REL-WM: GROUNDING WORLD MODEL IN PROGRESS-AWARE SCENE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Successful robot manipulation depends on how interactions between entities evolve toward task completion, yet standard learned world models do not explicitly represent this structure. We introduce ReL-WM (Relational Latent World Model), a framework that grounds latent dynamics in progress-aware scene graphs. These scene graphs retain task-relevant entities and describe their physical interactions, spatial configurations, and affordance compatibility, while temporal qualifiers capture intermediate relational changes. Using simulator-derived scene graphs during training and acting, ReL-WM learns a semantic state that evolves alongside visual dynamics during imagination. Relational milestones extracted from successful demonstrations define a bounded task-progress potential, whose predicted changes guide policy learning. Our experiments show improved success over visual and object-centric model-based baselines across tabletop and mobile manipulation, together with stronger transfer to new objects and zero-shot generalization to unseen layouts and lighting conditions. These findings support explicitly modeling interactions and task advancement as a useful foundation for world models in robot manipulation.

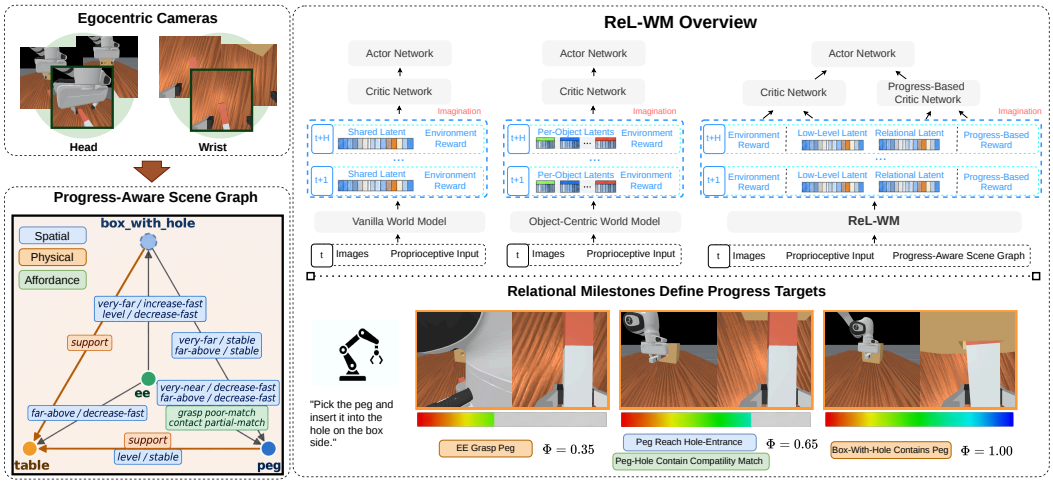

Figure 1: ReL-WM complements visual dynamics with a semantic state grounded in persistent entities and typed manipulation relations. Graph observations inform the acting policy; a learned semantic prior propagates relational information during imagination. Milestones define a bounded progress potential whose discounted changes shape policy learning.

1 INTRODUCTION

Recent advances in latent world models have established them as powerful engines for learning robotic manipulation policies, enabling agents to plan and train through action-conditioned imagination (Hafner et al., 2019; Hansen et al., 2024; Hafner et al., 2020; 2025). However, scaling these models to complex, multi-stage physical tasks reveals a fundamental vulnerability: observation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

prediction alone fails to capture the underlying relational structure of the physical world. Because purely visual dynamics prioritize pixel-level reconstruction, they are prone to physical hallucinations where they struggle to distinguish between visually similar but mechanically distinct interaction states. For instance, a gripper merely touching an object and a gripper securely holding it can produce nearly identical images, yet they yield drastically different outcomes under the next action. Conversely, changes in lighting and background can alter thousands of pixels without changing the manipulation task at all. To learn robust policies, a world model must explicitly preserve the former distinction while tolerating the latter, a capability that requires an explicit understanding of persistent entities and how their interactions evolve over time.

To improve task relevance, existing methods have sought to suppress nuisance variations or isolate controllable dynamics (Deng et al., 2022; Wang et al., 2022; Pan et al., 2022). Object-centric models take this a step further by exposing distinct entities and their dependencies (Kipf et al., 2020; Feng & Magliacane, 2023; Mosbach et al., 2025; Feng et al., 2025). While these structures support robust interaction modeling, they do not inherently specify whether a relation denotes a contact, a secure grasp, or a supporting surface, nor do they track how a specific spatial configuration advances the overarching task. Incorporating scene graphs to maintain persistent spatial and temporal memory provides a pathway to resolving this ambiguity (Kurenkov et al., 2023).

Crucially, successful manipulation requires distinguishing a static configuration from active change. A gripper near a sphere might be approaching it, withdrawing, or remaining still; equal geometric distances do not imply equal grasp readiness when orientation and gripper width are ignored. After a grasp is acquired, successful placement depends entirely on the object's evolving relation to its destination. To capture this, we must represent current physical and geometric relations alongside their recent temporal changes and their compatibility with demonstrated interaction poses, all while maintaining persistent entity identities as the camera moves.

To bridge this gap, we propose ReL-WM, a relational latent world model that explicitly grounds semantic states in progress-aware scene graphs. During observation, a graph posterior encodes the complex interaction state; during imagination, a semantic prior learns to predict how that state evolves alongside low-level visual dynamics. By extracting a fixed schedule of relational milestones from just a few successful demonstrations, ReL-WM defines a bounded task-progress potential that provides a dense shaping signal. The scene graph thus serves a dual purpose: it dictates exactly what the world model must predict about physical interactions, and it supplies the intermediate conditions necessary to guide policy updates.

In our experiments, we utilize privileged simulator information to construct these graphs during training and acting, conditioning the policy on the graph posterior. Future imagined states rely solely on the semantic prior. Using only five demonstrations per task to extract the necessary graph assets and progress schedules, our evaluation isolates the distinct benefits of integrating interaction structure into the complete reinforcement learning agent.

Our primary contributions are as follows:

- A manipulation-oriented graph representation that combines persistent entities, typed physical and geometric relations, and temporal qualifiers to expose intermediate interaction changes.
- A novel relational world model (ReL-WM) that grounds a semantic state in these graphs, predicting its action-conditioned evolution during imagination alongside standard visual dynamics.
- A milestone-based progress guidance mechanism, supported by an analysis of its finite-horizon actor update, and evaluated extensively through control tasks, object transfer, zero-shot generalization, and component ablations.

2 Related Work

While scene-graph memory provides persistent spatial context for robotic policies (Kurenkov et al., 2023; Qian et al., 2026), and recent methods like GraphMimic predict graph structures for imitation learning (Chen et al., 2025), these approaches typically treat relational graphs as auxiliary sensory inputs or static imitation targets. Similarly, while PALM (Liu et al., 2026) incorporates affordance reasoning into progress-aware policies, it operates outside the paradigm of action-conditioned generative dynamics. There remains a critical gap in integrating structural relational reasoning directly into the generative engine of a model-based RL agent. Relying solely on visual prediction during latent

imagination often yields physically inconsistent rollouts during complex interactions, and omits the structured intermediate milestones that help efficiently guide multi-stage policy learning. ReL-WM bridges this gap by elevating manipulation relations from mere observational context into an actively predicted semantic state inside an imagination-trained world model.

Furthermore, although foundational work like Graph Dreamer (Berkes et al., 2025) has explored world models for variable-size heterogeneous graphs, it focuses on general graph dynamics rather than the specific inductive biases required for physical AI, such as typed contact states and temporal affordance changes. Finally, while potential-based shaping (Ng et al., 1999), plan-derived potentials (Grześ & Kudenko, 2008), and shaping within learned world models (Li et al., 2026) provide robust theoretical foundations for policy learning, they traditionally rely on abstract or scalar states decoupled from the scene's actual physical geometry. ReL-WM unifies these distinct threads: it explicitly grounds the latent dynamics in temporal manipulation relations, and it leverages that exact same structured latent state to compute demonstration-derived progress milestones. This dual-purpose integration ensures that both representation learning and policy optimization are strictly anchored to the physical evolution of the manipulation task.

3 Progress-Aware Scene Graphs

Our scene graph captures task-relevant entities and their interactions through three design choices explicitly tailored for robotic manipulation:

- **Action-Centric Edges**: Because manipulation centers on the robot's influence on the world, we prioritize end-effector–object relations, supplemented by key object–object relations that define structural outcomes like support and containment (Table 5).
- **Temporal Evolution**: Task progress is fundamentally defined by change (Liu et al., 2026). Beyond static relational states, we encode recent continuous changes for all spatial and affordance relations. This exposes vital intermediate progress cues, such as an active approach trajectory, even while the coarse relational label remains unchanged.
- **Persistent Object Memory**: To handle partial observability in egocentric views, we decouple entity persistence from immediate visibility. The graph dynamically incorporates newly observed objects while retaining those that leave the camera frame, maintaining stable entity identities despite varying vertex counts.Formally, at time t, the resulting graph is $G_t = (\mathcal{V}_t, \mathcal{F}_t)$. The vertex set $\mathcal{V}_t$ maintains the active task-relevant entities (including the end effector), while the edge set $\mathcal{F}_t$ captures their valid relations, explicitly annotated with both absolute states and temporal changes.

Vertex Set. Given a natural-language instruction and a few frames uniformly sampled from one successful rollout, we use a large language model to identify task-relevant object categories. Each physical instance belonging to these categories is represented by a separate vertex, with the end effector assigned index $i = 0$. At time t, vertex i is represented as $v_{i,t} = \big(e_i, b_{i,t}, c_{i,t}\big)$, where e_i is a learned entity-type embedding, $b_{i,t}$ are normalized 2D bounding boxes across camera views, and $c_{i,t}$ is 3D centroid via depth backprojection using camera parameters and odometry. We retain vertices that leave the camera views and set their bounding boxes to zero. The persistent graph thereby serves as structured memory of previously observed, task-relevant entities. For navigation tasks, the target position is admitted as a vertex.

Relation Set. As shown in Table 5 and visualized in Fig. 2, we organize the relation families into `spatial`, `affordance` and `physical-state`. Every family carries an absolute-state label, while the `spatial` and `affordance` families additionally carry a temporal-change label over the window $t - K : t$, of fixed length K, that complements the coarse absolute state. These relations expose progress cues as structured relational change: `spatial` changes describe how the pairwise geometry between the end effector and object centers evolves, `affordance` changes indicate end-effector movement toward or away from a fine-grained, interaction-ready relative pose, and `physical-state` transitions mark established interactions. We omit relations involving vertices outside the camera views, except for end-effector relations, which can be estimated from robot proprioception. For the target position vertex, we only emit spatial relations between it and the end effector. The subsection currently risks appearing ad h

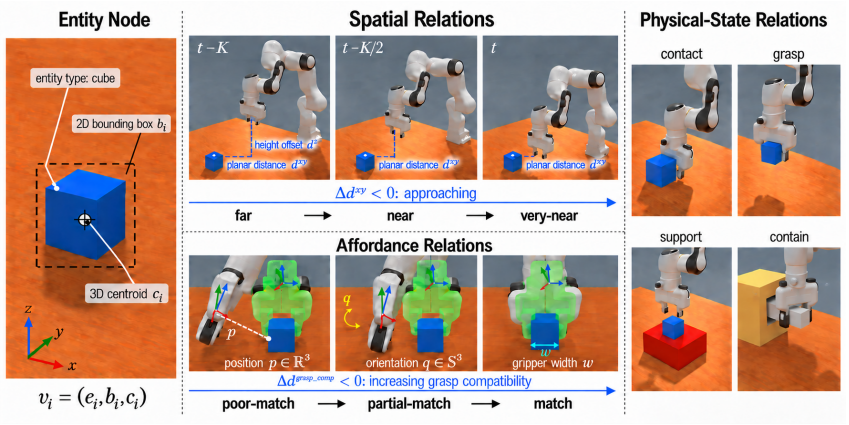

Figure 2: Entity nodes encode semantic type and geometry (left). Spatial and affordance relations describe proximity and interaction compatibility, with temporal qualifiers capturing changes such as approaching an object or improving grasp alignment (center). Physical-state relations identify contact, grasp, support, and containment (right). Together, they expose intermediate progress between discrete interaction events for world-model learning.

Let $\mathcal{I}_t$ denote the admissible relation instances maintained at time t. Following the hyper-relational representation Galkin et al. (2020); Lee & Whang (2025), we represent their qualified facts as $\mathcal{F}_t = \left\{\left(i, \rho, j;\ \text{abs} : \sigma^{\rho}_{ij,t},\ \text{temp} : \delta^{\rho}_{ij,t}\right)\right\}_{(i,\rho,j)\in\mathcal{I}_t}$, where ρ is relation type, $\sigma^{\rho}_{ij,t}$ denotes the current relation state and $\delta^{\rho}_{ij,t}$ its recent change; $\delta^{\rho}_{ij,t} = \varnothing$ for family without a temporal label.

3.1 From Relational Milestones to a Bounded Potential

Before training, we process five successful demonstrations to fix the relevant entities, affordance components, relation thresholds, and progress schedule $\mathcal{S}$. An LLM uses the instruction and demonstration evidence to organize recurring steps into ordered phases. For PlaceSphere, these are approach, grasp, transport, and settle. The schedule remains fixed during learning; no LLM is queried during acting or imagination.

Let $X_t = (G_t, \chi_t)$ contain the graph and available task-site predicates χ_t, such as `reached`. The schedule evaluates these predicates alongside the graph relations. Phase p has nonnegative milestone weights summing to budget W_p, with $\sum_p W_p = 1$. Let $u_p(X) \in [0, W_p]$ sum satisfied milestone weights, $A_p(X)$ indicate its prerequisites, and $V_p(X)$ indicate availability of its required facts. Let $L_p(X) = 1$ when a strictly later phase is observed to be complete. Then

$$\Phi_{\mathcal{S}}(X) = \sum_{p=1}^{P} C_p(X), \qquad C_p(X) = \max\{V_p(X)A_p(X)u_p(X),\ W_p L_p(X)\}. \tag{1}$$

Each C_p lies in $[0, W_p]$, so $0 \le \Phi_{\mathcal{S}} \le 1$. We use the target only when every phase is evaluable or covered by a later completed phase. Thus missing facts do not become zero-progress labels.

Later-phase completion preserves earlier credit: support of the sphere by the bin credits the earlier grasp even after release. This is a rule over the *current* configuration, not a stored achievement history; the potential can decrease. A phase's own completion predicate does not automatically replace its milestone score. Appendix B.2 formalizes these rules, and Appendix D gives every task schedule.

4 Relational Latent World Model

Our method augments DreamerV3 (Hafner et al., 2025) with progress-aware scene graph dynamics and leverages it to guide policy learning. A relation-aware encoder grounds the semantic state in scene graphs, while a learned prior predicts its evolution during imagination (Secs. 4.1 and 4.2). A progress head predicts a potential derived from relational milestones, whose discounted changes provide shaping rewards. The actor combines environment and progress advantages to learn from both signals (Sec. 4.3).

