DREAM: Differentiable Real-to-Sim-to-Real Engine for Learning Robotic Manipulation

Haozhe Lou[*,1], Mingtong Zhang[*,1], Haoran Geng[2], Hanyang Zhou[1], Sicheng He[1], Zhiyuan Gao[1], Siheng Zhao[1], Jiageng Mao[1], Pieter Abbeel[2], Jitendra Malik[2], Daniel Seita[1], Yue Wang[1]
[1]University of Southern California [2]University of California, Berkeley
[*]equal contribution

***Abstract*—Simulation provides a cost-effective and flexible platform for data generation and policy learning to develop robotic systems. However, bridging the gap between simulation and real-world dynamics remains a significant challenge, especially in physical parameter identification. In this work, we introduce a real-to-sim-to-real framework that leverages the Gaussian Splat representations to build a differentiable engine, enabling object mass identification from real-world visual observations and robot control signals, while enabling manipulation policy learning simultaneously. Through optimizing the mass of the manipulated object, our method automatically builds high-fidelity and physically plausible digital twins. Additionally, we propose a novel approach to train the force-aware grasping policies from limited data by transferring feasible human demonstrations into simulated robot demonstrations. Through comprehensive experiments, we demonstrate that our proposed framework achieves accurate and robust performance on mass identification across various object geometries and mass values. Those optimized mass values facilitate force-aware policy learning, achieving superior and high performance on object grasping, reducing the sim-to-real gap effectively.**

I. Introduction

Simulation has become an essential platform for robotics, providing a cost-effective and scalable platform that reduces the reliance on extensive robotics expertise. Through reusable and controlled data generation, simulation has driven significant advancements in accelerating policy learning [2, 18, 10, 1, 19, 35]. However, despite these benefits, replicating the visual realism and complex physical dynamics of the real world remains a significant challenge. High-fidelity physical simulations often demand specialized knowledge and complex modeling, which limits the scalability and robustness of simulation-based approaches for real-world deployment.

A long line of research has focused on bridging the sim-to-real gap, which arises when transferring models trained in simulation to real-world configurations. This gap remains a fundamental challenge in robotics. Simulation-based policies typically assume accurate knowledge and modeling of real-world configurations, including underlying physical parameters. However, differences between the estimated geometry and mass from visual observations and their real-world values increase the sim-to-real gap. Existing strategies to mitigate this gap include domain randomization [57, 49, 45], which enhances robustness by varying simulation parameters, and system identification [25, 55, 31], which refines simulation dynamics by calibrating with real-world observations. Advances in simulation fidelity [20, 38] and domain adaptation [6, 48, 11] have further facilitated the transfer of models from simulation to reality in robotics applications. Complementary to these efforts, real-to-sim frameworks attempt to construct digital twins that replicate real-world geometry and dynamics with high precision [12, 59, 29]. Nonetheless, building accurate digital twins typically requires integrating multiple approaches, such as geometric reconstruction and parameter identification. Despite these advances, achieving precise modeling from visual observations remains challenging for current real-to-sim methods.

This challenge is fundamentally tied to the problem of system identification—inferring physical parameters from visual observations to ensure simulated environments faithfully reflect real-world dynamics. Estimating object attributes and system dynamics from images is difficult, even with full system state access. While robust forward simulators [36] exist, their non-differentiability limits applicability to inverse problems. Surrogate gradient methods such as finite differences are commonly used [14, 47, 65], but scale poorly in high-dimensional settings. Recent progress in differentiable simulation improves learning efficiency. In particular, GradSim [27] enables end-to-end differentiation from visual observations to object-level physical parameters. Inspired by this, our work optimizes object mass directly from video, enabling force-aware grasping policy learning conditioned on mass and substantially improving performance.

To address these challenges, we introduce DREAM in this paper, a differentiable real-to-sim-to-real framework that builds our simulation engine upon differentiable simulation [27, 16, 40, 37] and Gaussian Splat representations [30]. This differentiable engine enables object mass identification through visual observations and robot control signals in robot-object interactions. Additionally, we propose a novel learning-based method for dexterous manipulation, where we transfer human demonstrations into simulation-executable robot demonstrations, then utilize the proposed method to optimize the grasp position and force simultaneously.

II. Problem Statement

We focus on the real-to-sim-to-real task, which aims to construct a simulation environment that closely mirrors real-world geometry, physics, and appearance. We assume access to several types of RGB videos: scene-centric video sequences, denoted as $\mathcal{I}_s$, which capture the static environment; object-

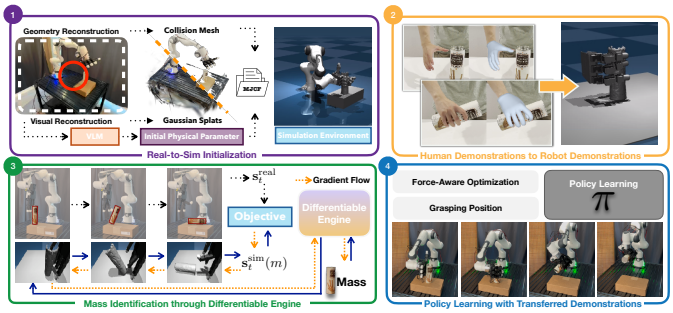

Fig. 1: **Overview of our method.** Our approach consists of four components: (1) Real-to-Sim, (2) Learning from Human Demonstrations, (3) Mass Identification, and (4) Policy Learning. We begin by capturing videos of the scene and human demonstrations. Robotic actions are then executed in both simulation and the real world to identify object mass via our differentiable physics engine. Lastly, a manipulation policy is trained using the demonstrations and identified mass.

centric videos, denoted as $\mathcal{I}_o$, which provide multiple views of the manipulated object to support accurate visual and geometric reconstruction; and human demonstration videos $\{\mathcal{I}_t\}_{t=1}^T$, which illustrate task execution. We also extract object trajectories from real-world robot rollouts, denoted as $\{\mathbf{s}_t^{\text{real}}\}_{t=1}^T$, and simulate corresponding trajectories $\{\mathbf{s}_t^{\text{sim}}\}_{t=1}^T$ within our framework.The scene and object videos ($\mathcal{I}_s, \mathcal{I}_o$) are used to initialize the real-to-sim process via visual and geometric reconstruction. Trajectories from both real and simulated rollouts enable mass identification through a differentiable engine, producing an optimized object mass m that is incorporated as a physical parameter in the simulator. Meanwhile, human demonstration videos $\{\mathcal{I}_t\}_{t=1}^T$ are translated into robot-executable trajectories $\{\mathbf{A}_t\}_{t=1}^T$ using our proposed method and are used to train a force-aware manipulation policy π.

III. Method

We propose a real-to-sim-to-real framework that constructs accurate simulation environments and identifies object mass via system identification using a differentiable engine from visual observations and robot control signals, enabling robust policy learning and sim-to-real transfer. The framework is built on MuJoCo [58], a general-purpose physics simulator, the differentiable engine Brax [16], and Gradsim [27]. It operates in four steps shown in Figure 1: Real-to-Sim (Section III-A), Mass Identification (Section III-B), Learning from Human Demonstrations (Section III-C), and Policy Learning (Section III-D). First, the scene and object are reconstructed from RGB videos $\mathcal{I}_s$ and $\mathcal{I}_o$, capturing static environments and target objects. The output simulation is formalized as $\mathcal{S} = \{\mathcal{K}, \boldsymbol{\theta}\}$ in MJCF format, where $\mathcal{K}$ denotes collision meshes and $\boldsymbol{\theta}$ the physical parameters. Next, the framework executes consistent robotic actions in simulation and the real world, collecting $\{\mathbf{s}_t^{\text{real}}\}_{t=1}^T$ and $\{\mathbf{s}_t^{\text{sim}}\}_{t=1}^T$ to identify object mass m. Third, human demonstrations $\{\mathcal{I}_t\}_{t=1}^T$ are translated into robot-executable trajectories $\{\mathbf{A}_t\}_{t=1}^T$. Finally, these trajectories are used to train a policy in simulation, which is then deployed in the real world directly.

A. Visual and Geometric Reconstruction

Our framework starts with reconstructing high-fidelity visual and geometric models of key elements in the manipulation environment with $\mathcal{I}_s$, including objects, robotic arms, dexterous hands and workspaces. This reconstruction ensures representations of both collision geometry and visual appearance. To integrate these models into differentiable simulation, we adopt a particle-based Gaussian Splat representation [30, 23], which enables photorealistic rendering and high-quality mesh generation for collision detection following [33]. Specifically, we process videos collected from mobile devices to train two ensembles of Gaussian primitives: one for collision geometry and another for visual appearance. Specifically, 2D Gaussian Splats with surface normal estimation [67] provide accurate geometry for simulation, while 3D Gaussian Splats ensure high-fidelity rendering. This process yields two complementary outputs: a collision mesh $\mathcal{K}$ and Gaussian particles $\mathcal{P}$.

B. Physical Identification from Robot-Object Interactions

Accurate identification of physical parameters $\boldsymbol{\theta}$ is essential for constructing physically plausible simulations. We begin by using a Vision-Language Model (VLM) [24] to generate an initial MJCF representation $\mathcal{S}$ from environment images and prompts, inspired by [71]. While this approach provides a reasonable structural prior, parameters inferred solely from visual input often deviate from real-world values due to the lack of observable physical cues [3, 53].

