From Visual Worlds to Actionable Worlds: Real-to-Sim for Robot Learning

ㅁ **성홍제** / 서울시립대학교

요 약

A world model predicts how an environment evolves under an agent's actions, enabling robots to learn and evaluate behaviors without executing every action in the real world. Recent systems have advanced toward explorable 3D environments and controllable long-horizon video generation. However, these models primarily represent visual appearance and do not explicitly provide the object-level geometry, articulation, contact states, and physical properties required for robot interaction. Likewise, 3D Gaussian Splatting offers high-fidelity scene reconstruction and fast rendering, but does not directly provide stable collision geometry or independently manipulable assets. Real-to-Sim for robot learning must therefore move beyond visual scene reconstruction by decomposing scenes into persistent objects, completing occluded geometry, constructing robot-ready assets, estimating physical properties, and predicting action-conditioned state transitions. This article reviews recent progress and discusses key directions toward object-centric and physics-aware robot world models.

I. Introduction

A world model represents an environment internally and predicts how that environment will change under an agent's actions. Early world-model studies learned compact spatial and temporal representations from observations and used the learned dynamics to train policies inside imagined environments [1]. More recent model-based reinforcement-learning methods, such as DreamerV3, have demonstrated that a learned world model can support policy learning across diverse control domains by predicting future states,

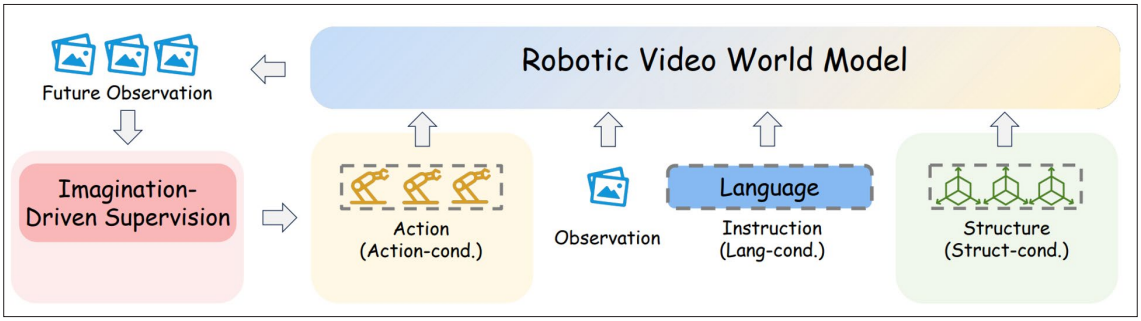
rewards, and episode continuation [2]. In this setting, the purpose of a world model is not to generate visually impressive videos. Its purpose is to provide sufficiently accurate predictions for decision making.

The meaning of the term world model has recently expanded. Advances in large-scale video generation have produced models that synthesize long and interactive visual sequences in response to camera or user control. HunyuanWorld 1.0 generates immersive 360-degree environments from text or images and converts panoramic world proxies into semantically layered 3D meshes [3]. WorldPlay generates long-horizon interactive video under keyboard and mouse control, while reconstructing context from past frames to improve geometric consistency [4]. SANA-WM further scales controllable generation to minute-long 720p videos using a hybrid linear diffusion transformer and 6-DoF camera conditioning [5]. These models show that generative systems are becoming capable of constructing explorable visual worlds rather than isolated images or short clips.

An important point, however, is often overlooked. Current generative world models mainly predict how the world will look after camera or user movement, whereas robot learning requires a model of how the world will change after physical interaction. Moving a camera changes the observation of a scene. Moving a robot changes the state of the scene itself. A robot action may introduce contact, collision, force transmission, articulation, deformation, or irreversible changes in object configuration.

This distinction reveals a mismatch between the current solution and the target problem. The dominant solution is long-horizon video generation, but the robotics problem is action-conditioned physical state prediction. A generated video may appear natural even when an object slightly penetrates a table, changes shape, or follows an incorrect trajectory. Such errors may have little influence on perceptual video quality, but they can completely change whether a robotic policy succeeds.

