World Translation: Appendix

Author Names Omitted for Anonymous Review. Paper-ID 350

I. Further Discussion

A. Information Preservation Assumption

The backward dynamics extraction problem assumes that all *relevant* dynamics information not contained in $(\mathbf{o}_t, \mathbf{a}_t)$ is preserved in the transition $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$. That is, the transition retains all information about hidden variables $\mathbf{h}_t$ that affect state evolution.

This assumption is more likely to hold under certain conditions. First, preservation is easier when observations and actions are high-dimensional relative to $\mathbf{h}_t$: the state transition provides sufficient "capacity" to encode low-dimensional hidden variable effects. Second, discrete hidden variables (e.g., collision events, contact states) create distinct signatures in transitions that are easier to detect than continuous ones (e.g., external force magnitudes), which may have subtler effects.

For robotics applications, these conditions are typically satisfied: observations include multi-joint positions and velocities, hidden variables are often discrete contact events or low-dimensional forces. Empirically, Table I in the main paper supports this: on the R5 manipulator task where $\mathbf{h}_t$ (external forces) and $\mathbf{c}$ (motor parameters) are tightly coupled, the full transition $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$ achieves higher R^2 (0.73–0.78) for predicting $\mathbf{h}_t$ than $(\mathbf{o}_t, \mathbf{a}_t)$ alone (0.66–0.73), confirming that the transition outcome preserves additional $\mathbf{h}_t$ information.

B. Design Tradeoffs

World Translation makes deliberate design choices that trade certain capabilities for others. Understanding these tradeoffs clarifies when the method is most appropriate.

Unpaired learning versus direct supervision. Unlike conventional dynamics models that minimize prediction error on paired samples, World Translation optimizes reconstruction, adversarial, and cycle consistency losses on unpaired data. This indirect optimization may sacrifice prediction accuracy compared to models trained with direct supervision. However, paired sim-real data with matching hidden variables is often impossible to obtain, since physical interactions cannot be precisely reproduced. The tradeoff enables learning from data that would otherwise be unusable, and the cycle consistency constraint encourages semantically meaningful mappings rather than superficial pattern matching.

Implicit representation versus explicit parameterization. The learned dynamics feature $\mathbf{z}_t$ captures effects of $\mathbf{h}_t$ and $\mathbf{c}$ without explicitly separating them. This implicit representation can capture phenomena beyond predefined parameter spaces, including unmodeled dynamics that system identification cannot reach. The tradeoff is reduced interpretability: when training becomes unstable or translation fails, diagnosing root causes is difficult without explicit structure.

These two tradeoffs combine to create practical difficulties. Unpaired adversarial training is inherently less stable than supervised learning, requiring careful hyperparameter tuning. Meanwhile, unlike image translation where outputs can be visually inspected, latent dynamics translation lacks intuitive evaluation criteria. We cannot directly observe whether $\mathbf{z}_t$ captures the intended semantics or whether translation preserves them, and must rely on indirect metrics like downstream prediction error or proxy tasks such as domain classification.

C. Challenges in Practice

Several practical challenges affect World Translation's effectiveness, as analyzed in our experiments.

Trajectory distribution mismatch. Different policies or domain-induced behavioral differences create distribution mismatch between domains. As shown in Section V-B, $(\mathbf{o}_t, \mathbf{a}_t)$ alone achieves 79–85% domain classification accuracy, indicating substantial distribution divergence. However, this information leakage has a beneficial side effect: $\mathbf{h}_t$ information in $(\mathbf{o}_t, \mathbf{a}_t)$ is preserved through translation. Our auxiliary components ensure $\mathbf{z}_t$ carries sufficient domain information for translation to succeed despite this leakage.

Coupling between $\mathbf{h}_t$ and $\mathbf{c}$. Translation is harder when hidden variables interact with domain characteristics. As demonstrated in the R5 experiments, where external forces couple with domain-specific motor gains, the method still improves over baselines but with smaller margins than locomotion tasks where contact events are naturally separable from motor parameters.

State overwriting. Our alignment procedure overwrites observable states while leaving hidden simulator states (e.g., internal contact buffers) unchanged. This can cause inconsistent physics decisions, though we find the effect minor when most relevant state is observable. Residual action methods [1, 2, 3] avoid this issue by modifying actions rather than states, keeping simulator dynamics consistent. However, action correction introduces its own constraint: the action space is typically lower-dimensional than the state space, limiting what dynamics differences can be corrected through actions alone.

