2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

SCOOP'D: Learning Mixed-Liquid-Solid Scooping via Sim2Real Generative Policy

Kuanning Wang[1], Yongchong Gu[1], Yuqian Fu[1], Zeyu Shangguan[2], Sicheng He[2], Xiangyang Xue[1†], Yanwei Fu[1†], Daniel Seita[2]

***Abstract*— Scooping items with tools such as spoons and ladles is common in daily life, ranging from assistive feeding to retrieving items from environmental disaster sites. However, developing a general and autonomous robotic scooping policy is challenging since it requires reasoning about complex tool-object interactions. Furthermore, scooping often involves manipulating deformable objects, such as granular media or liquids, which is challenging due to their infinite-dimensional configuration spaces and complex dynamics. We propose a method, SCOOP'D, which uses simulation from OmniGibson (built on NVIDIA Omniverse) to collect scooping demonstrations using algorithmic procedures that rely on privileged state information. Then, we use generative policies via diffusion to imitate demonstrations from observational input. We directly apply the learned policy in diverse real-world scenarios, testing its performance on various item quantities, item characteristics, and container types. In zero-shot deployment, our method demonstrates promising results across 465 trials in diverse scenarios, including objects of different difficulty levels that we categorize as "Level 1" and "Level 2." SCOOP'D outperforms all baselines and ablations, suggesting that this is a promising approach to acquiring robotic scooping skills. Project page: https://scoopdiff.github.io/**

I. Introduction

Scooping with tools like ladles or spoons is a fundamental skill in tasks ranging from cooking [1], assistive feeding [2], [3] and environmental cleanup [4]. While developing general-purpose robotic scooping could offer broad social and economic benefits, it remains a challenging problem due to the complexity of tool-object interactions and the need to handle mixtures of deformable materials such as liquids and granular media [5], [6]. Also, observations are often unreliable due to occlusions and fluid surfaces that cause reflections, refractions, and unstable depth sensing, which makes scene representations noisy and unreliable.

Prior work has explored scooping a single item [7] or using one type of granular medium [8], which significantly simplifies the task. In contrast, we aim to advance scooping to the next level by addressing more realistic and challenging settings—scooping from containers that have a mixture of liquids and multiple solid items, where distractor objects are often present. To tackle this, we equip a robotic manipulator arm with a ladle as its end-effector and task it with retrieving a subset of target items from these complex environments.

Since collecting real-world data for robotic scooping is time-consuming, costly, and potentially dangerous, we use a

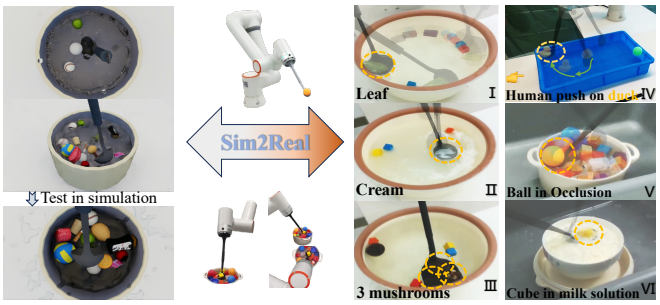

Fig. 1: **Examples via Sim2Real.** Our method, SCOOP'D, trains on data collected in simulation and generalizes to diverse real-world scenarios. Subfigures I–VI show examples with yellow circles marking target objects.

Sim2Real learning paradigm for efficient, scalable, and safe data collection, as done in other robotics applications [9], [10]. Moreover, scooping floating objects requires smooth and precise control, but reinforcement learning suffers from low sample efficiency and jerky motions [7], motivating our imitation learning approach. Based on OmniGibson [11], a recent simulator built on NVIDIA Omniverse, we implement an algorithmic demonstrator that provides demonstrations while using "privileged" ground-truth object state information from simulation. We obtain a new simulated multimodal scooping dataset (***SimScoop***) with 6,480 demonstrations.

We further propose a novel method, **SCOOP**ing with **D**iffusion (SCOOP'D), which learns robotic scooping from demonstrations (e.g., from *SimScoop*). SCOOP'D contains two Diffusion Policy models [12], where one learns a good initial "pre-scoop" ladle pose and the other learns fine-grained scooping motions. Our method is quick to train and benefits from advanced vision-based foundation models [13] such as SAM2 [14]. Critically, as in Fig. 1, our learned policy can be deployed directly to diverse real-world scenarios without any extra fine-tuning. Quantitatively, our method achieves over 80% success across 240 real-world trials on "Level 1" objects, outperforming baselines and ablations. This is a challenging zero-shot setting under a strict success criterion: scooped objects must contain exactly the target(s) and no others. Extensive experiments show that SCOOP'D generalizes across objects, multiple targets, varying occlusion severity ("Normal" and "Severe"), liquids, and containers.

To summarize, the contributions of the paper include:

- A simulation-based environment in OmniGibson for synthesizing diverse scooping demonstrations, along with the resulting 6,480-demo *SimScoop* data.
- The novel SCOOP'D method for learning from state-based demonstrations in simulation for Sim2Real transfer that leverages generative policies via diffusion for

[1] Fudan University, China. [2] Thomas Lord Department of Computer Science, University of Southern California, USA
† indicates the corresponding author

pre-scoop pose estimation and scooping motions.
- Extensive experiments across 465 real-world trials demonstrate promising and competitive results, with strong generalization across varied scenarios.

II. Related Work

A. Scooping and Manipulation of Deformable Objects

Scooping is a core challenge in robotic manipulation, with prior work addressing liquids alone [15], liquid-solid mixtures [16], granular media [8], and dough [17]. A common use case of robotic scooping is in assistive feeding [18], [19], [3], [20], where a robot with a fork or spoon retrieves appropriately-sized food items from a plate or bowl to provide to a user. Our work takes inspiration from assistive feeding in designing a generalizable Sim2Real scooping pipeline to handle multiple (typically solid) objects in liquid. Other prior work in robotic liquid manipulation focuses on complementary tasks, such as pouring [21], [22], [23] or understanding fluid dynamics [24].