To address this, we focus on identifying the object mass, which is a key parameter in dynamics that can be reliably measured. Accurate mass identification improves simulation fidelity and enables robust policy learning. We choose a planar pushing task with a virtual fulcrum assumption to reduce frictional effects, and optimize mass m to minimize the discrepancy between simulated and real-world trajectories [27]:

$$\min_{m>0} \mathcal{L}_{\text{traj}}(m) := \sum_{t=1}^{T} \left\| \mathbf{s}_t^{\text{sim}}(m) - \mathbf{s}_t^{\text{real}} \right\|_2^2, \tag{1}$$

where $\mathbf{s} = [\mathbf{p}, \mathbf{q}]^\top \in \mathbb{R}^7$ denotes the object's 6-DoF pose, consisting of position $\mathbf{p} \in \mathbb{R}^3$ and orientation represented as a unit quaternion $\mathbf{q} \in \mathbb{R}^4$. Specifically, $\mathbf{s}_t^{\text{real}}$ is obtained by FoundationPose [63] in the real world while $\mathbf{s}_t^{\text{sim}}(\mathbf{m})$ is obtained by executing the same actions in the simulation.

a) Dynamics Modeling: To simulate object motion, we adopt a standard rigid-body formulation of the Newton-Euler mechanism. Let $\mathbf{u}_t = [\mathbf{v}_t, \boldsymbol{\omega}_t]^\top$ denote the object's velocity at timestep t, where $\mathbf{v}_t$ and $\boldsymbol{\omega}_t$ are the linear and angular velocity components, respectively. We express the governing equation as the second order differential equation(ODE) [9]:

$$\mathbf{M}(\mathbf{s}_t, \mathbf{u}_t, m, \boldsymbol{\theta})\, \dot{\mathbf{u}}_t = \mathbf{f}(\mathbf{s}_t, \mathbf{u}_t, \boldsymbol{\theta}), \tag{2}$$

where $\mathbf{M}$ is the mass-inertia matrix [4] and $\mathbf{f}$ collects external and contact forces (3), gravity and torques. We adopt a compliant penalty-based contact model, parameterized by stiffness and damping $(k_e, k_d) \in \boldsymbol{\theta}$, which applies normal forces proportional to penetration depth and contact velocity [58, 15]:

$$\mathbf{f}_n(\mathbf{s}, \mathbf{u}_t, \boldsymbol{\theta}) = -\mathbf{n}\left(k_e\, C(\mathbf{s}) + k_d\, \dot{C}(\mathbf{u})\right), \tag{3}$$

where $\mathbf{f}_n$ is the contact force, $\mathbf{n}$ is the contact normal, $C(\mathbf{s})$ the penetration depth, and $\dot{C}(\mathbf{u})$ is the derivative of $C(\mathbf{s})$. This contact model is differentiable and readily integrated into our simulation framework. In practice, we implement the dynamics using a discrete-time update [15]:

$$\mathbf{s}_{t+1}^{\text{sim}} = G\big(\mathbf{s}_t^{\text{sim}}, \mathbf{u}_t,\, m,\, \mathbf{f}_t\big), \quad t = 0, \ldots, T-1. \tag{4}$$

Here, $\mathbf{f}_t$ denotes the external forces at timestep t, including actuator impulses, gravity, and object-ground contacts.

b) Differentiable Physics: To optimize (1), we compute gradients of the simulated trajectory with respect to the object mass m. Following the discrete adjoint method from [27], we adopt a semi-implicit Euler integration scheme for stability under contact dynamics. We couple kinematics from MjX/Brax [16] with rigid-body dynamics (2) and the contact model (3), forming a differentiable computation graph [21].

c) Semi-Implicit Euler Modeling: The update function $G(\cdot)$ in (4) is implemented using a semi-implicit Euler integration scheme:

$$\begin{aligned} &G\Big(\big[\mathbf{s}_t,\, \mathbf{u}_t\big], m,\, \boldsymbol{\theta}\Big) \\ &= \begin{bmatrix} \mathbf{s}_t \;+\; \Delta t\, \mathbf{u}_{t+1} \\ \mathbf{u}_{t+1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{s}_t \;+\; \Delta t\Big(\mathbf{u}_t + \Delta t\, \mathbf{M}^{-1}(\mathbf{s}_t, \mathbf{u}_t, m, \boldsymbol{\theta})\, \mathbf{f}\big(\mathbf{s}_t,\, \mathbf{u}_t\big)\Big) \\ \mathbf{u}_t + \Delta t\, \mathbf{M}^{-1}(\mathbf{s}_t, \mathbf{u}_t, m, \boldsymbol{\theta})\, \mathbf{f}\big(\mathbf{s}_t,\, \mathbf{u}_t\big) \end{bmatrix} \end{aligned} \tag{5}$$

where Δt is the integration timestep, and $f(\cdot)$ encapsulates both external and contact forces.

d) Differentiable Real-to-Sim-to-Real Optimization: We simulate the system starting from the initial condition $\mathbf{s}_0^{\text{sim}}$, and iteratively update the state via (4). To quantify the discrepancy between simulated and real-world trajectories, we define the trajectory loss between the simulated state $\mathbf{s}_t^{\text{sim}}$ and the corresponding real-world state $\mathbf{s}_t^{\text{real}}$ as:

$$\mathcal{L}_{\text{traj}}(m) = \sum_{t=0}^{T} \left\| \mathbf{s}_t^{\text{sim}} - \mathbf{s}_t^{\text{real}} \right\|_2^2. \tag{6}$$

This objective encourages the simulated trajectory, parameterized by mass m, to closely match the observed real-world dynamics over time. The gradient $\nabla_m \mathcal{L}_{\text{traj}}(m)$ is computed via automatic differentiation, using backpropagation as implemented in PyTorch [42], as follows:

$$\frac{\partial \mathcal{L}_{\text{traj}}}{\partial m} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_{\text{traj}}}{\partial \mathbf{s}_t^{\text{sim}}} \cdot \frac{\partial \mathbf{s}_t^{\text{sim}}}{\partial M_t} \cdot \frac{\partial \mathbf{M}_t}{\partial m}. \tag{7}$$

Unlike system identification methods such as GradSim [27], which rely on manually specified external forces, our approach supports end-to-end optimization by directly leveraging consistent robotic control signals in both simulation and the real world to model the external forces applied to the object. This creates a tight coupling between real-world and simulated trajectories, enabling us to capture contact dynamics through robot-object interactions.

Importantly, our method does not require ground-truth object mass or contact points. Object geometry and poses are obtained via Section III-A, while actuator signals and robot-object interactions are derived from the MJCF kinematic model. These serve as inputs to our differentiable framework for accurate mass optimization.

C. Transferring Human Demonstrations to Robot

After accurately modeling the scene and object, the next step is to collect real-world human demonstrations and transfer them into robot demonstrations for policy learning. Although learning directly from human demonstrations is intuitive, substantial differences between human and robotic hands complicate grasp interaction transfer, particularly due to varied object geometries and masses.

Our approach aims to transform human demonstrations captured from RGB video sequences $\{\mathcal{I}_t\}_{t=1}^{T}$ into executable robotic demonstrations within the simulation. Each video frame $\mathcal{I}_t$ is processed using HaMeR [44] and MCC-HO [64] to reconstruct detailed articulated models of the human hand and the manipulated object. At each timestep t, these methods output:

$$\mathbf{h}_t \in \mathrm{SE}(3) \times \mathbb{R}^{J_h}, \quad \mathbf{o}_t \in \mathrm{SE}(3), \tag{8}$$

where $\mathbf{h}_t$ encodes the 6-DoF wrist pose and finger joint angles (J_h), and $\mathbf{o}_t$ describes the object's 6-DoF pose. Subsequently, we employ Dex-Retargeting [46] to map these human hand-object poses $\mathbf{h}_t, \mathbf{o}_t$ to the robotic hand with J_r degrees of freedom. This produces preliminary robot actions:

$$\mathbf{A}_t = \mathcal{R}(\mathbf{h}_t, \mathbf{o}_t) \in \mathbb{R}^{J_r}, \tag{9}$$

where $\mathbf{A}_t$ represents target joint angles for robotic actuators. Given our assumption that the object geometry remains consistent between human demonstration and robotic manipulation, the resulting action set $\mathbf{A}_t$ directly serves as a data source for our policy learning.

