2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Unreal Robotics Lab: A High-Fidelity Robotics Simulator with Advanced Physics and Rendering

Jonathan Embley-Riches*, Jianwei Liu*, Simon Julier, and Dimitrios Kanoulas

Fig. 1: Photorealistic renderings created by our simulation framework. Top row (types of robots): Unitree Go1 quadruped, Skydio X2 quadropter and Unitree B1-Z1 qudruped mobile manipulator Bottom row: various adverse visual conditions.

***Abstract*— High-fidelity simulation is essential for robotics research, enabling safe and efficient testing of perception, control, and navigation algorithms. However, achieving both photorealistic rendering and accurate physics modeling remains a challenge. This paper presents a novel simulation framework, the Unreal Robotics Lab (URL), that integrates the advanced rendering capabilities of the Unreal Engine with MuJoCo's high-precision physics simulation. Our approach enables realistic robotic perception while maintaining accurate physical interactions, facilitating benchmarking and dataset generation for vision-based robotics applications. The system supports complex environmental effects, such as smoke, fire, and water dynamics, which are critical to evaluating robotic performance under adverse conditions. We benchmark visual navigation and SLAM methods within our framework, demonstrating its utility for testing real-world robustness in controlled yet diverse scenarios. By bridging the gap between physics accuracy and photorealistic rendering, our framework provides a powerful tool for advancing robotics research and sim-to-real transfer. Our open-source framework is available at `https://unrealroboticslab.github.io/`.**

I. INTRODUCTION

Simulation is essential for robotics research, enabling safe, efficient development, testing, and validation of robotic systems before real-world deployment. High-fidelity environments must simultaneously deliver realistic perception and accurate physics interactions—an open challenge due to competing computational demands and the complexity of real-world dynamics [1].

We introduce the **Unreal Robotics Lab (URL)**, a novel open-source framework that tightly integrates Unreal Engine's state-of-the-art rendering pipeline (Lumen global illumination, Nanite virtualized geometry, Niagara particles) [2] with MuJoCo's validated high-precision physics engine [3]. By embedding MuJoCo natively inside the Unreal ecosystem via shared-memory execution, URL achieves deterministic physics while exposing Unreal's full authoring, environmental-effect, and tools directly to roboticists.

A key capability is the effortless generation of complex dynamic phenomena — smoke, fire, water waves, dynamic lighting, and moving agents—that are rare or hazardous to capture in real datasets. These effects enable controlled benchmarking of vision-based policies under out-of-distribution conditions and the creation of large-scale training data for perception and navigation.

Unlike domain-specific simulators (e.g., autonomous driving or aerial robotics) or platforms focused on limited embodied-AI scenarios, URL is designed for **general-purpose robotic applications**, supporting quadrupeds, mobile manipulators, UAVs, and arms. While simulators such as Isaac Sim [4] offer broad capabilities, URL prioritises physical fidelity by retaining MuJoCo's superior accuracy and stability in contact-rich tasks and incorporates **CoACD-**

*equal contribution

The authors are with the Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK.

This work was supported by the UKRI FLF [MR/V025333/1] (RoboHike).

based convex decomposition [5] for improved non-convex object representation.

Our key contributions are:

- **Native "Unreal-First" Architecture:** We introduce a shared-memory execution model that embeds the MuJoCo engine directly within the Unreal Engine environment. This eliminates the latency and synchronization overhead of traditional client-server models.
- **High-Fidelity Physics-to-Visual Synchronization:** By mirroring the MuJoCo schema within the UE Actor-Component hierarchy, we enable seamless synchronization between MuJoCo's stable physics and UE's photorealistic rendering pipeline (Lumen, Nanite, Niagara).
- **Automated Environmental Processing and Geometry Conversion:** The framework provides a robust pipeline for bridging complex UE environmental assets to physics-ready formats, utilizing multi-threaded terrain sampling and automated CoACD-based conversion for arbitrary static meshes.
- **Comprehensive ROS-Compatible Middleware:** A unified interface for sensors and actuators that supports diverse robotic platforms with native support for ROS and ZeroMQ communication. This enables the "plug-and-play" deployment of modern planners (e.g., ViNT, NoMAD) and SLAM frameworks (e.g., ORB-SLAM3) directly within the simulation.
- **Generalized Benchmarking for Diverse Conditions:** We provide a structured framework for evaluating robotic systems under rare or extreme environmental conditions, complemented by a deterministic replay system that allows for robust domain randomization testing across varied visual environments.

II. RELATED WORK

Robotic simulation has advanced significantly in recent years, addressing key challenges in physics accuracy, photorealistic rendering, and real-time performance. This section reviews related work in three main areas: robotics-focused simulation platforms, physics engine comparisons, and benchmarking of SLAM and navigation under adverse visual conditions.

