DexMOTS : Dexterous Manipulation with Differentiable Simulation

Krishnan Srinivasan[1], Jeremy A. Collins[3], Eric Heiden[2], Ian Ng[1], Jeannette Bohg[1], and Animesh Garg[2,3]

[1] Stanford University, Stanford CA 94305, USA,
`krshna@stanford.edu`,
[2] NVIDIA Corporation, Santa Clara CA 95051, USA,
[3] Georgia Institute of Technology, Atlanta GA 30332, USA,

Abstract. Operating everyday objects, such as rotating the lid of a jar, often requires a complex sequence of contact interactions that are challenging for robot manipulators. We present a method where object-centric motion trajectories are defined by poses and wrenches, enabling the agent to learn contact interactions efficiently in differentiable simulation environments. Our approach, DexMOTS, integrates task-specific information with a differentiable simulator to provide policy gradients which backpropagate through an environment dynamics model to efficiently learn a goal-conditioned controller. Concretely, the trajectory of object pose and wrench requirements gives the necessary information to handle varying friction or damping forces, such as in the case of a tightening screw cap, or when pushing against a spring-loaded lever. To highlight the benefits of our approach, we developed a set of dexterous manipulation scenarios in which a robot has to control operable objects with joints of varying stiffnesses, shapes, and functional task requirements. Then, we demonstrate how object-centric motion trajectories can be leveraged to learn dexterous motion policies for different hand morphologies. We empirically validate our method against a set of model-based and model-free RL baselines and show that DexMOTS achieves up to 40% higher success rates on a suite of realistic contact-rich manipulation tasks. For further visualizations refer to our website: dexmots.github.io.

1 Introduction

Dexterous manipulation is a challenging domain for learning-based continuous control algorithms, with recent learning-based methods tackling grasping and in-hand manipulation tasks with dexterous hands [8, 25, 32, 37]. For household and personal robotics applications, most objects are designed specifically for human hands, requiring robots with dexterous capabilities to manipulate them. Tasks like spraying surface cleaner, opening a jar, or dispensing soap require dexterous manipulation to operate specific locations on the object. However, the interaction between grasping and in-hand manipulation is complex, especially in instances of tool use where an object-specific grasp is needed to articulate the object. We propose DexMOTS to learn policies that can interact with operable objects, which have to be articulated carefully by deliberate contact interactions by dexterous hands. DexMOTS allows learning policies that follow a desired object reference trajectory in pose-wrench space. Additionally, we introduce

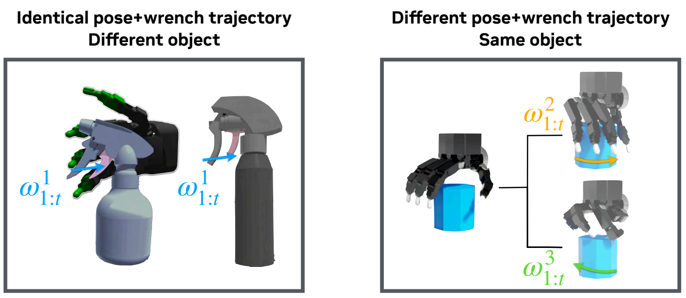

Fig. 1: Tasks from different object classes from our task set: the spray bottle and screw top. The sequences of target wrenches applied on the object's degree of freedom are represented with $\omega_{1:t}$. The left half visualizes a intra-object wrench reference trajectory ω for actuating multiple spray bottle levers. On the right, a planar rotation is used to apply either the counter-clockwise ω or clockwise torques ω to the object.

a contact-rich simulation benchmark with a suite of challenging dexterous manipulation tasks. **WarpManip** is distinct from prior benchmarks in its focus on learning to manipulate articulated tools from the PartNet mobility dataset [34]. On a variety of task-oriented grasping and manipulation scenarios, we compare our method to state-of-the-art baselines in model-based, model-free, and differentiable simulation-based reinforcement learning methods, as well as a simple MPC baseline.

We evaluate our suite of tasks from DexMOTS using standard SOTA RL methods from `rl_games` [23], and frame each task as a goal-conditioned reinforcement learning problem. Our contributions are: 1) developing a benchmark for dexterous manipulation tasks with GPU-accelerated differentiable simulation, 2) designing a suite of object-centric tasks studying rigid body and operable objects, and 3) creating an object-centric goal-conditioned method for dexterously manipulating operable objects.

2 Related Work

Current approaches to dexterous manipulation have mainly studied reposing tasks [3, 7], which multiple benchmarks [2, 14, 24] have implemented to evaluate ground truth and vision-based reinforcement learning methods. Our work sets out to highlight the challenge of task-oriented grasping and manipulation of objects involving two or more connected articulating bodies, as found in many objects that humans interact with by leveraging innate dexterity.

Dexterous Manipulation Task Benchmarks – Existing benchmarks with environments for dexterity [2, 3, 8, 25, 37] primarily focus on manipulating rigid objects, such as cubes or hard toys. However, objects like scissors or staplers require task-specific grasps and dexterous interactions in order to be used, and involve controlling a specific joint on the object. Recently, labeled interaction datasets collected by human demonstrators [9, 10] study how humans employ complex dexterous control in videos of interaction for operable objects with internal joints. Our work shows that even state-of-the-art reinforcement learning methods can struggle when adapting to new object instances or changes in the object's dynamics due to object instance-specific simulation parameters. These baselines also require extensive compute time, physics randomization, and a large number of samples for training. Furthermore, they do not generalize well to a single

Benchmark	**Robot hand types (DoF)**	**Sensing**	**Simulator**	**Speed**	**Objects**
TCDM [8]	3/4/5-fingered grippers	GT-state, Proprioception	Mujoco	$1,500$ fps	30 objects
Maniskill2 [14]	Two-fingered gripper	Proprioception, RGBD	SAPIEN (Warp-Sim)	$2,500$ fps	2000+ objects
IG-Envs [24]	Allegro Hand (16), Shadow Hand (24)	Visual and Force Sensors	IsaacGym	$22,000$ fps	Block, pen, and egg
DexArt [5]	XArm6 (6), Allegro Hand (16)	GT-state, RGBD	SAPIEN	$2,500$ fps	82 articulated objects
Ours	Allegro Hand (16), Claw (4)	GT-state, Proprioception, RGB	Warp	$400,000$ fps (proprioception), $9,000$ fps (RGB)	14 (operable) objects

Table 1: Current benchmarking methods for dexterous manipulation, their relative speeds for simulating physics, and support for dexterous manipulation and articulating objects.

policy architecture that can handle different object types. We draw inspiration from task variations presented in previous work [38] to highlight these challenges, and show that our method for goal representation enables better learning in these scenarios.