D. Policy Learning with Transferred Robot Demonstrations

We initialize policy learning using robot demonstrations $\{\mathbf{A}_t\}_{t=1}^{T}$ described in Section III-C. Each demonstration maps the reconstructed object's collision mesh vertices $\mathcal{K}$ as inputs to the corresponding robotic grasp pose. These observation-action pairs directly supervise training of the manipulation policy π_ϕ, which maps object-centric observations to dexterous grasp configurations.

a) Grasping Position Learning.: To capture an object's geometry and pose, we encode the vertices of its collision mesh $\mathcal{K}$ using positional encoding [56], forming the input to our policy. This policy conditions the observation $\mathbf{o}$ on the reconstructed collision mesh $\mathcal{K}$ and the identified mass m. Concretely, π_ϕ is a multi-head neural network that predicts dexterous hand joint positions $\hat{\mathbf{A}}$, contact-related rewards $\hat{\mathbf{r}}$, and a mass-related control force $\hat{\mathbf{f}}$.

$$\pi_\phi(\mathbf{o}) = \begin{bmatrix} \hat{\mathbf{A}} \\ \hat{\mathbf{r}} \\ \hat{\mathbf{f}} \end{bmatrix} \in \mathbb{R}^{19}, \quad \hat{\mathbf{f}} \;=\; \frac{m \cdot g}{n_{\text{active}}}. \tag{10}$$

where $\hat{\mathbf{A}} \in \mathbb{R}^{16}$ denotes the predicted joint positions, $\hat{\mathbf{r}} \in \mathbb{R}^2$ represents the contact constraint, and $\hat{\mathbf{f}} \in \mathbb{R}$ denotes the grasping force constraint. The variable n_{active} indicates the number of active contacts between the robotic hand and the object. The network uses fully connected layers with ReLU activations, followed by a pooling layer.

b) Force-Aware Optimization Design: At training onset, we define a two-dimensional contact constraint $\hat{\mathbf{r}}$: one term encourages sustained contact during the rollout, the other ensures object retention at the end. The policy dynamically influences this constraint based on hand–object interactions through our simulation engine and the simulation asset S from Section III-A and III-B. It remains high when active contact points exceed a threshold $N_{\min}$ over the time horizon H:

$$\forall t \in [t_0, t_0 + H] : n_{\text{active}}(t) \geq N_{\min},$$
$$\mathbb{I}_{\text{in\_hand}}(t) = \begin{cases} 1, & \text{if } n_{\text{active}}(t) \geq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

We subsequently retrain the manipulation policy with the force-based constraint in 10, enabling adaptive force control that responds to object mass variations [60, 69]. This enhances the robustness of grasp poses learned from demonstrations, ensuring stability under diverse dynamics.

Traditional position-based policies replicate grasp poses from human demonstrations [43, 34, 7, 61, 62, 51] but overlook unobserved forces, particularly those countering gravity. Applying uniform forces across varying object masses often leads to instability. To address this, we propose a hybrid control framework that combines position and force control, using a prediction module conditioned on the optimized mass m. This jointly optimizes the policy parameters ϕ and grasping force, enabling more robust and physically grounded manipulation.

IV. Experiment

A. Mass identification

We evaluate our mass identification method through object pushing experiments by applying identical actions in both the real world and simulation. The resulting trajectories are used to optimize object mass via our differentiable engine, as detailed in Section III-B. We assess the performance in two settings: (1) across objects with varying geometries, and (2) across replicas with identical geometry but different internal densities. Our method accurately recovers mass in both cases, demonstrating generalization to diverse shapes and sensitivity to subtle physical differences. As shown in Table II, mass is accurately identified with deviations under 13 grams, confirming the effectiveness of estimating physical parameters independent of geometry.

B. Tabletop Object Grasping Experiments

We compare our grasping policy against two baselines across various objects: (1) DexGraspNet2.0 [70], trained on large-scale simulation datasets, and (2) Human2Sim2Robot [34], a recent real-to-sim-to-real method that learns dexterous manipulation policies from RGBD videos of human demonstrations. All use the collision mesh $\mathcal{K}$ generated by our real-to-sim framework as input.

Object	Letter U	Letter A	Lego	Domino	Cookie	Ketchup
Inferred Mass (g)	500	500	300	800	500	1000
Identified Mass (g)	110	145	53	117	200	667
Ground Truth Mass (g)	125	134	59	106	210	726
Percentile Error (%)	12.0	9.0	8.6	9.3	4.8	8.1

TABLE I: Mass identification across diverse objects with varying shapes, sizes, and mass scales. The inferred mass is obtained from VLM as described in Section III-B.

Density	ρ_1	ρ_2	ρ_3
Identified Mass (g)	95	129	207
Ground Truth Mass (g)	82	125	218

TABLE II: Mass identification across objects with identical geometry but varying densities.

As shown in Figure 2, our method consistently outperforms baselines across eight objects with diverse geometries and mass properties, demonstrating the robustness of our approach. While baseline performance tends to degrade on heavier objects, our method maintains high success rates with lower variance across all cases. These results show the effectiveness of our force-aware optimization in enabling stable and reliable grasping across a wide range of object characteristics.

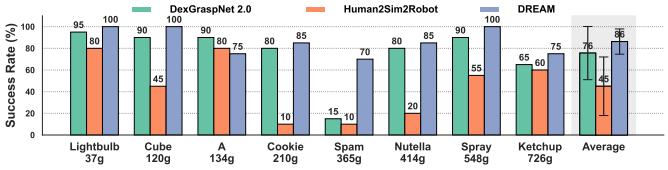

Fig. 2: **Quantitative Results of Grasping Policies.** Grasp success rates across eight objects with varying geometries and mass values, with the average and standard deviation of each method.

Figure 3 presents qualitative results of our force-aware grasping policy across a range of objects. The top row captures the motion leading to the pre-grasp pose, while the bottom row displays the resulting post-grasp configurations. These examples demonstrate the policy's ability to consistently achieve stable and secure grasps under varying object geometries and mass values.

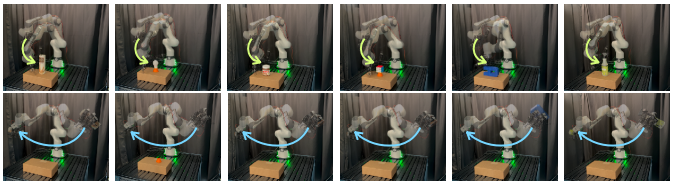

Fig. 3: **Qualitative Results of Our Policy.** We evaluate our force-aware grasping policy across various objects. The first row illustrates the approach to the pre-grasp pose, while the second row shows two post-grasp positions, demonstrating that the policy achieves stable, secure grasps.

V. Conclusion

DREAM is a differentiable real-to-sim-to-real framework that leverages differentiable physics simulation to create visually realistic and physically accurate digital twins from visual observations and robot control signals, enabling robust dexterous manipulation policies.

VI. Appendices

A. Additional Results

1) Mass Identification: Our method accurately recovers mass in both cases, demonstrating generalization to diverse shapes and sensitivity to subtle physical differences. The objects we used for mass identification are shown in Figure 4.

Fig. 4: **Objects for Mass Identification.** We conduct experiments on mass identification across diverse object geometries and identical geometries with varying densities. Our method accurately estimates mass in both settings, demonstrating robustness to shape and density variations.

As shown in Figure 5, simulations using the optimized mass closely match real-world object dynamics, while those using an incorrect lighter mass deviate significantly. This demonstrates that accurate mass identification improves both the physical realism and visual quality of simulated rollouts.

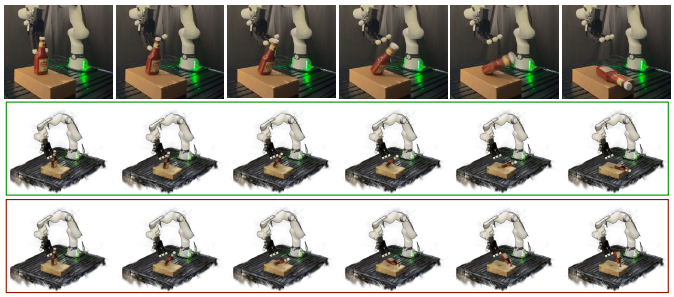

Fig. 5: **Quantitative Results of Mass Identification.** We show the real-world object pushing (top) and render object trajectories using Gaussian Splats: simulated with optimized mass (middle), and simulated with a lighter mass (bottom), all using the same robot actions. The optimized mass closely reproduces real-world dynamics, reducing the sim-and-real gap with high visual fidelity.

2) Effectiveness of Force-Based Control through Grasping: In our grasping experiments, we evaluate how incorporating force-based constraints conditioned on object mass influences sim-to-real performance. This setup highlights the need for mass-aware force control and demonstrates the impact of accurate mass identification on policy success.

We first evaluate our grasping policy on three objects that share identical geometry and demonstrations but differ in mass. Each policy is trained with a specific object mass to assess the impact of mass-aware force control. As shown in Figure 8, policies perform well only when training and evaluation masses matched: the medium-mass policy succeeds on the medium object but fails on the heavier and lighter ones due to under- and over-applied force, respectively. Mass mismatches likewise lead to unstable grasps for the other two policies. Table 6 confirms this trend, with the highest success rate (80%) on the training mass, while performance drops to 40% and 30% on mismatched cases. These results highlight the importance of accurate mass conditioning for robust, reliable grasping.

Train \ Eval	ρ_1	ρ_2	ρ_3
ρ_1	**75%**	30%	15%
ρ_2	40%	**80%**	30%
ρ_3	15%	40%	**95%**

Fig. 6: Cross-evaluation of grasping policies trained on different object densities and evaluated across varying masses. Each cell shows the grasp success rates. Policies perform well only when the training and evaluation masses match.

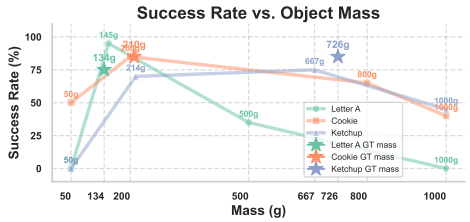

Fig. 7: Grasping success rates across three objects with different mass values.

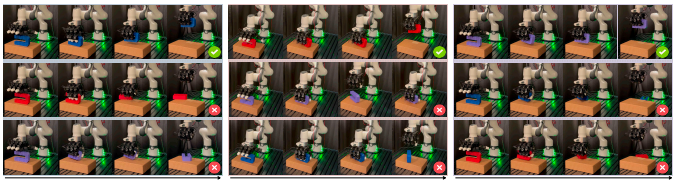

Fig. 8: **Qualitative Results.** Left to right: policies trained on medium, light, and heavy objects. Only the mass-matched policy achieves stable grasps, while mismatched ones fail due to excessive or insufficient force, causing bounce-off for lighter objects or slippage for heavier ones.

