Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

PIN-WM: Learning Physics-INformed World Models for Non-Prehensile Manipulation

Wenxuan Li[1,*] Hang Zhao[2,*] Zhiyuan Yu[2] Yu Du[1] Qin Zou[2,4] Ruizhen Hu[3,†] Kai Xu[1,†]
[1]National University of Defense Technology [2]Wuhan University
[3]Shenzhen University [4]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
[*]Equal contributions [†]Corresponding author
Project page: `https://pinwm.github.io`

***Abstract*—While non-prehensile manipulation (e.g., controlled pushing/poking) constitutes a foundational robotic skill, its learning remains challenging due to the high sensitivity to complex physical interactions involving friction and restitution. To achieve robust policy learning and generalization, we opt to learn a world model of the 3D rigid body dynamics involved in non-prehensile manipulations and use it for model-based reinforcement learning. We propose PIN-WM, a Physics-INformed World Model that enables efficient end-to-end identification of a 3D rigid body dynamical system from visual observations. Adopting differentiable physics simulation, PIN-WM can be learned with only few-shot and task-agnostic physical interaction trajectories. Further, PIN-WM is learned with observational loss induced by Gaussian Splatting without needing state estimation. To bridge Sim2Real gaps, we turn the learned PIN-WM into a group of Digital Cousins via physics-aware randomizations which perturb physics and rendering parameters to generate diverse and meaningful variations of the PIN-WM. Extensive evaluations on both simulation and real-world tests demonstrate that PIN-WM, enhanced with physics-aware digital cousins, facilitates learning robust non-prehensile manipulation skills with Sim2Real transfer, surpassing the Real2Sim2Real state-of-the-arts.**

(a) Few-shot task-agnostic data collection for world model learning

(b) Physics-aware digital cousins (c) Sim2Real policy transfer

Fig. 1: PIN-WM is learned from few-shot and task-agnostic physical interaction trajectories (random pushes of the blocks in this example), through end-to-end differentiable identification of 3D physics parameters essential to the push operation (a). The learned PIN-WM is then turned into a group of digital cousins via physics-aware perturbations (b). The resulting world models are then used to learn the task-specific policies with Sim2Real transferability (c).

I. Introduction

Non-prehensile robotic manipulation [50, 23, 86], which involves moving an object by pushing or poking, finds extensive applications in many real-world scenarios where grasping is infeasible due to the weight, size, shape, or fragility of the object, among others. Robotic push can be implemented with simpler end effectors, making systems more cost-effective and easier to deploy in certain environments [18, 67]. However, significant challenges arise from the difficulty of fully dictating the motion and pose of the object being pushed. The complex underlying dynamics, caused by factors such as friction, restitution, and inertia, make motion prediction difficult and complicate motion planning and control.

Some studies tackle non-prehensile manipulation with imitation learning [11, 77], where the reliance on expensive expert demonstrations limits their scalability. Others explore deep reinforcement learning (DRL) [69], leveraging trial-and-error in simulations to learn policies [39, 80]. However, the discrepancy between simulation and reality hinders the transfer of the learned policy to real-world environments [12, 45]. A promising alternative is to learn world models [25] of the environment dynamics in a data-driven manner, which can be used for predictive control or employed in model-based RL for better data-efficiency and Sim2Real generality [26, 28]. However, purely data-driven world models rely heavily on the quantity and quality of training data and struggle to generalize to out-of-distribution (OOD) scenarios [79].

It is well-recognized that incorporating structured priors into learning algorithms improves generalization with limited training examples [63]. A number of studies [53, 5, 67] have sought to integrate principles of physics into the development of world models. In doing so, two critical aspects require particular consideration. *The first is the differentiability of the physics parameter identification process.* ASID [53] identifies physics parameters for an established simulator using gradient-free optimization [65]. Baumeister et al. [5] adopt a similar method to learn dynamics for model predictive control (MPC) [73]. The absence of gradient feedback renders these methodologies critically dependent on data quality. However, collecting high-quality real-world trajectories itself is a challenging task [53]. Song and Boularias [67] employ a differentiable 2D physics simulator for learning planar sliding dynamics. However, 2D physics is insufficient to capture complex motions such as flipping an object through poking. *The second consideration*

lies in the necessity of state estimation in the optimization of world models. While most existing methods involve state estimation with additional modules [53, 5, 67], the recent advances in differentiable rendering [42, 75, 57] make it possible to optimize against an observational loss directly, thus saving the effort on state estimation.

We introduce **PIN-WM**, a **P**hysics-**IN**formed **W**orld **M**odel that allows end-to-end identification of a 3D rigid body dynamical system from visual observations. *First*, PIN-WM is a differentiable approach to the identification of 3D physics parameters, requiring only few-shot and task-agnostic physical interaction trajectories. Our method systematically identifies critical dynamics parameters essential to non-prehensile manipulation, encompassing inertial properties, frictional coefficients, and restitution characteristics [67, 53, 21]. *Second*, PIN-WM learns physics parameters by optimizing the rendering loss [56] induced by the 3D Gaussian Splatting [34] scene representation, facilitating direct learning from RGB images without additional state estimation modules. Consequently, the learned world model, with identified physics and rendering properties, can be readily applied to train vision-based control policies of non-prehensile manipulation using RL.

The learned PIN-WM, representing a digital twin [24] of the real-world rigid body system, may still exhibit discrepancies against reality due to the inaccurate and partial observations [43]. To bridge the Sim2Real gap, we turn the identified digital twin into plenty of digital cousins [15] through physics-aware perturbations which perturb the physics and rendering parameters around the identified values as means. Such purposeful randomization creates a group of *physics-aware digital cousins* obeying physics laws while introducing adequate varieties accounting for the unmodeled discrepancies. The resulting world model group allows learning robust non-prehensile manipulation policies with Sim2Real transferability.

Through extensive evaluation across diverse task scenarios, we demonstrate that PIN-WM is fast-to-learn and accurate, making it useful in learning robust non-prehensile manipulation skills with strong Sim2Real transfer. The overall performance surpasses the recent Real2Sim2Real state-of-the-arts [67, 53, 45] significantly. Our real-world experiments further showcase that PIN-WM facilitates Sim2Real policy transfer without real-world fine-tuning and achieves high success rates of 75% and 65% in the *Push* and *Flip* tasks, respectively. Our contributions include:

- We propose PIN-WM for accurate and efficient identification of world models of 3D rigid body dynamical systems from visual observations in an end-to-end fashion.
- We turn the identified digital twin into a group of physics-aware digital cousins through perturbing the physics and rendering parameters around the identified mean values, to support learning non-prehensile manipulation skills with robust Sim2Real transfer.
- We conduct real robot implementation to demonstrate that our approach enables learning control policies with minimal task-agnostic interaction data and attains high performance Real2Sim2Real without real-world fine-tuning.

II. Related Work

A. Non-Prehensile Manipulation

Non-prehensile manipulation [50, 23, 86] refers to controlling objects without fully grasping them. While offering flexibility, its motions are highly sensitive to contact configurations [78], requiring accurate dynamic descriptions and control policies. Mason [50] presents a theoretical framework for pushing mechanics and derive the planning by predicting the rotation and translation of an object pushed by a point contact. Akella and Mason [3] use a theoretical guaranteed linear programming algorithm to solve pose transitions and generate open-loop push plans without sensing requirements. Dogar and Srinivasa [17] adopt an action library and combinatorial search inspired by human strategies. The library can rearrange cluttered environments using actions such as pushing, sliding, and sweeping. Zhou et al. [85] model pushing mechanisms use sticking contact and an ellipsoid approximation of the limit surface, enabling path planning by transforming sticking contact constraints into curvature constraints. These methods, however, rely on simplified assumptions, either known physics parameters or idealized physical models, which are often violated in practice [60].

Deep learning methods have recently been applied to train non-prehensile policies. Some studies focus on imitation learning [35], which mimics expert behavior for specific tasks. Young et al. [77] emphasize the importance of diverse demonstration data for generalizing non-prehensile manipulation tasks, introducing an efficient visual imitation interface that achieves high success rates in real-world robotic pushing. Chi et al. [11] utilize diffusion models' multimodal action distribution capabilities [32] to imitate pushing T-shaped objects, demonstrating impressive robustness. While effective, imitation learning relies heavily on the quantity of real-world data. Otherwise, it is prone to state-action distribution shifts during sequential decision-making [6], which is a critical issue in non-prehensile tasks requiring precise contact point selection and control [86]. Hu et al. [33] conclude that imitation generalization follows a scaling law [41] with the number of environments and objects, recommending 50 demonstrations per environment-object pair. Such data requirements can be costly and prohibitive for scalability. Alternatively, deep reinforcement learning (DRL) can learn policies through trial and error in simulated environments [39, 80]. However, the large gap between simulation and reality poses significant challenges for transferring these policies to the real world [12, 45]. Building an interactive model that accurately captures real-world physical laws is crucial for learning feasible non-prehensile manipulation policies in real world.