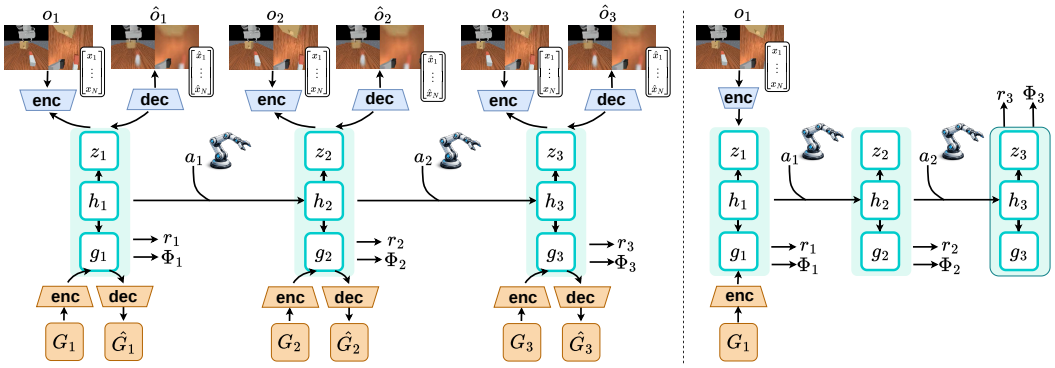

Figure 3: ReL-WM learning. Observation and graph posteriors infer visual and semantic states. Reconstruction and prior–posterior alignment train the model; imagined trajectories use the learned priors. Environment reward and predicted potential changes supply separate value targets and a combined actor advantage.

4.1 Scene Graph Encoder

Eq. The semantic posterior converts the variable-size scene graph G_t into a fixed-dimensional state g_t that captures both entity information and task-relevant pairwise interactions. We initialize each vertex from its entity type and geometry, and embed each directed relation using its type, absolute state, and temporal-change state:

$$x_{i,t}^{(0)} = f_\theta^{\text{node}}\big(e_i, b_{i,t}, c_{i,t}, i\big), \qquad r_{ji,t} = f_\theta^{\text{rel}}\big(\eta_\theta^{\text{type}}(\rho_{ji}), \eta_\theta^{\text{abs}}(\sigma_{ji,t}^{\rho}), \eta_\theta^{\text{temp}}(\delta_{ji,t}^{\rho})\big). \tag{2}$$

The temporal embedding is set to zero for relations without temporal annotations. We then apply L layers of relation-aware message passing:

$$m_{i,t}^{(\ell)} = \frac{\sum_{j \in \mathcal{I}_t(i)} f_{\theta,\ell}^{\text{msg}}(x_{j,t}^{(\ell)}, r_{ji,t})}{\max(1, |\mathcal{I}_t(i)|)}, \qquad x_{i,t}^{(\ell+1)} = f_{\theta,\ell}^{\text{upd}}(x_{i,t}^{(\ell)}, m_{i,t}^{(\ell)}). \tag{3}$$

Instead of encoding entities independently, the messages are conditioned on the semantic state of each entity pair. Each node representation therefore captures both the entity itself and its relations to the surrounding scene. The embedding and message-passing functions are shared across all vertices and edges, following relational message passing (Galkin et al., 2020). Finally, we aggregate the relation-aware node representations into a fixed-dimensional semantic posterior: $g_t = f_\theta^{\text{agg}}\left(\left\{x_{i,t}^{(L)} \mid v_{i,t} \in \mathcal{V}_t\right\}\right)$. This aggregation summarizes the variable-size, unordered scene graph while preserving the relational information propagated by the GNN.

4.2 Learning Visual and Relational Dynamics

We augment the RSSM with a semantic state g_t, in addition to the recurrent state h_t and low-level stochastic state z_t. The semantic posterior is grounded in the current scene graph, whereas the semantic prior predicts the same state from the recurrent dynamics:

$$h_t = f_\phi(h_{t-1}, z_{t-1}, g_{t-1}, a_{t-1}), \qquad \text{recurrent dynamics,} \tag{4}$$

$$\hat{g}_t = f_\theta^{\text{prior}}(h_t), \quad g_t = f_\theta^{\text{post}}(h_t, G_t), \qquad \text{semantic prior and posterior,} \tag{5}$$

$$\hat{z}_t \sim p_\phi(\cdot \mid h_t), \quad z_t \sim q_\phi(\cdot \mid h_t, o_t), \qquad \text{low-level prior and posterior.} \tag{6}$$

The graph posterior provides the dynamics with a structured prediction target at the level of entities and their interactions. Because g_t must preserve node semantics, geometry, absolute relation states, and temporal relation changes, it captures which entities are involved, how they are configured, and how their interactions are evolving. Predicting this state encourages the recurrent dynamics to model action-induced changes in contact, spatial configuration, and affordance compatibility, instead of relying exclusively on low-level visual variation.

The complete latent feature $s_t = [z_t, g_t, h_t]$ is provided to the prediction and policy heads. The fixed dimensionality of g_t decouples the latent dynamics from the number and ordering of graph elements.

Decoder. The graph decoder reconstructs the vertices and qualified relations in G_t from the semantic state g_t and node and relation identity queries:

$$(\hat{e}_i, \hat{b}_{i,t}) = f_\theta^{\text{node}}(g_t, i), \qquad (\hat{\sigma}_{ij,t}^{\rho}, \hat{\delta}_{ij,t}^{\rho}) = f_\theta^{\text{rel}}\left(g_t, i, j, \eta_\theta^{\text{type}}(\rho)\right). \tag{7}$$

Reconstruction is supervised only for maintained vertices and emitted relations, with temporal predictions masked when no temporal annotation is available. The complete latent feature $s_t = [z_t, g_t, h_t]$ predicts the observation, environment reward, continuation flag, and schedule-derived progress potential:

$$\hat{y}_t \sim p_\phi^y(\cdot \mid s_t), \qquad y_t \in \{o_t, r_t, c_t, \Phi_t\}, \tag{8}$$

During imagination, the change in predicted potential provides a shaping reward alongside the predicted environment reward. Image reconstruction uses a stop-gradient copy of g_t, so that the semantic state is shaped by the relational objective and not by pixel reconstruction.

4.3 Progress-aware policy learning

Policy learning from environment returns requires estimating the value of intermediate interactions from experience. We supplement this process with demonstration-derived milestone guidance, following reward-shaping approaches that incorporate subgoal and abstract-plan knowledge (Ng et al., 1999; Grześ & Kudenko, 2008). Our graph represents intermediate interaction states, while the schedule $\mathcal{S}$ assigns them task-specific progress credit. The policy is trained with the combined actor objective in Eq. (10); the progress reward defined below supplies its auxiliary learning signal.

Definition 1 (Imagined progress reward)**.** *Let $\hat{\Phi}_t = \Phi_\theta(s_t) \in [0,1]$ be the progress prediction, supervised by $\Phi_{\mathcal{S}}(X_t)$ during world-model learning. During imagination, s_t includes the predicted relational component. For an imagined transition $s_t \to s_{t+1}$, sample $\tilde{c}_{t+1}$ from the model's Bernoulli continuation predictor and set $\hat{\gamma}_{t+1} = \gamma\tilde{c}_{t+1}$, where $0 < \gamma < 1$. The progress reward is*

$$\hat{r}_{t+1}^{\text{prog}} = \hat{\gamma}_{t+1}\hat{\Phi}_{t+1} - \hat{\Phi}_t. \tag{9}$$

The same effective discount is used in return estimation.

Progress rewards are computed only during imagination; environment rewards remain unchanged in replay and evaluation.

Actor learning criterion. For $u \in \{\text{env}, \text{prog}\}$, let v^u be a separate critic and $R_t^{\lambda,u}$ its horizon-H λ-return target, bootstrapped with $v^u(s_H)$. Both streams use the same imagined trajectory, discounts, and $\lambda \in [0,1]$; the recursion is given in Eq. (69). Define $A_t^u = R_t^{\lambda,u} - v^u(s_t)$. The actor π_ψ minimizes

$$\mathcal{L}_{\text{actor}}(\psi) = -\mathbb{E}_t\Big[\omega_t\left(\log \pi_\psi(a_t \mid s_t)\ \text{sg}(\widetilde{A}_t) + \eta\,\mathcal{H}[\pi_\psi(\cdot \mid s_t)]\right)\Big], \quad \widetilde{A}_t = \frac{A_t^{\text{env}} + \beta A_t^{\text{prog}}}{S_R}. \tag{10}$$

Here, $S_R \geq 1$ is one shared running return scale, $\omega_t = \prod_{j=1}^t \hat{\gamma}_j$ with $\omega_0 = 1$, $\mathcal{H}$ is policy entropy, and sg stops gradients through the advantage. The coefficient β may change between updates but remains fixed within each update. Separate critics allow this weight to change without directly mixing or rescaling the environment critic's reward targets. Appendix B.4 specifies the scale, weight schedule, entropy coefficient η, and critic losses.

Proposition 1 (Policy invariance in the fixed imagined model)**.** *Fix the imagined model, bounded potential, and coefficient β, and assume bounded environment rewards. For the reward in Definition 1, the exact discounted action values satisfy $Q^\pi_{\text{env}+\beta\text{prog}}(s,a) = Q^\pi_{\text{env}}(s,a) - \beta\hat{\Phi}(s)$ for every policy π. Consequently, potential-based shaping preserves the optimal policies of the fixed imagined model (Ng et al., 1999; Grześ, 2017).*

Remark 1 (Effect on learning)**.** *The proposition concerns exact returns in a fixed imagined model. The exact progress critic is $v^{\text{prog}}(s) = -\hat{\Phi}(s)$, for which the raw progress advantage vanishes at every imagination horizon and every λ. With an approximate critic, the additional advantage depends on its approximation residuals (Lemma 2). This characterizes how shaping affects approximate learning, without guaranteeing improved performance. Progress prediction separately supervises the representation, including when $\beta = 0$.*

Appendix E gives the return recursion, proofs, and scope of the analysis under changing models and normalized actor updates.

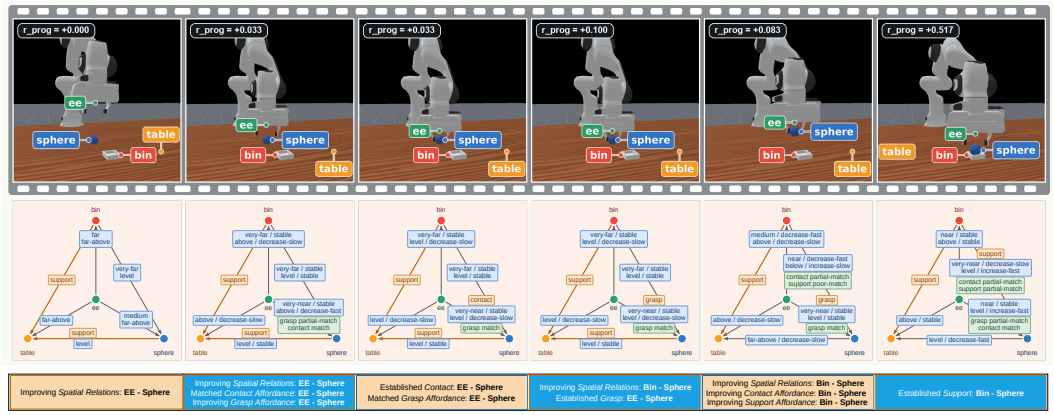

Figure 4: Relational progress along a successful Place-Sphere trajectory. Rows show milestone frames with transition rewards r_{prog}, task-focused scene graphs, and scheduled relational changes. Only transitions that establish a task-relevant relation or cross a scheduled progress threshold are shown.

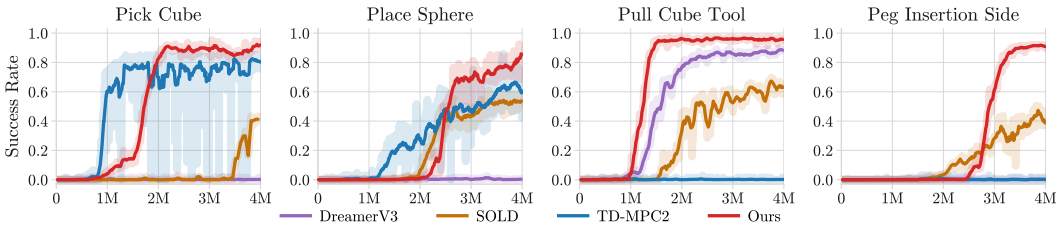

Figure 5: Success rates on four multi-stage robotic manipulation tasks as a function of environment steps, comparing our method with DreamerV3, SOLD, and TD-MPC2.

5 Experiments

We first examine whether the semantic state retains interaction information, then evaluate manipulation success, transfer, and generalization. Ablations separate progress prediction from its use in actor updates. We further evaluate the learned representations for imitation learning in simulation and on **real robot**(Appendix A.1). Additional performance evaluations, detailed analyses, and ablations are provided in Appendix A.

Benchmarks and comparisons. We compare with DreamerV3 (Hafner et al., 2025), TD-MPC2 (Hansen et al., 2024), and the object-centric world model SOLD (Mosbach et al., 2025). On ManiSkill (Tao et al., 2025), we train independently on `PickCube`, `PlaceSphere`, `PegInsertionSide`, and `PullCubeTool` for 4M environment steps each. On MS-HAB (Shukla et al., 2025), a Fetch robot navigates to and grasps YCB objects (Çalli et al., 2015) in ReplicaCAD scenes (Szot et al., 2021); each training setting uses 10M steps.

Implementation Details. Multi-object learning trains one instruction-conditioned policy on five equally sampled targets. Object transfer fine-tunes that policy for 3M steps on held-out targets. Scene generalization trains on five ReplicaCAD variations and evaluates without adaptation on 50 unseen variations. Illumination evaluation scales simulated light intensities by 0.1, 1.0, or 4.0 while retaining the scene and initial physical state. We report benchmark success, not shaped return. Appendix C details the task, input, and evaluation protocols.

5.1 Progress-aware representation

Figure 4 illustrates how our scene graph represents task progress along a successful Place-Sphere trajectory. During approach, changes in spatial relations and contact and grasp compatibility expose progress before physical interaction occurs. Contact and grasp then appear as distinct physical-state transitions, distinguishing touching the sphere from securing it. After grasping, the progress schedule shifts emphasis to the sphere–bin pair: relative-position changes and improving support compatibility

Table 1: Multi-object learning and object transfer. Success rates (%, mean ± standard deviation).

Setting	Object	DreamerV3	TD-MPC2	SOLD	ReL-WM
Multi-Object Learning	Master-Chef-Can	24.2 ± 5.1	19.5 ± 6.8	13.0 ± 9.3	**45.4** ± 5.9
	Pudding-Box	17.5 ± 7.0	12.9 ± 6.7	9.6 ± 6.1	**31.1** ± 6.8
	Sugar-Box	23.7 ± 4.1	18.2 ± 5.1	12.8 ± 9.5	**43.5** ± 5.1
	Tuna-Fish-Can	25.0 ± 5.4	20.8 ± 7.1	15.3 ± 11.2	**47.4** ± 4.6
	Bowl	18.2 ± 5.7	16.3 ± 5.6	10.8 ± 9.1	**39.1** ± 7.2
	Average	21.7 ± 5.5	17.5 ± 6.3	12.3 ± 9.0	**41.3** ± 5.9
Transfer Learning	Tomato-Soup-Can	17.9 ± 4.0	14.6 ± 3.3	9.5 ± 6.7	**43.6** ± 7.2
	Cracker-Box	7.5 ± 2.1	5.8 ± 2.7	2.3 ± 1.5	**18.8** ± 5.7
	Gelatin-Box	19.4 ± 3.6	12.9 ± 3.4	9.8 ± 6.2	**46.7** ± 8.6
	Average	14.9 ± 3.2	11.1 ± 3.1	7.2 ± 4.8	**36.4** ± 7.2

Table 2: Scene and illumination generalization. Zero-shot success rates (%, mean ± standard deviation).