A recent survey organizes robotic video world models into imagination-based, action-controllable, structure-aware, and foundation-scale models [6]. This progression shows that the field is moving beyond passive visual prediction toward futures that are causally aligned with robot actions and useful for downstream decision making. Nevertheless, even



<Figure 1> Unified progression of robotic video world models from imagination-based generation to action-controllable, structure-aware, and foundation-scale formulations [6]. The progression indicates that visually plausible future generation must increasingly incorporate action alignment and structural priors to become useful for robot learning.

an action-conditioned video is not automatically a simulation. For a robot to act inside the predicted world, objects must have persistent identities, explicit surfaces, collision geometry, articulation, and physical states.

In this article, we refer to visually explorable environments as worlds we see, and environments that support physical interaction and robot learning as worlds we can act in. Real-to-Sim is the process that connects the two. Its goal is not merely to reproduce real-world images in a virtual environment, but to transform real observations into editable, physically executable, and action-conditioned robot worlds.

II. From Visual World Generation to Robot-Ready Worlds

1. Video World Models Generate Observations, Not Physical States

Video world models predict future observations directly in pixel space. This representation is attractive because it preserves rich appearance, motion, and interaction cues without requiring manually defined object states. Large quantities of internet video can also be used for pretraining, providing general visual priors that are difficult to obtain from robot trajectories alone.

However, pixels are observations of a state rather than the state itself. Consider a robot attempting to grasp a cup. A video model may generate a sequence in which the fingers visually overlap with the cup and the cup subsequently moves. The video does not explicitly determine whether contact occurred, which part of the cup was contacted, how much force was applied, or whether the grasp would be stable under gravity. These quantities are critical for robotic control but are only implicitly represented in the generated pixels.

The same limitation applies to hidden states. If a robot places an object inside a drawer and closes it, the object disappears from view. When the drawer is opened again, the same object should reappear at the correct position and orientation. Maintaining a visually similar object is not sufficient. The model must preserve its identity and state while it is unobserved.

Video world models should therefore be viewed as one component of a robot world model rather than as a complete replacement for structured simulation. Their strength lies in modeling visual appearance and complex motion priors. Their strength lies in modeling visual appearance and complex motion priors. Their weakness is that they do not explicitly expose persistent and verifiable physical states.

V-JEPA 2 illustrates a promising intermediate direction. It first learns visual representations from more than one million hours of internet video and then post-trains an action-conditioned latent world model using a much smaller amount of robot interaction data [11]. This approach demonstrates that large-scale observational priors can support robot planning, but it also shows that passive video pretraining alone is insufficient. Robot-specific action alignment is still required.

2. 3D Gaussian Splatting Reconstructs Scenes, Not Assets

Real-to-Sim requires a 3D representation of the

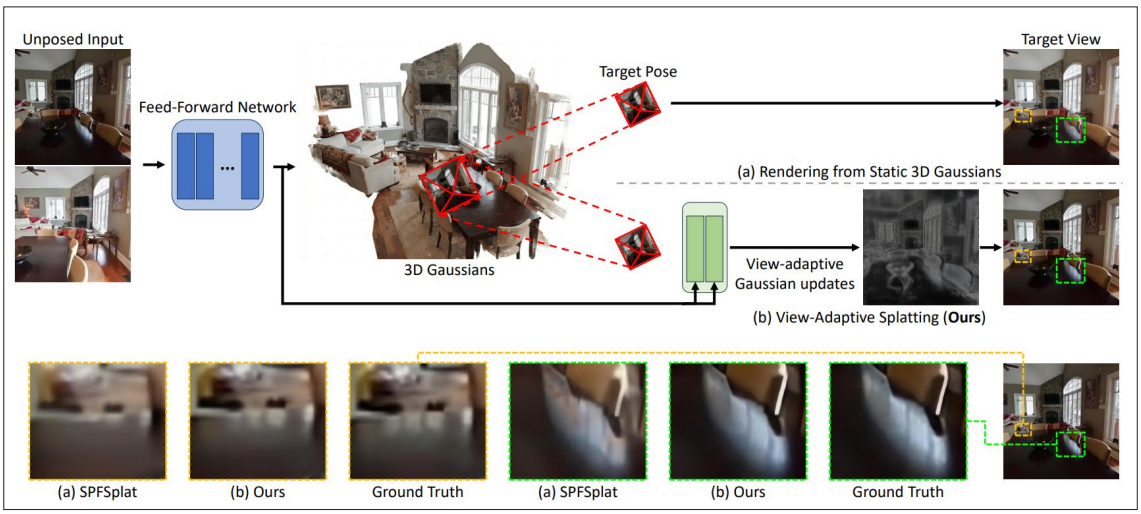
observed environment. Neural radiance fields and 3D Gaussian Splatting have significantly improved the reconstruction of real scenes from multi-view images. In particular, 3DGS represents a scene using explicit Gaussian primitives with position, scale, rotation, opacity, and view-dependent color. Its rasterization-based renderer enables high-quality real-time novel-view synthesis [7].