B. World Models for Policy Learning

World models [25], which learn the environment dynamics in a data-driven manner, provide interactive environments for effective policy training [26, 28]. Hafner et al. [26] propose Dreamer, a world model that learns a compact latent representation of the environment dynamics. The following

work [74] applies Dreamer to robotic manipulation tasks, demonstrating fast policy learning on physical robots. DINO-WM [84] leverages spatial patch features pre-trained with DINOv2 to learn a world model and achieve task-agnostic behavior planning by treating goal features as prediction targets. TD-MPC [28, 29] uses a task-oriented latent dynamics model for local trajectory optimization and a learned terminal value function for long-term return estimation, achieving superiority on image-based control. Building on the success of learning from large-scale datasets [7, 61], Mendonca et al. [54] leverage internet-scale video data to learn a human-centric action space grounded world model. However, purely data-driven world models rely heavily on the quantity and quality of training data and struggle to generalize to out-of-distribution (OOD) scenarios [79, 62]. This lowers the robustness of the learned policies transferred to the real world.

Incorporating structured priors into learning algorithms is known to improve generalization with limited training data [63, 8]. Recent advances in differentiable physics have opened up new possibilities for incorporating physical knowledge into world models. Lutter et al. [48] introduce a deep network framework based on Lagrangian mechanics, efficiently learning equations of motion while ensuring physical plausibility. Heiden et al. [31] augment a differentiable rigid-body physics engine with neural networks to capture nonlinear relationships between dynamic quantities. Other works [16] demonstrate analytical backpropagation through a physical simulator defined via a linear complementarity problem. ∇Sim [59] combine differentiable physics [16, 68] and rendering [9, 37, 38, 76] to jointly model scene dynamics and image formation, enabling backpropagation from video pixels to physical attributes. This approach was soon followed by improvements with advanced rendering techniques [46, 8], or enhanced physics engines [40].

Despite those advances, only a few studies [53, 5, 67] incorporate physical property estimation into world models for non-prehensile manipulation, relying on gradient-free optimization or simplified physical models that fail to effectively handle complex interactions. Gradient-free methods rely on high-quality trajectories for system identification; lacking such data, they are prone to local optima, as demonstrated by ASID using CEM [53]. Simplified physics models, such as the 2D physics engine adopted by Song and Boularias [67], inherently struggle to capture the full 3D dynamics of real-world interactions, leading to inaccurate predictions. In contrast, PIN-WM enables end-to-end identification of 3D rigid-body dynamics from visual observations using few-shot, task-agnostic interaction data, which facilitates the training of vision-based manipulation policies with RL. PIN-WM aligns with the original, narrow-scope definition of a world model [25]: a dynamics model tailored to a specific environment for precise model-based control. This contrasts with general-purpose world foundation models like Cosmos [2].

C. Domain Randomization

Domain Randomization trains a single policy across a range of environment parameters to achieve robust performance during testing. Peng et al. [60] enhance policy adaptability to varying environmental dynamics by randomizing the environment's dynamic parameters. Miki et al. [55] train legged robots in diverse simulated physical environments. During testing, the robots first probe the terrain through physical contact, then preemptively plan and adapt their gait, resulting in high robustness and speed. Tobin et al. [71] introduce randomized rendering, e.g., textures, lighting, and backgrounds, in simulated environments to enhance real-world visual detection. Yue et al. [81] randomize synthetic images using real image styles from auxiliary datasets to learn domain-invariant representations. Dai et al. [15] introduce an automated pipeline to transform real-world scenes into diverse, interactive digital cousin environments, demonstrating significantly higher transfer success rates compared to digital twins [24].

While these randomization methods provide a simple approach for efficiently transferring simulation-trained policies to the real world, their uniform sampling of environment parameters lacks proper constraints. This results in a generated space far larger than the real-world space, increasing learning burdens [52] and often producing conservative policies with degraded performance [19]. In contrast, we perturb the physics and rendering parameters around the identified values as means. Such purposeful randomization creates a group of *physics-aware digital cousins* obeying physics laws while introducing adequate varieties accounting for the unmodeled discrepancies. The developed world model facilitates the learning of robust non-prehensile manipulation policies that transfer effectively from simulation to real-world environments.

III. Method

A. Overall Framework

We develop real-world non-prehensile manipulation skills through a two-stage pipeline: Real2Sim system identification via our physics-informed world model, and Sim2Real policy transfer enhanced by physics-aware digital cousins. We provide an overview of our framework in Figure 2.

Real2Sim System Identification: The Real2Sim stage constructs our physics-informed world model, which identifies the physics parameters of the target domain from visual observations. A world model [26] predicts the next system observation $\mathbf{o}_{t+1}$ based on the current observation $\mathbf{o}_t$ and actions $\mathbf{a}_t$:

$$\mathbf{o}_{t+1} = \mathcal{W}(\mathbf{o}_t, \mathbf{a}_t, \boldsymbol{\omega}), \tag{1}$$

where t denotes the time step and $\boldsymbol{\omega}$ represents the learnable parameters. A comprehensive physical world for robot interaction should account for visual observations, physics, and geometry [1], and we follow the convention in previous non-prehensile manipulation works [67, 86] that the geometry is assumed to be known. Therefore, our PIN-WM $\mathcal{W} = \mathcal{I} \circ g$ focuses on learning visual observations and physics of the target domain, where $\mathcal{I}$ is the differentiable rendering function

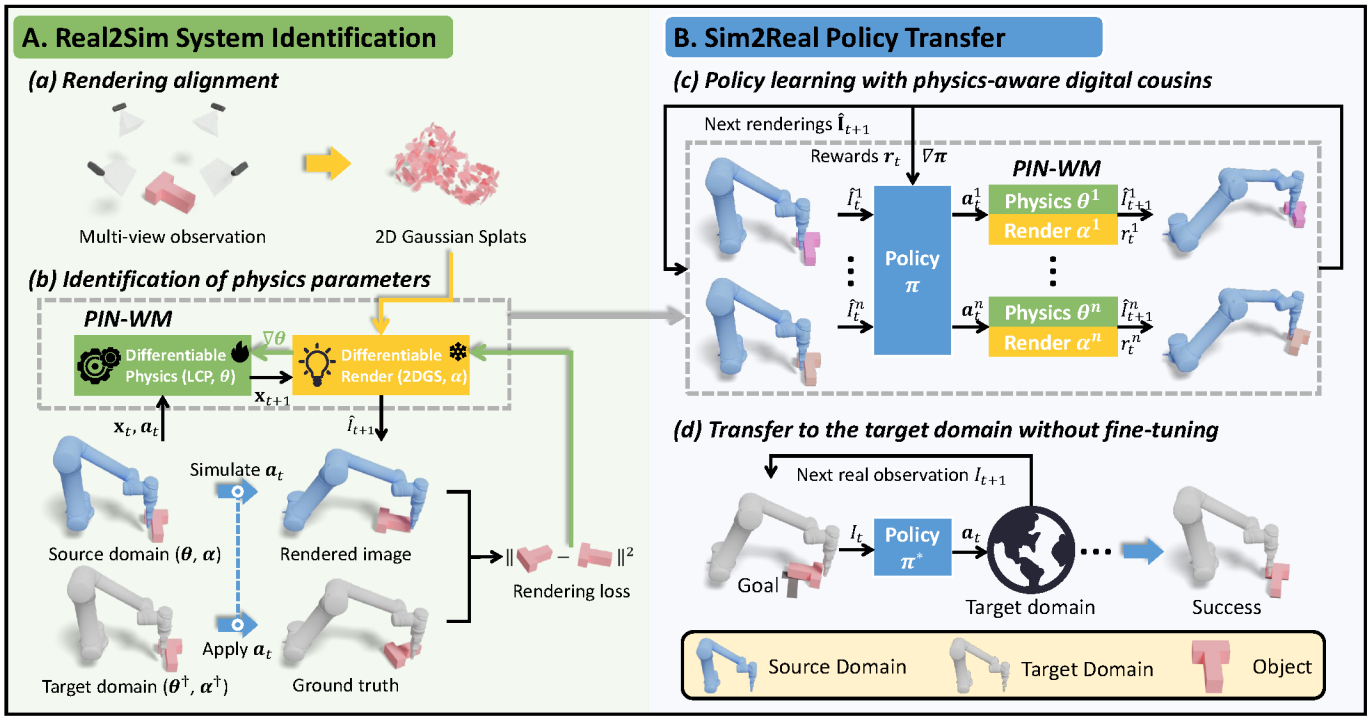

Fig. 2: Our Real2Sim2Real framework for learning non-prehensile manipulation policies. (a) The robot in the target domain moves around the object, capturing multi-view observations to estimate the rendering parameters $\boldsymbol{\alpha}$ of 2D Gaussian Splats. (b) Once optimized, $\boldsymbol{\alpha}$ is frozen. Both source and target domains apply the same task-agnostic physical interactions $\mathbf{a}_t$. In the source domain, dynamics are computed via LCP with physical parameters $\boldsymbol{\theta}$ to update the rendering. $\boldsymbol{\theta}$ is then optimized with the rendering loss between two domains. (c) The identified world model is then used for policy learning. Physics-aware perturbations are introduced to $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ to mitigate the remained discrepancies from inaccurate observations. (d) This ensemble of perturbed world models enhances the Sim2Real transferability of learned policies.

and g is the differentiable physics function. In more detail, the world model can be rephrased as:

$$I_{t+1} = \mathcal{I}(g(\mathbf{x}_t, \mathbf{a}_t, \boldsymbol{\theta}), \boldsymbol{\alpha}), \tag{2}$$

where g, parameterized by $\boldsymbol{\theta}$, predicts the next state $\mathbf{x}_{t+1}$ from current state $\mathbf{x}_t$ and action $\mathbf{a}_t$, and $\mathcal{I}$, parameterized by $\boldsymbol{\alpha}$, generates the image I_{t+1} corresponding to $\mathbf{x}_{t+1}$. Hence, $\boldsymbol{\omega} = \{\boldsymbol{\alpha}, \boldsymbol{\theta}\}$ forms all learnable parameters for $\mathcal{W}$. The goal of system identification is to optimize $\boldsymbol{\omega}$ so that the generated images resemble those observed in the target domain.