Game engines and simulation platforms offer photorealistic rendering [6], [7], [8], yet Unreal Engine provides distinct advantages through Lumen global illumination and Nanite virtualized geometry [2]. Lumen enables real-time global illumination and reflections without precomputed lighting—ideal for dynamic environments—while Nanite efficiently renders highly detailed assets with millions of triangles. Combined with its extensible ecosystem (third-party plugins, visual scripting, AI-driven NPCs, and cross-platform deployment), Unreal Engine is particularly well-suited for vision-based robotics research.

A. Robotics-Focused Simulation Platforms

Several simulators target robotic perception, control, and reinforcement learning [9], [10], each with different trade-offs in realism and efficiency. Gazebo [11] is a widely-used open-source platform that supports multiple physics engines (ODE [11], Simbody [12], Bullet [13], DART [14]), although its rendering lags behind modern game-engine solutions. MuJoCo [3] excels in accurate, efficient physics for control tasks but lacks built-in photorealistic rendering. GPU-accelerated frameworks such as Isaac Orbit [6] and Isaac Gym [15] prioritise speed for large-scale reinforcement learning, while Isaac Sim [4] (built on NVIDIA Omniverse) uses PhysX [16] for scalability at the expense of contact-dynamics accuracy compared with MuJoCo. Intel SPEAR [17] provides photorealistic indoor environments for embodied AI, and domain-specific simulators such as CARLA [8] and AirSim [18] leverage Unreal Engine for urban/aerial scenarios. Manipulation-focused suites RoboSuite [19] and RoboCasa [20] rely on MuJoCo physics but offer limited environmental complexity.

Although most platforms emphasise either physics fidelity or visual realism, few achieve both. URL bridges this gap by natively integrating MuJoCo's precise physics with Unreal Engine's state-of-the-art rendering in an editor-first, ROS-native framework designed for general-purpose robotics (quadrupeds, mobile manipulators, UAVs, and arms).

B. Physics Engines for Robotics Simulation

Physics engines are critical for realistic contact, soft-body, and fluid modelling. MuJoCo [3] is widely favored for its for soft-contact accuracy and stability in complex robotic tasks [21]. ODE (used in Gazebo) suffers from stability issues in high-DOF scenarios and requires extensive tuning. Bullet [13] is popular for reinforcement learning and manipulation but lags MuJoCo in precision. NVIDIA PhysX [16] offers GPU acceleration suitable for fast RL yet omits Coriolis forces and exhibits lower linear-stability accuracy. Unreal Engine's Chaos and Havok solvers [22] excel at real-time destruction and particle effects but are not optimised for high-precision joint dynamics or soft-contact robotics.

The continued adoption of MuJoCo's physics is further evidenced by the fact that next-generation GPU-accelerated simulators such as **Genesis** [23] and **Newton** [24] have adopted **MJWarp** [25] (a high-performance GPU backend directly derived from MuJoCo) as their core physics engine. This widespread use in the latest state-of-the-art platforms confirms that MuJoCo's contact modelling, accuracy, and stability remain unmatched—even as the field shifts toward massive parallel GPU simulation.

URL therefore retains native MuJoCo as the sole physics backend while inheriting Unreal Engine's advanced rendering, particle systems, and tools—delivering both photorealism and validated physical fidelity without compromise.

C. Benchmarking SLAM and Navigation Under Adverse Visual Conditions

Evaluating SLAM and navigation algorithms under adverse conditions (smoke, fog, dynamic lighting, or moving agents) remains difficult, as most public datasets and simulators capture only structured, well-lit scenes [26].

Envodat [27] provides one challenging real-world dataset, yet scale and diversity are still limited. URL addresses this limitation by synthetically generating controllable rare-event scenarios via Niagara particles and fluids, together with dynamic NPCs and vehicles scripted through Unreal's NavMesh and Behavior Trees.

Current simulators typically prioritise either high-precision physics (e.g., MuJoCo-based) or photorealistic rendering (e.g., CARLA, AirSim). While our work shares the high-level goal of SPEAR [17] in combining Unreal rendering with MuJoCo physics, the design philosophies differ fundamentally. SPEAR is structured as a Python-based OpenAI Gym environment for task-specific embodied-AI research. In contrast, URL is built for *systems-level* validation: it offers native ROS integration, direct scene authoring inside the Unreal Editor (with automatic asset processing and CoACD convex decomposition), and seamless import of any MuJoCo-compatible robot. This enables full-stack testing of complete robotic software pipelines exactly as they would be deployed in the real world, together with repeatable benchmarking of perception and planning robustness in behaviorally rich, non-static environments.