Recent feed-forward 3DGS models further remove expensive per-scene optimization. Given a small number of input images, these models directly predict camera poses and Gaussian scene representations. ViewSplat follows this direction and addresses the limited capacity of a single static Gaussian representation by predicting target-view-dependent updates to Gaussian position, scale, rotation, opacity, and color [8]. This design substantially improves fine details and view-dependent appearance while maintaining fast rendering.

The strength of this representation also reveals its limitation for robotics. View-dependent Gaussian updates are designed to produce a better image from a target viewpoint. However, the physical surface of an object cannot depend on the position of the camera. A cup must have the same collision boundary regardless of whether it is observed from the front or the side. ViewSplat explicitly notes that multi-view consistency is maintained through a family of view-conditioned approximations rather than through a single rigid geometry, leaving the construction of a consistent underlying geometry as an open direction.

More generally, an unconstrained Gaussian scene is optimized to explain input images. It is not required to define a watertight surface, distinguish object interiors from exteriors, or support stable contact computation. A visually accurate Gaussian scene may therefore contain floaters, semi-transparent regions, duplicated surfaces, or Gaussians that belong ambiguously to multiple objects.

This leads to a critical distinction between a scene



<Figure 2> Static and view-adaptive 3D Gaussian Splatting for feed-forward novel-view synthesis [8]. Target-view-conditioned Gaussian updates improve visual fidelity, but the resulting representation is optimized for rendering rather than for viewpoint-independent collision geometry.

representation and an asset representation. A scene representation is sufficient when the objective is to render the environment from new viewpoints. An asset representation is required when an individual object must be selected, moved, grasped, removed, or simulated independently.

3. From Monolithic Scenes to Object-Centric Assets

A robot interacts with objects rather than with an undifferentiated scene. Cups, boxes, tools, doors, drawers, and containers must therefore be represented as separate assets with persistent identities and individual coordinate systems. When an asset moves, its visual appearance, collision surface, articulation state, and physical properties must move together.

Separating the Gaussians associated with an object is a useful first step, but it does not by itself produce a robot-ready asset. A robot-ready asset typically requires an explicit visual representation and a simulator-compatible collision representation. In current physics engines, this is commonly implemented using a high-detail visual mesh together with a simplified collision mesh or a set of convex collision primitives. The asset should also include metric scale, a canonical coordinate system, and physical parameters. This article emphasizes mesh-based assets because they are broadly supported by existing physics engines, although the underlying requirement is a stable and explicit collision representation rather than a particular surface format.

Recent work is beginning to address the transition from monolithic worlds to object-centric scenes. WorldAct, for example, decomposes generated 3D worlds into actionable objects, reconstructs object-level meshes, restores the residual background, and enables collision-aware manipulation [9]. Its motivation is closely aligned with the central argument of this article: an explorable 3D world is not necessarily an interaction-ready world.

For real scenes, the problem is even more challenging because not every object surface is observed. The bottom of a cup may be occluded by a table, the interior of a drawer may be hidden, and the back side of an object may never appear in the captured views. 3D reconstruction should preserve the observed geometry, while a generative model completes only the unobserved regions.

The output of this process should not be judged only by visual plausibility. A completed mesh should preserve metric scale, remain consistent with visible evidence, avoid self-intersection, and form a stable surface for collision detection. When multiple completions are equally plausible, the system should retain multiple hypotheses or represent uncertainty rather than committing to one arbitrary shape.

III. Real-to-Sim Construction of Robot-Ready Assets

1. Object-Centric Scene Decomposition

The first requirement of a robot-ready world is a persistent object-centric representation. Multi-view images should be separated into static background elements, movable rigid objects, articulated objects, deformable objects, and the robot itself. Object

masks obtained independently from each frame are not sufficient, because masks from different views and timestamps must be associated with the same physical instance.