Sim2Real Policy Transfer: After system identification, we obtain the world model $\mathcal{W}$ as an interactive simulation environment. We can learn non-prehensile manipulation skills through reinforcement learning, where the learned policy is expected to achieve Sim2Real transfer without real-world fine-tuning. However, the identified world model may deviate from the real world due to inaccurate and partial observations [43]. We enhance policy transfer performance by introducing *physics-aware digital cousins* (PADC). PADC perturbs the identified system to generate meaningful training variations, which share similar physics and rendering properties while introducing distinctions to model unobserved discrepancies. This approach improves policy transferability and reduces the learning burden. The learned policy is then directly deployed in the target domain for manipulation tasks.

B. *Physics-INformed World Model*

In this section, we provide a detailed description of learning PIN-WM $\mathcal{W} = \mathcal{I} \circ g$. To fully characterize the dynamics g of our system, we adopt rigid body simulation [68] to formulate the dynamics of scene components that satisfy the momentum conservation. Therefore, we include the target object, the end-effector, and the floor in our environment state representation $\mathbf{x}_t = \{\mathbf{p}_t, \mathbf{q}_t, \boldsymbol{\xi}_t\}$, where $\mathbf{p}_t$, $\mathbf{q}_t$, and $\boldsymbol{\xi}_t$ represents their positions, orientations, and twist velocities, respectively.

We account for joint, contact, and friction constraints for rigid body simulation, and include the physical properties of those scene components that are most concerned by non-prehensile manipulation tasks [67, 53] into our physical parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{\mathrm{M}}, \boldsymbol{\theta}^{\mathrm{k}}, \boldsymbol{\theta}^{\mu}\}$, where $\boldsymbol{\theta}^{\mathrm{M}}$ represents the mass and inertia, $\boldsymbol{\theta}^{\mathrm{k}}$ represents restitution, and $\boldsymbol{\theta}^{\mu}$ represents friction coefficients. Under the rigid-body assumption where object motion follows the Newton-Euler equations, these parameters are sufficient to define collision, inertial response, and contact behavior [20, 68]. Properties like elasticity or plasticity fall outside the rigid-body scope.

The differentiable rendering function $\mathcal{I}$ is used to align the visual observations between the source domain and the target domain. Note that from a differentiable physics perspective, the floor is stationary, and the robot is the one applying force actively, whose dynamics will not be affected by other objects, so only the motion of the target object needs to be observed and aligned. Therefore, our rendering function only

consider the target object, that is $\mathcal{I}(\mathbf{x}_t, \boldsymbol{\alpha}) = \mathcal{I}(\mathbf{x}_t^o, \boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ represents the rendering parameters specifically defined for the target object, and the rendered image will change with the update of object pose.

The learning process of our PIN-WM starts with optimizing $\boldsymbol{\alpha}$ for rendering alignment, and uses optimized $\boldsymbol{\alpha}^*$ to guide the identification of physical parameters $\boldsymbol{\theta}$ for simulation.

Rendering Alignment: To optimize the rendering parameters $\boldsymbol{\alpha}$ for the target object o, the robot end-effector moves around o in its initial state $\mathbf{x}_0^o$ and captures multiple static scene images with an eye-in-hand camera $\mathbf{I}^s = \{I_0^s, ..., I_m^s\}$, as demonstrated in Figure 2(a). To make sure that the rendering function $\mathcal{I}$ can generalize to new viewpoints or object poses, we adopt 2D Gaussian Splatting (2DGS) [34] as the render. Compared to 3D Gaussian splatting [42], 2DGS is more effective in capturing surface details.

2DGS renders images by optimizing a set of Gaussian elliptical disks, which are defined in local tangent uv planes in world space:

$$P(u, v) = \mathbf{p}_k + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v = \mathbf{H}(u, v, 1, 1)^\top, \quad (3)$$

where $\mathbf{p}_k$ is the central point of the k-th 2D splat. $\mathbf{t}_u$ and $\mathbf{t}_v$ are principal tangential vectors, and $\mathbf{t}_w = \mathbf{t}_u \times \mathbf{t}_v$ presents the primitive normal. $\mathbf{R} = [\mathbf{t}_u, \mathbf{t}_v, \mathbf{t}_w]$ is a 3×3 rotation matrix and $\mathbf{S} = (s_u, s_v)$ is the scaling vector. The 2D Gaussian for static object representation can be equivalently represented by a homogeneous matrix $\mathbf{H}_0 \in 4 \times 4$:

$$\mathbf{H}_0 = \begin{bmatrix} s_u \mathbf{t}_u & s_v \mathbf{t}_v & \mathbf{0} & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{RS} & \mathbf{p}_k \\ \mathbf{0} & 1 \end{bmatrix}. \quad (4)$$

During the optimization, we randomly splat 2D disks onto the object surface G for high-quality rendering initialization. For an image coordinate (x, y), volumetric alpha blending integrates alpha-weighted appearance to render the image $\hat{I}$:

$$\hat{I}(x, y) = \sum_{i=1} \boldsymbol{\alpha}_i^c \boldsymbol{\alpha}_i^o \mathcal{G}_i(\mathbf{u}(x, y)) \prod_{j=1}^{i-1} \left(1 - \boldsymbol{\alpha}_j^o \mathcal{G}_j(\mathbf{u}(x, y))\right), \quad (5)$$

where $\boldsymbol{\alpha}_i^c$ and $\boldsymbol{\alpha}_i^o$ are the color and opacity of the i-th Gaussian, $\mathbf{u}(x, y)$ represents the intersection point between the ray emitted from the camera viewpoint through the image pixel (x, y) and the plane where the 2D Gaussian distribution resides in 3D space, $\mathcal{G}(\mathbf{u})$ is the 2D Gaussian value for intersection $\mathbf{u}$, indicating its weight.

This differentiable rendering models the visual observation of the target object o in its initial state, and the corresponding parameters $\boldsymbol{\alpha}$, including all 2DGS parameters, are optimized with the following loss function:

$$\mathcal{L} = \mathcal{L}_c + \omega_d \mathcal{L}_d + \omega_n \mathcal{L}_n, \quad (6)$$

where $\mathcal{L}_c$ combines rendering loss [56] $\mathcal{L}_r = \|\hat{\mathbf{I}} - \mathbf{I}^s\|_2^2$ with the D-SSIM term [42]. $\mathcal{L}_d$ and $\mathcal{L}_n$ are regularization terms for depth distortion and normal consistency [34], respectively.

Once the optimal parameters $\boldsymbol{\alpha}^*$ are obtained for the target object o in its initial state $\mathbf{x}_0^o$, any change of the object pose can lead to the rendering updating, achieved by transforming the 2DGS accordingly. In more detail, for any new object state $\mathbf{x}_t^o$, we can convert its corresponding pose to a transformation matrix $\mathbf{T}_t^o \in 4 \times 4$ [30]. We then apply $\mathbf{T}_t^o$ to initial Gaussian splats represented by $\mathbf{H}_0$, resulting in the transformed homogeneous matrix:

$$\mathbf{H}_t = \mathbf{T}_t^o (\mathbf{T}_0^o)^{-1} \mathbf{H}_0. \quad (7)$$

where $\mathbf{T}_0^o$ represents the static object pose in its initial state $\mathbf{x}_0^o$ as the reference. $\mathbf{H}_t$ can be used to render the new image for the target object with an updated state, denoted as $\mathcal{I}(\mathbf{x}_t, \boldsymbol{\alpha}^*)$.

Identification of Physics Parameters: With the optimized rendering parameter $\boldsymbol{\alpha}^*$, we further estimate the physics properties $\boldsymbol{\theta}$ for simulation, based on the gradient flow from visual observations established by the differential render. The robot interacts with the object in state $\mathbf{x}_t$ through a set of task-agnostic actions $\mathcal{A} = \{\mathbf{a}_t, ..., \mathbf{a}_{t+n-1}\}$ to collect a video $\mathbf{I}^d = \{I_{t+1}^d, ..., I_{t+n}^d\}$ capturing dynamics, as shown in Figure 2(b). The transformed observations $\hat{\mathbf{I}} = \{\mathcal{I}(\mathbf{x}_{t+i}, \boldsymbol{\alpha}^*)\}_{i=1}^n$ are then obtained in simulation with Equation 5, where $\mathbf{x}_{t+i} = g(\mathbf{x}_{t+i-1}, \mathbf{a}_{t+i-1}, \boldsymbol{\theta})$ is the updated state when applying action $\mathbf{a}_{t+i-1}$. The physics parameter $\boldsymbol{\theta}$ is then estimated by minimizing the discrepancy between the generation $\hat{\mathbf{I}}$ and observation $\mathbf{I}^d$. Therefore, we can represent the objective of the physics estimation as:

$$\min_{\boldsymbol{\theta}} \quad \mathcal{L}_r(\boldsymbol{\theta}) = \sum_{i=1}^{n} \|\mathcal{I}(g(\mathbf{x}_{t+i-1}, \mathbf{a}_{t+i-1}, \boldsymbol{\theta}), \boldsymbol{\alpha}^*) - I_{t+i}^d\|_2^2. \quad (8)$$

What remains now is to develop a differentiable physics model $\mathbf{x}_{t+1} = g(\mathbf{x}_t, \mathbf{a}_t, \boldsymbol{\theta})$ for simulation, predicting the next object pose $\mathbf{x}_{t+1}$ based on current state $\mathbf{x}_t$ and action $\mathbf{a}_t$. Note that previous work estimates physics parameters by differentiating the impact of external wrenches on objects [68]. However, for robotic manipulation, we cannot assume that all robot parts support wrench measuring. Therefore, we choose the translation $\mathbf{d}_t$ of robot end-effector as the action, i.e., $\mathbf{a}_t = \mathbf{d}_t$.