D. Scope and Boundaries

Some constraints are fundamental to what World Translation can achieve, defining the method's scope rather than representing failures.

Unmodeled phenomena. World Translation translates dynamics features extracted from source domain transitions. If a phenomenon exists in the target domain but is entirely absent from the source, such as a payload present in reality but not simulated, no amount of translation can recover it.

The source domain must contain the relevant physics, even if with incorrect parameters. This motivates ensuring simulators include all relevant phenomena, even approximately, before applying translation.

Indirect hidden variable influence. World Translation cannot directly control how hidden variables $\mathbf{h}_t$ evolve. It can only transform the dynamics features that result from those hidden variables. The underlying hidden variable dynamics remain governed by the simulator. This is appropriate when the simulator correctly models *when* events occur (e.g., contact timing) but incorrectly models *how* they affect dynamics (e.g., contact forces). For applications requiring control over hidden variable evolution itself, explicit physics modeling remains necessary.

E. Future Directions

Several extensions could broaden World Translation's applicability.

Multi-step and autoregressive training. Our current single-step formulation accumulates errors over long rollouts. Training on short trajectory segments rather than individual transitions, or incorporating autoregressive objectives, could improve multi-step accuracy. Model-predictive control that replans frequently provides another mitigation strategy at deployment time.

Visual observations. Incorporating images would enable dynamics translation for vision-based policies. The challenge is disentangling visual appearance from dynamics, since visual translation changes appearance without necessarily preserving dynamics-relevant content. A hierarchical approach that separately translates visual features and dynamics features may be effective.

Online adaptation. Updating the translation model during deployment could address distribution shift without offline retraining. The unpaired formulation is well-suited to online updates since new real-world transitions can be incorporated without requiring simulated counterparts.

Combination with action correction. World Translation provides target states $\mathbf{o}_{t+1}^R$ through dynamics translation; action correction methods learn to achieve target states through modified actions. This combination would address the state overwriting challenge: rather than directly overwriting simulator states, action correction finds actions that naturally drive the simulator toward translated targets, keeping internal simulator state consistent. This leverages World Translation's ability to handle unpredictable hidden variables while avoiding artifacts from state manipulation.

II. Implementation Details

A. Derivative-Based Network Architecture

For improved numerical stability and smoother predictions, our implementation operates on observation derivatives rather than raw observations.

Encoder Input. Instead of directly processing $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$, the encoder computes and normalizes the observation derivative:

$$\dot{\mathbf{o}}_t = \frac{\mathbf{o}_{t+1} - \mathbf{o}_t}{\Delta t} \tag{1}$$

The encoder then operates on $(\mathbf{o}_t, \mathbf{a}_t, \dot{\mathbf{o}}_t)$, allowing the network to focus on the rate of change rather than absolute positions.

Decoder Output. The decoder predicts an observation derivative which is integrated to produce the next state:

$$\hat{\mathbf{o}}_{t+1} = \mathbf{o}_t + D_\theta(\mathbf{o}_t, \mathbf{a}_t, \mathbf{z}) \cdot \Delta t \tag{2}$$

Similarly, the blind decoder outputs:

$$\tilde{\mathbf{o}}_{t+1} = \mathbf{o}_t + D_\psi(\mathbf{z}) \cdot \Delta t \tag{3}$$

This formulation ensures that predictions are anchored to the current state, improving training stability and producing physically plausible trajectories.

B. State Overwriting for Alignment

During World Translation Alignment, the aligned observation $\mathbf{o}_{t+1}^{\text{aligned}}$ must be written back to the simulator as physical state. This involves:

Observation Decomposition. The observation vector is parsed into physical components:

- Base angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ (body frame)
- Projected gravity $\mathbf{g}_{\text{proj}} \in \mathbb{R}^3$ (body frame)
- Joint positions $\mathbf{q} \in \mathbb{R}^n$ (relative to default)
- Joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$
- Base linear velocity $\mathbf{v} \in \mathbb{R}^3$ (body frame)

Coordinate Frame Conversion. Body-frame quantities are transformed to world frame for the simulator API:

$$\boldsymbol{\omega}^{\text{world}} = \mathbf{R}(\mathbf{q}_{\text{ori}})\boldsymbol{\omega}^{\text{body}} \tag{4}$$

$$\mathbf{v}^{\text{world}} = \mathbf{R}(\mathbf{q}_{\text{ori}}^{\text{new}})\mathbf{v}^{\text{body}} \tag{5}$$

where $\mathbf{R}(\cdot)$ denotes the rotation matrix corresponding to a quaternion.

