DOI: 10.1111/cgf.14741
EUROGRAPHICS 2023 / K. Myszkowski and M. Nießner
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 2

Learning to Transfer In-Hand Manipulations Using a Greedy Shape Curriculum

Yunbo Zhang[1], Alexander Clegg[3], Sehoon Ha[1], Greg Turk[1],Yuting Ye[2]

[1]Georgia Institute of Technology , [2]Meta Reality Lab, [3]Meta AI Research

Abstract

In-hand object manipulation is challenging to simulate due to complex contact dynamics, non-repetitive finger gaits, and the need to indirectly control unactuated objects. Further adapting a successful manipulation skill to new objects with different shapes and physical properties is a similarly challenging problem. In this work, we show that natural and robust in-hand manipulation of simple objects in a dynamic simulation can be learned from a high quality motion capture example via deep reinforcement learning with careful designs of the imitation learning problem. We apply our approach on both single-handed and two-handed dexterous manipulations of diverse object shapes and motions. We then demonstrate further adaptation of the example motion to a more complex shape through curriculum learning on intermediate shapes morphed between the source and target object. While a naive curriculum of progressive morphs often falls short, we propose a simple greedy curriculum search algorithm that can successfully apply to a range of objects such as a teapot, bunny, bottle, train, and elephant.

CCS Concepts
• ***Computing methodologies*** → ***Physical simulation; Motion capture;*** *Reinforcement learning; Learning from demonstrations;*

1. Introduction

Why do cartoon characters have four digits instead of five [BBC17]? It is because the hand is notoriously time-consuming to animate, whether in drawing or digitally, owing to its high degree of articulation. While finger motions can now be tracked by consumer devices as reference [Med, HLC*20], hand-object manipulation remains challenging to capture accurately and conveniently. Moreover, once captured, fixing and editing the result can often be as difficult and tedious as animating from scratch, because both the hand and the object need to move in coordination and convey the physical realism of the interaction. Among all hand-object interactions, in-hand manipulation is the hardest to animate and edit (when compared to other motions such as grasping and pick-n-place), due to frequent contact changes and the dynamic nature of the interaction.

In this work, we apply an imitation learning framework based on deep reinforcement learning (DRL) to physically simulate a reference motion of in-hand object manipulation. We further extend this learning framework to adapt the reference motion to target objects of different shapes and sizes than the source object. As a result, we can capture manipulations of simple primitive objects with the desired styles and actions, and then adapt the hand motion to manipulate more complex objects with various physical properties in a physically realistic way.

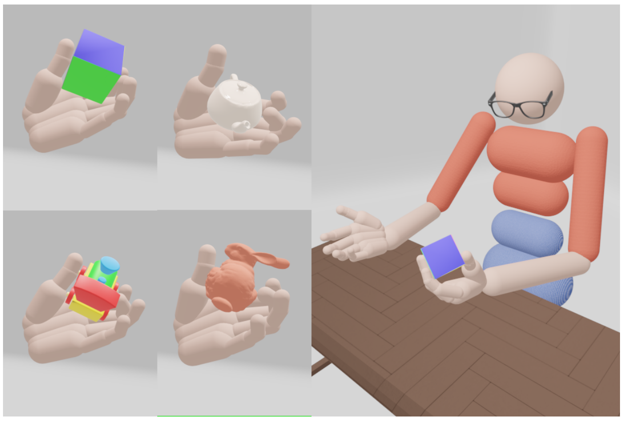


Figure 1: *Example of motion transfer from a cube to three different objects (left) and a two-hand manipulation (right).*

Our choice of DRL is inspired by its explosive success in simulating full body motions, including walking, running, jumping, ball bouncing and catching [WGH20, YTL18, LH18, MTA*20]. However, it has not been applied as widely to animations of object manipulation. The robotics community is successful in applying DRL to functional manipulation tasks with manipulators, such as object re-orientation or peg-in-hole [RKG*18, AAC*19, CHM*19], but with less emphasis on naturalness or realism of the hand motion. To achieve natural manipulations, the computer vision community instead captures the kinematic motions of hands interact-ing with objects from human performance [TGBT20, CYX*21, BTT*20, BHKH19, HLW*18]. The resulting datasets enable the synthesis of convincing grasping poses of a rich set of objects [TCBT22, WWZ*22, CKA*22, GTT*21], as well as dexterous and skillful finger maneuvers [ZYSK21]. However, these results are focusing on the kinematic motion and do not enforce proper physical constraints of the interaction. Our work benefits from high-quality and realistic motions of object manipulation, and we further enable physical simulation of the interactions such that our results are responsive to dynamic situations and are physically consistent.

An important challenge of object manipulation stems from the limitless possibility of object shapes. While human hands can effortlessly adapt to geometry variations, control policies have a harder time adjusting. Naive fine-tuning of an existing policy to new shapes is rarely successful. A sensible strategy is then to design a curriculum of increasingly difficult tasks to progressively guide the learning towards solving the target shape. A natural choice is then to use a series of shape morphs between the source and the target objects. Unfortunately, the task difficulty and training progress do not correlate with shape progression, which makes a naive linear curriculum ineffective. Instead, we take inspiration from the work of Wang et al. [WLCS19] that generates an open-ended curriculum with increasing complexity by co-evolving the policies and training environments. To this end, we propose a *Greedy Shape Curriculum* across the morph geometry variations. During training, we keep track of the most successful policy for each shape, and pick the most promising shape-policy pair for further training. After a few training epochs, we update the shape-policy pairs by testing the new policy on all shape morphs, then repeat the process. This strategy can effectively adapt an example manipulation of a simple primitive object to a diverse set of everyday objects automatically.

The success of a curriculum hinges on a capable learning framework. Ours is based on the popular DeepMimic formulation [PALvdP18] on locomotion tasks. When applying it to dexterous manipulations, however, we encountered several unique challenges. For instance, parameters of the contact model can greatly affect both the difficulty and realism of a learned policy. In our case, we carefully chose a set of stiff parameters so the resulting motions do not suffer from sinking or sliding contacts between the hand and the objects, and they work across all our examples without adjustments. We also found it more effective to let the policy output deltas from the reference motion as used in [BCHF19], instead of the common choice of direct PD targets [PALvdP18, WGH20]. In addition, defining an early termination criterion is more challenging for the control of an unactuated object, because small deviations in the object state could still become unrecoverable. With some deliberate design choices, we arrive at a robust imitation learning framework that can simulate natural and dexterous manipulation of a variety of objects. This framework serves as a solid foundation for the greedy shape curriculum.

Our three main contributions are:

- We create a framework with well designed simulation and learning configurations for robust dexterous hand manipulation.
- We propose a novel Greedy Shape Curriculum to automatically transfer an example manipulation of simple shapes to more challenging objects.
- We showcase that the proposed framework can learn robust policies on various manipulation tasks, even with dynamics variations.

2. Related Work

2.1. Hand Manipulation in Animation

Generating animation of realistic hand manipulation skills has been a long-studied topic in animation.Wheatland and colleagues [WWS*15] summarizes the state of the art in modeling and animation in hand manipulation problems. Early work exploited traditional optimal control theory to generate hand manipulation skills for rigid objects [Liu09, MPT12, BL14] or soft bodies [BYL16] by planning contact forces. Instead of solving an optimal control problem, Ye and colleagues [YL12] proposed a randomized sampling algorithm to find a set of diverse strategies. Zhang *et al.* [ZYSK21] developed a data-driven approach for synthesizing a variety of object manipulation motions via deep learning, but the generated motions are purely kinematic instead of simulated with physics. Reinforcement learning has also been used in generate manipulation animations. Andrews *et al.* [AK13] trains goal directed policies for one-handed manipulation without guidance from motion capture data. Another line of work has focused on synthesizing grasping motions for simulated hands [LFP07, CKA*22, HPSK21, PZ05] or whole-body characters [WWZ*22]. Inspired by prior work, we tackle the problem of hand manipulation by leveraging recent advances in DRL and physics-based simulation.