Task Representations for Learning Dexterous Manipulation – Many recent approaches to dynamic dexterous manipulation policies or controllers for object manipulation use deep reinforcement learning to effectively handle high degree-of-freedom control with hard-to-specify cost terms, and optionally high-dimensional input. A common task is to re-orient an object, such as a cube or a toy, from an arbitrary initial pose to some desired final pose [2, 3, 7, 29]. However, in contrast to our work, these works focus on reposing tasks where an object does not need to fully track a desired object motion but only reach the final pose. Our method first defines these motion trajectories explicitly in the joint configuration space of the object, where the pose of the object is considered as an additional 7-DoF free joint. Then the trained policy has to find a suitable grasp to produce this desired object motion trajectory. This is similar to prior work in dexterous functional grasping and manipulation [1, 8], which takes the object pose trajectory into account, but excludes the object's articulated joint poses in these trajectories. Recent work [4] has also shown that RGB videos of bimanual manipulation can be used to model object motion using a screw axis to learn and finetune imitation learning policies, further supporting the usefulness of object-centric motions as a representation for learning.

In [6, 10, 19, 25], feasible grasps to achieve a specific kind of object motion are found using labeled demonstrations that produce grasp heatmaps on different objects that impose a human prior over contact region and hand-pose (via reward functions and affordance maps). In contrast, our work discovers such grasps by choosing a starting hand pose close to the object, and samples feasible grasps with a prior method, DexGraspNet [33]. Additionally, object motion trajectories are created on the fly when only provided with a goal pose by performing object trajectory optimization. This we experimentally show allows the policy to achieve desired poses for each object better by breaking down the trajectory into waypoints, and use the force profile to determine the policy's relative stiffness.

Model-Based Learning and Differentiable Simulation – Next, we describe methods that use model-based reinforcement learning or differentiable simulation which introduce a structural prior for several robotic continuous control tasks adjacent to grasping and manipulation. In Grasp'D [32], a differentiable simulator similar to ours is used to explore the space of grasping parameters that yield natural grasps according to a differentiable simulation-based grasp metric that evaluates grasp stability. Our work does not consider the same grasping metric since our tasks involve object motion trajectories. Instead, we use the reward signal from the goal-conditioned control task directly to implicitly find grasps that can generate a desired wrench on the object and produce smooth trajectories. Other related works, such as SAM-RL [20] and PODS [28], use a hybrid model-based reinforcement learning and differentiable simulation-based approach to learn manipulation tasks. A key distinction of our work is that while we also leverage a transition model for computing gradients, we use a short horizon to learn a general reactive policy used for dexterous manipulation skills. In contrast, their approach does not consider applications of dexterous manipulation, where the tasks a) require higher degree of freedom control, and b) manipulate an object where sliding friction or revolute friction with other surfaces may be considered. Lastly, our method, DEXMOTS, extends the differentiable simulation-based policy learning algorithm Short-Horizon Actor-Critic (SHAC) [36]. In particular, we propose a goal-conditioned variant that leverages dense task signals from contact, such as the forces applied on the object, to enable dexterous manipulation on tasks such as screwing, grasping, and lifting. One limitation of this approach, however, is that in order to bridge the sim-to-real gap, the simulator must closely match the real task's physics for the policy to be transferrable. Independent works [16] have shown to be capable of learning complex real-world dynamics from data collected on real hardware, which is then used to fit the simulation hyperparameters by leveraging differentiable physics.

3 Problem Statement

This work aims to train a policy that tracks object-centric reference trajectories. We formulate this problem as a goal-conditioned reinforcement learning problem. Let $\langle \mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{T}, r, \gamma, \rho \rangle$ be a Markov Decision Process (MDP) with states $s_t \in \mathcal{S}$, goals $g \in \mathcal{G}$ and actions $a_t \in \mathcal{A}$ indexed by time $t \in 0, 1, \cdots, T$ where T is the maximum episode length. We define the state space as the full ground-truth state vector of the hand's pose, joint angles, and velocities, and the object pose and velocity. As described in Sec. 4, we generate an object-centric reference trajectory consisting of subgoals $g_t = \{p_t, w_t\} \in \mathcal{G}$, which define the desired object pose p and wrench w at each time step t. Each object pose p_i is defined by a position $x^o \in \mathbb{R}^3$ and orientation $o^o \in SO(3)$. A wrench $w_t^o = \{f_t^o, t_t^o\}$ is defined by the forces and torques applied to the object at time step t. In summary, the pose goals p_i^o determine the position and orientation of the object throughout the trajectory, while the wrench goals determine the forces and torques required to achieve the desired pose of the object. Since this includes the internal joint torques and forces for articulated objects, the goal-conditioned agent can directly learn on the links of the object to minimize the wrench and pose tracking error.

