HANDFUL: Sequential Grasp-Conditioned Dexterous Manipulation with Resource Awareness

Ethan Foong*[1§], Yunshuang Li*[2], Hao Jiang[2], Gaurav S. Sukhatme[2], Daniel Seita[2]

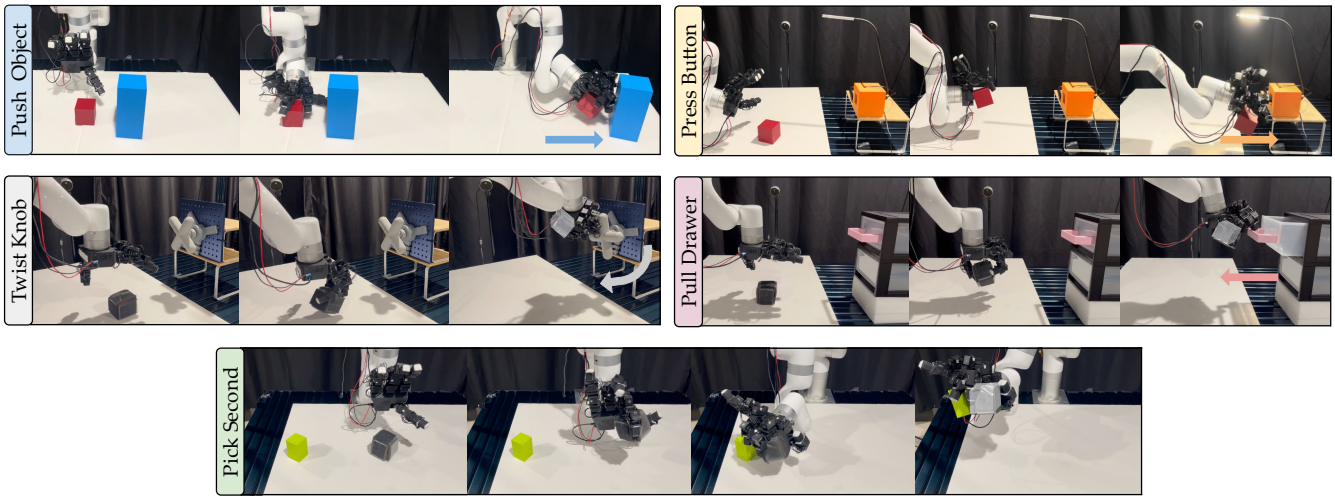

Fig. 1: HANDFUL enables sequential dexterous manipulation by learning resource-aware grasps that preserve specific fingers for downstream subtasks. For each task, the robot first selects an appropriate initial grasp of the object (a block), and then executes a second subtask using the remaining fingers. We show real-world rollouts of HANDFUL using a LEAP Hand [28] for tasks that involve pushing (top left), pressing (top right), twisting (middle left), pulling (middle right), and grasping a second object (bottom).

***Abstract*— Dexterous robot hands offer rich opportunities for multifunctional manipulation, where a robot must execute multiple skills in sequence while maintaining control over previously grasped objects. Most prior work in dexterous manipulation focuses on single-object, single-skill tasks. In contrast, our insight is that many sequential tasks require resource-aware grasps that conserve fingers for future actions. In this paper, we study sequential grasp-conditioned dexterous manipulation, where a robot first grasps an object and then performs a second, distinct manipulation subtask while preserving the initial grasp. We introduce HANDFUL, a learning framework that models finger usage as a limited resource and encourages exploration of resource-aware grasps through finger-level contact rewards. These grasps are subsequently selected for downstream tasks via curriculum-based policy learning. We further propose HANDFUL-Bench, a simulation benchmark that introduces sequential dexterous manipulation tasks across multiple second-subtask objectives, including pushing, pulling, and pressing, under a shared grasp-conditioned setup. Extensive simulation results demonstrate that prioritizing resource-aware grasps improves second-subtask success and robustness compared to a baseline that greedily optimizes the initial grasp before attempting the second subtask. We additionally validate our approach on a real dexterous LEAP hand. Together, this work establishes resource-aware grasp planning as a key principle for multifunctional dexterous manipulation. Supplementary material is available on our website: *handful-dex.github.io*.**

*Equal Contribution.
[1]Northwestern University, [2]University of Southern California.
§This work was initiated through Ethan Foong's participation in the USC Viterbi Summer Undergraduate Research Experience (SURE) program and continued as a collaboration thereafter.

I. Introduction

Dexterous multi-fingered hands with high degrees-of-freedom (DoF) offer manipulation capabilities that extend beyond those of parallel-jaw grippers. With many independently actuated fingers, such hands can perform tasks that require great dexterity, such as reorientation of mechanically complex objects [20]. However, despite their versatility, much of prior work in dexterous manipulation focuses on isolated skills, such as grasping [32], [43], pushing [9], or in-hand rotation [21], [22], [37], typically applied to a single object and optimized for immediate task completion.

In contrast, many real-world manipulation scenarios are inherently sequential and multifunctional. A robot clearing a table or organizing a workspace must often first acquire an object and then perform a second task while continuing to hold it. For example, after grasping a container, a robot may need to push a button, pull a drawer, or reach for a second object without relinquishing its initial grasp. These scenarios require the robot to maintain force closure on the held object while simultaneously executing large reconfiguration motions with other fingers. Critically, success depends not only on the final grasp stability, but also on how finger contacts are allocated after the first grasp. In many cases, grasps optimized purely for force closure or stability occupy fingers and contact regions that are needed for subsequent actions, rendering downstream tasks infeasible. These settings therefore require resource-aware grasps that conserve

fingers and contact surfaces for future subtasks.

In this paper, we propose HANDFUL, whole-**HAN**d **D**exterity **F**or seq**U**ential task **L**earning, a framework for sequential dexterous manipulation that treats finger usage as a limited resource to be allocated across subtasks. Rather than optimizing grasps for immediate stability, HANDFUL learns resource-aware grasping policies that preserve unused fingers and contact regions for future actions. These grasps are then selected to serve as starting states to downstream subtasks through curriculum-based policy learning, which identifies the grasp modalities that best support subsequent subtasks. In addition, we introduce HANDFUL-Bench, a new simulation benchmark built on ManiSkill [29] with the LEAP Hand [28]. The benchmark consists of tasks that share a common first subtask (grasping), while varying the second-subtask objective, such as pushing, pulling, or acquiring a second object. By fixing the first grasping subtask and varying the second objectives, HANDFUL-Bench isolates the role of different grasping modalities and enables systematic evaluation of multifunctional dexterous policies. We will make the benchmark publicly available to facilitate reproducible experiments and further research.

To summarize, our main contributions include:

- A framework for sequential dexterous manipulation that treats finger usage as a limited resource and learns grasp states that preserve resources for future subtasks.
- A curriculum-based training approach for efficiently identifying which grasping modalities best support second-subtask policy learning.
- HANDFUL-Bench, a simulation benchmark for multi-step dexterous manipulation that highlights the role of different resource-aware grasping modalities in downstream task performance.
- Extensive simulation experiments demonstrating improved success rates over stability-focused baselines, along with a retrieval-based execution strategy that enables robust real-world deployment.

II. Related Work

A. Dexterous Multi-Object and Multi-Step Manipulation

Dexterous manipulation is a long-standing area of research [25]. Broadly, this research area attempts to leverage hands with multiple fingers and high degrees-of-freedom (DoF) to enable greater capabilities. Techniques in dexterous hand manipulation range from analytical (often using some variant of force closure analysis [14], [13] to learning-based techniques, such as learning from demonstrations to perform manipulation tasks using teleoperation [31], or leveraging large-scale datasets for dexterous grasping [32], [43], [34], [16] or nonprehensile [9] maneuvers.

Within dexterous manipulation, key research subareas include multi-object and multi-step manipulation. Multi-object grasping is a well-studied problem [35], [39], [38] for which researchers have recently made key new advances in analysis [36], algorithms [6], [10], [15] and hardware [4]. In contrast to these works, we study a general framework for multi-step manipulation which involves controlling multiple objects. Some of the closest-related prior works have studied reinforcement learning approaches for chaining multi-step manipulation policies [2]. We complement this line of work by studying scenarios where the robot must maintain control of an object during a second subtask.

B. Simulators and Benchmarks in Robot Manipulation

Simulators and benchmarks play critical roles in advancing robotic manipulation by enabling rapid, fair, and reproducible comparisons of different algorithms. In robot learning for manipulation, researchers utilize simulation frameworks such as IsaacLab [18] (an updated version of IsaacGym [17]), MuJoCo [30], and PyBullet [3]. IsaacLab and MuJoCo both provide GPU-acceleration, enabling rapid and large-scale reinforcement learning runs, and additionally come with robot models [41] and tasks [40]. Closely related to our work, ManiSkill [29] proposes a GPU-parallelized robot simulation framework based on SAPIEN [33]. ManiSkill also comes with diverse robot manipulation tasks and APIs to control robots. This is similar to software that acts as benchmarks for testing robot manipulation algorithms, ranging from LIBERO [12], robosuite [44], RoboCasa [19], and RLBench [7]. These are geared for testing generalist manipulation policies. Other simulation benchmarks test more specific, yet valuable, manipulation domains, such as deformable object manipulation [11], [27] and dexterous grasping [1]. Our benchmark, HANDFUL-Bench, is complementary to these works. Built on ManiSkill, it provides the first simulation benchmark to rigorously study sequential and multi-step robotic object manipulation.

III. Problem Statement and Assumptions

We study sequential grasp-conditioned dexterous manipulation in which a "task" consists of a sequence of two "subtasks," $g^{(1)} \rightarrow g^{(2)}$, where the second must be completed while maintaining control of the first. We assume that the first subtask $g^{(1)}$ is to grasp an object and that the second subtask $g^{(2)}$ must preserve the initial grasp. The goal is for a robot to autonomously compose and execute both subtasks. We assume a single-arm robot with a four-fingered hand, such as the LEAP Hand [28] and available state-based observations. We assume that the scene contains at least two objects within reachable range of the robot.