Our approach of adapting an existing manipulation motion to different objects can also be cast as a motion retargeting problem. Retargeting of complex interactions often considers the spatial relationship of two people [HKT10, JKL18], or between the person and the environment [AAKC13], or through analysis of the object geometry and affordance in the case of manipulations [SDT*22, TGBT20], without enforcing physical correctness in the results. Recent work starts to utilize geometry representation to improve the learning of manipulation control [SHX*22]. More recently, Yang *et al.* [YYL22] trains policies to control hands to manipulate chopsticks through reconstructing hand motions then then retarget to finish the manipulation tasks. We instead show that direct imitation learning can be successful in retargeting to different objects with a well designed curriculum.

2.2. Robotic Hand Manipulation

Dexterous hand manipulation has also been an essential topic in robotics. Early work [HM00, SI91, HGL*97, SH18] typically casts manipulation into motion planning problems, which allow robots to achieve various motions, such as tapping, tumbling, or rolling manipulations. However, these motion planning approaches often generate open-loop trajectories which cannot handle dynamic perturbations. This limitation has been overcome by incorporating tactile sensor feedbacks during manipulation [TAY10, LMH*13]. These methods can allow certain amounts of deviation from the planned trajectory, but they often require accurate dynamic models for both hands and objects.

Because of hand manipulation's complex and discontinuous nature, researchers have investigated deep reinforcement learning (DRL) algorithms that can solve challenging motor control problems. A common approach [ZGR*19, NKLK20, CM21, ABC*20, AAC*19] is to cast manipulation into an MDP directly to learn skills without human demonstrations. Andrychowicz *et al.* [ABC*20] demonstrated the successful learning of dexterous cube manipulation on a real robot hand by combining pose estimation and deep reinforcement learning. The work is extended to solve a Rubik's cube with a humanoid robot hand by applying proper randomization to system dynamics and designing a curriculum with increasing difficulty. Note that training times for these policies are high, with the Rubik's cube policy requiring several months of training on 64 V100 GPUs and 920 worker machines.

While focusing on a single type of object, other work [CXA22, HMAP21] trained a control policy in simulation to reorient a wide variation of geometries in hand, including the YCB dataset [CSB*17] and various toys. However, this approach of finding a manipulation policy without human demonstrations often leads to unnatural and jerky motion. Many researchers [GELA16, RKG*18, RWPM21, JSK*20, KGTL16] also have utilized human demonstration to complete dexterous manipulation tasks. For instance, Garcia *et al.* [GHJK20] developed a learning framework by estimating the needed contacts to accomplish the tasks on a physics simulator from human motions. Our approach also uses deep reinforcement learning to solve a dexterous hand manipulation problem from the captured hand and object movements.

2.3. Learning to Imitate Reference Motions

Since the pioneering work of Liu *et al.* [LYVdP*10, LYG15] and Peng *et al.* [PALvdP18], learning to imitate a given reference motion has been a promising approach for developing natural and effective physics-based motion controllers. Starting from a short motion clip of walking, running, or kicking, researchers [PKM*18, LPLL19, PCZ*20, PRL*19, CMM*18, MHG*19] have extended the motion imitation framework from various perspectives, including interactivity, data diversity, and character complexity. For instance, researchers trained an interactive controller to complete locomotion tasks from a wide range of walking motions. Won *et al.* [WGH20] invented a scalable motion imitation framework by employing an ensemble of expert networks. Lee *et al.* [LPLL19] trained a full-body muscular skeleton character to track the motion capture data by introducing a hierarchical controller. Researchers [WGH21] also developed a technique to solve control of multiple characters in competitive games with a manually designed multi-stage learning. Some prior work [LLLL21, WL19] have discussed how to adapt policies to variants of the training mocap sequences, rather than simply following the given reference motion. Our work also employs the motion imitation framework while applying it to dexterous hand manipulation, which has not yet been fully investigated with motion imitation.

2.4. Curriculum learning

Curriculum learning [BLCW09] is one of the common techniques for solving challenging problems. It generates tasks with increasing difficulties until it finds solutions for the most difficult scenarios. Many prior works [KP12, HTS*17, YCBVdP08, TKH21] design a curriculum manually using prior knowledge to train locomotion controllers for complex tasks. Instead of a naive curriculum, recent works [WL19, XLKvdP20] propose adaptive curriculum methods based on the value function of the trained policy. In another closely related line of work, researchers [WLCS19, WLR*20] propose algorithms to design an open-ended curriculum for policies and environments without specifying any target task. In our work, we take inspiration from these ideas and propose a new curriculum learning framework, Greedy Shape Curriculum, to generate complex manipulation trajectories with various objects.

3. Simulation Setup

We perform physics simulation using Mujoco [TET12] inspired by successful prior work in hand manipulation [GHJK20, ABC*20].

3.1. Motion Capture Sequences

We use the object manipulation motion capture dataset from Zhang et al. [ZYSK21]. The data was captured at 120 frames per second with the Optitrack [Opt22] system using reflective markers. A total of 19 markers are placed on each hand: four on the thumb, three on each finger, and three more on the back of the hand. The Optitrack system outputs object motions as rigid bodies, and the finger motions are reconstructed using DeepLabels [HLW*18].

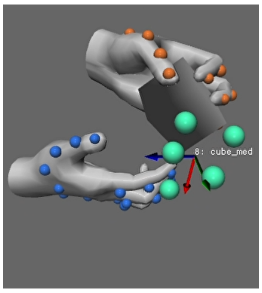


Figure 2: *A still frame from an input motion capture sequence.*

3.2. Hand and Object Models

The dataset comes with a hand model and object models. The hand model consists of 20 degrees of freedom, and we model each finger segment as capsules (see Figure 3). For efficient collision detection in the simulation, objects are modeled either as convex primitive shapes such as a cube or a cylinder, or approximated as a combination of convex primitives if they are nonconvex (e.g. torus and wine glass). For simplicity purpose, we only enable the collision between the hand and the object. Self collision within the hand are disabled during the manipulation training.

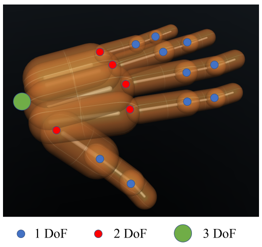


Figure 3: *Hand model with* 20 *rotational axes.*

3.3. Contact Parameters

We find that the contact-rich nature of in-hand manipulation tasks demands greater care when choosing simulation parameters related to collision resolution. Mujoco formulates contacts as a constrained optimization problem rather than a linear complementary problem to control the compliance of contact surfaces. In all of our experiments, we choose Mujoco built-in contact parameters that correspond to highly stiff surfaces. Specifically, we set the fixed constraint impedance to 0.95. We also reduce the solver's time constant from 0.02 to 0.005, which is the inverse of the natural frequency times the damping ratio. Unfortunately, such stiff contacts increase the instability of the simulator. As a mitigation, we choose elliptic friction cones and the RK4 integrator, and run the simulation at 600Hz. We additionally found slipping could be a common failure. To increase friction, we set the ratio of friction-to-normal contact impedance to 5, and strengthen the friction forces with higher normal forces. For more details of the contact model, please refer to the official documentation [TET12]. We found the above settings empirically works over all the motion sequences we tried, allowing us to achieve effective dexterous manipulation without the fingers "sinking" into the object's surface or sliding over sharp edges.