Second, we evaluate whether policies conditioned on automatically identified mass can match the performance of those trained with object mass. As shown in Figure 7, success rates consistently peak at either the ground-truth or identified mass. Notably, policies using identified mass often match or even exceed those using ground-truth values, while substantially outperforming policies conditioned on arbitrary masses. These results underscore the effectiveness of our mass identification approach in enabling robust, force-aware grasping without requiring access to true mass values.

These experiments demonstrate that accurate mass is essential for effective force-aware grasping. Policies trained with object mass consistently outperform those trained on mismatched masses, and policies conditioned on automatically identified mass achieve comparable performance to those using ground-truth values. Together, these results validate our mass identification framework as a practical and reliable solution for enabling robust grasping without prior knowledge of object mass.

B. Preliminaries

This paper aims to accurately reconstruct the physical process of human hand grasping using only visual observations and robot control signals, without requiring access to ground-truth physical parameters. Our approach is grounded in two key components. The first is a differentiable, particle-based physics simulation framework [16], which enables gradient-based optimization of physical properties such as object mass. The second is a Real-to-Sim reconstruction pipeline based on Gaussian Splatting [33], which allows us to build photorealistic and spatially consistent 3D scenes from video input. By combining these two components, we construct a fully differentiable pipeline that bridges real-world perception and physical simulation, supporting accurate modeling of dynamic hand-object interactions and enabling robust policy learning in simulation.

Robotic simulation engines such as MuJoCo [58], Isaac Sim [38], and GradSim [26, 17] are fundamentally built upon the Lagrangian formulation of mechanics [32], which models the evolution of physical systems by tracking a fixed set of particles or reference points through space and time. This approach assumes a consistent and predefined structure in the simulation environment, typically described using formats such as MJCF or URDF. These configurations specify the number and arrangement of system components, such as joints, links, and actuated elements, which remain constant throughout the simulation. At each discrete timestep, the state of every object is updated based on dynamic and kinematic equations that reflect the physical principles embedded in the simulation engine. As a result, the evolution of object poses, velocities, and contact interactions is governed by the engine's internal numerical solvers and integration schemes. This structured and physics-informed representation is crucial for accurately modeling force transmission, contact behavior, and motion in robotic manipulation scenarios.

a) Differentiable Physics.: A foundational assumption of our framework is that once the static scene reconstruction is completed, the physical configuration of the environment remains unchanged throughout the system identification and policy training stages. That is, no additional objects or robots are introduced to, nor are existing components removed from, either the simulation environment or the real-world scene. Consequently, the states of all entities captured during the observation phase remain consistent and are used directly for deploying control signals in both simulation and real-world execution. This guarantees the fidelity of simulation rollouts and the alignment of dynamics between domains.

Our system architecture is governed by two fundamental categories of equations. The first involves *kinematic equations* [13], which model the articulated motion of the robotic arm and hand, accounting for joint angles, velocities, and end-effector trajectories. These equations underpin the robot's ability to reach and manipulate objects in a controlled fashion. The second set comprises *dynamic equations* [26], which govern the interactions between the object, the robotic hand, and the supporting surface (e.g., table). These dynamics describe the forces and torques that arise during contact, enabling accurate simulation of object responses.

To simulate and optimize object behavior, we employ a dual-engine architecture consisting of MJX (the JAX-based backend of Brax) and GradSim. For spatial representation within the differentiable physics engine, we use the object's mesh vertices as the fundamental particles. These vertices serve as geometric and physical descriptors that enable fine-grained modeling of object-hand and object-environment interactions.

MJX is used to model robot kinematics and extract detailed contact information during simulation rollouts. It provides precise contact points, surface normals, and force vectors arising from interactions with the robotic hand. This information is crucial for establishing accurate boundary conditions for system identification and subsequent policy learning.

In parallel, GradSim [26] offers a PyTorch-based framework for gradient-based simulation of object dynamics. It models the effects of gravity, inertial forces, and external perturbations (such as pushes from the robot or collisions with the ground), enabling smooth gradient flow through time. This setup facilitates efficient mass parameter optimization and supports end-to-end training pipelines involving both perception and control.

A key assumption in our setup is that the relative poses between the object, the ground, and the robotic hand within the simulation closely approximate those in the real world. This alignment is critical to ensure that simulated contact events reflect real-world conditions, enabling high-fidelity modeling of physical interactions. To this end, we align object placement using estimated poses obtained from visual tracking pipelines such as FoundationPose, ensuring consistent coordinate frames.

Although our framework incorporates Position-Based Dynamics (PBD) [39] for stability and efficiency, we introduce tailored modifications to enhance collision detection and contact resolution. Specifically, we refine the broad-phase collision detection algorithm to better handle high-resolution meshes and non-convex geometries. This is essential for accurately modeling complex objects with fine surface detail and for ensuring robust gradient propagation during contact-rich interactions.

By combining MJX's strengths in kinematic modeling and contact extraction with GradSim's gradient-based physical simulation, our framework enables end-to-end mass identification and force-aware policy training. These capabilities lay the foundation for accurate and generalizable robotic manipulation in real-world settings, bridging the sim-to-real gap through physically grounded learning.

1) Particle-Based Physics Simulation: Particle-based physics simulation is extensively used in computational physics and graphics for modeling dynamic behaviors of objects [28]. Unlike traditional methods that rely on continuous volumes or polygonal meshes, particle-based methods discretize objects into numerous discrete particles,

each endowed with physical attributes such as mass m_i, position $\mathbf{x}_i$, and velocity $\mathbf{v}_i$, as well as material properties including elasticity, friction, and damping. This discrete representation allows the efficient and realistic simulation of complex behaviors, especially beneficial in scenarios involving deformable or fragmented objects, fluids, and granular materials.

The center of mass $\mathbf{COM}$ for a particle-based system can be computed by:

$$\mathbf{COM} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i}. \tag{12}$$

The inertia tensor $\mathbf{I}$, which describes an object's resistance to rotational acceleration, is computed relative to the center of mass as:

$$\mathbf{I} = \sum_i m_i \left[\|\mathbf{r}_i\|^2 \mathbf{E} - \mathbf{r}_i \mathbf{r}_i^\top \right], \quad \text{where} \quad \mathbf{r}_i = \mathbf{x}_i - \mathbf{C} \tag{13}$$

with $\mathbf{E}$ denoting the identity matrix.

a) Position-Based Dynamics (PBD).: Position-Based Dynamics (PBD) is a widely adopted paradigm in real-time and interactive physics simulation due to its stability, simplicity, and efficiency in handling constraint-driven dynamics [5]. Unlike traditional force-based methods that compute motion by integrating forces and torques explicitly, PBD enforces physical consistency by iteratively projecting particle positions to satisfy a set of predefined geometric and physical constraints. This projection-based formulation naturally accommodates large simulation time steps, making it particularly suitable for high-speed applications such as robotic manipulation and interactive environments.

Prediction Step (Implicit Integration). The simulation begins by predicting particle states using semi-implicit Euler integration, which offers numerical stability and reduces oscillations during stiff interactions. For each particle i, the translational motion is computed as:

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \Delta t \frac{\mathbf{f}_i^t}{m_i}, \tag{14}$$

$$\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i^t + \Delta t \, \mathbf{v}_i^{t+\Delta t}, \tag{15}$$

where $\mathbf{f}_i^t$ is the external force (e.g., gravity or contact impulses), m_i is the particle mass, and Δt is the simulation timestep. For rigid-body components, angular motion is predicted using:

$$\boldsymbol{\omega}_i^{t+\Delta t} = \boldsymbol{\omega}_i^t + \Delta t \, \mathbf{I}_i^{-1} \left(\boldsymbol{\tau}_i^t - \boldsymbol{\omega}_i^t \times (\mathbf{I}_i \boldsymbol{\omega}_i^t) \right), \tag{16}$$

$$\mathbf{q}_i^{t+\Delta t} = \mathbf{q}_i^t + \frac{\Delta t}{2} \tilde{\boldsymbol{\omega}}_i^{t+\Delta t} \mathbf{q}_i^t, \tag{17}$$

where $\mathbf{I}_i$ is the inertia tensor, $\boldsymbol{\tau}_i^t$ is the external torque, and $\mathbf{q}_i^t$ is the orientation represented as a unit quaternion. Here, $\tilde{\boldsymbol{\omega}}_i = [0, \boldsymbol{\omega}_i^\top]^\top$ embeds angular velocity into the quaternion algebra.

Constraint Projection Step. Once predicted states are available, positional constraints are enforced through iterative corrections. Each constraint $C(\mathbf{x}_i, \mathbf{q}_i) \geq 0$ represents a physical requirement (e.g., no interpenetration, fixed distances, volume preservation) and is resolved using a gradient-based position correction scheme. For constraint satisfaction, the positional update is computed as:

$$\Delta \mathbf{x}_i = -\lambda \frac{1}{m_i} \nabla_{\mathbf{x}_i} C(\mathbf{x}_i), \quad \text{with} \quad \lambda = \frac{C(\mathbf{x}_i)}{\sum_j \frac{1}{m_j} \|\nabla_{\mathbf{x}_j} C(\mathbf{x}_j)\|^2}, \tag{18}$$

where the Lagrange multiplier λ ensures physically consistent constraint enforcement. Iterative Gauss-Seidel or Jacobi solvers are used to converge the system to a valid constraint-satisfying configuration.