Setting	Evaluation	DreamerV3	TD-MPC2	SOLD	ReL-WM
Scene Generalization	Training scene	45.7 ± 8.2	27.7 ± 5.8	24.5 ± 7.3	**65.3** ± 4.2
	Unseen scenes	34.8 ± 11.8	21.8 ± 9.2	12.3 ± 8.6	**56.4** ± 5.1
Illumination Generalization	Dark ($0.1\times$)	23.4 ± 7.3	13.0 ± 4.1	11.7 ± 9.8	**50.5** ± 6.2
	Bright ($4.0\times$)	35.1 ± 7.8	20.4 ± 6.1	17.1 ± 6.7	**58.3** ± 5.3
	Average	29.3 ± 7.6	16.7 ± 5.1	14.4 ± 8.3	**54.4** ± 5.8

indicate progress toward placement. The largest displayed reward coincides with the bin establishing support, marking completion of the placement phase.

The trajectory illustrates how geometric and affordance changes complement discrete interaction milestones to describe progress throughout manipulation. The table remains represented as context, while the schedule credits milestones involving the entity pairs relevant to each phase. This connects progress rewards to interpretable changes in manipulation state. Complete task schedules are provided in Appendix D. Additional analyses of graph reconstruction and latent sensitivity are presented in Appendix A.

5.2 Model-Based Reinforcement Learning

Figure 5 shows that our method achieves the highest final success rates across all four tasks, which show the value of representing manipulation as a sequence of task-relevant interactions. For example, in `PegInsertionSide`, the randomized hole configuration requires the policy to coordinate grasping, alignment, and insertion for each episode. The corresponding schedule in appendix D distinguishes reaching the hole mouth from achieving insertion alignment and containment, providing intermediate guidance beyond geometric proximity.

We further inspect evaluation trajectories to identify incomplete task execution and shortcuts that meet the success criterion while bypassing the intended interaction sequence. We provided two qualitative samples of representative trajectories in A.4 and include the analysis here:

PlaceSphere. We observe that DreamerV3 may roll the sphere toward the bin for an immediate reward jump, but then gets stuck there and fails to complete placement, whereas ReL-WM follows the grasp-lift-place sequence and leaves the sphere supported by the bin.

PullCubeTool. We observe a less visible failure: DreamerV3 moves the cube directly with the end effector, satisfying the positional success criterion without using the tool. ReL-WM instead grasps the L-shaped tool and uses it to pull the cube into the target region.

5.3 Multi-Object Learning, Transfer Learning and Generalization

Multi-object learning and object transfer. Table 1 shows that ReL-WM consistently outperforms the baselines across training objects and held-out targets after fine-tuning. Across these settings, target identity and geometry change, while the underlying interaction sequence remains similar. Our relational representation expresses approach, grasp acquisition, and return-to-rest through shared semantics, providing comparable manipulation states for world-model and policy learning across

objects. Task-focused entity selection associates these states with the current objective, while the progress schedule assigns credit to completing the corresponding interaction stages. This common structure provides a basis for sharing behaviors across training objects and adapting them to new targets. The transfer results are consistent with this interpretation.

Scene and illumination generalization. Table 2 shows that ReL-WM preserves much of its training-scene performance on unseen layouts and remains effective under both dim and bright illumination. These shifts impose different requirements: layout changes alter geometry and visibility, whereas illumination changes preserve the physical configuration. Robustness across both conditions is consistent with a representation that retains spatial information needed for control while grounding progress in stable interaction semantics.

5.4 Ablation Studies

To evaluate the contributions of semantic latent dynamics, progress-based learning, and robustness to graph noise, we conduct ablation studies on task PlaceSphere. Table 3 summarizes the results.

Ablations	**Success Rate (%)**
Full method ($\beta = 0$)	78.2 ± 7.4
Full method ($\beta = 0.05$)	$\mathbf{86.7 \pm 6.3}$
Full method ($\beta = 0.1$)	84.9 ± 6.6
Full method ($\beta = 0.2$)	82.4 ± 7.0
w/o progress head	71.9 ± 7.8
w/o semantic latent dynamics	24.9 ± 5.5
w/o graph arm (DreamerV3)	3.1 ± 2.1
Full method (1 relation corrupted)	79.8 ± 7.5
Full method (2 relations corrupted)	72.2 ± 8.2
Full method (3 relations corrupted)	68.1 ± 7.3

Table 3: Ablation studies on PlaceSphere, evaluating progress weighting (light yellow), world-model components (light green), and robustness to corrupted scene graph relations (light blue).

Semantic latent dynamics. To assess whether progress-based rewards alone account for the performance gain, we consider *w/o semantic latent dynamics*, which retains progress-based rewards as an additional learning signal but removes the semantic latent and scene graph dynamics modeling. Removing this component substantially reduces success, suggesting that progress-based rewards alone are insufficient to recover the performance of the full method. This comparison supports the benefit of learning a semantic representation with explicit relational dynamics beyond using semantic information only to provide reward guidance. We additionally consider *w/o graph arm*, corresponding to the DreamerV3 reference.

Progress prediction and shaping. To isolate reward shaping from representation learning, we compare *w/o progress head* against *Full method* ($\beta = 0$). Predicting progress without using it for shaping ($\beta = 0$) still improves success, proving the auxiliary prediction objective inherently strengthens the learned representation. Activating shaping achieves peak performance at the default weight, though aggressive weighting degrades it. This confirms that progress guidance must complement, rather than overpower, the environment objective (see Appendix A for advantage magnitude analysis).

Robustness to graph noise. To assess sensitivity to imperfect graph information, we inject noise into scene graph relations at half of the steps in each episode. At each selected step, we randomly choose one, two, or three relations from the scene graph relation pool and replace each selected label with an adjacent relation category, such as *far* with *medium*. Success decreases as more relations are corrupted, but the method retains substantial task performance across the tested settings. These results indicate tolerance to intermittent errors between neighboring relation categories, while also showing that graph quality affects policy learning.

6 Conclusion

We presented ReL-WM, a world model that grounds a semantic state in progress-aware scene graphs and predicts its evolution alongside visual dynamics. Relational milestones from a few demonstrations define a bounded progress potential whose predicted changes shape policy learning without altering the optimal policies of the imagined model. Across tabletop and mobile manipulation, ReL-WM improves success over visual and object-centric baselines and transfers better to new objects, unseen layouts, and lighting changes. Ablations attribute most of the gain to the relational semantic state, with progress prediction and moderate shaping adding further improvement.

AI USE STATEMENT

In this work, we used generative AI tools to help prepare the data, implement and edit research code, and draft and improve the readability of portions of the manuscript. We did not use generative AI for other research tasks requiring disclosure. The authors reviewed all AI-assisted work and take responsibility for the final content of this work, including its data, code, text, and claims.

REPRODUCIBILITY STATEMENT

We provide detailed experimental settings, training procedures, models, datasets, evaluation protocols, and parameters in Appendix to facilitate reproducibility

ETHICS STATEMENT

This work does not involve new human-subject data collection or deployment decisions. We use existing research datasets and follow their established research-use protocols.

REFERENCES

Anaïs Berkes, Donna Vakalis, Yoshua Bengio, and David Rolnick. Graph dreamer: Temporal graph world models for sample-efficient and generalisable reinforcement learning. In *Women in Machine Learning Workshop at NeurIPS*, 2025. URL https://openreview.net/forum?id=pHmgNUZixd.

Berk Çalli, Aaron Walsman, Arjun Singh, Siddhartha S. Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015. doi: 10.48550/arXiv.1502.03143. URL https://arxiv.org/abs/1502.03143.

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. DiWA: Diffusion policy adaptation with world models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3378–3400. PMLR, 2025. URL https://proceedings.mlr.press/v305/chandra25a.html.

Guangyan Chen, Te Cui, Meiling Wang, Chengcai Yang, Mengxiao Hu, Haoyang Lu, Yao Mu, Zicai Peng, Tianxing Zhou, Xinran Jiang, Yi Yang, and Yufeng Yue. GraphMimic: Graph-to-graphs generative modeling from videos for policy learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1756–1768, June 2025.

Zichen Jeff Cui, Hengkai Pan, Aadhithya Iyer, Siddhant Haldar, and Lerrel Pinto. DynaMo: In-domain dynamics pretraining for visuo-motor control. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/3b8db54b629e00537b59cbc6612026d7-Abstract-Conference.html.

Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International conference on machine learning*, pp. 4956–4975. PMLR, 2022.

Fan Feng and Sara Magliacane. Learning dynamic attribute-factored world models for efficient multi-object reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Fan Feng, Phillip Lippe, and Sara Magliacane. Learning interactive world model for object-centric reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8187faaf6759ef6d4e93293339bc656e-Abstract-Conference.html.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7346–7359. Association for Com-

putational Linguistics, 2020. URL https://aclanthology.org/2020.emnlp-main.596/.

Marek Grześ. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 565–573. IFAAMAS, 2017.

Marek Grześ and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. In *Proceedings of the 4th IEEE International Conference on Intelligent Systems*, volume 2, pp. 10–22–10–29. IEEE, 2008. doi: 10.1109/IS.2008.4670492. URL https://www.cs.kent.ac.uk/people/staff/mg483/documents/grzes08alamas_plan-based.pdf.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gax6VtDB.

Andrey Kurenkov, Michael Lingelbach, Tanmay Agarwal, Emily Jin, Chengshu Li, Ruohan Zhang, Fei-Fei Li, Jiajun Wu, Silvio Savarese, and Roberto Martín-Martín. Modeling dynamic environments with scene graph memory. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17976–17993. PMLR, 2023.

Jaejun Lee and Joyce Jiyoung Whang. Structure is all you need: Structural representation learning on hyper-relational knowledge graphs. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2025.

Yao-Hui Li, Zeyu Wang, Xin Li, Wei Pang, Yingfang Yuan, Zhengkun Chen, Boya Zhang, Riashat Islam, Alex Lamb, and Yonggang Zhang. From scalar rewards to potential trends: Shaping potential landscapes for model-based reinforcement learning, 2026. URL https://arxiv.org/abs/2602.03201.

Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, Fangqiang Ding, and Ismini Lourentzou. Palm: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. SOLD: Slot object-centric latent dynamics models for relational manipulation learning from pixels. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44911–44935. PMLR, 2025. URL https://proceedings.mlr.press/v267/mosbach25a.html.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287. Morgan Kaufmann, 1999.

Minting Pan, Xiangming Zhu, Yunbo Wang, and Xiaokang Yang. Iso-Dream: Isolating and leveraging noncontrollable visual dynamics in world models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9316769afaaeeaad42a9e3633b14e801-Abstract-Conference.html.

Jianing Qian, Qinhe Peng, Emmanuel Panov, Leonor Fermoselle, Dinesh Jayaraman, Bernadette Bucher, and Tarik Kelestemur. Expanding spatial and temporal context for robotic imitation learning with scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28010–28020, 2026.

Arth Shukla, Stone Tao, and Hao Su. ManiSkill-HAB: A benchmark for low-level manipulation in home rearrangement tasks. In *International Conference on Learning Representations*, 2025.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems*, 2021.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Demonstrating GPU parallelized robot simulation and rendering for generalizable embodied AI with ManiSkill3. In *Proceedings of Robotics: Science and Systems (RSS)*, Los Angeles, CA, USA, 2025. doi: 10.15607/RSS.2025.XXI.021.

Tongzhou Wang, Simon Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 22591–22612. PMLR, 2022. URL https://proceedings.mlr.press/v162/wang22c.html.

Eric Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003. doi: 10.1613/jair.1190. URL https://arxiv.org/abs/1106.5267.

Appendix Overview

This appendix provides additional empirical analyses (Appendix A), method and training details (Appendix B), experimental settings and baselines (Appendix C), task-specific progress schedules (Appendix D) and theoretical analysis (Appendix E).

A Additional Results and Analysis

A.1 ReL-WM Representations for Imitation Learning

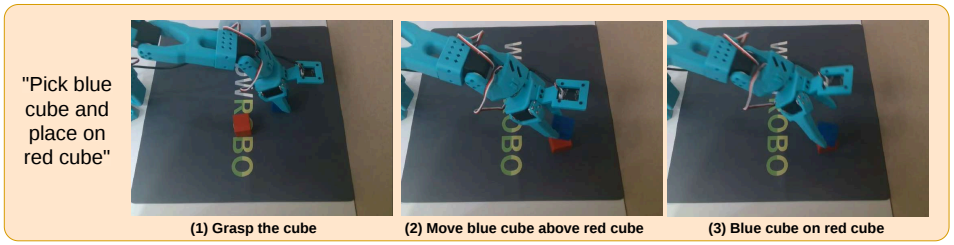

Figure 6: Real-robot cube stacking task. The robot grasps the blue cube, positions it above the red cube, and places it on top to form a stack.

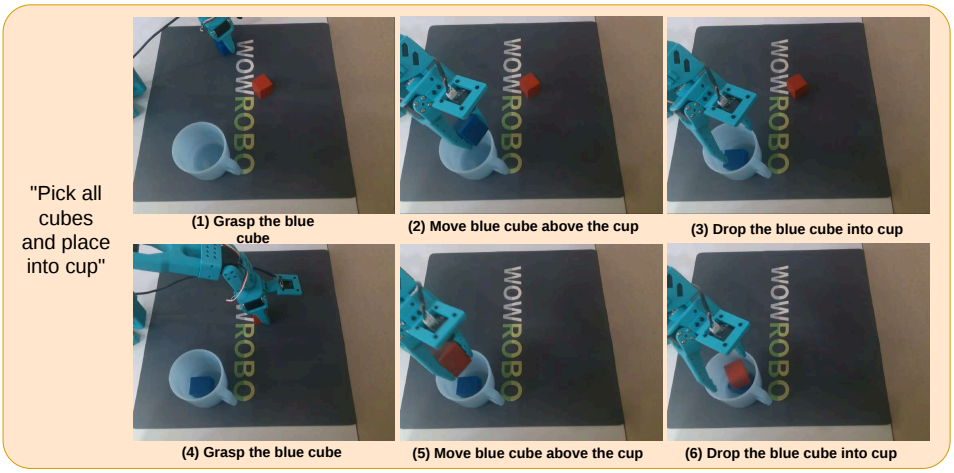

Figure 7: Real-robot cubes-in-cup task. The robot sequentially grasps the blue and red cubes and places both inside the cup, completing two successive pick-and-place operations.

Prior work demonstrates that predictive representations learned through world modeling can provide effective inputs for imitation policies (Hu et al., 2025; Cui et al., 2024; Chandra et al., 2025). We build a simple experiment focused on evaluating whether ReL-WM provides a more informative representation for learning from demonstrations. Details are shown below.

Experimental setup and implementation details. We evaluate imitation learning on four simulated manipulation tasks in ManiSkill: Pick Cube, Stack Cube, Place Sphere, and Peg Insertion Side, and two real-robot tasks, Stack Cube and Cubes-to-Cup. For each simulation task, we collect 1,000 demonstration episodes using motion-planning-based controllers. For the real-robot tasks, we use recorded demonstrations of stacking the blue cube on the red cube and sequentially placing both cubes into a cup. Within each task, ReL-WM and DreamerV3 are trained on the same demonstration trajectories.