4 Object-Centric Task Plans

We consider the manipulation of objects with and without *operable* parts. Specifically, objects with operable parts, such as the lid of a bottle or the lever of a spray bottle, require control of additional degrees of freedom on the object, in addition to the base object pose. To affect an object's state in this way, it is important to take actions (or at a higher

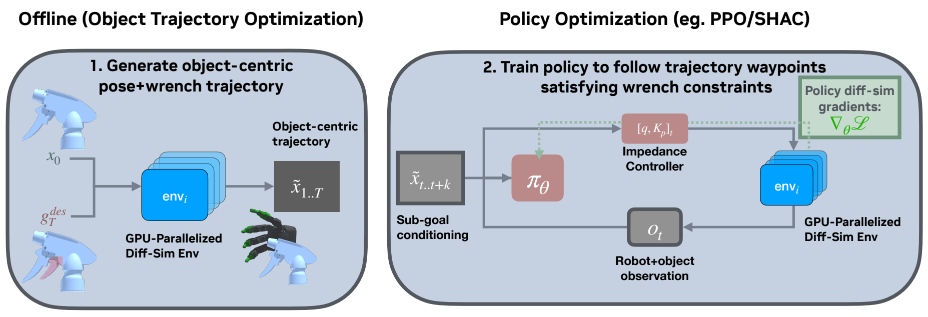
Offline (Object Trajectory Optimization)
1. Generate object-centric pose+wrench trajectory
x_0
g_T^{des}
env$_i$
GPU-Parallelized Diff-Sim Env
Object-centric trajectory
$\tilde{x}_{1..T}$
Policy Optimization (eg. PPO/SHAC)
2. Train policy to follow trajectory waypoints satisfying wrench constraints
Policy diff-sim gradients: $\nabla_\theta \mathcal{L}$
$[q, K_p]_t$
Impedance Controller
$\tilde{x}_{t..t+k}$
Sub-goal conditioning
π_θ
env$_i$
GPU-Parallelized Diff-Sim Env
o_t
Robot+object observation

Fig. 2: An overview of our method, which has two distinct phases: 1) object trajectory optimization, yielding the desired object pose+wrench trajectorym and 2) policy optimization, where the robot agent's motion policy is learned with policy gradients from differentiable simulation.

level, grasps) that apply forces to the object to generate a wrench that results in the desired motion. For instance, tightening a bottle cap requires a variation in the applied force/torque over time while changing the pose of the cap, as visualized by the different arrows in Fig. 1. Our hypothesis is that *providing the trajectory of wrenches to apply to an object guides an agent towards certain grasps and contact modes which enable the application of desired forces on the object, and achieve a desired motion*. We explore this effect in a goal-space ablation study in Sec. 7.

While there are several methods to generate an object's motion trajectory with poses and wrenches, we leverage a differentiable simulator and follow a gradient-based approach to find the trajectory of net wrenches applied to the object to reach a desired goal pose p_T^o. First, given the object's initial pose p_0, we linearly interpolate the pose of the object over time to follow a smooth pose reference trajectory $\bar{p}_{1:T}$. Next, we simulate a sequence of wrenches $\omega_t^o \in \mathbb{R}^6$ applied to the object initialized at pose p_0 to obtain an object pose trajectory $\hat{p}_{1:T}$. Finally, we minimize the pose tracking error $\mathcal{L}_{pose} = \sum_{t=1}^{T} \|\hat{p}_t^o - \bar{p}_t^o\|$ by differentiating back through the differentiable simulator to compute the gradient $\nabla_{w_t^o} \mathcal{L}_{pose}$, and perform a gradient update to improve the wrench sequence tracking the desired reference motion.

Our gradient-based approach for producing wrench trajectories can also be computed for a pose-only trajectory that is generated from a human demonstration, such as those collected in [8]. Once an initial pose-trajectory motion plan is given for an object, the net forces and wrenches applied over the trajectory are approximated by taking the finite differences of the angular and linear velocities. The advantage of doing so is allowing re-use across different embodiments, since the poses are object-centric and do not require the hand poses, which we show performs just as well or better than pose-only trajectories in our experiments (Sec. 7). In summary, the object-centric pose-wrench trajectories yield a compact yet expressive task space while learning to dexterously manipulate objects in scenarios involving variable amounts of friction or damping forces, such as rotating a bottle cap which tightens, or pushing against a spring-loaded lever.

5 Policy Learning with Differentiable Simulators

In this paper, we propose DexMOTS, an actor-critic learning algorithm that combines policy learning in differentiable simulation with goal-conditioning on object-centric goals. As discussed in Section 3, we cast the problem as a Goal-Conditioned MDP where the goal state contains the object goal pose x_{goal} and desired wrench trajectory w_{des}. This task conditioning captures temporal information about the desired tool-use affordance.

1. Reward Model – First, we define an object-centric task reward, which is computed based on the difference between current and desired object pose and wrench at each time-step (time-index omitted for brevity):

$$\boldsymbol{r}_o = \lambda_{\text{pose}} r_{\text{pose}} + \lambda_{\text{wrench}} r_{\text{wrench}} \tag{1}$$

where $r_{\text{pose}} = \exp(-\|p_o - \bar{p}_o\|)$, $r_{\text{wrench}} = \exp(-\|\omega_o - \bar{\omega}_o\|)$, with object pose $p_o \in \mathbb{R}^6$ and wrench $\omega_o \in \mathbb{R}^6$, each containing a linear and angular component. As previously done by [2], we choose exponentiated rewards for object tracking errors to simplify the normalization of different unit distance scales when computing an object's displacement from its desired pose. This also ensures the reward gradients do not approach zero as the position or orientation goal is close to being reached. For clarity, we vectorize the object-centric reward as $\boldsymbol{r}_o = [r_{\text{pose}}, r_{\text{wrench}}]$, with the corresponding cost coefficients written as $\boldsymbol{\lambda}_o = [\lambda_{\text{pose}}, \lambda_{\text{wrench}}]$.