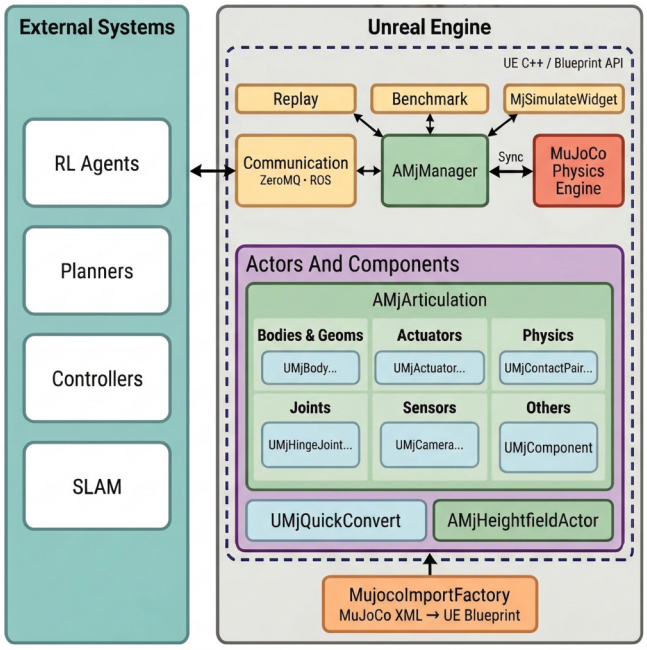


Fig. 3: Unreal - MuJoCo Simulator system diagram.

III. Simulator Architecture

A. System Overview

We propose an "Unreal Engine-first" simulation framework (Fig. 3) that embeds the MuJoCo physics engine entirely within the Unreal Engine (UE) ecosystem. By discarding traditional client-server communication models in favor of in-process, shared-memory execution, the architecture ensures absolute physical determinism. This design enables the physics simulation to natively leverage UE's photorealistic rendering pipeline (Lumen, Nanite), environmental effects (Niagara), and AI navigation (NavMesh) without the latency or synchronization overhead inherent in distributed architectures.

B. World Authoring and Asset Instantiation

A core principle of this framework is the comprehensive mirroring of the MuJoCo schema within UE's Actor-Component hierarchy. Rather than providing an abstracted wrapper, the system exposes nearly the entire MuJoCo specification—including global defaults, solver options, and runtime configurations—as native, strongly-typed properties directly within the UE interface. The framework caters to both legacy MuJoCo users and native Unreal Engine development through a tiered ingestion pipeline.

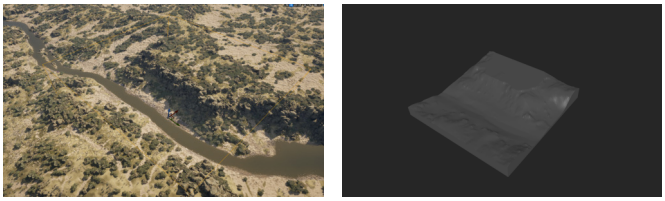
Users can integrate legacy models by dragging standard MuJoCo XML files directly into the UE Content Browser. This triggers a dedicated factory that parses the MJCF specification to automatically generate a structurally equivalent UE Blueprint. Articulated robots and environmental objects are instantiated as `AMjArticulation` Actors, which manage a recursive tree of localized components such as `UMjBody`, `UMjGeom`, `UMjJoint`, and `UMjTendon`. This process automatically imports and converts all referenced meshes and primitives into native UE assets, creating a persistent representation that benefits from UE's advanced material and rendering systems.

For native authoring, users can instantiate an empty `AMjArticulation` and procedurally build the hierarchy through the standard UE "Add" interface. The architecture abstracts away from ID-based indexing and manual pointer arithmetic; users can access live physics data directly through the articulation's component references without managing raw `mjModel` or `mjData` addresses. Scene construction is finalized by placing these Blueprints within a level containing an `AMjManager` Actor. Upon initialization, the manager performs a scene-wide scan to aggregate all active Actors into a contiguous specification. To optimize performance, the framework utilizes asynchronous asset serialization and caching for mesh generation, ensuring complex geometries are processed once. The system maintains bidirectional compatibility, allowing users to export UE-authored levels back to standardized MuJoCo formats.

C. Environmental Sampling and Geometry Processing

To bridge the gap between UE's diverse environmental assets and MuJoCo's collision requirements, the framework provides automated conversion utilities. The `AMjHeightfieldActor` facilitates terrain interaction by executing multi-threaded, bounded raycasts to sample the environment. This actor is not restricted to UE Landscapes; it can sample any geometry not already managed by the MuJoCo solver, specifically any Actor lacking an `AMjArticulation` or a `UMjQuickConvertComponent`. This allows for the generation of optimized MuJoCo heightfields from complex, non-articulated environmental geometry while minimizing memory overhead (Fig. 4). For discrete objects,

the `UMjQuickConvertComponent` enables the instant conversion of arbitrary UE static meshes into MuJoCo-compatible collision bodies using CoACD [5] for automatic convex hull decomposition.



(a) Unreal Engine Landscape (b) MuJoCo HeightField

Fig. 4: Unreal Engine Landscape conversion to MuJoCo HeightField.

