comparing evaluation on the real setup and in a virtual environment. The consistency between the two curves shows that the faster virtual-only evaluation is a reliable proxy for real-world testing, providing a practical alternative for policy assessment. Error bars represent 95% confidence intervals.

cally solving the task under these challenging configurations. By combining the original 30 real-world demonstrations with the 30 new simulated demonstrations, we retrain the policy and observe a significant improvement to 80% success rate.

Lastly as a baseline, we also collect 30 demonstrations in simulation only. The resulting sim-only policy attains a similar success rate during deployment. The success rate of the offline only policy is comparable to the policy trained on real demonstrations. This is largely owing to the kinematic nature of the PushT task which means that successfully policy rollouts are mostly dependent on poses and not dynamic effects. This is precisely what Embodied Gaussians can easily correct upon deployment.

This experiment highlights a straightforward yet powerful application of real-is-sim's offline mode. The cost of augmenting the data used for policy training using real-is-sim's simulator is much lower than doing so in the real world. As a comparison, we also train a policy using 60 real-world demonstrations collected without augmentation. Interestingly, this approach yields a similar performance improvement as the augmented dataset, underscoring the effectiveness of real-is-sim's offline mode in efficiently enhancing policy training.

Representation Flexibility Another major benefit of real-is-sim's offline mode is its flexibility in extracting diverse representations from the Embodied Gaussians simulation. For instance, it enables access to privileged information—such as precise object poses—that would be difficult to obtain in the real world. Additionally, virtual cameras can be freely positioned within the system, facilitating visuomotor policy learning from multiple perspectives. In our experiments, we demonstrate that policies leveraging different representations—including state-based object poses, a gripper-mounted virtual camera, and a static virtual camera—all successfully solve the task (Fig. 6). Notably, these approaches achieve performance comparable to, and in some cases surpassing, the baseline policy that relies solely on real-world images and robot actions.

Interestingly, the policy using the gripper-mounted virtual camera achieves the highest performance (82%) among all representations. This policy also exhibits behaviors such as as actively searching for the T-block when it leaves the camera's field of view (see videos on our website). These findings align with prior work highlighting the advantages of wrist-mounted cameras over static ones in certain scenarios [9]. Within the real-to-sim framework, we can systematically evaluate different virtual camera configurations [10] and their impact on policy learning—all using the same set of demonstrations. This flexibility allows us to identify the optimal representation for a given manipulation task in offline mode and seamlessly deploy it in online execution.

V. RELATED WORK

Limitations of Real-World Policy Evaluation: Evaluating robotic policies directly in the real world, while reliable, suffers from being labor-intensive, time-consuming [11], [12], and facing scalability bottlenecks due to manual processes [13]. Consequently, researchers explore simulation for safer, repeatable, and scalable evaluation [14], [15], [16], [17], often focusing on improving simulation fidelity [14], [17], [16], [15]. Our framework differs from this approach by integrating a continuously synchronized simulator throughout the entire policy lifecycle (training, evaluation, deployment). This real-time alignment via sensor data enables direct deployment of simulation-trained policies without further tuning and ensures a unified observation and control space for scalable evaluation.

Addressing the Sim-to-Real Gap in Robotic Manipulation: Simulation offers a scalable alternative to real-world evaluation [11], but the sim-to-real gap remains a key challenge. Existing strategies include domain randomization to enhance robustness [18] and domain adaptation to align representations [19], as well as methods focusing on aligning simulation dynamics [20], [21] or using real-to-sim-to-real approaches [22], [23]. Frameworks like DeepMind's robot soccer also combine neural rendering with domain randomization [24]. While these methods aim to bridge the sim-to-real gap by making policies more robust or adapting representations, our "real-is-sim" paradigm eliminates this gap entirely. By continuously aligning the simulator with the real world and treating it as the execution environment, we decouple the policy from sim-real discrepancies. The challenge of domain alignment shifts to the simulator itself, which is addressed through real-time updates.

Leveraging World Models for Robotics: Learned world models enable robots to predict future outcomes for planning and control. These range from 2D predictors [25] to more structured 3D physically grounded models, such as those