Velocity Update Step. After the constraints are enforced, particle velocities are updated to reflect the corrected positions:

$$\mathbf{v}_i^{t+\Delta t} \leftarrow \frac{\mathbf{x}_i^{t+\Delta t} - \mathbf{x}_i^t}{\Delta t}. \tag{19}$$

This ensures consistency between position corrections and subsequent dynamics, maintaining momentum while preserving the stability advantages of PBD.

Discussion. The particle-based formulation enables fine-grained spatial resolution and direct manipulation of geometric attributes, which is particularly beneficial for simulating high-DOF robotic hands interacting with rigid, deformable or complex-shaped objects. Furthermore, the implicit treatment of constraints circumvents many of the numerical instabilities associated with stiff force-based models, making PBD highly suitable for differentiable simulation settings where robustness and gradient flow are important.

2) Gaussian Splatting: Gaussian Splatting has emerged as a powerful technique in robotic real-to-sim pipelines for capturing scenes, objects, and backgrounds with high geometric fidelity and photorealistic detail. It enables flexible and efficient modeling of complex environments from monocular video input, facilitating accurate spatial reconstruction and rendering. In our framework, we adopt the real-to-sim pipeline proposed in [33], which transforms real-world scanned videos into simulation-ready assets. By leveraging Gaussian Splatting, we efficiently align the reconstructed object meshes with the simulation environment, enabling seamless integration.

To further enhance geometric consistency, we incorporate the stable normal constraint introduced in [66, 67], which enforces consistent surface normals across reconstructed points. This constraint is particularly important for preserving fine surface details and mitigating noise, especially in scenes with complex geometry or intricate textures.

Together, this process allows us to recover two critical components for our differentiable physics modeling: (1) the object's 3D geometry and (2) its relative pose with respect to the robotic arm, both of which are essential for accurate system identification and simulation alignment.

C. Implementation Details

1) Implementation of Real-to-Sim Reconstruction: We begin by constructing a visually and geometrically precise digital twin of the target environment, leveraging a particle-based Gaussian splatting approach [30, 22]. From environment-centric ($\mathcal{I}_s$) video streams captured by a mobile device, we

obtain calibrated camera trajectories via structure-from-motion (SfM) [41, 50]. The pipeline then trains two disjoint ensembles of Gaussian primitives, each pursuing a separate objective.

1) Volumetric rendering set. We maintain a set of 3D Gaussians

$$\mathcal{P}^{\text{rend}} = \left\{(x_i, y_i, z_i, r_i, g_i, b_i, o_i, s_i, \Sigma_i)\right\}_{i=1}^{N_{\text{rend}}},$$

where $(x_i, y_i, z_i) \in \mathbb{R}^3$ is the center of the i-th Gaussian, $(r_i, g_i, b_i) \in [0,1]^3$ its RGB color, $o_i \in [0,1]$ the opacity coefficient for alpha blending, $\Sigma_i \in \mathbb{R}^{3\times3}$ a symmetric positive-definite covariance specifying anisotropic extent, s_i represent the semantic and instance id of the gaussian, and N_{rend} the total count of such primitives. These particles are optimized exclusively for photometric fidelity, enabling differentiable volume splatting and achieving real-time novel-view synthesis.

2) Surface reconstruction set. Geometry is approximated with a separate set of 2D surface-aligned Gaussians

$$\mathcal{P}^{\text{surf}} = \left\{(x_j, y_j, z_j, \mathbf{t}_{u,j}, \mathbf{t}_{v,j}, s_{u,j}, s_{v,j})\right\}_{j=1}^{N_{\text{surf}}},$$

where $(x_j, y_j, z_j) \in \mathbb{R}^3$ represents the disk center, $\mathbf{t}_{u,j}, \mathbf{t}_{v,j} \in \mathbb{R}^3$ are orthonormal tangent vectors, and $s_{u,j}, s_{v,j} > 0$ set the standard deviations along those directions. The outward surface normal is

$$\mathbf{n}_j = \mathbf{t}_{u,j} \times \mathbf{t}_{v,j}.$$

This ensemble is trained with depth distortion and normal consistency terms for geometric accuracy, remaining untouched by photometric loss.

After training, the surface Gaussians in $\mathcal{P}^{\text{surf}}$ are rasterized into multi-view depth maps, fused into a truncated signed-distance field, and converted via marching cubes into a triangle mesh. Surface normals are estimated [67], giving the final collision mesh $\mathcal{M}$. Since $\mathcal{P}^{\text{rend}}$ and $\mathcal{P}^{\text{surf}}$ do not share parameters and employ disjoint loss functions, improvements in appearance do not degrade geometric fidelity.

2) Constructing MJCF Models Using Reconstructed Gaussian and Mesh Representations: The MuJoCo XML Control Format (MJCF) encodes key simulation components, including an object's kinematic structure, PID control gains, stiffness parameters, collision geometries $\mathcal{K}$ along with the surface point cloud $\mathcal{P}^{\text{surf}}$ [68], and specifications of actuated joints. To construct a complete MJCF model from our reconstructed Gaussian splats and mesh representations, we first embed the static environment as an unmovable background and define the reconstructed object as a free joint body within the simulation environment.

We then align the reconstructed Gaussian coordinate frame and chirality with MuJoCo's convention, following the transformation procedure described in [33]. To ensure simulation realism, we extract the relative pose between the object and the robotic arm in the real-world scene and apply this transformation as the initial configuration of the free joint object in simulation. After integrating all relevant positional and control information, we use Vision-Language Models (VLMs) to infer initial estimates of physical parameters, including object mass, which are critical for downstream simulation fidelity.

The resulting MJCF model, with accurately aligned coordinates, initial pose, and geometry, provides a strong foundation for subsequent system identification and physics-based policy learning. It also enables high-fidelity rendering and precise real-to-sim transitions.

3) Implementation of Mass Identification: This section addresses two key aspects of our mass identification framework: (1) the strategy for mass-inertia modeling, and (2) the set of adaptive parameters necessary to support mass learning across objects with diverse physical properties and geometric variations.

a) Mass-Inertia Modeling.: In conventional settings, an object's ground-truth mass is typically distributed uniformly across its constituent particles, as defined in Equation 12. However, this strategy often leads to numerical instability and gradient explosion within real-to-sim-to-real optimization frameworks, particularly when dealing with high-resolution objects that contain over 50,000 vertices but possess relatively low mass [8]. Under such conditions, the resulting average particle mass can fall below 10^{-6} kg, introducing significant numerical errors.

To mitigate this issue, we assign the full object mass to each particle. Gravitational forces are uniformly applied to all particles, and external forces are scaled proportionally to the number of sampled vertices. This formulation preserves numerical stability by avoiding exceedingly small per-particle mass values.

Additionally, because the number of vertices varies across reconstructed objects, we adaptively select a subset of active vertices that lie on contact surfaces between the object and the robotic fingers. This further improves simulation fidelity and ensures relevant physical interactions are emphasized.

To guarantee consistency between the real-world observations and simulation environment, we explicitly synchronize frame rates, temporal bounds (start and end times), and spatial centering between the FoundationPose tracking system and the MuJoCo simulation defined in MJCF format.

b) Contact Modeling, Explicit Gradient Representation, Adaptive Learning Parameters.: We extract contact points and corresponding forces from robotic action rollouts conducted in both simulated and real-world environments. In the simulation, following the real-to-sim reconstruction, objects are placed in relative positions consistent with their real-world configurations. To ensure stable contact modeling within a Position-Based Dynamics (PBD) framework, objects are initialized slightly above the ground (e.g., $[0.05,\, 0.05,\, \frac{\text{Height}}{2} + 0.01]$), preventing premature ground contact and maintaining simulation stability.

Precise temporal synchronization across real-world object trajectories, robot control signals, and simulation rollouts is essential for reliable mass identification. We leverage FoundationPose [63] to obtain accurate object pose estimates, and align simulation timelines accordingly to ensure consistency between observed and simulated motion.

For explicit gradient computation, we implement a semi-implicit integration scheme following the formulation intro-

duced in [27], enabling differentiable backpropagation through contact events and object dynamics.

c) Adaptive Learning Strategy: To accommodate objects with varying mass scales, we employ an adaptive learning strategy. Initially, particle masses are uniformly set to approximately 0.002 kg per vertex, but this baseline must be adjusted according to the object's overall mass to ensure stable convergence. For heavier objects, such as a ketchup bottle (0.8 kg), training requires higher learning rates and longer schedules, often up to 2000 epochs, to achieve convergence. In contrast, medium-mass objects (0.1 kg) typically converge efficiently within 100 epochs using a moderate learning rate. Lightweight objects (0.05 kg) benefit from learning rate decay and similarly converge within 100 epochs.

Successful mass learning also depends on several key factors. The duration of the applied impulse, determined by the active contact interval between the robotic fingers and the object, directly influences the estimated dynamics. We select the active tracking frame from FoundationPose to mark the critical transition from motion onset to rest. Additionally, we apply a canonical re-centering vector to align object positions in simulation space, reducing variation introduced by camera viewpoint differences. Finally, the estimated contact area is adjusted proportionally to the object's vertex count and active contact regions, allowing accurate modeling of the hand-object interaction.