This problem combines video object segmentation, multi-view association, 3D reconstruction, and object tracking. An object may become partially or fully occluded, but its identity and state must remain in the world model. The resulting object representation should support a canonical coordinate system so that appearance and geometry remain attached to the same asset when it moves.

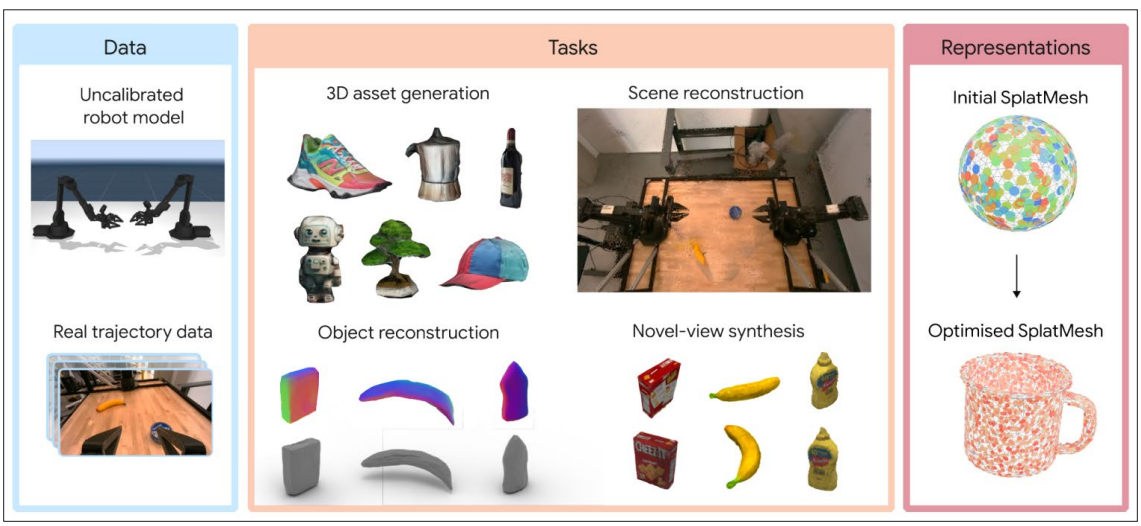
Static structures also require careful treatment. A wall may remain part of the Gaussian background, but a table should usually provide an explicit collision surface. Although the table does not move during a task, it determines whether an object is supported, falling, or penetrating the environment. The distinction between background and asset should therefore depend on physical relevance rather than only on whether an element is visually salient.

2. Hybrid Representation of Appearance and Physics

3DGS and mesh representations should not be treated as competing alternatives. They solve different problems. 3DGS is effective for photorealistic appearance, complex lighting, view-dependent effects, and fast rendering. Meshes are effective for surface definition, collision detection, articulation, and physics simulation.

A practical Real-to-Sim world should combine the two representations. Static scene appearance can be represented by 3DGS, while physically relevant structures and movable objects are represented by explicit meshes. Each object may additionally carry surface-aligned Gaussians or a learned texture to preserve realistic appearance during rendering.

Splatting Physical Scenes provides a representative



<Figure 3> End-to-end Real-to-Sim framework that jointly reconstructs scene appearance, explicit object geometry, robot pose, camera parameters, and physical properties from RGB observations and robot proprioception [10]. The framework combines 3D Gaussian appearance with mesh geometry suitable for physics simulation.

example of this direction [10]. Its SplatMesh representation couples 3D Gaussians for appearance with an explicit triangle mesh for geometry. Differentiable rendering and differentiable physics jointly refine object geometry, visual appearance, robot poses, camera parameters, and physical parameters from imperfect robot trajectory data. The explicit mesh can be imported into a physics engine, while the attached Gaussians maintain high-quality visual rendering.

This hybrid design resolves an important mismatch. A pure 3DGS scene provides visual fidelity but limited physical executability. A pure mesh scene provides collision geometry but may fail to reproduce the complex appearance of the real environment. Real-to-Sim requires both.

3. Articulation and Physical Property Estimation

Recovering a mesh is not sufficient for many everyday objects. Doors, drawers, scissors, lids, and tools consist of multiple connected parts. Their behavior is constrained by articulation. A drawer should be separated into a body and a movable link connected by a prismatic joint. A door should contain a revolute joint with an appropriate axis and range.