D. Unified Interaction and Bi-Directional Physics

To maintain strict determinism, the MuJoCo simulation executes on a dedicated asynchronous thread. The solver steps at the specific timestep defined within the MuJoCo model, which is exposed and configurable via the `AMjManager`. Interaction is unified across the framework's API; every MuJoCo property, function, and sensor is accessible via both C++ and UE Blueprints. This includes a comprehensive sensor interface where all MuJoCo-native sensors are exposed as UE components, providing researchers with Blueprint nodes to access real-time data streams. Furthermore, the framework includes a native graphical UI (`MjSimulateWidget`) that replicates the functionality of MuJoCo's standalone *Simulate*, allowing for real-time visualization, state monitoring, and interactive control within the UE viewport. While the framework handles pointer management, users retain the ability to interact with the raw MuJoCo structs if required for custom solver logic. This flexibility is complemented by a robust one-way kinematic coupling (*mocap*) which allows UE's native Chaos physics and AI systems to influence the MuJoCo solver in real-time. For example, an NPC governed by the UE NavMesh can be assigned a `UMjQuickConvertComponent` with its "Unreal Driven" property enabled, transforming it into a *mocap* body that moves in synchronization with UE's physics.

E. Communication, Metrics, and Replay

The framework supports multiple communication modalities, allowing users to choose between a modified ROS plugin [28] or a high-performance ZeroMQ (ZMQ) bridge. These plugins automate the mapping of MuJoCo and UE sensors to their respective topics or sockets based on the robot's configuration. For empirical evaluation, a `MetricManager` provides a structured framework for recording time-series data directly to CSV. Furthermore, a high-fidelity Replay System caches joint kinematics and sensor streams, enabling researchers to record physical trajectories and deterministically replay them across iteratively augmented visual environments for robust domain randomization testing.

F. Example Deployment of Robots, Controllers, and Visual Planners

The system provides a unified framework in which robots, controllers, and planners operate as in real-world deployments. All components communicate through ROS, ensuring seamless integration with external systems.

1) Example Robot Integrations: The framework supports quadrupeds, quadruped mobile manipulators, UAVs, and robotic manipulators, enabling diverse experimental setups for locomotion, aerial navigation, and manipulation.

2) Controllers: The system includes joint controllers based on position and torque, as well as more advanced controllers such as Walk-These-Ways [29] and VBC [30]. UAVs use a PID-based controller, whereas manipulators rely on inverse kinematics (IK) methods such as Mink [31]. Since controllers operate within the ROS ecosystem, new control strategies can be easily integrated.

3) Planners: Motion planning is handled using learning-based and optimization-based planners such as ViNT [32], GNM [33], and NoMAD [34]. The ROS-based architecture of the framework ensures compatibility with additional external planners.

4) Visual SLAM: The system includes visual SLAM techniques such as OrbSLAM2 [35], OrbSLAM3 [36], and MASt3R-SLAM [37], enabling real-time localization and mapping. These methods integrate directly into the framework, enabling realistic perception and mapping in simulated environments.

IV. Experiments

A. Simulation and Real-World Comparision

To evaluate the alignment between simulated and real-world visual inputs for image encoders, we employ Grad-CAM [38], EigenCAM [39], cosine similarity and KL Divergence [40]. These methods assess both spatial attention and feature distribution similarity. Specifically, spatial attention is computed using Grad-CAM and EigenCAM in equations 1 and 2 respectively to assess feature distribution similarity. **Grad-CAM** (Gradient-weighted Class Activation Mapping) computes gradients of class score y^c (pre-softmax logit for class c) w.r.t. feature maps A^k (spatial activations from channel k in the final conv layer):

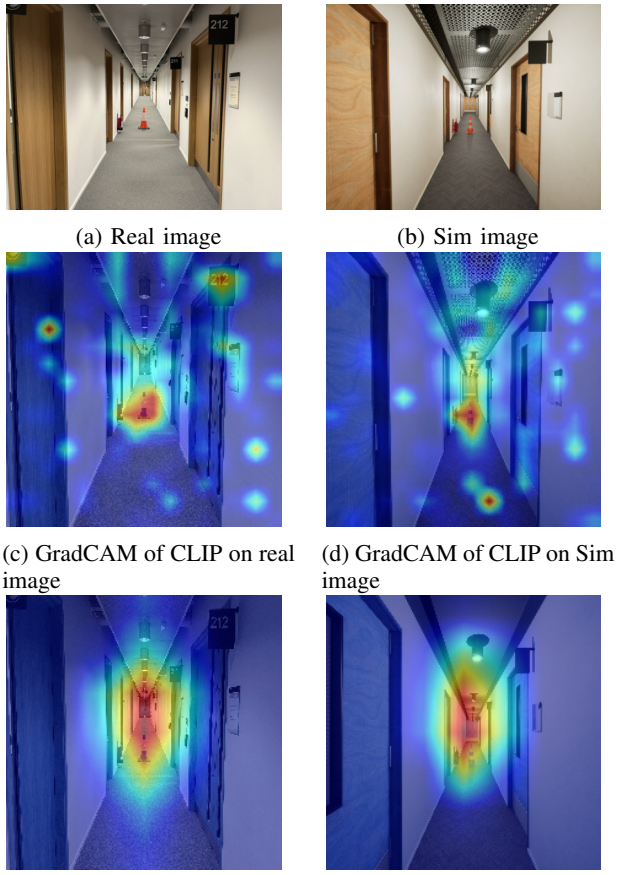
$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}, \quad L^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right) \quad (1)$$