Orientation Recovery. The new orientation quaternion is derived from the projected gravity vector:

$$\text{roll} = \arctan 2(g_y, -g_z) \tag{6}$$

$$\text{pitch} = \arctan 2(g_x, \sqrt{g_y^2 + g_z^2}) \tag{7}$$

The yaw angle is preserved from the current orientation, as the projected gravity only constrains roll and pitch.

Joint Position Conversion. Relative joint positions are converted to absolute:

$$\mathbf{q}^{\text{abs}} = \mathbf{q}^{\text{rel}} + \mathbf{q}^{\text{default}} \tag{8}$$

C. Augmented Simulation with World Translation

We augment the simulation environment by wrapping it with a translation layer that intercepts each step and aligns the simulator state to the target domain. Algorithm 1 describes this procedure.

Algorithm 1: Augmented Simulation with World Translation

Input: Simulator environment, trained encoder E_ϕ, decoder D_θ, generator $G_{S\to R}$

for *each simulation step with action* $\mathbf{a}_t$ **do**

Step simulator with $\mathbf{a}_t$ to obtain $\mathbf{o}^S_{t+1}$;

`// Translate dynamics`

$\mathbf{z}^S \leftarrow E_\phi(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}^S_{t+1})$;

$\mathbf{z}^{S\to R} \leftarrow G_{S\to R}(\mathbf{z}^S)$;

$\mathbf{o}^{S\to R}_{t+1} \leftarrow D_\theta(\mathbf{o}_t, \mathbf{a}_t, \mathbf{z}^{S\to R})$;

`// Align simulator state`

`OverwriteState`(simulator, $\mathbf{o}^{S\to R}_{t+1}$);

$\mathbf{o}_{t+1} \leftarrow$ `ComputeObservations`(simulator);

return $\mathbf{o}_{t+1}$;

D. Network Architectures

All networks are MLPs with ReLU activation. Table I lists the hidden layer dimensions. The latent dimension is 16. The FiLM [4] parameter generator uses hidden dimension 64. The CycleGAN [5] generators use residual connections: $G(\mathbf{z}) = \text{MLP}(\mathbf{z}) + \mathbf{z}$.

TABLE I: Network architectures (hidden layer dimensions).

Component	Hidden Dims
Backward Dynamics Extraction (VAE)	
Encoder E_ϕ	[128, 256, 256, 128]
Decoder D_θ	[128, 256, 256, 128]
Blind Decoder D_ψ	[128, 256, 256, 128]
Domain Classifier C_ξ	[64, 64, 16]
Unpaired Domain Translation (CycleGAN)	
Generator G	[128, 256, 256, 128]
Discriminator D	[256, 128, 64, 32]

E. Training Hyperparameters

Table II lists the training hyperparameters. All networks use the Adam optimizer. The VAE and CycleGAN are trained jointly with a batch size of 16384.

TABLE II: Training hyperparameters.

Hyperparameter	Value
Backward Dynamics Extraction (VAE)	
Learning rate	0.001
KL weight β	0.003
Blind loss weight λ_b	0.3
Blind loss temperature σ_b	0.3
Domain classifier weight λ_c	0.1
Unpaired Domain Translation (CycleGAN)	
Generator learning rate	0.0005
Discriminator learning rate	0.0001
Cycle consistency weight λ_{cyc}	2.0
Identity weight λ_{id}	0.1

F. Latent Space Visualization

We use UMAP [6] rather than t-SNE [7] for latent space visualization because UMAP supports out-of-sample projection. Specifically, we fit the UMAP reducer on original latents from both domains $[\mathbf{z}^S, \mathbf{z}^R]$, then project translated latents $\mathbf{z}^{S\to R}$ into the same embedding space using the fitted transform. This ensures fair comparison: the translated latents are mapped using the same learned projection as the original data.