IV. Method: HANDFUL

HANDFUL addresses sequential grasp-conditioned dexterous manipulation by modeling finger usage as a limited resource to be allocated across subtasks. We use reinforcement learning to learn two subpolicies that map state-based inputs to continuous actions: a grasping policy $\pi^{(1)}$ that produces resource-aware grasps, followed by a second-subtask policy $\pi^{(2)}$ that completes the task while preserving the initial grasp. Our method has three main parts: (1) learning a diverse set of dexterous finger-constrained grasping policies with finger-level contact rewards that conserve unused fingers (Sec. IV-A); (2) learning dexterous finger-constrained second-subtask

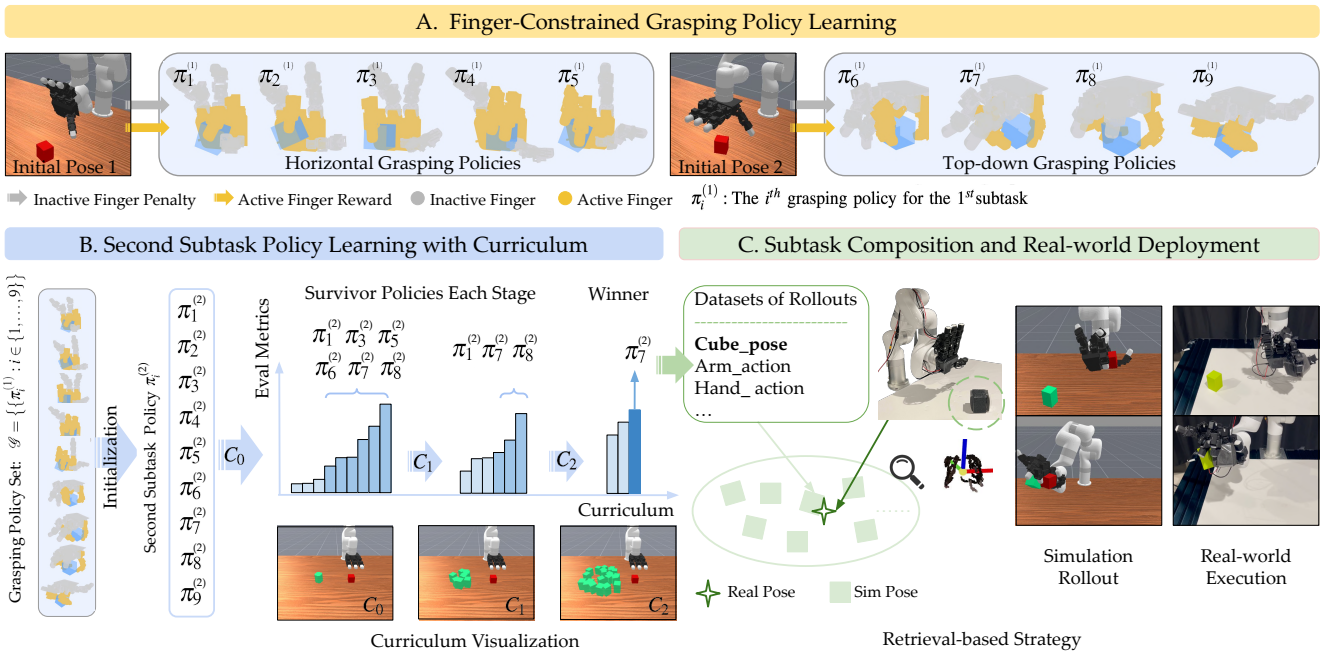

Fig. 2: Overview of HANDFUL. **(A, Sec. IV-A)**: Multiple grasping policies are trained with active and inactive finger constraints to encourage resource-aware grasps that preserve fingers for future manipulation. **(B, Sec. IV-B)**: For each grasp, a second-stage manipulation policy is trained via a multi-stage curriculum, where survivor policies are selected at increasing environment difficulty levels. **(C, Sec. IV-C)**: Successful policies for grasping and manipulation are composed sequentially. Real-world object poses are matched to simulated rollouts via retrieval-based execution, enabling sim-to-real transfer.

policies with curriculum learning to identify grasp modalities that maximize second-subtask success (Sec. IV-B), and (3) composing learned subtasks to deploy them in the real-world via a retrieval-based execution strategy (Sec. IV-C).

A. Finger-Constrained Grasping Policy Learning

The first stage in all benchmarked tasks is grasping. Because the grasping subtask competes with subsequent subtasks for finger allocation, we propose a reward function that explicitly encourages exploration of diverse, low-resource grasping policies through finger-level contact rewards. To do this, we define the notion of "active" and "inactive" fingers within each subtask. Active fingers specify which fingers should interact with the current objective in each subtask, whereas inactive fingers are deliberately conserved for the following subtask. For example, in the first grasping subtask, the active fingers are used to grasp and hold the first object. In the second subtask, the roles reverse: the active fingers from the first subtask stabilize the grasped object, while the previously inactive fingers are used to accomplish the new objective (e.g., twisting a knob or pushing a second block). By incorporating the notion of active and inactive fingers into the reward function, we explicitly specify how finger resources are allocated across subtasks within the overall task. With this notion in mind, we define our reward r_t during the grasping subtask at time t as:

$$r_t = w_g r_g + w_v p_v + w_c p_c + w_a r_a + w_{ina} p_{ina} \quad (1)$$

where we design the following new reward components:

$$r_a = \frac{1}{|M|} \sum_{i \in M} \exp(-\lambda_a d_i) \quad \text{Active Finger Reward} \quad (2)$$

$$p_{ina} = \text{clip}(-\sum_{j \in J} f_j, -\beta_{ina}, 0) \quad \text{Inactive Finger Penalty} \quad (3)$$

Here, Eq. 2 incentivizes the M active contact points of the active fingers to remain close to the object by increasing the reward as the distance between the contact points and target object d_i decreases. The palm is also a contact point and included in the active contact set, and $\lambda_a \in \mathbb{R}$ denotes a scaling factor. Eq. 3 discourages inactive fingers $j \in J$ from touching the object by penalizing their contact forces f_j up to a maximum of $\beta_{ina} \in \mathbb{R}$ to preserve these fingers for the next subtask. In addition, r_g refers to a general grasping reward, adopted from [8], [23], [29], with reaching and lifting rewards. Finally, p_v penalizes excessive arm velocity and p_c penalizes collisions with the table. The $w_g, w_v, w_c, w_a, w_{ina}$ are scalar weights used to form a weighted sum of the reward components. See the appendix for more details.

To provide diverse starting states for the second subtasks, we train grasping policies using every one- and two-finger combination on the four-finger hand, where selected fingers are designated as active while the remaining fingers are inactive. To encourage diverse grasp modalities, we train all grasping policies from two initial hand poses: (1) above the object with the palm facing downward, and (2) level with the object with the palm facing horizontally towards it. This results in *nine* unique and feasible finger configurations (see Fig. 2). We denote the corresponding set of nine grasping policies as $\mathscr{G} = \{\{\pi_i^{(1)} : i \in \{1, \ldots, 9\}\}$. We train

Algorithm 1 Curriculum-Based Grasp Selection

Require: $\mathscr{G}$, curriculum $\{C_0, C_1, C_2\}$, steps $\{k_0, k_1, k_2\}$, number of survivor grasps to next stage $m_1 = 6$, $m_2 = 3$.

for $g_i \in \mathscr{G}$ **do**
  Initialize $\pi_i^{(2)}$ from terminal states of g_i
  Train under C_0 for k_0 steps
  $\Pi \leftarrow \Pi \cup \{\pi_i^{(2)}\}$
end for
for $j = 1, 2$ **do**
  Sort Π by p_{st}, p_{sa}, p_{ar} ▷ In priority order
  $\Pi \leftarrow$ select top m_j policies
  Continue training under C_j for k_j steps
end for
return $\arg\max_{\pi \in \Pi} p_1(\pi)$

the grasping policies using Soft Actor-Critic (SAC) [5], but HANDFUL is compatible with other reinforcement learning algorithms such as Proximal Policy Optimization (PPO) [26].

B. Second Subtask Policy Learning with Curriculum

Starting from a diverse set of grasping policies $\mathscr{G}$, we learn finger-constrained second-subtask policies. We adopt a forward initialization training scheme inspired by Chen et al. [2], in which each second-subtask policy is trained from terminal states generated by a first-subtask grasping policy $\pi_i^{(1)}$. Different grasps induce distinct hand configurations and contact patterns, which affect the feasibility and performance of downstream tasks. However, training second-subtask policies on top of all nine grasping policies is time-consuming and computationally inefficient. To address this, we introduce a grasp selection strategy combined with curriculum learning to (1) efficiently identify the terminal grasp states best suited for a specific second subtask, and (2) progressively improve second-subtask policy performance.

We construct a curriculum with three stages, C_0, C_1, C_2, of progressively increasing environment randomization levels. We define a candidate grasp as the terminal states of a grasping policy and the subtask policy initialized from those states. Candidate grasps are ranked in priority order: first by terminal success rate p_{st} of the second-subtask policy, with any-time success rate p_{sa} and average return p_{ar} serving as successive tie-breakers. Training begins in C_0 for k_0 steps. We then promote six "survivor" grasps to C_1. After further training, three "survivor" grasps are promoted to C_2. The policy with the highest final success rate is the final "winner policy" for the task. Denoting its index as i^*, we obtain

$$\pi^{(1)} = \pi_{i^*}^{(1)}, \quad \pi^{(2)} = \pi_{i^*}^{(2)}$$

as our final set of policies. See Alg. 1 for details.

The primary reward components for the second-subtask environments correspond to their main objective (see Sec. V for the list of second subtasks). Due to our resource-aware design, the active and inactive finger roles are reversed between subtasks. Fingers that were active during grasping now stabilize the object, while previously inactive fingers execute downstream manipulation. We therefore retain the active-finger reward in Eq. 2 and the inactive-finger penalty in Eq. 3, but apply them to the reversed finger sets. This encourages the original grasping fingers to maintain object control while ensuring that the newly active fingers remain free. Finally, for any reward terms involving the fingertips in the second subtask, we compute the value using only the active fingertips, allowing the inactive fingertips to prioritize grasp maintenance. Further details are in the appendix.