Closely related prior work includes ToolFlowNet [7], SCONE [25], and LAVA [16]. ToolFlowNet [7] studies imitation learning from point cloud data and predicts dense 3D movement of tool points. During scooping, this approach outperformed reinforcement learning baselines which exhibited jerky and rapid motions. However, it assumes that the trajectory data is unimodal, but such data is often multimodal. It also struggles to track the object when displaced by the ladle. SCONE [25] uses active perception to interact with a solid or granular material in a bowl before scooping. LAVA [16] proposes a hierarchical policy framework that divides the task of scooping into high-level decision-making, mid-level action refinement, and low-level execution. Both SCONE and LAVA rely on demonstrations via kinesthetic teaching, which can be cumbersome to obtain. We use simulation to avoid collecting physical demonstrations. We also study solid-*liquid* manipulation, resulting in more complex object movements at test time compared to the tasks from [25].

B. Imitation Learning from Simulation

To learn scooping, we use imitation learning [26], which trains a model to mimic actions from demonstrations. Imitation learning methods include behavioral cloning [27] and inverse reinforcement learning [28]. More recent techniques predict a *sequence* of actions to mitigate distribution shift [29]. In this direction, we leverage Diffusion Policy [12], a popular approach for imitation learning that uses a diffusion model [30] to effectively deal with multimodal and high-dimensional action distributions. In this work, we have two separate Diffusion Policy models. The first predicts a "pre-scooping" ladle pose for better initialization, while the second predicts fine-grained scooping actions.

To get demonstrations, we design an algorithmic demonstrator that uses ground-truth information in simulation. This strategy is inspired by other Sim2Real works such as fabric smoothing [31] and scissor cutting [32], and we adapt it to scooping. However, popular simulators in the robot learning community, such as PyBullet [33], MuJoCo [34], and IsaacGym [35] do not support liquid manipulation. Other works that study liquid manipulation in simulation include SoftGym [36], FluidLab [37], DAXBench [38], and OmniGibson [11]. We empirically find that OmniGibson has the best combination of simulation accuracy and usability.

C. Perception for Sim2Real and Object Detection

In this work, we learn a policy using Sim2Real, without real-world data collection. While techniques such as domain randomization over images [10], [39] have been beneficial for learning complex vision-based manipulation tasks [40], [9], there remains a large visual Sim2Real gap [41] between scooping in simulation and the real world, where reflections and occlusions further hinder perception. We address this gap by using a lower-dimensional state representation that consists of object poses. Object pose estimation is a well-studied problem in robotics [42] and in recent years, pre-trained foundation models [13] such as GroundingDINO [43], [44] and SAM2 [14] have facilitated generalizable pose estimation. We use these methods for real-time object segmentation and tracking, which helps us estimate object poses.

III. Proposed Dataset and Method

Problem Statement: We study robotic scooping from a container on a tabletop with a mixture of liquid and solids. We use water as the liquid medium, which supports multiple floating solid objects. The robot is a standard manipulator equipped with a ladle as its end-effector. The robot executes actions to adjust the ladle's 6-DoF pose. An RGBD camera provides image data each time step. We define a *trial* as an instance of the robot scooping task. At the start of each trial, a human places a mixture of items in the container. A text prompt informs the robot of the target item(s) to be scooped.

Since generalizable robotic scooping requires manipulating highly complex liquid-solid mixtures, we propose to learn scooping from demonstrations $\mathscr{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$ consisting of observations $\mathbf{o}_t$ and expert actions $\mathbf{a}_t$ at each time step t. We use Diffusion Policy [12] and disentangle the scooping task into two key steps: (i) reaching the pre-scoop ladle pose from a default pose and (ii) executing the subsequent trajectory. Consequently, we use two Diffusion Policy models for these steps. The first, f_ϕ, predicts the pre-scoop ladle pose. The second, π_θ, produces delta scooping actions $\mathbf{a}_t$. To enhance the model's practicality, we equip it with the ability to determine the target item(s) based on the given text description. To achieve this, we first feed the text prompt to GroundingDINO [43], [44] to obtain the target bounding box(es). The box(es) are used as the visual prompt for SAM2 [14] which provides real-time object segmentation and tracking, enabling object pose estimation. Then, we design a geometry-aware network g_ψ based on PointNet++ [45] to extract the object's states.

In the following, we present how we collect data in simulation and develop SCOOP'D. See Fig. 2 for an overview.

A. SimScoop Dataset

Simulation Environment. We use OmniGibson [11], which is powered by NVIDIA Omniverse. As shown in Fig. 2,