4. Motion Imitation

The foundation of our algorithm is an imitation learning framework. In the following sections, we describe the observations, actions, reward functions, and the training procedure. We explain these for a single hand, but our method can trivially incorporate both hands by increasing the state and observation dimensions.

4.1. Problem Formulation

Given motion capture trajectories for a hand (or both hands) and an object, we aim to learn an effective policy that can manipulate the object by following a reference trajectory.

We formulate the control problem as a partially observable Markov Decision Process (PoMDP) with tuple $(\mathcal{S}, \mathcal{O}, o, \mathcal{A}, \mathcal{T}, r, o, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{O}$ is the observation space for the hand and the object, $\mathcal{A}$ is the action space for actuating the hand, $\mathcal{T}$ is the transition dynamics (physics simulation), r is the reward function, o is the observation emission function, and γ is the discount factor. At a high level, we want to find a parameterized control policy π_θ so that it will maximize the expected sum of rewards over a distribution of trajectories

$$\pi_{\theta^*} = \arg\max_{\theta} \mathbb{E}_{(s_0, s_1, \ldots, s_T)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, \pi_\theta(s_t)) \right]. \tag{1}$$

4.2. Observation representation

The observation space can be divided into four components: the states of the simulated hand and the object $\{\mathbf{x}_{hand}, \mathbf{x}_{obj}\}$, reference states of hand and object from mocap $\{\bar{\mathbf{x}}_{hand}, \bar{\mathbf{x}}_{obj}\}$, their differences from simulation $\{\bar{x}_{hand} \ominus x_{hand}, \bar{x}_{obj} \ominus x_{obj}\}$, and contact information $\{\mathcal{C}\}$. As studied in Bergamin et al. [BCHF19], using the difference between simulation and reference as observation improves both training speed and quality.

Simulated states: The positional state of the hand $\mathbf{x}_{hand} = (\mathbf{x}_{h\_root}, \mathbf{R}_{h\_root}, \cos(q), \sin(q))$ is a $46D$ vector containing the pose of the root and all joint angles of the hand, where $\mathbf{x}_{h\_root} \in \mathbb{R}^3$ is the $3D$ position of the wrist, $\mathbf{R}_{h\_root} \in \mathbb{R}^3$ orientation of the wrist represented as axis-angle, and $q \in \mathbb{R}^{20}$ are all the joint angles in a hand. $\mathbf{x}_{obj} = (\mathbf{x}_{o\_root}, \mathbf{R}_{o\_root})$ is a $6D$ vector containing the position and axis-angle orientation of the simulated object. $\mathbf{v}_{hand}$ and $\mathbf{v}_{obj}$ are $26D$ and $6D$ vectors containing the velocities of the hand and the object. Note that we actuate the hand's orientation directly, and we found the use of axis-angle for hand orientation to be especially important for success.

Reference states: $\bar{\mathbf{x}}_{hand}$ and $\bar{\mathbf{x}}_{obj}$ are the reference poses of the hand and object at the current frame expressed in the same format as the pose of simulated hand and object.

State differences: $\bar{x}_{hand} \ominus x_{hand}$ and $\bar{x}_{obj} \ominus x_{obj}$ are the differences between the simulated pose and the reference pose of both the hand and object. For all the rotational information, we evaluate the rotation differences in $SO(3)$ and express the difference into the corresponding format of axis angle or sine and cosine of Euler angles used in the observation.

Contact information: We include an additional $19D$ vector $\mathcal{C}$ to capture the contact forces between the hand and the object. We surround each finger capsule with a contact sensor that is a slightly larger capsule with a 10% larger radius. These contact sensors register contact forces from the object at each control step, and we record the sum of all contact forces exerted by the object on each rigid segments of the hand through the contact sensors. Because the sensors correspond to finger segments, their readings indicate how much contact forces are being exerted, and implicitly inform where the contacts are.

Combining all the described observation components, we have a $207D$ vector that describes the state of the simulation when we are considering a manipulating task involving a single hand. When we train the policy to track a two-hand manipulation sequence, we double the observations for the hand and end up with a $390D$ vector.

4.3. Action Representation

Similar to the approach in [BCHF19], at each time step t, the policy outputs an action $\mathbf{a}_t = \{\Delta\mathbf{x}, \Delta\mathbf{R}, \Delta\mathbf{q}\}$, a $26D$ vector specifying the spatial displacement to the hand's reference pose, where $\Delta\mathbf{x} \in \mathbb{R}^3$ is the hand root linear displacement, $\Delta\mathbf{R} \in \mathbb{R}^3$ is the root orientation displacement expressed in axis-angle, and $\Delta\mathbf{q} \in \mathbb{R}^{20}$ is the joint angle displacement for each joint. We apply an exponential action filter with $\alpha = 0.3$ to generate smoother motions. Once the PD target is computed, we compute joint torques using the stable-PD controller [TLT11]. The full control loop is shown in Figure 4

4.4. Reward Function

Our goal is to track the reference motions of both the hands and the object as closely as possible. Inspired by the original DeepMimic paper [PALvdP18], we design our reward function as follows:

$$r = w_{od} r_{od} + w_{or} r_{or} + w_{hd} r_{hd} + w_{hr} r_{hr} + w_{hj} r_{hj} \tag{2}$$

which consists of the object position term r_{od}, the object rotation term r_{or}, the hand position term r_{hd}, the hand orientation term r_{hr}, and the hand joint term r_{hj}. To enforce a match between the simulated object and the reference object's position and orientation, we

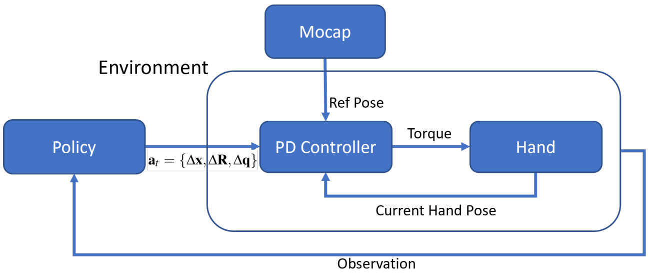


Figure 4: *Overview of the control loop*

define the terms r_{od} and r_{or}:

$$r_{od} = \exp\left(-k_{od}\|\hat{x}_{obj} - x_{obj}\|^2\right), \tag{3}$$

and

$$r_{or} = \exp\left(-k_{or}\|\hat{q}_{obj}^{-1}q_{obj}\|^2\right), \tag{4}$$

which compares the object's position x_{obj} and orientation q_{obj} to their desired values. For all N rigid segment of the hand with the index i, we define the reward terms r_{hd} and r_{hr}:

$$r_{hd} = \exp\left(-k_{hd}\sum_{i=1}^{N}\|\hat{x}_i - x_i\|^2\right), \tag{5}$$

and

$$r_{hr} = \exp\left(-k_{hr}\sum_{i=1}^{N}\|\hat{q}_i^{-1}q_i\|^2\right), \tag{6}$$

where x_i and q_i represent the position and orientation of the ith body segment. In addition enforcing the hand rigid segment's tracking, we also define the reward term r_{hj}

$$r_{hj} = \exp\left(-k_{hj}\sum_{i=1}^{M}\|\hat{\theta}_i - \theta_i\|^2\right), \tag{7}$$

by comparing all the current and desired joint angles, θ and $\hat{\theta}$. For all experiments, we set the weights as $w_{od} = 4$, $w_{or} = 4$, $w_{hd} = 0.05$, $w_{hr} = 0.05$, and $w_{hj} = 0.1$.