We adopt a two-stage training procedure. First, we pretrain each world model using batches of 16 sequences, with a training length of 64 steps and up to 8 preceding steps for recurrent-state initialization. ReL-WM additionally learns graph-conditioned semantic features and predicts task progress from graph-derived potential targets, with the progress objective optimized jointly with the

world model. We then freeze the world model and train a policy conditioned on its posterior features. DreamerV3 provides the deterministic and stochastic latent states, while ReL-WM additionally includes its semantic latent state. A lightweight adapter maps these features to a single conditioning token using two hidden layers of width 1024 with LayerNorm and SiLU activations. This token is combined with the task instruction to condition the SmolVLA action expert. Both methods use the same pretrained SmolVLA checkpoint, adapter hidden architecture, and conditioning-token dimension.

For both simulation and real-robot tasks, we optimize the adapter and action expert using conditional flow matching on normalized demonstration action chunks, while keeping the world model and remaining pretrained backbone fixed. Training uses AdamW with a peak learning rate of 10^{-4}. Posterior features are computed causally from observations and preceding actions, and padded action targets are excluded from the loss. At inference, we generate action chunks with five Euler integration steps and execute the first five actions before replanning. Policy training uses only demonstration action supervision, without progress-shaped rewards or reinforcement-learning updates.

Table 4 reports success rates on real-robot and simulation tasks. Table 4 reports success rates on real-robot and simulation tasks.

Table 4: Imitation-learning success rates. Both methods condition the same SmolVLA action expert on frozen world-model representations.

Method	Real Robot		Simulation			
	Stack Cube	Cubes-To-Cup	Pick Cube	Stack Cube	Place Sphere	Peg Insertion Side
ReL-WM	0.75	0.70	0.60	0.85	0.80	0.65
DreamerV3	0.35	0.15	0.20	0.45	0.25	0.15

A.2 Graph Reconstruction

We evaluate whether the semantic state retains the graph information required by the identity-conditioned decoder. For node i, the decoder predicts its entity category and bounding boxes from (g_t, i). For relation instance (i, ρ, j), it predicts the absolute and temporal labels from the semantic state, endpoint identities, and relation-type embedding:

$$\left(\hat{\sigma}^{\rho}_{ij,t}, \hat{\delta}^{\rho}_{ij,t}\right) = f^{\mathrm{rel}}_{\theta}\left(g_t, i, j, \eta^{\mathrm{type}}_{\theta}(\rho)\right).$$

Ground-truth bounding boxes and centroids are not supplied as decoder queries. Consequently, successful reconstruction requires the semantic state to retain the corresponding graph information.

Figure 8 reports entity-category, absolute-relation, and temporal-relation classification accuracy on training batches throughout world-model learning. Invalid graph positions are excluded. Temporal accuracy is computed only for relations with an available temporal annotation.

Entity-category accuracy exceeds 95% within the first 0.15M environment steps on all four ManiSkill tasks. Absolute-relation accuracy exceeds 95% within 0.3M steps. Temporal-relation accuracy reaches this level after approximately 0.2–0.4M steps on PickCube, PlaceSphere, and PullCubeTool, and after approximately 0.9M steps on PegInsertionSide.

All three reported accuracies exceed 99% by the end of training. The later convergence of the temporal labels on PegInsertionSide is consistent with the greater variety of relative motions required for grasping, alignment, and horizontal insertion.

A.3 Latent Sensitivity to Graph Perturbations

Reconstruction accuracy evaluates whether the decoder can recover the graph targets, but it does not directly show how the graph encoder responds to individual changes in its input. We therefore perform a controlled perturbation analysis on graphs collected from rollouts.

For each original graph G, we construct a perturbed copy G' using one of four edit types:

1. Replace one absolute relation label with another valid label from the same relation domain;

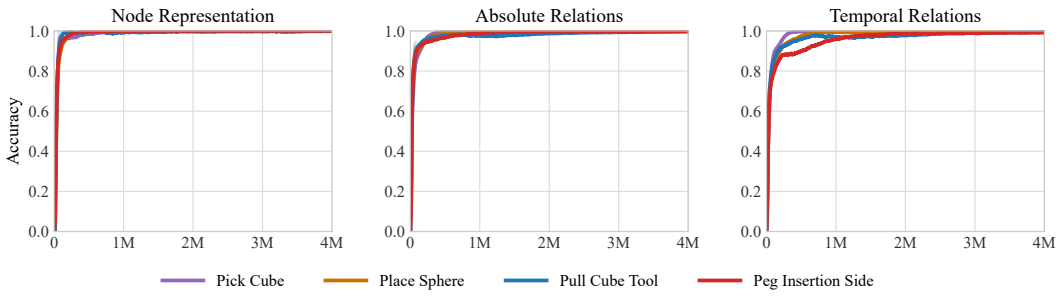

Figure 8: **Graph reconstruction during training.** Entity-category accuracy (left), absolute-relation accuracy (middle), and temporal-relation accuracy (right), measured on training batches as a function of environment steps on four ManiSkill tasks. Invalid graph positions are masked, and temporal accuracy is evaluated only when a temporal annotation is available.

2. Replace one available temporal relation label with another valid temporal label;

3. Shift one object centroid by 1–5 cm while retaining its relation labels;

4. Exchange relation labels between two entity pairs while preserving the overall label counts.

The first two edits test sensitivity to relational state. The centroid edit tests sensitivity to a small geometric change that does not alter the discrete relation labels. The final edit tests whether the encoder represents which entity pair carries a relation, rather than only the multiset of labels. Pairing an unedited graph with itself provides the numerical control.

Let $E(G) \in \mathbb{R}^d$ denote the pooled graph representation. We measure the unnormalized RMS distance

$$D_{\mathrm{RMS}}(G, G') = \sqrt{\frac{1}{d} \sum_{k=1}^{d} \left(E_k(G) - E_k(G')\right)^2}. \tag{11}$$

Results are averaged over five random seeds.

Figure 9 shows a control distance of 3.7×10^{-8}, indicating that repeated encoding of an unchanged graph produces numerically identical representations up to floating-point precision. Absolute-label edits, temporal-label edits, and pairwise label swaps produce distances of approximately 4–6×10^{-2}. Centroid shifts produce a smaller but measurable distance of 2.2×10^{-3}.

The response to absolute and temporal edits shows that both label types affect the pooled representation. The response to pairwise swaps shows that this sensitivity is associated with relation assignment, rather than only with label counts. The smaller response to centroid shifts is consistent with the limited magnitude of the applied geometric changes.

A.4 Qualitative Policy Trajectories

We inspect representative evaluation trajectories to distinguish endpoint success from the sequence of interactions used to reach that endpoint.

PlaceSphere. Figure 10 shows DreamerV3 moving the sphere toward the bin without completing placement. ReL-WM first grasps and lifts the sphere, transports it toward the bin, and then leaves it supported by the bin. The trajectory follows the approach–grasp–transport–settle structure represented by the PlaceSphere progress schedule.

PullCubeTool. Figure 11 illustrates that benchmark success and interaction fidelity can differ. DreamerV3 moves the cube directly with the end effector, satisfying the positional success condition without following the intended tool-use sequence. ReL-WM instead grasps the L-shaped tool, brings it into contact with the cube, and uses it to pull the cube into the target region.

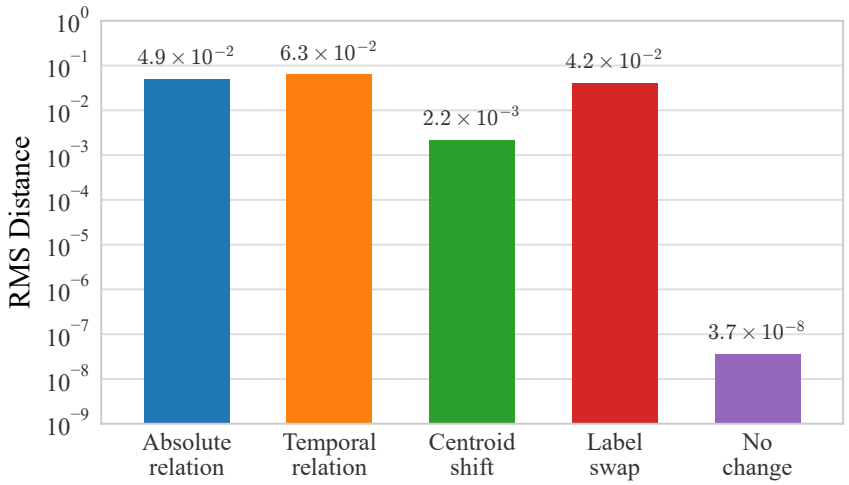

Figure 9: **Latent sensitivity to graph perturbations.** RMS distance between the pooled representations of an original graph and a perturbed copy. Perturbations change an absolute label, a temporal label, an object centroid, or the assignment of relation labels to entity pairs. The unchanged pair provides the numerical control. The vertical axis is logarithmic, and results are averaged over five random seeds.

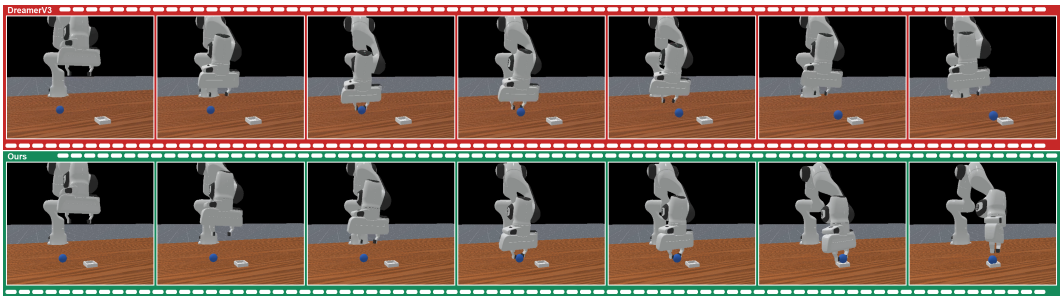

Figure 10: **PlaceSphere evaluation trajectories.** DreamerV3 (red) moves the sphere toward the bin without completing placement. ReL-WM (green) follows the grasp–lift–place sequence and leaves the sphere supported by the bin.

A.5 Progress-Advantage Diagnostics

The progress coefficient controls the contribution of the progress advantage to the actor update. To measure this contribution before the common return normalization, we record

$$\rho_{\text{prog}} = 100 \, \frac{\beta(n) \, \text{mean}_t \left[|A_t^{\text{prog}}| \right]}{\text{mean}_t \left[|A_t^{\text{env}}| \right] + 10^{-8}}. \tag{12}$$

Because the two advantages contain the same number of imagined positions, this mean-absolute ratio is equivalent to their ℓ_1-norm ratio apart from the numerical stabilizer. The statistic is computed before division by the common return scale S_R and therefore preserves the relative magnitude of the two raw advantage streams.

Figure 12 presents this diagnostic for individual runs. Exponential smoothing is applied only to improve visual readability. The shaded regions describe local temporal variation within the plotted runs rather than uncertainty across independent training seeds.

At the beginning of training, $\beta(n) = 0$, so progress does not contribute to the actor advantage. Its influence increases as the coefficient is introduced between 0.2M and 0.7M environment steps. The recorded ratio can subsequently change even after $\beta(n)$ has reached its final value because the potential predictor, progress critic, environment critic, and world model continue to learn.

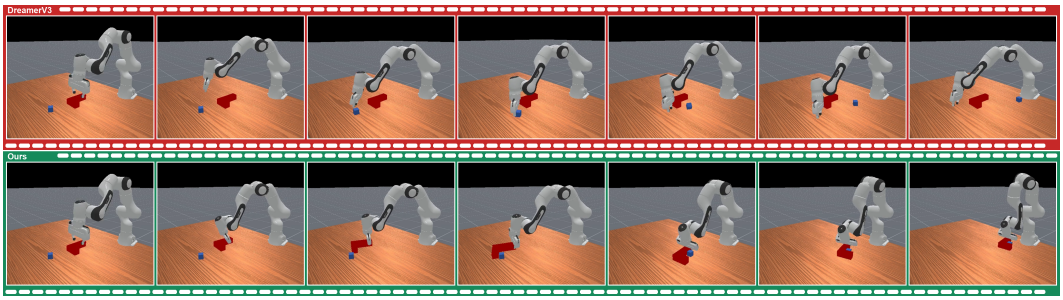

Figure 11: **PullCubeTool evaluation trajectories.** DreamerV3 (red) moves the cube directly with the end effector. ReL-WM (green) grasps the L-shaped tool and uses it to pull the cube into the target region.

Among the tested nonzero weights, $\beta_{\max} = 0.05$ produces the smallest and comparatively steadiest relative contribution and achieves the highest mean success rate in the main ablation table. Larger weights produce stronger fluctuations without improving mean success.

The progress advantage depends on predicted potentials and learned values within finite imagination windows. Errors in either quantity can therefore appear in the progress-return target, and larger progress weights amplify their contribution to the actor update. The observed association supports using moderate progress weighting.

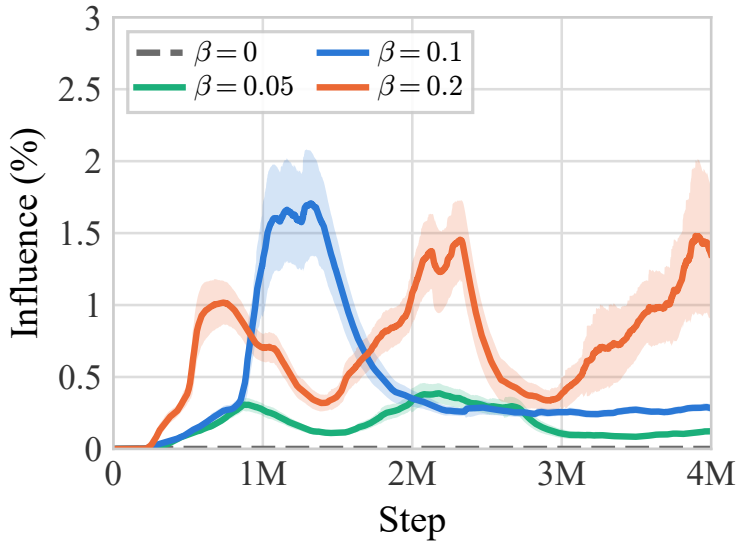

Figure 12: **Relative contribution of the progress advantage.** The plotted statistic is the weighted mean absolute progress advantage as a percentage of the mean absolute environment advantage, as defined in Eq. (12). Lines show exponentially smoothed measurements from individual runs; shaded regions indicate local temporal variation rather than variation across independent seeds.

B METHOD AND TRAINING DETAILS

B.1 SCENE-GRAPH CONSTRUCTION

B.1.1 SIMULATOR INFORMATION AND GRAPH MAINTENANCE

At each environment step, we construct the scene graph from privileged simulator information. Object poses, the end-effector pose, gripper width, contact forces, and the simulator grasp predicate provide the geometric and physical information required to construct vertex attributes and relation labels. In particular, full object poses are used to transform object-local affordance components into the current scene configuration.

Before training, the task-mining phase processes successful rollouts to obtain the task-relevant object set, admissible interaction types for each entity, object-local affordance components, and relation-specific discretization thresholds. These assets remain fixed throughout world-model and policy training.

Each task-relevant physical instance is represented by a separate vertex. Vertex attributes include its entity type, normalized two-dimensional bounding boxes across camera views, and three-dimensional centroid. The end effector is assigned index zero. Vertices are maintained persistently: when an object leaves all camera views, we retain its vertex and set its bounding boxes to zero.