C. Subtasks Composition and Real-world Deployment

To perform the full sequential task, we first train and evaluate the policies in simulation. Then, we compose the learned policies by first executing $\pi^{(1)}$ to obtain a terminal grasp state, followed by $\pi^{(2)}$ to complete the downstream objective while preserving the grasp.

While policies are trained in simulation, directly executing them in the real world is difficult due to the sim-to-real gap. To enable safe and efficient real-world deployment, we therefore adopt a retrieval-based strategy built on a dataset of successful simulated rollouts. For the best-performing policy of each task in the real world, we collect N trajectories under randomized initial object poses in the first grasping subtask, forming a dataset $\mathscr{D} = \{\tau_k : k \in \{1, \ldots, N\}\}$, where each trajectory $\tau_k = \{(s_t^{(k)}, a_t^{(k)})\}_{t=0}^{T}$ corresponds to a collision-free execution in simulation. During real-world execution, we segment the first object to grasp using SAM2 [24] and estimate its 3D position $\mathbf{p}^{\text{real}} \in \mathbb{R}^3$ from fused point clouds captured by two camera views. We then retrieve the trajectory τ_{k^*} whose initial state best matches the observed pose:

$$k^* = \arg\min_k \|\mathbf{p}_0^{(k)} - \mathbf{p}^{\text{real}}\|_2, \quad k \in \{1, \ldots, N\} \tag{4}$$

where $\mathbf{p}_0^{(k)}$ denotes the initial first object position in τ_k. We execute the selected trajectory τ_{k^*} in the real world. This retrieval-based strategy improves robustness compared to directly deploying learned RL policies, reducing the sim-to-real gap by grounding execution in pre-validated trajectories.

V. Benchmark: HANDFUL-Bench

To accelerate research in sequential dexterous manipulation, we design a benchmark, HANDFUL-Bench, which will be made open-source. We implement our benchmark in ManiSkill [29], a robotics simulator with strong community support and GPU acceleration for learning algorithms. Each task in HANDFUL-Bench follows a shared structure. The robot first grasps a target object and then performs one of several second-stage tasks while maintaining the grasp. We consider the following downstream tasks (see Fig. 3):

- `Push Object`: The robot must push a secondary object, without rotating or tipping it, to a target position.
- `Press Button`: The robot must reach and insert its finger into a hole to press a button.
- `Twist Knob`: The robot must twist a knob a target number of degrees.
- `Pull Drawer`: The robot must pull a drawer open using the available fingers.

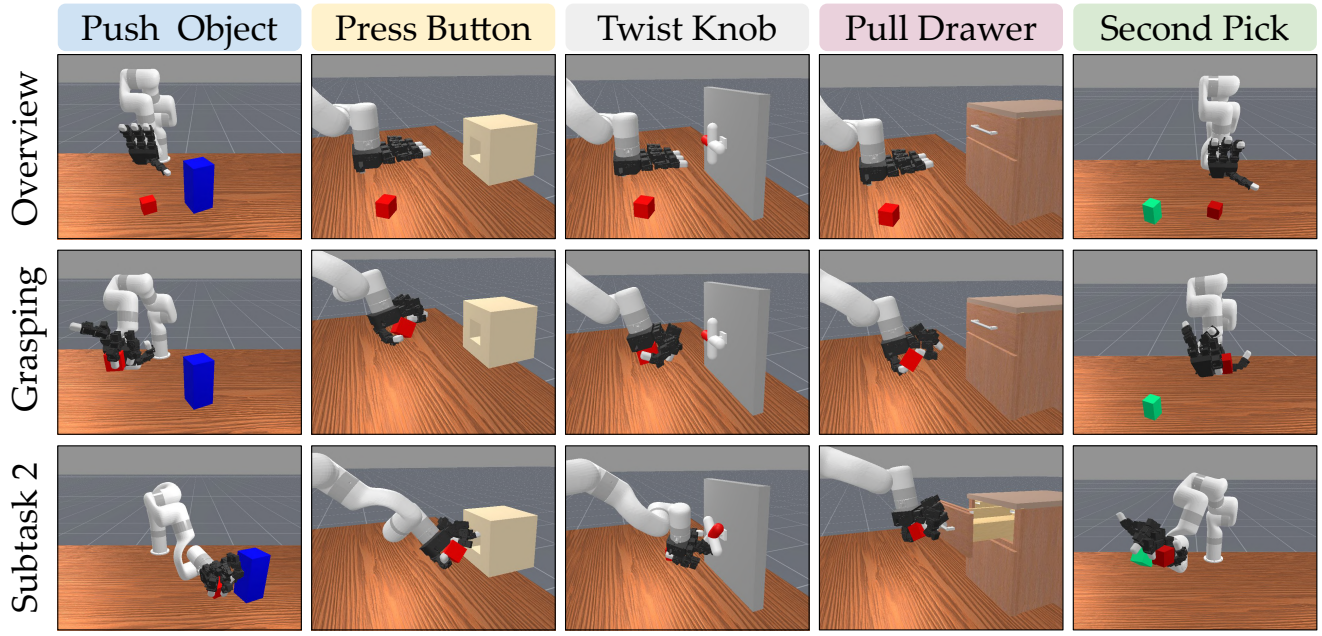

Fig. 3: Overview of HANDFUL-Bench (Sec. V). The top row shows the starting scenarios for the five tasks. The middle row shows the outcome after grasping the block with resource-aware grasping policies. The bottom row shows successful configurations after performing the second subtask while preserving the initial grasp. The examples above are test-time executions of HANDFUL.

- `Pick Second`: The robot must grasp and lift a second object to a target region.

In HANDFUL-Bench, while the first grasping subtask is shared across all tasks, the second subtasks impose distinct and often competing demands on finger allocation, contact patterns, and space within the hand. Initial grasps optimized solely for force closure can occupy regions of the hand or fingers required for subsequent tasks. As a result, effective manipulation in HANDFUL-Bench may require initial grasps that conserve resources by leaving fingers or in-hand spaces free, and in turn, adopt less conventional hand poses to accommodate downstream objectives. Furthermore, grasps that facilitate one second subtask may be suboptimal or infeasible for another, indicating that a diverse set of grasping modalities is necessary for success throughout the task suite.

Across all environments, we use a 23-dimensional action space consisting of 16 delta joint commands for the hand and 7 for the arm. Robot control is implemented via a PD joint delta position controller. Each subtask provides robot state observations, which include the arm and hand joint positions and velocities, the palm and fingertip poses, and the arm's wrist position. We also include subtask-specific observations such as object poses, relative pose information (e.g., arm wrist to block), object half sizes (if varying), and articulation joint position values (e.g., knob rotation). We also provide a one hot vector of the current active finger indices in anticipation of policies that can switch grasping or second-subtask policy modalities based on this conditioning. Finally, to encourage stable grasps throughout, episodes run for the full duration of each subtask rather than terminating upon first success. We thus use terminal success rate p_{st}, whether the task is successfully completed at the final timestep, as our primary evaluation metric.

VI. Simulation Experiments

We study the following questions: (1) Is the diversity of grasping policies useful for learning the second subtask? (2) Does curriculum learning increase policy robustness? (3) Do early curriculum stages reliably predict final task performance? (4) Does the selection process retain the best candidate grasps? (5) Does task decomposition help training?

A. Simulation Experiment Setup

We use HANDFUL-Bench, and initialize the simulation environments with an xArm7 robot arm and a LEAP Hand [28] as the end-effector. While we use the LEAP Hand, our method, HANDFUL, is not specific to the platform and in principle could be applied to other dexterous multi-fingered hands (e.g., the Allegro Hand).

B. Methods in Simulation

We first train nine grasping policies with different hand poses for the first grasping subtask as stated in Sec. IV-A, achieving an average success rate of $94.67 \pm 2.60\%$, indicating relatively uniform performance when trained solely for grasping. We then initialize nine parallel second-subtask training runs for each task, starting from the successful terminal states generated by each of the grasping policies. These are our candidate grasps. Next, we utilize curriculum-based grasp selection as stated in Sec. IV-B. The results reported in Table I are the performance of "the winner policy" of each task evaluated in the second-subtask environment, starting from the matching terminal grasp state. Success is defined as both maintaining a stable grasp on the first-subtask object until the end and achieving the goal of the second subtask.

Method	Push Object	Press Button	Twist Knob	Pull Drawer	Pick Second
HANDFUL (Ours)	**69.90 ± 5.54**	**77.75 ± 2.15**	**61.52 ± 5.47**	**78.94 ± 1.77**	76.54 ± 3.63
w/o Curriculum	69.47 ± 1.18	76.80 ± 7.32	58.50 ± 9.48	71.38 ± 20.16	**80.42 ± 6.72**
w/o Finger Constraint	66.69 ± 5.66	44.26 ± 40.58	49.44 ± 5.73	58.99 ± 17.36	0.00 ± 0.00
Phase-based Reward	32.38 ± 6.36	10.08 ± 19.23	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

TABLE I: Success rate across tasks in simulation over five seeds. For each task, only the selected grasp is evaluated for both curriculum and non-curriculum training. Here, "w/o Curriculum" and "w/o Finger Constraint" are ablations of our method (HANDFUL) with the identified component removed (see Sec. VI-B for details). For each task, we bold the metric with the highest mean value.

We compare our approach, HANDFUL, with the following two ablations and one baseline:

- **w/o Curriculum:** Our method without the curriculum-based training. This directly trains the second-subtask policy once under C_2 using the same terminal grasp states as the selected "winner policy" for each task.
- **w/o Finger Constraint:** Our method without learning finger-constrained grasping. All fingers are designated active in both subtasks, with no separation between the fingers responsible for holding the first object and those performing the second-subtask objective.
- **Phase-based Reward:** Training the whole task in a unified environment, using two phase-based rewards based on the current timestep of the environment.