4) Computation detail and Timing: For mass identification, each training epoch takes approximately 0.4 to 0.8 seconds, with convergence typically achieved within 200 epochs.

D. Implementation of DREAM's Grasping Policy

Table III details the neural network architecture used in our *GraspMLP*, while Algorithm 1 and Algorithm 2 describe the training pipeline. For standard objects, the grasping policy is trained with approximately 200 demonstrations per object. For objects with higher geometric or dynamic complexity, we scale the dataset to include up to 5000 demonstrations, ensuring sufficient coverage of the variance necessary for robust policy learning. Empirically, we find that the integration of a lightweight policy network, accurate modeling of human hand-object interactions, and precise physics-informed constraints enables reliable and high-performance grasping behavior tailored to each object.

1) Computational Details and Timings: Our grasping policy is trained on datasets containing 200 to 300 demonstration poses per object by default, which results in a training duration of approximately 2 minutes per object on one NVIDIA RTX 4090 GPU. For more complex or high-variance objects that require additional data coverage, we scale the training dataset to include up to 5000 demonstrations. In such cases, the training time increases to approximately 20 minutes per object, due to the additional dataset batch size.

Inference is highly efficient. Once the policy is trained and deployed, it requires only a reconstructed URDF or MJCF representation as input, capturing the object's geometry, pose, and physical properties. Given such input, the policy predicts a stable grasp configuration in approximately 0.5 seconds per object pose. This low-latency inference time makes the system practical for real-time and on-robot applications, particularly in scenarios that demand quick adaptation to dynamic object placements or orientations.

Overall, our framework demonstrates a favorable trade-off between training cost and deployment efficiency, with scalable training capabilities and low runtime overhead for inference.

Algorithm 1 Force-Aware Policy Training

Input: Set of object meshes and masses: $\{(\mathbf{K}_i, \mathbf{M}_i)\}_{i=1}^{N}$
Output: Learned actions and forces: $\{(\mathbf{Action}_i, \mathbf{Force}_i)\}_{i=1}^{N}$

1: **for** each demonstration $(\mathbf{K}_i, \mathbf{M}_i)$ **do**
2: Extract human hand poses and object poses using HaMeR [44] and MCC-HO [64].
3: Retarget human hand poses and corresponding end-effector poses onto the robotic hand.
4: **Positional Encoding:** Encode vertices using positional encoding to obtain feature representations.
5: **Dataset Construction:** Prepare training batches comprising encoded vertices, object mass $\mathbf{M}_i$, and ground-truth actions. Load corresponding MJCF files generated by Real2Sim.
6: **Stage One Training (Supervised):** Train the policy network by setting force and contact head ground-truth labels to 1, optimizing initial grasp prediction.
7: **Stage Two Training (Simulation-based Refinement):** Roll out predicted actions within the MuJoCo simulator using the Real2Sim-generated MJCF files. Compute force and contact rewards from simulation outcomes and perform backpropagation to refine the model.
8: **Real-world Deployment:** Deploy the grasping policy onto the real robotic system using the reconstructed object mesh, executing predicted actions with force control.
9: **end for**

Component	Operation	Output Dim.	Details
Input	Vertex Features	$N \times 3$	Vertices (XYZ)
Linear 1	FC + ReLU	256	$3 \rightarrow 256$
Linear 2	FC + ReLU	256	$256 \rightarrow 256$
Linear 3	FC + ReLU	256	$256 \rightarrow 256$
Action Head	Linear	16	Joint actions
Reward Head	Linear + Sigmoid	2	Contact constraint
Force Head	Linear + Sigmoid	1	Grasping force

TABLE III: Architecture of the proposed GraspMLP network. The input consists of per-vertex 3D coordinates. The shared backbone maps the input into a latent feature space, which is subsequently decoded into separate heads for predicting joint actions, contact-based reward signals, and grasping force.

E. Implementation of Baselines on Object Grasping

a) Human2Sim2Robot Baseline.: In the Human2Sim2Robot framework [34], we operate under the

Algorithm 2 Two-phase Training Procedure

Initialize: model parameters θ, optimizer, dataloader $\mathcal{D}$, environment $\mathcal{E}$, loss functions: MSELoss ($\mathcal{L}_{\text{MSE}}$), BCELoss ($\mathcal{L}_{\text{BCE}}$).
Phase 1: Supervised Pre-training
for epoch $= 1, \ldots, E_1$ **do**
  for batch $(x, a, r, f) \sim \mathcal{D}$ **do**
    Compute predictions: $(\hat{a}, \hat{r}, \hat{f}) \leftarrow \text{model}(x; \theta)$
    Compute losses:
      $\mathcal{L}_a \leftarrow \mathcal{L}_{\text{MSE}}(\hat{a}, a)$
      $\mathcal{L}_r \leftarrow \mathcal{L}_{\text{BCE}}(\hat{r}, r)$
      $\mathcal{L}_f \leftarrow \mathcal{L}_{\text{MSE}}(\hat{f}, f)$
    Backpropagate total loss: $\mathcal{L} = \mathcal{L}_a + \mathcal{L}_r + \mathcal{L}_f$
    Update parameters θ
  end for
end for
Phase 2: Environment Interaction
for epoch $= 1, \ldots, E_2$ **do**
  for batch $x \sim \mathcal{D}$ **do**
    Predict actions and rewards: $(\hat{a}, \hat{r}, \hat{f}) \leftarrow \text{model}(x; \theta)$
    Execute $\hat{a}$ in environment $\mathcal{E}$ and observe rewards r_{env} and contact-based forces f_{env}
    Compute scaled ground-truth force: $f_{\text{env}} = \text{clip}(\frac{m \cdot g \cdot \text{num\_contacts}}{f_{\max}}, 0, 1)$
    Compute losses:
      $\mathcal{L}_r \leftarrow \mathcal{L}_{\text{BCE}}(\hat{r}, r_{\text{env}})$
      $\mathcal{L}_f \leftarrow \mathcal{L}_{\text{MSE}}(\hat{f}, f_{\text{env}})$
    Backpropagate weighted loss: $\mathcal{L} = 0.8\mathcal{L}_r + 0.3\mathcal{L}_f$
    Update parameters θ
  end for
end for

assumption that the grasping end-effector pose extracted from human demonstration videos is both accurate and physically feasible for robot execution. These grasp poses—typically obtained from hand-object interaction sequences—are directly retargeted to the Leap Hand using the official retargeting implementation provided by the Leap Hand repository, preserving the spatial fidelity of the original grasp intent.

For a fair and consistent baseline comparison, we replace the original demonstration assets and object meshes used in Human2Sim2Robot with our own Real-to-Sim reconstructed meshes, which incorporate photogeometric fidelity and physical realism as described in Section Real2sim. Using these assets, grasping policies are trained until convergence, which generally requires approximately 20,000 training epochs to stabilize reward signals and behavior.

At deployment, we assume that the relative end-effector pose remains feasible under the Franka arm and CuRobo motion planning stack. That is, we expect the grasp pose transferred from human demonstrations to be executable without requiring additional replanning or corrections during real-world trials. While this assumption aligns with the original baseline setting, it introduces potential limitations in robustness, particularly under challenging object configurations.

It is important to note that the original controller, fabric, used in Human2Sim2Robot—including closed-loop visual servoing and grasp adjustment mechanisms—is not publicly available. Consequently, our reimplementation focuses solely on static inference: given a fixed RGBD frame and known object pose, the system predicts a single-step grasp action without online feedback or corrective replanning. This constraint is taken into account in our evaluations to ensure fair comparison.

b) DexGraspNet 2.0 Baseline.: We adopt the two-stage grasping pipeline proposed in DexGraspNet 2.0 [70], which separates grasp pose generation from execution via motion planning. However, rather than directly regressing relative translations and rotations from synthetic training data, our method infers these grasp parameters through MCC-HO, a pretrained model that extracts meaningful grasp features from real human hand-object interactions captured in video. These interactions are grounded in geometry reconstructed through our Real-to-Sim pipeline, where object vertices derived from point clouds are directly used to estimate feasible grasp poses in 3D space.

Once grasp poses are generated, we utilize CuRobo for trajectory planning and execution. The planned trajectories are constrained by the robotic arm's kinematic and dynamic limits, ensuring safe and feasible real-world deployment of the inferred grasp poses.

To ensure fair comparison with DexGraspNet 2.0, which assumes a fixed object mass of approximately 0.1 kg across all test scenarios. We limit our evaluation to objects of similar mass to match the conditions under which their policy was trained. However, in contrast to this fixed-mass assumption, our approach explicitly optimizes grasp strategies using the mass identified through our differentiable real-to-sim-to-real pipeline. This enables force-aware grasping, as the identified mass is used to refine force predictions and enhance grasp stability.

By leveraging human demonstrations and accurate physical modeling, our approach generalizes more robustly across varying object shapes and dynamic properties, offering improved realism and adaptability compared to methods relying solely on simulated training data and heuristic mass assumptions.

c) Object Tracking and Motion Planning.: We employ FoundationPose [63] for real-time 6-DoF object pose estimation during manipulation. This robust visual tracking system provides temporally consistent pose predictions that enable dynamic, collision-aware trajectory planning for the robotic end-effector. These object pose estimates serve as a foundation for constructing grasping trajectories in cluttered or dynamic environments.