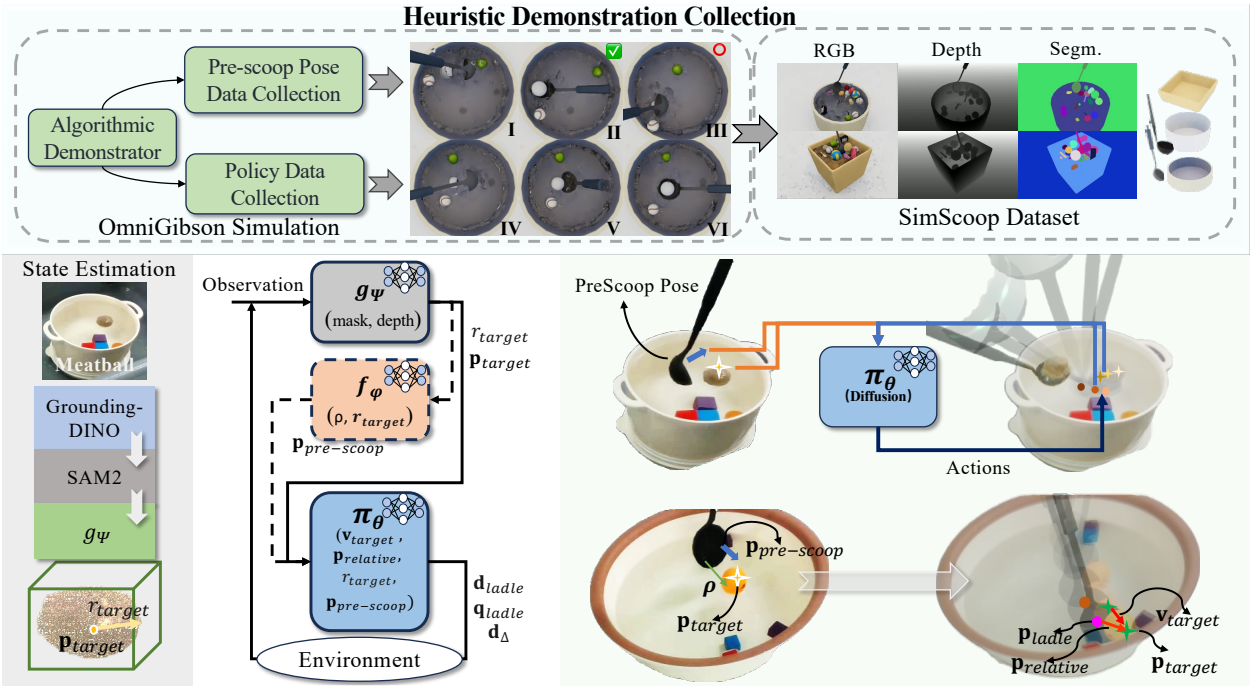

Fig. 2: **Our SCOOP'D Method**. The first row shows the heuristic demonstration collection. Using OmniGibson simulation, we leverage an algorithmic demonstrator for SimScoop dataset collection. The second row shows how deployment works. The left part shows how we obtain the state of the target item from text ("meatball"), detection, live video stream segmentation, and regression with the partial point cloud. The middle part shows the pipeline of our method. We use f_ϕ to generate a pre-scoop pose based on ρ and r_{target}, then move the ladle directly to the generated pose. Then we leverage π_θ for closed-loop scooping. Our π_θ takes in $\mathbf{p}_{\text{relative}}$, $\mathbf{v}_{\text{target}}$, $\mathbf{p}_{\text{pre-scoop}}$ and r_{target}, and outputs $\mathbf{a}_t$; f_ϕ is executed only once. The right part shows the execution. We demonstrate the execution process in both the top and bottom containers, with the states specifically marked in the bottom for extra clarity.

our simulation mainly contains different containers, several objects (e.g., balls), and ladles. The diameter and height of the first-row middle container are about 0.4 m and 0.2 m. We modify the number of maximum micro-particle samples in the simulation environment to change the volume of water in the container. However, we do not tune the physical parameters of the simulator to align it with real-world liquids.

Heuristic Scooping Strategy. To collect demonstrations, we implement a motion-adaptive heuristic scooping strategy that uses ground-truth simulation information; it defines a curve for the ladle's bowl to move it below the target and then lifts up the ladle. The curved trajectory allows gentler water entry, provides a gradual approach in cluttered scenes, and positions the ladle to enclose targets, which can help capture and improve adaptability to different object states. As sketched in Fig. 3, the ladle's position is the origin of its coordinate frame, at the bottom center of its bowl. Let v be the vertical distance from the target item (center) to the ladle, and ρ be the horizontal distance from the target item to the ladle's bowl. The ladle moves in a circular arc about a point which is at distance h directly above the target. After this, the ladle's bowl is under the target and faces up. Through these quantities, we can compute the ladle's pose and moving direction for each state. We directly use object position together with constraints among parameters to guide efficient sampling.

However, this does not account for how the target may move, which requires shifting the ladle's trajectory. Let $d \in \mathbb{R}^3$ represent the small 3D movement of the target in a time step. To maintain a stable circular motion when the target

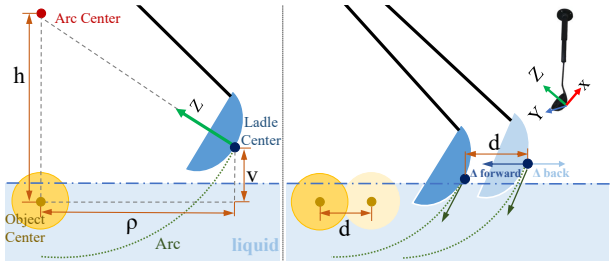

Fig. 3: **Heuristic scooping strategy.** We sketch a ladle and a target item (circle). The ladle's center of rotation is at the bottom of its "bowl." It follows the dotted circular arc to go underneath the item, and then lifts up.

moves, we first shift the ladle according to d, and then the ladle follows its circular trajectory. Furthermore, while this gives us a basic scooping trajectory, the ladle may contact the container wall while scooping. If the ladle hits the wall, our heuristic strategy immediately moves the ladle away from it. Once the ladle's bowl is sufficiently close and directly underneath the target, the ladle moves upwards to scoop it.

Collecting Simulated Training Data. We collect two datasets in simulation: a **small** dataset with 600 demonstrations and a **large** dataset with 6,480 demonstrations.

For the **small** dataset, we divide the simulated data collection into two parts: (i) a pre-scoop pose and (ii) a scooping motion. See Fig. 2 for a visualization. In both data collection phases, we use a "PoolBall" object as the target object, since it leads to relatively stable simulation performance. During our experiments, we generalize to scooping other objects. We also sample other objects as "obstacles" in simulation.

To collect ladle pre-scoop pose candidates, we sample object size and position, and sample ladle pose parameters to avoid object-ladle collisions. We initialize the ladle at this

sampled pose and execute the heuristic scooping strategy. We retain a ladle pose for training only when the heuristic method achieves a successful scoop with a limited number of collisions before lifting. This setup, inspired by pre-grasp poses [46], provides coherent starting points for scooping, improving robustness, and facilitating efficient policy learning across varied trajectories. By explicitly generating valid pre-scoop poses, and training a generative model to predict poses (see Sec. III-B) we assist the subsequent scooping. For policy data collection, we also sample object size and position. The ladle is initialized by the learned pre-scoop pose generation module. We then apply the heuristic scooping strategy, inserting a "*offset movement*" upon ladle-target collisions to slightly offset the ladle from the target (this step is omitted in pre-scoop pose collection). These interventions generate data for collision-recovery scenarios caused by imperfect initialization or movement noise.