where α_k^c is the Global Average Pooling (GAP)-weighted importance of channel k for class c (i, j: spatial indices; Z: width $\times$ height). Heatmap L^c is the weighted sum of maps with ReLU, yielding a saliency map of regions influencing the model's class c prediction.

EigenCAM generates class-agnostic saliency maps via principal component analysis (PCA):

$$C = \frac{1}{Z}\sum_{i,j}(A_{ij}^k - \mu)(A_{ij}^k - \mu)^T, \quad L_{\text{EigenCAM}} = v_1 A^k \quad (2)$$

where A_{ij} is the activation vector across channels at position i, j; μ is its mean over spatial locations, and v_1 is the first principal component (eigenvector of C). The map L is the projection onto v_1.



(a) Real image (b) Sim image

(c) GradCAM of CLIP on real image (d) GradCAM of CLIP on Sim image

(e) EigenCam of EfficientNet on real image (f) EigenCam of EfficientNet on Sim image

Fig. 5: Comparison of real-world (left) and simulated (right) images, along with their corresponding Grad-CAM (CLIP) and EigenCAM (EfficientNet-B0) heatmaps. The highlighted class label is "cone".

Cosine Similarity measures feature alignment between real and simulated embeddings:

$$S_{\text{cosine}}(P, Q) = \frac{P \cdot Q}{\|P\|\|Q\|} \tag{3}$$

where P and Q are feature vectors. Higher values indicate stronger similarity.

KL Divergence quantifies distribution differences:

$$D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{4}$$

Lower values indicate closer alignment. To compare models trained on real-world data, we use CLIP [41] and the EfficientNet-B0 [42] visual encoder used in ViNT [32]. We first capture an image from the real world and recreate it within our simulator. Then both images are processed using Grad-CAM amd Eigen-Cam to generate attention heatmaps, as shown in Figure 5. In both cases, the models primarily attend to the traffic cone in the center of the hallway as well as the end of the corridor, suggesting that the most salient features are preserved between the real and simulated environments. This alignment indicates that the models are focusing on the same high-level structures regardless of domain, reinforcing the realism of the simulated scene in terms of feature representation. The similarity in attention patterns also suggests that key objects remain distinguishable across domains, which is critical for transfer learning and sim-to-real applications. We further quantified the visual similarity between our hand-crafted simulations and real-world scenes by computing the KL divergence and cosine similarity for three scenes. These simulations were not built as photo-realistic replicas but as a proof of concept that our system can generate scenes visually representative of real conditions. As detailed in Table I, the Hallway scene, for example, yielded a KL divergence of 0.4679 and a cosine similarity of 0.7157, and an average of 0.4739 and 0.7094 in all scenes. These findings, along with our GradCAM and EigenCAM comparisons, confirm that our simulation is visually realistic enough for our evaluation purposes.

Metric	Office Hallway	Flat Hallway	Bedroom	Avg
KL divergence	0.4679	0.4435	0.5103	0.4739
Cosine similarity	0.7157	0.7231	0.6895	0.7094

TABLE I: Image comparison metrics between real-world scenes and simulated recreations.

B. Visual Navigation and SLAM Benchmarking

We benchmark standard methods to demonstrate simulator realism. We evaluated visual navigation (Experiment 1) with GNM [33] and ViNT [32], and visual SLAM (Experiment 2) with ORB-SLAM2 [35], ORB-SLAM3 [36], and MASt3R-SLAM [37]. Each method was run 5 times per condition, averaging results across trials. Both experiments used the Unitree Go1 quadruped with its onboard camera and the Walk-These-Ways low-level controller for realistic deployment. The benchmarks were conducted in two distinct environments—warehouse and residential house—under three adversity levels: none, minor and major, classified by human judgment. UE's Niagara system simulated environmental effects (smoke, rain, pollution, snow), while dynamic lighting and NPC movements were added for major adversity. Figure 1 shows example environments and Figure 6 shows examples of adversity levels.

(a) None (b) Minor (c) Major

Fig. 6: Examples of the different adversity levels for the House Environment.

1) Experiment 1: Visual Navigation: For visual navigation benchmarking, the robot was initialized at identical starting locations across environments and adversity levels. Each method had a fixed duration to reach the goal using a pre-built image-based topological map. The warehouse and residential house were used for this experiment. Methods were not fine-tuned, relying on a simplified topological map approach from the authors' original repository [34], which may limit performance [43].