Some articulation can be inferred from geometry and category-level priors, but visual observation alone may be ambiguous. Active robot interaction can provide stronger evidence. A robot may push or pull a handle in multiple directions and observe the resulting motion. The observed trajectory can then be used to infer whether the mechanism is prismatic, revolute, compliant, or fixed.

The same principle applies to physical properties. Mass, center of mass, inertia, friction, restitution, and damping are generally not identifiable from a static image. A heavy object with low friction and a light object with high friction may respond similarly to one weak push. Multiple exploratory interactions are therefore required to identify the parameters that determine state transitions.

Let p denote the physical parameters of an asset, $y_{1:T}$ a real observed trajectory, and $\hat{y}_{1:T}(p)$ the corresponding simulated trajectory. System identification can be formulated as

$$p^* = \arg\min_p \sum_{t=1}^{T} D\big(y_t, \hat{y}_t(p)\big),$$

where D measures differences in object pose, velocity, articulation, or contact state. Differentiable physics enables gradients from observation error to update physical parameters directly. When the observations do not uniquely determine the parameters, the system should estimate a distribution over plausible values and train policies across that distribution.

This interaction-based estimation changes the role of the robot. The robot is not only a policy executor. It also becomes an active sensor that reduces uncertainty in the constructed world.

Ⅳ. Toward Actionable Robot World Models

1. Robot Actions Are Embodiment Dependent

A video world model requires an action-conditioning signal. Camera motion is relatively easy to represent because a 6-DoF camera trajectory has the same geometric meaning across scenes. Robot actions are more difficult. A joint command is meaningful only with respect to a particular robot, controller, kinematic structure, and control frequency.

Conditioning a shared world model directly on robot-specific vectors forces the model to learn two processes simultaneously. First, it must learn how a command is realized as robot motion. Second, it must learn how the environment responds to that motion. These processes should be separated whenever an explicit robot model is available.

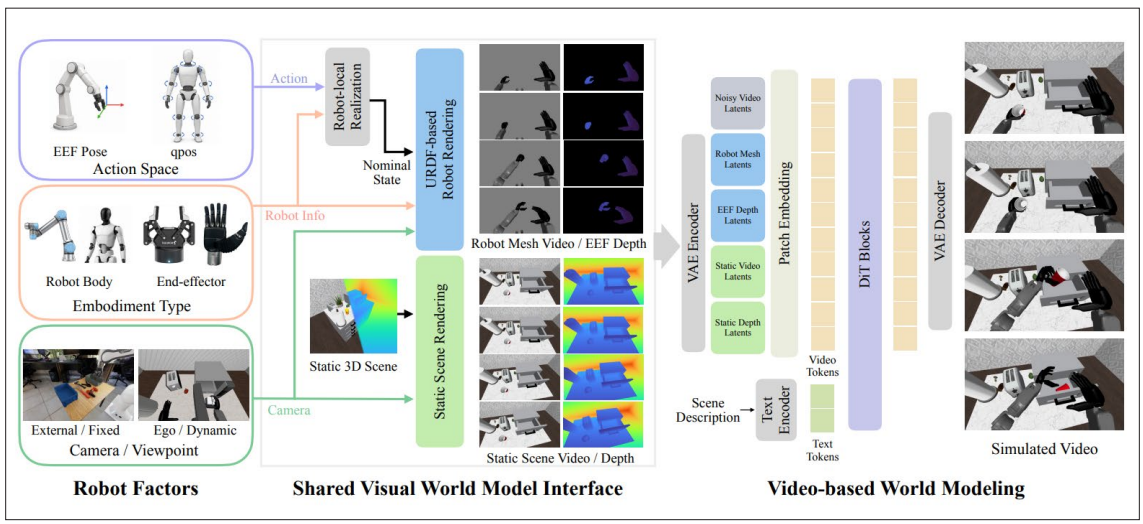
Robot-Factored World Models takes this approach [12]. A candidate action is first passed through the robot's controller and kinematics to obtain a nominal trajectory that is available before scene interaction. The trajectory is then rendered using the robot URDF. Static RGB and depth provide the scene context, while rendered robot geometry provides the action condition. The learned video model can therefore focus on predicting the scene response around the expected robot motion. This interface also reduces dependence on a particular joint representation and supports generalization to unseen robot embodiments.