C. Simulation Results

1) Grasping Policy Diversity: In Table I, policies trained with finger constraints (HANDFUL and w/o Curriculum) consistently outperform those without (w/o Finger Constraint), with the gap widening for tasks that require greater finger availability, such as `Press Button`, `Pull Drawer`, and `Pick Second`. Notably, w/o Finger Constraint completely fails on `Pick Second` (0%), suggesting that without separate finger allocation between subtasks, the policy struggles to free sufficient fingers and in-hand space for dexterous tasks. The high variance on `Press Button` (±40.58) further reflects this difficulty. This result highlights the importance of promoting and maintaining grasp pose diversity in these grasp-conditioned manipulation tasks.

2) Curriculum Learning: See Fig. 4 for learning curves comparing HANDFUL versus w/o Curriculum. HANDFUL achieves comparable success rates (see Table I) in all five tasks while reducing total training time by 40% (54 million steps versus 90 million steps if training second-stage policies for all nine candidate grasps). We also observe that curriculum learning may stabilize second-stage policy training, possibly due to the early stages of the curriculum providing a more consistent environment to learn successful strategies for each task, though further investigation would be needed to confirm this. Together, these results suggest that the primary contribution of curriculum learning in our setup is its practical benefits to training efficiency and stability.

3) Candidate Grasp Selection Validation: We investigate whether early curriculum stages reliably predict final task performance and whether the selection process consistently retains the best candidate grasps through a case study on `Pick Second`. This is a representative task that is a strong test for our selection criteria: training is volatile due to tight constraints on in-hand space and finger placement, and this volatility is consequential. Accidentally eliminating the top 2-3 grasp candidates can cause a larger performance drop off compared to tasks such as `Push Object`, where all grasps achieve satisfactory performance.

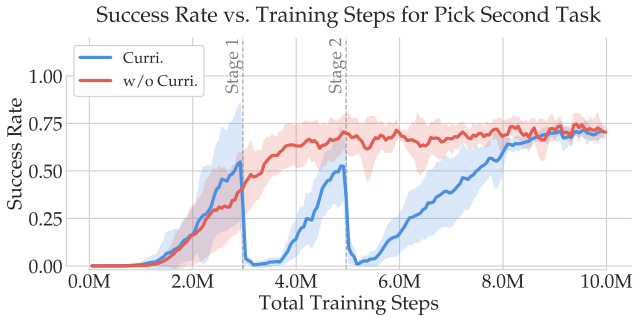

Fig. 4: Performance of HANDFUL with curriculum learning vs without curriculum learning over full training with 10 million steps.

(i) *Predictivity of Early Curriculum Stages:* To assess whether early stages of the curriculum reliably predict final performance, we train all candidate grasps through C_0, C_1, and C_2 without selection and record the performance of their second-subtask policies in each stage. Table II shows that the top-performing grasp candidates in C_2 consistently perform well in the prior stages, validating that early stage performance is a predictive signal for grasp selection.

(ii) *Selection Stability Across Seeds:* Using the same experiment from (i), we simulate what our selection criteria would select at each stage to evaluate whether it consistently retains the top-performing candidate grasps. Table III shows that the best performing candidate grasp is never prematurely eliminated and the second best is eliminated only once, just before C_2. There is some volatility in identifying the third-best grasp, though the grasp candidates selected over it perform only moderately worse in C_2 as shown in Table II.

4) Task Decomposition: In Table I, all approaches trained with task decomposition significantly outperform our baseline phase-based reward method which fails to solve three tasks on any of the five seeds. Qualitatively, these policies struggle heavily with the shifting reward landscape during rollouts, often prioritizing the second-subtask objectives without learning stable grasps. These results emphasize the importance of task decomposition in multi-step tasks.

VII. Real-World Experiments

In this section, we evaluate whether trajectories generated in simulation can reliably transfer to the real world.

Curriculum	$\pi_1^{(2)}$	$\pi_2^{(2)}$	$\pi_3^{(2)}$	$\pi_4^{(2)}$	$\pi_5^{(2)}$	$\pi_6^{(2)}$	$\pi_7^{(2)}$	$\pi_8^{(2)}$	$\pi_9^{(2)}$
C_0	51.8	0.0	42.8	0.0	0.0	29.4	36.4	18.4	1.6
C_1	65.2	0.0	31.8	0.0	11.8	27.4	41.2	22.5	3.6
C_2	77.8	0.0	41.8	0.1	13.2	39.3	66.0	52.2	33.3

TABLE II: Early Stage Curriculum Predictivity for `Pick Second`. For each grasp, we report the average maximum training "success at end" rate p_{st} over 3 seeds for each stage of the curriculum. Highlighted are the 1st, 2nd, and 3rd best performing grasps in each stage.

Curriculum	$\pi_1^{(2)}$	$\pi_2^{(2)}$	$\pi_3^{(2)}$	$\pi_4^{(2)}$	$\pi_5^{(2)}$	$\pi_6^{(2)}$	$\pi_7^{(2)}$	$\pi_8^{(2)}$	$\pi_9^{(2)}$
C_0	3	3	3	3	3	3	3	3	3
C_1	3	0	3	0	1	3	3	2	3
C_2	3	0	1	0	0	1	2	1	1

TABLE III: Selection Stability Validation for the `Pick Second` task. For each grasp, we report the number of times (out of 3 seeds) it reached each stage of the curriculum. Highlighted are the 1st, 2nd, and 3rd best performing grasps in the final stage.

A. Real-World Experiment Setup

As in the simulation setup, all experiments are conducted on a 7-DoF xArm7 robotic manipulator with a four-finger, 16-DoF LEAP Hand [28] (see Fig. 5). We use two Intel RealSense L515 RGB-D cameras mounted at complementary viewpoints to estimate the 3D pose of the block. The point clouds from both cameras are fused and segmented using SAM2 [24] to estimate the block state for retrieval-based execution (Sec. IV-C). To isolate the sequential grasp-conditioned manipulation component of our approach, we fix the position of the second object across trials, while randomizing the position of the block in a 10 by 10 cm region in front of the robot.

In each evaluation trial, the retrieved trajectory is executed in an open-loop manner until completion. The first subtask succeeds if the robot holds the block until execution ends. The second succeeds if the robot correctly manipulates the second object, regardless of the first block's state. To analyze performance at different stages, we categorize each trial into four mutually exclusive outcomes. "Success (Both)" indicates that both Subtask 1 and Subtask 2 are completed successfully within a single trial. "Fail Subtask 1 Only" denotes trials where Subtask 1 fails but Subtask 2 succeeds. "Fail Subtask 2 Only" denotes trials where Subtask 2 fails but Subtask 1 succeeds. "Fail Both" corresponds to trials where neither subtask is successfully completed. These four categories sum to the total number of trials (15 per task).

B. Real-World Experiment Results

We deploy our retrieval-based strategy stated in IV-C with the dataset we collected in simulation, consisting of 50 successful trajectories for each task. Over 15 trials per task (Table IV), HANDFUL achieves full two-stage success rates of 66.7% (`Push Object`), 53.3% (`Press Button`), 40.0% (`Twist Knob`), 40.0% (`Pull Drawer`), and 26.7% (`Pick Second`). Notably, the policy successfully completes at least 40% of trials on four out of five tasks, despite being trained primarily in simulation.

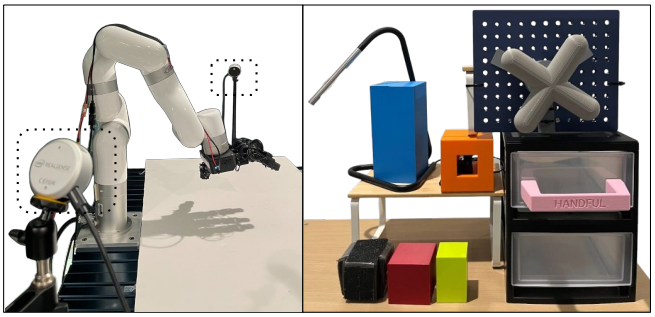

Fig. 5: Left: the real world experiment setup with the xArm7 robot, LEAP hand, and the two Intel L515 cameras indicated with dotted boxes. Right: the different objects we use for experiments.

Tasks	`Push Object`	`Press Button`	`Twist Knob`	`Pull Drawer`	`Pick Second`
Success (Both)	10	8	6	6	4
Fail Subtask 1 Only	4	4	6	6	7
Fail Subtask 2 Only	0	0	0	0	1
Fail Both	1	3	3	3	3

TABLE IV: Real-world experiment results. For each task, we report the number of trials (out of 15) categorized by success and failure types. Each column sums to 15 trials. See Sec. VII for details.

Across tasks, the majority of failures occur during the second subtask, while isolated second subtask only failures are rare. This indicates that the learned grasp is generally stable for the first manipulation and that performance degradation primarily arises from the increased demands of sequential, multi-functional execution. A representative failure mode is object dropping during the second action, underscoring the challenge of maintaining grasp stability under changing contact conditions. Overall, these results demonstrate that HANDFUL transfers effectively from simulation to the real world and is capable of executing complex, sequential, dexterous manipulation across diverse tasks.

C. Comparison with Demonstration-Based Approach

To complement the results in Sec. VII-B, we evaluate all five tasks using 3D Diffusion Policy (DP3) [42], trained on 30 DexCap [31] teleoperated demonstrations per task using the segmented block point cloud as input. This comparison highlights the tradeoffs between HANDFUL, which relies on reinforcement learning in simulation, and demonstration-based learning. DP3 achieves a 38.6% success rate over 75 rollouts (15 each task), underperforming our method on four of the five tasks. We observe systematic differences between the two approaches: teleoperated data rarely leverages palm contact, likely due to teleoperation interface limitations, and exhibits a consistent bias toward using the index finger for the second subtask while reserving other fingers for grasp stabilization, a pattern observed across operators (see Fig. 6 for visualizations). In contrast, HANDFUL more explicitly utilizes the palm and produces hand poses that fall outside the typical distribution of human demonstrations but better exploit the mechanical capabilities of the robot hand.

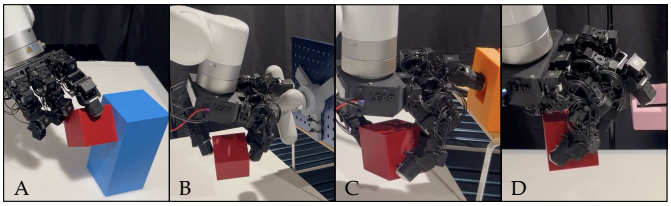

Fig. 6: Hand poses in the human teleoperated data. Across tasks, operators predominantly relied on the index finger for the second subtask and used the remaining fingers for grasp stabilization, reflecting a consistent bias and limited pose diversity.

