Local Policies Enable Zero-shot Long-horizon Manipulation

Murtaza Dalal*[1] Min Liu*[1] Walter Talbott[2] Chen Chen[2]
Deepak Pathak[1] Jian Zhang[2] Ruslan Salakhutdinov[1]
[1]Carnegie Mellon University, [2]Apple

Fig. 1: **Zero-shot Long-horizon Manipulation** Our approach trains a library of generalist manipulation skills in simulation and transfers them zero-shot to long-horizon manipulation tasks. We show a single, text-conditioned agent can manipulate unseen objects, in arbitrary poses and scene configurations, across long-horizons in the real world, solving challenging manipulation tasks with complex obstacles.

***Abstract*— Sim2real for robotic manipulation is difficult due to the challenges of simulating complex contacts and generating realistic task distributions. To tackle the latter problem, we introduce ManipGen, which leverages a new class of policies for sim2real transfer: local policies. Locality enables a variety of appealing properties including invariances to absolute robot and object pose, skill ordering, and global scene configuration. We combine these policies with foundation models for vision, language and motion planning and demonstrate SOTA zero-shot performance of our method to Robosuite benchmark tasks in simulation (97%). We transfer our local policies from simulation to reality and observe they can solve unseen long-horizon manipulation tasks with up to 8 stages with significant pose, object and scene configuration variation. ManipGen outperforms SOTA approaches such as SayCan, OpenVLA, LLMTrajGen and VoxPoser across 50 real-world manipulation tasks by 36%, 76%, 62% and 60% respectively. Video results at mihdalal.github.io/manipgen**

I. Introduction

How can we develop generalist robot systems that plan, reason, and interact with the world like humans? Tasks that humans solve during their daily lives, such as those shown in Figure 1, are incredibly challenging for existing robotics approaches. Cleaning the table, organizing the shelf, putting items away inside drawers, etc. are complex, long-horizon problems that require the robot to act capably and consistently over an extended period of time. Furthermore, such a generalist robot should be able to do so without requiring task-specific engineering effort or demonstrations. Although large-scale data-driven learning has produced generalists for vision and language [1], such models don't yet exist in robotics due to the challenges of scaling data collection. It often takes significant manual labor cost and years of effort to just collect datasets on the order of 100K-1M trajectories [2]–[5]. Consequently, generalization is limited, often to within centimeters of an object's pose for complex tasks [6], [7].

Instead, we seek to use a large-scale approach via simulation-to-reality (sim2real) transfer, a cost-effective technique for generating vast datasets that has enabled training generalist policies for locomotion which can traverse complex, unstructured terrain [8]–[13]. While sim2real transfer has shown success in industrial manipulation tasks [14]–[16], including with high-dimensional hands [17]–[20], these efforts often involve training and testing on the same task in simulation. Can we extend sim2real to open-world manipulation, where robots need to solve any task from text instruction? The core bottlenecks are: 1) accurately simulating contact dynamics [21] – for which strategies such as domain randomization [17], [22], SDF contacts [14], [15], [23], and real world corrections [16] have shown promise,

*equal contribution.

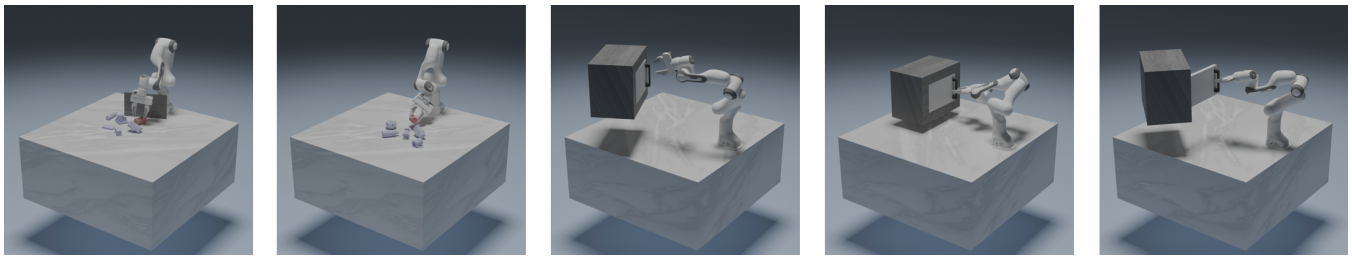


Fig. 2: **Training Environments** We train local policies (left to right) on picking, placing, handle grasping, opening and closing.

2) generating all possible scene and task configurations to ensure trained policies generalize and 3) acquiring long-horizon behaviors themselves, which may require potentially intractable amounts of data for as the horizon grows.

To address points 2) and 3), our solution is to note that for many manipulation tasks of interest, the skill can be simplified to two steps: achieving a pose near a target object, then performing manipulation. The key idea is that of *locality of interaction*. Policies that observe and act in a region local to the target object of interest are by construction:

• **absolute pose invariant**: they reason about a much smaller set of relative poses between the objects and the robot.
• **skill order invariant**: transition from the termination set of one policy and initiation set of the next via motion planning.
• **scene configuration invariant**: they solely observe the local region around the point of interaction.

We propose a novel approach that leverages the strong generalization capabilities of existing foundation models such as Visual Language Models (VLMs) for decomposing tasks into sub-problems [1], processing and understanding scenes [24] and planning collision-avoidant motions [25]. Specifically, given a text prompt, our approach outputs a plan to solve the task (using a VLM), estimates where to go and moves the robot accordingly (using motion planning) and deploys local policies to perform interaction. As a result, a simple scene generation approach (Fig. 2) can produce strong transfer results across many manipulation tasks (Fig. 1).

Our contribution is an approach to training agents at scale solely in simulation that are capable of solving a vast set of long-horizon manipulation tasks in the real world *zero-shot*. Our method generalizes to unseen objects, poses, receptacles and skill order configurations. To do so, our method, ManipGen, 1) introduces a novel policy class for sim2real transfer 2) proposes techniques for training policies at scale in simulation 3) and deploys policies via integration with VLMs and motion planners. We perform a thorough, real world evaluation of ManipGen on **50** long-horizon manipulation tasks in **five** environments with up to **8** stages, achieving a success rate of **76%**, outperforming SayCan, OpenVLA, LLMTrajGen and VoxPoser by **36%**, **76%**, **62%** and **60%**.

II. Related Work

Long-horizon Robotic Manipulation Sense-Plan-Act (SPA) has been explored extensively over the past 50 years [26]–[31]. Traditionally, SPA assumes access to accurate state estimation, a well-defined model of the environment and low-level control primitives. SPA, while capable of generalizing to a broad set of tasks, can require manual engineering and systems effort to set up [32], struggles with contact-rich interactions [33], [34] and fails due to state-estimation errors [35]. By contrast, our method can be deployed to new tasks using generalist models which have minimal setup cost, train polices for contact-rich interactions and handle state-estimation issues by training with significant local randomization.

Zero/Few-shot Manipulation Using Foundation Models The robotics community has begun to investigate VLM's capabilities for controlling robots in a zero/few-shot manner [36]–[44]. Work such as SayCan [36] and TidyBot [39] are similar to our own. They behavior clone / design a library of skills and use LLMs to perform task planning over the set of skills. Our work focuses primarily on designing the structure of skills for low-level control, decomposing them into motion planning and sim2real local policies. On the other hand, works such as LLMTrajGen [45] and CoPa [46] directly prompt VLMs to output sequences of end-effector poses, but are limited to short horizon tasks. Finally, PSL [44] and Boss [42] use LLMs to accelerate the RL training process for long-horizon tasks, yet must train on the test task, unlike our method which can solve a wide array of manipulation tasks zero-shot.

Sim2real approaches in robotics Transferring RL policies trained with procedural scene generation enables generalist robot locomotion [8]–[12]. However, these policies are typically limited to a single skill, such as walking, or slight variations like different velocities or headings. Sim2real transfer has also been explored for dexterous manipulation skills [17]–[19], [22], [47] and contact-rich manipulation [14]–[16]. In our work, we train a variety of skills for manipulation and demonstrate zero-shot generalization on numerous unseen tasks. We outperform methods that use end-to-end sim2real transfer [48] as well as real world corrections [16]. ManipGen is orthogonal to human correction approaches, and can benefit from real-world data as well.