We formulate this system identification process as a velocity-based Linear Complementarity Problem (LCP) [16, 68] which solves the equations of motion under global constraints. Here, we use LCP to first estimate $\boldsymbol{\xi}_{t+1}$ from $\mathbf{x}_t$, and then further use those two together to update the remaining $\mathbf{p}_{t+1}$ and $\mathbf{q}_{t+1}$. In more detail, given a time horizon H which describes the duration of an action's effect, LCP updates twist velocities of each scene component $\boldsymbol{\xi}_t$ to $\boldsymbol{\xi}_{t+1}$ after H, where $\boldsymbol{\xi}_t$ includes linear velocities $\mathbf{v}_t$ and angular velocities $\boldsymbol{\Omega}_t$. The updated velocity $\boldsymbol{\xi}_{t+1} = \{\mathbf{v}_{t+1}, \boldsymbol{\Omega}_{t+1}\}$ is then used to calculate the updated pose $\{\mathbf{p}_{t+1}, \mathbf{q}_{t+1}\}$ integrated by the semi-implicit Euler method [47]:

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + H \cdot \mathbf{v}_{t+1}, \\ \mathbf{q}_{t+1} &= \text{normalize}(\mathbf{q}_t + \frac{H}{2}([0, \boldsymbol{\Omega}_{t+1}] \otimes \mathbf{q}_t)), \end{aligned} \quad (9)$$

where $\mathbf{p}_t$ and $\mathbf{q}_t$ are the object's position and orientation represented by a quaternion. $[0, \boldsymbol{\Omega}_{n+1}]$ represent quaternion

constructed from the angular velocity $\mathbf{\Omega}_{n+1}$, and $\otimes$ denotes the quaternion multiplication.

The LCP is solved following the framework by Cline [13], where the goal is to find velocities $\boldsymbol{\xi}_{t+1}$ and Lagrange multipliers $\boldsymbol{\lambda}_e, \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_f, \boldsymbol{\gamma}$ satisfying the momentum conservation when a set of constraints is included:

$$\begin{aligned}
&\boldsymbol{\theta}^{\mathrm{M}}\boldsymbol{\xi}_{t+1} = \boldsymbol{\theta}^{\mathrm{M}}\boldsymbol{\xi}_t + \mathbf{f}^{\mathrm{g}} \cdot H + \mathbf{J}_e\boldsymbol{\lambda}_e + \mathbf{J}_c\boldsymbol{\lambda}_c + \mathbf{J}_f\boldsymbol{\lambda}_f, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Rigid Body Dynamics Equation)} \\
&\mathbf{J}_e\boldsymbol{\xi}_{t+1} = 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Joint Constraints)} \\
&\mathbf{J}_c\boldsymbol{\xi}_{t+1} \geq -\boldsymbol{\theta}^{\mathrm{k}}\mathbf{J}_c\boldsymbol{\xi}_t \geq -\mathbf{c}, \qquad\qquad\qquad\qquad \text{(Contact Constraints)} \\
&\mathbf{J}_f\boldsymbol{\xi}_{t+1} + \mathbf{E}\boldsymbol{\gamma} \geq 0, \quad \boldsymbol{\theta}^{\mu}\boldsymbol{\lambda}_c \geq \mathbf{E}^{\top}\boldsymbol{\lambda}_f, \quad \text{(Friction Constraints)}
\end{aligned} \tag{10}$$

where $\mathbf{f}^{\mathrm{g}}$ is the gravity wrench, $\boldsymbol{\lambda}_e, \boldsymbol{\lambda}_c, \boldsymbol{\lambda}_f, \boldsymbol{\gamma}$ are constraint impulse magnitudes, $\mathbf{E}$ is a binary matrix making the equation linearly independent at multiple contacts, and $\mathbf{J}_e, \mathbf{J}_c, \mathbf{J}_f$ are input Jacobian matrices describing the joint, contact, and friction constraints, please refer to [16] for construction details. Here, joint constraints ensure that connected objects maintain a specific relative pose, contact constraints prevent interpenetration, and friction constraints enforce the maximum energy dissipation principle.

We adopt the primal-dual interior point method [51] as the LCP solver to obtain the solution $\boldsymbol{\xi}_{t+1}$ while establishing gradient propagation from $\boldsymbol{\xi}_{t+1}$ to $\boldsymbol{\theta}$. We then apply the method described in [4] to derive the gradients of the solution with the objective in Equation 8. The output of the physics model g is contributed by both $\boldsymbol{\theta}$ and the last state $\mathbf{x}_t$, where $\mathbf{x}_t$ also depends on $\boldsymbol{\theta}$. Therefore, the gradients with respect to $\boldsymbol{\theta}$ can be expressed as:

$$\begin{aligned}
\frac{d\mathcal{L}_r(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \sum_{i=1}^{n} &(\mathcal{I}(g(\mathbf{x}_{t+i-1}, \mathbf{a}_{t+i-1}, \boldsymbol{\theta}), \boldsymbol{\alpha}^*) - I^d_{t+i}) \\
&\cdot (\frac{\partial \mathcal{I}}{\partial g}\frac{\partial g}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{I}}{\partial g}\frac{\partial g}{\partial \mathbf{x}_{t+i-1}}\frac{\partial \mathbf{x}_{t+i-1}}{\partial \boldsymbol{\theta}}).
\end{aligned} \tag{11}$$

As LCP is widely adopted by mainstream simulators [14, 49], the estimated $\boldsymbol{\theta}$ can be compatible with existing simulation environments [36, 10] as well.

Since we use a velocity-based LCP, the end-effector translation $\mathbf{d}$ can be converted into velocity $\boldsymbol{\xi}^e = \mathbf{d}/H$, where H is the action time horizon. This equation holds because the robot's mass is typically much greater than the object's mass, allowing its own dynamics to be ignored. The pose and velocity of the floor is kept stationary during the whole process, and only its geometry and physical parameters are used for solving the dynamics equation. Moreover, in robot manipulation, the action time horizon H is usually inequivalent to simulation step size h, while the latter is set small enough to ensure accurate object pose integration (Equation 9). The input sub-action for each recursion is derived by dividing the original action $\mathbf{a}_t$ into H/h segments. We propagate recursive derivatives of Equation 11 across H/h simulation time steps and optimize $\boldsymbol{\theta}$.

C. Physics-aware Digital Cousins

The learned world model $\mathcal{W}$ reduces the gap with real worlds and provides an interactive environment for manipulation policy learning. However, inconsistencies with the real world remain due to inaccurate and partial observations. Domain randomization [71] randomizes system parameters in the source domain during training to cover the problem space of the target, but it often lacks adequate constraints, increasing training burdens and reducing policy performance. Therefore, We propose *physics-aware digital cousins*, which perturb the rendering and physics near the system's identified parameters, as illustrated in Figure 2(c).

We adopt all estimated rendering parameters $\boldsymbol{\alpha}^*$ and physics parameters $\boldsymbol{\theta}^*$ for generating digital cousins. For rendering, we adopt the spherical harmonics (SH) parameters $\boldsymbol{\alpha}^{\mathrm{sh}} \subset \boldsymbol{\alpha}^*$ [58], which represent the directional rendering component of the 2D Gaussian. Perturbing $\boldsymbol{\alpha}^{\mathrm{sh}}$ allows modeling of lighting and material variations. We randomize the system parameters $\boldsymbol{\omega}^r = \{\boldsymbol{\alpha}^{\mathrm{sh}}, \boldsymbol{\theta}^*\}$ by sampling $\tilde{\boldsymbol{\omega}}^r$ from a uniform distribution:

$$\tilde{\boldsymbol{\omega}}^r \sim \mathcal{U}(\boldsymbol{\omega}^r \cdot (1-\delta), \boldsymbol{\omega}^r \cdot (1+\delta)) \tag{12}$$

where δ indicates perturbation magnitude. For SH parameters, randomization is applied separately to each splat. To ensure zero-shot policy transfer, we perturb the identified parameters with $\delta = 0.1$ to generate digital cousins. Our physics-informed world model is compatible with arbitrary reinforcement learning methods; we adopt Proximal Policy Optimization (PPO) [66] for its ease of implementation. The learned policy is then directly deployed in the target domain world for manipulation tasks, as shown in Figure 2(d).

IV. Results and Evaluations

With our experimental evaluations, we aim to answer the following questions:

- Does our method outperform other Real2Sim2Real methods in learning deployable manipulation policies?
- Does PIN-WM achieve more accurate system identification compared to existing approaches?
- Does the proposed physics-aware digital cousins (PADC) help with policy transfer?
- Can our method deliver superior performance in real-world settings?

