Visual Imitation Learning of Non-Prehensile Manipulation Tasks with Dynamics-Supervised Models

Abdullah Mustafa[1], Ryo Hanai[1], Ixchel G. Ramirez-Alpizar[1], Floris Erich[1], Ryoichi Nakajo[1], Yukiyasu Domae[1] and Tetsuya Ogata[2]

***Abstract*— Unlike quasi-static robotic manipulation tasks like pick-and-place, dynamic tasks such as non-prehensile manipulation pose greater challenges, especially for vision-based control. Successful control requires the extraction of features relevant to the target task. In visual imitation learning settings, these features can be learnt by backpropagating the policy loss through the vision backbone. Yet, this approach tends to learn task-specific features with limited generalizability. Alternatively, learning world models can realize more generalizable vision backbones. Utilizing the learnt features, task-specific policies are subsequently trained. Commonly, these models are trained solely to predict the next RGB state from the current state and action taken. But only-RGB prediction might not fully-capture the task-relevant dynamics. In this work, we hypothesize that direct supervision of target dynamic states (Dynamics Mapping) can learn better dynamics-informed world models. Beside the next RGB reconstruction, the world model is also trained to directly predict position, velocity, and acceleration of environment rigid bodies. To verify our hypothesis, we designed a non-prehensile 2D environment tailored to two tasks: "Balance-Reaching" and "Bin-Dropping". When trained on the first task, dynamics mapping enhanced the task performance under different training configurations (Decoupled, Joint, End-to-End) and policy architectures (Feedforward, Recurrent). Notably, its most significant impact was for world model pretraining boosting the success rate from 21% to 85%. Although frozen dynamics-informed world models could generalize well to a task with in-domain dynamics, but poorly to a one with out-of-domain dynamics. Code available at: https://github.com/Automation-Research-Team/DynamicsMapping-2D**

I. Introduction

You are holding a tray with a bottle on top. How can you move it around as efficiently as possible without dropping it? For one, you have to always keep an eye on it as you navigate. Upon observing some erroneous motion (sliding or tipping), you must adjust your trajectory to ensure stability. Even for humans, such non-prehensile (no grasping) manipulation [1] poses a significant challenge. In contrast to quasi-static tasks [2] (non-prehensile pushing or prehensile grasping), our motivating example requires more dynamic manipulation, ensuring minimal (compounding) errors and a robust recovery policy. Previously, with expert knowledge, such policies were attained through state-based controllers [3] [4]. Yet, our interest lies in exploring a vision-based data-driven approach. Such an approach can reduce the need for specialized state estimation and tracking hardware and software. Also, it would do without simplified mathematical models, expert knowledge, or approximate cost functions.

Recently, vision-based imitation learning approaches have been successfully applied to long-horizon, fine-manipulation, and in-the-wild tasks [5] [6] [7]. However, our target dynamic manipulation tasks present significant challenges under randomized conditions. We hypothesize that current approaches prompt the vision backbone to prioritize position/shape-related features. While this may suffice for quasi-static manipulation, it may be inadequate with dynamic scenarios.

In Reinforcement learning settings [8], higher-order dynamical states such velocity and acceleration are essential components of the observation space to learn successful dynamic policies. However, adaptation of such dynamical states into an imitation learning framework has not been studied. To address this gap, we propose introducing "Dynamics Mapping" as an additional training objective compelling the vision backbone to also consider dynamic features such as velocity and acceleration.

"Dynamics Mapping" can be integrated with End-to-End learning frameworks [9] [10], as well as pretraining more general world models [11] [12] [13] (for its improved generalization). In either case, more robust policies are achieved by resolving the latent features ambiguity arising from relying solely on policy loss or RGB observation reconstruction loss. The dynamics-informed latent can both better imitate the dynamic trajectory as well as generalize to new tasks with similar dynamics.

In this work, we validate our approach in a challenging 2D non-prehensile setting on two main tasks. First, we train our world model on one task where significant improvement was attained. Subsequently, the frozen (will not be further trained) world model is adapted to another task with in-domain (similar) dynamics showing good generalization. Conversely, adaptation to tasks with out-of-domain dynamics showed inadequate generalization.

The first contribution of this work is the proposal of "Dynamics Mapping", a method designed to encourage dynamics-informed latent learning from visual inputs (Section III). Secondly, we have developed a dynamic non-prehensile setting, which could serve as a benchmark for vision-based imitation learning (Section IV). Lastly, we verified our approach superiority through comparative analysis with different baselines, demonstrating enhanced task

[1]A. Mustafa, R. Hanai, I. Ramirez-Alpizar, F. Erich, R. Nakajo, and Y. Domae are with the National Institute of Advanced Industrial Science and Technology (AIST), Japan {am-mustafa, ryo.hanai, ixchel-ramirezalpizar, floris.erich, ryoichi-nakajo, domae.yukiyasu}@aist.go.jp

[2]T. Ogata is with the Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo 169-8555, Japan and also with the National Institute of Advanced Industrial Science and Technology (AIST), Japan ogata@waseda.jp

performance and generalization (Section V).

II. Related Work

Our work addresses the potential of learning dynamics-informed **world models** as a backbone for **visual imitation learning**, enhancing performance in dynamic tasks, particularly **non-prehensile manipulation**. This section provides a concise overview of these key concepts.