III. Methods

To build agents capable of generalizing to a wide class of long-horizon robotic manipulation tasks, we propose a novel approach (ManipGen) that hierarchically decomposes manipulation tasks, takes advantage of the generalization capabilities of foundation models for vision and language and uses large-scale learning with our proposed policy class to learn manipulation skills. We begin by describing our framework (Fig. 3) and formulate local policies. We then discuss how to train local policies for sim2real transfer. Finally, we outline deployment: integrating VLMs, Motion Planning and sim2real policy learning to foster broad generalization.

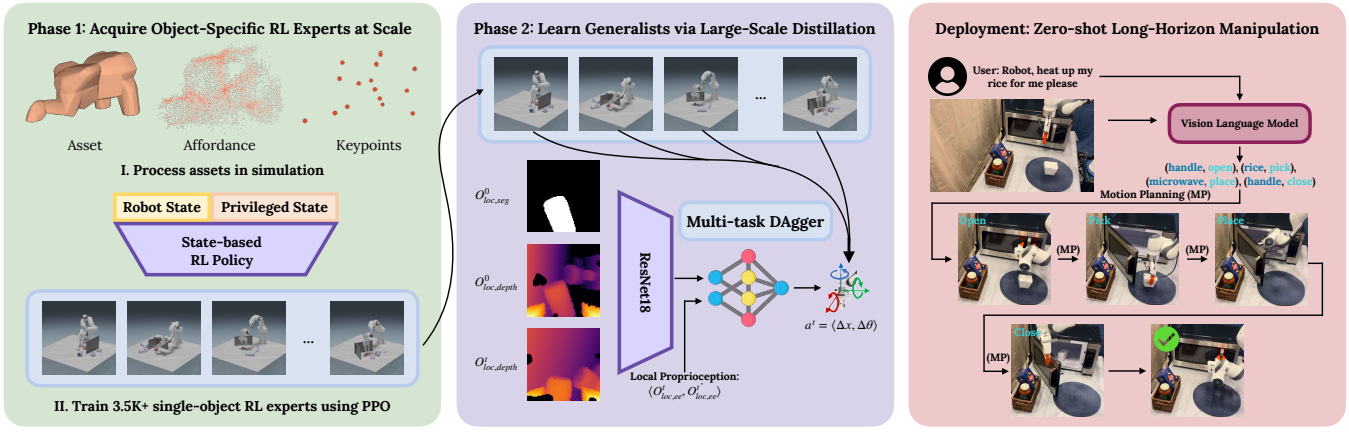


Fig. 3: **ManipGen Method Overview** (*left*) Train 1000s of RL experts in simulation using PPO (*middle*) Distill single-task RL experts into generalist visuomotor policies via DAgger (*right*) Text-conditioned long-horizon manipulation via task decomposition (VLM), pose estimation and goal reaching (Motion Planning) and sim2real transfer of local policies

A. Framework

We can decompose any task the robot needs to complete into a problem of learning a set of temporally abstracted actions (skills) as well as a policy over those skills [49]. Given a language goal g, and observation O, we can select our high-level policy, $p_\theta(g_k|g, O)$ to be a pre-trained VLM, where g_k is the k'th language subgoal. The choice of skill will be extracted from g_k below. State-of-the-art VLMs have been shown to be capable of decomposing robotics tasks into high-level language subgoals [36]–[39] because they are trained using a vast corpus of internet-scale data and have captured powerful, visually grounded semantic priors for what various real world tasks look like.

Any policy class can be used to define the skills, denoted as $p_{\phi_k}(a^t|g_k, O^t)$, which take in the kth sub-goal g_k and current observation O^t. However, note that many manipulation skills (*e.g.* picking, pushing, turning, etc.) can be decomposed into a policy π_{reach} to achieve target poses near objects, $X_{targ,k}$, followed by policy π_{loc} for contact-rich interaction. Accordingly, $p_{\phi_k}(a^t|g_k, O^t) = \pi_{reach}(\tau_{reach}|g_k, O^t)\pi_{loc}(a^t_{loc}|O^t_{loc})$. To implement π_{reach}, we need to interpret language sub-goals g_k to take the robot from its current configuration $q_{k,i}$ to some target configuration $q_{k,f}$ such that X_{ee} (the end-effector pose) is close to $X_{targ,k}$. We structure the VLM's sub-goal predictions g_k as (object, skill) tuples and interpret these plans into robot poses by pairing any language conditioned pose estimator or affordance model (to predict $X_{targ,k}$) with an inverse kinematics routine (to compute $q_{k,f}$). Motion planning is used to implement π_{reach} by predicting collision-avoidant trajectories τ_{reach} to achieve the target configuration $q_{k,f}$.

Finally, we instantiate local policies (π_{loc}) to be invariant to robot poses as well as object poses, order of skill execution and scene configurations with: 1) initialization region s_{init} near a target region/object of interest which has pose $X_{targ,k}$, 2) local observations O^t_{loc}, independent of the absolute configuration of the robot and scene and only observing the environment around the interaction region and 3) actions a^t_{loc}relative to the local observations. Overall:

$$\pi_{loc}(a^t_{loc}|O^t_{loc}), s_{init} = \{s \mid ||X_{ee} - X_{targ,k}||^2 < \epsilon\}$$

B. Training Local Policies for Sim2Real Manipulation

To train local policies, we adapt the standard two-phase training approach [11], [12], [16], [19], [47], [50] in which we first train state-based expert policies using RL, then distill them into visuomotor policies for transfer. While local policies can generalize across scene arrangements, robot configurations, and object poses, broad object-level generalization requires diverse training data. To achieve this, we train a vast array of *single-object* state-based RL policies and distill them into *generalist* visuomotor policies per skill.

While such local policies can cover a broad set of manipulation skills (pick and place, articulated/deformable object manipulation, assembly, etc.), in this work, we focus on training the following skills π_{loc}: **pick**, **place**, **grasp handle**, **open** and **close** (Fig. 2) as a minimal skill library to demonstrate generalist manipulation capabilities for a specific class of tasks. **Pick** grasps any free rigid objects. **Place** sets the object down near the initial pose. **Grasp Handle** grasps the handle of any door or drawer. **Open and Close** pull or push doors and drawers to open or close them.

To train robust local policies via RL, they require a diverse set of training environments, carefully designed observations and action spaces and well-defined reward functions enabling them to acquire behaviors in a manner that will transfer to the real world. We describe how to in this section.

Data Generation We need to first specify a set of objects to manipulate, an environment, and an initial local state distribution. For pick/place, we train on 3.5K objects from UnidexGrasp [51], randomly spawned on a table top. To ensure local policies can learn obstacle avoidance and constrained manipulation, we spawn clutter objects and obstacles in the scene. We sample initial poses in a half-sphere, with the gripper pointing toward the object (for picking) and near the placement location (for placing). For local articulated object manipulation, the region of interaction only contains the handle (2.6K objects of Partnet [52]) and door/drawer surface (designed as cuboids). We randomize the size, shape, position, orientation, joint range, friction and damping coefficients, covering a wide set of real world articulated objects. We sample initial poses in a half-sphere around the handle

(for grasp handle) and a randomly sampled initial joint pose (open/close). Finally we collect valid pre-grasp poses (antipodal sampling [53]) for picking and grasping handles and rest poses (from UnidexGrasp) for learning placing.

Observations We use a single observation space for all RL experts, accelerating learning by incorporating significant amounts of privileged information. Blind local policies can struggle to learn to manipulate objects with complex geometries as it is often necessary to have some notion of object shape to know how to manipulate. Thus, we propose to use a low-dimensional representation of the object shape by performing Farthest Point Sampling (FPS) on the object mesh with a small set number of desired keypoints K (16). Furthermore, to ease the burden of credit assignment and thereby accelerate learning, we incorporate the individual reward components $\{\mathbf{r}\}$ and an indicator for the final observation $\mathbb{1}\{t = T\}$. RL observations are $O^t = \langle X^t_{ee}, \dot{X}^t_{ee}, X^t_{obj}, \{FPS^t{}_{obj}\}^K_{k=1}, \{\mathbf{r}\}^t, \mathbb{1}\{t = T\}\rangle$

Actions We use the action space from Industreal [14] which has been shown to successfully transfer manipulation policies from sim2real for precise assembly tasks. Our policies predict delta pose targets for a Task Space Impedance (TSI) controller, where $a = [\Delta x; \Delta\theta]$, where Δx is a position error and $\Delta\theta$ is a axis-angle orientation error.