We conduct experimental evaluations in both simulation and the real world. Simulators provide ground truth for evaluating system identification accuracy and hence offer comprehensive answers to the first three questions, while the real-world tests are used to validate the effectiveness of policy deployment regarding the last question.

We evaluate our method on rigid body motion control. The robot's objective is to perform a sequence of non-prehensile actions to move an object into a target pose. Actions are specified as translations of the end-effector. We set up two tasks: *push* [11] and *flip* [86]. The push task is to move a planar object on a plane to a target pose, involving 2D translation in the xy-plane and 1D rotation around the z-axis. The flip task

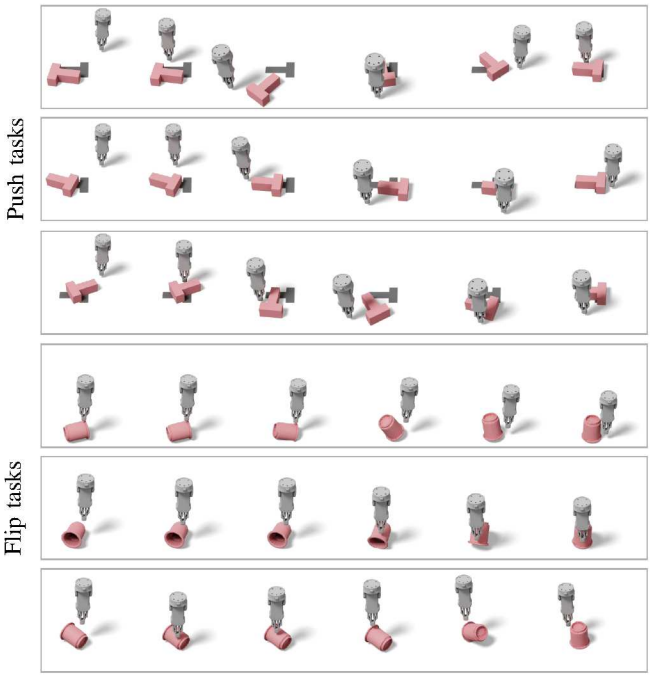

Fig. 3: Manipulation trajectories in simulation obtained by our method for both *push* and *flip* tasks.

is to poke an object to turn it from a lying pose to an upside-down pose, which requires 3D rotation and 3D translation.

A. Evaluations in Simulation

Experiment setup: In simulation, we collect a *single* task-agnostic trajectory that the target object is pushed forward along a straight line by the robot end-effector for a predefined distance in the target domain. After that, any access to the target domain is prohibited. Since our estimated parameters are compatible with existing simulators, we integrate estimations to the Bullet engine [14] for high-performance physics simulation. With the learned simulator, we train manipulation policies with only RGB images as input. We maintain 32 parallel threads for efficient training, each running an independent physics-aware digital cousin. The initial object pose is randomized for each episode. After the episode terminates for each thread, the environment is replaced with a newly sampled digital cousin. Trained policies are then directly deployed to the target domain for evaluation. For both push and flip tasks, we set a relatively low friction in the target domain to highlight the importance of physics identification. Figure 3 demonstrates several manipulation trajectories obtained by our method for both push and flip tasks with different initial states.

Evaluation metrics: To answer the first three questions, our evaluation metrics focus on both the manipulation policies and the world models. We measure the success rate $Succ\%$ of a policy if the task is completed within a threshold of 100 steps for push and 25 steps for flip. We also consider the required number of steps to complete a task, denoted as $\#Steps$. We evaluate the accuracy of a world model using *one-step error* [44] which measures the distance between the final object states after applying one sampled action to the identified model and the target-domain simulator. This error is computed separately for translation and rotation differences, measured in meters and radians, respectively.

TABLE I: Comparisons on policy performance in the target domain.

Methods	Tasks			
	Push		*Flip*	
	$Succ\%$	$\#Steps$	$Succ\%$	$\#Steps$
Random	0%	100.0	0%	25.0
Dreamer V2 [27]	1%	99.9	0%	25.0
Diffusion Policy [11]	13%	91.1	10%	23.3
RoboGSim [45]	19%	82.6	21%	20.5
Domain Rand [60] + $\mathcal{I}$	33%	73.6	32%	20.0
2D Physics [67] + $\mathcal{I}$	55%	60.6	8%	22.6
ASID [53] + $\mathcal{I}$	58%	57.6	11%	22.2
PIN-WM w/o PADC	92%	32.1	70%	12.0
PIN-WM w/ PADC	**97%**	**30.1**	**83%**	**11.4**

Baseline methods: We compare our method with various types of approaches for training non-prehensile manipulation skills, including:

- *Methods that rely purely on data.* A representative is the well-known Dreamer V2 [27], which is a latent-space dynamics model from data for handling high-dimensional observations and learning robust policies. Given their strong reliance on data quantity, we provide 100 task-agnostic trajectories. Based on the learned expert policy, we train non-prehensile manipulation skills via imitating expert demonstrations [35] of 100 *task-completion* trajectories similar to Chi et al. [11].
- *Methods with pre-defined physics-based world models.* We use Bullet [14] as the default simulator. Following the standard domain randomization approach [60], we randomize the physics parameters in Bullet, including mass, friction, restitution, and inertia, across a broad range for policy training. We employ our learned rendering function $\mathcal{I}$ as the renderer. We set a variant with fixed, random physics and rendering parameters where no system identification or randomization is involved, denoted as *Random*. We also compare with RoboGSim [45], which optimizes only rendering parameters using 3DGS [42] but not physics parameters.
- *Methods with learned physics-based world models.* We compare with ASID [53] and the method of Song and Boularias [67]. The former performs system identification using gradient-free optimization. The latter leverages differentiable 2D physics (thus referred to as 2D Physics). Since neither of the two methods learns rendering parameters and their trained policies cannot work without aligned visual input, we add our rendering function $\mathcal{I}$ to enhance these two methods.

Note that all physics-based methods being compared are trained with the same task-agnostic trajectories as PIN-WM, for fair comparison. All policies are trained until no significant success rate performance can be gained and are then deployed directly to the target domain for evaluation. We also conduct an ablation study of our method that trains policies without PADC. More implementation details of baseline methods are provided in Appendix A.

Comparisons on policy performance: We conduct 100 episodes of tests for each method and report the comparison results in Table I. Our method achieves the best performance for both non-prehensile manipulation tasks, thanks to the accurate system identification of PIN-WM and the meaningful digital cousins of PADC. Without PADC, our method still outperforms others, although with a performance decrease.

The purely data-driven world model Dreamer V2 [27], albeit having access to more task-agnostic data, fails to accurately approximate the dynamics of the target domain, resulting in poor performance of the trained and deployed policies. Diffusion Policy [11], relies on more expensive task-completion data, also presents inferior performance due to the limited training data quantity and hence poor out-of-distribution generalization. These results highlight the importance of incorporating physics priors in learning world models.

For those methods with pre-defined physics-based world model, Domain Rand [60] + $\mathcal{I}$ introduces excessive randomness, making the task learning more difficult. We provide a detailed explanation in Appendix B of why Domain Rand + $\mathcal{I}$ struggles; smaller-scale randomizations around ground-truth physical parameters improve its effectiveness, though knowing these parameters is unrealistic. RoboGSim [45] optimizes only for rendering parameters but not physics ones, also leading to performance degradation. In contrast, our physics-aware digital cousin design, perturbing the physics and rendering parameters around the identified values as means, creates meaningful digital cousins allowing for learning robust policies with Sim2Real transferability.

Moreover, the policies trained with physics-based alternatives exhibit unsatisfactory performance in the target domain. One reason is that their world models failed to effectively capture the target-domain dynamics. ASID [53] leads to sub-optimal solutions due to its inefficient gradient-free optimization. Although 2D Physics [67] accounts for 2D differentiable physics and achieves satisfactory push performance, its performance on the *flip* task degrades since it involves 3D rigid body dynamics. These experiments collectively demonstrate that an accurate identification of both physics and rendering parameters is crucial for learning non-prehensile manipulation skills. Although PIN-WM performs physical parameter identification under the assumption of perfect geometry, we also provide experiments in Appendix B showing that, even with noise, the training environment constructed by PIN-WM effectively supports policy training.

Comparisons on system identification: We compare the accuracy of system identification of both data-driven and physics-based approaches. This is done by measuring the one-step error after applying the same randomly sampled action to the same surface point of the target object. The results are reported in Table II.

We observe that the data-driven method Dreamer V2 [27], as expected, suffer catastrophic performance degradation when generalizing to new state and action distributions. ASID [53] shows lower accuracy compared to PIN-WM in both push and flip tasks, since it is difficult for gradient-free optimization to find a good solution in a finite time due to the large search space. Although 2D Physics [67] adopts a differentiable framework, the 2D model finds difficulties in handling 3D rigid body dynamics, resulting in low prediction accuracy. In contrast, our method learns 3D rigid body physics parameters through differentiable optimization, achieving superior performance in both push and flip scenarios. We present the learning curves of the push task in Figure 4, demonstrating the stability and efficiency of PIN-WM during training. We can observe that Dreamer V2 quickly converges on the training dataset, but it does not generalize well on the test dataset. We also provide the physical parameters identified by each method, along with the ground truth parameters, in Appendix B.