In contrast, t-SNE computes embeddings from scratch for each dataset. To visualize translated latents with t-SNE, one would need to recompute the entire embedding including the translated data, which changes the coordinate system and invalidates direct visual comparison between before and after translation.

Table III lists the UMAP hyperparameters.

TABLE III: UMAP hyperparameters.

Hyperparameter	Value
`n_neighbors`	15
`min_dist`	0.1
`random_state`	42

III. Experiment Details

A. Dynamics Modeling Accuracy

We evaluate dynamics modeling accuracy using a dual-environment teacher-forcing protocol that isolates prediction error from policy behavior.

The evaluation uses a dual-environment setup where $2N$ environments run in parallel: N source-domain environments (indices $[0, N)$) with modified physics parameters, and N target-domain environments (indices $[N, 2N)$) with target physics parameters. At each evaluation step:

1) State synchronization: The full robot state (root pose, root velocity, joint positions, joint velocities) is copied from target-domain environments to their paired source-domain environments, ensuring both domains start from identical physical states $\mathbf{o}_t$ and $\mathbf{h}_t$.
2) Action computation: The policy computes actions $\mathbf{a}_t$ using target-domain observations, ensuring all methods uses identical actions.
3) Parallel simulation: Both domains step forward using the same $(\mathbf{o}_t, \mathbf{a}_t)$ and $\mathbf{h}_t$. The source domain produces $\mathbf{o}^S_{t+1}$ (raw simulation output), while the target domain produces $\mathbf{o}^R_{t+1}$ (ground truth).
4) Prediction: Each method generates its prediction $\mathbf{o}^{S\to R}_{t+1}$ from the available information. For World Translation: $\mathbf{o}^{S\to R}_{t+1} = \text{Decode}(\mathbf{o}_t, \mathbf{a}_t, G_{S\to R}(\text{Encode}(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}^S_{t+1})))$.
5) Error computation: Prediction error is computed as $\|\mathbf{o}^{S\to R}_{t+1} - \mathbf{o}^R_{t+1}\|^2$ after normalizing both tensors using an empirical normalizer (running mean and standard deviation) fitted on the low-h_t task data.

This teacher-forcing protocol ensures that prediction errors do not compound across timesteps, as each method receives ground-truth states at every step. This isolates dynamics modeling accuracy from the confounding effects of error accumulation, enabling fair comparison across methods with different error magnitudes.



(a) G1 Humanoid (b) Go2 Quadruped (c) R5 Manipulator

Fig. 1: Robot platforms used in our experiments. Each robot is evaluated under two task conditions: low-h_t (left in each panel) and high-h_t (right in each panel). (a) G1 humanoid: velocity tracking on flat terrain vs. irregular stepping blocks. (b) Go2 quadruped: velocity tracking without payload vs. with ball-in-box payload. (c) R5 manipulator: end-effector tracking in free space vs. under external forces.

1) G1 Humanoid: We evaluate on the Unitree G1 humanoid (29-DOF) with two task conditions that vary the influence of hidden variables h_t on dynamics. The dynamics observation $\mathbf{o} \in \mathbb{R}^{67}$ includes base angular velocity, projected gravity, joint positions, joint velocities, and base linear velocity.

For the low-h_t condition, the robot performs velocity tracking on flat terrain. Hidden variables consist primarily of foot contact events, which follow predictable patterns during steady-state bipedal walking.

For the high-h_t condition, the robot walks on irregular stepping blocks (0.5m $\times$ 0.5m grid) with randomized heights ranging from 0.0 to 0.30m. The uneven terrain creates unpredictable foot contact timing and forces that cannot be inferred from observation history, as the exact terrain geometry under each foot is unobservable.

To simulate domain gap, source and target domains use different physics parameters. The source domain applies the following ratios relative to the target domain:

- Actuator stiffness: $3.0\times$
- Actuator damping: $1.5\times$
- Static and dynamic friction: $1.5\times$
- Total mass: $0.8\times$

2) Go2 Quadruped: We evaluate on the Unitree Go2 quadruped (12-DOF) with two task conditions that vary the influence of hidden variables h_t on dynamics. The dynamics observation $\mathbf{o} \in \mathbb{R}^{33}$ includes base angular velocity, projected gravity, joint positions, joint velocities, and base linear velocity.

