RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots

Soroush Nasiriany[1], Abhiram Maddukuri[1,*], Lance Zhang[1,*], Adeet Parikh[1], Aaron Lo[1], Abhishek Joshi[1], Ajay Mandlekar[2], Yuke Zhu[1,2]

[1]The University of Texas at Austin, [2]NVIDIA Research; [*]Denotes equal contribution

robocasa.ai

Fig. 1: **Overview of RoboCasa.** RoboCasa is a simulation framework for training generalist robot agents. Four pillars underlie RoboCasa: (1) Diverse assets, including 120 kitchen scenes and 2,500+ 3D objects, created with generative AI tools; (2) Cross-embodiment support for mobile manipulators and humanoid robots; (3) Diverse tasks created with the guidance of large language models; (4) Massive training datasets with over 100K trajectories.

***Abstract*—Recent advancements in Artificial Intelligence (AI) have largely been propelled by scaling. In Robotics, scaling is hindered by the lack of access to massive robot datasets. We advocate using realistic physical simulation as a means to scale environments, tasks, and datasets for robot learning methods. We present RoboCasa, a large-scale simulation framework for training generalist robots in household environments. RoboCasa features realistic and diverse scenes focusing on kitchen environments. We provide thousands of 3D assets across over 150 object categories and dozens of interactable furniture and appliances. We enrich the realism and diversity of our simulation with generative AI tools, such as object assets from text-to-3D models and environment textures from text-to-image models. We design a set of 100 tasks for systematic evaluation, including composite tasks generated by the guidance of large language models. To facilitate learning, we provide high-quality human demonstrations and integrate automated trajectory generation methods to substantially enlarge our datasets with minimal human burden. Our experiments show a clear scaling trend in using synthetically generated robot data for large-scale imitation learning and show great promise in harnessing simulation data in real-world tasks. Videos and open-source code are available on the project website.**

I. Introduction

Recent breakthroughs in Artificial Intelligence have been driven by training giant neural network models on Internet-scale datasets. Unlike computer vision and natural language processing domains, where massive visual and text data are abundant from online sources, robotic data is relatively scarce. A key question in Robotics is how to acquire robotic training data that captures the vast diversity and complexity of the real world. Several prominent recent attempts have been made to create large, diverse datasets for training generalist robot models [2, 9, 5, 20]. While these datasets have advanced robots' generalization abilities in narrow domains, there remains a considerable gap between the capabilities achieved thus far and general-purpose robots that can be reliably deployed in the wild. It raises the question — what is a viable path forward toward scaling in robot learning?

As collecting ever-larger datasets in the real world would require unrealistic amounts of capital and labor, many turn to simulation as a promising alternative to producing large

quantities of synthetic data for model training. We expect simulation to play an integral role in scaling robot learning for the following reasons. First, once a feature-rich, high-fidelity simulator is created, we can generate large amounts of robot data at low costs. This is exemplified by recent automated data generation methods, such as MimicGen [34] and Optimus [6], which exploit the privileged information of simulation to generate data with minimal human labor. Second, the creation of realistic simulations has been facilitated by rapid advances in generative AI. Today's generative AI tools are capable of generating images, synthesizing 3D assets, and writing source codes [38, 35, 41]. These tools can be employed to create millions of scenes procedurally, import novel categories of objects, and program natural tasks and reward functions. Finally, simulation democratizes and accelerates robot learning research, enabling rapid prototyping of new ideas and reproducible research.

To unleash the potential of simulation, it must satisfy three core criteria. First, the simulator must guarantee **realism** in physics, rendering, and underlying models to enable transfer to the real world. Second, the simulator must satisfy **diversity** in the scenes, assets, and tasks it offers. Generative AI will be crucial in enabling this diversity *at scale*. Finally, a simulator alone is not sufficient to train a highly capable generalist robot agent. The simulation must be accompanied by **large robot datasets** that capture the diversity of scenes and behaviors that it has to offer. Numerous prior attempts at creating simulations have partially satisfied some of these criteria, yet none have satisfied all.

We present `RoboCasa`, a large-scale simulation framework centered around home environments for training generalist robots. RoboCasa builds upon RoboSuite [48], a modular, fast, and easy-to-use framework based in MuJoCo. RoboCasa inherits these features and goes far beyond by offering a large array of scenes, objects, and hardware platforms suited for building a general-purpose home robot. In this initial release, we focus our efforts on kitchen scenes. To capture realistic and diverse scenes, we consult numerous architecture and home design magazines and compile several kitchen layouts and styles reflecting the diversity of kitchens in homes around the world. We model these kitchens according to standard size and spatial specifications and fit them with a large repository of interactable furniture and appliances spanning cabinets, stoves, microwaves, coffee machines, and more. Furthermore, we curate a repository of over 2,500 objects across over 150 categories, the majority of which are generated by text-to-3D tools. RoboCasa has cross-embodiment support for mobile manipulators of diverse forms, such as single-arm mobile platforms, humanoid robots, and quadruped robots with arms.