4.5. Terminal Condition

As studied in DeepMimic [PALvdP18], early termination of a rollout when the simulation enters an unrecoverable state can save computation on low value trajectories. We design the early termination criteria to restrict how much the object's state is allowed to deviate from the reference: either $d_{thr} = 10cm$ in translation or $\phi_{thr} = 60^\circ$ in rotation. We choose these thresholds to allow the hands to explore its action space more freely, but this still eliminates irredeemable failures.

5. Greedy Shape Curriculum for Novel Objects

It would be undesirable to record a new motion capture sequence for each new object that we want to manipulate. Instead, we would like to generalize an existing motion example to different objects in simulation. For example, we may want to manipulate a teapot or a toy train using the same reference motion for a cube. However, it is not straightforward to learn an effective policy for a new shape because it often requires significant changes in the control strategy.

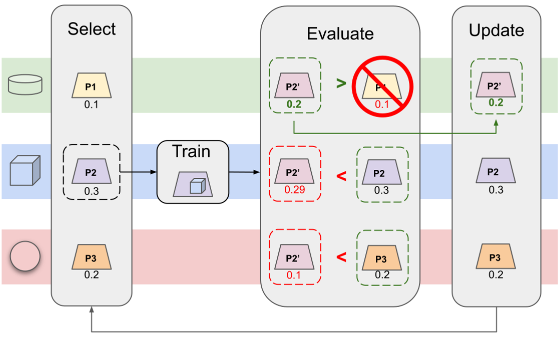


Figure 5: *Illustration of our greedy shape curriculum. Each iteration of the algorithm (1) selects and trains the most promising (policy, shape) pair, (2) evaluates the updated policy on all shapes, and (3) overwrites a shape's policy pairing if the new policy is better than the cached policy. Example policy performance metrics are displayed numerically below each policy shape.*

Our key intuition is that we can co-train policies on a set of intermediate shapes morphing between the original object and the target object as a curriculum. A naive method would be to use a training curriculum that starts the learning from the source object and gradually morph the shape to the target in a linear progression. In practice, however, this linear curriculum is often unsuccessful because the morphing progression may not exactly correlate to the task's difficulty. A better tuned morphing algorithm might be able to give stronger correlation between morphing progression and training difficulty, but such algorithm requires additional human effort, and may not be able to generalize across different source target pairs.

Instead, we design a novel training schedule that allows greedily switching between any intermediate shape morphs for a more flexible curriculum. Our algorithm maintains a collection of best policies for each shape. For every $K = 20$ policy iteration, it selects the best performing *unsuccessful* morph and its paired policy for the next round of training. Once the policy is further trained, the newly updated policy's performance is evaluated across the entire collection of shapes, and overrides existing policies if it performs better (Figure 5). Despite its greedy nature, we found this automated curriculum more effective in policy transfer than naive fining-tuning or a fix curriculum. The full procedure is described in Algorithm 1.

A key component of our greedy shape curriculum is a *goodness score* that describes how likely a policy will succeed on a given shape. This metric will be used to update the best policy for a shape, and for selecting the next (shape, policy) pair for training. An obvious choice would be the average episodic reward. However, the consideration here is slightly different. A high episodic reward imposes a more strict constraint to the quality of object pose match-

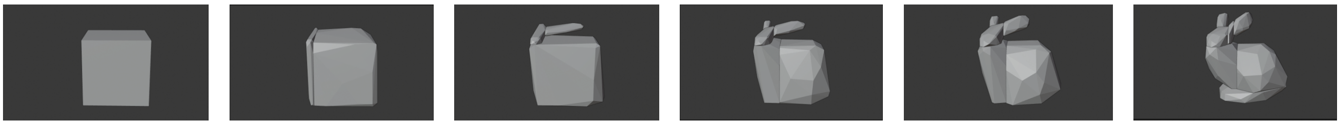


Figure 6: *Morph stages of the collision shapes for transferring the cube motion to a bunny after applying V-HACD.*

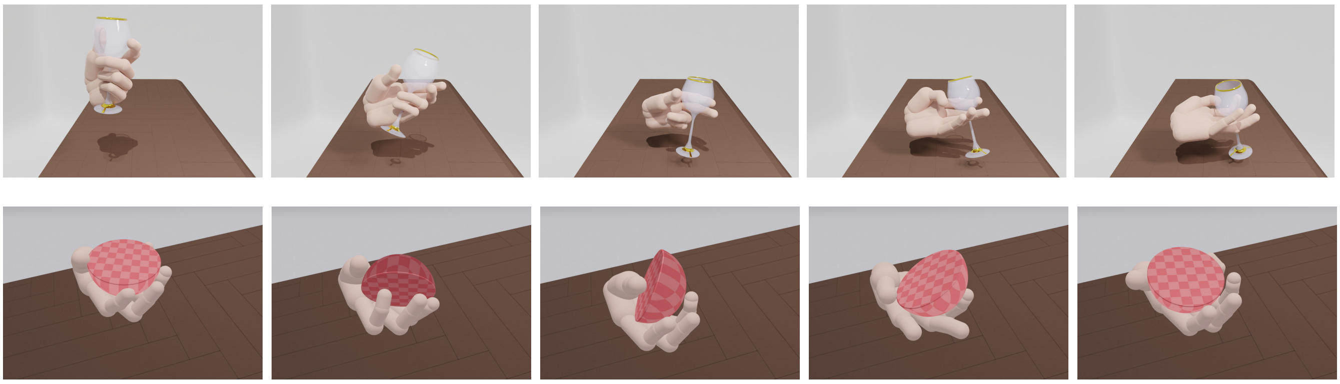


Figure 7: *Still frames from manipulation sequences involving a wineglass (**top**) and a hemisphere (**bottom**).*

ALGORITHM 1: Greedy Shape Curriculum

```
Initialize score list $S$
Initialize policy list $\Pi$
Initialize current shape $s = 0$ and current policy $\pi = \Pi[s]$
for $i = 0, 1, 2, \ldots$ do
    if $i \mod k == 0$ then
        for Every shape $j$ do
            score = rollout on $j$ using policy $\pi$
            if score > $S[j]$ then
                $S[j]$ = score
                $\Pi[j] = \pi$
            end if
        end for
        $s$ = Get best unsuccessful shape
        $\pi = \Pi[s]$
    end if
    PPO using $s$ and $\pi$
end for
```

ing, making it hard to achieve when the target shape is too different from the source shape. A low episodic reward, on the other hand, cannot guarantee the completion of a rollout. On the other hand, the rollout length alone is too simple and fails to reflect the quality of the motion. We want a criteria with high tolerance to object deviation but with low tolerance to failure of completion. To this end, we design our criteria as a combination of the rollout duration and tracking accuracy, and this works robustly in practice. As described in Equation 8, we use the product between the normalized episode length and the sum of hand joint reward of the rollout as the goodness score of a policy for a given shape. This encourages the resulting policy to use a similar manipulation strategy to the input. We consider a score higher than $d = 0.55$ as *successful*, and we only pick from the unsuccessful shape morphs for policy training to make progress.

$$f = \frac{L}{T} \cdot \frac{\sum_0^L r_{joint}}{T} \tag{8}$$

Our greedy schedule is effectively an exploitation strategy, and we still need to balance it with some exploration to avoid local minima. If a particular shape is repeatedly picked for training and starving other shapes, we instead randomly select another shape and its paired policy in the next iteration. This is especially helpful when a policy gets “stuck” on a challenging frame towards the end of a sequence while other shapes have not had much training yet. Training progress on other shapes can then help improve such challenging cases later. Similarly, if we are successful on all shapes before the compute budget has been reached, we randomly pick a policy to continue training for further improvement.