Success Weighted by Collision (SC). Following [44], we adopt the SC metric:

$$SC = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{1}{1+c_i}, \tag{5}$$

where S_i is the success indicator for episode i, c_i is the number of collisions, and N is the total episodes. Higher SC values indicate fewer collisions in successful runs, while lower values reflect frequent or severe collisions. Alongside SC, we measure success rate, total collisions, time to goal, and goal distance.

Env	Adversity	Method	Success rate↑	Collisions↓	Time to Goal↓	Goal Dist↓	Weighted SC↑
House	Major	GNM	0.2	13	0	7.01	0.0154
		ViNT	0	9.2	0	8.41	0
	Minor	GNM	1	1.2	71.59	1.99	0.5333
		ViNT	0.4	2.8	0	4.75	0.09
	None	GNM	1	1.8	147.86	1.99	0.3667
		ViNT	1	0.8	141.35	2.00	0.7333
Ware-house	Major	GNM	0	12.8	0	8.33	0
		ViNT	0	6.4	0	7.96	0
	Minor	GNM	0.4	4	107.62	4.69	0.1067
		ViNT	0.2	6.6	54.91	6.61	0.0250
	None	GNM	0.8	3.4	102.86	2.53	0.1889
		ViNT	0.8	4.4	90.14	2.59	0.3667

TABLE II: Performance of different Visual navigation methods under various environments and varying adverse effects.

Table II presents the results, including Weighted SC to factor in collision penalties. State-of-the-art methods perform well under normal conditions, with ViNT generally outperforming GNM. Under minor adversity, GNM maintains a higher success rate but takes longer, while ViNT completes faster when successful. However, performance declines as environmental adversity increases (e.g., obstacles, smoke, rain), with both methods experiencing more collisions, longer times to goal, and lower SC. In severe adversity, neither method succeeds in the Warehouse, leading to zero SC. These failures are likely attributable to the dense, volumetric smoke simulated by the Niagara system, which caused persistent occlusion and aliasing of key visual features, preventing the vision-based planners from identifying a valid path. This suggests that these models struggle in extreme conditions due to limited training data, highlighting the need for training in diverse high-adversity conditions, which can be generated more effectively in simulation. The ability to create controlled but realistic adverse scenarios in a simulated environment provides a path to improve robustness in real-world deployments.

2) Experiment 2: Visual SLAM: This experiment used our replay system, where a teleoperated robot followed predefined trajectories optimized for SLAM. Initial runs recorded robot poses and sensor data, which were later replayed with identical conditions but varying visual effects. Each SLAM method was then evaluated, and performance metrics were computed.

Absolute Trajectory Error (ATE) measures the global accuracy of an estimated trajectory by computing the RMSE between estimated positions $\hat{\mathbf{p}}_i$ and ground truth positions $\mathbf{p}_i$:

$$\text{ATE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{p}_i - \hat{\mathbf{p}}_i)^2} \tag{6}$$

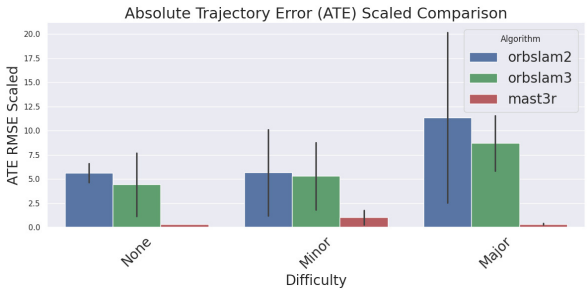
Coverage measures the ratio of the estimated trajectory length to the ground truth:

$$\text{Coverage} = \frac{\sum_{i=1}^{n-1}\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{\sum_{i=1}^{n-1}\|\hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i\|} \tag{7}$$

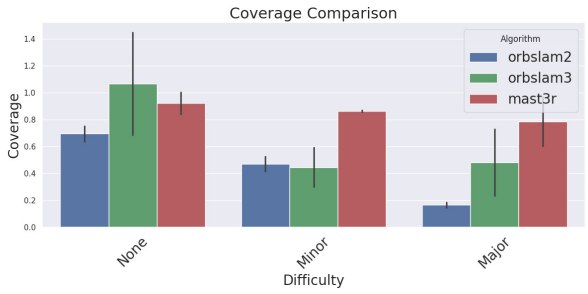
Note that since monocular SLAM estimates trajectories with arbitrary scale, Umeyama's method [45] is applied for scale and pose correction before computing coverage. Poor tracking or alignment can result in coverage exceeding 100%.