A relation is admissible only when its endpoint types are valid for that relation, its endpoints have the required mined interaction types, and the necessary calibration and affordance components are available. Object–object pairs with no movable endpoint are omitted because their relations cannot be affected by the policy.

A relation is emitted only when every object endpoint lies within at least one camera view. The end effector need not be visually observed because its state is available from robot proprioception. When an object leaves all camera views, its incident relations are omitted even though its vertex is retained. If a relation becomes admissible again, its absolute label is recomputed from the current state. Temporal labels require a new contiguous measurement history, as described in Appendix B.1.7.

B.1.2 RELATION VOCABULARY

The vocabulary separates three kinds of information: realized physical interactions, coarse pairwise geometry, and compatibility with an interaction mode observed in successful rollouts. Spatial proximity and affordance compatibility are represented separately. Proximity specifies where two entities are relative to each other, whereas compatibility evaluates whether their relative pose is suitable for a particular interaction.

Table 5 summarizes the relation families, admissible endpoint types, and associated absolute and temporal variables. Here, `ee-obj` denotes an end-effector–object pair and `obj-obj` denotes an object–object pair.

Physical-state relations carry only an absolute state. Spatial and affordance relations additionally carry a temporal-change state computed from their underlying continuous measurements over the window $t - K : t$. The temporal state qualifies the same relational fact and does not form an independent edge.

B.1.3 PHYSICAL-STATE RELATIONS

The `contact` state is obtained by thresholding the simulator contact force, and `grasp` is given by the simulator grasp predicate. Their absolute labels are `holds` and `not-holds`. Contact and grasp are evaluated independently, so a grasped object may simultaneously satisfy both relations.

For object–object pairs, `support` is determined from contact and the vertical component of the pairwise force. The direction identifies which endpoint supports the other. The `contain` state tests whether the containee's mined key point lies within the entry volume of the container. Specifically, the key point must lie within the radial extent of the opening and within the valid interval along the entry axis.

Table 5: Relation families, admissible endpoint types, and associated absolute and temporal variables. A dash indicates that no separate temporal variable is used.

Family	Pair type	Absolute relation	Change relation
Physical state	ee-obj obj-obj	contact	–
	ee-obj	grasp	–
	obj-obj	support	–
	obj-obj	contain	–
Spatial	ee-obj obj-obj	planar-distance	planar-distance -change
	ee-obj obj-obj	height-offset	height-offset -change
Affordance	ee-obj obj-obj	contact- compatibility	contact-compatibility -change
	ee-obj	grasp- compatibility	grasp-compatibility -change
	obj-obj	support- compatibility	support-compatibility -change
	obj-obj	contain- compatibility	contain-compatibility -change

We store each object–object pair in a fixed endpoint order. Consequently, directed physical relations take values in

$$\{\texttt{not-holds}, \texttt{src-holds}, \texttt{dst-holds}\},$$

where `src-holds` indicates that the source is the supporter or container, and `dst-holds` indicates the reverse direction. This convention keeps the edge identity unchanged when the physical state changes. The four physical predicates are evaluated independently, and multiple predicates may hold for the same pair.

We do not construct separate temporal labels for physical-state relations. Their transitions remain recoverable from consecutive absolute states, so

$$\delta^{\rho}_{ij,t} = \varnothing, \qquad \rho \in \{\texttt{contact}, \texttt{grasp}, \texttt{support}, \texttt{contain}\}.$$

B.1.4 Spatial Relations

For an end-effector–object pair, `planar-distance` is the horizontal distance between the tool center point (TCP) and the object centroid, while `height-offset` is their signed vertical displacement:

$$d^{xy}_{0i,t} = \left\| c^{xy}_{0,t} - c^{xy}_{i,t} \right\|_2, \qquad d^{z}_{0i,t} = c^{z}_{0,t} - c^{z}_{i,t}. \tag{13}$$

A positive height offset places the first endpoint above the second.

For an object–object pair, we apply the same measurements to the corresponding interaction-surface anchors when a unique pair of mined anchors is available. For example, support geometry is measured between the support surface and the bottom surface of the supported object. This avoids measuring interaction geometry from body-frame origins, which may lie far from the relevant surfaces. When suitable anchors are unavailable, the object centroids are used.

The continuous measurements are discretized using thresholds obtained during task mining. Planar distance is mapped to

$$\{\texttt{very-near}, \texttt{near}, \texttt{medium}, \texttt{far}, \texttt{very-far}\},$$

while signed height offset is mapped to

$$\{\texttt{far-below}, \texttt{below}, \texttt{level}, \texttt{above}, \texttt{far-above}\}.$$

The height thresholds are centered around zero so that zero displacement consistently corresponds to `level`. End-effector–object and object–object measurements are calibrated separately because they

occupy different spatial ranges. The resulting thresholds are shared across all training episodes of the task. A spatial relation without a valid mined calibration is not emitted.

The two closest planar-distance states, `very-near` and `near`, define the proximity gate used by the affordance relations. This gate determines whether fine-grained interaction compatibility is exposed as an absolute state; it does not itself indicate that the configuration is compatible.

B.1.5 AFFORDANCE RELATIONS

Multi-modal affordance components. Successful interactions define an object-local affordance set for each interaction type. Components are represented in object coordinates and transformed using the current object pose. Multiple components are retained to represent distinct successful interaction modes.

For end-effector–object grasping, each component contains an interaction anchor, an approach direction, and a preferred gripper width. The active component is selected using the TCP position and orientation and kept stable within the episode. End-effector contact uses the same interaction anchor and orientation but does not include gripper-width mismatch.

Object–object contact uses paired surface anchors and, when available, their outward normals. Support uses a support-surface anchor, surface normal, footprint, and the corresponding bottom surface of the supported object. Containment uses a container entry point, entry axis, opening radius, and depth, together with a key point and axis on the containee. Partner-specific components prevent an interaction region observed for one object pair from being incorrectly applied to another.

Continuous compatibility measurements. For each admissible relation, we compute a normalized incompatibility score:

$$d^{\rho}_{ij,t} = \frac{1}{|\mathcal{M}^{\rho}_{ij,t}|} \sum_{m \in \mathcal{M}^{\rho}_{ij,t}} \min\left(\frac{\epsilon^{m}_{ij,t}}{\alpha_m}, 1\right), \qquad d^{\rho}_{ij,t} \in [0,1]. \tag{14}$$

Here, $\mathcal{M}^{\rho}_{ij,t}$ contains the available mismatch components, $\epsilon^{m}_{ij,t}$ is the mismatch for component m, and α_m is its normalization scale. Missing optional components are excluded from the average rather than treated as maximal mismatch.

The mismatch components are relation-specific:

- **Grasp compatibility:** interaction-anchor distance, approach-direction mismatch, and gripper-width mismatch.
- **Contact compatibility:** interaction-anchor distance and orientation mismatch.
- **Support compatibility:** planar displacement outside the support footprint, vertical surface gap, and surface-orientation mismatch.
- **Containment compatibility:** radial displacement outside the opening, displacement outside the valid axial interval, and entry-axis mismatch.

A lower incompatibility score indicates a relative configuration closer to a successful interaction mode.

Absolute compatibility states. Fine-grained compatibility is exposed only after the pair enters the `very-near` or `near` planar-distance states:

$$\sigma^{\rho}_{ij,t} = \begin{cases} \texttt{unobserved}, & \text{the pair is not near or the score is unavailable,} \\ \texttt{match}, & \text{the pair is near and } d^{\rho}_{ij,t} < \frac{1}{3}, \\ \texttt{partial-match}, & \text{the pair is near and } \frac{1}{3} \le d^{\rho}_{ij,t} < \frac{2}{3}, \\ \texttt{poor-match}, & \text{the pair is near and } d^{\rho}_{ij,t} \ge \frac{2}{3}. \end{cases} \tag{15}$$

Here, "near" includes both `near` and `very-near`.

The `unobserved` label denotes an admissible relation whose fine-grained compatibility is not currently exposed, whereas `poor-match` denotes an evaluated but incompatible relative configuration. Relation omission has a different meaning: it indicates that the endpoints or required relation assets are unavailable.

Table 6: Absolute-state label domains. The direction of `support` and `contain` is encoded relative to the fixed source and destination ordering.

Family	Relation	Absolute labels
Physical state	`contact`	`not-holds`, `holds`
	`grasp`	`not-holds`, `holds`
	`support`	`not-holds`, `src-holds`, `dst-holds`
	`contain`	`not-holds`, `src-holds`, `dst-holds`
Spatial	`planar-distance`	`very-near`, `near`, `medium`, `far`, `very-far`
	`height-offset`	`far-below`, `below`, `level`, `above`, `far-above`
Affordance	`contact-compatibility`	`match`, `partial-match`, `poor-match`, `unobserved`
	`grasp-compatibility`	`match`, `partial-match`, `poor-match`, `unobserved`
	`support-compatibility`	`match`, `partial-match`, `poor-match`, `unobserved`
	`contain-compatibility`	`match`, `partial-match`, `poor-match`, `unobserved`

Whenever $d^{\rho}_{ij,t}$ remains measurable, its continuous value is retained for temporal labeling even when the absolute state is `unobserved`. This allows the graph to capture movement toward or away from an interaction-ready configuration before the pair crosses the proximity gate.

For object pairs already connected by a physical relation at episode initialization, we omit their compatibility relations so that the affordance set represents configurations that can be acted upon rather than static scene layout.

B.1.6 Absolute-State Label Domains

Table 6 lists the complete absolute-state label domains. Physical relations explicitly represent both satisfied and unsatisfied states. The direction of support and containment is encoded by the `src-holds` and `dst-holds` labels.

Spatial relations take an ordered geometric label whenever they are emitted. The `unobserved` label is used only for admissible affordance relations whose fine-grained compatibility is not currently exposed. It is not a replacement for an omitted relation.

B.1.7 Temporal-Change Relations

Continuous temporal measurements. For every spatial or affordance relation with a contiguous history of raw measurements, we compute its signed change over K steps:

$$\Delta u^{\rho}_{ij,t} = u^{\rho}_{ij,t} - u^{\rho}_{ij,t-K}. \tag{16}$$

Here, $u^{\rho}_{ij,t}$ is the continuous measurement underlying the absolute relation. For spatial relations, it is planar distance or signed height offset. For affordance relations, it is the continuous incompatibility score $d^{\rho}_{ij,t}$.

A negative planar-distance change denotes approach, while a positive change denotes separation. A negative height-offset change means that the source moves downward relative to the destination, while a positive change means that it moves upward. Because a lower incompatibility score is preferable, a negative affordance change indicates improving compatibility and a positive change indicates worsening compatibility.

Temporal discretization. Let $m_{\rho} > 0$ denote the robust change scale estimated during task mining. For spatial relations, the scale is estimated separately for end-effector–object and object–object pairs.

Table 7: Temporal-change label domains over $[t-K, t]$. Labels describe the signed change in the continuous measurement underlying each absolute relation. Physical-state relations have no separate temporal variable.

Family	Change relation	Change labels
Spatial	`planar-distance -change`	`decrease-fast`, `decrease-slow`, `stable`, `increase-slow`, `increase-fast`
	`height-offset -change`	`decrease-fast`, `decrease-slow`, `stable`, `increase-slow`, `increase-fast`
Affordance	`contact-compatibility -change`	`decrease-fast`, `decrease-slow`, `stable`, `increase-slow`, `increase-fast`
	`grasp-compatibility -change`	`decrease-fast`, `decrease-slow`, `stable`, `increase-slow`, `increase-fast`
	`support-compatibility -change`	`decrease-fast`, `decrease-slow`, `stable`, `increase-slow`, `increase-fast`
	`contain-compatibility -change`	`decrease-fast`, `decrease-slow`, `stable`, `increase-slow`, `increase-fast`

The signed change is discretized as

$$\delta^{\rho}_{ij,t} = \begin{cases} \texttt{decrease-fast}, & \Delta u^{\rho}_{ij,t} < -3m_{\rho}/5, \\ \texttt{decrease-slow}, & -3m_{\rho}/5 \le \Delta u^{\rho}_{ij,t} < -m_{\rho}/12, \\ \texttt{stable}, & -m_{\rho}/12 \le \Delta u^{\rho}_{ij,t} < m_{\rho}/12, \\ \texttt{increase-slow}, & m_{\rho}/12 \le \Delta u^{\rho}_{ij,t} < 3m_{\rho}/5, \\ \texttt{increase-fast}, & \Delta u^{\rho}_{ij,t} \ge 3m_{\rho}/5. \end{cases} \tag{17}$$

Table 7 lists the temporal label domains. The vocabulary is shared across spatial and affordance relations, but its interpretation follows the underlying measurement. For example, `decrease-fast` indicates rapid approach for planar distance and rapid improvement for an incompatibility score.

History maintenance and missing observations. A temporal label is produced only after $K+1$ consecutive valid measurements. If a relation is omitted or its raw measurement becomes unavailable, its history is cleared. Its temporal target remains masked until a new valid window has accumulated. Temporal history therefore never connects measurements across an interval in which the relation was unavailable.

The affordance proximity gate does not by itself clear the continuous score history. An affordance relation may consequently have an absolute state of `unobserved` while still carrying a valid temporal state. This temporal state describes whether the pair is moving toward or away from an interaction-ready configuration before the absolute compatibility state becomes exposed.

The temporal vocabulary does not include an `unobserved` label. Unavailable temporal evidence is represented by masking, while `stable` denotes an observed change whose magnitude lies within the central interval of Eq. (17).

B.2 Task Mining and Progress-Potential Construction

Task specification from demonstrations. Before world-model training, we process five successful demonstration episodes to construct a fixed task specification. The task instruction and frames sampled uniformly from a successful rollout are used to identify task-relevant entity categories and their semantic roles, such as the manipulated object, destination, and interaction site. Evidence from the successful demonstrations determines the admissible interaction types and provides the object-local affordance components used to evaluate contact, grasp, support, and containment compatibility.

The demonstrations also provide the measurements used to determine spatial discretization thresholds and temporal-change scales. All mined assets are constructed before reinforcement learning and

remain fixed throughout subsequent world-model and policy training. They are not re-estimated from individual training episodes.

Milestone extraction and phase organization. We identify relational milestones that recur across successful demonstrations and use their onset statistics to obtain an initial temporal ordering. An LLM then uses the task instruction to resolve the semantic roles and organize the milestones into an ordered sequence of weighted phases.

Each phase specifies three elements: weighted relational conditions that receive partial credit, predicates that indicate phase completion, and optional prerequisites that gate its partial credit. For example, the PlaceSphere schedule separates approaching the sphere, acquiring a grasp, transporting the sphere toward the bin, and establishing support by the bin. This organization allows spatial and affordance changes to provide intermediate supervision before the corresponding physical interaction is completed.

The complete task-specific roles, phase definitions, milestone conditions, and weights are listed in Appendix D.

Within-phase credit. Let $\mathcal{J}_p$ denote the set of weighted milestones in phase p. For milestone j, let $w_{p,j} \geq 0$ be its weight and let $I_{p,j}(G_t) \in \{0, 1\}$ indicate whether its credited relational condition is satisfied in the current graph. The partial score of the phase is

$$u_p(G_t) = \sum_{j \in \mathcal{J}_p} w_{p,j} I_{p,j}(G_t). \tag{18}$$

The phase budget is

$$W_p = \sum_{j \in \mathcal{J}_p} w_{p,j}, \qquad \sum_{p=1}^{P} W_p = 1. \tag{19}$$

Threshold milestones are cumulative within a phase. For example, if the credited planar-distance sequence is

$$\texttt{far}, \quad \texttt{medium}, \quad \texttt{near}, \quad \texttt{very-near},$$

then satisfying `near` also satisfies the `far` and `medium` thresholds. The task tables use "or better" to denote this convention.