When collecting training data, the pre-scoop pose collection has a 64% success rate, while the policy demonstration has an 82% success rate. We only keep successful cases.

As shown in Fig. 2, we also construct more complex scenes with containers, ladles, and objects of varying types and sizes, along with more distractors. We use the heuristic method to collect data in a single stage, by first sampling a pre-scoop pose and then executing a scoop. These setups introduce significant motion uncertainty due to ladle-object interactions, liquid flow, and clutter. The dataset includes RGB, depth, and segmentation images from three views, along with object and ladle states and motion data. This **large** data has 6,480 demonstrations, each with 80 frames, totaling over 518k frames and 1.5M RGBD images.

B. SCOOP'D Methodology

Pre-Scoop Poses Generation. We employ a pre-scoop pose, inspired by the human tendency to select a coherent entry point for smoothly scooping a target from the mixture. We use a 1D CNN-based Diffusion [12] model f_ϕ, to generate pre-scoop ladle poses. The goal of f_ϕ is to produce a diverse set of effective pre-scooping poses for different containers, objects, and ladles, based on fundamental properties of the target object (such as its radius r_{target}) and human-controllable parameters (ρ). The observations, which are the object's estimated radius r_{target} and our manually specified value ρ, are fed into f_ϕ, which predicts the values of h and v. These predicted values, together with ρ and the object's position, determine the ladle's pre-scoop pose.

Meanwhile, the ladle starts from a random pose and directly moves to the pre-scoop pose, during which the object may move. Although the pre-scoop pose is fixed, the ladle tracks the object's recent motion upon arrival, effectively achieving a state as if the object were stationary.

Diffusion Policy for Scooping. We also use a 1D CNN-based Diffusion Policy model π_θ to execute scooping. The **input** $\mathbf{o}_t$ to π_θ is a 10D vector containing (i) the 3D relative position of the target object: $\mathbf{p}_{\text{relative}} = \mathbf{p}_{\text{target}} - \mathbf{p}_{\text{ladle}}$, (ii) the 1D estimated radius of the target r_{target}, (iii) the 3D parameterized representation of the pre-scooping pose $\mathbf{p}_{\text{pre-scoop}} = (\rho, h, v)$, and (iv) the 3D movement of the target in the previous step: $\mathbf{v}_{\text{target}} = \mathbf{p}^{(t)}_{\text{target}} - \mathbf{p}^{(t-1)}_{\text{target}}$.

The **output** of π_θ is a 10D vector $\mathbf{a}_t$ that contains (i) the 3D direction of ladle's main movement: $\mathbf{d}_{\text{ladle}} = [d_x, d_y, d_z]$, executed with a constant step length s, (ii) the 4D ladle orientation represented by a unit quaternion $\mathbf{q}_{\text{ladle}} = [q_x, q_y, q_z, q_w]$, and (iii) the 3D *offset movement*, $\mathbf{d}_\Delta = [\Delta x, \Delta y, \Delta z]$. These parameters determine the ladle's motion, while the ladle also directly follows $\mathbf{v}_{\text{target}}$ as mentioned in Sec. III-A. Therefore, the aggregated motion of the ladle is:

$$\mathbf{d}_{\text{agg}} = s \cdot \mathbf{d}_{\text{ladle}} + \mathbf{d}_\Delta + \mathbf{v}_{\text{target}}.$$

As mentioned earlier, $\mathbf{p}_{\text{relative}}$ considers the relative motion between the object and the ladle. Our strategy aligns the object and ladle dynamics into a shared representation, which reduces the complexity and captures the essential motion patterns. Our f_ϕ outputs the pre-scoop pose, which is important for π_θ to understand the current state of the ladle and the heuristic trajectory to scoop the target object. Therefore, we condition π_θ on parameters of the pre-scoop pose: ρ, h and v. See the Appendix for model details.

Geometry-Aware Localization and Scale Estimation. In addition to f_ϕ and π_θ, our system incorporates a third learned component, a network g_ψ based on PointNet++ [45], which predicts an object's center and longest radius from its point cloud. The longest radius serves as a scale reference that enables consistent handling across different shapes and helps reduce collision risks during manipulation. We first use SAM2 [14] on the image to obtain a coarse segmentation of the target object. While SAM2 provides reasonable masks, directly estimating geometric properties from its segmentation can be unreliable under partial occlusions or segmentation noise, especially when liquids or surrounding obstacles limit depth visibility. Instead, we convert the segmented depth region into a point cloud and feed it into g_ψ, which captures fine-grained 3D geometric details to estimate the center and longest radius robustly. To train g_ψ, we leverage the YCB dataset [47], which offers diverse objects commonly used in manipulation tasks, including several basic shapes that resemble the custom objects used in our scooping scenario. Using PyRender, we render depth images of YCB objects, generate normalized point clouds, and train g_ψ from scratch for accurate center and scale estimation.

Sim2Real Transfer. To account for the Sim2Real physics gap, we add slight random noise to target and ladle motions during data collection for robustness. In deployment, we only calibrate the camera and deploy the policy directly.