The agent-specific reward $\boldsymbol{r}_a$ penalizes the magnitude of the actions at each time-step and includes an additional reward term to encourage the fingertips to be near the object's surface by using their distance to the object. Specifically, we use

$$\boldsymbol{r}_a = \lambda_{act} r_{act} + \lambda_f r_f, \tag{2}$$

containing the action penalty $r_{act} = -\|a\|$, and the distance to the object centroid is $r_f = \sum_{i=1}^{N_f} \exp(-\|\mathbf{x}^{f_i} - \mathbf{x}^o\|)$, where $\mathbf{x}^{f_i}$ is the position of the robot's fingertip, N_f is the number of fingers, and $\mathbf{x}^o$ is the position of the object's center. We vectorize the agent-specific reward coefficients as $\boldsymbol{\lambda}_a$. To encourage dexterous grasps that manipulate the objects more from the distal joints, we apply a weighted λ_{act} at each finger with a higher action penalty applied at the distal joints than at the proximal joints. Since this will depend on the number of joints, links, and morphology of the robot, this reward term is agent-dependent. The full combined object and agent reward functions used to train the policy are then given by summing Eq. 1 and 2:

$$r = \boldsymbol{\lambda}_o^T \boldsymbol{r}_o + \boldsymbol{\lambda}_a^T \boldsymbol{r}_a. \tag{3}$$

Given the current state of the object and manipulator, as well as the desired wrench trajectory, the policy is optimized by an actor-critic method similar to [36] that maximizes the cumulative returns from Eq. 3 (see Sec. 5.2).

We use additional force and torque reward terms from the object-centric reward function in Eq. 1 which supplement the task reward for the policy.

From a desired pose-wrench trajectory, the reward landscape for policy learning becomes better conditioned in order to converge to a grasp that enables the desired affordance. This is highlighted in Fig. 4, which visualizes the policy gradient objective landscape near a point of contact for the two-fingered Claw environment when rotating a fixed valve (see details in Sec. 7). The x and y-axes plot the subspace of joint values for the left and right distal links, and the level sets indicate local changes in the expected reward by perturbations to the joint actions. The gradients, shown by the arrows, are computed via the differentiable simulation backend, Warp. Under pose-only goal-conditioning, the reward landscape is flat and sparse, making policy learning difficult. In contrast, the reward landscape under pose-wrench conditioning favors actions going toward the four corners of the plot, thus leading to higher rewards (i.e. more rotation).

2. Policy Learning with Object-Centric Goals – After acquiring object-centric trajectories, the dexterous motion policy π_θ is learned to control a hand to articulate the object according to the desired trajectory, similar to trajectory-conditioned policies in [8, 15]. Since the goal changes over time, the policy must be conditioned on the goals or the rewards become non-Markovian, since the same state-action tuple (s_t, a_t) at time-step i in the trajectory may return a different reward at time-step j. Our method is therefore a goal-conditioned actor-critic method, with a target-critic to stabilize the critic learning objective [12]. To facilitate exploration during the learning process, the policy samples an action $a \sim \mathcal{N}(\mu_\theta(s_t, \mathbf{g_t}), \sigma_\theta) \in \mathcal{A}$.

In the goal-conditioned setting, the critic estimates a goal-conditioned value function:

$$\bar{Q}_\pi(\boldsymbol{s}_t, \boldsymbol{g}_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi}\left(\sum_{t'=t}^{T} \gamma^{t'-t} r(\boldsymbol{s}_t, \boldsymbol{g}_t, \boldsymbol{a}_t)\right), \tag{4}$$

which calculates the cumulative return from the policy π that is conditioned on $\mathbf{g}_t$. This is done in the finite-horizon case as the object-centric trajectories $p_{t:t+h}$ are considered solve-able within a fixed-length horizon H.

The critic learns to optimize the following training objective:

$$\mathcal{L}_\phi = \underset{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{g}) \in \{\tau_i\}}{E}\left[\|Q_\phi(\boldsymbol{s}, \boldsymbol{g}, \boldsymbol{a}) - \tilde{Q}_\pi(\boldsymbol{s}, \boldsymbol{g}, \boldsymbol{a})\|^2\right], \tag{5}$$

which approximates the true action-value function from Eq. 4. To approximate the action-value Q with low variance, we use TD(λ) [31] using an exponentially weighted average of different k length sub-trajectories sampled from a replay buffer $\mathcal{D}_{\pi_g}$,

$$\tilde{Q}(\mathbf{s}_t, \boldsymbol{g}_t, \boldsymbol{a}_t) = (1 - \lambda)\left(\sum_{k=1}^{h-t-1} \lambda^{k-1} G_t^k\right) + \lambda^{h-t-1} G_t^{h-t}, \tag{6}$$

where $G_t^k = \left(\sum_{l=0}^{k-1} \gamma^l r_{t+l}\right) + \gamma^k Q_\phi(\mathbf{s}_{t+k}, g_h, a_{t+k})$ is the k-step return from time t.

In actor-critic methods, the policy π_θ is trained in conjunction with the critic optimizing Eq. 6 to increase the likelihood of sampling higher value actions. Since the critic's parameters can change rapidly during the course of policy learning, we employ the target critic $Q_{\phi'}$ with polyak averaged updates [12, 26] for better policy gradients computed from Eq. 6. When computing the policy gradients via Monte-Carlo rollouts, it is important to have enough samples (N) to cover the support of the state distribution induced by the policy (ρ_π). This becomes difficult to compute when the state-space $\mathcal{S}$ is large and the rewards are goal-conditioned. However, this highlights an advantage of using model-based learning, which computes the gradient from the task reward directly with respect to the sampled actions to update the policy parameters, and is discussed further in the following section.