6. Results

To validate our approach, we evaluate the performance of our policy on a variety of mocap sequences. These include single-handed manipulations of several distinct objects, as well as passing and rotating an object between two hands. We show that the resulting policies can endure a moderate amount of dynamic perturbations. We also demonstrate successful transfer of manipulations to novel everyday objects using our greedy shape curriculum. Please refer to the accompanying video for visual evaluation.

For all of our results, we use Proximal Policy Optimization (PPO) [SWD*17], a common on-policy reinforcement learning algorithm. The policy architecture is a fully connected network with two hidden layers, each of which has 256 units with *tanh* activation functions. Each policy is trained with 32 million observation/action pairs over 800 iterations. This training setting is held consistently across all trainings for original object sequences and shape morphs.

Because our source shapes are convex, we can use a simple

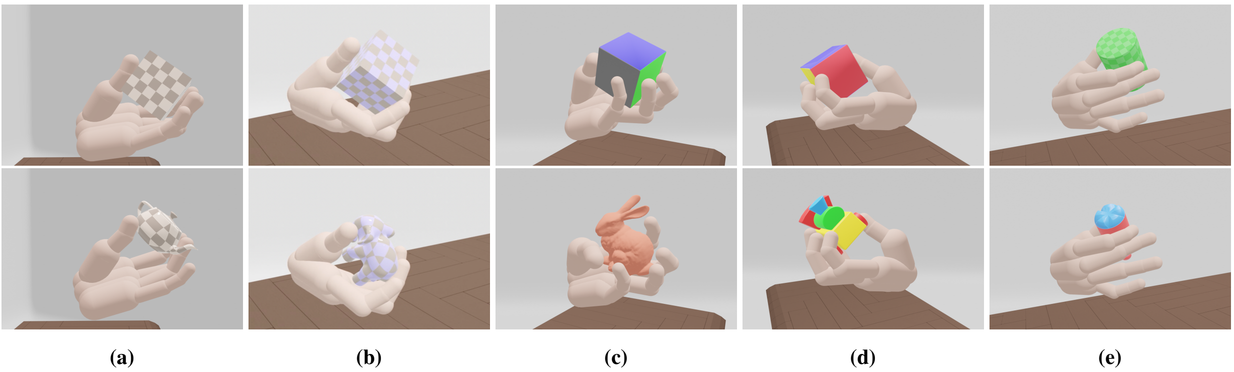


Figure 8: *Examples of transferring the manipulation of an original shape (top) to a target shape (bottom). Left to right, the examples shown are: cube to teapot, cube to elephant, cube to bunny, cube to toy train, and cylinder to bottle.*

projection-based shape morphing technique in order to generate intermediate shapes. To create the morphs, we project all vertices from the target shape to the surface of the source shape. We then use the intermediate positions along the projection paths as the vertex positions for each morph. For each shape pair, we create four intermediate morphs with vertices linearly interpolated between the corresponding positions on the source and target shapes. To create Mujoco compatible models, we use the commonly used convex decomposition method V-HACD [MLP16] with a maximum voxel resolution of $40 \times 40 \times 40$ to generate a collection of convex parts for each of the intermediate morphs. All the morphing operations are retrieved from a pre-processing stage in Blender [The02]. Figure 6 shows an example of the collision shapes of the source, target, and all intermediate morphs between a cube model and a bunny model. In later sections, we refer to each shape by its interpolation distance from the original shape along with the shape name. For any intermediate shape, the shape name is referred to as "Morph". For example, the sequence of morphs from the cube to the bunny are denoted as Cube(0.0), Morph(0.2), Morph(0.4),Morph(0.6),Morph(0.8), and Bunny(1.0).

We conduct our experiments on AWS C4 compute nodes with 36 virtual cores on each machine. Each learner process executes 800 iterations of PPO, collecting 32 million rollout samples in total with 8 parallel worker processes at a rate of 350 frames per second. This results in an end-to-end policy training duration of about 40 hours, depending on the complexity of the object's geometry and the number of contacts.

6.1. Reproducing single dynamic sequences

We first demonstrate the success of our approach across a diverse set of sequences involving a variety of objects and manipulation skills. First, we train a set of policies to track single-handed manipulation sequences, involving different objects over a 4-second horizon. These results indicate that our approach is successful for sequences involving both primitive convex objects such as cubes, cylinders, and hemispheres, as well as more complex, concave geometries such as a torus and a wineglass. Figure 7 (**top**) shows still frames from the manipulation of a wineglass, an example of a non-convex object, and (**bottom**) shows still frames from rotating a hemisphere in the hand. Additionally, we show that our approach can be trivially extended to control two-handed manipulations that require coordination, such as passing and cooperative rotation shown in Figure 1.

Even though each policy is trained to follow one object manipulation sequence, it is able to endure moderate dynamic perturbations. To demonstrate this, at random frames through the simulation, we apply forces of $8N$ on all fingertips for 0.25 second. Despite these perturbations, the policies adapt to these forces and still successfully track the motion. Similarly, a policy can easily track a motion sequence when modifying the masses and friction coefficients of the manipulated objects. These results are best seen in the supplementary video.

We summarize a quantitative analysis of success rates and mean reward of our final policies across the full set of motions in Table 1. To compute each entry in the table, a sample population of 500 episode rollouts are initialized from the first frame of each motion capture sequence, and stochastic policy actions are applied until the final frame is reached successfully or the object tracking error crosses a threshold value. For a more detailed analysis and comparison, we cache the episode length of each trial, and plot the percentage of the rollouts that meet or exceed a range of rollout length thresholds. In our experience, a success rate was too simple to capture the performance of the policy for our problem because

Table 1: *Success rate on example sequences with stochastic policy execution (500 samples). The maximum cumulative reward is 500.*

Experiment	Sequence	Success %	Cumulative Reward
Single Hand	Cube 1	94	463.16
	Cube 2	81.6	415.18
	Cube 3	98.6	480.45
	Cylinder 1	75.6	403.88
	Hemisphere 1	88.6	431.58
	Torus 1	85	432.55
	Wineglass 1	94.2	458.99
Two Hand	Cube Passing	82.6	417.66
	Cube Rotation	99.6	482.63

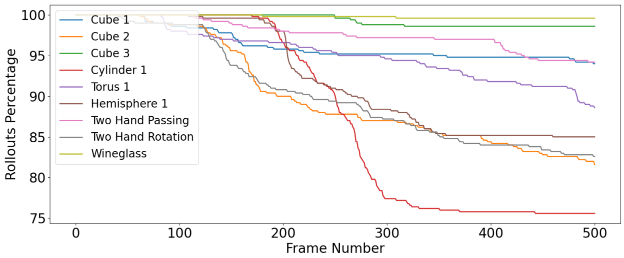


Figure 9: *Completion percentage of all sequences. This plot illustrates the relationships between the difficulty of a given frame and the associated success rate. The least performing sequence (Cylinder1) has more than 75% success rate to complete the entire rollout.*

it does not capture various difficulties of different frames. To this end, we plot "completion percentage" that shows the percentage of successful policies (Y-axis) to reach the given frame (X-axis). We summarized completion percentages in Figure 9. The figure shows how difficult each sequence is, and how often a final trained policy can successfully track the object to the end of the sequence. For instance, the cylinder 1 motion shows a significant drop of the completion percentage around frame 300, where thumbs are having trouble rotating a standing cylinder back into the palm, and thus where the cylinder may fall out of the hand.