Rewards We train RL policies (π_{loc_k}) in simulation using reward functions we design to elicit the desired behavior per skill k. We propose a reward framework that encompasses our local skills: $\mathbf{r} = c_1 r_{ee} + c_2 r_{obj} + c_3 r_{ee,obj} + c_4 r_{action} + c_5 r_{succ}$. $\mathbf{r}$ specifies behavior for a broad range of manipulation tasks which involve moving the end-effector to specific poses (often right before contact) as well as a target object to desired poses and need to do so while maintaining certain constraints on the relative motion between the end-effector and the object as well as pruning out undesirable actions. r_{ee} encourages reaching/maintaining specific end-effector poses, r_{obj} restricts/encourages specific object poses or joint configurations, $r_{ee,obj}$ constrains the end-effector motion relative to the object(s) in the scene, r_{action} restricts or penalizes undesirable actions and r_{succ} is a binary success reward.

C. Generalist Policies via Distillation

In order to convert single-object, privileged policies into real world deployable skills, we distill them into multi-object, generalist visuomotor policies using DAgger [54].

Multitask Online Imitation Learning Empirically the standard, off-policy version of DAgger with interleaved behavior cloning (to convergence) and large dataset collection does not perform well. The policy ends up modeling data from policies whose state visitation distributions deviate significantly from the current policy. On the other hand, on-policy variants of DAgger, which take a single gradient step per environment step [10], [19], [47], [50], can produce unstable results in the multi-task regime since the policy only gets data from a single object in a batch. We introduce a simple variant of DAgger which smoothly trades off between the two extremes by incorporating a replay buffer of size K that holds the last $K * B$ trajectories in memory. Training alternates between updating the agent for a single epoch on this buffer and collecting a batched set of trajectories (size B) from the environment for the current object.

Observation Space Design for Locality For local policies to transfer effectively to the real robot, the observation space and augmentations must be designed with transfer in mind. To imitate a privileged expert, our observation space must be expressive – providing as much information as possible to the agent. The observations must also be local to enable all of the properties of locality, and augmentations must ensure the policy is robust to noisy real world vision.

Local observations use wrist camera depth maps. Depth maps transfer well from sim2real for locomotion [10]–[12], [50], and wrist views are inherently local and improve manipulation performance [55]–[57]. To further enforce locality, we clamp depth values to a max depth of $30cm$ and then normalize the values to between 0 and 1. Since local wrist-views often get extremely close to the object during execution, it can become difficult for the agent to understand the overall object shape. Thus, we include the initial local observation $O^0_{loc,depth}$ at every step with a segmentation mask of the target object ($O^0_{loc,seg}$) so that the local policy is aware of which object to manipulate. We transform absolute proprioception into local by computing observations relative to the first time-step ($O_{loc,ee} = [X^0_{ee,t} - X^0_{ee}]$) and incorporate velocity information ($O_{\dot{loc},ee,t}$), which improves transfer. Our observation space is $\mathbf{O^t_{loc}} = \langle O^t_{loc,depth}, O^0_{loc,seg}, O^0_{loc,depth}, O^t_{loc,ee}, O^{\dot{t}}_{loc,ee}\rangle$.

Augmentations To enable robustness to noisy real world observations, namely edge artifacts and irregular holes, we augment the clean depth maps we obtain in simulation. For edge artifacts, in which we observe dropped pixels and noisiness along edges, we use the correlated depth noise via bi-linear interpolation of shifted depth from [58] which tends to model this effect well. We also observe that real world depth maps tend to have randomly placed irregular holes (pixels with depth 0). As a result, we compute random pixel-level masks and Gaussian blur them to obtain irregularly shaped masks that we then apply to the depth image. We also use random camera cropping augmentations which has been shown to improve visuomotor learning performance [57].

D. Zero-shot Text Conditioned Manipulation

Given our framework and trained local policies, how do we now deploy them in the real-world, to solve a wide array of manipulation tasks in a zero shot manner?

To enable our system to solve long-horizon tasks, $p_\theta(g_k|g, O)$, decomposes the task into a skill chain to execute given task prompt g. We implement p_θ as GPT-4o, a SOTA VLM. Given the task prompt g, descriptions of the pre-trained local skills and how they operate, and images of the scene O, we prompt GPT-4o to give a plan for the task structured as a list of (object, skill) tuples. For example, for the task shown in Fig. 3, GPT outputs ((handle, open), (rice pick), (microwave, place), (handle, close)). We then need a language conditioned pose estimator (to compute $X_{targ,k}$) that generalizes broadly; we opt to use Grounded SAM [24] due to its strong open-set segmentation capabilities. To estimate

$X_{targ,k}$, we can segment the object pointcloud, average it to get a position and use its surface normals to select a collision-free orientation. One issue is that Grounding Dino [59], used in Grounded SAM, is very sensitive to the prompt. As a result, we pass its predictions back into GPT-4o to adjust the object prompts to capture the correct object.

For predicting τ_{reach}, while any motion planner can be used, we select Neural MP [25] for its fast (2s) and strong real-world planning. Given $X_{targ,k}$, we compute target joint state $q_{k,f}$, plan with Neural MP open-loop and execute the predicted τ_{reach} on the robot using a PID joint controller. We then execute the appropriate local policy (as predicted by the VLM) on the robot to perform manipulation. We alternate between motion planning and deploying local policies until the task is complete. Finally, we note that the particular choice of models is orthogonal to our method.

IV. Experimental Setup and Results

We pose the following experimental questions that guide our evaluation: 1) Can an autonomous agent control a robot to perform a wide array of *long-horizon* manipulation tasks zero-shot? 2) How does our approach compare to methods that learn from online interaction? 3) For direct sim2real transfer, how do Local Policies compare against end-to-end learning and other transfer techniques that leverage human correction data? 4) To what degree do the design decisions made in ManipGen affect the performance of the method?

A. Training and Deployment Details

Architecture and Training We train all RL policies at scale using PPO [60] in GPU-parallelized simulation [61]. We train for 500 epochs, with an environment batch size of 8192 and max episode length of 120 steps per skill. To learn visuomotor policies to perform high-frequency (60 Hz) end-effector control, we pair Resnet-18 [62] and Spatial Softmax [63] with a two layer MLP decoder (4096 hidden units). Finally, for training, minimizing Mean Squared Error loss is sufficient for learning multitask policies via DAgger. In early experiments, we found that our architecture performs comparably to using LSTMs [64], Transformers [65], and ACT [6] and is faster to train (5-10x) and deploy (2x).

Hardware Setup We use the Franka Panda robot arm with the UMI [66] gripper fingertips and a wrist-mounted Intel Realsense d405 camera for obtaining local observations (84x84 resolution). Note for local observations, using depth sensing that is accurate at short range (such as the d405) is crucial to obtaining high quality local depth maps. We perform hole-filling and smoothing to clean the depth maps. Following Transic [16], we do not model the compliance of the UMI gripper in simulation, but instead transfer policies trained with rigid fingertips to the real world, which performs well in practice. For real world control, we use a TSI end-effector controller at 60 Hz with (Leaky) Policy Level Action Integration (PLAI) [14]. We use Leaky PLAI with .001 position action scale, .05 rotation action scale for pick and .005 rotation action scale for all other skills.

	Bread	Can	Milk	Cereal	CanBread	CerealMilk	Average
Stages	*2*	*2*	*2*	*2*	*4*	*4*	
Online Learning:							
DRQ-v2	52%	32%	2%	0%	0%	0%	14%
RAPS	0%	0%	0%	0%	0%	0%	0%
PSL	100%	100%	100%	100%	90%	85%	96%
Zero-Shot:							
TAMP	90%	100%	85%	100%	72%	71%	86%
SayCan	93%	100%	90%	63%	63%	73%	80%
Ours	100%	100%	99%	97%	97%	91%	**97%**

TABLE I: **Robosuite Benchmark Results.** ManipGen zero-shot transfers to Robosuite, outperforming end-to-end and hierarchical RL methods as well as traditional and LLM planning methods.

Finally, we use 4 calibrated Intel Realsense d455 cameras for global view observations (640x480).

B. Simulation Comparisons and Analysis

Robosuite Benchmark Results We first evaluate against the long-horizon manipulation tasks used in PSL [44] from the Robosuite benchmark [67] in simulation which has a set of challenging long-horizon manipulation tasks (**PickPlace{Bread, Milk, Cereal, CanBread, CerealMilk}**). We compare to end-to-end RL methods [68], hierarchical RL [44], [69], task and motion planning [70] and LLM planning [36]. In these experiments, we *zero-shot* transfer our trained policies to Robosuite and evaluate their performance against methods that use task specific data (Tab. I). ManipGen outperforms or matches PSL, the SOTA method on these tasks, across the board, achieving an average success rate of 97.33% compared to 95.83%. These results demonstrate that ManipGen can outperform methods that are trained on the task of interest [44], [68], [69] as well as planning methods that have access to privileged state info [36], [70].