Visual Imitation Learning. Imitation learning is a powerful data-driven learning paradigm, yielding impressive outcomes as demonstrated in RT-1 [14]. Contrasting state-based learning, which utilizes low-dimensional input, learning directly from high-dimensional image data presents greater complexity. Nevertheless, images are more easily obtained while entailing rich scene information. In visuomotor control, the predominant model architecture integrates a visual encoder backbone—such as CNN (Convolutional Neural Network) or ViT (Vision Transformer)—with a low-level control policy, including MLP (Multi-Layer Perceptron), LSTM (Long Short-Term Memory), or Transformer models. With the shared objective of maximizing expert action likelihood for a given observation, various learning approaches may be adopted. Both the encoder and the policy can be trained jointly end-to-end by back-propagating the policy loss through both modules [9] [14]. Despite its simplicity, end-to-end approaches may suffer from reduced sample efficiency and limited generalization. Incorporating future observation reconstruction as an auxiliary loss [10] can stabilize learning and enhance sample efficiency. However, their proposed architecture predicts future states based solely on current state without action-based conditioning. Consequently, its internal transition model may over-fit the training dataset and lack the required level of generalization.

World Modeling. Towards improved generalization, world models can be trained for both latent encoding and as transition models. Typically, world models are trained independently of any task-specific policy. They can be trained to be as general as needed by training on a large dataset such as GAIA-1, which comprises 4,700 hours of autonomous driving [15]. Upon training, such models can be integrated with various control policies such as optimal control [12], reinforcement learning [11] [13], and offline reinforcement learning [16]. Embed2Control [12] learns a world model with linearized dynamics for next-state prediction. The model is then integrated with an iLQR (iterative Linear Quadratic Regulator) control policy that optimizes an approximate quadratic reward. Recurrent World Models [11] independently learn a vision encoder (based on VAE (Variational Auto-Encoder), then model the transition dynamics with a recurrent model, and finally optimize a shallow policy. Decoupling the vision encoder and transition model might fail to encode relevant features from complex scenes without sufficient temporal context. Dreamer(v3) [13] avoids this by jointly training the encoder and an RSSM (recurrent state space model), resulting in state-of-the-art performance. Using an offline dataset, DITTO [16] uses the world model to run an on-policy RL optimization algorithm (REINFORCE), utilizing the policy loss as a distance-based reward.

Compared to reinforcement learning, the application of world models in imitation learning settings has not been widely investigated, likely because the expert demonstrations dataset provides a sufficient training signal to optimize the vision-backbone/policy jointly. In such scenarios, training a world model independently might not be fully justified without a pressing need for generalization. Additionally, an independently-trained world model might not perform as well, being task-agnostic. Our work aims to evaluate the use of world models in imitation learning settings and provides the necessary adjustments to address more challenging dynamic tasks.

Non-prehensile Manipulation. The dynamic tasks of interest encompass the non-prehensile manipulation, where objects are carefully controlled by considering dynamic interactions. Existing works typically employ open-loop control using state-based optimal controllers [3] [4]. In [3], the objective was to navigate the robot end-effector to a target pose while balancing an object on a tray. A model predictive controller (MPC) was utilized to plan a trajectory that minimizes the distance to the goal while satisfying balancing constraints. In [4], sequences of motion primitives were identified for a desired motion, and these motions were executed using an LQR controller based on linearized dynamics. In this work, we aim to implement closed-loop vision-based control. We simulated two tasks with analogous objectives of balance-reaching and object tossing. Our motivating goal is reproducing these tasks in the real-world utilizing vision-based control.

III. Dynamics Mapping

This study addresses vison-based imitation learning of manipulation tasks that involve dynamic motions, specifically focusing on simulated 2D non-prehensile manipulation tasks.

Problem Setup. Given a set of expert demonstrations $D = \{\tau_0, \tau_1, \cdots, \tau_n\}$, where each trajectory $\tau_i = \{(I_t, [P_t, V_t, A_t], x_t, a_t)\}_{t=0}^{T}$ includes the raw visual observations I_t, other state information (i.e. proprioceptive) x_t, and the expert action a_t. Additionally, the per-object dynamics state comprises the rigid object's COM (center of mass) position P_t, velocity V_t, and acceleration A_t. The primary objective is to validate the significance of incorporating those dynamics states into the learning process.

A. Model Architecture

The overall model architecture, depicted in Fig. 1, comprises three main components: a world model, a policy, and a set of decoding networks. The world model's fundamental objective is to encode the input RGB image I_t into a lower-dimensional latent state z_t, capturing the necessary environment state for precise action prediction. Towards that, a simple CNN-based encoder (ENC$_{\phi_1}$, parameterized by ϕ_1) is used for latent encoding ($z_t = \text{ENC}_{\phi_1}(I_t)$). However, to tackle sequential tasks with non-Markovian dynamics (reliance on past states), the world model incorporates a transition dynamics model with a recurrent module —comprising two sequential LSTM cells—to preserve the memory essential for dynamics prediction and long-horizon tasks execution. Given the encoded latent z_t, current LSTM hidden states ($h_t = (h_{1_t}, h_{2_t})$), and the commanded action a_t, the transition model (DYN$_{\phi_2}$, parameterized by ϕ_2) predicts the next-state latent $\hat{z}_{t+1}$ and updates the hidden states ($\hat{z}_{t+1}, h_{t+1} = \text{DYN}_{\phi_2}(z_t, a_t, h_t)$). The outputs from the transition module can then be employed for decoding environmental states or predicting future actions.