6.2. Transferring to Complex Objects using a Greedy Shape Curriculum

We demonstrate that using our greedy shape curriculum, we can easily train policies that can manipulate both the source object and the target object by tracking a provided motion clip. For example, Figure 8 shows the simulation result after adapting the manipulation of a simple shape (e.g. cube or cylinder) to a more complex object (teapot, elephant, bunny, toy train, bottle).

We compare our method with a naive curriculum learning method where a policy will only proceed to the next shape morph when the previous one receives a goodness score above the threshold. Two examples are shown in figure 10. In the figures, blue dashed lines indicate the policy training progression, and the black dots indicate which shapes are receiving a better score when evaluating the current policy. We highlight the first success of each shape in green dots labeled with the respective score. Greedy shape curriculum results are shown on the top row, and the naive curriculum results are on the bottom row.

Figure 10 (a) shows an example in which the intermediate morphs do not introduce a smooth linear curriculum. The naive curriculum (bottom row) spends the majority of its compute budget on Morph(0.2) and eventually stops at Morph(0.8) without reaching the target shape. In contrast, the greedy shape curriculum (top row) skips training on these two difficult morphs, yet still finds a successful policy for the target shape by utilizing the other intermediate morphs. Training starts on Morph(0.4) and finishes much sooner than the 800 iteration budget at iteration 320, by alternating among Cube(0.0), Morph(0.6), and the target Bunny(1.0). The two difficult shapes, Morph(0.2) and Morph(0.8), do not get trained on and are not successful at the end.

Our greedy shape curriculum can also be more efficient than the naive curriculum in easy cases when both methods succeed. In Figure 10 (b), the greedy shape curriculum picks the target shape Elephant(1.0) for training at iteration 80 by inheriting the policy trained on Morph(0.6) at iteration 20. After further training, it is able to successfully manipulate the elephant at iteration 120. On the other hand, the naive curriculum has to receive high goodness score on all five previous shapes before succeeding on the target shape Elephant(1.0) at iteration 240.

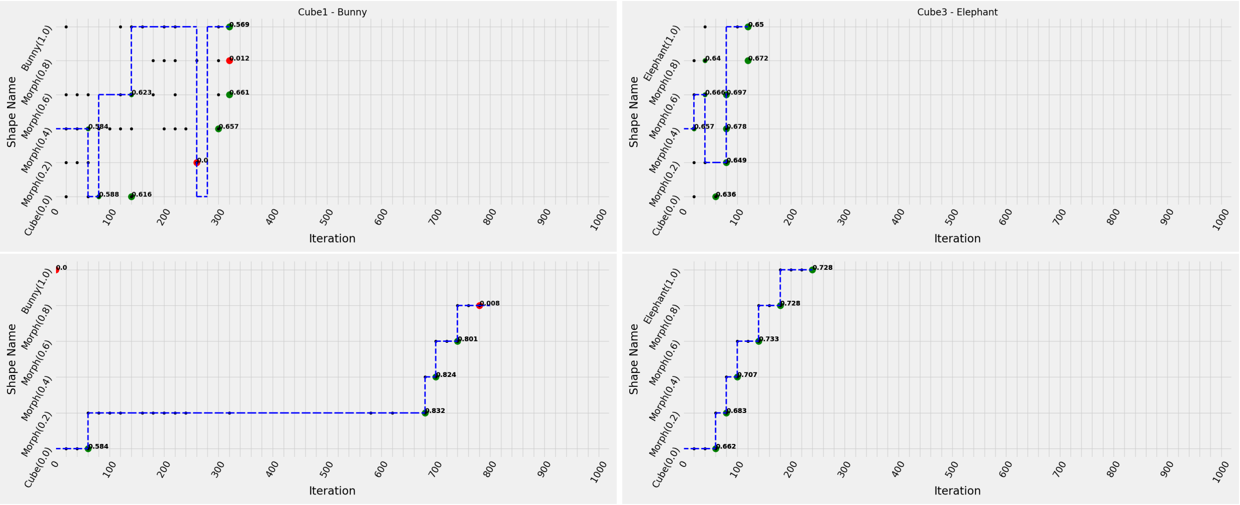
To highlight the benefits of our method, we compare greedy shape curriculum with four different baselines: 1) Training a single policy on the target shape; 2) Training a single policy over both the source and the target shapes; 3) Training a single policy over the collection of morphs from the source and the target shapes; 4) Training a collection of policies using a naive curriculum over all morphs. For each of the baselines, we take the mean policies trained from four random seeds, apply them on the target shape, and record whether or not they can successfully complete the rollout without hitting the early termination criterion. All trials are trained using 32 million samples over 800 iterations. As shown in Table 2, Greedy Shape Curriculum has the highest likelihood of producing working policies for all the target shapes under a fixed sample budget. In comparison, training on the target shape directly has a lower chance of success. Using a naive curriculum may be helpful in some cases but could be harmful in others, because only a subset of the sample budget is allocated to improve the target shape. On the other hand, training one generalized policy on multiple shapes is making the problem more challenging and is therefore more difficult to succeed, especially when the compute budget is limited.

6.3. Refining successful policies

Due to the simplification on disabling the self collision within the hand, the generated animation sometimes have the hand interpenetrating between the fingers. To mitigate the issue, we take the trained policy on target morph from the result of Greedy shape curriculum, and refine the training with a hand model with self collision enabled. This simple refinement can often result in more physically realistic motion qualities for the manipulation. A comparison is shown in Figure 11.

7. Limitations

Although we have been successful in tracking reference manipulations, we still sometimes observe noticeable penetrations between the hand and the object. Such artifacts are due to the contact handling mechanism in the physics engine. They are more pronounced when the object has a narrow or thin feature, such as the stem of the wine glass. In addition, we found that having a self-collision enabled hand makes the physics simulation very fragile that rollouts can easily get early terminated during training. Successful policies on target morphs can hardly be trained under this setup, and we end up disabling the self-collisions within the hand, and lead to some finger interpenetration in our results. Even though further refining



(a) **(b)**

Figure 10: *Comparison between our greedy shape curriculum (top) and naive curriculum (bottom) method. Part (a) shows policy training for cube-to-bunny, and part (b) shows cube-to-elephant. Note that our method often trains more quickly than the naive approach. The graph at the bottom of (a) shows failure of the naive approach to even find a solution. The blue dashed lines show which shape is being trained at each iteration. Black dots indicate which shapes can receive higher goodness score when using the currently trained policy. Green dots indicate the first successful iteration with the corresponding goodness score, and red dots means the shape was not able to be solved when training completes.*

Table 2: *Comparing the success rates of our method against four different baseline methods. Each method is trained with four random seeds.*

Original Sequence - Target Shape	Direct Target	Source + Target	All morphs	Naive Curriculum	Greedy Shape Curriculum
Cube1 - Teapot	50%	0%	0%	50%	**75%**
Cube1 - Bunny	**100%**	0%	0%	50%	**100%**
Cube1 - Train	**100%**	50%	0%	75%	**100%**
Cube2 - Bunny	50%	0%	0%	0%	**100%**
Cube2 - Elephant	**25%**	0%	0%	0%	**25%**
Cube3 - Elephant	50%	0%	0%	**100%**	**100%**

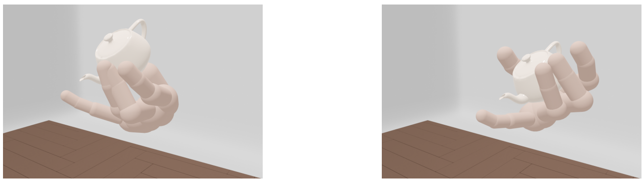


Figure 11: *Interpenetration between fingers (**left**) is resolved after refining the policy with self-collisions enabled (**right**).*

the resulting policies may mitigate the artifacts, it's not guaranteed to reach an interpenetration-free rollout.