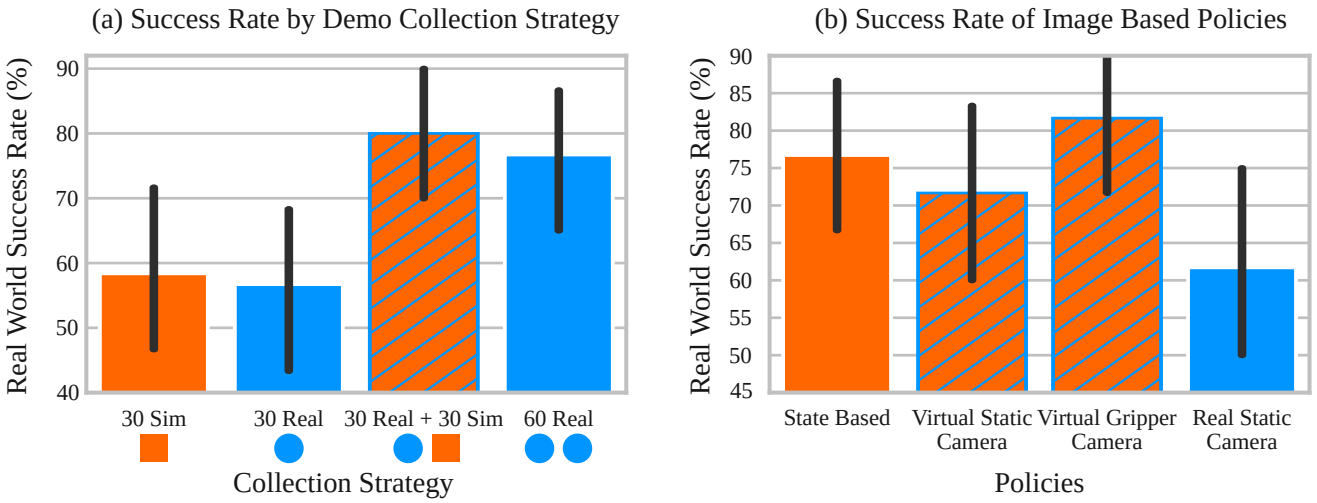

Fig. 8: (a) Success rates of different demonstration collection strategies. (b) Success rates of policies using different representations to condition the flow matching policy. Error bars are 95% confidence intervals.

utilizing Gaussian Splatting [26], [1], which allow for real-time tracking and simulated interaction. Our work builds upon this by exploring the integration of such advanced 3D world models within a behavior cloning framework. This aims to harness the advantages of simulation directly in real-world tasks by training policies on the dynamically updated, real-time digital twin provided by these models.

VI. Limitations

While our framework demonstrates promising results, its applicability is fundamentally constrained by the fidelity of the underlying physical simulator. Scenarios that diverge significantly from the simulator's modeling assumptions present challenges for both forward prediction and corrective mechanisms. Future work could address these limitations by improving simulator accuracy through adaptive parameter estimation—for example, real-time optimization of physical properties such as friction or mass, as explored in [26] and [27].

Another limitation of the current framework is its reliance on visual forces to correct the simulator state. This restricts applicability to scenarios where objects are frequently visible and do not undergo prolonged occlusion or manipulation without observation.

Moreover, for Embodied Gaussians to function effectively, visual corrections must not conflict significantly with the dynamics predicted by the physics simulator. The approach assumes that both the simulator and the visual feedback are aligned, that is, they attempt to move the object in compatible directions. When this assumption breaks down and the corrections diverge, synchronization between the simulation and the real world may not be reliable. However, this limitation can be addressed. The synchronization mechanism can be extended with additional corrective signals beyond visual forces. For example, semantic cues, such as correspondence detection via methods like TAPIR [28] or loss augmentation using segmentation masks as in [29], could provide more robust alignment in challenging settings. The current framework was designed to demonstrate the simplest viable instantiation. Future extensions could incorporate these more sophisticated correction signals to broaden applicability across diverse and visually ambiguous environments.

Another limitation lies in the scene initialization process. In this work, we employed a simple scheme that is sensitive to factors such as the number and placement of cameras, object complexity, and visibility. However, recent advances in learned priors for geometry initialization, such as InstantMesh [30] and InstantSplat [31], offer promising alternatives that could replace or enhance our approach, expanding the set of objects that can be initialized effectively.

Although we focused on the PushT task to validate the core contributions, the proposed methodology is broadly applicable to other manipulation domains. Extending this approach to tasks such as multi-object rearrangement, cloth manipulation, or non-prehensile pushing will require adapting the simulator's dynamics and corrective policies to accommodate more complex interactions.

VII. Conclusions

We introduced real-is-sim, a unified framework for policy training, evaluation, and deployment that maintains a continuously corrected simulation throughout the entire pipeline. By treating simulation as a persistent, real-time interface, enabled by Embodied Gaussians, we eliminate the disconnect between training and deployment domains. This reframes policy evaluation as a simulation-native process, even in real-world settings. Our results on the PushT task show strong alignment between offline and real-world performance, demonstrating that real-is-sim enables scalable, efficient evaluation without sacrificing fidelity. We hope this framework serves as a foundation for future real-world learning algorithms that leverage this tight coupling between simulation and reality.