While the world models capture the general transition dynamics, a task-specific policy is trained based on those world model states towards a target objective. One choice for the goal-conditioned policy is a simple Feedforward MLP architecture (i.e. Dreamer [13]). For a goal g_t and a latent state z_t, the action is calculated as in Equation (1).

$$\hat{a}_t = \pi_\theta(z_t, h_t, g_t) \tag{1}$$

However, such simple policy cannot handle non-Markovian transitions without dynamics-informed latent features. Analogous to the transition module, a recurrent policy π_θ—incorporating two additional LSTM cells—predicts the action $\hat{a}_t$ given the goal position g_t, the encoded latent z_t, the latest world model hidden states h_t, and its own internal states ($h_{\pi_t} = (h_{\pi_{1_t}}, h_{\pi_{2_t}})$) as in Equation (2).

$$\hat{a}_t, h_{\pi_{t+1}} = \pi_\theta(z_t, g_t, h_t, h_{\pi_t}) \tag{2}$$

While the world model and the policy could be trained using only the policy loss (refer to the following section), this approach can be sample-inefficient and task-specific. Alternatively, a set of decoding networks are introduced, providing the world model an additional learning signal. Typically, only an unsupervised RGB reconstruction loss is employed; however, we propose introducing supervised dynamics prediction losses for improved performance. The decoders input comprises the encoded latent $\hat{z}_t$ and world model hidden states (h_t). For RGB decoding, a simple CNN decoder (DEC$_{\zeta_1}$ with weights ζ_1) is utilized ($\hat{I}_t = \text{DEC}_{\zeta_1}(\hat{z}_t, h_{1_t}, h_{2_t})$). Regarding dynamics decoding, a set of MLP networks (with weights ζ_2,ζ_3,ζ_4) are employed ($[\hat{P}, \hat{V}, \hat{A}]_t = \text{DEC}_{\zeta_{[2,3,4]}}(\hat{z}_t, h_{1_t}, h_{2_t})$). The dynamics decoder share the input decoding layer (2-layers MLP) followed by state specific decoding head.

B. *Model Training Approaches*

In the literature, two primary approaches for training the discussed model architecture are identified: **End-to-End** training and **joint** training. In this work, we have also introduced a **decoupled** training method. For both joint and decoupled training scenarios, we suggest the incorporation of **supervised dynamics loss** to learn dynamics-informed models.

End-to-End Training. Given a task-specific dataset D, an end-to-end (E2E) training approach is commonly adopted [9]. The world model and policy are jointly optimized by minimizing the policy loss as in Equation (3). The loss is computed as the L1-norm between the true and predicted actions. Although straightforward, this approach might be sample-inefficient, necessitating larger datasets for improving performance. Moreover, such task-specific world model might exhibit limited generalization across diverse tasks.

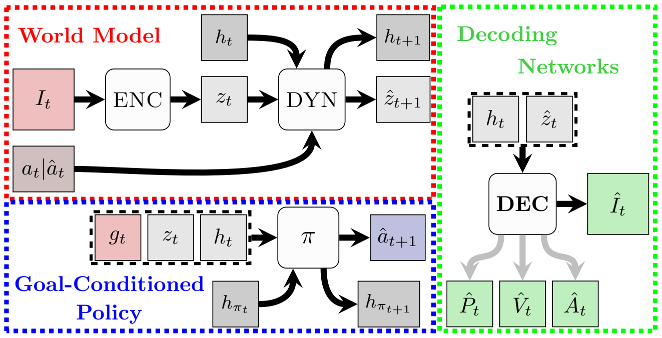


Fig. 1: The overall model architecture constitutes a world model, a policy, and a set of decoding networks. The model inputs include RGB state I_t, action a_t, and goal g_t. The internal states include latent z_t and hidden states h_t and h_{π_t} . The outputs include predicted action $\hat{a}_t$, dynamics $[\hat{P}_t, \hat{V}_t, \hat{A}_t]$, and reconstructed RGB $\hat{I}_t$.s

$$L_{\text{E2E}}(\theta, \phi_{1,2}) = E_D\left[\|a_t - \hat{a}_t(\theta, \phi_{1,2})\|\right] \tag{3}$$

Joint Training. Besides the policy loss, the RGB reconstruction loss over the next-state observation can serve as an auxiliary optimization objective [10]. This approach can enhance task-specific performance and capture the underlying environment dynamics. Alongside the discussed policy loss (denoted next as L_π), a cyclic reconstruction loss (Equation (4a)) is utilized incorporating a weighted sum of the next RGB reconstruction loss ($\|I_t - \hat{I}_t(\phi, \zeta_1)\|_2$) and the next-state latent loss ($\|\text{ENC}_{\phi_1}(I_t) - \hat{z}_t(\phi, \zeta_1)\|_2$). The model is then optimized with a weighted joint loss (with a hyperparameter β_{Joint}) as in Equation (4b). Proper tuning of β_{Joint} is crucial for balancing task-specific performance with model generalization.

$$\begin{aligned} L_{\text{RGB}}(\phi_{1,2}, \zeta_1) = E_D\big[\|I_t - \hat{I}_t(\phi_{1,2}, \zeta_1)\|_2 + \\ \beta_z\|\text{ENC}_{\phi_1}(I_t) - \hat{z}_t(\phi_{1,2})\|_2\big] \end{aligned} \tag{4a}$$

$$L_{\text{Joint}}(\theta, \phi_{1,2}, \zeta_1) = E_D\left[L_{\text{RGB}} + \beta_{\text{Joint}} L_\pi\right] \tag{4b}$$

Decoupled Training. Inspired by reinforcement learning settings [13] [11], on contrast to joint training, we propose a decoupled training approach. The world model is initially pretrained using an unsupervised RGB reconstruction loss as in Equation (4a). Upon training, the world model's weights are frozen and then employed for learning various tasks that share dynamics to the training dataset D. To train the policy, the dataset is first processed to pre-compute the latent z_t and the hidden state h_t. Subsequently, the policy π_θ is optimized with L1-loss, where only policy weights θ are optimized ($L_\pi(\theta) = E_D\left[\|a_t - \hat{a}_t(\theta)\|\right]$). Such pretraining regime can leverage more diverse datasets enabling improved generalization. Compared to joint training, it might also mitigate the need for additional tuning of the RGB/policy loss weightings.