TABLE II: Comparisons on system identification accuracy across different methods, using *one-step error* of the predicted trajectory. "Trans." and "Rot." are translation and rotation errors, respectively.

Methods	Tasks			
	Push		*Flip*	
	Trans.	Ori.	Trans.	Ori.
Dreamer V2 [27]	7.6×10^{-2}	5.6×10^{-1}	2.3×10^{-1}	2.0
2D Physics [67]	4.2×10^{-2}	5.4×10^{-1}	2.1×10^{-1}	1.6
ASID [53]	3.0×10^{-2}	4.0×10^{-1}	1.4×10^{-1}	1.6
PIN-WM	$\mathbf{1.7\times10^{-2}}$	$\mathbf{1.3\times10^{-1}}$	$\mathbf{1.4\times10^{-2}}$	**0.3**

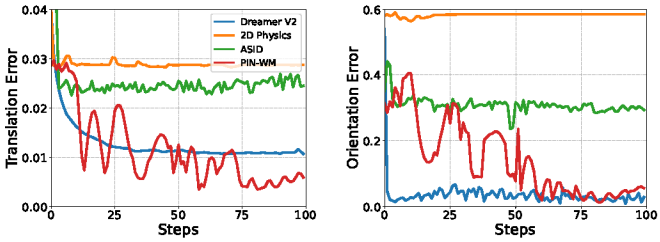

Fig. 4: Transition and orientation errors of *push* task during training.

B. Evaluations in Real-World

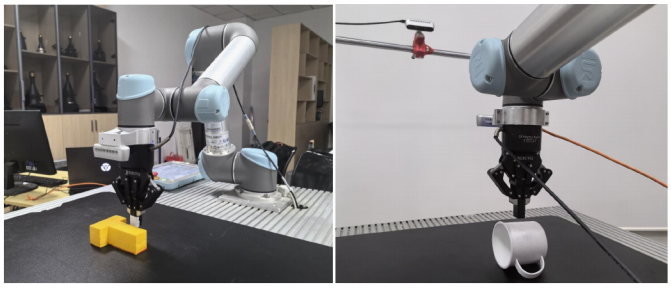

Fig. 5: Our real-world experiment setup.

Experiment setup: Our hardware setup consists of a robot, an eye-in-hand camera, and an eye-to-hand camera, as shown in Figure 5. Given a real-world object o and its mesh geometry G, we use FoundationPose [72] to estimate initial object pose $\mathbf{T}_0^o$ in the world coordinate system. We set the mesh geometry G with pose $\mathbf{T}_0^o$ in the simulator, and sample a series of surface points on the transformed mesh $\mathbf{T}_t^o G$ as the initialization for 2D Gaussian Splatting. The robot then moves around the object and captures the time-lapse video sequence

TABLE III: Real-world deployment performance.

Methods	Tasks			
	Push		*Flip*	
	Succ%	*#Steps*	*Succ%*	*#Steps*
Random	0%	100.0	0%	25.0
Domain Rand [60] + $\mathcal{I}$	10%	87.5	25%	18.5
RoboGSim [45]	30%	72.7	15%	21.2
2D Physics [67] + $\mathcal{I}$	35%	70.7	5%	24.1
ASID [53] + $\mathcal{I}$	40%	64.6	10%	22.4
PIN-WM w/o PADC	65%	45.2	60%	13.2
PIN-WM w/ PADC	**75%**	**37.5**	**65%**	**11.3**

$\mathbf{I}^s$, while recording the corresponding camera pose sequence $\{\mathbf{T}_0^c, ..., \mathbf{T}_n^c\}$. We segment the region of interest with SAM 2 [64] and optimize the 2D Gaussian with the objective in Equation 6, aligning rendering with the real world. After that, we apply a straight line of translational actions to push the object o in the real world. A dynamic video $\mathbf{I}^d$ is captured by the eye-to-hand camera to be used for optimizing the physics parameters $\boldsymbol{\theta}$ with Equation 8.

Baseline methods: In the real-world setting, collecting a large amount of trjactories, either task-agnostic or task-completion, is highly expensive. Therefore, we only compare with policy learning methods requiring *no or few-shot real-world data*, thus excluding data-driven methods such as Dreamer V2 [27] and Diffusion Policy [11].

Comparisons on policy performance: We evaluate real-world performance with both *push* and *flip* tasks under identical initial conditions across 20 trials. The push task requires pushing the T-shaped object to the red target position, while the flip task involves flipping a mug from its side to an upside-down pose. The results are summarized in Table III, showing that PIN-WM can complete the task with higher success rates and fewer steps.

Conventional simulators with random parameters fail to produce transferable policies due to physical and rendering misalignment. Although having integrated our rendering function $\mathcal{I}$, the naive randomization of Domain Rand [60] still creates noisy variations that degrade policy learning. This is also demonstrated by the comparisons with RoboGSim [60], which aligns rendering but not physics parameters. With a more accurate estimation of physics parameters, 2D Physics [67] and ASID [53] obtain slightly better results but are still inferior to PIN-WM. By learning both physical parameters and rendering representations through differentiable optimization, together with our physics-aware digital cousins design, our approach attains much better performance in real-world deployments.

We show comparisons of real-world trajectories of *push* task in Figure 6. Our method successfully pushes the T-shaped object to the target pose with a few steps. In contrast, alternative approaches either require longer trajectories or fail to complete the task. We also verify the effectiveness of our method by demonstrating how it completes the *push* task on a larger T-shaped object in Figure 7, demonstrating its adaptability to varied shapes and sizes. Appendix C provides real-world comparisons for pushing objects on a slippery glass plane, including both the T-shaped object and a cube object. We provide trajectories about flipping a mug in Figure 8 and a cube object in Appendix C.

V. Conclusions

We have presented a method of end-to-end learning physics-informed world models of 3D rigid body dynamics from visual observations. Our method is able to identify the 3D physics parameters critical to non-prehensile manipulations with few-shot and task-agnostic interaction trajectories. To realize Sim2Real transfer, we turn the identified digital twin to a group of physics-aware digital cousins through perturbing the physics and rendering parameters around the identified mean values. Experiments demonstrate the robustness and effectiveness of our method when compared with different types of baseline methods.

Limitation and future work: We see several opportunities for future research. First, we use visual observations to guide the optimization of physical parameters, thus rendering alignment places a key role here. We find that the various shadows generated with the robot's movement can distort rendering loss estimation, compromising the accuracy of learned physical properties. This issue could potentially be resolved by incorporating differentiable relighting [22] into Gaussian Splatting to better model lighting conditions. Additionally, our current framework focus on rigid-body dynamics, and it would be interesting to explore ways to integrate more advanced differentiable physics engines, such as the Material Point Method (MPM) [46], to extend PIN-WM's capability to handle deformable objects. We are also engaged in applying PIN-WM to real-world applications in industrial automation [70, 82, 83].

VI. Acknowledgements

This work was supported in part by the NSFC (62325211, 62132021, 62322207), the Major Program of Xiangjiang Laboratory (23XJ01009), Key R&D Program of Wuhan (2024060702030143), Shenzhen University Natural Sciences 2035 Program (2022C007), and Guangdong Laboratory of Artificial Intelligence and Digital Economy Open Research Fund (GML-KF-24-35).

References

[1] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Suenderhauf. Physically embodied gaussian splatting: A visually learnt and physically grounded 3d representation for robotics. In *Conference on Robot Learning*, 2024.

[2] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *IEEE Conference on Computer Vision and Pattern Recognition*, 2025.

[3] Srinivas Akella and Matthew T Mason. Posing polygonal objects in the plane by pushing. *The International Journal of Robotics Research*, 1998.