ManipGen Analysis and Ablations. We study design decisions proposed in our method by training single object pick policies on 5 objects (remote, can, bowl, bottle, camera) and testing on held out poses. We begin with our observation space design choices: ManipGen achieves 97.44% success rate in comparison to (94.33%, 96.64%, 97.25%) for removing key-point observations, success observation and reward observations respectively. Incorporating key-point observations is the most impactful change, enabling the agent to perceive the shape of the target object. Next, we evaluate how the level of locality (the size of the region around the target object that we initialize over) affects learning performance. At convergence, we find that ManipGen (8cm max distance from target) achieves 97.44% success rate while performance diminishes with increasing distance (95.65%, 89.55%, 72.52%) for 16cm, 32cm and 64cm respectively.

For DAgger, we analyze our observation design choices and find that including velocity information, the first observation, and changing proprioception to be relative to the first frame are crucial to the success of our method. While ManipGen gets 94.3% success, removing velocity info and using absolute proprioception hurt significantly (89.92% and 90.94%) while removing the first observation drops performance to 93.13%. We also vary the DAgger buffer size, from 1 (on-policy), 10, 100, and 1000 (off-policy) for multitask training (with 3.5K objects, not 5). We find that 100 performs best, achieving 85% in simulation averaged

Tasks	Ours	Transic	Direct Transfer	DR. & Data Aug. [48]	HG-Dagger [75]	IWR [76]	BC [72]
Stabilize	95%	**100%**	10%	35%	65%	65%	40%
Reach and Grasp	**95%**	**95%**	35%	60%	30%	40%	25%
Insert	**80%**	45%	0%	15%	35%	40%	10%
Avg	**90%**	80%	15%	36.7%	43.3%	48.3%	25%

TABLE II: **Transic Benchmark Results** ManipGen achieves SOTA results on the Transic [16] benchmark in terms of task success rate without using any real world data, outperforming direct transfer, imitation learning and human-in-the-loop methods.

across 100 held out objects, out performing (78%, 82% and 75%) for 1, 10 and 1000 respectively.

C. Real World Evaluation

FurnitureBench Results To evaluate the sim2real capabilities of local policies (Tab. II), we deploy ManipGen on FurnitureBench [71], comparing against a wide array of direct-transfer [48], imitation learning [72], [73], offline RL [74] and human-in-the-loop methods [16], [75], [76] from Transic [16]. These tasks are single stage; we train local policies to perform pushing (**Stabilize**), picking (**Reach and Grasp**) and insertion (**Insert**). We predict a start pose to initialize the local policy from and deploy the simulation-trained policies. ManipGen matches or outperforms end-to-end direct transfer methods (75%, 53.3%), imitation methods (55%, 82.7%, 65%, 75%, 86.7%) and sim2real methods that leverage additional correction data [16]. For Insert, local policies are able to outperform Transic without using any real world data, achieving 80% while Transic achieves 45%. These experiments demonstrate ManipGen improves over end-to-end learning and is capable of handling challenging initial states, contact-rich interaction and precise motions.

Zero-shot Long-horizon Manipulation To test the generalization capabilities of our method, we propose 5 diverse long-horizon manipulation tasks (Fig. 1) which involve pick and place, obstacle avoidance and articulated object manipulation. **Cook**: put food into a pot on a stove (2 stages), **Replace**: take a pantry item out of the shelf, put it on a tray and take an object from the tray and put it in the shelf (4 stages), **CabinetStore**: open a drawer in the cabinet, put an object inside and close it (4 stages). **DrawerStore**: open a drawer, put two personal care items inside and close the drawer (6 stages) and **Tidy**: clean up the table by putting all the toys into a bin (8 stages). Each task has a unique object set (5 objects), receptacle (pot, shelf, etc.) and text description. We run 10 evaluations per task, randomizing which objects are present and their poses, receptacle poses, and target poses. All poses are randomized over the table and we select a diverse set of evaluation objects.

Evaluation Criteria For each task we identify the stages required for completion. A trial is considered successful if the final state meets the task's goal as specified. Additionally, we track the number of stages completed in each trial. We conduct 10 trials per task, reporting the success rate and average number of stages completed.

Comparisons We evaluate SOTA text-conditioned manipulation approaches: SayCan [36], LLMTrajGen [45] and VoxPoser [41]. For SayCan, we use our VLM and motion planning system with engineered interaction primitives to assess the role of training local policies. We additionally compare against a pre-trained manipulation model OpenVLA [77], fine-tuning it per task using 25 demonstrations on held-out objects, poses, and scene configurations. Given a text prompt, we record task success rates and stages completed.

	Cook	Replace	CabinetStore	DrawerStore	Tidy	Avg
Stages	*2*	*4*	*4*	*6*	*8*	*4.8*
OpenVLA	0% (0.1)	0 (0.0)	0% (0.0)	0 (0.0)	0 (0.0)	0% (.02)
SayCan	80% (1.7)	10% (1.3)	70% (3.5)	20% (3.6)	20% (4.8)	40% (3.0)
LLMTrajGen	70% (1.5)	0% (0.6)	0% (0.6)	0% (1.0)	0% (2.6)	14% (1.3)
VoxPoser	70% (1.4)	0% (0.8)	0% (0.8)	0% (0.9)	10% (4.4)	16% (1.7)
Ours	**90% (1.9)**	**80% (3.7)**	**90% (3.9)**	**60% (4.7)**	**60% (7.2)**	**76% (4.3)**

TABLE III: **Zero-shot Long Horizon Manipulation** We report task success rate and average number of stages completed per real world task. ManipGen outperforms all methods on each task, achieving 76% with 4.28/4.8 stages completed on average.

Across all 5 tasks (Tab. III), ManipGen achieves 76% **zero-shot success rate**, outperforming all methods. Notably, our local policies are not trained on *any of these specific objects* or in *these specific configurations* and require *no real-world adaptation*. ManipGen is able to avoid obstacles while performing manipulation of unseen objects in arbitrary poses and configurations. Failure cases stemmed from 1) vision failures as open-set detection models such as Grounding Dino [59] detected the wrong object, 2) imperfect motion planning, resulting in collisions with the environment during execution which dropped objects sometimes and 3) local policies failing to manipulate from sub-optimal initial poses. In general, DrawerStore and Tidy are the most challenging tasks due to their horizon, and consequently all methods, including our own perform worse (60% for ours, 20% for best baseline).

SayCan is the strongest baseline (40% success), achieving non-zero success on every task by leveraging the generalization capabilities of vision-language foundation models in a structured manner. However, when initial poses are not ideal or the task requires contact-rich control, pre-defined primitives fall apart (10-20% success). LLMTrajGen excelled at top-down pick-and-place (Cook: 70%) but struggled with obstacle avoidance (Replace) and articulated object manipulation (Store) due to limited prompt coverage. VoxPoser matches LLMTrajGen in Cook but failed at precise rotations and tight-space tasks (Replace, Store) and frequently generated incorrect plans for long-horizon tasks (Tidy). Finally, OpenVLA failed to solve any task, unable to generalize to held-out objects and poses even with few-shot data. We attempted to evaluate it on its training objects and it still performs poorly with strong pose randomization.

V. Discussion

We present ManipGen, a method for long-horizon manipulation tasks with unseen objects and configurations via generalist policies for sim2real transfer. We propose local policies, a novel policy class invariant to pose, skill order, and scene configuration, enabling broad generalization. For deployment, we leverage foundation models for vision, language and motion planning to solve long-horizon manipulation tasks from text prompts. Across 50 real-world long-horizon manipulation tasks, ManipGen achieves 76% *zero-shot* success, surpassing SOTA planning and imitation methods.

Acknowledgement

We thank Russell Mendonca, Ananye Agarwal and Yash Narang for their insightful discussions and feedback. We additionally thank Russell Mendonca, Shikhar Bahl, Lili Chen, and Unnat Jain for feedback on early drafts of this paper. This work was supported in part by the NSF Graduate Fellowship, Apple, and ONR grant N00014-23-1-2368.