Schedule evaluation. Let $A_p(G_t) \in \{0, 1\}$ indicate that all prerequisites of phase p are satisfied. A phase without prerequisites has $A_p(G_t) = 1$. Let $D_p(G_t) \in \{0, 1\}$ indicate that all completion predicates of phase p are satisfied. Both quantities are evaluated from the current graph and are not stored as temporal latches.

A phase receives its full budget when a strictly later phase is complete. Define

$$L_p(G_t) = \mathbb{1}\left[\exists q > p : D_q(G_t) = 1\right]. \tag{20}$$

This cumulative rule preserves earlier progress after the agent reaches a later stage, even when an earlier temporary relation no longer holds. For example, completion of the final PlaceSphere support phase preserves credit for the earlier grasp phase after the gripper releases the sphere.

Let $V_p(G_t) \in \{0, 1\}$ indicate that all graph facts referenced by phase p are available and unambiguous. The contribution of phase p is

$$C_p(G_t) = \max\left\{V_p(G_t) A_p(G_t) u_p(G_t),\ W_p L_p(G_t)\right\}. \tag{21}$$

The first term gives partial credit when the phase can be evaluated, and its prerequisites hold. The second assigns the full phase budget when completion of a later phase establishes that the earlier phase has been passed.

Completion of phase p does not, by itself, replace its own weighted milestone score with W_p. Its completion predicate instead determines whether preceding phases receive their full budgets. The weighted milestones and completion predicates are listed separately in the task-specific schedules.

Missing graph facts are distinguished from observed conditions that are not satisfied. The schedule target is considered valid when every phase is either directly evaluable or covered by completion of a

later phase:

$$M_{\mathcal{S}}(G_t) = \prod_{p=1}^{P} \mathbb{1}\left[V_p(G_t) = 1 \ \vee \ L_p(G_t) = 1\right]. \tag{22}$$

For valid graphs, the progress potential is

$$\Phi_{\mathcal{S}}(G_t) = \sum_{p=1}^{P} C_p(G_t) \in [0, 1]. \tag{23}$$

Frames with $M_{\mathcal{S}}(G_t) = 0$ are masked in the progress-prediction objective rather than assigned zero progress.

The potential is computed from the current graph rather than maintained as a running maximum. It can therefore decrease when the current relational configuration deteriorates. Cumulative credit arises only from the explicitly ordered phase-completion rules above.

B.3 Network Architecture

Latent state and recurrent dynamics. The world model combines a low-level categorical stochastic state z_t, a deterministic recurrent state h_t, and a deterministic semantic state g_t. Their concatenation,

$$s_t = [z_t, g_t, h_t], \tag{24}$$

is used by the prediction and behavior-learning heads.

The recurrent transition incorporates the previous recurrent, stochastic, and semantic states together with the previous action, as specified in Eq. (4). The low-level posterior infers z_t from the recurrent state and current observation. The semantic posterior is conditioned on the recurrent state and current scene graph:

$$g_t = f_\theta^{\text{post}}(h_t, G_t). \tag{25}$$

The semantic prior predicts the corresponding representation from the recurrent dynamics:

$$\hat{g}_t = f_\theta^{\text{prior}}(h_t). \tag{26}$$

During imagination, future observations and scene graphs are unavailable, so the recurrent world model uses the predicted stochastic and semantic priors.

Graph encoder. For maintained vertex i, the graph encoder combines its entity type, multi-camera bounding boxes, three-dimensional centroid, and vertex identity:

$$x_{i,t}^{(0)} = f_\theta^{\text{node}}\big(e_i, b_{i,t}, c_{i,t}, i\big). \tag{27}$$

Here, e_i denotes the entity-type embedding, $b_{i,t}$ contains the normalized bounding boxes across camera views, $c_{i,t}$ is the world centroid, and i identifies the maintained vertex.

For a qualified relation from vertex j to vertex i, the relation embedding combines its relation type, absolute-state label, and temporal-change label:

$$r_{ji,t} = f_\theta^{\text{rel}}\left(\eta_\theta^{\text{type}}(\rho_{ji}), \eta_\theta^{\text{abs}}(\sigma_{ji,t}^{\rho}), \eta_\theta^{\text{temp}}(\delta_{ji,t}^{\rho})\right). \tag{28}$$

The temporal embedding is set to zero for relations without a valid temporal annotation.

Two layers of relation-aware message passing update the vertex representations. At layer ℓ, incoming messages are averaged:

$$m_{i,t}^{(\ell)} = \frac{\sum_{j \in \mathcal{I}_t(i)} f_{\theta,\ell}^{\text{msg}}\left(x_{j,t}^{(\ell)}, r_{ji,t}\right)}{\max\left(1, |\mathcal{I}_t(i)|\right)}, \tag{29}$$

followed by

$$x_{i,t}^{(\ell+1)} = f_{\theta,\ell}^{\text{upd}}\left(x_{i,t}^{(\ell)}, m_{i,t}^{(\ell)}\right). \tag{30}$$

A permutation-invariant aggregation of the final vertex representations produces a fixed-dimensional graph representation. The graph hidden dimension and semantic-state dimension are both 512. This

fixed-dimensional interface allows the recurrent world model to process graphs with different numbers of maintained vertices and emitted relations.

Graph decoder. Following Eq. (7), the graph decoder uses identity-only queries. For node i, it reconstructs the entity category and bounding boxes from the semantic representation and node identity:

$$(\hat{e}_i, \hat{b}_{i,t}) = f_\theta^{\text{node}}(g_t, i). \tag{31}$$

For relation instance (i, ρ, j), the decoder uses the two endpoint identities and an embedding of the relation type:

$$\left(\hat{\sigma}_{ij,t}^{\rho}, \hat{\delta}_{ij,t}^{\rho}\right) = f_\theta^{\text{rel}}\left(g_t, i, j, \eta_\theta^{\text{type}}(\rho)\right). \tag{32}$$

The endpoint identities determine which pair is reconstructed, while the relation-type embedding identifies the requested relational quantity.

Bounding boxes and centroids are not used as decoder-query inputs. Bounding boxes are reconstruction targets, while bounding boxes and centroids enter the semantic representation through the graph encoder. Consequently, reconstructing node geometry requires the semantic state to preserve the corresponding graph information.

Reconstruction is evaluated only for maintained vertices and emitted relations. Bounding-box targets are masked for camera views in which the object is not visible. Temporal targets are masked when the required measurement history is unavailable. Relation-type masks restrict absolute and temporal predictions to the label domains associated with the requested relation.

Prediction heads and image reconstruction. The complete latent state s_t is supplied to the observation, reward, continuation, progress, policy, and value heads. The progress head predicts the schedule-derived potential through a sigmoid output, ensuring that its prediction lies in $[0, 1]$. Separate environment and progress critics estimate the returns of their respective reward streams.

The image decoder receives a stop-gradient copy of g_t. Consequently, image reconstruction does not directly update the semantic state through this input. Semantic alignment, graph reconstruction, and progress prediction provide the explicit learning signals for the relational representation.

In the MS-HAB multi-object setting, instruction embeddings are included in the observation encoding but are not reconstruction targets. Network dimensions, output representations, and optimization settings are reported in Table 8.

B.4 Training Objectives

Let $\mathcal{O}$ denote the set of observation components reconstructed by the world model. Instruction embeddings are excluded from $\mathcal{O}$. Replay-based expectations are taken over sampled sequence positions using the validity masks specified below, while actor–critic objectives are evaluated on imagined trajectories using cumulative discount weights.

In classification losses, the first argument of CE denotes the predicted class distribution and the second denotes the categorical target. Thus, $e_{t,i}$ in the entity loss is an entity-class index rather than the learned entity embedding used by the graph encoder. The same convention applies to absolute and temporal relation labels.

The complete objective is

$$\begin{aligned}\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{dyn}} + 0.1\mathcal{L}_{\text{rep}} + \sum_{k\in\mathcal{O}} \mathcal{L}_{\text{rec}}^{k} + \mathcal{L}_{\text{rew}} + \mathcal{L}_{\text{con}} \\ + \mathcal{L}_{\text{graphdyn}} + 0.05\mathcal{L}_{\text{graphrep}} + \mathcal{L}_{\text{node}} + \mathcal{L}_{\text{relabs}} + \mathcal{L}_{\text{reltemp}} \\ + \mathcal{L}_{\text{prog-model}} + \mathcal{L}_{\text{policy}} + \mathcal{L}_{\text{value}} + \mathcal{L}_{\text{prog-value}} + 0.3\mathcal{L}_{\text{repval}}.\end{aligned} \tag{33}$$

All unmarked coefficients are one. A stop-gradient operation blocks differentiation through its argument for the corresponding objective; it does not freeze that representation under the other objectives.

B.4.1 World-Model Learning

Latent dynamics and representation. Let $q_\phi(z_t \mid h_t, o_t)$ and $p_\phi(z_t \mid h_t)$ denote the categorical stochastic posterior and prior. Suppressing their conditioning variables for readability, the two KL objectives are

$$\mathcal{L}_{\text{dyn}} = \mathbb{E}_t \left[\max \left(\tau_{\text{KL}}, D_{\text{KL}} \left[\text{sg}(q_\phi(z_t)) \,\|\, p_\phi(z_t) \right] \right) \right], \tag{34}$$

$$\mathcal{L}_{\text{rep}} = \mathbb{E}_t \left[\max \left(\tau_{\text{KL}}, D_{\text{KL}} \left[q_\phi(z_t) \,\|\, \text{sg}(p_\phi(z_t)) \right] \right) \right], \tag{35}$$

where sg denotes stop-gradient and $\tau_{\text{KL}} = 1$ is the free-nat threshold.

The dynamics objective updates the latent prior and recurrent dynamics. The representation objective updates the stochastic posterior and has coefficient 0.1 in Eq. (33).

Observation reconstruction. For the reconstructed observation component o_t^k, the objective is

$$\mathcal{L}_{\text{rec}}^k = -\mathbb{E}_t \left[\log p_\phi(o_t^k \mid s_t) \right]. \tag{36}$$

Image observations use an MSE distribution, while low-dimensional state observations use a symlog-MSE distribution. Each reconstructed observation component has unit weight. Instruction embeddings are encoded in the MS-HAB setting but are not reconstruction targets.

The image decoder receives a stop-gradient copy of g_t. Therefore, image reconstruction does not directly update the semantic state through that input. The semantic representation instead receives explicit learning signals from semantic alignment, graph reconstruction, progress prediction, and the remaining task objectives.

Reward and continuation prediction. The reward head predicts the environment reward on its original scale using a symlog-spaced two-hot distribution:

$$\mathcal{L}_{\text{rew}} = -\mathbb{E}_t \left[\log p_\phi(r_t \mid s_t) \right]. \tag{37}$$

The progress shaping reward is not included in this prediction target and is not added to rewards stored in replay.

The continuation target is

$$c_t = 1 - \mathbf{1}[\text{terminal at } t]. \tag{38}$$

The continuation head is trained using binary cross-entropy:

$$\mathcal{L}_{\text{con}} = -\mathbb{E}_t \left[c_t \log \hat{c}_t + (1 - c_t) \log(1 - \hat{c}_t) \right]. \tag{39}$$

Semantic dynamics. Let g_t^q denote the graph-conditioned semantic posterior and let g_t^p denote the semantic prior predicted from the recurrent state. For a d-dimensional vector, define

$$\text{rms}(x) = \frac{x}{\sqrt{d^{-1} \sum_{i=1}^{d} x_i^2 + \epsilon}}. \tag{40}$$

The semantic alignment objectives are

$$\mathcal{L}_{\text{graphdyn}} = \mathbb{E}_t \left[\left\| \text{rms}(g_t^p) - \text{sg}\left(\text{rms}(g_t^q) \right) \right\|_2^2 \right], \tag{41}$$

$$\mathcal{L}_{\text{graphrep}} = \mathbb{E}_t \left[\left\| \text{rms}(g_t^q) - \text{sg}\left(\text{rms}(g_t^p) \right) \right\|_2^2 \right]. \tag{42}$$

The two losses have the same forward value but send gradients to different branches. $\mathcal{L}_{\text{graphdyn}}$ trains the semantic prior and recurrent dynamics toward the graph-conditioned posterior. $\mathcal{L}_{\text{graphrep}}$ trains the semantic posterior and graph encoder toward the predicted prior. Their coefficients are 1 and 0.05, respectively. Terminal frames whose graph inputs are masked are excluded from both objectives.

B.4.2 Graph Reconstruction and Progress Prediction

Node reconstruction. Following Eq. (7), the node decoder receives the semantic state and the identity i of the node being reconstructed:

$$(\hat{e}_{t,i}, \hat{b}_{t,i}) = f_\theta^{\text{node}}(g_t, i). \tag{43}$$

Bounding boxes and centroids are not supplied as decoder-query inputs. Bounding boxes are reconstruction targets, while node geometry enters g_t through the graph encoder.

The node objective combines entity classification and bounding-box regression:

$$\mathcal{L}_{\text{node}} = \frac{1}{2}\left(\mathcal{L}_{\text{entity}} + \mathcal{L}_{\text{bbox}}\right), \tag{44}$$

$$\mathcal{L}_{\text{entity}} = \mathbb{E}_{t,i}\left[\text{CE}(\hat{e}_{t,i}, e_{t,i})\right], \tag{45}$$

$$\mathcal{L}_{\text{bbox}} = \mathbb{E}_{t,i,c}\left[\text{SmoothL1}_{0.1}(\hat{b}_{t,i,c}, b_{t,i,c})\right]. \tag{46}$$

Entity classification is evaluated over maintained vertices. Bounding-box regression is evaluated only for camera views in which the vertex is visible and is averaged over the four normalized box coordinates.

Relation reconstruction. For relation instance (i, ρ, j), the relation decoder receives the semantic state, the two endpoint identities, and the relation-type embedding:

$$\left(\hat{\sigma}^{\rho}_{ij,t}, \hat{\delta}^{\rho}_{ij,t}\right) = f^{\text{rel}}_{\theta}\left(g_t, i, j, \eta^{\text{type}}_{\theta}(\rho)\right). \tag{47}$$

For emitted relations, the absolute-state objective is

$$\mathcal{L}_{\text{relabs}} = \mathbb{E}_{t,e}\left[\text{CE}(\hat{\sigma}_{t,e}, \sigma_{t,e})\right]. \tag{48}$$

Classes that do not belong to the requested relation type are masked. For example, a planar-distance relation is evaluated only over the planar-distance label domain, while a support relation is evaluated only over the corresponding physical-state labels.

Let $\mathcal{E}^{\text{temp}}_t$ denote the emitted relations with valid temporal annotations. Their temporal-label objective is

$$\mathcal{L}_{\text{reltemp}} = \mathbb{E}_{t,e\in\mathcal{E}^{\text{temp}}_t}\left[\text{CE}(\hat{\delta}_{t,e}, \delta_{t,e})\right]. \tag{49}$$

Relations without an available temporal label are excluded. Node and relation losses are first averaged within each frame so that frames containing more graph elements do not receive disproportionately large weight.