C. Application to Diverse Scooping Scenarios

When we collect training data, we get scooping demonstrations of one object at a specific region of the container. However, we also apply SCOOP'D to alternative scenarios.

a) Objects in Different Regions: A practical difficulty with real-world robotic scooping is hitting the robot's kinematic limits. To address this for the circular containers we test, we divide each into four (non-equal) parts around the container's center. The part directly opposite the robot and the two lateral parts are roughly equal in size, while the part

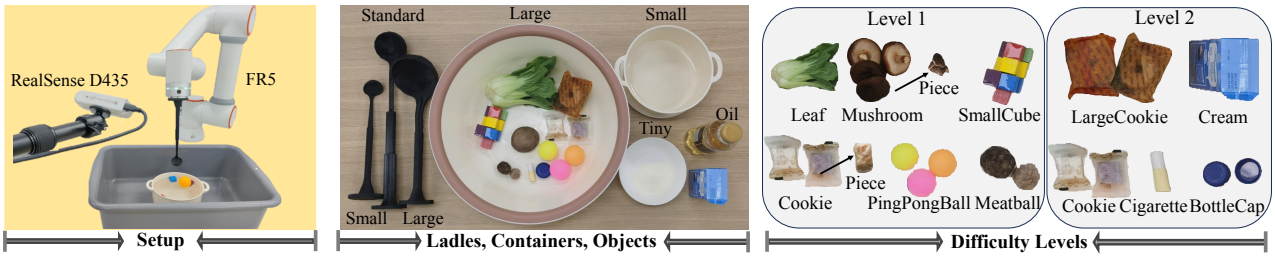

Fig. 4: **Real-world experimental setup.** Left: a third-person view of the setup. A third-person RealSense D435 camera captures RGBD image observations. Middle: we show different ladles, containers, and objects that we use during physical experiments. The robot shown above (to the left) is holding the smallest ladle and operating on the small container (shown in the upper right corner). Right: we show "Level 1" and "Level 2" (i.e., more challenging) objects that we use in our scooping experiments. See Sec. IV-A for more details.

closest to the robot is smaller and reflects where the robot may hit kinematic limits. We adjust the policy model's input and output for consistency with demonstrations, applying region-based rotations: no rotation for the opposite area, 90° and 270° for the left and right, and 180° for the closest region. We do these straightforward calculations from standard observational data. Therefore, we can directly adapt the Diffusion Policy models f_ϕ and π_θ to handle targets everywhere in the container.

b) Sequential Multi-Object Scooping: Learning to scoop multiple objects directly from imitation learning is challenging, since a robot must prevent scooped objects from falling out, handle scattered object positions, and manage the size disparity between the ladle and objects. To address this, we sequentially scoop one item at a time, keeping previously scooped items in the ladle. After each scoop, the ladle lifts and moves to the pre-scoop pose for the next object.

IV. Experiments

A. Experimental Setup

Simulation Setup. We construct two types of scooping environments by occlusion severity: (1) a simple setting with a single target object placed randomly in a large container (diameter 0.4 m), and (2) a challenging setting with a smaller container densely filled with distractors, leading to severe congestion. In both settings, we use the same ladle tool and camera configuration (Fig. 2, top center). To simulate realistic conditions, the initial positions of all objects are randomized across trials.

Real-World Setup. See Fig. 4 for our setup. We use the low-cost ($4,000) 6-DoF Fairino Robot 5 (FR5), and attach a ladle as its end-effector. We test three mixture containers with tiny, small, and large dimensions, and three ladles in small, standard, and large sizes, where size-matched ladles improve precision by avoiding obstacles in cluttered scenes. The experiments use objects inspired by assistive feeding and environmental clean-up applications. Example objects include food (e.g., mushrooms) and litter (e.g., bottle caps).

We categorize objects into two difficulty levels: Level 1 (easy) and Level 2 (hard). Each object presents its challenges—such as the leaf being lightweight and easily moved, the PingPong ball's fast motion, and the mushroom's irregular shape. See Figure 4 for our categorization; the cookie can be either Level 1 or 2 depending on the ladle, as the size relationship between the ladle's bowl and the object significantly impacts the scooping difficulty.

As a pre-processing step before each trial, we move the robot to random positions within the environment and capture image observations. The pre-scoop pose is generated first, providing an ideal starting point, then the robot executes the policy π_θ to scoop the targets. We train f_ϕ on the small dataset, and π_θ on varying scales (small or large); unless noted, π_θ is trained on the small dataset. See the Appendix for more details about the network and training process.

Evaluation Metrics. For single-object scooping, a success is scooping *exactly* the target object above a height threshold. A failure is any other scenario, including when the target and non-targets are scooped together. For multi-object scooping, we employ several evaluation metrics (see Table III).

B. Baselines and Ablations

1) Baselines: In the real world, we compare against six strong baselines: **LAVA** [16], **Heuristic Method** (Sec. III-A), **RGB-Based Diffusion Policy**, **Real-world State Diffusion Policy**, and the vision-language-action model from Physical Intelligence, π_0 **(zero-shot and fine-tune)** [48].

Since LAVA's code is unavailable, we re-implement its low-level wall-guided scooping policy for center and edge object placements. To simplify alignment, we use our segmentation module and manual object alignment, significantly reducing the difficulty—LAVA's rule-based open-loop alignment struggles with floating objects. Items are manually placed near center or wall, with liquid-induced perturbations, and we evaluate success rates using its wall-guided policy.

For the heuristic method, since we test in real, we use estimated rather than ground-truth object states. Since detecting collisions from the camera is challenging, our adaptation omits the "move-away" action when the ladle collides with the container. We test two versions of the method in real: one that uses our pre-scoop pose generation module, and another that relies on sampled pre-scoop poses, following the same process as pre-scoop pose data collection.

To compare with the following two baselines, we trained SCOOP'D on 50 randomly sampled demonstrations from the original 600-demonstration dataset (in simulation). For the RGB-based Diffusion Policy, we sampled 50 scooping demonstrations with ping pong balls and small cubes in the container (in the real-world), and train an end-to-end network. For the real-world state Diffusion Policy, we also use 50 manually demonstrated scooping trajectories with ping pong balls and small cubes. Object states of demonstrations

Occlusion Severity	Data Scale	Apple	PoolBall	Strawberry	SoftBall	Cork	Egg	Average
Normal	Small (600 demos)	18/20	17/20	18/20	18/20	20/20	19/20	91.7%
Severe	Small (600 demos)	15/20	15/20	14/20	14/20	13/20	17/20	73.3%
Severe	Large (6,480 demos)	17/20	16/20	14/20	17/20	14/20	18/20	80.0%

TABLE I: **Scooping results in simulation.** We report the success rate of SCOOP'D over 20 trials for each of six objects in simulation. The last column averages the success across all six objects. All results are based on an action horizon of 1. See Sec. V-A for more details.