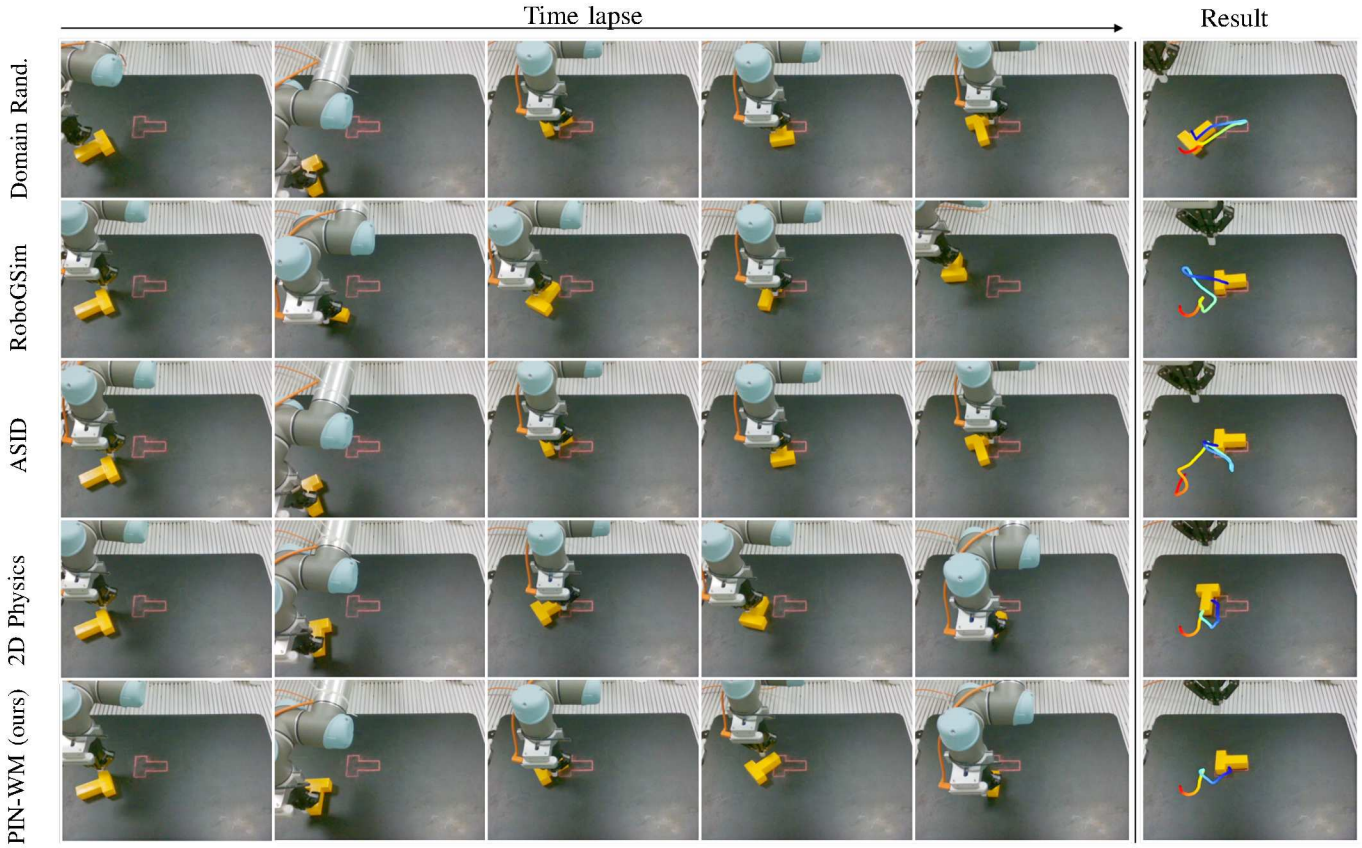

Fig. 6: Real-world trajectories of different methods on the *push* task.

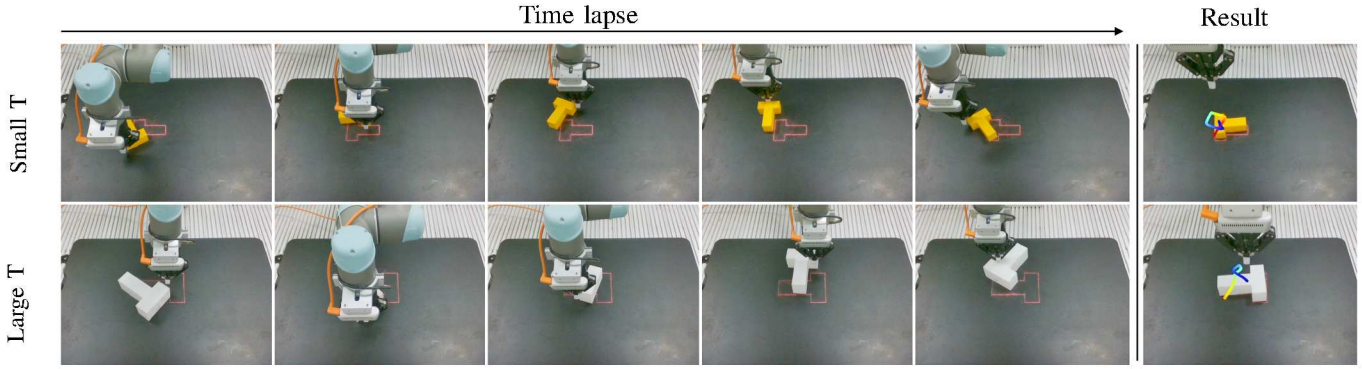

Fig. 7: Real-world trajectories of pushing T-shaped objects of different sizes obtained by our method.

[4] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, 2017.

[5] Fabian Baumeister, Lukas Mack, and Joerg Stueckler. Incremental few-shot adaptation for non-prehensile object manipulation using parallelizable physics simulators. *arXiv preprint arXiv:2409.13228*, 2024.

[6] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *Advances in Neural Information Processing Systems*, 2024.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

[8] Junyi Cao, Shanyan Guan, Yanhao Ge, Wei Li, Xiaokang Yang, and Chao Ma. Neuma: Neural material adaptor for visual grounding of intrinsic dynamics. In *Advances in Neural Information Processing Systems*, 2024.

[9] Rongsen Chen, Junhong Zhao, Fang-Lue Zhang, Andrew Chalmers, and Taehyun Rhee. Neural radiance fields for dynamic view synthesis using local temporal priors. In *Computational Visual Media*, 2024.

[10] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards

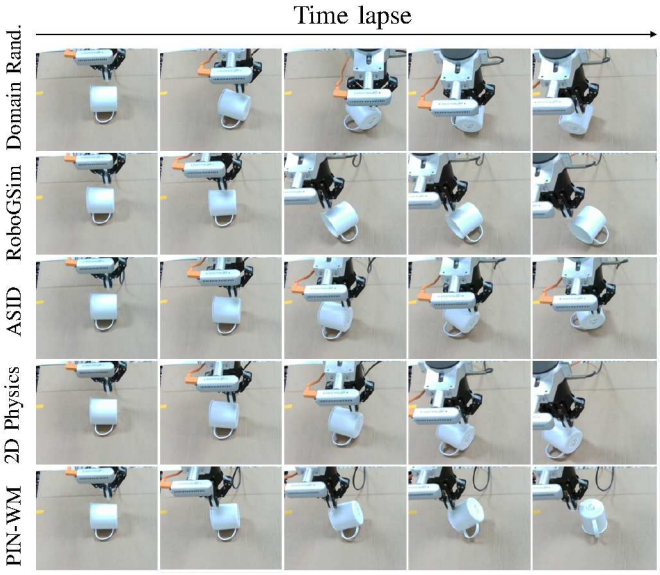

Fig. 8: Real-world trajectories of different methods on the *flip* task.

human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

[11] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2023.

[12] Ignasi Clavera, David Held, and Pieter Abbeel. Policy transfer via modularity and reward guiding. In *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[13] Michael Bradley Cline. *Rigid body simulation with contact and constraints*. PhD thesis, University of British Columbia, 2002.

[14] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

[15] Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. In *Conference on Robot Learning*, 2024.

[16] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in Neural Information Processing Systems*, 2018.

[17] Mehmet Remzi Dogar and Siddhartha S Srinivasa. A framework for push-grasping in clutter. In *Robotics: Science and Systems*, 2011.

[18] Henrik Ebel, Daniel Niklas Fahse, Mario Rosenfelder, and Peter Eberhard. Finding formations for the non-prehensile object transportation with differentially-driven mobile robots. In *Symposium on Robot Design, Dynamics and Control*, 2022.

[19] Ben Evans, Abitha Thankaraj, and Lerrel Pinto. Context is everything: Implicit identification for dynamics adaptation. In *International Conference on Robotics and Automation*, 2022.

[20] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[21] Juan Del Aguila Ferrandis, Joao Moura, and Sethu Vijayakumar. Learning visuotactile estimation and control for non-prehensile manipulation under occlusions. In *Annual Conference on Robot Learning*, 2024.

[22] Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing. In *European Conference on Computer Vision*, 2024.

[23] Radian Gondokaryono, Mustafa Haiderbhai, Sai Aneesh Suryadevara, and Lueder A Kahrs. Learning nonprehensile dynamic manipulation: Sim2real vision-based policy with a surgical robot. *IEEE Robotics and Automation Letters*, 2023.

[24] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, 2017.

[25] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[26] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[27] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[28] Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

[29] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

[30] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[31] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *IEEE International Conference on Robotics and Automation*, 2021.

[32] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

[33] Yingdong Hu, Fanqi Lin, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *Workshop on X-Embodiment Robot Learning*, 2024.

[34] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger,

and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024.

[35] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 2017.

[36] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

[37] Xinyi Jing, Qiao Feng, Yu-Kun Lai, Jinsong Zhang, Yuanqiang Yu, and Kun Li. State: Learning structure and texture representations for novel view synthesis. *Computational Visual Media*, 2023.

[38] Xinyi Jing, Tao Yu, Renyuan He, Yukun Lai, and Kun Li. Frnerf: Fusion and regularization fields for dynamic view synthesis. *Computational Visual Media*, 2024.

[39] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, 2018.

[40] Rama Krishna Kandukuri, Michael Strecke, and Joerg Stueckler. Physics-based rigid body object tracking and friction filtering from rgb-d videos. In *International Conference on 3D Vision*, 2024.

[41] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[42] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023.

[43] Keisuke Kinoshita and Michael Lindenbaum. Robotic control with partial visual information. *International Journal of Computer Vision*, 2000.

[44] Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.

[45] Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator. *arXiv preprint arXiv:2411.11839*, 2024.

[46] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming C. Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *International Conference on Learning Representations*, 2023.

[47] Zhehao Li, Qingyu Xu, Xiaohan Ye, Bo Ren, and Ligang Liu. Difffr: Differentiable sph-based fluid-rigid coupling for rigid body control. *ACM Transactions on Graphics*, 2023.

[48] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019.

[49] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[50] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 1986.

[51] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 2012.

[52] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, 2020.