Progress prediction. The progress head predicts a bounded scalar

$$\hat{\Phi}_t = \Phi_\theta(s_t) \in [0, 1]. \tag{50}$$

Its supervision target is obtained by evaluating the fixed schedule on the observed scene graph:

$$\Phi_t = \Phi_{\mathcal{S}}(G_t). \tag{51}$$

The schedules used for ManiSkill and MS-HAB are reported in Appendix D.

We train the progress head using

$$\mathcal{L}_{\text{prog-model}} = \frac{\sum_t m_t \ \text{Huber}_{0.1}\left(\hat{\Phi}_t - \Phi_t\right)}{\max\left(1, \sum_t m_t\right)}, \tag{52}$$

where m_t indicates that the frame is non-terminal and that the relations required to evaluate the schedule target are available. Missing graph facts are masked rather than interpreted as zero progress.

B.4.3 Imagined Returns and Actor–Critic Learning

Imagined trajectories and continuation. The actor and critics are trained from latent imagination trajectories with horizon

$$H = 15. \tag{53}$$

The base discount and return parameter are

$$\gamma = 1 - \frac{1}{333}, \qquad \lambda = 0.95. \tag{54}$$

At imagined transition $s_t \to s_{t+1}$, the continuation head defines a Bernoulli distribution from which

$$\tilde{c}_{t+1} \sim \text{Bernoulli}(\hat{c}_{t+1}) \tag{55}$$

is sampled. The effective continuation discount is

$$\hat{\gamma}_{t+1} = \gamma \, \mathbb{1}\left[\tilde{c}_{t+1} = 1\right]. \tag{56}$$

The same sampled continuation flag is used for both reward streams and for the cumulative trajectory weight

$$\omega_t = \prod_{i=1}^{t} \hat{\gamma}_i, \qquad \omega_0 = 1. \tag{57}$$

The progress reward follows Eq. (9):

$$\hat{r}_{t+1}^{\text{prog}} = \hat{\gamma}_{t+1} \hat{\Phi}_{t+1} - \hat{\Phi}_t. \tag{58}$$

The environment and progress rewards refer to the same transition.

Returns and combined advantage. Let $R_t^{\lambda,\text{env}}$ and $R_t^{\lambda,\text{prog}}$ denote the lambda returns defined in Eq. (69). Each return is bootstrapped with its corresponding critic at the imagination horizon. The two raw advantages are

$$A_t^{\text{env}} = R_t^{\lambda,\text{env}} - v^{\text{env}}(s_t), \tag{59}$$

$$A_t^{\text{prog}} = R_t^{\lambda,\text{prog}} - v^{\text{prog}}(s_t). \tag{60}$$

They are combined before normalization:

$$\widetilde{A}_t = \frac{A_t^{\text{env}} + \beta(n) A_t^{\text{prog}}}{S_R}, \tag{61}$$

where

$$S_R = \max\left(1, P_{95} - P_5\right) \tag{62}$$

is the running percentile scale of the combined return. Using one common scale preserves the relative magnitude of the environment and progress advantages.

The progress coefficient is warmed up according to

$$\beta(n) = \begin{cases} 0, & n \le 2 \times 10^5, \\ \beta_{\max} \dfrac{n - 2 \times 10^5}{5 \times 10^5}, & 2 \times 10^5 < n < 7 \times 10^5, \\ \beta_{\max}, & n \ge 7 \times 10^5, \end{cases} \tag{63}$$

where n is the number of environment steps and $\beta_{\max} = 0.05$ in the reported default configuration. The coefficient is constant across the depths of a single imagination update.

Actor objective. The actor is trained using the normalized combined advantage:

$$\mathcal{L}_{\text{policy}} = -\mathbb{E}_t \left[\omega_t \left(\log \pi(a_t \mid s_t) \, \text{sg}(\widetilde{A}_t) + \eta \, \mathcal{H}[\pi(\cdot \mid s_t)] \right) \right], \tag{64}$$

where

$$\eta = 3 \times 10^{-4}. \tag{65}$$

Critic objectives. For $u \in \{\text{env}, \text{prog}\}$, each critic predicts a symlog-spaced two-hot value distribution:

$$\mathcal{L}_{\text{critic}}^{u} = \mathbb{E}_t \left[\omega_t \left(-\log p_{v^u} \left(\text{sg}(R_t^{\lambda,u}) \mid s_t \right) - \log p_{v^u} \left(\text{sg}(\bar{v}^u(s_t)) \mid s_t \right) \right) \right]. \tag{66}$$

Here,

$$\mathcal{L}_{\text{value}} = \mathcal{L}_{\text{critic}}^{\text{env}}, \qquad \mathcal{L}_{\text{prog-value}} = \mathcal{L}_{\text{critic}}^{\text{prog}}. \tag{67}$$

The slow critics $\bar{v}^u$ are updated after every training step using an interpolation fraction of 0.02. Keeping separate critics prevents the environment-value target from changing as the progress coefficient is warmed up.

Replay value regularization. The environment critic is additionally trained on posterior features from replay:

$$\mathcal{L}_{\text{repval}} = \mathbb{E}_t \left[(1 - \ell_t) \left(-\log p_{v^{\text{env}}} \left(\text{sg}(\widetilde{R}_t^{\lambda}) \mid s_t \right) - \log p_{v^{\text{env}}} \left(\text{sg}(\bar{v}^{\text{env}}(s_t)) \mid s_t \right) \right) \right], \tag{68}$$

where ℓ_t marks the last observation of a sampled replay sequence and $\widetilde{R}_t^{\lambda}$ is the replay-based environment return.

Unlike the imagined value objective, this loss remains attached to the posterior world-model features and therefore supplies value-aware gradients to the learned representation. Its coefficient in Eq. (33) is 0.3.

B.5 Training Procedure and Hyperparameters

Training alternates between environment interaction, replay-based model updates, and actor-critic updates on imagined trajectories. Task-mined graph assets and progress schedules remain fixed. Replay stores the original environment reward; progress rewards are computed from the current potential head during imagination.

Table 8 lists the shared ReL-WM configuration. Benchmark-specific interaction budgets are reported in Appendix C.4.

Table 8: ReL-WM hyperparameters shared across benchmarks. MLP sizes are layers $\times$ units. The progress critic uses the same architecture as the environment critic.

Hyperparameter	Value
Relational representation	
Temporal-change window K	5 steps
Maximum vertices and edges	8 and 168
Message-passing layers	2
Entity and relation-label embedding size	64
Graph hidden dimension	512
Semantic state dimension	512
Graph decoder hidden dimension	256
Graph reconstruction loss weights	1.0
Semantic dynamics and representation weights	1.0, 0.05
World model	
Image resolution	112×112
Deterministic state dimension	4096; 8 GRU blocks
Stochastic latents $\times$ classes	32×32
RSSM hidden dimension	512
CNN channels; kernel size	$(64, 96, 128, 128)$; 5
Encoder and decoder MLPs	3×512
Activation and normalization	SiLU, RMSNorm
Categorical uniform mixture	1%
Free nats	1.0
Reconstruction, reward, continuation weights	1.0
Dynamics and representation weights	1.0, 0.1
Reward and critic outputs	Symexp two-hot, 255 bins
Imagined actor–critic learning	
Imagination horizon	15
Discount γ	$1 - 1/333$
Return parameter λ	0.95
Actor and critic MLPs	3×512
Actor standard deviation range	$[0.1, 1.0]$
Actor entropy coefficient	3×10^{-4}
Return normalization scale	$\max(1, P_{95} - P_5)$
Return normalization decay	0.99
Slow critic decay and regularizer weight	0.98, 1.0
Replay critic loss weight	0.3
Progress prediction and shaping	
Progress head	1×512, sigmoid output
Progress-head loss	Huber, $\delta = 0.1$
Progress-head and critic loss weights	1.0
Default final progress weight	0.05
Progress warmup interval	200k–700k environment steps
Progress weights in ablations	$\{0, 0.05, 0.1, 0.2\}$
Optimization and replay	
Optimizer	LaProp, $\epsilon = 10^{-20}$
Moment decay rates	$(0.9, 0.999)$
Learning rate	4×10^{-5}
Learning-rate warmup	1000 updates
Adaptive gradient clipping	0.3
Batch size $\times$ sequence length	16×64
Replay capacity	7.5×10^5 transitions
Replay ratio	64

C Experimental Setup and Baselines

C.1 Benchmarks and Tasks

ManiSkill. We evaluate on four ManiSkill manipulation tasks (Tao et al., 2025) that require multiple successive interaction stages:

- **PickCube:** grasp a cube and transport it to a target position.
- **PlaceSphere:** grasp a sphere, transport it to a shallow bin, release it, and leave it supported by the bin.
- **PegInsertionSide:** grasp a peg, align it with a horizontally oriented hole, and insert it. The hole configuration is randomized between episodes.
- **PullCubeTool:** grasp an L-shaped tool and use it to pull a distant cube into the robot's reachable region.

For PullCubeTool, the benchmark success condition requires the cube to lie within $0.6\,\mathrm{m}$ planar distance of the robot base. This endpoint condition does not by itself establish that the policy used the tool. We therefore report benchmark success while separately examining whether representative trajectories follow the intended tool-use sequence.

PickCube and PlaceSphere have horizons of 50 control steps, while PegInsertionSide and PullCubeTool have horizons of 100 control steps.

MS-HAB. We use the `NavigateAndPick` task in MS-HAB (Shukla et al., 2025). A Fetch mobile manipulator must navigate toward and grasp a target YCB object (Çalli et al., 2015) in a ReplicaCAD scene (Szot et al., 2021).

ReplicaCAD contains 105 apartment scenes organized into five macro-variations, each containing 21 micro-variations. Macro-variations alter the placement of large furniture, such as refrigerators and kitchen counters, while micro-variations alter smaller furnishings and object arrangements.

Unlike the "magical grasp" mechanism in the original Habitat 2.0 benchmark (Szot et al., 2021), MS-HAB uses physically simulated grasping through low-level robot control. The task therefore requires spatial approach, physical contact, grasp acquisition, and maintenance of the grasp while the end effector returns toward its rest position.

MS-HAB training rollouts use a horizon of 100 control steps. Evaluation allows up to 200 control steps to accommodate variations in navigation distance and scene layout.

C.2 Observations, Control, and Supervision

Visual and low-dimensional observations. All image observations are processed at 112×112 resolution. For ManiSkill, the agent receives RGB observations from the task camera together with the environment-provided low-dimensional robot and task state. Actions use joint-position increments through the `pd_joint_delta_pos` controller. The environment is stepped once for every selected action.

For MS-HAB, the visual observation contains RGB images from the Fetch head and hand cameras. The agent additionally receives the low-dimensional robot state. In multi-object experiments, a language instruction identifies the target object and conditions both the world model and policy.

Rewards and episode boundaries. We use the normalized dense reward mode provided by each benchmark. The returned environment reward is retained without an additional affine transformation and is stored unchanged in replay. The environment-reward head and environment critic are trained on this reward stream.

The progress reward is computed separately during latent imagination. It is not added to the reward stored in replay and is not used to define benchmark success.

The data interface distinguishes true environment termination from time-limit truncation. The continuation target is determined by the terminal indicator, while episode boundaries reset the recurrent state and graph temporal history. Evaluation reports task success rather than the progress potential.

C.3 Baseline Methods

We compare ReL-WM with three model-based reinforcement-learning methods. All methods use the same task definitions, control interface, environment-reward setting, and interaction budgets for the corresponding experiment.

DreamerV3. DreamerV3 (Hafner et al., 2025) learns a recurrent latent world model and trains an actor and critic from imagined trajectories. It provides the direct reference for evaluating the semantic-state, graph-reconstruction, and progress-learning extensions introduced in ReL-WM.

TD-MPC2. TD-MPC2 (Hansen et al., 2024) learns a decoder-free latent dynamics model and uses model predictive control to select actions. It provides a comparison with a planning-based latent model rather than an imagination-trained actor.

SOLD. SOLD (Mosbach et al., 2025) learns object-centric slots from visual observations and predicts their dynamics for policy learning through imagination. It provides the object-centric comparison for evaluating the contribution of explicit entity identities and typed relations.

The ReL-WM settings in Table 8 apply only to our method. The baselines retain their respective representation, dynamics, and action-selection mechanisms.

C.4 Training Budgets and Evaluation

Each ManiSkill task is trained independently for 4M environment steps. Each MS-HAB training setting uses 10M environment steps in total. Object-transfer experiments begin from the multi-object model and use an additional 3M environment steps, as described in Appendix C.5.

During training, actions are sampled from the policy distribution. Evaluation uses the mode of the learned policy and does not perform additional policy adaptation unless the experiment is explicitly identified as transfer learning.

Success rates are reported as percentages. The multi-object, transfer, generalization, and ablation tables report the mean and standard deviation over the experiment repetitions used for those results.

C.5 Multi-Object, Transfer, and Generalization Protocols

C.5.1 Multi-Object Learning and Object Transfer

We train one instruction-conditioned agent on `NavigateAndPick` in the `PrepareGroceries` setting. The five training targets are Master-Chef-Can, Pudding-Box, Sugar-Box, Tuna-Fish-Can, and Bowl.

Training uses 10M environment steps in total, with balanced sampling across the five target objects. At the beginning of each episode, a language instruction specifies the current target. The instruction conditions both the world model and the policy, while the graph associates the target role with the corresponding object instance.

For object transfer, the multi-object policy is fine-tuned for an additional 3M environment steps on held-out target objects in the same scene. The held-out objects are Tomato-Soup-Can, Cracker-Box, and Gelatin-Box.

The transfer setting changes target identity and geometry while retaining the shared interaction structure: approaching the instructed object, establishing a compatible grasp, acquiring the object, and returning the end effector toward its rest position. Transfer results are evaluated after adaptation and are therefore distinct from the zero-shot scene and illumination evaluations below.

C.5.2 Scene Generalization

Scene generalization uses `NavigateAndPick` with `004_sugar_box` as the target. The policy is trained on five ReplicaCAD micro-variations drawn from a single macro-variation.

Evaluation uses 50 unseen micro-variations. Ten belong to the same macro-variation as the training scenes, while the remaining 40 are distributed evenly across the other four macro-variations. This protocol separately tests generalization to new arrangements within a familiar large-scale layout and to arrangements from different large-scale layouts.

All scene-generalization results are zero-shot: the world model and policy receive no gradient updates on the evaluation scenes.

C.5.3 Illumination Generalization

Illumination evaluation keeps the scene configuration and initial physical state fixed while varying only the simulator lighting. The intensities of all ambient, point, and directional lights are scaled by 0.1, 1.0, or 4.0, corresponding to dim, nominal, and bright conditions.

The changes are applied by the simulator during rendering rather than as post-processing transformations of the recorded images. The physical state, task target, and action interface are unchanged across the three conditions.

All illumination results are zero-shot, without world-model or policy adaptation.

C.6 Ablation Settings

All ablations are performed on `PlaceSphere` using the same training budget and evaluation protocol as the corresponding full method.

Progress-weight ablation. We evaluate

$$\beta_{\max} \in \{0, 0.05, 0.1, 0.2\}.$$

All nonzero values use the warmup schedule described in Appendix B.4.3.

Setting $\beta_{\max} = 0$ removes βA_t^{prog} from the actor advantage. The relational world model, semantic prior and posterior, graph reconstruction losses, progress predictor, and progress critic remain present. This setting therefore isolates the effect of using predicted progress to shape the policy update while retaining progress-related representation learning.

Progress-head ablation. The "without progress head" variant removes the progress-prediction objective, progress shaping reward, progress critic, and progress advantage. It retains the graph encoder, semantic state, semantic dynamics objectives, and graph-reconstruction objectives. This comparison measures whether explicit progress prediction provides an additional benefit beyond the relational world model.