We are still limited in how much the target shape can deviate from the source shape, and we cannot guarantee to always find a successful policy. With extreme changes in shape or size, the original manipulation strategy as captured may no longer be suitable or feasible. Related to this, our method cannot discover and explore novel manipulation skills that deviate significantly from the input. Even when successful (eg. the object is not dropped), the resulting finger motions may at times appear unnatural. The simple imitation term in the reward function and in the goodness score trades off exploration and exploitation, and could become a limiting factor in inventing new ways to interact with an object. An Adversarial Motion Prior [PMA*21] could be a promising mitigation.

Lastly, we found that some mocap sequences are particularly difficult to simulate because of noisy or erroneous frames. Although the motion imitation can learn a robust policy and "correct" these invalid frames via physics simulation, they tend to significantly slow down the learning process.

8. Conclusion

We have demonstrated training of policies for in-hand manipulation of objects based on motion capture data. Moreover, using a greedy shape curriculum, we can also adapt a given motion to a new shape. Because we use physics simulation, any disturbances or changes in physical properties are reflected in the object and hand motions. Our results closely mimic the fluidity and naturalness of real hands in the mocap examples.

One interesting direction for future work would be to reuse the existing motion clips for generating novel manipulation motions. Building a manipulation motion graph [LPLL19] to smoothly join manipulation mocap clips would be one possible approach. Unlike locomotion data where activities are highly repetitive, it is much harder to find suitable transition points with similar object and contact states, making this a challenging direction to explore.

Currently, policies generated from our method are only specialized to the target shape that it is trained on. We believe it is a first step towards solving a universal policy that can generalize to arbitrary shapes and task goals, which would be fruitful future work

9. Acknowledgement

We acknowledge Noah Maestre for his help on preparing the rendering setup of the animations. This work is partially supported by Meta Reality Lab.

References

[AAC*19] Akkaya I., Andrychowicz M., Chociej M., Litwin M., McGrew B., Petron A., Paino A., Plappert M., Powell G., Ribas R., et al.: Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113* (2019). 1, 3

[AAKC13] Al-Asqhar R. A., Komura T., Choi M. G.: Relationship descriptors for interactive motion adaptation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2013), SCA '13, p. 45–53. 2

[ABC*20] Andrychowicz O. M., Baker B., Chociej M., Jozefowicz R., McGrew B., Pachocki J., Petron A., Plappert M., Powell G., Ray A., et al.: Learning dexterous in-hand manipulation. *The International Journal of Robotics Research 39*, 1 (2020), 3–20. 3

[AK13] Andrews S., Kry P. G.: Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics 37*, 7 (2013), 830–839. 2

[BBC17] BBC: Disney animators explain why cartoon characters only have three fingers, 2017. URL: https://www.bbc.com/news/av/entertainment-arts-39479802. 1

[BCHF19] Bergamin K., Clavet S., Holden D., Forbes J. R.: Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG) 38*, 6 (2019), 1–11. 2, 4

[BHKH19] Brahmbhatt S., Ham C., Kemp C. C., Hays J.: Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 8709–8719. 2

[BL14] Bai Y., Liu C. K.: Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 1560–1565. 2

[BLCW09] Bengio Y., Louradour J., Collobert R., Weston J.: Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (2009), pp. 41–48. 3

[BTT*20] Brahmbhatt S., Tang C., Twigg C. D., Kemp C. C., Hays J.: ContactPose: A dataset of grasps with object contact and hand pose. In *The European Conference on Computer Vision (ECCV)* (August 2020). 2

[BYL16] Bai Y., Yu W., Liu C. K.: Dexterous manipulation of cloth. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 523–532. 2

[CHM*19] Chebotar Y., Handa A., Makoviychuk V., Macklin M., Issac J., Ratliff N., Fox D.: Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 8973–8979. 1

[CKA*22] Christen S., Kocabas M., Aksan E., Hwangbo J., Song J., Hilliges O.: D-grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20577–20586. 2

[CM21] Charlesworth H. J., Montana G.: Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning. In *International Conference on Machine Learning* (2021), PMLR, pp. 1496–1506. 3

[CMM*18] Chentanez N., Müller M., Macklin M., Makoviychuk V., Jeschke S.: Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th annual international conference on motion, interaction, and games* (2018), pp. 1–10. 3

[CSB*17] Calli B., Singh A., Bruce J., Walsman A., Konolige K., Srinivasa S., Abbeel P., Dollar A. M.: Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research 36*, 3 (2017), 261–268. 3

[CXA22] Chen T., Xu J., Agrawal P.: A system for general in-hand object re-orientation. In *Conference on Robot Learning* (2022), PMLR, pp. 297–307. 3

[CYX*21] Chao Y.-W., Yang W., Xiang Y., Molchanov P., Handa A., Tremblay J., Narang Y. S., Van Wyk K., Iqbal U., Birchfield S., et al.: Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 9044–9053. 2

[GELA16] Gupta A., Eppner C., Levine S., Abbeel P.: Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), IEEE, pp. 3786–3793. 3

[GHJK20] Garcia-Hernando G., Johns E., Kim T.-K.: Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 9561–9568. 3

[GTT*21] Grady P., Tang C., Twigg C. D., Vo M., Brahmbhatt S., Kemp C. C.: Contactopt: Optimizing contact to improve grasps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 1471–1481. 2

[HGL*97] Han L., Guan Y.-S., Li Z., Shi Q., Trinkle J. C.: Dextrous manipulation with rolling contacts. In *Proceedings of International Conference on Robotics and Automation* (1997), vol. 2, IEEE, pp. 992–997. 2

[HKT10] Ho E. S. L., Komura T., Tai C.-L.: Spatial relationship preserving character motion adaptation. 2

[HLC*20] Han S., Liu B., Cabezas R., Twigg C. D., Zhang P., Petkau J., Yu T.-H., Tai C.-J., Akbay M., Wang Z., Nitzan A., Dong G., Ye Y., Tao L., Wan C., Wang R.: MEgATrack: Monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (TOG) 39*, 4 (2020). 1

[HLW*18] Han S., Liu B., Wang R., Ye Y., Twigg C. D., Kin K.: Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–10. 2, 3

[HM00] Huang W. H., Mason M. T.: Mechanics, planning, and control for tapping. *The International Journal of Robotics Research 19*, 10 (2000), 883–894. 2

[HMAP21] Huang W., Mordatch I., Abbeel P., Pathak D.: Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062* (2021). 3

[HPSK21] Hwang J.-P., Park G., Suh I. H., Kwon T.: Primitive object grasping for finger motion synthesis. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 266–278. 2

[HTS*17] Heess N., TB D., Sriram S., Lemmon J., Merel J., Wayne G., Tassa Y., Erez T., Wang Z., Eslami S., et al.: Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* (2017). 3

[JKL18] Jin T., Kim M., Lee S.-H.: Aura mesh: Motion retargeting to preserve the spatial relationships between skinned characters. *Computer Graphics Forum 37*, 2 (2018), 311–320. 2