Once the object pose is reliably tracked, we incorporate wrist pose predictions generated by MCC-HO [64], a pretrained model designed to reconstruct hand-object interaction trajectories from human demonstration videos. The wrist poses extracted from these interactions represent feasible, human-

derived grasping configurations. Together with the Real-to-Sim object pose, they define the target end-effector pose required for grasp execution.

To generate collision-free motion plans, we formulate a constrained inverse kinematics (IK) optimization problem using CuRobo [54]. Specifically, we seek the robot joint configuration $\mathbf{A}^*$ that minimizes the distance between the robot's forward kinematics (FK) output and the desired end-effector pose $\mathbf{X}_{\text{ee}}^{\text{des}}$, while remaining within the robot's collision-free configuration space $\mathcal{Q}_{\text{free}}$:

$$\mathbf{A}^* = \arg\min_{\mathbf{A}\in\mathcal{Q}_{\text{free}}} \left\|\text{FK}(\mathbf{A}) - \mathbf{X}_{\text{ee}}^{\text{des}}\right\|_p. \tag{20}$$

Here, $\mathbf{X}_{\text{ee}}^{\text{des}}$ is derived from aligning the object pose (reconstructed via Gaussian Splatting and photogrammetry) with the wrist pose from human demonstration, forming a grounded and physically meaningful grasp target. The norm $\|\cdot\|_p$ (typically L_2) measures the spatial error in SE(3) between the planned and desired poses.

This enables physically plausible and task-relevant grasp execution that leverages real-world perception, human demonstration priors, and differentiable simulation to close the sim-to-real loop.

F. Real-World Experiments

In our experimental setup, the scene is composed of five primary components: a static table, a fixed background, a target object, a robotic arm, and a robotic hand. Both the table and background remain stationary and unchanging throughout the duration of each experiment, providing a consistent spatial context. The robotic arm and hand are fully actuated and precisely controlled, with all joint movements accurately tracked to ensure reproducibility and reliable system behavior.

The target object is entirely passive, which is not actuated or directly controlled. Its motion arises solely from physical interactions with the robotic hand, such as contact-induced forces during grasping or pushing. This object-centric dynamic behavior forms the basis for our system identification and policy learning tasks.

For visual tracking, we employ a third-person Intel RealSense D435i RGB-D camera positioned to capture the entire manipulation workspace. To estimate the 6-DoF object pose over time, we use FoundationPose [63], a real-time object pose estimation framework that ensures robust, frame-consistent predictions even under occlusion or clutter.

To reconstruct the geometric details of the experimental scene—including the table, object, and robot—we supplement the depth camera data with smartphone-based photogrammetry. Capturing a short monocular video using a mobile phone, we apply multi-view stereo techniques to generate dense 3D reconstructions of the environment. This process enables us to build high-resolution object meshes and spatially aligned scene representations, which are later used for initializing simulation environments and real-to-sim transfers.

Together, this combination of accurate tracking and high-fidelity geometric reconstruction provides the foundation for grounded simulation, physical parameter identification, and robust real-world policy deployment.

1) Hardware Setup: We employ two distinct robotic hands in our experimental framework to accommodate the varying requirements of system identification and dexterous manipulation: the Allegro Hand and the LEAP Hand, each equipped with 16 independently actuated degrees of freedom (DoF). These platforms are selected to balance mechanical precision and torque capabilities across the experimental tasks.

The **Allegro Hand** is a widely used 16-DoF anthropomorphic robotic hand developed specifically for research in dexterous manipulation. It features internalized wiring and a compact mechanical structure, minimizing external interference during physical interactions. Its low-profile design and clean joint layout simplify kinematic and dynamic modeling, making it well-suited for physical parameter identification tasks such as object mass estimation. The reduced presence of external cabling allows for more stable contact modeling and cleaner gradient flow during differentiable physics-based optimization.

The **LEAP Hand** [52] is a high-torque, cost-efficient robotic hand designed with modularity and real-world applicability in mind. It is constructed from a combination of 3D-printed components and off-the-shelf actuators, enabling easy customization, repair, and experimentation. Critical mechanical attributes—including finger length, joint stiffness, and inter-finger spacing—can be modified to suit specific manipulation scenarios or object geometries. The LEAP Hand features a novel tendon-driven kinematic structure that enables highly dexterous and human-like articulation. Each joint is capable of exerting torques that exceed those of the human hand, while maintaining realistic velocities up to approximately 8 radians per second.

A core design principle of the LEAP Hand is to maximize the proportion of mass allocated to actuators relative to the hand's total weight, thereby enhancing grip strength while preserving a compact form factor. This focus enables it to handle heavy or irregularly shaped objects that require strong and adaptive force control. Importantly, the LEAP Hand includes integrated current- and torque-limiting mechanisms, allowing for both powerful and delicate manipulation. These features make it especially suitable for executing real-world grasping tasks, where force control must be both robust and compliant.

In our experiments, we regulate the grasping force exerted by the LEAP Hand by tuning its actuator current limits, which are linearly correlated with the applied joint torques. This control scheme enables precise modulation of contact force based on object mass and surface properties, a critical requirement for sim-to-real generalization in force-aware policy learning.

By leveraging the complementary strengths of the Allegro and LEAP Hands, our framework supports both accurate physical modeling and high-performance real-world manipulation, facilitating end-to-end real-to-sim-to-real learning and deployment.

a) Rationale for Using Different Hands: We employ the **Allegro Hand** for mass identification experiments due to its compact, self-contained mechanical design, which minimizes

external interference. Its internalized wiring and low-torque actuation contribute to stable and noise-free contact dynamics, making it ideal for tasks that require accurate gradient propagation and precise system identification. These attributes are particularly advantageous when using differentiable physics to estimate object mass from robot-object interactions, where mechanical noise or inconsistent contact can significantly degrade optimization performance. The consistent kinematics and low-inertia structure of the Allegro Hand further improve the fidelity of object dynamics modeling during the real-to-sim identification stage.

We utilize the **LEAP Hand** [52] for grasping and manipulation tasks due to its high-torque capabilities and modular, human-like kinematic structure. The LEAP Hand features tendon-driven actuation with robust motors that can generate significantly higher forces than the Allegro Hand, enabling it to perform reliable grasps on objects with varying shapes, weights, and compliance. This is particularly important when evaluating real-world policy deployment, where robustness and grasp stability are critical. Its design prioritizes strength and dexterity, making it suitable for executing force-aware policies under physically realistic conditions. The hand's current-controlled actuation also enables precise regulation of grasping force, which we leverage in our policy to adapt to different object masses.

However, the LEAP Hand includes exposed wiring and tendon routing, which introduce mechanical noise and modeling complexity, especially during sensitive parameter estimation stages such as mass identification. These structural factors can interfere with accurate contact modeling and introduce inconsistencies in force feedback during differentiable simulation.

Through decoupling the roles of the two hands—using the Allegro Hand for precise physical parameter estimation and the LEAP Hand for robust manipulation—we are able to optimize each stage of our real-to-sim-to-real framework. This separation of concerns allows our framework to balance accuracy and practicality, supporting both high-fidelity modeling and real-world deployment across a diverse set of manipulation scenarios.

2) Dataset Collection and Experiment Deployment: To support accurate real-to-sim modeling, we collect approximately 300 RGB images per scene using a third-person RGB-D camera (Intel RealSense D435i) or scanning device like Iphone. These images are used for high-fidelity 3D reconstruction, which captures both the object geometry and environmental context. The full reconstruction process typically takes around 30 minutes per scene and produces a Gaussian Splats representation.

Then we convert the reconstructed visual assets into simulation-ready MJCF. This conversion encodes the object geometry as collision meshes, specifies object kinematics, and initializes physical parameters for use in simulation environments such as MuJoCo. We also extract the relative pose between the object and the robotic base, which is crucial for alignment during simulation deployment.

Our experimental environment comprises a 7-DoF robotic arm (Franka Emika Panda), a dexterous robotic hand (Allegro or LEAP, depending on the task), and a static table on which the object is placed. During data collection and evaluation, the robotic system executes predefined control trajectories or learned policies while interacting with the object. Simultaneously, FoundationPose [63] provides real-time 6-DoF object pose tracking using third-view RGB-D video input. This ensures precise alignment between real-world motion and the corresponding simulated trajectories.

All collected sensor data—including RGB frames, depth maps, robot joint states, and object poses—are synchronized and logged for later use in simulation, policy training, and evaluation. This structured dataset serves as the basis for mass identification and grasping policy learning, enabling consistent real-to-sim-to-real transfer across experiments.

G. Broader Impact

Our work advances the integration of learning from human demonstrations into robotic systems, significantly enhancing the capability of robots to interpret, replicate, and interact with the physical world in a meaningful and robust manner. By combining accurate, physics-grounded system identification with imitation learning, we enable precise and reliable grasping and manipulation, even in dynamically challenging and unstructured environments. This progress has far-reaching implications for the safe and effective deployment of robots in real-world settings such as assistive healthcare, collaborative manufacturing, household automation, and logistics.

Furthermore, by narrowing the gap between human intent and robotic execution, our framework accelerates the acquisition of complex manipulation skills without requiring extensive task-specific programming or dense instrumentation. This has the potential to lower the barrier to entry for robotics adoption, making advanced robotic systems more accessible to researchers, developers, and end-users across domains. Ultimately, our approach contributes toward democratizing robotic learning by leveraging intuitive human demonstrations and scalable, data-driven simulation.