6 GPU-Parallelized Differentiable Simulation

Differentiable simulators can compute model-based gradients through the transition function by taking the gradient of the actions and states with respect to the forward step operator $\mathcal{F}$. However, propagating model-based gradients over a long horizon poses a challenge where the policy gradient from the actor objective (Eq. 8) over many time-steps will tend to vanish or explode. Naively using differentiable simulators with many contacts, as is the case in contact-rich manipulation, exacerbates this challenge, as discussed in prior literature [11, 18, 39].

Physics Simulator – In this section, we provide a short overview of the physics used in our differentiable simulator, for which further details can be found in [22]. We develop a novel differentiable simulator in this work for controlling objects with articulated joints with multi-fingered hands. The forward transition kernel operator $\mathcal{F}(s,a) = s'$ computes the next state according to the dynamical system

$$\boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} = \boldsymbol{J}^T(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x},\dot{\boldsymbol{x}}) + \boldsymbol{c}(\boldsymbol{x},\dot{\boldsymbol{x}}) + \boldsymbol{\tau}(\boldsymbol{x},\dot{\boldsymbol{x}},\ddot{\boldsymbol{x}}) \tag{7}$$

The physics simulator used integrates Newton's equations of motion in terms of the body positions, $\boldsymbol{x} = \{x_1, ..., x_n\}$, where external forces on them are represented by f, with c giving Coriolis terms, and τ combining terms for joint-space actuation for both the object and hand. Our physics simulator computes this forward step to update the positions of the rigid bodies in the environment subject to a vector of constraint functions imposed by contact $\boldsymbol{C} = [C_1(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x})]$. This is solved via a quasi-Newton method to yield the positional update $\Delta\boldsymbol{x} = M^{-1}\nabla\boldsymbol{C}(\boldsymbol{x}_i)^T\Delta\boldsymbol{\lambda}_i$, where $\boldsymbol{\lambda}_i$ is the vector of the Lagrange multipliers for the constraint C_i. The contact points between rigid shapes and meshes are yielded from a collision detection step and serve as contact constraints (C) in these positional updates.

We implement our simulator in Warp, a software library for writing simulation kernels in Python which is GPU accelerated, as well as the gradient of the kernels via automatic differentiation [21]. Our simulator incorporates SDF-based collision handling when computing rigid contacts with meshes, which preserves dense area contacts without compromising simulation speed. Additionally, we choose extended position-based dynamics (XPBD) [22] to simulate the articulated rigid-body mechanisms and contact dynamics in reduced coordinates. XPBD is a flexible simulation approach that handles contacts and joints between rigid bodies via constraints and accurately resolves complex multi-contact interactions involving linear, rolling, and torsional friction [22]. Analogously to the soft contact formulation in [32, 35], we introduce smoothness to the rigid contact resolution by adding relaxation to the contact constraints. Such relaxation allows for slight constraint violations which softens the otherwise discrete contact response when collisions between rigid bodies occur. For our benchmark tasks, the loss landscape is more difficult to optimize due to the high number of potential contacts between the hand and the object. In the simulated Allegro hand environments, we adopt a relaxation of the rigid contact constraints applied when using implicit position-based dynamics integration [22]. In the case of our simulated manipulation environments, this allows a smoother continuous gradient through points of contact between the multifinger end-effector and the object.

Automatic Differentiation through Forward Simulation – Through the construction of the forward step kernel $\mathcal{F}$ which caches the computed matrices used to generate s' (as given in Eq. 7), given a differentiable loss function $\mathcal{L}$, we can compute the following backwards gradient adjoint terms:

– the action gradient given by: $\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}}{\partial s'} \frac{\partial \mathcal{F}}{\partial a}$

– the previous state gradient, given by: $\frac{\partial \mathcal{L}}{\partial s} = \frac{\partial \mathcal{L}}{\partial s'} \frac{\partial \mathcal{F}}{\partial s}$

These terms are concatenated using the chain rule to compute gradients through the entire trajectory.

Gradient Computation with Truncated Horizon – In practice, caching the intermediate forward kernels produces a memory bottleneck, slowing down training and resulting in noisy gradient signals, especially as the length of the horizon grows. We resolve this issue by breaking the full episode length T into shorter sub-trajectories of $H \ll T$ time-steps, applying actor-critic updates at the end of each sub-trajectory. While in prior work [36] horizon lengths were fixed to 16 and 32, this does not work well for contact-rich tasks due to a) the increased noise in computing gradients for our contact-rich tasks, and b) the increased amount of GPU memory used due to the large number of mesh-to-mesh contact constraints. We address this with a shorter H that can be treated as a parameter tuned by the frequency of stiff contacts, which produce discontinuous gradient signals. Adopting the Adaptive Horizon Actor-Critic (AHAC) framework [13], the horizon lengths are adjusted according to a heuristic designed to keep the stiff contact forces within a fixed constraint threshold $\bar{C}$. This yields shorter horizons truncating the compute graph which reduces the overall variance of computed gradients. Returning to Eq. 4, the actor training objective for the short horizon from time-step t to $t + H$ is defined as:

$$J(\theta) = \sum_{h=t}^{H+t-1} \gamma^{h-t} r(\boldsymbol{s}_h, \boldsymbol{a}_h) + \gamma^h V_\psi(\boldsymbol{s}_{t+H}) \tag{8}$$

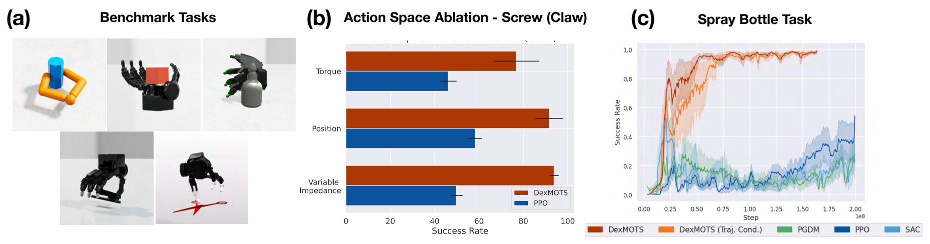

Fig. 3: (a) Visualization of the benchmark tasks: Planar rotation with the claw (two-finger) and Allegro hand, Cube Rotate, Spray Bottle, Scissors, and Stapler. (b) Bar plot showing the relative performance of our method compared to two other baselines, PPO (model-free RL), and PGDM (model-based RL), for different choices of action space, the planar rotation task using a pedagogical Claw (two-fingered). (c) Learning curves comparing DexMOTS (with and without trajectory conditioning), PGDM, PPO, and SAC success rate on the spray task in the operable object set.