Scaled ATE adjusts ATE for localization failures:

$$\text{Scaled ATE} = \frac{\text{ATE}}{\text{Coverage}} \tag{8}$$



(a) Scaled ATE



(b) Coverage

Fig. 7: comparision of different Visual SLAM algorithms under different visual effects

From Table III and Fig. 7, Visual SLAM performance degrades with increasing visual adversity, highlighting sensitivity to challenging conditions. ORB-SLAM2 struggles the

Env	Adver.	OrbSLAM2 Mono			OrbSLAM3 Mono			MASt3R-SLAM		
		ATE↓	Scaled ATE↓	Cover-age↑	ATE↓	Scaled ATE↓	Cover-age↑	ATE↓	Scaled ATE↓	Cover-age↑
Ware-house	None	4.23	6.61	0.639	5.29	7.72	0.685	0.324	0.322	1.01
	Minor	5.33	10.1	0.527	5.21	8.80	0.592	1.55	1.77	0.872
	Major	2.96	20.2	0.147	4.29	5.87	0.731	0.252	0.417	0.605
House	None	3.49	4.64	0.752	1.69	1.17	1.45	0.274	0.326	0.840
	Minor	0.513	1.23	0.415	0.547	1.81	0.302	0.257	0.300	0.855
	Major	0.475	2.53	0.188	2.70	11.6	0.233	0.275	0.284	0.966

TABLE III: Performance of Different VSLAM Methods under Varying Visual Effects.

Env	Adver.	Frames tracked [%] ↑		
		OrbSLAM2 Mono	OrbSLAM3 Mono	Mast3R SLAM
Mine	Clean	7.83	59.6	100
	Smoky	0.00	38.5	43.6

TABLE IV: Performance of Different VSLAM Methods under Varying Visual Effects in the Real-World.

most, with sharp declines in coverage and increased ATE, particularly in the warehouse environment. ORB-SLAM3 performs well in clean conditions but deteriorates significantly under adversity, suggesting reduced robustness to occlusions, lighting changes, and dynamic elements. MASt3R-SLAM maintains the lowest ATE and highest coverage across conditions, demonstrating superior resilience to visual disturbances. Under severe visual adversity, all methods experience increased ATE and reduced coverage, highlighting the inherent challenges of maintaining reliable SLAM performance in dynamic or visually degraded environments. These replay-based evaluations provide a controlled, repeatable framework for assessing SLAM robustness across different levels of adversity, reinforcing the need for training and evaluation in diverse, high-adversity conditions to enhance real-world deployment reliability.

3) Experiment 3: Real-world: Visual SLAM algorithms were evaluated on realworld tunnel dataset [46] under two conditions: clear and smoky, where a section of the tunnel was obscured by smoke generated from a smoke machine. Examples of these conditions are shown in Fig. 8. The percentage of tracked frames under these conditions was measured for various visual SLAM algorithms, results can be seen in Tab. IV, where it can be seen that the tracking performance of all SLAM algorithm degrades significantly under smoky conditions, consistent with the findings from our simulated experiments.

4) Discussion: Our experiments collectively demonstrate that current state-of-the-art visual navigation and SLAM algorithms, while proficient in ideal conditions, remain brittle to the types of realistic, adverse visual phenomena that are produced by our framework and that are encountered in the realworld. The observed performance degradation highlights a critical gap in current training and testing paradigms. Futhermore, because adverse real-world datasets are scarce, there is a reinforced need for high-fidelity simulators such as URL, which can generate controlled, repeatable, and diverse high-adversity scenarios to advance the development of more robust robotic perception systems.

(a) Clean (b) With smoke

Fig. 8: Examples of Realworld Tunnel data [46] used for evaluating VSLAM algorithm under adverse visual conditions.

V. CONCLUSION

This paper presents a high-fidelity robotics simulation framework that integrates advanced photorealistic rendering with precise physics modeling to support vision-based robotics research and benchmarking. Using Unreal Engine's rendering capabilities and MuJoCo's accurate physics, our system enables evaluation of robotic navigation and SLAM methods under diverse and adverse environmental conditions. Experimental results, such as utilizing the Replay System to test varied visual effects on identical trajectories, and Niagara-driven adversity benchmarks highlighting planner failures in volumetric smoke, confirm that our simulator provides a robust testbed for evaluating real-world robustness in controlled, repeatable scenarios.

Despite its strengths, the system has certain limitations, including the lack of deformable terrain simulation, one-way kinematic coupling for UE driven Actors, and the absence of validated sensor noise models for depth and LiDAR data. To address these limitations and further improve our simulator, we plan to focus on narrowing the sim-to-real perceptual gap by implementing procedural material randomization, generate large-scale synthetic datasets to enhance vision-based model training, introduce VR teleop for data generation, expand automated scene generation, and enable bilateral physical interactions between MuJoCo and UE. Overall, our platform serves as a robust foundation for benchmarking and data generation, bridging the gap between simulation and real-world robotics deployment.

REFERENCES

[1] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *Advances in neural information processing systems*, vol. 34, pp. 251–266, 2021.