[JSK*20] Jeong R., Springenberg J. T., Kay J., Zheng D., Zhou Y., Galashov A., Heess N., Nori F.: Learning dexterous manipulation from suboptimal experts. *arXiv preprint arXiv:2010.08587* (2020). 3

[KGTL16] Kumar V., Gupta A., Todorov E., Levine S.: Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095* (2016). 3

[KP12] Karpathy A., Panne M. v. d.: Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence* (2012), Springer, pp. 325–330. 3

[LFP07] Li Y., Fu J. L., Pollard N. S.: Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions on visualization and computer graphics 13*, 4 (2007), 732–747. 2

[LH18] Liu L., Hodgins J.: Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 1

[Liu09] Liu C. K.: Dextrous manipulation from a grasping pose. In *ACM SIGGRAPH 2009 papers*. 2009, pp. 1–6. 2

[LLLL21] Lee S., Lee S., Lee Y., Lee J.: Learning a family of motor skills from a single motion clip. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–13. 3

[LMH*13] Li Q., Meier M., Haschke R., Ritter H., Bolder B.: Rotary object dexterous manipulation in hand: a feedback-based method. *International Journal of Mechatronics and Automation 3*, 1 (2013), 36–47. 2

[LPLL19] Lee S., Park M., Lee K., Lee J.: Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG) 38*, 4 (2019), 1–13. 3, 9

[LYG15] Liu L., Yin K., Guo B.: Improving sampling-based motion control. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 415–423. 3

[LYVdP*10] Liu L., Yin K., van de Panne M., Shao T., Xu W.: Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*. 2010, pp. 1–10. 3

[Med] MediaPipe: Mediapipe hands. URL: https://google.github.io/mediapipe/solutions/hands. 1

[MHG*19] Merel J., Hasenclever L., Galashov A., Ahuja A., Pham V., Wayne G., Teh Y. W., Heess N.: Neural probabilistic motor primitives for humanoid control. *International Conference on Learning Representations (ICLR)* (2019). 3

[MLP16] Mamou K., Lengyel E., Peters A.: Volumetric hierarchical approximate convex decomposition. In *Game Engine Gems 3*. AK Peters, 2016, pp. 141–158. 7

[MPT12] Mordatch I., Popović Z., Todorov E.: Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation* (2012), pp. 137–144. 2

[MTA*20] Merel J., Tunyasuvunakool S., Ahuja A., Tassa Y., Hasenclever L., Pham V., Erez T., Wayne G., Heess N.: Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 39–1. 1

[NKLK20] Nagabandi A., Konolige K., Levine S., Kumar V.: Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning* (2020), PMLR, pp. 1101–1112. 3

[Opt22] Optitrack: Motion capture cameras, 2022. URL: https://optitrack.com/cameras/. 3

[PALvdP18] Peng X. B., Abbeel P., Levine S., van de Panne M.: Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 2, 3, 4, 5

[PCZ*20] Peng X. B., Coumans E., Zhang T., Lee T.-W., Tan J., Levine S.: Learning agile robotic locomotion skills by imitating animals. In *Proceedings of Robotics: Science and Systems (RSS)* (2020). 3

[PKM*18] Peng X. B., Kanazawa A., Malik J., Abbeel P., Levine S.: Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG) 37*, 6 (2018), 1–14. 3

[PMA*21] Peng X. B., Ma Z., Abbeel P., Levine S., Kanazawa A.: Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–20. 9

[PRL*19] Park S., Ryu H., Lee S., Lee S., Lee J.: Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–11. 3

[PZ05] Pollard N. S., Zordan V. B.: Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 311–318. 2

[RKG*18] Rajeswaran A., Kumar V., Gupta A., Vezzani G., Schulman J., Todorov E., Levine S.: Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)* (2018). 1, 3

[RWPM21] Radosavovic I., Wang X., Pinto L., Malik J.: State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), IEEE, pp. 7865–7871. 3

[SDT*22] Simeonov A., Du Y., Tagliasacchi A., Tenenbaum J. B., Rodriguez A., Agrawal P., Sitzmann V.: Neural descriptor fields: Se(3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)* (2022), pp. 6394–6400. 2

[SH18] Sundaralingam B., Hermans T.: Geometric in-hand regrasp planning: Alternating optimization of finger gaits and in-grasp manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 231–238. 2

[SHX*22] She Q., Hu R., Xu J., Liu M., Xu K., Huang H.: Learning high-dof reaching-and-grasping via dynamic representation of gripper-object interaction. *ACM Trans. Graph. 41*, 4 (jul 2022). 2

[SI91] Sawasaki N., Inoue H.: Tumbling objects using a multi-fingered robot. *Journal of the Robotics Society of Japan 9*, 5 (1991), 560–571. 2

[SWD*17] Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O.: Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017). 6

[TAY10] Tahara K., Arimoto S., Yoshida M.: Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand. In *2010 IEEE International Conference on Robotics and Automation* (2010), IEEE, pp. 4322–4327. 2

[TCBT22] Taheri O., Choutas V., Black M. J., Tzionas D.: Goal: Generating 4d whole-body motion for hand-object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 13263–13273. 2

[TET12] Todorov E., Erez T., Tassa Y.: Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 5026–5033. 3, 4

[TGBT20] Taheri O., Ghorbani N., Black M. J., Tzionas D.: Grab: A dataset of whole-body human grasping of objects. In *European Conference on Computer Vision* (2020), Springer, pp. 581–600. 2

[The02] The Blender Foundation: Blender, 2002. URL: https://www.blender.org/. 7

[TKH21] Tang Z., Kim D., Ha S.: Learning agile motor skills on quadrupedal robots using curriculum learning. In *International Conference on Robot Intelligence Technology and Applications* (2021). 3

[TLT11] Tan J., Liu K., Turk G.: Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications 31*, 4 (2011), 34–44. 4

[WGH20] Won J., Gopinath D., Hodgins J.: A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 33–1. 1, 2, 3

[WGH21] Won J., Gopinath D., Hodgins J.: Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–11. 3

[WL19] Won J., Lee J.: Learning body shape variation in physics-based characters. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–12. 3

[WLCS19] Wang R., Lehman J., Clune J., Stanley K. O.: Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753* (2019). 2, 3

[WLR*20] Wang R., Lehman J., Rawal A., Zhi J., Li Y., Clune J., Stanley K.: Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning* (2020), PMLR, pp. 9940–9951. 3

[WWS*15] Wheatland N., Wang Y., Song H., Neff M., Zordan V., Jörg S.: State of the art in hand and finger modeling and animation. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 735–760. 2

[WWZ*22] Wu Y., Wang J., Zhang Y., Zhang S., Hilliges O., Yu F., Tang S.: Saga: Stochastic whole-body grasping with contact. In *European Conference on Computer Vision* (2022), Springer, pp. 257–274. 2

[XLKvdP20] Xie Z., Ling H. Y., Kim N. H., van de Panne M.: Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 213–224. 3

[YCBVdP08] Yin K., Coros S., Beaudoin P., Van de Panne M.: Continuation methods for adapting simulated skills. In *ACM SIGGRAPH 2008 papers*. 2008, pp. 1–7. 3

[YL12] Ye Y., Liu C. K.: Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–10. 2

[YTL18] Yu W., Turk G., Liu C. K.: Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–12. 1

[YYL22] Yang Z., Yin K., Liu L.: Learning to use chopsticks in diverse gripping styles. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–17. 2

[ZGR*19] Zhu H., Gupta A., Rajeswaran A., Levine S., Kumar V.: Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 3651–3657. 3

[ZYSK21] Zhang H., Ye Y., Shiratori T., Komura T.: Manipnet: neural manipulation synthesis with a hand-object spatial representation. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–14. 2, 3