Supervised Dynamics Loss. Instead of solely relying on the unsupervised RGB loss to train the world model, we propose integrating a supervised dynamics prediction loss to obtain a dynamics-informed world model. The position, velocity, and acceleration-based losses are utilized either independently or in combination with other losses. Like the RGB reconstruction loss, the dynamics L2-loss is calculated as shown in Equation (5). The incorporation of such losses will encourage the world model to capture features relevant to dynamics within its latent embedding and hidden states. Consequently, better policies can be trained for different dynamic tasks.

$$L_{X\in[P,V,A]}(\phi,\zeta) = E_D\left[\|X_t - \hat{X}_t(\phi,\zeta)\|_2\right] \qquad (5)$$

To investigate the significance of supervised dynamics mapping, it was introduced to both the joint and the decoupled training approaches. In the context of decoupled training, failure of the world model to capture task-relevant features would lead to poor performance. Thus, dynamics-informed models would be of greater use to the decoupled training scenarios for dynamic tasks.

IV. Task Description

A. General Overview

In this study, we initially focused on 2D tasks before progressing to 3D simulations or real-world tasks. The tasks were simulated using the Python PyMunk [17] 2D physics simulation library. Given the simulation state, the visuals were rendered in PyGame [18]; an Open-GL based renderer.

The visual inputs I_t are 64×64 RGB images. The action a_t is three-dimensional, comprising relative position updates ($a_t = \Delta X_t, \Delta Y_t, \Delta\theta_t]$). The absolute position is pre-processed into an acceleration command ($a_{\text{acc}_t} = \frac{a_{\text{pos}_t} - 2*a_{\text{pos}_{t-1}} + a_{\text{pos}_{t-2}}}{\Delta t^2}$) before sending to the PyMunk engine. The dynamic states ($[P_t, V_t, A_t]$) are a concatenation of the cart's and the block's X, Y, θ state (i.e. $P_t = [P_{c_x}, P_{c_y}, P_{c\theta}, P_{b_x}, P_{b_y}, P_{b\theta}]$). The position and velocity states were directly provided by PyMunk, while the acceleration was differentially estimated ($A_t = \frac{V_t - V_{t-1}}{\Delta t}$).

Figure 2 illustrates the normalized (to [-1,1] range) position action (applied to the cart) and the dynamics states (of the block) of a sample trajectory. Position control was favored given its smoother trajectories, which are easier to imitate, as opposed to the rapidly changing acceleration state.

B. Tasks Specification

A set of 2D tasks was designed within the same environment, utilizing identical simulation parameters. We had three main tasks as shown in Fig. 3. All tasks involve non-prehensile manipulation of a single block (initially vertical) placed on a position-controlled cart (initially horizontal), aiming towards a goal state (green).

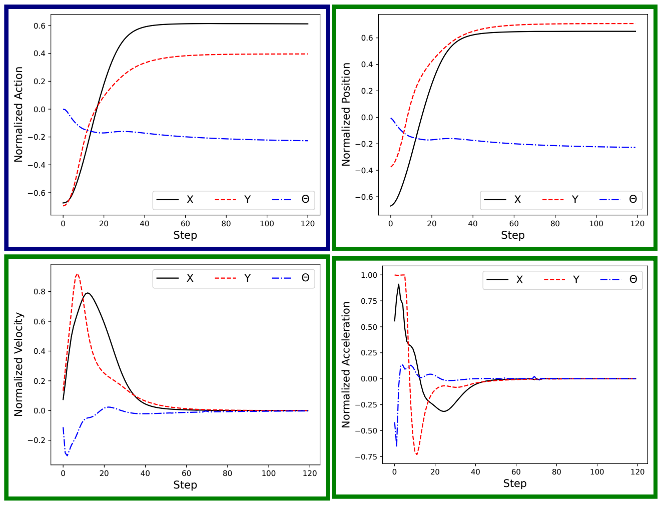
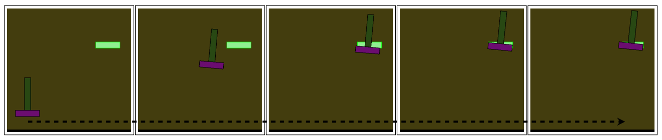


Fig. 2: A sample dataset trajectory for the "Balance-Reaching" task) depicting the normalized position action (in blue) and different dynamics states (in green).

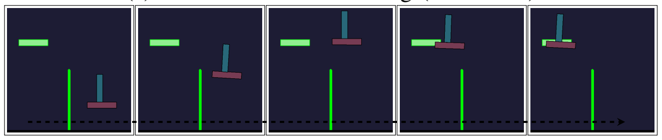
Task1: Balance-Reaching (Fig. 3a). The objective is for the cart to reach the target goal without the block falling off. In the absence of precise control, the block would lose balance and tip over. Moreover, careful acceleration control is necessary to stop at the goal within minimal time.

Task2: Balance-Reaching[v2] (Fig. 3b). Mostly similar to task1, with the need to avoid an obstacle that is added at the center (green segment).

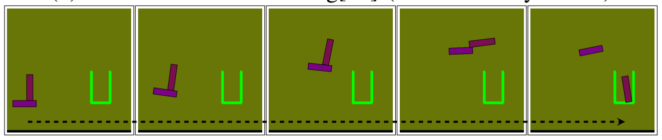
Task3: Bin-Dropping (Fig. 3c). Contrary to block balancing objective, the goal here is to drop the block into the green bin by tipping it over, thereby altering the dynamics distribution.



(a) **Task1:** Balance-Reaching (Core Task)



(b) **Task2:** Balance-Reaching[v2] (in-Domain Dynamics)



(c) **Task3:** Bin-Dropping (out-of-Domain Dynamics)

Fig. 3: Sample trajectories depicting the objective of different target tasks.