[2] Epic Games, "Unreal Engine." [Online]. Available: https://www.unrealengine.com

[3] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[4] NVIDIA, "Isaac Sim." [Online]. Available: https://github.com/isaac-sim/IsaacSim

[5] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[6] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, *et al.*, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[7] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min, *et al.*, "Habitat 3.0: A co-habitat for humans, avatars and robots," *arXiv preprint arXiv:2310.13724*, 2023.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[9] K. Ellis, H. Zhang, D. Stoyanov, and D. Kanoulas, "Navigation among movable obstacles with object localization using photorealistic simulation," in *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[10] S. Shanks, J. Embley-Riches, J. Liu, A. M. Delfaki, C. Ciliberto, and D. Kanoulas, "Dreamernav: Learning-based autonomous navigation in dynamic indoor environments using world models," *Frontiers in Robotics and AI*, 2025.

[11] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[12] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: multibody dynamics for biomedical research," *Procedia Iutam*, vol. 2, pp. 241–261, 2011.

[13] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[14] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. Karen Liu, "Dart: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

[15] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[16] NVIDIA, "NVIDIA PhysX." [Online]. Available: https://developer.nvidia.com/physx-sdk

[17] M. Roberts, R. Prakash, R. Wang, Q. Leboutet, S. R. Richter, S. Leutenegger, R. Tang, M. Müller, G. Ros, and V. Koltun, "SPEAR: A simulator for photorealistic embodied ai research," http://github.com/spear-sim/spear.

[18] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[19] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu, "robosuite: A Modular Simulation Framework and Benchmark for Robot Learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[20] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu, "RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots," in *Robotics: Science and Systems*, 2024.

[21] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 4397–4404.

[22] Havok, "Havok Physics." [Online]. Available: https://www.havok.com/

[23] G. Authors, "Genesis: A generative and universal physics engine for robotics and beyond," December 2024. [Online]. Available: https://github.com/Genesis-Embodied-AI/Genesis

[24] NVIDIA, Google DeepMind, and Disney Research, "Newton: An open-source, gpu-accelerated physics simulation engine built upon nvidia warp," 2026, accessed: March 2026. [Online]. Available: https://github.com/newton-physics/newton

[25] Google DeepMind and NVIDIA, "MuJoCo Warp (MJWarp): Gpu-optimized version of the mujoco physics simulator," 2026, accessed: March 2026. [Online]. Available: https://github.com/google-deepmind/mujoco_warp

[26] J. Embley-Riches, C. Ciliberto, and D. Kanoulas, "Sage: Scalable automated generation of environments for robotics," *2025 IEEE International Conference on Advanced Robotics (ICAR)*, pp. 226–232, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:284924962

[27] L. Nwankwo, B. Ellensohn, V. Dave, P. Hofer, J. Forstner, M. Villneuve, R. Galler, and E. Rueckert, "Envodat: A large-scale multisensory dataset for robotic spatial awareness and semantic reasoning in heterogeneous environments," *arXiv preprint arXiv:2410.22200*, 2024.

[28] P. Mania and M. Beetz, "A framework for self-training perceptual agents in simulated photorealistic environments," in *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019.

[29] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[30] M. Liu, Z. Chen, X. Cheng, Y. Ji, R. Qiu, R. Yang, and X. Wang, "Visual Whole-Body Control for Legged Loco-Manipulation," *The 8th Conference on Robot Learning*, 2024.

[31] K. Zakka, "Mink: Python inverse kinematics based on MuJoCo," July 2024. [Online]. Available: https://github.com/kevinzakka/mink

[32] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "ViNT: A Foundation Model for Visual Navigation," in *Conference on Robot Learning*. PMLR, 2023, pp. 711–733.

[33] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7226–7233.

[34] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "Nomad: Goal masked diffusion policies for navigation and exploration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 63–70.

[35] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[36] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[37] R. Murai, E. Dexheimer, and A. J. Davison, "Mast3r-slam: Real-time dense slam with 3d reconstruction priors," *arXiv preprint arXiv:2412.12392*, 2024.

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[39] M. B. Muhammad and M. Yeasin, "Eigen-cam: Class activation map using principal components," in *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[40] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[42] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[43] R. Dhruv, "Issue #5 - visualnav transformer," GitHub Issues, 2025, accessed: 2025-03-01. [Online]. Available: https://github.com/robodhruv/visualnav-transformer/issues/5

[44] A. Eftekhar, L. Weihs, R. Hendrix, E. Caglar, J. Salvador, A. Herrasti, W. Han, E. VanderBil, A. Kembhavi, A. Farhadi, *et al.*, "The One RING: a Robotic Indoor Navigation Generalist," *arXiv preprint arXiv:2412.14401*, 2024.

[45] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[46] V. Kubelka, E. Fritz, and M. Magnusson, "Do we need scan-matching in radar odometry?" in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 13 710–13 716.