References

[1] R. OpenAI, "Gpt-4 technical report," *arXiv*, pp. 2303–08 774, 2023. 1, 2

[2] O.-X. E. Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023. 1

[3] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn, "Droid: A large-scale in-the-wild robot manipulation dataset," 2024. 1

[4] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, "Bridge data: Boosting generalization of robotic skills with cross-domain datasets," *arXiv preprint arXiv:2109.13396*, 2021. 1

[5] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning (CoRL)*, 2023. 1

[6] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023. 1, 5

[7] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv: Arxiv-2401.02117*, 2024. 1

[8] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020. 1, 2

[9] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021. 1, 2

[10] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning*. PMLR, 2023, pp. 403–415. 1, 2, 4

[11] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," *arXiv preprint arXiv:2309.05665*, 2023. 1, 2, 3, 4

[12] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023. 1, 2, 3, 4

[13] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024. 1

[14] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. S. Narang, "Industreal: Transferring contact-rich assembly tasks from simulation to reality," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, Eds., 2023. [Online]. Available: https://doi.org/10.15607/RSS.2023.XIX.039 1, 2, 4, 5

[15] B. Tang, I. Akinola, J. Xu, B. Wen, A. Handa, K. Van Wyk, D. Fox, G. S. Sukhatme, F. Ramos, and Y. Narang, "Automate: Specialist and generalist assembly policies over diverse geometries," *arXiv preprint arXiv:2407.08028*, 2024. 1, 2

[16] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, "Transic: Sim-to-real policy transfer by learning from online correction," *arXiv preprint arXiv: Arxiv-2405.10315*, 2024. 1, 2, 3, 5, 6, 12

[17] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019. 1, 2

[18] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," *arXiv preprint arXiv:2210.13702*, 2022. 1, 2

[19] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. Van Wyk, "Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics," *arXiv preprint arXiv:2407.02274*, 2024. 1, 2, 3, 4

[20] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," *arXiv preprint arXiv:2211.11744*, 2022. 1

[21] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033. 1

[22] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020. 1, 2

[23] Y. S. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, Á. Moravánszky, G. State, M. Lu, A. Handa, and D. Fox, "Factory: Fast contact for robotic assembly," in *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*, K. Hauser, D. A. Shell, and S. Huang, Eds., 2022. [Online]. Available: https://doi.org/10.15607/RSS.2022.XVIII.035 1

[24] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, "Grounded sam: Assembling open-world models for diverse visual tasks," 2024. 2, 4

[25] M. Dalal, J. Yang, R. Mendonca, Y. Khaky, R. Salakhutdinov, and D. Pathak, "Neural mp: A generalist neural motion planner," *arXiv preprint arXiv:2409.05864*, 2024. 2, 5, 12

[26] A. I. Center, "Shakey the robot," 1984. 2

[27] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981. 2

[28] D. E. Whitney, "The mathematics of coordinated control of prosthetic arms and manipulators," 1972. 2

[29] M. Vukobratović and V. Potkonjak, *Dynamics of manipulation robots: theory and application*. Springer, 1982. 2

[30] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg, "Real-time perception meets reactive motion generation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1864–1871, 2018. 2

[31] R. R. Murphy, *Introduction to AI robotics*. MIT press, 2019. 2

[32] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5678–5684. 2

[33] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001. 2

[34] D. E. Whitney, *Mechanical assemblies: their design, manufacture, and role in product development*. Oxford university press New York, 2004, vol. 1. 2

[35] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013. 2

[36] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv: Arxiv-2204.01691*, 2022. 2, 3, 5, 6

[37] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue:

Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022. 2, 3

[38] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147. 2, 3

[39] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv preprint arXiv:2305.05658*, 2023. 2, 3

[40] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023. 2

[41] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023. 2, 6

[42] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim, "Bootstrap your own skills: Learning to solve new tasks with large language model guidance," *Conference on Robot Learning*, 2023. 2

[43] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023. 2

[44] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov, "Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks," in *International Conference on Learning Representations (ICLR)*, 2024. 2, 5, 12

[45] T. Kwon, N. Di Palo, and E. Johns, "Language models as zero-shot trajectory generators," *IEEE Robotics and Automation Letters*, 2024. 2, 6

[46] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao, "Copa: General robotic manipulation through spatial constraints of parts with foundation models," *arXiv preprint arXiv:2403.08248*, 2024. 2

[47] A. Agarwal, S. Uppal, K. Shaw, and D. Pathak, "Dexterous functional grasping," in *7th Annual Conference on Robot Learning*, 2023. 2, 3, 4

[48] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *arXiv preprint arXiv: Arxiv-1710.06537*, 2017. 2, 6

[49] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370299000521 3

[50] S. Uppal, A. Agarwal, H. Xiong, K. Shaw, and D. Pathak, "Spin: Simultaneous perception interaction and navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 133–18 142. 3, 4

[51] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4737–4746. 3, 9

[52] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 909–918. 3, 9

[53] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 438–13 444. 4, 9

[54] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635. 4

[55] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn, "Vision-based manipulators need to also see from their hands," *arXiv preprint arXiv:2203.12677*, 2022. 4

[56] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," 2023. 4

[57] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *arXiv preprint arXiv:2108.03298*, 2021. 4

[58] J. T. Barron and J. Malik, "Intrinsic scene properties from a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 17–24. 4

[59] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023. 5, 6

[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv: Arxiv-1707.06347*, 2017. 5, 10

[61] 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: Arxiv-2108.10470*, 2021. 5

[62] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi, "Learning human-to-humanoid real-time whole-body teleoperation," *arXiv preprint arXiv: Arxiv-2403.04436*, 2024. 5

[63] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 512–519. 5

[64] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation MIT-Press*, 1997. 5

[65] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017. 5

[66] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," *arXiv preprint arXiv:2402.10329*, 2024. 5, 12

[67] 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," *arXiv preprint arXiv: Arxiv-2009.12293*, 2020. 5

[68] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering visual continuous control: Improved data-augmented reinforcement learning," *arXiv preprint arXiv:2107.09645*, 2021. 5

[69] M. Dalal, D. Pathak, and R. R. Salakhutdinov, "Accelerating robotic reinforcement learning via parameterized action primitives," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 847–21 859, 2021. 5

[70] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448. 5

[71] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, Eds., 2023. [Online]. Available: https://doi.org/10.15607/RSS.2023.XIX.041 6, 12

[72] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1. Morgan-Kaufmann, 1988. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf 6

[73] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv: Arxiv-2108.03298*, 2021. 6

[74] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv: Arxiv-2110.06169*, 2021. 6

[75] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," *arXiv preprint arXiv: Arxiv-1810.02890*, 2018. 6

[76] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *arXiv preprint arXiv: Arxiv-2012.06733*, 2020. 6

[77] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024. 6

Appendix

VI. RL Training Details

In this section, we provide a detailed description of the data generation process, the exact reward definitions and the specific hyper-parameters we use to train our skills.

A. Data Generation

For training generalist pick and place skills, we require a large dataset of common objects that can simulate well with contact. As a result, we train policies using the UnidexGrasp dataset [51] which contains 3.5K objects of 133 categories such as bowls, cups, bottles, cameras, remotes, etc. Since our policies are local, we can simply generate scenes with a single object spawned on a table top. However, such an agent may not generalize well to manipulation in tight spaces and among clutter, when it needs to perform local obstacle avoidance and constrained manipulation. For robustness, we train with randomly sampled clutter objects (UnidexGrasp) and obstacles (cuboids) that we spawn in the scene.

For picking, to define an initial state distributions which ensures locality (within ϵ of the target object), we sample poses in a half-sphere above the table with radius $\epsilon = 0.08$ and an additional error tolerance of 0.05 around the target object that are always pointing towards the object (ensuring the object is visible from the wrist camera). For placing, we execute the pick policy and then sample initial poses in a cuboid of with side length $\epsilon = 0.2$ around the picking pose. In both cases we ensure that the sampled poses are not in contact with anything in the environment (aside from the in-hand object if present).