7 Experiments

Our experiments investigate whether:

1. Differentiable simulation is effective for policy-learning, and it outperforms comparable RL baselines in wall-clock time, sample efficiency, and task success.
2. Object-centric task spaces form a smoother loss landscape for policy learning to optimize, and improve policy robustness to dynamics perturbations in the environment.
3. DEXMOTS, leveraging a smoothed policy learning landscape, can also follow a manually collected object pose trajectory from the TCDM benchmark [8].

Environment	Ours	Ours (pose-only)	MPC	SAC	PPO	PGDM
Repose Cube	**96.0** $\pm$ **3.0**	85.2 ± 4.3	18.0 ± 3.1	74.8 ± 8.2	89.4 ± 2.9	53.6 ± 6.8
Spray Bottle	**89.6** $\pm$ **2.9**	63.4 ± 8.3	8.4 ± 4.3	47.8 ± 5.8	51.8 ± 8.7	83.8 ± 4.0
Scissors	**42.4** $\pm$ **5.3**	41.6 ± 7.5	5.0 ± 1.9	37.6 ± 5.3	18.4 ± 7.1	39.6 ± 7.2
Stapler	81.6 ± 7.4	63.5 ± 8.4	14.2 ± 4.4	26.8 ± 12.6	27.0 ± 6.3	**88.4** $\pm$ **13.7**
Screw-top (Claw)	92.6 ± 6.4	**93.8** $\pm$ **2.1**	8.6 ± 2.8	17.0 ± 3.7	47.4 ± 3.8	86.2 ± 10.4
Average	**80.4** $\pm$**5.0**	69.5 ± 6.1	10.8 ± 3.3	40.8 ± 7.1	46.8 ± 5.8	70.3 ± 8.4

Table 2: Success rate (% of 100 episodes), reaching a final pose error of less than ϵ_{task}, averaged over five randomly seeded training runs for RL methods (DEXMOTS, PGDM, and PPO), and five seeded runs of MPC. The MPC formulation uses the predictive sampling method with tracking cost gradients to update planned actions, and the PGDM and PPO are mirrored from [8**?**].

Task Setup – We introduce three differentiable simulation environments for dexterous manipulation using object-centric pose-wrench trajectories: (1) *planar object rotation* (Screw), where an applied torque to rotate an object must be controlled, and the object joint friction is unknown and randomized per episode, (2) *object reposing*, where applied forces repose an object to a desired position and orientation, and (3) *object articulation*, where the fingers of a dexterous hand must form a stable grasp to operate an object's articulated revolute joint to a desired state.

In the *planar object rotation* task, the object trajectories τ_o characterize the required torque to rotate the joint to its desired angle θ. In the *reposing* task, the cube rotate task from [3] is used and the policy is conditioned on net torques applied to the object to reach a desired orientation pose. The last task (constituting experiment 4), evaluates our method on a realistic task of grasping and articulating objects from the Partnet Mobility benchmark [27].

Metrics & Baselines – For evaluation, we determine if a learned dexterous policy is able to solve the above tasks with our method and a set of baselines, shown in Table 2, where we record the following metrics: a) the final task success rate, b) the time taken to solve the task.

For the baseline methods, we compare a fast parallelized implementation of Proximal Policy Optimization (PPO) [23, 30] (which does not feature any goal-trajectory conditioning), Pre-Grasp informed Dexterous Manipulation (PGDM) [8] (which extends PPO with pose-trajectory conditioning along with grasp initialization), and trajectory optimization MPC baseline [17] which directly updates the time-indexed control parameters with cost gradients from the simulator. Note that for the PGDM baseline, we use

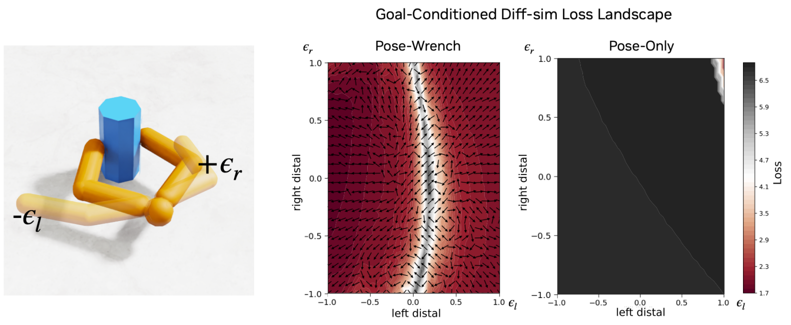

Fig. 4: Visualizing the object-centric loss landscape (negative reward) for rotating the prism from an initial contact state (left), with (center) and without (right) the desired pose-wrench goal. By differentiating a single time-step reward (minimizing the distance to the target pose-wrench) with respect to the robot joint positions, yields a smoother landscape for optimization.

the same pre-sampled grasp pose as our method without running the additional CEM optimization step as in the original paper [8], as our method does . Note that MPC is included only as a lower-bound on the success rate for the harder tasks, since it requires more memory to optimize higher-DoF, long-horizon tasks.