[53] Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Dieter Fox, and Abhishek Gupta. ASID: Active exploration for system identification in robotic manipulation. In *International Conference on Learning Representations*, 2024.

[54] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. In *Robotics: Science and Systems*, 2023.

[55] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 2022.

[56] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 2021.

[57] Tai-Jiang Mu, Hao-Xiang Chen, Jun-Xiong Cai, and Ning Guo. Neural 3d reconstruction from sparse views using geometric priors. *Computational Visual Media*, 2023.

[58] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 2022.

[59] J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2021.

[60] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE*

International Conference on Robotics and Automation, 2018.

[61] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[62] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, 2021.

[63] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

[64] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[65] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer, 2004.

[66] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[67] Changkyu Song and Abdeslam Boularias. Learning to slide unknown objects with differentiable physics simulations. In *Robotics: Science and Systems*, 2020.

[68] Michael Strecke and Joerg Stueckler. Diffsdfsim: Differentiable rigid-body dynamics with implicit shapes. In *International Conference on 3D Vision*, 2021.

[69] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[70] Bingjie Tang, Iretiayo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S. Sukhatme, Fabio Ramos, and Yashraj S. Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024.

[71] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[72] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2024.

[73] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 2017.

[74] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, 2022.

[75] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *Computational Visual Media*, 2024.

[76] Guo-Wei Yang, Zheng-Ning Liu, Dong-Yang Li, and Hao-Yang Peng. Jnerf: An efficient heterogeneous nerf model zoo based on jittor. *Computational Visual Media*, 2023.

[77] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. In *Conference on Robot Learning*, 2021.

[78] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *IEEE International Conference on Intelligent Robots and Systems*, 2016.

[79] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 2020.

[80] Weihao Yuan, Johannes A Stork, Danica Kragic, Michael Y Wang, and Kaiyu Hang. Rearrangement with nonprehensile manipulation using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2018.

[81] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *IEEE International Conference on Computer Vision*, 2019.

[82] Hang Zhao, Zherong Pan, Yang Yu, and Kai Xu. Learning physically realizable skills for online packing of general 3d shapes. *ACM Transactions on Graphics*, 2023.

[83] Hang Zhao, Juzhan Xu, Kexiong Yu, Ruizhen Hu, Chenyang Zhu, and Kai Xu. Deliberate planning of 3d bin packing on packing configuration trees. *arXiv preprint arXiv:2504.04421*, 2025.

[84] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[85] Jiaji Zhou, Yifan Hou, and Matthew T Mason. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *The International Journal of Robotics Research*, 2019.

[86] Wenxuan Zhou, Bowen Jiang, Fan Yang, Chris Paxton, and David Held. Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation. In *Conference on Robot Learning*, 2023.

APPENDIX

A. Implementation Details for Baselines

All the baselines are implemented carefully to ensure fair comparison. We used the official implementations with default hyperparameters for Diffusion Policy [11], 2D Physics [67], and Dreamer V2 [27]. A history of recent states and actions is used as input for the "Domain Rand + $\mathcal{I}$" (denoted as DR) baseline [60]. All RL-based policies are trained using PPO [66], with the same model architecture, reward function, hyperparameters, and stopping criterion based on the success rate. The reward signal for policy learning is a handcrafted function to encourage the robot to push the object toward the target pose: $r = -d_t - d_r$, where d_t and d_r refer to the translation distance and rotation distance, respectively. Diffusion Policy is trained with successful trajectories collected from expert policies trained in the environment with GT physical parameters, without any randomization.

B. More Experimental Results

1) Superiority over uniform randomization: The only difference between our method and the DR baseline is the different ranges of physics parameters that are used for domain randomization, where DR uses a large range $\mathbf{R}$ to ensure that it covers the target parameters $\boldsymbol{\theta}^\dagger$ while our method uses a much smaller range around the learned parameters $\boldsymbol{\theta}^*$. The low performance of DR is mainly due to the large range of $\mathbf{R}$ and the performance can be improved by shrinking the range around $\boldsymbol{\theta}^\dagger$ (even though this is not feasible in real application scenarios), and using GT $\boldsymbol{\theta}^\dagger$ directly can obtain the best result, as presented in Table IV.

TABLE IV: Success rates across different ranges of randomization.

Tasks	GT	PIN-WM w/ PADC	DR ($\mathbf{R}$/4)	DR ($\mathbf{R}$/2)	DR ($\mathbf{R}$)
Push	**98%**	97%	78%	56%	33%
Flip	**89%**	83%	61%	43%	32%

2) Identified physical parameters: System identification is inherently ill-posed, as multiple parameter sets can explain the same observations. To fairly compare the accuracy across different methods, we estimate one parameter at a time while keeping the others fixed at their GT values. Since inertia is typically represented as a 3×3 matrix in simulation, we do not conduct this experiment. Our method consistently achieves the best performance, as shown in Table V.

TABLE V: Identified physical parameters.

Methods	*Push*			*Flip*		
	Friction	Mass (kg)	Restitution	Friction	Mass (kg)	Restitution
GT	3.00×10^{-2}	1.00	0.00	2.00×10^{-1}	1.00	0.00
ASID	4.45×10^{-2}	0.75	1.12×10^{-2}	2.59×10^{-1}	1.36	1.12×10^{-2}
2D Physics	–	19.25	–	–	29.60	–
PIN-WM	$\mathbf{3.05 \times 10^{-2}}$	**0.76**	$\mathbf{4.18 \times 10^{-4}}$	$\mathbf{2.11 \times 10^{-1}}$	**1.19**	$\mathbf{1.69 \times 10^{-5}}$

3) Robustness to geometry noise: Geometry noise will affect collision detection accuracy and, consequently, system identification. To evaluate this, we conduct experiments on noisy inputs, in which the noise conforms to a Gaussian distribution with a mean of 0 and a variance of $\sigma\%L$, where $\sigma \in \{0, 0.5, 1.0, 3.0\}$ and L is the length of object's bounding box diagonal. We report one-step error of the predicted trajectories after applying the same random actions to the object. As shown in Table VI, while the prediction errors increase with higher noise, our method remains robust even at $\sigma = 3.0$, outperforming ASID [53] using perfect geometry.

TABLE VI: One-step errors across different noise levels.

Methods	*Push*		*Flip*	
	Trans.	Ori.	Trans.	Ori.
PIN-WM ($\sigma = 0.0$)	$\mathbf{1.7 \times 10^{-2}}$	$\mathbf{1.3 \times 10^{-1}}$	$\mathbf{1.4 \times 10^{-2}}$	$\mathbf{2.7 \times 10^{-1}}$
PIN-WM ($\sigma = 0.5$)	2.1×10^{-2}	2.1×10^{-1}	4.2×10^{-2}	9.6×10^{-1}
PIN-WM ($\sigma = 1.0$)	2.3×10^{-2}	1.6×10^{-1}	4.2×10^{-2}	9.5×10^{-1}
PIN-WM ($\sigma = 3.0$)	2.9×10^{-2}	3.0×10^{-1}	7.7×10^{-2}	1.5
ASID	3.0×10^{-2}	4.0×10^{-1}	1.4×10^{-1}	1.6

C. Further Real-World Evaluations

We further validate the effectiveness of PIN-WM in identifying real-world physical parameters. We push a T-shaped object and a cube object on a slippery glass plane. Even small touches cause noticeable displacement, which poses higher demands on the robot's control precision. The smaller mass and volume of the cube make it harder to manipulate. Real-world trajectories are provided in Figures 9 and 10, separately. PIN-WM demonstrates strong performance on both objects, successfully pushing them to the target positions. In contrast, the baseline ASID consistently pushes the objects with excessive force just before reaching the target position, making it difficult to complete the task. We also conduct flip experiments on the small cube, with the trajectories provided in Figure 11. The quantitative results of these experiments are summarized in Table VII.

TABLE VII: Success rate comparisons on different real-world tasks.

Methods	Push T (Slippery)	Push Cube (Slippery)	Flip Cube
ASID	5%	0%	5%
PIN-WM	**45%**	**40%**	**60%**

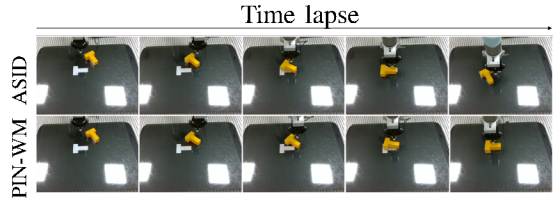

Fig. 9: *Push* T-shaped object on a slippery plane.

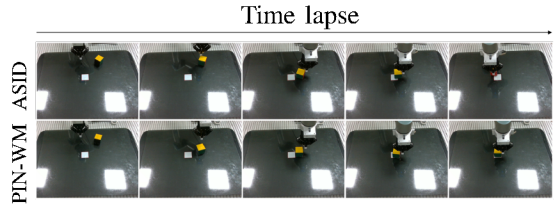

Fig. 10: *Push* cube object on a slippery plane.

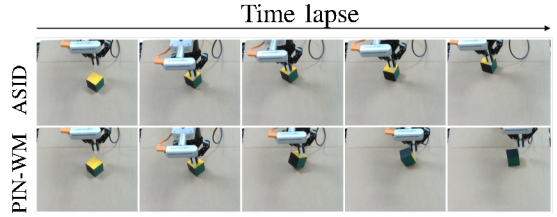

Fig. 11: *Flip* a multicolored cube to change its top-surface color.