These assets allow us to simulate a wide range of behaviors in kitchen scenes. This release includes 100 tasks for systematic evaluation. The first 25 are atomic tasks that feature foundational robot skills, such as picking and placing, opening and closing doors, and twisting knobs. They serve as the basic building blocks to scaffold complex long-horizon tasks. The other 75 are composite tasks involving a sequence of robot skills. We design these composite tasks to capture naturalistic kitchen activities by soliciting suggestions from large language models (LLMs). Our key intuition is that LLMs are trained on human-centered Internet content, effectively capturing the ecological statistics of human behaviors. We obtain a list of activities from LLMs, such as washing dishes, frying, and restocking cabinets. Using these activities to ground our task design, we prompt the LLM to suggest concrete tasks for each activity. We look to expand the list of tasks in future releases.

We complement our tasks with high-quality human demonstrations across all 100 tasks. To augment our datasets, we extend MimicGen [34] to generate 100K additional trajectories for our atomic tasks. We train policies with behavioral cloning on human demonstrations and automatically generated data. We find that generated data significantly improves generalization, hinting at a promising path for scaling in robotics. Furthermore, we show in a real-world kitchen environment that co-training with our simulation data significantly increases task success in real-robot deployment. We summarize our contributions as follows:

- We develop the RoboCasa simulation framework featuring diverse, realistic kitchen scenes, thousands of high-quality object assets, and cross-embodiment mobile manipulators. We employ generative AI tools to create environment textures and 3D objects.
- We introduce a set of 100 tasks for systematic evaluation, including 25 atomic tasks representing foundational sensorimotor skills and 75 composite tasks generated with the guidance of large language models.
- We provide a large multi-task dataset for model training, including large-scale synthetically generated trajectories. We show a clear scaling trend when using generated data and show the utility of simulation data in real-world tasks.

II. Related Work

Simulation Frameworks for Robotics. Many simulation frameworks have been built for robotics — we provide a thorough comparison between RoboCasa and popular frameworks in Table I. Some are limited to tabletop manipulation [33, 15, 6, 28], but RoboCasa, along with others [21, 40, 26, 27, 10, 34], support mobile manipulation. RoboCasa also supports room-scale scenes, unlike other frameworks that include mobile manipulation in smaller portions of a room [10, 34]. RoboCasa runs realistic physics for all interactions, including object grasping and placement, unlike some other mobile manipulation frameworks such as AI2-THOR [21] and Habitat 2.0 [40]. RoboCasa is one of the few frameworks to feature photorealistic rendering [21, 40, 27, 10, 6] and multiple robot embodiments [21, 26, 27, 6, 34]. RoboCasa also has a large collection of tasks, scenes, and objects — Behavior-1K [27] is the only other framework to feature many tasks, scenes, and objects. But critically, RoboCasa is the only one to support AI-generated tasks and assets, ensuring potentially limitless diversity in scenes and tasks. Unlike many other simulation platforms (including Behavior-1K [27]), we provide large-scale datasets of task demonstrations through a combination

Feature	RoboCasa	AI2-THOR	Habitat 2.0	iGibson 2.0	RLBench	Behavior-1K	robomimic	ManiSkill 2	OPTIMUS	LIBERO	MimicGen
Mobile Manipulation	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✓
Room-Scale Scenes	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗
Realistic Object Physics	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
AI-generated Tasks	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
AI-generated Assets	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Photorealism	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗
Cross-Embodiment	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✓
Num Tasks	100	-	3	6	100	1000	8	20	10	130	12
Num Scenes	120	-	1	15	1	50	3	-	4	20	1
Num Object Categories	153	-	46	-	28	1265	-	-	-	x	-
Num Objects	2509	3578	169	1217	28	5215	15	2144	72	x	40
Human Data	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓
Machine-Generated Data	✓	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓
Num Trajectories	100K+	-	-	-	-	0	6K	30K	245K	5K	50K

TABLE I: **Comparison to Popular Simulation Frameworks used in the Robot Learning Literature.**

of human teleoperation and the MimicGen system [34] (more discussion below) and provide a thorough analysis of agents trained via imitation learning across our large collection of tasks. The convergence of diverse scenes, tasks, and assets alongside the extensive dataset provided by RoboCasa will fulfill a crucial requirement not addressed by any other simulation frameworks in the robot learning community.

Datasets and Benchmarks for Robotics. Recently, several large-scale data collection efforts have been made for robotics. One approach is self-supervised learning, where trial-and-error is used to collect data for tasks like grasping and pushing [24, 36, 18, 19, 44, 7]. This can take significant time to generate high-quality data due to the trial-and-error process. A popular alternative is to collect robot demonstrations via human teleoperation, where a human controls a robot to guide it through different tasks [46, 29, 30, 32, 9, 16]. Several recent efforts have scaled this paradigm up by using teams of human operators over extended periods of time [9, 1, 16, 2]. However, most of these efforts focus on collected real-world datasets. By contrast, we focus on collecting large-scale datasets in simulation, where results are easier to reproduce and thorough evaluations are possible due to lower human burdens.