DexMOTS ***Ablations*** – Our ablations highlight two main ideas in our approach: a) the use of a pose-wrench object-centric task-conditioning, and b) the variable stiffness action space for controlling the hand. In DexMOTS (*Goal pose*), the policy is only conditioned on the *final* goal pose of the object, while the default. In the *goal pose-trajectory*, it is conditioned on the all object goal poses $\leq H$ steps from the beginning of the short horizon, where H is the horizon length used in Eq. 8. Finally, in *goal wrench*, the policy is conditioned on the desired pose and wrench of the object H time-steps in the future, capturing the net wrench and pose to apply at that step.

Experiment 1: (Learning in differentiable simulation) – First, we determine how learning a policy using gradients through the simulator affects the sample efficiency of learning and success rate. As shown in Figure 3(c), DexMOTS yields better performance both in fewer simulation steps, as shown by the object planar rotation tasks. To compare across simulators to highlight that our simulator does not produce any disadvantages to non first-order gradient-based methods, we ran the Repose Cube task in IsaacGymEnvs [24], and found that PPO, SAC, and PGDM performs within error margins on the repose task. One advantage of our method across the different tasks in Table 2 is that it requires minimal hyperparameter tuning per-task. The aggregate reward coefficients in Eq. 3 were consistent across all of the tasks.

Experiment 2: (Action Space Ablation) – Next, we compare three different controllers: variable impedance, position, and torque control, in the pedagogical Screw task to show the advantages of using variable impedance control. As shown in Fig. 3b, policies trained using the variable impedance (VI) action space consistently converged to the lowest final goal-pose error across all 3 baseline algorithms. Notably, in the Screw benchmark, PGDM in the joint position action space achieves comparable performance to the VI action space, only being out-performed by DexMOTS policy trained with VI actions. However, in the rigid object Repose Cube task, PGDM does not outperform even PPO due to its additional complexity in the reward function and unnecessary pose trajectory conditioning. As the task complexity grows for the operable articulated objects tasks, we find the model-free baselines, with the exception of PGDM, do not perform well at the tasks, with Table 2 showing the best policies being from either DEXMOTS (using variable impedance) or DEXMOTS (goal-pose).

Experiment 3 (Goal Space Ablations): – Next, we analyze how different choices of task specifications affect final task success. Table 2 shows the performance DEXMOTS with and without (goal-pose) wrench trajectories. The final success metric for the DexMOTS with goal-wrench trajectories is higher than goal-pose only on the Spray Bottle, Scissors, and Repose Cube tasks and Stapler tasks, and is close to optimal for the Screw task. We hypothesize this is a result of the smoother objective landscape (visualized for 1 step horizons in Fig. 4) and contributes to an easier-to-optimize objective for first-order gradient methods.

Experiment 4 (Realistic Operable-object Tasks): – We evaluate DEXMOTS on a realistic task where a four-fingered Allegro hand articulates *operable objects* from the SAPIEN articulated object set [27]. Herein, the challenge is in applying wrenches to the object using the Allegro hand asset, which is distinct from the hand used in the original TCDM benchmark tasks [8]. Using our differentiable simulator, we acquire an object-centric wrench trajectory generating the desired motion of the object in the simulator, without needing to retarget joint positions to our robot hand. Then, we condition the agent on the goals generated from the resulting trajectory, and compare our method with the PPO and PGDM baselines. As shown in Table 2, DEXMOTS outperforms these baselines in both the overall task success metric (L2 error from the final goal pose below ϵ_{task}) and required number of samples needed to converge (Fig. 3(c)) relative to the model-free counter-parts. Notably, first-order methods (MPC) are unsuccessful for all of the tasks, mainly due to the high-dimensional action and state spaces.

8 Conclusion and Future Work

In this work, we present our method, DEXMOTS, as an approach to learn policies for object-centric tasks that require contact-rich dexterous manipulation. Specifically, we consider a class of tasks where the state of articulating objects must be controlled, while also exhibiting varying degrees of stiffness during a trajectory. We apply DEXMOTS on a new benchmark of tool-use tasks, WARPMANIP, involving interaction with a stiff revolute joint (like a screw-top lid or door knob) and a dexterous grasp with a stapler, spray bottle, and scissors, and show that our method empirically outperforms model-based and model-free baselines. One potential extension of our work would be to incorporate differentiable grasping metrics, such as those explored in [32], to find realistic pre-grasps to initialize training from. Following the example of [16], another future step would be closing the sim-to-real gap by using our differentiable simulation environment to learn the real world physics parameters through data collection, eventually allowing the policy to also run on a real robot at test time. Lastly, another promising future direction is to evaluate the utility of object-centric task spaces for multi-task learning across objects with similar shapes and varying stiffnesses, such as different types of bottles, as a way to extend DEXMOTS to generalize to object classes.

Acknowledgment

This work was supported by the National Science Foundation under Grant Number 2342246.

Bibliography

[1] Agarwal, A., Uppal, S., Shaw, K., Pathak, D.: Dexterous functional grasping (2023)

[2] Allshire, A., Mittal, M., Lodaya, V., Makoviychuk, V., Makoviichuk, D., Widmaier, F., Wüthrich, M., Bauer, S., Handa, A., Garg, A.: Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World TriFinger. In: IEEE/RSJ Int'l Conf on Intelligent Robots and Systems (IROS) (2022)

[3] Andrychowicz, O.M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., Zaremba, W.: Learning dexterous in-hand manipulation. The Int'l Journal of Robotics Research (IJRR) (2020)

[4] Bahety, A., Mandikal, P., Abbatematteo, B., Martín-Martín, R.: Screwmimic: Bimanual imitation from human videos with screw space projection. In: Robotics: Science and Systems (RSS), 2024 (2024)

[5] Bao, C., Xu, H., Qin, Y., Wang, X.: Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21,190–21,200 (2023)

[6] Brahmbhatt, S., Ham, C., Kemp, C.C., Hays, J.: ContactDB: Analyzing and predicting grasp contact via thermal imaging. In: IEEE Conf on Computer Vision and Pattern Recognition (CVPR) (2019)