VIII. Limitations

Our experiments use a relatively controlled setting, with state-based observations and limited randomization (e.g., grasping only a block). This simplifies evaluation but does not fully capture real-world diversity. Additionally, the task suite includes a small set of downstream manipulation objectives chosen to stress different finger-level resource constraints; broader task coverage would be needed to evaluate generalization to more complex manipulation sequences. Finally, while our method is not tied to a specific hand design, we only use the LEAP hand, and future work should test with other robotic hands.

IX. Conclusion

We propose HANDFUL, a pipeline for dexterous multi-step manipulation that learns finger-constrained grasping policies and uses curriculum-based grasp selection for downstream subtasks, alongside HANDFUL-Bench, a simulation benchmark to accelerate future research. Our results suggest that in multi-step, multi-object manipulation tasks, effective subtask policy learning may require explicitly accounting for shared physical resources, such as finger availability and in-hand space, that are easy to overlook when learning skills in isolation. We hope that this perspective inspires future work in multi-step and multi-object dexterous hand manipulation.

X. Acknowledgement

We thank Yiyang Ling for assistance with 3D printing. We are grateful to Sumeet Batra, Shashank Hegde, Darren Chiu, Guangyao Shi and Ellen Novoseller for insightful discussions on the project methodology, and to Zeyu Shangguan and Ruohai Ge for valuable feedback on the manuscript.

References

[1] C. Bao, H. Xu, Y. Qin, and X. Wang, "Dexart: Benchmarking generalizable dexterous manipulation with articulated objects," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

[2] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu, "Sequential dexterity: Chaining dexterous policies for long-horizon manipulation," in Conference on Robot Learning (CoRL), 2023.

[3] E. Coumans and Y. Bai, "PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning," 2016–2020.

[4] J. Eom, S. Y. Yu, W. Kim, C. Park, K. Y. Lee, and K.-J. Cho, "Mogrip: Gripper for multiobject grasping in pick-and-place tasks using translational movements of fingers," Science Robotics, 2024.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in International Conference on Machine Learning (ICML), 2018.

[6] S. He, Z. Shangguan, K. Wang, Y. Gu, Y. Fu, Y. Fu, and D. Seita, "Sequential multi-object grasping with one dexterous hand," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2025.

[7] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison, "RLBench: The Robot Learning Benchmark & Learning Environment," in IEEE Robotics and Automation Letters (RA-L), 2020.

[8] H. Jiang, Y. Wang, H. Zhou, and D. Seita, "Learning to Singulate Objects in Packed Environments using a Dexterous Hand," in International Symposium on Robotics Research (ISRR), 2024.

[9] Y. Li, Y. Ling, G. S. Sukhatme, and D. Seita, "Learning Geometry-Aware Non-prehensile Pushing and Pulling with Dexterous Hands," in IEEE International Conference on Robotics and Automation (ICRA), 2026.

[10] Y. Li, B. Liu, Y. Geng, P. Li, Y. Yang, Y. Zhu, T. Liu, and S. Huang, "Grasp multiple objects with one hand," in IEEE Robotics and Automation Letters (RA-L), 2024.

[11] X. Lin, Y. Wang, J. Olkin, and D. Held, "SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation," in Conference on Robot Learning (CoRL), 2020.

[12] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning," in Neural Information Processing Systems, 2023.

[13] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, "Deep differentiable grasp planner for high-dof grippers," in Robotics: Science and Systems (RSS), 2020.

[14] T. Liu, Z. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, "Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator," in IEEE Robotics and Automation Letters (RA-L), 2022.

[15] H. Lu, Y. Dong, Z. Weng, F. T. Pokorny, J. Lundell, and D. Kragic, "Grasping a handful: Sequential multi-object dexterous grasp generation," in IEEE Robotics and Automation Letters (RA-L), 2025.

[16] T. G. W. Lum, A. H. Li, P. Culbertson, K. Srinivasan, A. D. Ames, M. Schwager, and J. Bohg, "Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer," in Conference on Robot Learning (CoRL), 2024.

[17] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning," arXiv preprint arXiv:2108.10470, 2021.

[18] M. Mittal, P. Roth, J. Tigue, A. Richard et al., "Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning," arXiv preprint arXiv:2511.04831, 2025.

[19] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu, "RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots," in Robotics: Science and Systems (RSS), 2024.

[20] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," arXiv preprint arXiv:1910.07113, 2019.

[21] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-Hand Object Rotation via Rapid Motor Adaptation," in Conference on Robot Learning (CoRL), 2022.

[22] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "General In-Hand Object Rotation with Vision and Touch," in Conference on Robot Learning (CoRL), 2023.

[23] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, "DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation," in Conference on Robot Learning (CoRL), 2022.

[24] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "SAM 2: Segment Anything in Images and Videos," arXiv preprint arXiv:2408.00714, 2024.

[25] D. Rus, "In-Hand Dexterous Manipulation of Piecewise-Smooth 3-D Objects," in International Journal of Robotics Research (IJRR), 1999.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv:1707.06347, 2017.

[27] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks," in IEEE International Conference on Robotics and Automation (ICRA), 2021.

[28] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," in Robotics: Science and Systems (RSS), 2023.

[29] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su, "ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI," arXiv preprint arXiv:2410.00425, 2024.

[30] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A Physics Engine for Model-Based Control," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.

[31] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, "DexCap: Scalable and Portable Mocap Data Collection System for Dexterous Manipulation," in Robotics: Science and Systems (RSS), 2024.

[32] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, "Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation," in IEEE International Conference on Robotics and Automation (ICRA), 2023.

[33] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A SimulAted Part-based Interactive ENvironment," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[34] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen et al., "Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2023.
[35] T. Yamada and H. Yamamoto, "Static grasp stability analysis of multiple spatial objects," Journal of Control Science and Engineering, vol. 3, 2015.
[36] K. Yao and A. Billard, "Exploiting kinematic redundancy for robotic grasping of multiple objects," in IEEE Transactions on Robotics, 2023.
[37] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang, "Rotating without Seeing: Towards In-hand Dexterity through Touch," in Robotics: Science and Systems (RSS), 2023.
[38] T. Yoshikawa, T. Watanabe, and M. Daito, "Optimization of power grasps for multiple objects," in IEEE International Conference on Robotics and Automation (ICRA), 2001.
[39] Y. Yu, K. Fukuda, and S. Tsujio, "Computation of grasp internal forces for stably grasping multiple objects," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2001.
[40] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel, "MuJoCo Playground," in Robotics: Science and Systems (RSS), 2025.
[41] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022.
[42] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations," in Robotics: Science and Systems (RSS), 2024.
[43] J. Zhang, H. Liu, D. Li, X. Yu, H. Geng, Y. Ding, J. Chen, and H. Wang, "DexGraspNet 2.0: Learning Generative Dexterous Grasping in Large-scale Synthetic Cluttered Scenes," in Conference on Robot Learning (CoRL), 2024.
[44] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A Modular Simulation Framework and Benchmark for Robot Learning," in arXiv preprint arXiv:2009.12293, 2020.

XI. APPENDIX

A. Candidate Grasp Selection Validation on All Tasks

We extend the case study from Section VI-C.3 to all tasks. Table V and Table VI report curriculum predictivity and selection stability across `Push Object`, `Press Button`, `Twist Knob`, and `Pull Drawer`.

Curriculum	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9
Push Object									
C_0	80.9	10.9	51.1	76.2	58.3	31.4	45.7	86.7	71.4
C_1	78.2	20.2	58.2	67.5	52.1	29.5	44.7	80.5	65.8
C_2	58.6	28.5	21.5	46.2	52.3	43.4	54.4	75.3	62.3
Press Button									
C_0	32.2	18.7	49.6	19.9	26.1	74.7	73.4	84.8	76.4
C_1	18.8	3.8	42.5	25.2	57.1	58.0	58.3	82.5	54.0
C_2	48.9	10.2	62.2	42.8	56.7	68.0	71.2	83.1	55.7
Twist Knob									
C_0	30.1	0.5	14.3	0.7	4.0	4.5	14.9	57.5	28.8
C_1	32.7	0.7	25.3	1.2	7.0	10.5	18.9	61.3	29.8
C_2	45.2	10.2	38.7	4.0	15.6	16.5	28.8	63.7	38.3
Pull Drawer									
C_0	48.0	0.0	8.2	1.8	0.0	0.0	19.1	59.6	65.9
C_1	56.0	0.0	3.1	0.0	0.0	0.0	11.6	79.8	63.3
C_2	69.3	0.0	9.2	1.8	0.0	0.0	11.7	83.2	68.5

TABLE V: Early Stage Curriculum Predictivity. We report the average terminal success rate p_{st} (%) over 3 seeds for each curriculum stage. This shows how performance on early stages (C_0,C_1) correlates with final stage success (C_2). Gold, silver, and bronze highlights indicate the top three performing policies per row.

1) Predictivity of Early Curriculum Stages: The pattern observed for `Pick Second` in the main paper holds broadly across tasks. The top-performing candidates in the final curriculum stage consistently rank among the leaders in earlier stages, confirming that early performance is a reliable signal for grasp selection. Early curriculum stages seem more predictive for more difficult tasks (e.g., `Twist Knob`, `Pull Drawer`), where there is high variance between grasp types allowing early stages to more clearly reveal separation between strong and weak grasp candidates.

2) Selection Stability Across Seeds: Selection stability is similarly strong across tasks. The best-performing candidate grasp is never prematurely eliminated in any seed of any task, and the second and third-best candidates are often retained. As in `Pick Second`, there is some volatility in selecting these second and third-best grasps, but the impact is minor since the candidates selected over them still tend to perform comparably (e.g., π_9 in Twist Knob).