For local articulated object manipulation of objects such as doors and drawers, the design and global structure of the asset is not important. In fact, the only component of interest to the local policy is the handle. Accordingly, we sample from a dataset of 2.6K door and drawer handles from the PartNet dataset [52] and build door and drawer assets out of cuboids (Fig. 2) as they are straightforward to randomize. We define drawers as boxes to be pulled straight out and doors as boxes to be opened using a vertical hinge joint. We randomize the size, shape, position, orientation, articulated joint range, friction and damping coefficients of the articulated objects, which covers a wide set of real world articulated objects. Detailed randomization distributions are presented in Tab. IV. For the grasp handle skill, we sample initial poses pointing toward the handle in a half sphere (in this case vertical half-sphere, away from the door) with radius $\epsilon = 0.08$ and an additional error tolerance of 0.05. For opening and closing, after sampling a random initial pose of the articulated joint, we execute the grasp handle policy and add a small noise to ensure diversity of the final end-effector pose.

Finally, we collect valid pre-grasp and rest poses in simulation to help train our local policies. Specifically, we randomly sample grasp poses on the object mesh using antipodal sampling [53] (1K per rest pose for UniDexGrasp objects and 2.5K for PartNet objects). We then move the Franka arm to pick/grasp the handle of the object using the pre-sampled grasp poses and save the successful poses. We also utilize the success rate of this scheme to filter out rest object poses that are not graspable (*e.g.*, an upside down bowl). We also generate a wide set of object rest poses for training the placing policy using the initial poses from the UnidexGrasp dataset, and augmenting them by rotating about the z-axis with 8 different angles and testing to ensure the objects remain at rest.

Parameter	Range / Distribution
UniDexGrasp objects	
Object size	$[0.06, 0.30]$
Initial object position (XY)	$\mathcal{U}(0.3, 0.7) \times \mathcal{U}(-0.2, 0.2)$
Initial object rotation (Z-axis)	$\mathcal{U}(-\pi, \pi)$
Articulated objects	
Door size	$\mathcal{U}(0.25, 0.40) \times \mathcal{U}(0.20, 0.50)$
Door damping	$\mathcal{U}(0.01, 0.02)$
Door friction	$\mathcal{U}(0.025, 0.050)$
Door joint range	$[0, \pi/2]$
Drawer size	$\mathcal{U}(0.25, 0.50) \times \mathcal{U}(0.08, 0.25)$
Drawer damping	$\mathcal{U}(0.10, 0.20)$
Drawer friction	$\mathcal{U}(0.25, 0.50)$
Drawer joint range	$[0, 0.3]$
Distance to robot base	$\mathcal{U}(0.65, 0.75)$
Object Orientation	$\mathcal{U}(-\pi/2, \pi/2)$

TABLE IV: Randomization for data generation.

B. Rewards

We provide additional details on how to specify rewards for each skill. Specifically, we define $r_{ee}, r_{obj}, r_{ee,obj}$ and r_{action}.

Pick involves moving the gripper so the object can be easily grasped. Instead of encouraging the agent to move towards the overall object pose, which is not necessarily the pose to achieve for grasping, we provide dense signal for learning to grasp using pre-sampled grasp poses ($\{X_{targ}\}$). We encourage the agent to minimize the key-point distance [**?**] to the nearest grasp pose:

$$r_{ee,grasp} = \sum_{i=1}^{N} e^{\min_{\{X_{targ}\}} \|X^i_{ee} - X^i_{targ}\|^2}$$

Another challenge is that of picking in tight spaces, in which the policy changing directions and too frequently may cause damage and failure to complete the task. Thus, we encourage the agent to minimize its change in gripper orientation while interacting, by adding a penalty term on the angle between the current and previous gripper pose along the gripper's central axis (v),

$$r_{ee,init} = -\arccos\Big((R^t_{ee} v)(R^{t-1}_{ee} v)\Big)$$

We also add a term to minimize the contact force on the gripper which discourages contact with any part of the scene.

$$r_{ee,obj} = \max(\max(f_{left}, f_{right}), 0)$$

Finally, when picking, the agent should try to minimize moving the target object: we set

$$r_{obj} = e^{\|X^t_{obj,xy} - X^0_{obj,xy}\|^2}$$

to penalize changes in object pose. All other reward components have a constant set to 0.

Place requires the agent to carefully set the object down near the initial pose. To provide dense signal for placing and ensure stability, we use a key-point distance reward on the object pose to encourage it to reach a stable absolute rest pose (r_{obj})

$$r_{obj} = e^{\sum_{i=1}^{8} \|X^i_{obj,FPS} - X^i_{rest,FPS}\|^2}$$

and a key-point distance reward to the nearest grasp pose to encourage the agent to maintain a stable pose relative to the object itself while moving (r_{ee}). This reward is the same as in pick.

Grasp Handle enables the agent to be able to grasp the handle of any door or drawer, a necessary skill to interact with articulated objects. This skill uses similar rewards to pick, but adapted for articulation: 1) key-point-based rewards for reaching pre-grasp poses (r_{ee}) 2) restricting the gripper orientation in task irrelevant directions ($r_{ee,z}$) 3) restricting the gripper orientation relative to the handle in the x-axis ($r_{ee,obj}$) 4) discouraging the agent from moving the object by minimizing the motion of the joint (r_{obj}).

Same as in pick, we wish to use pre-grasp poses in order to provide dense signal to the agent and ensure that it does not change orientation too much. We use the same key-point-based reward $r_{ee,grasp}$, but instead restrict the policy's movement in the z-direction, an axis along which motion is not beneficial to solving the task

$$r_{ee,z} = -|X^t_{ee,z} - X^0_{ee,z}|$$

We further encourage the agent to only move in task relevant directions, by encouraging the agent to minimize the angle between the gripper and handle, in the x-axis of the door frame:

$$r_{ee,obj} = e^{\min((R^t_{handle} R^t_{ee} v)_x + thresh, 0)}$$

Similar to pick, we discourage the agent from moving the object, in this case by minimizing the motion of the joint

$$r_{obj} = -|q^t - q^0|$$

Open and Close involve opening and closing articulated objects, having already grasped the handle. We can solve the task with a dense absolute difference reward (r_{obj}) between the current and target joint angles (0 for close, and q_{limit}, the joint limit of the object for open). To ensure the agent maintains its grasp while smoothly moving the articulated to joint to the desired configuration, we use $r_{ee,z}$ from Grasp handle, penalize movement relative to the door/drawer handle ($r_{ee,obj}$), and discourage (r_{action}) taking actions that cause the handle to slip out (moving in the y or z axes).

These skills utilize a dense reward for solving the task

$$r_{obj} = |q^t - q_{targ}|$$

where q_{targ} is 0 for close and is q_{limit} the joint limit of the object for open. As with grasp handle, we restrict the policy's movement along the z-axis using $r_{ee,z}$. We additionally penalize any movement of the end-effector in the handle frame,

$$r_{ee,obj} = \|R^t_{handle} X^t_{ee} - R^{t-1}_{handle} X^{t-1}_{ee}\|$$

and any sampled actions that cause the agent to move in the y or z axes:

$$r_{action} = -\|(R^t_{handle}(a))_{yz}\|^2$$

These rewards ensure that the agent maintains its grasp while smoothly moving the articulated to joint to the desired configuration.

We train all RL policies with PPO [60]. Detailed hyper-parameters are presented in Tab. V and Tab. VI.

Hyperparameter	Value
Num. envs (Isaac Gym, state-based)	8192
Num. rollout steps per policy update	32
Num. learning epochs	5
Episode length	120
Discount factor	0.99
GAE parameter	0.95
Entropy coeff.	0.0
PPO clip range	0.2
Learning rate	0.0005
KL threshould for adaptive schedule	0.16
Value loss coeff.	4.0
Max gradient norm	1.0

TABLE V: Hyper-parameters for PPO.

Skill	#Max epoch	#Save best	#Early stop
Pick	500	100	200
Place	500	100	200
Grasp Handle	500	100	100
Open	500	100	100
Close	500	100	100

TABLE VI: Training epochs and early stop criterion for each task. #Max epoch is the maximum number of iterations to run. #Save best is the first iteration to begin saving best checkpoints. # Early stop suggests terminating early if there is no improvement after certain number of iterations.

VII. Distillation Training Details

A. Multitask DAgger

In our multitask DAgger implementation, we incorporate a replay buffer of size K that holds the last $K \times B$ trajectories in memory. The training process alternates between updating the policy for a single epoch on this buffer and collecting a batched set of trajectories (size B) from the environment for the current object. In practice, we find that K=100, B=32 performs well, which means for a single object we collect 32 simultaneous trajectories at a time, and we can hold data for up to 100 objects in our buffer which is constantly refreshed as we collect new data. A practical issue is loading all objects from the dataset into simulation simultaneously is unfeasible. To address this, we split the dataset into batches of 100 objects and sequentially launch training on each batch

for 100 epochs. Detailed multitask DAgger parameters are presented in Tab. VII.