[7] Chen, T., Xu, J., Agrawal, P.: A system for general in-hand object re-orientation. In: Conf on Robot Learning (CORL) (2022)

[8] Dasari, S., Gupta, A., Kumar, V.: Learning dexterous manipulation from exemplar object trajectories and pre-grasps. In: IEEE Int'l Conf on Robotics and Automation (2023)

[9] Fan, Z., Parelli, M., Kadoglou, M.E., Kocabas, M., Chen, X., Black, M.J., Hilliges, O.: HOLD: Category-agnostic 3d reconstruction of interacting hands and objects from video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 494–504 (2024)

[10] Fan, Z., Taheri, O., Tzionas, D., Kocabas, M., Kaufmann, M., Black, M.J., Hilliges, O.: ARCTIC: A dataset for dexterous bimanual hand-object manipulation. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)

[11] Freeman, C.D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., Bachem, O.: BRAX: A Differentiable Physics Engine for Large Scale Rigid Body Simulation. arXiv:2106.13281 (2021)

[12] Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Int'l Conf on machine learning. PMLR (2018)

[13] Georgiev, I., Srinivasan, K., Xu, J., Heiden, E., Garg, A.: Adaptive horizon actor-critic for policy learningin contact-rich differentiable simulation. In: International Conference on Machine Learning. PMLR (2024)

[14] Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., Su, H.: Maniskill2: A unified benchmark for generalizable manipulation skills. In: International Conference on Learning Representations (2023)

[15] Hansen, N., Wang, X., Su, H.: Temporal difference learning for model predictive control (2022)

[16] Heiden, E., Millard, D., Coumans, E., Sheng, Y., Sukhatme, G.S.: NeuralSim: Augmenting Differentiable Simulators with Neural Networks. In: IEEE Int'l Conf on Robotics and Automation (ICRA) (2021)

[17] Howell, T., Gileadi, N., Tunyasuvunakool, S., Zakka, K., Erez, T., Tassa, Y.: Predictive sampling: Real-time behaviour synthesis with mujoco. arXiv preprint arXiv:2212.00541 (2022)

[18] Howell, T.A., Cleac'h, S.L., Kolter, J.Z., Schwager, M., Manchester, Z.: DOJO: A Differentiable Simulator for Robotics. arXiv:2203.00806 (2022)

[19] Jiang, H., Liu, S., Wang, J., Wang, X.: Hand-object contact consistency reasoning for human grasps generation. In: IEEE/CVF Int'l Conf on Computer Vision and Pattern Recognition (CVPR) (2021)

[20] Lv, J., Feng, Y., Zhang, C., Zhao, S., Shao, L., Lu, C.: Sam-rl: Sensing-aware model-based reinforcement learning via differentiable physics-based simulation and rendering. arXiv:2210.15185 (2022)

[21] Macklin, M.: NVIDIA Warp (2023)

[22] Macklin, M., Müller, M., Chentanez, N.: XPBD: position-based simulation of compliant constrained dynamics. In: Int'l Conf on Motion in Games (2016)

[23] Makoviichuk, D., Makoviychuk, V.: rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games (2022)

[24] Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., State, G.: Isaac gym: High performance gpu-based physics simulation for robot learning (2021)

[25] Mandikal, P., Grauman, K.: Dexvip: Learning dexterous grasping with human hand pose priors from video. In: Conf on Robot Learning (CORL) (2022)

[26] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Int'l Conf on Machine Learning (ICML) (2016)

[27] Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[28] Mora, M.A.Z., Peychev, M., Ha, S., Vechev, M., Coros, S.: Pods: Policy optimization via differentiable simulation. In: Int'l Conf on Machine Learning (2021)

[29] Morgan, A.S., Nandha, D., Chalvatzaki, G., D'Eramo, C., Dollar, A.M., Peters, J.: Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning. In: IEEE Int'l Conf on Robotics and Automation (ICRA) (2021)

[30] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 (2017)

[31] Sutton, R.S., Barto, A.G., et al.: Introduction to reinforcement learning. MIT press Cambridge (1998)

[32] Turpin, D., Wang, L., Heiden, E., Chen, Y.C., Macklin, M., Tsogkas, S., Dickinson, S., Garg, A.: Grasp'd: Differentiable contact-rich grasp synthesis for multi-fingered hands. In: European Conf on Computer Vision. Springer (2022)

[33] Wang, R., Zhang, J., Chen, J., Xu, Y., Li, P., Liu, T., Wang, H.: Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. 2023 IEEE International Conference on Robotics and Automation (ICRA) pp. 11,359–11,366 (2022). URL https://api.semanticscholar.org/CorpusID:252734719

[34] Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A.X., Guibas, L.J., Su, H.: SAPIEN: A simulated part-based interactive environment. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[35] Xu, J., Chen, T., Zlokapa, L., Foshey, M., Matusik, W., Sueda, S., Agrawal, P.: An End-to-End Differentiable Framework for Contact-Aware Robot Design. In: Proceedings of Robotics: Science and Systems. Virtual (2021). DOI 10.15607/RSS.2021.XVII.008

[36] Xu, J., Makoviychuk, V., Narang, Y., Ramos, F., Matusik, W., Garg, A., Macklin, M.: Accelerated policy learning with parallel differentiable simulation. In: Int'l Conf. on Learning Representations (ICLR) (2022)

[37] Xu, Y., Wan, W., Zhang, J., Liu, H., Shan, Z., Shen, H., Wang, R., Geng, H., Weng, Y., Chen, J., et al.: Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. arXiv preprint arXiv:2303.00938 (2023)

[38] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., Levine, S.: Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In: Conference on robot learning, pp. 1094–1100. PMLR (2020)

[39] Zhong, Y.D., Han, J., Brikis, G.O.: Differentiable physics simulations with contacts: Do they have correct gradients w.r.t. position, velocity and control? (2022)