For the low-h_t condition, the robot performs velocity tracking on flat terrain without payload. Hidden variables consist primarily of ground contact events, which are relatively predictable during steady-state walking.

For the high-h_t condition, a 5 kg ball is placed inside a box mounted on the robot's back. The ball moves freely with low friction (static: 0.1, dynamic: 0.05), creating unpredictable interaction forces that significantly affect the robot's dynamics. These payload-induced disturbances represent hidden variables that cannot be inferred from observation history, as the ball's motion inside the box has no observable precursor.

To simulate domain gap, source and target domains use different physics parameters. The source domain applies the following ratios relative to the target domain:

- Actuator stiffness: $3.0\times$
- Actuator damping: $1.5\times$
- Static and dynamic friction: $1.5\times$
- Total mass: $0.8\times$

3) R5 Manipulator: We evaluate on the ARX R5 manipulator (6-DOF, fixed-base) with two task conditions that vary the influence of hidden variables h_t on dynamics. The dynamics observation $\mathbf{o} \in \mathbb{R}^{12}$ includes joint positions and joint velocities.

For the low-h_t condition, the robot performs end-effector pose tracking in free space. Hidden variables are minimal as the manipulator operates without external interactions.

For the high-h_t condition, random external forces ranging from 0 to 200N are applied to the end-effector. The force magnitude and direction are resampled every 1.5–3.0 seconds, creating unpredictable disturbances that significantly affect joint dynamics. These external forces represent hidden variables that cannot be inferred from observation history, as the robot has no force sensing in its observation space.

To simulate domain gap, source and target domains use different physics parameters. The source domain applies the following ratios relative to the target domain:

- Actuator stiffness: $0.5\times$
- Actuator damping: $1.3\times$
- Total mass: $0.7\times$

The source domain uses reduced actuator stiffness to amplify the effect of external forces on joint dynamics, creating a more challenging scenario where hidden variables and domain characteristics are tightly coupled.

B. Representation Analysis

The representation analysis in the main paper uses the following implementation details. All probes (classifiers and decoders) are trained online during simulation rollouts with 256 parallel environments, 100 warmup steps, and 5000 evaluation steps.

For domain classification, we use 3-layer MLPs: input $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 1 for $\mathbf{z}$ classifiers, and input $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 1 for transition-based classifiers. BCE loss is used with sigmoid output.

For hidden variable (h_t) prediction, we use 3-layer MLPs: input $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 3 for $\mathbf{z}$ decoders, and input $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 3 for transition-based decoders. Forces are normalized by the maximum magnitude (200N) before computing MSE loss.

C. Downstream Policy

The base policy uses an MLP with hidden dimensions [512, 256, 128] and ELU activation, trained with PPO using 4096 parallel environments, 24 steps per environment per iteration, and learning rate 10^{-3} with adaptive schedule.

Transition data for both simulation and real-world domains is collected by deploying the base policy while tracking velocity commands via joystick, ensuring similar trajectory distributions across domains. Data is recorded at the control frequency of 50 Hz.

For the real-world Go2 experiment, we collect approximately 20 minutes of transition data, yielding roughly 60,000 single-step transitions $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$. The ball-in-box payload is mounted on the robot during the entire collection procedure, matching the target domain condition. We use joystick-controlled velocity commands and intentionally vary the commanded linear and angular velocities to ensure coverage across the operational range. No noise filtering or preprocessing is applied to the proprioceptive observations; at 50 Hz, the Go2's onboard sensors provide sufficiently clean readings, and we empirically observe that the translation model trains stably on raw sensor data.

Note that the real-world observation excludes base linear velocity, as accurate linear velocity estimation is unavailable on the Go2 without external motion capture. In contrast, our simulation experiments (Section III-A) include base linear velocity in the observation to minimize its involvement in $\mathbf{h}_t$.

REFERENCES

[1] J. Hanna and P. Stone, "Grounded action transformation for robot learning in simulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[2] H. Karnan, S. Desai, J. P. Hanna, G. Warnell, and P. Stone, "Reinforced grounded action transformation for sim-to-real transfer," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[3] T. He, J. Luo, W. He, T. Xiao, Y. Zhang, Z. Shi, S. Fidler, Y. Zhu, and L. Liu, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," in *Robotics: Science and Systems (RSS)*, 2025.

[4] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[5] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[6] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[7] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.