To increase task complexity and encourage improved generalization, the tasks were randomized in terms of the cart width, block height, cart/target XY position, and block shift around the cart center. The cart, block, and background colors are randomized acting as visual distractors. Figure 4 illustrates randomization samples for task1.

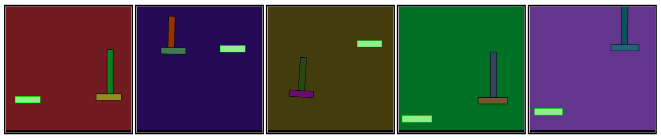
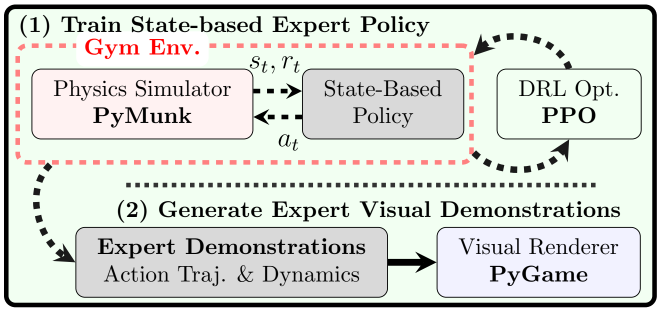


Fig. 4: Example dataset randomization including start/target positions, the cart/block dimensions, and objects colors.

C. *Expert Dataset Generation*

In the context of imitation learning, an expert dataset is necessary. For the dynamic tasks described, obtaining expert demonstrations from humans (i.e. via keyboard (discrete control) or 3D mouse (continuous control)) proved unfeasible. Objects were frequently dropped, and successful trajectories were sub-optimal (too slow) and unnatural (moving in one DOF at a time). Additionally, we could not identify simple scripted policies that can successfully complete those tasks. Consequently, we settled on training an expert DRL policy to generate the target dataset.

First, we trained a set of expert policies based on a state-based reinforcement learning optimization algorithm as depicted in Fig. 5. Compared to visual RL, State-based environments are typically easier to optimize. A PyMunk environment of the target task is first wrapped into a Gym environment [19], specifying the state-based observation space (objects positions, velocities, and dimensions, as well as the target position), action space (cart acceleration), and task-specific reward (continuous for tasks 1 and 2, based on distance to goal, or sparse for task 3, based on whether the object is dropped or in the bin). The policy is then optimized via the PPO algorithm [20], utilizing the Stable-Baselines3 library [21]. We produced a dataset of successful trajectories by employing various policy checkpoints, introducing multi-modal behavior. These trajectories were subsequently rendered in PyGame to obtain the visual RGB states.


(1) Train State-based Expert Policy
Gym Env.
Physics Simulator PyMunk
s_t, r_t
a_t
State-Based Policy
DRL Opt. PPO
(2) Generate Expert Visual Demonstrations
Expert Demonstrations Action Traj. & Dynamics
Visual Renderer PyGame
Fig. 5: Expert dataset generation utilizing a PPO-optimized state-based policy.

V. Results & Discussions

For all tasks, we generated 300 training trajectories and 50 evaluation trajectories. The tasks were simulated with a 1ms time-step (Δt) and controlled at a 20 Hz rate. The visual image (64×64) represented a $1\text{m} \times 1\text{m}$ space (1pix=15.625mm). Such discretization would limit the accuracy of dynamics prediction and goal reaching. The primary task, "Balance-Reaching," involved training different world model architectures and policies. For the other two tasks, we trained only the policy on top of frozen world models to assess model generalization.

A. *Qualitative Evaluation (Task1)*

The world model was qualitatively evaluated by visualizing the RGB image reconstruction and the dynamics prediction as depicted in Fig. 6. The RGB reconstruction fairly captured the objects' colors and their approximate positions. However, it struggled to accurately represent the objects' dimensions, relative position, and angular positions. As a result, relying solely on RGB might limit performance on dynamic tasks. Regarding dynamics, the model predicted slowly varying position states with higher accuracy compared to velocity or rapidly-changing acceleration states. The inaccuracy of goal reconstruction suggested the need for direct goal position incorporation into the policy.

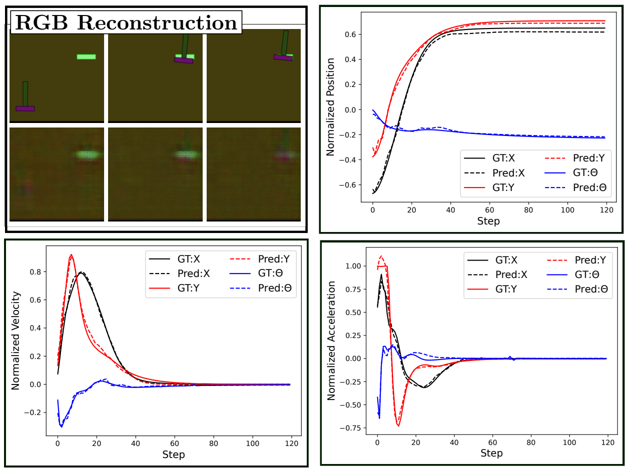
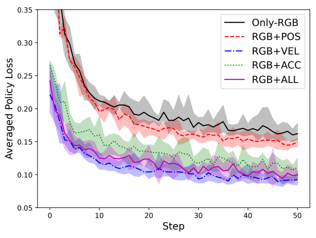
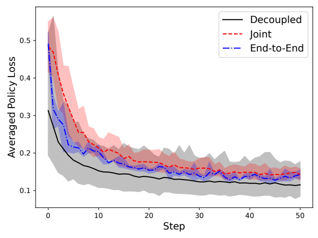