Hyperparameter	Value
Num. envs (Isaac Gym, vision)	128
Episode length	120
Num. rollout steps per policy epoch	120
Num. learning epochs	1
Buffer size	100 * 128
Learning rate	0.0001
Batch size	2048

TABLE VII: Hyper-parameters for Multitask DAgger.

B. Data Augmentation

In addition to random camera cropping, we also apply *edge noise* and *random holes* to enhance robustness to real world observations.

Edge artifacts To model the noisiness along object edges, we use the correlated depth noise via bi-linear interpolation of shifted depth. Given a depth map of size $H \times W$, we construct a grid $\{0,\cdots,H-1\}\times\{0,\cdots,W-1\}$. For each node on the grid, we apply a random shift $\mathcal{N}(0,0.5)$ with probability 0.8. We then perform bilinear interpolation between the original depth values and the adjusted grid to generate a new depth map.

Random holes We observe that, even after hole filling, the real world depth maps still contain irregular holes (especially for reflective surfaces and dim environments). To model these holes, we create a random pixel-level mask from $\mathcal{U}(0,1)$. This mask is smoothed with Gaussian blur and normalized to the range $[0,1]$. Based on the mask, we zero out pixels with mask values exceeding a threshold randomly sampled from $\mathcal{U}(0.6,0.9)$. The randomization is applied to a depth map with probability 0.5.

We summarize hyper-parameters for DAgger data augmentation in Tab. VIII and visualize them in Fig. 4. Note that the resolution of our depth map is 84×84. The visual effect with these hyper-parameters will change on different resolutions.

Hyper-parameter	Value
Edge Noise	
Gaussian Noise Std	0.5
Noise Accept Prob	0.8
Random Holes	
Gaussian Blur Kernel Size	[3, 27]
Gaussian Blur Std	[1, 7]
Mask Threshold	[0.7, 0.9]
Hole Keep Prob	0.5

TABLE VIII: Hyper-parameters for DAgger data augmentation.

VIII. Deployment Details

In this section, we describe our real-world deployment system in detail. We begin by providing a high-level overview of ManipGen deployment in pseudocode below:

Tagging One challenge with Grounded SAM is prompting: the Grounding Dino module is quite sensitive to the input prompt and does not always detect the correct object unless the prompt is formatted well (descriptive, includes colors and texture and shape). Since the VLM will be used to prompt SAM with the target object for each skill, we need the VLM to output tags for objects such that Grounded SAM will trigger on the correct object. As a result, we include an initial tagging phase, in which we have the VLM list all objects that are present in the scene, which we then pass into Grounded SAM to get a list of tags and associated segmentation masks based on how Grounded SAM interprets the scene. We then pass the tagged image, as well as the original image back into the VLM for planning

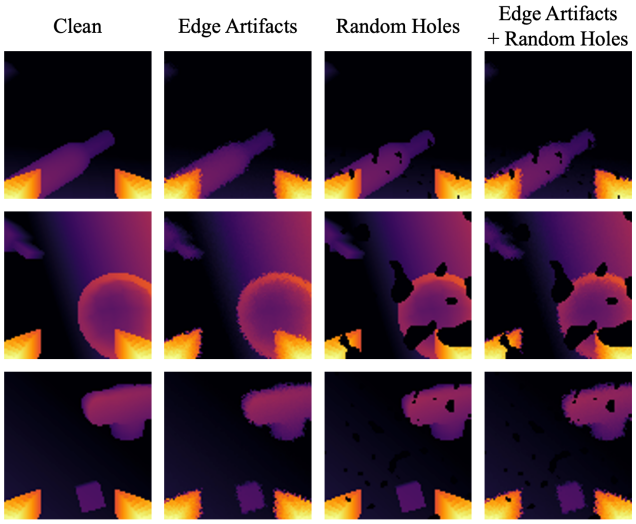


Fig. 4: **Depth Augmentation** Visualization of edge artifacts and random holes on depth maps.

```
def manipgen(robot, vlm, pose_est, mp, lp):
    """
    robot: control interface for robot
    vlm: Visual Language Model (e.g. GPT-4o)
    mp: Motion Planner (e.g. Neural MP)
    pose_est: Pose Estimator (e.g.Grounded SAM)
    lp: Dict(skill->local policy net)
    """
    obs = robot.reset()

    plan = vlm.plan(text_prompt, obs)

    # The plan is of the form (object, skill).
    for obj, skill in vlm_plan:
        # pose estimation
        pose = pose_est.estimate(obj, skill, obs)

        # Motion Planning and Execution
        # takes robot to near the obj
        mp.motion_plan(pose, robot, obs)

        # Local Policies Execution
        # perform local interaction
        obs = robot.execute_skill(lp[skill])
```

Planning For the VLM to output a plan (of the same format as in PSL), we provide a system prompt to the VLM that provides it a detailed description of the skill library, their effects on the environment and when they can be used. We also provide it a list of hints as to what consistitutes reasonable plans, this is necessary as existing VLMs still lack strong spatial reasoning capabilities inherently, though prompting seems to alleviate this issue to a certain degree. We include several in-context examples that allow the model to understand the output format, prompt the model to justify its decisions (which helped produce better plans) and format the output as a JSON string (which resolved most of the parsing/formatting issues).

After planning, ManipGen loops through the plan, estimating the pose of the object/region of interest, motion planning there, and then executing the appropriate local policy.

Pose Estimation We begin by using Grounded-SAM to segment the object of interest, averaging the 3D points of the segmented pixels to determine its position. Orientation is then estimated depending on specific cases. Based on whether there is obstacle above the object, we leverage VLM to classify pick and place tasks into two scenarios: open-space (*e.g.*, table surfaces) and tight-space (*e.g.*, microwaves). We let the robot gripper point downwards in open-space setting. For open-space picking, we project the object's points onto the XY plane and apply damped least square to fit the points, estimating the object's longest axis to get gripper's rotation along Z-axis. For open-space placing, we simplify the problem by selecting a fixed orientation pointing down. In tight-space pick and place, we first sample a set of robot poses around the object. We then capture point cloud of the current scene (excluding the robot) and evaluate the number of points in collision with each sampled pose. To bias towards poses that are further from obstacles, we apply Gaussian noise $\mathcal{N}(0.0, 0.1)$ to the points, and select the pose with minimal collision. For articulated objects, we estimate position of the handle and whether it is vertical or horizontal. Then we sample a set of target poses around the handle and use the same collision-checking method as in tight-space scenario to identify the target pose.

Motion Planning We use the released Neural MP code and checkpoint to perform motion planning given the target pose and current point-cloud. We perform open-loop motion planning with test-time optimization batch size of 64 (the paper used 100) and max path length of 100. Neural MP can produce paths that are jerky and non-smooth at times. As a result, we perform EMA smoothing with $\alpha = 0.9$ to reduce the jerkiness of the trajectories.

Local Policies Once initialized near the object of interest, we segment the target object in the first frame using Grounded SAM). We store this mask along with the depth map of the first frame and then we deploy the local policy. At each step, we pass in the segmentation mask of the target object, the first frame depth map, the current frame depth map and the proprioception to the policy. We run the Task Space Impedance Controller on the robot at 60Hz for a fixed, skill specific duration (max 8s). Then, depending on the skill, we either open or close the gripper and begin executing the next stage.

IX. Experiment Details

Hardware For all of our experiments, we use a Franka Emika Panda Robot, which is a 7 degree of freedom manipulator arm. We control the robot using the Industreallib library (https://github.com/NVlabs/industreallib) using the Task Space Impedance Controller. For deployment, we use Leaky PLAI with action scales, thresholds and skill deployment durations chosen per skill (Table IX). For all skills, we used a position gain of 1000 and rotation gain of 50.

	Pick	Place	Grasp Handle	Open	Close
Duration	5s	4s	6s	8s	4.5s
Action Scale Pos	.002	.002	.002	.003	.005
Action Scale Rot	.05	.05	.004	.0005	.0005
Leaky PLAI Thresh X, Y	.02	.02	.02	0.02	0.03
Leaky PLAI Thresh Z	.02	.02	.02	.003	.003
Leaky PLAI Thresh Rot (Deg)	4	4	2	.005	.005

TABLE IX: Configurations for skill deployment.