H. Limitation:

The DREAM framework currently only supports rigid-body dynamics and relies solely on mass as the primary learnable parameter. Additionally, the current policy is specifically conditioned on the same object demonstrated by humans, making it an object-specific policy rather than a generalized solution. Lastly, once the real-to-sim stage concludes, our simulation engine requires the absence of human interaction with the real-world operation scene to maintain consistency under the assumed Lagrangian dynamics framework.

VII. List of Notations

Symbol	Description
I_s	Scene-centric RGB video sequences
I_o	Object-centric RGB video sequences
$\{I_t\}_{t=1}^{T}$	Human demonstration RGB video sequences
$\{s_t^{\text{real}}\}_{t=1}^{T}$	Real-world object trajectories
$\{s_t^{\text{sim}}\}_{t=1}^{T}$	Simulated object trajectories
m	Optimized object mass
π	Force-aware manipulation policy
S	Simulation environment representation (MJCF)
K	Collision mesh for object geometry
θ	Physical simulation parameters
P	Gaussian splatting particles for visual appearance
s_t	Object's state at timestep t (position and orientation)
u_t	Object's velocity at timestep t
v_t	Linear velocity component at timestep t
ω_t	Angular velocity component at timestep t
M	Mass-inertia matrix
f	External and contact forces
f_n	Contact force vector
k_e, k_d	Contact stiffness and damping parameters
$G(\cdot)$	Discrete-time update function
Δt	Simulation timestep
L_{traj}	Trajectory loss function
h_t	Human hand pose at timestep t
o_t	Object pose at timestep t
A_t	Robot action at timestep t
π_ϕ	Learned grasping policy (parameterized by ϕ)
$\hat{A}$	Predicted robot joint positions
$\hat{r}$	Predicted contact constraint
$\hat{f}$	Predicted grasping force constraint
n_{active}	Number of active contacts between robot and object
ρ	Object density parameter

TABLE IV: List of Notations

References

[1] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023.

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Martin Asenov, Michael Burke, Daniel Angelov, Todor Davchev, Kartic Subr, and Subramanian Ramamoorthy. Vid2param: Modelling of dynamics parameters from video, 2020. URL https://arxiv.org/abs/1907.06422.

[4] David Baraff. Rigid body simulation. *SIGGRAPH Course Notes 1992*, 19, 1992.

[5] Jan Bender, Matthias Müller, and Miles Macklin. A survey on position based dynamics, 2017. In *Proceedings of the European Association for Computer Graphics: Tutorials*, EG '17, Goslar, DEU, 2017. Eurographics Association. doi: 10.2312/egt.20171034. URL https://doi.org/10.2312/egt.20171034.

[6] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[7] Hongyi Chen, Yunchao Yao, Yufei Ye, Zhixuan Xu, Homanga Bharadhwaj, Jiashun Wang, Shubham Tulsiani, Zackory Erickson, and Jeffrey Ichnowski. Web2grasp: Learning functional grasps from web images of hand-object interactions, 2025. URL https://arxiv.org/abs/2505.05517.

[8] Peter Yichen Chen, Chao Liu, Pingchuan Ma, John Eastman, Daniela Rus, Dylan Randle, Yuri Ivanov, and Wojciech Matusik. Learning object properties using robot proprioception via differentiable robot-object interaction, 2025. URL https://arxiv.org/abs/2410.03920.

[9] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019. URL https://arxiv.org/abs/1806.07366.

[10] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation, 2021. URL https://arxiv.org/abs/2111.03043.

[11] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. In *Icml workshop on new frontiers in learning, control, and dynamical systems*, 2023.

[12] Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[13] Peter I Corke. A simple and systematic approach to assigning denavit–hartenberg parameters. *IEEE transactions on robotics*, 23(3):590–594, 2007.

[14] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[15] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4397–4404, 2015. doi: 10.1109/ICRA.2015.7139807.

[16] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation. 2021. URL http://github.com/google/brax.

[17] Clement Fuji-Tsang, Masha Shugrina, Jean-Francois Lafleche, Charles Loop, Towaki Takikawa, Jiehan Wang, Wenzheng Chen, Sanja Fidler, Jason Gorski, Rev Lebaredian, Jianing Li, Michael Li, Krishna Murthy Jatavallabhula, Artem Rozantsev, Frank Shen, Edward Smith, Gavriel State, and Tommy Xiang. Kaolin: A pytorch library for accelerating 3d deep learning research. https://github.com/NVIDIAGameWorks/kaolin, 2019.

[18] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[19] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. In *Robotics: Science and Systems (RSS)*, 2024.

[20] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer, 2020. URL https://arxiv.org/abs/2011.03148.

[21] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation, 2020. URL https://arxiv.org/abs/1910.00935.

[22] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[23] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

[24] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[25] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario

Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[26] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=c_E8kFWfhp0.

[27] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control, 2021. URL https://arxiv.org/abs/2104.02646.

[28] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342896. doi: 10.1145/2897826.2927348. URL https://doi.org/10.1145/2897826.2927348.

[29] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[31] Wisama Khalil, Maxime Gautier, and Philippe Lemoine. Identification of the payload inertial parameters of industrial manipulators. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4943–4948, 2007. doi: 10.1109/ROBOT.2007.364241.

[32] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification, 2023. URL https://arxiv.org/abs/2303.05512.

[33] Haozhe Lou, Yurong Liu, Yike Pan, Yiran Geng, Jianteng Chen, Wenlong Ma, Chenglong Li, Lin Wang, Hengzhen Feng, Lu Shi, Liyi Luo, and Yongliang Shi. Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation, 2024. URL https://arxiv.org/abs/2408.14873.

[34] Tyler Ga Wei Lum, Olivia Y Lee, C Karen Liu, and Jeannette Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. *arXiv preprint arXiv:2504.12609*, 2025.

[35] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[36] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601152. URL https://doi.org/10.1145/2601097.2601152.

[37] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, page 49–54, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345927. doi: 10.1145/2994258.2994272. URL https://doi.org/10.1145/2994258.2994272.

[38] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034.

[39] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[40] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. ISSN 1047-3203. doi: https://doi.org/10.1016/j.jvcir.2007.01.005. URL https://www.sciencedirect.com/science/article/pii/S1047320307000065.

[41] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L. Schönberger. Global structure-from-motion revisited, 2024. URL https://arxiv.org/abs/2407.20219.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

[43] Austin Patel, Andrew Wang, Ilija Radosavovic, and Jitendra Malik. Learning to imitate object interactions from internet videos. *arXiv:2211.13225*, 2022.

[44] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024.

[45] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of

robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[46] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. In *Robotics: Science and Systems*, 2023.

[47] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[48] Allen Z Ren, Hongkai Dai, Benjamin Burchfiel, and Anirudha Majumdar. Adaptsim: Task-driven simulation adaptation for sim-to-real transfer. *arXiv preprint arXiv:2302.04903*, 2023.

[49] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[50] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016. doi: 10.1109/CVPR.2016.445.

[51] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos, 2022. URL https://arxiv.org/abs/2212.04498.

[52] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023.

[53] Trevor Standley, Ozan Sener, Dawn Chen, and Silvio Savarese. image2mass: Estimating the mass of an object from its image. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 324–333. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/standley17a.html.

[54] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. curobo: Parallelized collision-free minimum-jerk robot motion generation. *arXiv preprint arXiv:2310.17274*, 2023.

[55] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[56] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[57] 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 *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[58] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[59] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

[60] Pierre Vassiliadis, Gerard Derosiere, Cecile Dubuc, Aegryan Lete, Frederic Crevecoeur, Friedhelm C. Hummel, and Julie Duque. Reward boosts reinforcement-based motor learning. *iScience*, 24(7):102821, Jul 2021. doi: 10.1016/j.isci.2021.102821.

[61] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning, 2023. URL https://arxiv.org/abs/2304.00464.

[62] Zhenyu Wei, Zhixuan Xu, Jingxiang Guo, Yiwen Hou, Chongkai Gao, Zhehao Cai, Jiayu Luo, and Lin Shao. $\mathcal{D}(\mathcal{R}, \mathcal{O})$ grasp: A unified representation of robot and object interaction for cross-embodiment dexterous grasping, 2025. URL https://arxiv.org/abs/2410.01702.

[63] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6d pose estimation and tracking of novel objects. In *CVPR*, 2024.

[64] Jane Wu, Georgios Pavlakos, Georgia Gkioxari, and Jitendra Malik. Reconstructing hand-held objects in 3d. *arXiv preprint arXiv:2404.06507*, 2024.

[65] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural Scene De-rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[66] Chongjie Ye, Yinyu Nie, Jiahao Chang, Yuantao Chen, Yihao Zhi, and Xiaoguang Han. Gaustudio: A modular framework for 3d gaussian splatting and beyond. *arXiv preprint arXiv:2403.19632*, 2024.

[67] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. Stablenormal: Reducing diffusion variance for stable and sharp normal. *ACM Transactions on Graphics (TOG)*, 2024.

[68] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017.

[69] Hui Zhang, Zijian Wu, Linyi Huang, Sammy Christen, and Jie Song. Robustdexgrasp: Robust dexterous grasping of general objects, 2025. URL https://arxiv.org/abs/2504.05287.

[70] Jialiang Zhang, Haoran Liu, Danshi Li, Xinqiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes, 2024. URL https://arxiv.org/abs/2410.23004.

[71] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*. Springer, 2024.