Fig. 6: Qualitative evaluation of the image reconstruction and the dynamics prediction. **(Task1)**

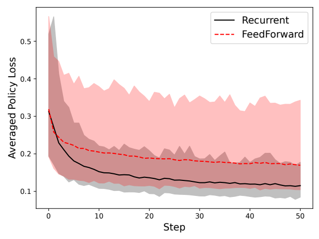
Prior to quantitative evaluation, the policy loss can serve as a strong indicator of the model performance by depicting the general tendencies over multiple runs and architectures. In Fig. 7a, under decoupled training, dynamics-informed models had lower losses than "Only-RGB" models as shown . The velocity state was the most effective when compared to position—offering limited dynamic information—or acceleration, which varies too rapidly. Combining them only increased the loss relative to the velocity-based model. In Fig. 7b), decoupled training realized lower losses compared to joint or end-to-end training thanks to limited over-fitting and the need for loss weight fine-tuning (β_{Joint}). In Fig. 7c, recurrent policies had lower losses compared to feedforward ones as they can handle non-Markovian transitions. Incorporation of different dynamical states had varying effects where position (Fig.7d) was the least effective, and velocity (Fig. 7e) had the most notable improvement.



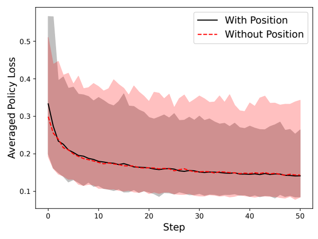
(a) w & w/o Dynamics



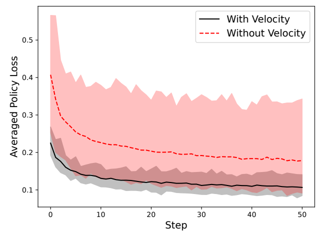
(b) Training Approach



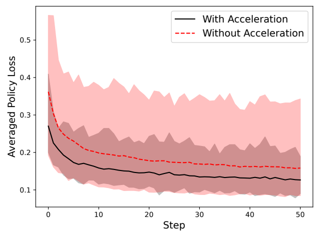
(c) Policy Type



(d) w & w/o Position



(e) w & w/o Velocity



(f) w & w/o Acceleration

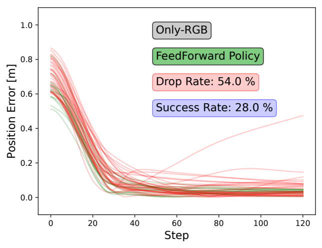
Fig. 7: Policy loss comparison of different model architectures for (a) Decoupled and (b) Joint training. **(Task1)**

B. *Quantitative Evaluation (Task1)*

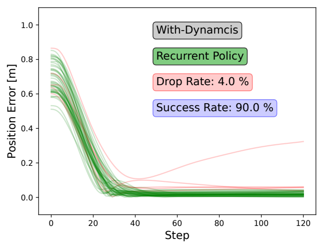
We evaluated the performance for task1 ("Balance-Reaching"). The evaluation dataset comprised 50 randomized trajectories (lasting 120 steps each). Each model was evaluated with three random seeds.

Evaluation Criteria. Figure 8 illustrates the evaluation criteria comparing two models under decoupled training. The first model is an Only-RGB model with feedforward policy, and the second utilizes dynamics-supervision and a recurrent policy. The first criterion is balancing the block throughout the trajectory. The "Drop Rate" refers to the percentage failures due to block drops. The second criterion is mean position error over the last 10 steps. A trajectory qualifies as successful if the block remains in place and the final error is under 50 mm (approximately 3 pixels). Introducing dynamics (Fig. 8b) improves performance over "Only-RGB" (Fig. 8a) by reducing both the drop rate and position error.

Decoupled Training. We compared a world model trained with an "Only-RGB" reconstruction loss against dynamics-informed models that include position (P), velocity (V), acceleration (A), or their combinations (with equal loss weightings). Given our task1 dataset, we trained a single world model (per architecture) as well as three randomly



(a) Only-RGB



(b) With Dynamics

Fig. 8: Visualization of the position error trajectory with/without dynamics mapping. Red color represents failure due to dropped block. **(Task1)**

seeded policies. Table I compares the drop rate (DR), the final position error (PE), and the success rate (SR) across models.

With a feedforward policy, the "Only-RGB" model exhibited a high drop rate resulting in an overall lower success rate. Its position error was also higher on average. The model policy, trained on poor latent features without dynamics, failed to learn appropriate multi-modal behavior and frequently went out of distribution. Introducing a recurrent policy can significantly improve the "Only-RGB" performance as it learns necessary transition dynamics during policy learning.

With feedforward policies, introducing position (P) prediction had a slightly improved performance. Yet, due to its static, non-temporal nature, relatively high drop rates were noted. The best performance was attained by incorporating velocity, resulting in the lowest DR and PE as well as the highest SR. Compared to acceleration, velocity trajectories were smoother and easier to predict (Fig. 6). Moreover, ground-truth acceleration was differentially estimated, whereas velocity was directly reported. Consequently, acceleration-based models had a relative increase in DR and PE. The combination of position/velocity improved over their standalone performance by reducing the drop rates, and position error. While the velocity addressed the out-of-distribution states causing object drop, the position improved the goal tracking performance. Other combinations (e.g. P+V+A) had comparable performance. Similar tendencies can be noted with recurrent policies with an overall higher performance. Dynamics-supervision is less significant since necessary dynamics transition are learnt by the recurrent policy to optimize policy loss.