The robot is mounted to a fixed base pedestal behind a desk of size .762m by 1.22m with variable height. For global views (used for the VLM, pose estimation and Neural MP), we use four calibrated depth cameras, Intel Realsense 455, placed around the scene in order to accurately capture the environment. We project the depth maps from each camera into 3D and combine the individual point-clouds into a single scene representation for Neural MP and pose estimation. For input to Neural MP, we further process the point-clouds according to the paper [25]. While the VLM and the pose estimators can take in multiple views in principle, in practice we found that their results (predicted plans, pose estimations) were significantly less reliable and consistent when using more than a single camera. As a result, for each task, we select one out of the four global view cameras to use for plan prediction and pose estimation. Finally, for local views, we use the d405 camera mounted on the wrist, and pass its depth maps (after clamping and normalization) as input to the policy.

A. Simulated Comparison (Robosuite) Details

We zero-shot transfer our local policies (trained in IsaacGym) to the robosuite tasks. Note, we do not train on the Robosuite objects at all, we use our data generation and training pipeline to train local policies in IsaacGym and transfer the policies to Robosuite. To deploy our method, we modify the environment to use the UMI [66] gripper and Task Space Impedance Control. We use the same LLM-planning and motion planning infrastructure as used in the PSL [44] paper and evaluate our policies using 100 trials per task. All other numbers for baselines are taken from the PSL paper.

B. Furniture Bench Experiment Details

We replicate the exact task setup from the Transic [16] paper: 3D printed objects from FurnitureBench [71] in the

exact poses specified in their paper. In this experiment, we train local policies for these specific tasks and then deploy them. As a result, this experiment compares our method of sim2real transfer (local policies) with end-to-end sim2real transfer (w/ and w/o data aug) as well as Transic (which uses real-world data).

C. Real World Long-horizon Manipulation Details

For each of the following tasks, we specify the large object (if present) in each task (as a receptacle) as well as the list of objects we randomize, the task description in detail, and the task prompt provided to ManipGen and the baselines. Unless otherwise stated, all objects (handles for articulated objects) to interact with are positioned within 0.8 meters of the robot base to ensure they are within the gripper's reach. Some example scene arrangements are presented in Fig. 5.

1) Cook (2 stages): Pick up a food item on the cutting board and put it in a pot on the stove.

Task prompt: Put [OBJ] in the black pot.

Large object: black pot (39cm × 32cm × 14cm)

Randomized objects: carrot (17.0cm × 2.5cm × 2.5cm), cassava (20.0cm × 6.1cm × 6.0cm), corn (17.7cm × 4.2cm × 4.3cm), spice box (12.8cm × 9.0cm × 3.0cm), soup can (10.0cm × 6.7cm × 6.7cm)

Randomization: We put the food item on a cutting board, the pot on a stove, and randomize their poses across the table within the gripper's reach.

2) Replace (4 stages): Fetch a pantry item from the shelf, put it on a wooden board on the table and take an object from the table, put it on a white plate.

Task prompt: Place [OBJ A] on the wooden board, and put [OBJ B] on the white plate in the shelf.

Large objects: wooden board (51cm × 18.8cm × 1.2cm), shelf (80cm × 60cm × 23cm)

Randomized objects: [OBJ A] brown coffee package (15.7cm × 8.0cm × 6.0cm), blue coffee package (15.7cm × 8.0cm × 6.0cm), ketchup bottle (17.5cm × 9.3cm × 5.5cm), mustard bottle (17.5cm × 9.3cm × 5.5cm), gochujang bottle (17.4cm × 6.6cm × 4.1cm); [OBJ B] spice jar (8.0cm × 4.0cm × 4.0cm), pepper container (8.2cm × 5.8cm × 3.1cm), biscuit pack (10.5cm × 5.6cm × 2.5cm)

Randomization: The shelf is placed on the left or right end of the table with randomized orientation between 0 and 30 degrees. [OBJ A] and the white plate are randomly placed on the second or third level of the shelf. The wooden board and [OBJ B] are randomly placed on the other half of the table. We ensure that center of the board and [OBJ B] are at least 30cm away from base of the shelf.

3) CabinetStore (4 stages): Open the drawer with blue handle in the cabinet, put an office supply inside, and close the drawer.

Task Prompt: Store [OBJ] in the drawer with blue handle.

Large objects: cabinet (80cm × 71cm × 40cm, https://www.amazon.com/gp/product/B0CHHTZ52F/ref=ewc_pr_img_12?smid=A38QU35WKLBIKI&psc=1)

Randomized objects: computer mouse (10.5cm × 6.5cm × 3.8cm), tape (10.6cm × 6.4cm × 6.3cm), screw driver (18.8cm × 3.2cm × 3.2cm), plug (4.9cm × 4.9cm × 5.4cm), staple container (10.1cm × 5.9cm × 4.0cm)

Randomization: The cabinet is placed on the left or right end of the table, with its back aligned to the table edge. The office supply is randomly placed on a wooden tray on the other half of the table. We ensure that after fully opening the drawer, the office supply is at least 20cm from edge of the drawer.

4) DrawerStore (6 stages): Open a drawer with blue handle, put two personal care items inside, and close the drawer.

Task Prompt: Arrange [OBJ A] and [OBJ B] in the drawer with blue handle.

Large objects: drawer (80cm x 60cm x 23cm, https://www.amazon.com/gp/product/B0BJPLBSHQ/ref=ewc_pr_img_1?smid=A2XKE81PYMCHT4&psc=1)

Randomized objects: brush (8.4cm × 6.4cm × 5.4cm), sunscreen bottle (18.9cm × 5.7cm × 3.8cm), soap (9.4cm × 5.4cm × 2.5cm), toothpaste (14.1cm × 5.7cm × 3.6cm), sanitizer bottle (17.0cm × 6.8cm × 4.5cm)

Randomization: The drawer is placed on the left or right end of the table, with its back aligned to the table edge. We randomly select two personal care items and place them on a wooden tray on the other half of the table. We ensure that after fully opening the drawer, the personal care items are at least 20cm from edge of the drawer.

5) Tidy (8 stages): Clean up the table by putting 4 toys on the table into a bin.

Task Prompt: Sort all the toys into the black bin.

Large objects: black bin

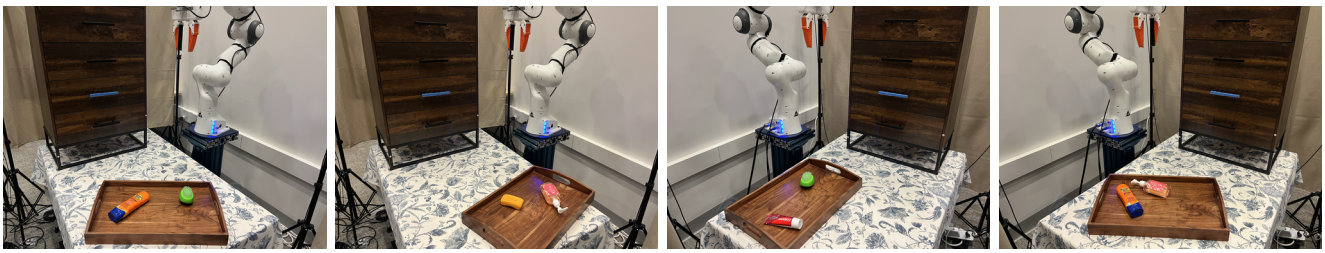
Randomized objects: stuffed carrot (17.2cm × 4.5cm × 4.5cm), stuffed corn (21.3cm × 5.7cm × 6.0cm), stuffed owl (15.0cm × 7.6cm × 6.6cm), stuffed dog (18.7cm × 7.8cm × 8.4cm), stuffed dice (6.6cm × 6.6cm × 6.6cm), tiny bottle (7.4cm × 2.9cm × 2.9cm), toy teapot (15.1cm × 12.5cm × 12.5cm), toy banana (8.8cm × 2.9cm × 4.0cm), toy corn (12.0cm × 3.8cm × 3.8cm), play-doh container (7.6cm × 6.3cm × 6.3cm)

Randomization: The bin is placed on the left, right, or front end of the table with randomized orientation between 0 and 180 degrees. The toys are scattered on the other half of the table. We ensure the toys are at least 20cm away from edge of the bin.

(a) Cook

(b) Replace

(c) Cabinet Store



(d) Drawer Store

(e) Tidy

Fig. 5: Example scene layouts for real world evaluation.