Container-Ladle Size	PingPongBall	Mushroom	Cookie	SmallCube	Leaf	Meatball	Average
Large-Standard	18/20	17/20	17/20	16/20	16/20	17/20	84.2%
Small-Small	19/20	16/20	15/20	17/20	15/20	15/20	80.8%

TABLE II: **Real-world scooping results.** We study the effect of the container and ladle size for scooping six items under "Normal" occlusion severity. We test with a large container and a standard ladle (top row) and a small container with small ladle (bottom row). We conduct 20 trials of SCOOP'D and report the success rate. In all cases, we use an action horizon of 1. We train π_θ on our small dataset. This table reports the **Level 1** objects for each ladle.

were estimated via an external camera, and the policy was trained end-to-end without a pre-scoop pose.

We also use π_0, a state-of-the-art vision-language-action model, as a baseline. The language instruction is one sentence, and requests to scoop the target. We test π_0 in two settings: zero-shot and fine-tuned on 50 demonstrations.

2) Ablations: We study four ablations trained on the original 600 demonstrations of SCOOP'D: First, we remove the pre-scoop pose generation f_ϕ. Instead, we get the pre-scoop pose using the same sampling method as we did for collecting pre-scoop pose data. Second, we randomly sample the pre-scoop poses within a sufficiently large sparse space, and then execute the policy. Third, we remove the PointNet++ module g_ψ by directly using the average position of the partially segmented point cloud as the object's center. The radius is determined by the longest distance from any point in the cloud to this center. Fourth, for the target's relative motion, we directly input the target's position and the ladle's position into the policy, instead of using $\mathbf{p}_{\text{relative}}$.

V. Results

A. Simulation Results

Table I reports our main simulation results for scooping. We evaluate scooping six objects for 20 trials each, and for different occlusion severities and data scales for π_θ. The success rate is 91.7% under "Normal" occlusion severity and "Small" data scale. The table also presents results under "Severe" occlusions, where the success rate declines but is still relatively high at 73.3%. During tests, we observe that the target frequently moves beneath the ladle and nearby objects. Training on our large dataset improves scooping performance in occluded scenes, increasing the success rate from 73.3% to 80%. A "Severe" case is shown in the third row of Fig. 5. We also observe two common failure modes. First, some "correct" scooping motions nonetheless fail when objects fall through the ladle bowl due to simulation inaccuracies in collision checks. Second, GroundingDINO [43] might not correctly detect objects, which may be because its training data is primarily based on real-world images.

B. Real-World Results

We show our main real-world scooping results in Table II. SCOOP'D achieves 82.5% success rate across 240 trials with various Level 1 objects, containers, and ladles, highlighting the strong generalization ability of our Sim2Real approach.

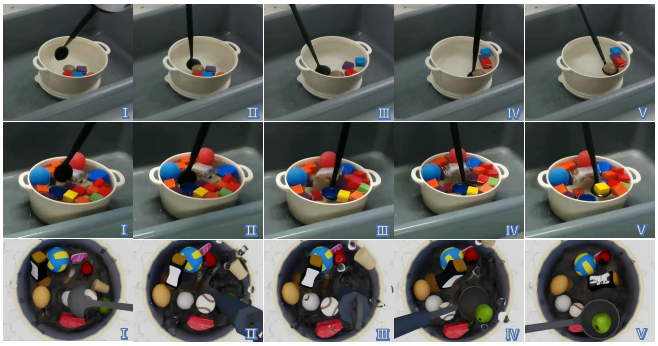

Fig. 5: **Visualization of Scooping.** We show our learned policy scooping targets in clutter. We show when it reaches the pre-scoop pose (I), when it moves towards the target (II, III, IV), and finally when it successfully scoops it (V). The first row shows the meatball in a mildly occluded scene, the second row depicts a yellow cube under heavy occlusion in real, and the third row presents a green apple in a severely occluded scene in simulation.

Despite the challenging objects with diverse physical properties, the scooping performance among all objects is $\geq$75% for each experiment setting. For different ladles, using the standard ladle shows a similar success rate as the small ladle, despite how we only collect the small dataset for one ladle size. Our method is also insensitive to the action horizon, as shown in the Appendix.

The failure cases mainly stem from kinematic limits of the robot hardware. One failure comes from losing sight of the object, which could be optimized by using multiple cameras or an eye-on-hand [49] camera. Other cases include where, while scooping the target object, other objects that we do not want to scoop are also lifted.

See Fig. 1 (I) for a qualitative result. The leaf's irregular shape and lightweight nature often cause it to float away, but it is successfully scooped. See Fig. 5 for another example, where we scoop a meatball near obstacles using the small ladle. Unlike Fig. 2, where the meatball is scooped out in one motion, obstacles affect the perception of its state, causing it to slide away. Both of the qualitative results indicate that SCOOP'D generates valid pre-scoop poses, and that our policy can scoop in challenging dynamic situations.

a) Scooping Multiple Objects: See results in Table III for scooping up to 3 objects. For PingPongBall scooping, the average number of scooped targets is higher when scooping 2 targets compared to 3. The main reason is that when scooping a new object, the ladle needs to hold the previously scooped items. As a result, there are often failures when scooping the third item because the ladle cannot contain the first two. On the other hand, for objects like mushrooms with irregular

Object (Count)	average targets (↑)	average obs (↓)	success w/o obs (↑)	success w/obs (↑)
SmallCube (1)	1	0	5/5	5/5
SmallCube (2)	1.8	0	4/5	4/5
SmallCube (3)	2.4	0.4	3/5	1/5
PingPongBall (1)	0.8	0.2	4/5	3/5
PingPongBall (2)	1.6	0.6	4/5	1/5
PingPongBall (3)	1.2	0.2	1/5	1/5
Mushroom (1)	0.8	0	4/5	4/5
Mushroom (2)	1.6	0.2	4/5	3/5
Mushroom (3)	1.6	0	1/5	1/5

TABLE III: **SCOOP'D results for scooping multiple objects in the real world.** SmallCube: For 1 and 2 objects, we use the standard ladle size. For 3, we use the large version. PingPongBall: We use the large ladle to scoop 1, 2 and 3 balls, as it can hold up to three PingPongBalls. Mushroom: We use the large ladle to scoop 1, 2, or 3 mushrooms. This ladle can also hold up to three. For each row, we present (respectively): the average number of scooped targets (higher is better), the average number of obstacles scooped (lower is better), the success rate of scooping a specific number of targets without and then with considering obstacles (w/o obs, w/obs).