ACKNOWLEDGMENT

The QUT authors acknowledge continued support from the Queensland University of Technology (QUT) through the Centre for Robotics. Their work was partially supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (Project DP220102398).

REFERENCES

[1] J. Abou-Chakra, K. Rana, F. Dayoub, and N. Suenderhauf, "Physically embodied gaussian splatting: A visually learnt and physically grounded 3d representation for robotics," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=AEq0onGrN2

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[3] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: Position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*, ser. MIG '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 49–54. [Online]. Available: https://doi.org/10.1145/2994258.2994272

[4] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *Robotics: Science and Systems XIX*, 2023.

[5] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[6] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," 2023. [Online]. Available: https://arxiv.org/abs/2210.02747

[7] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," in *NVIDIA GPU Technology Conference (GTC)*, 2022.

[8] K. Rana, J. Abou-Chakra, S. Garg, R. Lee, I. Reid, and N. Suenderhauf, "Affordance-centric policy learning: Sample efficient and generalisable robot policy learning using affordance-centric task frames," *arXiv preprint arXiv:2410.12124*, 2024.

[9] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn, "Vision-based manipulators need to also see from their hands." in *ICLR*, 2022.

[10] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, "Rvt: Robotic view transformer for 3d object manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 694–710.

[11] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine, "Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world," *arXiv preprint arXiv:2503.24278*, 2025.

[12] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel, "Robot learning as an empirical science: Best practices for policy evaluation," *arXiv preprint arXiv:2409.09491*, 2024.

[13] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2024.

[14] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[15] X. Li, K. Hsu, J. Gu, O. Mees, K. Pertsch, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao, "Evaluating real-world robot manipulation policies in simulation," in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 06–09 Nov 2025, pp. 3705–3728.

[16] S. Silwal, K. Yadav, T. Wu, J. Vakil, A. Majumdar, S. Arnaud, C. Chen, V.-P. Berges, D. Batra, A. Rajeswaran *et al.*, "What do we learn from a large-scale study of pre-trained visual representations in sim and real environments?" in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 17 515–17 521.

[17] M. Khanna, Y. Mao, H. Jiang, S. Haresh, B. Shacklett, D. Batra, A. Clegg, E. Undersander, A. X. Chang, and M. Savva, "Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 384–16 393.

[18] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[19] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] A. Z. Ren, H. Dai, B. Burchfiel, and A. Majumdar, "Adaptsim: Task-driven simulation adaptation for sim-to-real transfer," 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. 3434–3452.

[21] N. Pfaff, E. Fu, J. Binagia, P. Isola, and R. Tedrake, "Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups," arXiv preprint arXiv:2503.00370, 2025.

[22] M. T. Villasevil, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal, "Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[23] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, "Transic: Sim-to-real policy transfer by learning from online correction," in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 06–09 Nov 2025, pp. 1691–1729.

[24] D. Tirumala, M. Wulfmeier, B. Moran, S. Huang, J. Humplik, G. Lever, T. Haarnoja, L. Hasenclever, A. Byravan, N. Batchelor, N. Sreendra, K. Patel, M. Gwira, F. Nori, M. Riedmiller, and N. Heess, "Learning robot soccer from egocentric vision with deep reinforcement learning," arXiv preprint arXiv:2405.02425, 2024.

[25] S. S. Yang, Y. Du, K. Ghasemipour, J. Tompson, L. Kaelbling, D. Schuurmans, and P. Abbeel, "Learning interactive real-world simulators," arXiv preprint arXiv:2310.06114, 2024.

[26] H. Jiang, H.-Y. Hsu, K. Zhang, H.-N. Yu, S. Wang, and Y. Li, "Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos," *arXiv preprint arXiv:2503.17973*, 2025.

[27] A. Longhini, M. Büsching, B. P. Duisterhof, J. Lundell, J. Ichnowski, M. Björkman, and D. Kragic, "Cloth-splatting: 3d cloth state estimation from RGB supervision," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=WmWbswjTsi

[28] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman, "TAPIR: Tracking any point with per-frame initialization and temporal refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 061–10 072.

[29] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis," in *3DV*, 2024.

[30] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan, "Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models," *arXiv preprint arXiv:2404.07191*, 2024.

[31] Z. Fan, K. Wen, W. Cong, K. Wang, J. Zhang, X. Ding, D. Xu, B. Ivanovic, M. Pavone, G. Pavlakos, Z. Wang, and Y. Wang, "Instantsplat: Sparse-view gaussian splatting in seconds," 2024.