3) Limitations: Because we train only a single grasping seed per candidate strategy, it is difficult to cleanly attribute each grasp's second-subtask performance to its intrinsic task suitability versus the stability of that particular grasp instance (which may vary with the training seed). In our experiments, candidates such as π_1 and π_8 perform strongly across nearly all tasks, which may reflect broad task compatibility, high grasp stability in this seed, or both. Running multiple grasping seeds per strategy and evaluating them on second subtasks would help disentangle these factors and yield more concrete conclusions about which grasps are optimal for each task. Videos of the best grasp strategies per second-subtask are available on our website: https://handful-dex.github.io/.

Curriculum	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9
Push Object									
C_0	3	3	3	3	3	3	3	3	3
C_1	3	0	2	3	2	0	2	3	3
C_2	3	0	0	2	0	0	0	3	1
Press Button									
C_0	3	3	3	3	3	3	3	3	3
C_1	2	0	2	1	1	3	3	3	3
C_2	0	0	2	0	1	1	1	3	1
Twist Knob									
C_0	3	3	3	3	3	3	3	3	3
C_1	3	1	2	1	2	1	2	3	3
C_2	2	0	1	0	0	1	1	3	1
Pull Drawer									
C_0	3	3	3	3	3	3	3	3	3
C_1	3	0	1	1	2	3	2	3	3
C_2	3	0	0	0	0	0	0	3	3

TABLE VI: Selection Stability across Tasks. We report the number of seeds (out of 3) that successfully transition through curriculum stages C_i for nine candidate grasping policies (π_1 to π_9). Gold, silver, and bronze highlights denote the top three policies by average final stage success rate.

B. Hyperparameters of SAC

All experiments use Soft Actor-Critic (SAC) [5] with the hyperparameters listed in Table VII. Grasping policies are trained for 6M steps. Second-subtask policies are trained across three curriculum stages (C_0,C_1,C_2) of 3M, 2M, and 5M steps respectively (10M total); in the second and third stages, learning starts after 409,600 steps to allow the carried-over policy to populate the replay buffer with rollouts in the new stage environment before gradient updates begin.

C. Domain Randomization

To promote robust policies, we apply domain randomization across all tasks per episode. The grasped block is initialized within a 10 by 10 cm square region with a random yaw rotation (intentionally conservative, as the tabletop must simultaneously accommodate all task-relevant objects). Second-subtask object placement follows task-specific distributions. The push block is spawned within a 10 by 10 cm square and the push goal pose within a 40 by 40 cm one. Both are spawned with the same random yaw. The knob is displaced by up to $\pm$15 cm with $\pm$15° rotation noise and a uniformly randomized initial joint position in $[-\pi,\pi]$ radians. The button is displaced by up to $\pm$25 cm with $\pm$15° rotation noise. The cabinet is placed along a 90° arc with radius 0.8 m facing the robot base. It is then perturbed an additional $\pm$5 cm and $\pm$15° from that position and rotation. Finally, the second block in `Pick Second` is initialized within a 30 by 30 cm square.

Physical properties are randomized for the grasped block and all task-relevant objects. Mass is scaled by a uniform factor in $[0.5,1.5]\times$ the nominal value, and surface friction and restitution are sampled uniformly from $[0.1,0.5]$ and

Hyperparameter	Value
Training Duration	
Grasping stages	6M steps
Second-subtask stages (C_0, C_1, C_2)	3M, 2M, 5M steps
Learning starts (C_0 and Grasping)	51,200 steps
Learning starts (C_1, C_2)	409,600 steps
Environment	
Parallel training environments	512
Episode Length (Grasping)	75
Episode Length (All second-subtasks)	100
SAC	
Replay buffer size	4,000,000
Batch size	1,024
Training frequency	2,048 steps
Update-to-data ratio (UTD)	0.5
Discount factor γ	0.95
Target network smoothing τ	0.005
Policy learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Entropy coefficient α	0.2 (autotuned)
Policy update frequency	1
Target network update frequency	1

TABLE VII: Hyperparameters of HANDFUL. All values are shared across tasks unless otherwise noted.

$[0.0, 0.1]$ respectively. The same mass and friction randomization is applied to all robot links. Finally, the grasped block is sampled from a set of five size variants spanning approximately 2.5–3.0 cm per side.

As described briefly in Section IV-B, we apply environment randomizations within our second-subtask environments based on curriculum stage. In C_0 we'd like our second-subtask policies to focus on finding successful strategies for the subtask so we remove almost all environment randomizations other than the initial position of the hand and grasped object (which come from the end states of our grasping policies). In C_1 we introduce position and rotation randomizations at half strength to allow the policies trained in C_0 under no randomizations to slowly generalize to more diverse initial conditions. Finally, in C_2 we add all randomizations at full strength (including mass, friction, and restitution which are not randomized in C_0 and C_1) to obtain the final policy. Visualizations of curriculum stage randomization distributions are available on the project website.

D. Subtask Reward and Success Criteria Details

Throughout sections XI-E, XI-F, XI-G, XI-H, XI-I, and XI-J, M denotes the grasping finger set as defined in Section IV-A, and J denotes the second subtask manipulation finger set (which is all non-grasping fingers). These sets are fixed across all subtasks. As in the main paper, M is active during the grasping subtask and J is active during the second subtasks. One subtlety is that in the grasping subtask only, M includes the palm as an additional contact point to enable palm grasps. We use a similar idea to aid grasping of the second object in `Pick Second`. We do not find this necessary for initial grasp maintenance or later manipulation in other subtasks.

Symbol	Description	Value
d_r	Distance from palm to target block	—
d_i	Distance from contact point $i \in M$ to block	—
d_l	Distance from block to goal	—
$\sum_{j \in J} f_j$	Total force of inactive fingers on target block	—
T	Total contact force of hand on table	—
$\mathbf{v}_a$	Robot joint velocity vector	—
λ_r	Scaling factor for reach reward	5.0
λ_a	Scaling factor for active finger reward	20.0
λ_l	Scaling factor for lift reward	5.0
λ_c	Scaling factor for collision penalty	0.05
α_g	Grasp contact threshold	0.05 m
β_{ina}	Inactive finger penalty cap	1.5
β_c	Collision penalty cap	1.0

TABLE VIII: Variables and parameters for the grasping subtask reward.

For the success criteria of all subtasks, two conditions we commonly check are the grasped block being off the table and all of the M grasping fingers being within 0.07 m of the grasped object's center. This helps us ensure that throughout all the subtasks, the initial grasped object stays grasped by the correct fingers. This is especially important in the second subtasks where the robot hand may accomplish the main objective (like pulling the drawer), but inadvertently drop the grasped object. For simplicity, we combine these conditions and refer to them as whether the object is grasped.

E. Grasping Policy Reward and Success Criteria

In the grasping subtask, success is triggered when the robot lifts the target block within 0.03 m of a goal location while the block is grasped.

Following the same reward structure and notation as Section IV-A, the full reward r_t at time t is:

$$r_t = w_r r_r + w_{gr} r_{gr} + w_l r_l + w_a r_a + w_{ina} p_{ina} + w_v p_v + w_c p_c \tag{5}$$

where r_r, r_{gr}, and r_l together constitute the general grasping reward r_g from the main paper:

$$r_r = 1 - \tanh(\lambda_r d_r) \quad \text{Reach Rew.} \tag{6}$$

$$r_{gr} = \frac{1}{|M|} \sum_{i \in M} \mathbf{1}(d_i < \alpha_g) \quad \text{Grasp Rew.} \tag{7}$$

$$r_l = 1 - \tanh(\lambda_l d_l) \quad \text{Lift Rew.} \tag{8}$$

$$r_a = \frac{1}{|M|} \sum_{i \in M} \exp(-\lambda_a d_i) \quad \text{Active Finger Rew.} \tag{9}$$

$$p_{ina} = \text{clip}(-\sum_{j \in J} f_j, -\beta_{ina}, 0) \quad \text{Inactive Finger Pen.} \tag{10}$$

$$p_v = -\|\mathbf{v}_a\|_2 \quad \text{Velocity Pen.} \tag{11}$$

$$p_c = \text{clip}(-\lambda_c T, -\beta_c, 0) \quad \text{Contact Pen.} \tag{12}$$

with hyperparameter definitions in Table VIII. We use weights $w_r = 2.0$, $w_{gr} = 2.5$, $w_l = 40.0$, $w_a = 7.5$, $w_{ina} = 1.0$, $w_v = 1.5$, $w_c = 1.0$.

Reward Rationale and Details. Here we expand on the constituent components of r_g from section IV-A. The *reach* reward provides a positive signal when the palm of the hand is near the block. The *grasp* reward increases with

Symbol	Description	Value
d_i	Distance from finger $i \in M$ to grasped block	—
d_r	Distance from palm to push block	—
d_j	Distance from pushing finger $j \in J$ to push block	—
d_p	Distance from push block to goal	—
θ	Rotation error between push block and goal	—
$\sum_{j\in J} f_j$	Total force of pushing fingers on grasped block	—
T	Total force of hand and grasped block on table	—
$\mathbf{v}_a$	Robot joint velocity vector	—
λ_{gs}	Scaling factor for grasp stability reward	20.0
λ_r	Scaling factor for reach reward	5.0
λ_{pp}	Scaling factor for push proximity reward	5.0
λ_θ	Scaling factor for rotation reward	5.0
λ_c	Scaling factor for soft collision penalty	0.05
β_{int}	Interference penalty cap	1.0
ε	Collision detection threshold	0.01
β_c	Binary collision penalty magnitude	4.0
d_0	Initial push block to goal distance (average)	0.4 m

TABLE IX: Variables and parameters for the pushing subtask reward.

the number of active contact points within a threshold α_g of the block, promoting more secure grasps. The *lift* reward increases as the robot lifts the block toward the goal. The remaining components follow the main paper: r_a incentivizes the M active contacts (palm and fingertips) to remain close to the block, p_{ina} discourages the J inactive fingers from contacting the block to preserve them for the next subtask, p_v penalizes excessive arm velocity, and p_c penalizes hand-table collisions.

F. Pushing Policy Reward

In the pushing subtask, success is defined as the push block being within 0.03 m and 0.5 radians of the goal site's position and rotation plus whether the initial object (grasped block) is grasped.