Occlusion	Data Scale	SmallCube	PingPongBall	Average
Normal	Small (600 demos)	17/20	19/20	90.0%
Severe	Small (600 demos)	12/20	12/20	60.0%
Severe	Large (6,480 demos)	15/20	14/20	72.5%

TABLE IV: **Comparisons of different objects and occlusion severity for real-world scooping.** See Sec. V-B for more discussion. The "Normal" results are the same as those of SmallCube and PingPongBall in Table II.

shapes or small cubes, it is easier to keep the scooped items stable without them falling off the ladle or flowing away while scooping additional items. The probability of obstacles being scooped increases as more objects are scooped. See Fig. 1 (III) for a qualitative result.

b) Different Liquids: We test changing the liquid and observe that SCOOP'D shows some degree of "liquid generalization." Due to space limitations, we defer details to the Appendix. See Fig. 1 (VI) for a qualitative result.

c) Different Occlusion Severity: In Table IV, we show experimental results with varying numbers of distractors. The "Normal" setting corresponds to the first row in Fig. 5 (distractors $<30\%$ of surface), while the "Severe" setting (see the second row of Fig. 5) involves heavy occlusion with a dense container filled with distractors. We compare the scooping performance for different data scales and occlusion levels. Compared to the "Normal" severity, performance drops under "Severe" occlusion, as expected. However, training on our larger dataset increases the performance under the same "Severe" occlusion, demonstrating the potential for improvements by scaling up the data.

d) Data Scalability: We test scalability by training on datasets of different sizes. As shown in Table I and IV, success rates improve in occluded scenes, indicating that our model effectively benefits from larger and more diverse data.

e) Baselines: For LAVA, results in Table V show that SCOOP'D outperforms LAVA; we observe that LAVA's fixed trajectories do not scale well. A slight disturbance can cause objects to move away, making them difficult to reach the center. Moreover, pushing floating objects to the wall and scooping can also cause drift or squeeze them out.

For the "Heuristic, sample pre-scoop pose" method, when scooping the small cube, the ladle often collides with the cube before reaching underneath it. This causes the cube or even the container to move heavily, making it even harder to scoop using the heuristic method compared to our policy.

Method	SmallCube	PingPongBall
SCOOP'D (ours, 600 demos)	17/20	19/20
SCOOP'D (ours, 50 demos)	13/20	14/20
Baselines		
RGB-Based Diffusion Policy (50 demos)	4/20	3/20
Real-world State Diffusion Policy (50 demos)	6/20	5/20
π_0 (zero-shot) [48] (50 demos)	0/20	0/20
π_0 (fine-tune) [48] (50 demos)	2/20	4/20
LAVA* [16]	7/20	6/20
Heuristic, sampled pre-scoop pose	5/20	16/20
Heuristic, w/pre-scoop pose	9/20	16/20
Ablation Studies		
SCOOP'D sampled pre-scoop pose (600 demos)	15/20	18/20
SCOOP'D w/o pre-scoop pose (600 demos)	0/20	0/20
SCOOP'D w/o PointNet++ (600 demos)	14/20	17/20
SCOOP'D w/o relative motion (600 demos)	7/20	14/20

TABLE V: **Comparisons of different methods and ablations for real-world scooping.** The "*" denotes simplified conditions for LAVA. See Sec. IV-B and V-B for more discussion. The SCOOP'D results are the same as those of SmallCube and PingPongBall in Table II.

For "Heuristic, w/pre-scoop pose," the success rate of scooping the small cube improves as our f_ϕ generates more reliable poses. However, without the policy to adjust the movement, collisions still occur. Additionally, the heuristic methods are rigid and do not generalize well in the real world, leading to collisions during the ladle's movement.

For the "RGB-Based Diffusion Policy," visual features fail to accurately capture the spatial relationship between the ladle (underwater) and the object, leading to imprecise scooping. The ladle often misses the target, and even when it gets close, suboptimal contact angles or positions tend to push the object out of the container, causing failure.

For "Real-world State Diffusion Policy", unlike in simulation, where a structured approach first moves the ladle to a pre-scoop pose, the real-world policy directly predicts robot motions in a larger action space, increasing reliance on abundant high-quality data and reducing training efficiency.

For "π_0 (zero-shot)," the robot moves almost randomly. For "π_0 (fine-tune)," it moves toward the target but inaccurately, often colliding and failing. This may be because π_0 is mainly trained on tasks with grippers, and thus lacks exposure to ladle use and scooping applications.

f) Ablations: We show results in Table V. The "sampled pre-scoop pose" shows a slight decrease in the success rate for the small cube. Qualitatively, there are significantly more collisions when the ladle gets closer to the cube, causing the ladle to push the object and occasionally collide with the container. Despite this, due to our well-trained policy, the cube can still be scooped after a few additional steps. For "w/o pre-scoop pose," while our trained policy can often get closer to the target, accurately scooping the object in a closed-loop manner is still highly challenging. For "w/o PointNet++," we attain close (but worse) performance compared to SCOOP'D, suggesting that our PointNet++ module is necessary for accurate object state predictions. Failure cases here are largely due to inaccurate object center estimations. For "w/o relative motion," the success rate is much lower than that of SCOOP'D. Intuitively, relative motion reduces the complexity, making optimization easier.

g) Scooping Level 2 Objects: Table VI reports results for scooping challenging objects with SCOOP'D, which are

	LargeCookie	Cookie	BottleCap	Cigarette	Cream	Average
SCOOP'D	11/20	9/20	9/20	12/20	7/20	48.0%

TABLE VI: **Real-world SCOOP'D results on Level 2 items.**

lower than in Table II but still show robustness under these highly challenging unseen conditions. For LargeCookie, we use a standard-size ladle to scoop the brown large cookie with wrapping paper, and for Cookie, we use the small ladle to scoop the purple cookie with wrapping paper. This is challenging since the objects have an uneven mass distribution and are much larger than the ladle's bowl. Other objects bring their own challenges: BottleCap can sink, SAM2 often fails to segment Cigarette, and Cream spreads easily.