Joint Training. We evaluated different models over three random seeds. The policy loss is optimized jointly with the world model losses ($\beta_{\text{Joint}} = 1$). The "End-to-End" model employed only the policy loss to optimize both modules. As listed in Table I, "End-to-End" and "Only-RGB" models (no dynamics) had improved performance over their decoupled training counterparts. The "End-to-End" model had marginally lower performance over the "Only-RGB" model due to potential over-fitting. Compared to decoupled training, the joint approach performance dropped slightly. Carefully tuning the loss weight hyperparameter β_{Joint} can improve

TABLE I: **Base Task – Task1.** Comparison of different models performance w & w/o dynamics.

	Decoupled [FeedForward]			**Decoupled [Recurrent]**			**Joint [Recurrent]**		
Architecture	**DR**[a][%] ↓	**PE**[mm] ↓	**SR**[%] ↑	**DR**[%] ↓	**PE**[mm] ↓	**SR**[%] ↑	**DR**[%] ↓	**PE**[mm] ↓	**SR**[%] ↑
End-to-End	-	-	-	-	-	-	19 (9.0)	71 (21.9)	57 (10.6)
Only-RGB	55 (2.5)	69 (15.6)	21 (5.7)	14 (1.6)	31 (1.3)	73 (3.8)	17 (5.0)	92 (31.4)	64 (5.9)
RGB+[P]	31 (8.4)	33 (2.8)	62 (9.1)	4 (3.3)	**21 (2.7)**	**92 (4.3)**	**5 (0.9)**	58 (17.1)	67 (8.1)
RGB+[V]	11 (2.5)	41 (5.8)	79 (1.9)	7 (1.9)	29 (5.0)	85 (3.8)	22 (6.5)	84 (31.6)	51 (10.9)
RGB+[A]	15 (3.4)	80 (21.2)	53 (5.7)	5 (0.9)	36 (3.5)	79 (4.1)	15 (3.8)	43 (4.1)	65 (3.8)
RGB+[P+V]	**7 (5.0)**	**25 (2.2)**	**85 (5.2)**	5 (2.5)	23 (3.7)	89 (0.9)	9 (1.9)	54 (11.0)	59 (9.6)
RGB+[P+A]	34 (4.3)	56 (4.7)	42 (8.6)	**2 (1.6)**	32 (6.1)	84 (1.6)	6 (4.3)	**34 (2.1)**	**79 (6.2)**
RGB+[V+A]	9 (2.5)	31 (7.0)	80 (1.6)	8 (4.3)	28 (7.0)	82 (7.1)	19 (0.9)	48 (6.1)	57 (5.7)
RGB+[P+V+A]	9 (5.0)	25 (1.6)	84 (7.5)	5 (0.9)	25 (2.2)	87 (1.9)	7 (2.5)	43 (7.2)	70 (8.6)

[a]DR = "Drop Rate", PE = "Position Error", and SR = "Success Rate".

the performance without compromising generalization. Nevertheless, like decoupled training, improved performance is generally noted upon introducing dynamics losses.

Note on Real-World Applications: After verifying the benefits of dynamics for improved performance in simulated settings, extension to real-world applications is a natural next objective. Considering the challenges of acquiring ground-truth dynamics states in real-world settings, we posit that models trained in simulation—addressing the sim-to-real gap effectively—can be zero-shot transferred to real-world settings [22]. During real-world inference, task-relevant dynamics will be implicitly encoded into the model's latent solely from the visual input, eliminating the need for explicit ground-truth dynamics.

C. *World Model Generalization*

The dynamics-informed models demonstrated an improved performance when evaluated on same task as the training dataset. Furthermore, we would like to verify whether these models exhibit good generalization behavior. To this end, we reuse a frozen world model, trained on task1, to train recurrent policies (3 seeds) for two additional tasks. Task2, while experiencing similar dynamics (in-domain), had a slightly different objective. Conversely, task3 had quite different dynamics (out-of-domain), due to increased angular positions and velocities.

In-Domain Generalization – Task2 The results presented in Table II indicate a good transfer of the learnt model to the new task. Unlike expectations, task2 dataset was better imitated resulting in higher success rates. Similar to task1, dynamics-supervision and decoupled training advantage were observed.

Out-of-Domain Generalization – Task3 For this task, the success rate (block dropped in bin) is the only relevant metric. As shown in Table III, for out-of-domain tasks, dynamics-supervision with decoupled training had limited improvement in performance compared to "Only-RGB" models. On the other hand, under joint training, dynamics-supervision had marginally lower performance,

The suboptimal performance of dynamics-based models can be visually explained in Fig. 9. These models, including the "Only-RGB" model, had poor RGB reconstruction, failing to accurately capture the cart/block pose and the goal's shape and position. Such behavior is influenced by both

TABLE II: **In-Domain Generalization – Task2.** Comparison of different models performance w & w/o dynamics.

	Decoupled		**Joint**	
Arch.	**DR**[%] ↓	**SR**[%] ↑	**DR**[%] ↓	**SR**[%] ↑
End-to-End	-	-	4 (1.6)	72 (4.3)
Only-RGB	1 (0.9)	80 (5.9)	4 (1.6)	73 (4.7)
+[P]	0 (0.0)	**91 (3.8)**	1 (0.9)	**86 (4.9)**
+[V]	2 (1.6)	80 (4.3)	3 (2.5)	79 (6.6)
+[A]	2 (1.6)	82 (4.9)	3 (2.5)	77 (6.6)
+[P+V]	1 (0.9)	**90 (3.3)**	4 (3.3)	75 (6.2)
+[P+A]	0 (0.0)	87 (2.5)	**0 (0.0)**	76 (7.1)
+[V+A]	1 (0.9)	83 (8.1)	3 (0.9)	83 (4.1)
+[P+V+A]	2 (1.6)	84 (3.3)	1 (1.9)	**87 (6.6)**

TABLE III: **Out-of-Domain Generalization – Task3.** Comparison of different models performance w & w/o dynamics.