For this reward function, we follow the same reward structure and notation as Section XI-E. As this is a second subtask, the set of M grasping fingers is now designated inactive and the set of J pushing fingers is active. The full reward r_t at time t is:

$$\begin{aligned} r_t = {} & w_{gs} r_{gs} + w_r r_r + w_{pp} r_{pp} + w_p r_p \\ & + w_\theta r_\theta + w_{int} p_{int} + w_v p_v + w_c p_c \end{aligned} \tag{13}$$

which is the weighted sum of eight reward components:

$$r_{gs} = \frac{1}{|M|} \sum_{i\in M} \exp(-\lambda_{gs} d_i) \quad \text{Grasp Stability Rew.} \tag{14}$$

$$r_r = 1 - \tanh(\lambda_r d_r) \quad \text{Reach Rew.} \tag{15}$$

$$r_{pp} = \frac{1}{|J|} \sum_{j\in J} \exp(-\lambda_{pp} d_j) \quad \text{Push Proximity Rew.} \tag{16}$$

$$r_p = 1 - \frac{d_p}{d_0} \quad \text{Place Rew.} \tag{17}$$

$$r_\theta = \exp(-\lambda_\theta \theta) \quad \text{Rotation Rew.} \tag{18}$$

$$p_{int} = \text{clip}(-\sum_{j\in J} f_j, -\beta_{int}, 0) \quad \text{Interference Pen.} \tag{19}$$

$$p_v = -\|\mathbf{v}_a\|_2 \quad \text{Velocity Pen.} \tag{20}$$

$$p_c = -\beta_c \mathbf{1}(T > \varepsilon) - \lambda_c T \quad \text{Collision Pen.} \tag{21}$$

Symbol	Description	Value
d_i	Distance from finger $i \in M$ to grasped block	—
d_j	Distance from finger $j \in J$ to button	—
$\sum_{j\in J} f_j$	Total force of pressing fingers on grasped block	—
T	Total force of hand and grasped block on table	—
$\mathbf{v}_a$	Robot joint velocity vector	—
λ_{gs}	Scaling factor for grasp stability reward	20.0
λ_r	Scaling factor for reach reward	5.0
λ_c	Scaling factor for collision penalty	0.05
β_{int}	Interference penalty cap	1.0

TABLE X: Variables and parameters for the press button subtask reward.

with hyperparameter definitions in Table IX. We use weights $w_{gs} = 7.5$, $w_r = 3.0$, $w_{pp} = 3.0$, $w_p = 20.0$, $w_\theta = 5.0$, $w_{int} = 1.0$, $w_v = 0.1$, $w_c = 1.0$.

Reward Rationale and Details. r_{gs}, p_{int} are the renamed versions of r_a and p_{ina} from the grasping subtask. They motivate grasp maintenance with the grasping fingers and discourage pushing fingers from touching the grasped block to focus on the pushing block respectively. p_v is also conceptually identical to the grasping subtask: penalizing joint velocity. The *reach* reward encourages the hand to move toward the push block. The *push proximity* reward incentivizes the J pushing fingers to remain close to the push block. The *place* reward increases linearly as the push block approaches its goal pose, normalized by the average initial distance d_0. The *rotation* reward penalizes angular deviation between the push block and goal orientation, where θ is the geodesic rotation error in radians. The *collision* penalty slightly differs from the grasping subtask: it adds a binary term $-\beta_c \mathbf{1}(T > \varepsilon)$ on top of the soft term, and T additionally includes contact forces between the grasped block and table to discourage dropping the grasped block.

G. Button Press Policy Reward

In the pressing subtask, success is only triggered if one of the J pressing fingers is within 0.04 m of the goal site (simulating a button press), the grasped block is grasped and crucially, if there is no contact between the hand and the button guard/donut.

We follow the same reward structure and notation as Section XI-F. The M grasping fingers are designated to maintain the initial grasp while one of the J pressing fingers must press the button. The full reward r_t at time t is:

$$r_t = w_{gs} r_{gs} + w_r r_r + w_{int} p_{int} + w_v p_v + w_c p_c \tag{22}$$

which is the weighted sum of five reward components:

$$r_{gs} = \frac{1}{|M|} \sum_{i\in M} \exp(-\lambda_{gs} d_i) \quad \text{Grasp Stability Rew.} \tag{23}$$

$$r_r = 1 - \tanh(\lambda_r \min_{j\in J} d_j) \quad \text{Reach Rew.} \tag{24}$$

$$p_{int} = \text{clip}(-\sum_{j\in J} f_j, -\beta_{int}, 0) \quad \text{Interference Pen.} \tag{25}$$

$$p_v = -\|\mathbf{v}_a\|_2 \quad \text{Velocity Pen.} \tag{26}$$

$$p_c = -\lambda_c T \quad \text{Collision Pen.} \tag{27}$$

Symbol	Description	Value
d_i	Distance from finger $i \in M$ to grasped block	—
d_r	Distance from palm to valve	—
d_j	Distance from twisting finger $j \in J$ to valve	—
$\dot{\theta}_{dir}$	Valve rotation velocity in target direction	—
θ	Accumulated valve rotation	—
$\sum_{j\in J} f_j$	Total force of twisting fingers on grasped block	—
T	Total force of hand/grasped block on obstacles	—
$\mathbf{v}_a$	Robot joint velocity vector	—
λ_{gs}	Scaling factor for grasp stability reward	20.0
λ_r	Scaling factor for valve reach reward	5.0
λ_{fr}	Scaling factor for fingertip reach reward	10.0
λ_{vel}	Scaling factor for velocity reward	5.0
θ_{succ}	Success rotation threshold	$\frac{4\pi}{3}$
λ_c	Scaling factor for collision penalty	0.05
β_{int}	Interference penalty cap	1.0

TABLE XI: Variables and parameters for the twist knob subtask reward. T includes contact forces between the hand and the wall/table; and between the cube and the wall, table, and knob.

with definitions and values in Table X. Weights are set to $w_r = 7.5, w_{gs} = 2.0, w_{int} = 1.0, w_v = 0.1, w_c = 1.0$.

Reward Rationale and Details. r_{gs}, p_{int}, and p_v are conceptually identical to the pushing subtask. The *reach* reward uses the minimum distance among inactive fingers $j \in J$ to the button rather than an average, encouraging whichever finger is closest to make contact. p_c retains the form from the grasping subtask, but T additionally includes contact forces between any hand link and the obstacle box, encouraging the robot to avoid collision by precisely pressing the button with its fingertip.

H. Twist Knob Policy Reward

In the twist knob subtask, there is a success if the valve is rotated more than $\frac{4\pi}{3}$ radians in the target direction and the grasped block is grasped. The full reward r_t is defined as:

$$\begin{aligned} r_t = w_{gs} r_{gs} + w_r r_r + w_{fr} r_{fr} + w_{vel} r_{vel} \\ + w_m r_m + w_{int} p_{int} + w_v p_v + w_c p_c \end{aligned} \tag{28}$$

where the constituent components are:

$$r_{gs} = \frac{1}{|M|} \sum_{i\in M} \exp(-\lambda_{gs} d_i) \quad \text{Grasp Stability Rew.} \tag{29}$$

$$r_r = 1 - \tanh(\lambda_r d_r) \quad \text{Reach Rew.} \tag{30}$$

$$r_{fr} = \frac{1}{|J|} \sum_{j\in J} \exp(-\lambda_{fr} d_j) \quad \text{Fingertip Reach Rew.} \tag{31}$$

$$r_{vel} = \tanh(\lambda_{vel} \dot{\theta}_{dir}) \quad \text{Velocity Rew.} \tag{32}$$

$$r_m = \text{clip}(\theta / \theta_{succ}, -1, 1) \quad \text{Motion Rew.} \tag{33}$$

$$p_{int} = \text{clip}(-\sum_{j\in J} f_j, -\beta_{int}, 0) \quad \text{Interference Pen.} \tag{34}$$

$$p_v = -\|\mathbf{v}_a\|_2 \quad \text{Joint Velocity Pen.} \tag{35}$$

$$p_c = \text{clip}(-\lambda_c T, -5.0, 0) \quad \text{Collision Pen.} \tag{36}$$

with definitions and values in Table XI. We set $w_{gs} = 7.5, w_r = 3.0, w_{fr} = 7.5, w_{vel} = 3.0, w_m = 10.0, w_{int} = 1.0, w_v = 0.05, w_c = 2.0$.

Symbol	Description	Value
d_i	Distance from finger $i \in M$ to grasped block	—
d_r	Distance from palm to handle	—
d_j	Distance from manipulation finger $j \in J$ to handle	—
q	Joint position of cabinet drawer	—
q_{target}	Target joint position for success	—
$\sum_{j\in J} f_j$	Total force of manipulation fingers on grasped block	—
T	Total force of hand and grasped block on table	—
$\mathbf{v}_a$	Robot joint velocity vector	—
λ_{gs}	Scaling factor for grasp stability reward	20.0
λ_r	Scaling factor for handle reach reward	5.0
λ_{fr}	Scaling factor for fingertip reach reward	5.0
λ_c	Scaling factor for collision penalty	0.05
β_{int}	Interference penalty cap	1.0

TABLE XII: Variables and parameters for the pull drawer subtask reward.

Reward Rationale and Details. The reward structure prioritizes both grasp stability and functional manipulation. r_{gs} provides a dense signal ensuring the newly inactive fingers M remain in contact with the grasped block, while p_{int} prevents the active twisting fingers J from exerting unwanted forces on the grasped object. To guide the manipulation, r_r attracts the palm toward the valve, and r_{fr} provides granular guidance by attracting individual $j \in J$ fingertips to the knob handles. Task progress is captured by r_{vel}, which rewards rotation velocity in the target direction $\dot{\theta}_{dir}$, and r_m, which measures accumulated rotation θ. The collision penalty p_c captures contact forces T across five object pairs: hand–wall, hand–table, grasped block–wall, grasped block–table, and grasped block–knob. The hand-wall and hand-table penalties help reduce unsafe contact, while the grasped block penalties discourage dropping or dislodging of the block by any of the environment object (in particular, the knob).