2. Predicting Structured State and Visual Observation Together

A robot world model should predict more than the next image. Let s_t denote a structured world state and a_t a robot action. The model should estimate

$$p(s_{t+1:t+H}|s_t, a_{t:t+H-1}),$$

where s_t includes robot configuration, object poses, velocities, articulation states, contact relations, and persistent object identities. A renderer can



<Figure 4> Visual interface for a robot-factored world model. Static RGB and depth carry scene and viewpoint information, while rendered nominal robot geometry carries the action. The video model predicts the resulting scene response [12].

subsequently convert the predicted state into future RGB, depth, or tactile observations.

Separating state prediction from observation generation has several advantages. If the predicted object pose is correct but the image is blurry, the error can be attributed to appearance rendering. If the image appears plausible but the object pose is incorrect, the error lies in the dynamics model. A pixel-only world model makes this distinction difficult.

Explicit physics and learned dynamics should also be combined. A physics engine can compute gravity, rigid-body motion, collision, and known articulation constraints. A neural model can predict residual effects that are difficult to model analytically, including uncertain friction, compliance, deformable contact, and unmodeled actuator behavior. A hybrid transition can be written as

$$s_{t+1} = F_{\text{physics}}(s_t, a_t, p) + F_{\text{residual}}(s_t, a_t).$$

This formulation avoids asking a video generator to rediscover all physical laws from pixels while retaining sufficient flexibility to reduce the reality gap.

3. Long-Horizon Memory and Hierarchical Prediction

Robotic tasks often extend over many interaction steps. A robot may open a drawer, remove a cup, place it on a table, and close the drawer. Predicting every future RGB frame autoregressively causes visual and state errors to accumulate.

Long-horizon world models should therefore maintain object-centric memory rather than storing only a sequence of previous frames. The memory should record persistent object states and meaningful events, such as an object entering a container or a joint reaching its limit.

A hierarchical representation can further separate long-term logical transitions from short-term physical motion. H-WM combines a high-level logical world model with a low-level visual world model so that symbolic task states remain stable while local visual transitions provide perceptual grounding [13]. Such a representation is particularly useful when the physical execution of a short action must remain consistent with a long sequence of task-level constraints.

The objective is not to replace geometry and physics with symbolic reasoning. Instead, logical state, object-centric state, continuous dynamics, and visual observations should provide complementary levels of abstraction.

4. Real-to-Sim-to-Real as a Closed Loop

An initial Real-to-Sim reconstruction will always contain uncertainty. Occluded geometry is inferred rather than observed, camera and robot calibration may be inaccurate, and physical properties are estimated from a limited number of interactions. The resulting simulator should therefore not remain fixed.

A policy trained in simulation can be deployed on the real robot to collect additional observations. When the policy succeeds in simulation but fails in reality, the discrepancy provides information about the world model. A failure may indicate inaccurate collision geometry, an incorrect joint axis, an erroneous friction estimate, or a missing residual dynamic effect.

The simulator should be updated with this evidence,

followed by further policy learning and real-world deployment. Real-to-Sim should thus be understood as a continuing identification process rather than as a one-time reconstruction step.

Real-is-Sim demonstrates a related closed-loop perspective by continuously aligning an embodied Gaussian digital twin with the real environment and using the simulator throughout data collection, policy training, evaluation, and deployment [14]. Its results also highlight an important use of Real-to-Sim: evaluating policies in a simulator whose success rates remain correlated with real-world performance.

5. Evaluation Beyond Rendering Quality

The evaluation of 3D reconstruction typically emphasizes PSNR, SSIM, LPIPS, or geometric reconstruction errors. Video world models are commonly evaluated through visual quality, temporal consistency, and action adherence. These measurements remain useful, but they do not determine whether a world model is suitable for robot learning.

A robot-ready world model should be evaluated at three levels. At the representation level, evaluation should measure object decomposition, metric geometry, articulation, and persistent identity. At the dynamics level, it should measure trajectory accuracy, contact timing, final object state, and long-horizon consistency. At the policy level, it should measure whether policies trained or ranked in the model exhibit the same success trends in the real environment.

Policy ranking is particularly important. A simulator may have systematic visual or physical errors and still remain useful if it correctly distinguishes strong policies from weak policies. Conversely, a photorealistic simulator may be misleading if incorrect contact dynamics reverse the relative performance of two policies.