Arch.	**Decoupled** (SR[%])	**Joint** (SR[%])
End-to-End	-	47 (4.1)
Only-RGB	44 (0.0)	46 (1.6)
+[P]	46 (4.3)	**51 (4.1)**
+[V]	52 (1.6)	40 (2.8)
+[A]	**55 (6.2)**	41 (12.3)
+[P+V]	53 (3.4)	42 (8.2)
+[P+A]	46 (2.8)	39 (4.1)
+[V+A]	41 (3.4)	46 (9.1)
+[P+V+A]	**54 (2.8)**	45 (5.2)

the world model (encoded state) and the decoder. Moreover, poor dynamics predictions in cases of extreme angular states suggest out-of-distribution (OOD) behavior, leading to OOD latent features and hidden states that pose challenges for both learning and inference.

VI. CONCLUSIONS

In this study, we propose the incorporation of supervised dynamics prediction losses to learn dynamics-informed world models. In our simplified non-prehensile manipulation tasks, the proposed models exhibited lower policy losses and enhanced performance for both decoupled and joint training regimes. Generalizing these world models to additional tasks suggests the importance of avoiding out-of-distribution states for transferable performance.

Limitations & Future Work The accuracy of vision-based dynamics inference is dependent on image size. For different tasks, a compromise between accuracy and computational efficiency necessitates consideration of varying im-

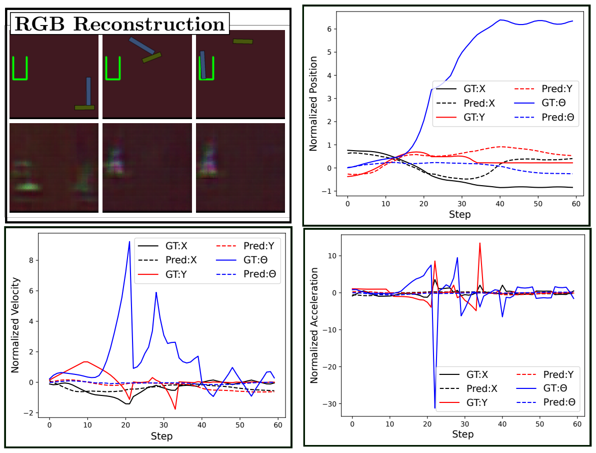


Fig. 9: Qualitative evaluation of the out-of-domain image reconstruction and the dynamics prediction. (Task3)

age sizes. Moreover, the precision of predictions is bounded by the ground-truth training dataset's accuracy. Better simulation engines and reduced simulation and control time steps may be required for certain applications. Hyperparameter tuning represents another investigation direction to assess the robustness of our proposal. In addition to the baselines presented, further comparisons with SOTA baselines—namely, Diffusion Policy [6] and Dreamer [13]—are planned. In assessing our proposal, we opted for simple neural network architectures, including vanilla CNNs and LSTMs. A CNN model proved adequate for encoding/decoding the current simple visuals, while LSTMs were chosen for their effective online performance in tasks with extended horizons. Future research will incorporate more recent models for both visual (e.g., ResNet, ViT) and sequential (e.g., Transformers, Diffusion Models) processing. Investigations will also extend to 3D simulated and real-world environments to explore sim2real generalization. Moreover, examining tasks involving multiple manipulated objects presents a promising direction for generalizing to object-centric models (e.g., GNN [23]). Lastly, we seek models that learns directly from image/action pairs without reliance on specialized dynamics datasets. Optical flow models (e.g., Flowformer [24]) can potentially estimate target scene dynamics.

ACKNOWLEDGMENT

The manuscript was reviewed for linguistic accuracy with the assistance of GPT-4. This work was supported by JST [Moonshot R&D][Grant Number JPMJMS2031].

REFERENCES

[1] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.
[2] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, "Visual imitation made easy," in *Conference on Robot Learning (CORL)*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1992–2005.
[3] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7986–7993, 2023.
[4] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4066–4073.
[5] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," in *arXiv*, 2024.
[6] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
[7] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *2023 Robotics: Science and Systems (RSS)*, 2023.
[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.
[9] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, Jan. 2016.
[10] K. Suzuki, H. Ito, T. Yamada, K. Kase, and T. Ogata, "Deep predictive learning : Motion learning concept inspired by cognitive robotics," 2023.
[11] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.
[12] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.
[13] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.
[14] A. Brohan, et al., "Rt-1: Robotics transformer for real-world control at scale," in *Robotics: Science and Systems (RSS)*, 2023.
[15] A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado, "Gaia-1: A generative world model for autonomous driving," 2023.
[16] B. DeMoss, P. Duckworth, N. Hawes, and I. Posner, "Ditto: Offline imitation learning with world models," 2023.
[17] V. Blomqvist, "Pymunk, a python 2d physics library," Nov. 2007–2024. [Online]. Available: https://pymunk.org
[18] P. Shinners, "Pygame, a python modules designed for writing video games." 2011–2024. [Online]. Available: https://pygame.org
[19] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, "Gymnasium," Mar. 2023. [Online]. Available: https://zenodo.org/record/8127025
[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.
[21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html
[22] R. Hanai, Y. Domae, I. G. Ramirez-Alpizar, B. Leme, and T. Ogata, "Force Map: Learning to Predict Contact Force Distribution from Vision," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.
[23] L. Ugadiarov and A. I. Panov, "Graphical object-centric actor-critic," *ArXiv*, vol. abs/2310.17178, 2023.
[24] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "FlowFormer: A transformer architecture for optical flow," *ECCV*, 2022.