I. Pull Drawer Policy Reward

In the pull drawer subtask, success is triggered if the drawer is opened more than q_{target}, the drawer handle is static, and the initial object is grasped. The full reward r_t is:

$$\begin{aligned} r_t = w_{gs} r_{gs} + w_r r_r + w_{fr} r_{fr} + w_o r_o \\ + w_{int} p_{int} + w_v p_v + w_c p_c \end{aligned} \tag{37}$$

where the constituent components are:

$$r_{gs} = \frac{1}{|M|} \sum_{i\in M} \exp(-\lambda_{gs} d_i) \quad \text{Grasp Stability Rew.} \tag{38}$$

$$r_r = 1 - \tanh(\lambda_r d_r) \quad \text{Handle Reach Rew.} \tag{39}$$

$$r_{fr} = \frac{1}{|J|} \sum_{j\in J} (1 - \tanh(\lambda_{fr} d_j)) \quad \text{Finger Reach Rew.} \tag{40}$$

$$r_o = q / q_{target} \quad \text{Open Rew.} \tag{41}$$

$$p_{int} = \text{clip}(-\sum_{j\in J} f_j, -\beta_{int}, 0) \quad \text{Interference Pen.} \tag{42}$$

$$p_v = -\|\mathbf{v}_a\|_2 \quad \text{Joint Velocity Pen.} \tag{43}$$

$$p_c = -\lambda_c T \quad \text{Collision Pen.} \tag{44}$$

with definitions and values in Table XII. We set $w_{gs} = 5.0$, $w_r = 1.0$, $w_{fr} = 5.0$, $w_o = 15.0$, $w_{int} = 1.0$, $w_v = 0.1$, $w_c = 3.0$.

Symbol	Description	Value
d_i	Distance from finger $i \in M$ to grasped block	—
d_r	Distance from palm to second block	—
d_j	Distance from contact point j to second block	—
d_p	Distance from second block to goal	—
h	Current height of second block	—
h_0	Initial height of second block	—
g_M	Fraction of M fingers grasping grasp block	—
$\sum_{j\in J} f_j$	Total force of J fingers on grasped block	—
T	Total force of robot and grasped block on table	—
$\mathbf{v}_a$	Robot joint velocity vector	—
λ_{gs}	Scaling factor for grasp stability reward	20.0
λ_r	Scaling factor for reach reward	5.0
λ_{fd}	Scaling factor for finger distance reward	20.0
λ_p	Scaling factor for place reward	5.0
α_g	First block grasp contact threshold	0.07 m
α_p	Second block grasp contact threshold	0.05 m
h_{max}	Maximum lift bonus height	0.1 m
λ_c	Scaling factor for collision penalty	0.05
β_{int}	Interference penalty cap	1.0

TABLE XIII: Variables and parameters for the two-pick subtask reward.

Reward Rationale and Details. r_{gs}, p_{int}, p_v, and p_c are conceptually identical to prior subtasks. r_r attracts the palm toward the handle, and r_{fr} guides the J fingertips toward it. r_o measures normalized joint progress toward the target joint position q_{target}. To avoid reward competition, once the joint begins opening, r_r is overridden to its maximum value, and once sufficiently open, both r_r and r_{fr} are similarly overridden, allowing the policy to focus entirely on maintaining the initial grasp.

J. Two-Pick Policy Reward

In the two-pick subtask, the robot is successful if the second block is lifted within 0.03 m of the goal site and the initial block is grasped. The full reward r_t is:

$$\begin{aligned} r_t = {} & w_{gs} r_{gs} + w_r r_r + w_{fd} r_{fd} + w_{sg} r_{sg} \\ & + w_h r_h + w_p r_p + w_{int} p_{int} + w_v p_v + w_c p_c \end{aligned} \tag{45}$$

where the constituent components are:

$$r_{gs} = \frac{1}{|M|} \sum_{i \in M} \exp(-\lambda_{gs} d_i) \quad \text{Grasp Stability Rew.} \tag{46}$$

$$r_r = 1 - \tanh(\lambda_r d_r) \quad \text{Reach Rew.} \tag{47}$$

$$r_{fd} = \frac{\sum_{j \in J \cup \{p\}} \exp(-\lambda_{fd} d_j)}{|J| + 2} \quad \text{Finger Dist. Rew.} \tag{48}$$

$$r_{sg} = \frac{1}{|J|} \sum_{j \in J} \mathbf{1}(d_j < \alpha_p) \quad \text{Second Grasp Rew.} \tag{49}$$

$$r_h = \text{clip}\left(\frac{h - h_0}{h_{max}}, 0, 1 \right) \cdot g_M \quad \text{Height Rew.} \tag{50}$$

$$r_p = 1 - \tanh(\lambda_p d_p) \quad \text{Place Rew.} \tag{51}$$

$$p_{int} = \text{clip}(-\sum_{j \in J} f_j, -\beta_{int}, 0) \quad \text{Interference Pen.} \tag{52}$$

$$p_v = -\|\mathbf{v}_a\|_2 \quad \text{Joint Velocity Pen.} \tag{53}$$

$$p_c = -\lambda_c T \quad \text{Collision Pen.} \tag{54}$$

where $g_M = \frac{1}{|M|} \sum_{i \in M} \mathbf{1}(d_i < \alpha_g)$ is the initial grasped block's grasp fraction. With definitions and values in Table XIII. We set $w_{gs} = 7.5$, $w_r = 3.0$, $w_{fd} = 7.5$, $w_{sg} = 2.5$, $w_h = 20.0$, $w_p = 30.0$, $w_{int} = 1.0$, $w_v = 0.05$, $w_c = 0.2$.

Reward Rationale and Details. r_{gs}, p_{int}, p_v, and p_c are conceptually identical to prior subtasks. r_r draws the palm toward the second block. r_{fd} incentivizes the J fingertips to remain close to the second block. For this term only, we also include the palm as a contact point in J with the palm weighted by a factor of 2 to enable palm + fingertip grasps like our initial grasping stage. r_{sg} provides a binary signal for how many of the J fingertips are within threshold α_p of the block, promoting a secure grasp before lifting. The height reward r_h increases as the second block is lifted above its initial height h_0, but is gated by g_M to encourage the policy to maintain the original grasp while lifting. r_p increases as the second block approaches its goal.

Grasping	$\pi_1^{(2)}$	$\pi_2^{(2)}$	$\pi_3^{(2)}$	$\pi_4^{(2)}$	$\pi_5^{(2)}$	$\pi_6^{(2)}$	$\pi_7^{(2)}$	$\pi_8^{(2)}$	$\pi_9^{(2)}$
Sim	99.3	95.7	95.6	93.7	97.4	92.6	91.5	91.9	94.3
Real - Black	20	80	50	70	50	70	30	50	20
Real - Red	0	70	60	80	70	40	10	0	0
Real - Avg	10	75	55	75	60	55	20	25	10

TABLE XIV: Success rates of nine grasping policies in both simulation and real-world settings. Evaluations were conducted using two different blocks (see Fig. 5) over a 10 cm by 10 cm region in the real world for 10 trials each.

K. Sim-to-Real Experiments Details

When executing rollouts in simulation (results discussed in Sec. VII), we randomize the block position within a 10 by 10 cm region in front of the robot, matching the distribution of the data collected during simulation.

Before executing the full trajectories of the two subtasks, we first evaluated the nine grasping policies in the real world. The blocks (see Fig. 5) were placed uniformly across the testing region, and each policy was tested under the same conditions. The black block features a slightly deformable surface with higher friction, while both the black and red blocks are approximately 7 by 7 cm cube. Table XIV summarizes the success rates in both simulation and real-world settings, and Fig. 7 visualizes these results. We observe a clear distribution shift in success rates across different grasping policies. In simulation, all nine policies achieve consistently high performance with low variance (on average $94.7 \pm 2.5\%$). In contrast, real-world performance varies significantly across policies.

We hypothesize two main reasons for this discrepancy: (1) More challenging grasping policies rely heavily on real-time adjustments for successful execution. When deployed in an open-loop replay setting, the absence of feedback can significantly degrade performance. (2) Grasping policies that establish more stable contact exhibit a smaller sim-to-real gap. In particular, policies that utilize the palm ($\pi_1^{(2)}$, $\pi_2^{(2)}$, $\pi_3^{(2)}$, $\pi_4^{(2)}$, $\pi_5^{(2)}$) are less affected, as the larger and more stable contact surface provides greater robustness compared to fingertip-based grasps. This observation supports our design choice of explicitly incorporating palm-based grasping. Therefore, based on the above observations, in real-world

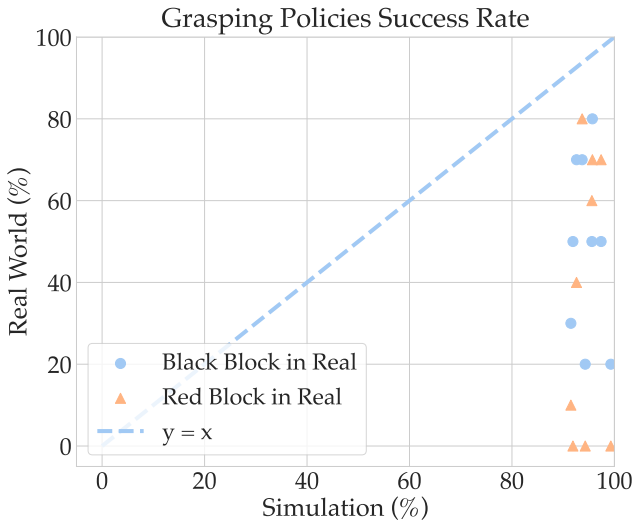

Fig. 7: Success rates of the nine grasping policies in simulation versus real-world settings. Each point represents a policy evaluated on either the black or red block. The dashed line $y = x$ denotes the ideal case with no sim-to-real gap, where success rates in simulation and real world are identical. While in reality, simulation performance remains consistently high, real-world results exhibit substantial variability, revealing a notable sim-to-real gap across different grasping policies.

experiments (see Sec. VII-B), we execute trajectories whose corresponding first subtask grasping policies both reach the final stage, i.e., C_2 of curriculum training and achieve the highest success rates in the preceding evaluations.