Analysis on limitations. While SCOOP'D shows some generalization, it may struggle with more complex scooping tasks, such as scooping submerged objects deep in the container or deformable items such as cream. Moreover, SCOOP'D takes pretrained GroundingDINO and SAM2 models directly for object localization, without finetuning them for better performance. These are left for future work.

VI. Conclusion

In this paper, we present SCOOP'D, a method for learning scooping policies from efficient algorithmic demonstrators in simulation. Our method uses Sim2Real generative models to imitate scooping behavior. Experimental results across a variety of real-world scooping scenarios suggest that SCOOP'D obtains promising success rates. We hope this inspires future work on autonomous and generalizable robotic scooping.

ACKNOWLEDGMENT

The paper is supported by Shanghai Municipal Science Technology Major Project (2025SHZDZX025G02), and National Natural Science Foundation of China (62521004).

References

[1] J. Liu *et al.*, "Robot cooking with stir-fry: Bimanual non-prehensile manipulation of semi-fluid objects," *RA-L*, 2022.
[2] S. W. Brose *et al.*, "The role of assistive robotics in the lives of persons with disability," *American journal of physical medicine & rehabilitation*, 2010.
[3] D. Park *et al.*, "A multimodal execution monitor with anomaly classification for robot-assisted feeding," in *2017 IROS*, 2017.
[4] N. Ruangpayoongsak *et al.*, "A floating waste scooper robot on water surface," in *2017 ICCAS*, 2017.
[5] J. Sanchez *et al.*, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *IJRR*, 2018.
[6] J. Zhu *et al.*, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics & Automation Magazine*, 2022.
[7] D. Seita *et al.*, "Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds," in *CoRL*, 2022.
[8] C. Schenck *et al.*, "Learning robotic manipulation of granular media," in *CoRL*, 2017.
[9] M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *IJRR*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:51894399
[10] J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IROS*, 2017.
[11] C. Li *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *CoRL*, 2022.
[12] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257378658
[13] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.
[14] N. Ravi *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.
[15] Y. Niu *et al.*, "Goats: Goal sampling adaptation for scooping with curriculum reinforcement learning," in *2023 IROS*, 2023.
[16] A. Bhaskar *et al.*, "Lava: Long-horizon visual action based food acquisition," in *2024 IROS*, 2024.
[17] C. Qi *et al.*, "Learning generalizable tool-use skills through trajectory generation," in *2024 IROS*, 2024.
[18] J. Grannen *et al.*, "Learning bimanual scooping policies for food acquisition," *arXiv preprint arXiv:2211.14652*, 2022.
[19] R. K. Jenamani *et al.*, "Flair: Feeding via long-horizon acquisition of realistic dishes," *arXiv preprint arXiv:2407.07561*, 2024.
[20] P. Sundaresan *et al.*, "Learning sequential acquisition policies for robot-assisted feeding," *arXiv preprint arXiv:2309.05197*, 2023.
[21] G. Narasimhan *et al.*, "Self-supervised transparent liquid segmentation for robotic pouring," in *2022 ICRA*, 2022.
[22] C. Schenck and D. Fox, "Visual closed-loop control for pouring liquids," in *2017 ICRA*, 2017.
[23] H. Lin *et al.*, "Pourit!: Weakly-supervised liquid perception from a single image for visual closed-loop robotic pouring," in *ICCV*, 2023.
[24] C. Schenck and D. Fox, "Spnets: Differentiable fluid dynamics for deep neural networks," in *CoRL*, 2018.
[25] Y.-L. Tai *et al.*, "Scone: A food scooping robot learning framework with active perception," in *CoRL*, 2023.
[26] T. Osa *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, 2018.
[27] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, 1988.
[28] B. D. Ziebart *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, 2008.
[29] T. Z. Zhao *et al.*, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.
[30] J. Ho *et al.*, "Denoising diffusion probabilistic models," *NeurIPS*, 2020.
[31] D. Seita *et al.*, "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor," in *2020 IROS*, 2020.
[32] J. Lyu *et al.*, "Scissorbot: Learning generalizable scissor skill for paper cutting via simulation, imitation, and sim2real," *arXiv preprint arXiv:2409.13966*, 2024.
[33] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.
[34] E. Todorov *et al.*, "Mujoco: A physics engine for model-based control," in *2012 IROS*, 2012.
[35] V. Makoviychuk *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *ArXiv*, vol. abs/2108.10470, 2021.
[36] X. Lin *et al.*, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *CoRL*, 2020.
[37] Z. Xian *et al.*, "Fluidlab: A differentiable environment for benchmarking complex fluid manipulation," *ArXiv*, vol. abs/2303.02346, 2023.
[38] S. Chen *et al.*, "Daxbench: Benchmarking deformable object manipulation with differentiable physics," *arXiv preprint arXiv:2210.13066*, 2022.
[39] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.
[40] I. Akkaya *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.
[41] N. Jakobi *et al.*, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *European conference on artificial life*, 1995.
[42] Y. Xiang *et al.*, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *ArXiv*, vol. abs/1711.00199, 2017.
[43] S. Liu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *ECCV*, 2024.
[44] M. Oquab *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.
[45] C. R. Qi *et al.*, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *NeurIPS*, 2017.
[46] S. Dasari *et al.*, "Learning dexterous manipulation from exemplar object trajectories and pre-grasps," in *2023 ICRA*. IEEE, 2023.
[47] B. Calli *et al.*, "Yale-cmu-berkeley dataset for robotic manipulation research," *IJRR*, 2017.
[48] K. Black *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.
[49] K. Hsu *et al.*, "Vision-based manipulators need to also see from their hands." in *ICLR*, 2022.