The final objective is therefore not to minimize only image reconstruction error. It is to construct a world whose predictions are sufficiently reliable to support robot learning, planning, failure detection, and policy evaluation.

V. Conclusion

Recent generative world models and 3D reconstruction methods can produce increasingly realistic and explorable environments. However, visual plausibility alone is insufficient for robot learning. Robots require persistent object identities, stable collision geometry, articulation, contact states, and physical properties that remain consistent under interaction.

Real-to-Sim should therefore move beyond scene reconstruction toward object-centric, robot-ready world construction. 3DGS can provide high-fidelity appearance, while explicit meshes and physics models provide collision and interaction. These representations should be combined with action-conditioned state prediction, active physical property estimation, and continuous updates from real-world robot experience.

The central challenge is not to generate longer or more realistic videos, but to construct worlds in which robot actions produce reliable physical consequences. Progress toward this goal will require world models that are visually realistic, physically executable, and directly useful for policy learning and evaluation.

참 고 문 헌

[1] D. Ha and J. Schmidhuber, "Recurrent World Models Facilitate Policy Evolution," *Advances in Neural Information Processing Systems*, 2018.
[2] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Control Tasks through World Models," *Nature*, vol. 640, pp. 647-653, 2025.
[3] HunyuanWorld Team, "HunyuanWorld 1.0: Generating Immersive, Explorable, and Interactive 3D Worlds from Words or Pixels," *arXiv preprint* arXiv:2507.21809, 2025.
[4] W. Sun et al., "WorldPlay: Towards Long-Term Geometric Consistency for Real-Time Interactive World Modeling," *Proceedings of the International Conference on Machine Learning*, 2026.
[5] H. Zhu et al., "SANA-WM: Efficient Minute-Scale World Modeling with Hybrid Linear Diffusion Transformer," *arXiv preprint* arXiv:2605.15178, 2026.
[6] B. Hou et al., "World Model for Robot Learning: A Comprehensive Survey," *arXiv preprint* arXiv:2605.00080, 2026.
[7] 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, article 139, pp. 1-14, 2023.
[8] M. Jeong, S. Min, S. Lee, and H. Seong, "ViewSplat: View-Adaptive 3D Gaussian Splatting for Feed-Forward Synthesis," *Proceedings of the European Conference on Computer Vision*, 2026.
[9] J. Hu, J. Guo, J. Cen, C. Yang, S. Li, and W. Shen, "WorldAct: Activating Monolithic 3D Worlds into Interactive-Ready Object-Centric Scenes," *arXiv preprint* arXiv:2605.15843, 2026.
[10] B. Moran, M. Comi, A. Byravan, S. Bohez, T. Erez, Z. Li, and L. Hasenclever, "Splatting Physical Scenes: End-to-End Real-to-Sim from Imperfect Robot Data," *arXiv preprint* arXiv:2506.04120, 2025.
[11] M. Assran et al., "V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning," *arXiv preprint* arXiv:2506.09985, 2025.
[12] B. Kim, T. Kim, H. Cha, and H. Joo, "Robot-Factored World Models via Robot Rendering," *arXiv preprint* arXiv:2607.22535, 2026.
[13] W. Chen et al., "H-WM: Robotic Task and Motion Planning Guided by Hierarchical World Model," *arXiv preprint* arXiv:2602.11291, 2026.
[14] J. Abou-Chakra, L. Sun, K. Rana, B. May, K. Schmeckpeper, M. V. Minniti, and L. Herlant, "Real-is-Sim: Bridging the Sim-to-Real Gap with a Dynamic Digital Twin for Real-World Robot Policy Evaluation," *arXiv preprint* arXiv:2504.03597, 2025.

저 자 소 개



성 홍 제

- 2018년 : 연세대학교 전기전자공학부 학사
- 2023년 : 연세대학교 전기전자공학과 박사
- 2023년 ~ 2024년 : 삼성전자 종합기술원 Staff Researcher
- 2024년 ~ 2025년 : 삼성전자 AI센터 Staff Researcher
- 2025년 ~ 현재 : 서울시립대학교 전자전기컴퓨터공학부 조교수
- 주관심분야 : 영상처리, 로봇비전, 자율주행