DreamerV3 reference. The DreamerV3 reference removes the graph-conditioned semantic state, its learned prior, graph reconstruction, semantic alignment, progress prediction, progress critic, and progress advantage. The remaining agent follows the standard visual DreamerV3 world-model and imagination-based actor–critic structure.

The main text reports success rates for these variants. Appendix A.5 additionally examines the relative magnitude of the progress and environment advantages for the progress-weight ablation.

D Task-Specific Progress Schedules

D.1 Shared Scoring Conventions

The following tables specify semantic roles, ordered phases, milestone conditions, and additive weights. Each task has a total phase budget of one.

For planar distance, the credited order is

`far`, `medium`, `near`, `very-near`.

For compatibility, it is

`poor-match`, `partial-match`, `match`.

"Or better" includes the named state and subsequent states in the corresponding order. Thus, satisfying a stronger threshold also satisfies weaker credited thresholds. The `very-far` and `unobserved` states receive no credit from these threshold ladders.

To avoid repeating identical rows, some table entries list a sequence of thresholds and a matching sequence of weights. Each threshold is a separate milestone, and its weight is added when that threshold is satisfied.

Table 9: Initial approach phase shared by the four ManiSkill schedules. Every listed threshold contributes its corresponding additive weight.

Milestone	Credited thresholds	Additive weights
EE–object planar distance	`far` or better; `medium` or better; `near` or better; `very-near`	0.0375 each
EE–object grasp compatibility	`poor-match` or better; `partial-match` or better; `match`	0.033333, 0.033333, 0.033334

Table 10: PickCube schedule after the shared approach phase (0.25; Table 9).

Milestone	Credited states	Additive weights
Grasp (0.15)		
EE grasps cube	`holds`	0.15
Carry to goal (0.60)		
Cube–goal planar distance	`far` or better; `medium` or better; `near` or better	0.075 each
Cube–goal height offset	{`far-below`, `below`, `level`}; {`below`, `level`}; `level`	0.066667, 0.066667, 0.066666
Cube reaches goal	`reached`	0.175

Directional physical milestones are written semantically, such as "bin supports sphere." Their implementation uses the appropriate `src-holds` or `dst-holds` state for the stored endpoint ordering. Task-specific `reached` predicates denote environment goal conditions rather than additional affordance classes.

D.2 ManiSkill Schedules

D.2.1 Shared Initial Approach Phase

All four ManiSkill schedules begin with a phase of weight 0.25 that approaches the object to be grasped: the cube, sphere, peg, or tool. Contact with that object completes the phase.

D.2.2 PickCube-v1

The semantic roles are `movable = cube` and `destination = goal_site`. The schedule consists of approach, grasp, and carry-to-goal phases. Table 10 specifies the task-specific phases after the shared approach phase.

Grasp acquisition completes the grasp phase. Reaching the goal site completes the carry phase. There is no separate `very-near` milestone for cube–goal distance because the environment's `reached` predicate supplies the final position criterion.

D.2.3 PlaceSphere-v1

The semantic roles are `movable = sphere` and `destination = bin`. The shared approach phase is followed by grasp, transport, and settle phases.

Grasp acquisition completes the grasp phase. Sphere–bin contact completes the transport phase. Support of the sphere by the bin completes the settle phase.

D.2.4 PegInsertionSide-v1

The semantic roles are `movable = peg`, `destination = box-with-hole`, and `site = hole-mouth site`. The hole-mouth site is updated from the randomized hole geometry in each episode.

The reach-hole-mouth phase completes when the peg head reaches the hole mouth and the peg contacts the box-with-hole. These two conditions also gate containment-alignment credit. The alignment

Table 11: PlaceSphere schedule after the shared approach phase (0.25; Table 9).

Milestone	Credited states	Additive weights
Grasp (0.10)		
EE grasps sphere	`holds`	0.10
Transport (0.30)		
Sphere–bin planar distance	`far` or better; `medium` or better; `near` or better; `very-near`	0.05 each
Sphere–bin support compatibility	`poor-match` or better; `partial-match` or better; `match`	0.033333, 0.033333, 0.033334
Settle (0.35)		
Bin supports sphere	Physical support holds	0.35

Table 12: PegInsertionSide schedule after the shared approach phase (0.25; Table 9).

Milestone	Credited states	Additive weights
Grasp (0.10)		
EE grasps peg	`holds`	0.10
Reach hole mouth (0.15)		
Peg-head–site planar distance	`far` or better; `medium` or better; `near` or better; `very-near`	0.015 each
Peg-head–site height offset	{`below`, `level`, `above`}; `level`	0.03 each
Peg head reaches hole mouth	`reached`	0.03
Containment alignment (0.15)		
Peg–hole containment compatibility	`poor-match` or better; `partial-match` or better; `match`	0.05 each
Contain (0.35)		
Box-with-hole contains peg	Physical containment holds	0.35

phase completes when containment compatibility is `match`; physical containment completes the final phase.

The gate prevents alignment credit before the peg reaches and contacts the hole. Gate and completion conditions are evaluated from the current frame and are not stored as temporal latches.

D.2.5 PULLCUBETOOL-V1

The semantic roles are `movable = L-shaped tool`, `destination = cube`, and `reference = pull-goal region`. The shared approach phase targets the tool.

Grasping the tool completes the grasp phase. Tool–cube contact completes the reach-cube phase. The pull phase completes when the cube reaches the pull-goal region, defined by a planar distance below $0.6\,\mathrm{m}$ from the robot base.

Height-offset milestones are omitted because the cube remains on the table. Tool–cube support compatibility is omitted because a reliable calibration is unavailable.

D.3 MS-HAB PICK SCHEDULE

MS-HAB picking uses three ordered phases: approach, acquire, and return to rest. The `object` role denotes the instructed target, and `rest` denotes the end-effector rest site.

Grasp acquisition completes both the approach and acquire phases. The return phase requires the target to remain grasped and rewards motion toward the rest site. It completes when the end effector reaches that site while maintaining the grasp. Contact is not scored separately. Phase completion and prerequisites are evaluated from the current graph.

Table 13: PullCubeTool schedule after the shared approach phase (0.25; Table 9).

Milestone	Credited states	Additive weights
Grasp tool (0.10)		
EE grasps tool	`holds`	0.10
Reach cube (0.50)		
Tool–cube planar distance	`far` or better; `medium` or better; `near` or better; `very-near`	0.08 each
Tool–cube contact compatibility	`poor-match` or better; `partial-match` or better; `match`	0.06 each
Pull (0.15)		
Cube–pull-region planar distance	`far` or better; `medium` or better; `near` or better; `very-near`	0.0225 each
Cube reaches pull-goal region	`reached`	0.06

Table 14: MS-HAB picking schedule. Listed thresholds contribute additive weights. The three phase budgets sum to one.

Milestone	Credited states	Additive weights
Approach (0.35)		
EE–object planar distance	`far` or better; `medium` or better; `near` or better; `very-near`	0.0375 each
EE–object height offset	{`below`, `level`, `above`}; `level`	0.04 each
EE–object grasp compatibility	`poor-match` or better; `partial-match` or better; `match`	0.03, 0.04, 0.05
Acquire (0.20)		
EE grasps object	`holds`	0.20
Return to rest (0.45)		
EE maintains object grasp	`holds`	0.10
EE–rest planar distance	`far` or better; `medium` or better; `near` or better	0.04 each
EE–rest height offset	{`below`, `level`, `above`}	0.06
EE reaches rest site	`holds`	0.17

E Analysis of progress-based shaping

We analyze the imagined MDP $\hat{\mathcal{M}}$ on model states $s = [z, g, h]$. Write $\hat{\Phi}_t = \Phi_\theta(s_t) \in [0, 1]$. The transition, reward, continuation, and potential predictors, and the scalar weight β, are fixed for the policy-invariance statement. We assume bounded environment rewards and $0 \leq \hat{\gamma}_{t+1} \leq \gamma < 1$. Both reward streams use the same imagined trajectory, discounts, and λ. The implementation samples binary continuation flags; the analysis also applies to soft continuation discounts. The imagination horizon H is a computational cutoff, not a terminal state.

Return targets and linearity. For $u \in \{\mathrm{env}, \mathrm{prog}\}$, the horizon-$H$ λ-return is

$$\begin{aligned} R_t^{\lambda,u} &= \hat{r}_{t+1}^u + \hat{\gamma}_{t+1}\big[(1-\lambda)v^u(s_{t+1}) + \lambda R_{t+1}^{\lambda,u}\big], \\ R_H^{\lambda,u} &= v^u(s_H), \end{aligned} \tag{69}$$

where $0 \leq \lambda \leq 1$. Define the raw advantage by $A_t^u = R_t^{\lambda,u} - v^u(s_t)$.

For a fixed β, linearity gives

$$R_t^{\lambda,\mathrm{env}} + \beta R_t^{\lambda,\mathrm{prog}} = R_t^{\lambda,\mathrm{comb}},$$

where the combined return uses the reward $\hat{r}^{\mathrm{env}} + \beta\hat{r}^{\mathrm{prog}}$ and bootstrap $v^{\mathrm{env}} + \beta v^{\mathrm{prog}}$. The identity follows by backward induction: it holds at H, and both component recursions use the same discount and λ coefficients.

Lemma 1 (Telescoping shaping return)**.** *Let* $\Gamma_{t,k} = \prod_{j=t+1}^{k} \hat{\gamma}_j$*, with* $\Gamma_{t,t} = 1$*. For the shaping reward in Eq.* (9)*,*

$$\sum_{k=t}^{H-1} \Gamma_{t,k}\hat{r}_{k+1}^{\mathrm{prog}} = -\hat{\Phi}_t + \Gamma_{t,H}\hat{\Phi}_H. \tag{70}$$

Proof. Substituting $\hat{r}_{k+1}^{\mathrm{prog}} = \hat{\gamma}_{k+1}\hat{\Phi}_{k+1} - \hat{\Phi}_k$ and using $\Gamma_{t,k}\hat{\gamma}_{k+1} = \Gamma_{t,k+1}$ gives

$$\sum_{k=t}^{H-1} \big(\Gamma_{t,k+1}\hat{\Phi}_{k+1} - \Gamma_{t,k}\hat{\Phi}_k\big).$$

Adjacent terms cancel, leaving the two boundary terms. □

Proof of Proposition 1.

Proof. At an imagined termination, the continuation discount is zero, so the boundary term in Eq. (70) vanishes. On continuing trajectories,

$$0 \leq \Gamma_{t,H}\hat{\Phi}_H \leq \gamma^{H-t} \longrightarrow 0.$$

Thus the full discounted shaping return from a decision state $s_t = s$ equals $-\hat{\Phi}(s)$, independently of subsequent actions. Taking expectations under any policy, including conditioning on the first action, gives

$$V_{\mathrm{prog}}^\pi(s) = Q_{\mathrm{prog}}^\pi(s, a) = -\hat{\Phi}(s).$$

Consequently,

$$Q_{\mathrm{env}+\beta\mathrm{prog}}^\pi(s, a) = Q_{\mathrm{env}}^\pi(s, a) - \beta\hat{\Phi}(s).$$

The offset depends only on the starting state and is identical across actions and policies, preserving the optimal policies of the fixed imagined model.

This is the potential-based shaping argument of Ng et al. (1999), applied to the imagined MDP. Almost-sure finite termination is not required. A zero terminal discount removes the terminal potential contribution without requiring the predictor to output zero at terminal states (Grześ, 2017). □

Equation (70) concerns the discounted reward sum. For the bootstrapped return at $\lambda = 1$, it implies

$$R_t^{1,\mathrm{prog}} = -\hat{\Phi}_t + \Gamma_{t,H}\big(\hat{\Phi}_H + v^{\mathrm{prog}}(s_H)\big).$$

The following lemma characterizes the advantage for every $\lambda \in [0, 1]$.

Lemma 2 (Finite-horizon progress advantage)**.** *Define the critic residual* $e_k = v^{\mathrm{prog}}(s_k) + \hat{\Phi}_k$ *for* $t \le k \le H$*. For the returns in Eq.* (69)*,*

$$\begin{aligned} A_t^{\mathrm{prog}} &= \sum_{k=t}^{H-1} \lambda^{k-t} \Gamma_{t,k} \big(\hat{\gamma}_{k+1} e_{k+1} - e_k\big), \\ |A_t^{\mathrm{prog}}| &\le 2 \max_{t \le k \le H} |e_k|. \end{aligned} \tag{71}$$

At $\lambda = 0$*, only the first summand remains. In particular, if* $e_k = 0$ *throughout the rollout, then* $A_t^{\mathrm{prog}} = 0$ *for every horizon and every* λ*.*

Proof. Set $B_t = R_t^{\lambda,\mathrm{prog}} + \hat{\Phi}_t$. Substituting Eq. (9) into the return recursion yields

$$B_t = \hat{\gamma}_{t+1}\big[(1-\lambda) e_{t+1} + \lambda B_{t+1}\big], \qquad B_H = e_H.$$

Since $A_t^{\mathrm{prog}} = B_t - e_t$, we obtain

$$A_t^{\mathrm{prog}} = \hat{\gamma}_{t+1} e_{t+1} - e_t + \lambda \hat{\gamma}_{t+1} A_{t+1}^{\mathrm{prog}}, \qquad A_H^{\mathrm{prog}} = 0.$$

Unrolling this recursion proves the identity.

For the bound, let $E = \max_{t \le j \le H} |e_j|$. The terminal condition gives $|B_H| \le E$. If $|B_{k+1}| \le E$, then

$$|B_k| \le \hat{\gamma}_{k+1}\big[(1-\lambda)|e_{k+1}| + \lambda |B_{k+1}|\big] \le E.$$

Backward induction therefore gives $|B_k| \le E$ throughout the rollout. Hence $|A_t^{\mathrm{prog}}| \le |B_t| + |e_t| \le 2E$. □

The lemma characterizes the raw progress advantage; it does not establish that this additional signal improves the environment objective. Its interpretation is consistent with the connection between potential-based shaping and value initialization (Wiewiora, 2003), without asserting equivalence of the full training algorithm to value initialization.

Scope under changing models. Policy invariance does not require the potential to accurately measure task progress; its usefulness during approximate learning is an empirical question. As the model, potential predictor, and shaping coefficient change, the argument applies separately to each fixed snapshot. It does not establish convergence of the changing learning process or preservation of optimal policies in the real environment.

Raw advantages and normalized actor updates. With an exact progress critic, $R_t^{\lambda,\mathrm{prog}} = -\hat{\Phi}_t$ and $A_t^{\mathrm{prog}} = 0$ for every imagination horizon and every λ. Values at terminal states do not affect these returns because their incoming discount is zero. The combined return nevertheless equals

$$R_t^{\lambda,\mathrm{env}} + \beta R_t^{\lambda,\mathrm{prog}} = R_t^{\lambda,\mathrm{env}} - \beta \hat{\Phi}_t.$$

The implementation computes S_R from running percentiles of the combined return: $S_R = \max(1, P_{95} - P_5)$. Because $\hat{\Phi}_t$ varies across states, this scale can change even when the raw progress advantage is zero. In Eq. (10), the entropy bonus is added outside the advantage normalization, so its relative strength can change as well. Vanishing raw progress advantage therefore does not imply identical normalized actor updates. A fixed entropy regularizer shared by two exact objectives would retain the same policy-independent shaping offset.