Another approach is to leverage algorithmic trajectory generators in simulation [15, 45, 17, 10, 6], but these efforts often make use of privileged information and hand-designed heuristics, and can consequently be difficult to apply to arbitrary tasks without significant human effort. Some recent efforts have used Large Language Models to generate datasets in simulation [43, 12], but it can still involve carefully engineered pipelines. To combine the quality and wide applicability of human teleoperation with the scale of using pre-programmed demonstrators in simulation, we collect a set of human demonstrations in simulation and then leverage MimicGen [34]. This recently proposed data generation system synthesizes additional demonstrations using a set of human demonstrations to generate a much larger dataset.

Learning from Large Offline Datasets. Behavioral Cloning [37] is a popular method for learning policies offline from a set of demonstrations. It trains a policy to imitate the actions in the dataset. It has been used extensively in prior works [46, 31, 9, 2, 16, 6, 17]. Offline Reinforcement Learning [25] is an alternative method that tries to prefer certain dataset actions over others using a reward function. It has also been used to learn from large offline robot manipulation datasets [19, 3, 11, 23, 22]. In this work, we use Behavioral Cloning using a Transformer-based [42] visuomotor policy similar to other works [6] to train agents on large offline datasets. We also compare to other popular policy architectures such as diffusion models [14, 4] and recurrent neural networks [33].

III. RoboCasa Simulation

We outline the core simulation components of RoboCasa. We highlight our efforts to create diverse and realistic kitchen scenes, furniture and appliances, and objects.

A. Core Simulation Platform

We adopt RoboSuite [48] as the core simulation platform on which we develop RoboCasa. We chose RoboSuite because of its focus on physical realism, high speed, and modular design, which allows us to scale to large-scale scenes. We directly inherit several core components of RoboSuite, including the environment model formats and robot controllers. Crucially, in order to support room-scale environments, we extend RoboSuite to accommodate mobile manipulators, including robots mounted on wheeled bases, humanoid robots, and quadrupeds with arms. We also support high-quality rendering with NVIDIA Omniverse, allowing us to capture photorealistic images (see Figure 1).

B. Kitchen Scenes

In this initial release, we focus on household tasks centered around kitchen activities. We created a large array of kitchen scenes with fully interactive cabinets, drawers, and appliances. We consult online home design and architecture magazines

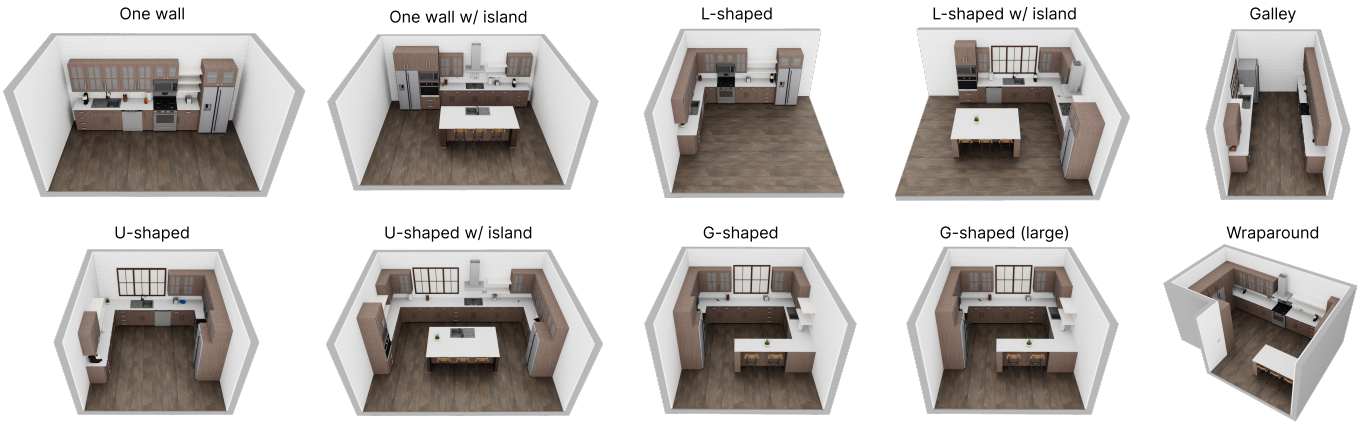

Fig. 3: **Kitchen Floor Plans.** We consult home planning and architecture magazines and compile a list of common kitchen floor plans. Our floor plans take on a variety of shapes and sizes, from basic designs (*e.g.*, one wall) to high-end ones (*e.g.*, U-shaped w/ island).

to compile a diverse list of kitchen floor plans. We model 10 floor plans (Figure 3) ranging from basic designs found in apartments to more elaborate designs found in high-end homes. Each kitchen can be configured to take on a custom architectural style. After consulting architecture magazines, we compile the popular kitchen styles, including Industrial, Scandinavian, Coastal, Modern, Traditional, Mediterranean, Rustic, and more. Each style features a unique combination of design elements, including textures, appliance choices, and cabinet panels and handles. For example, Scandinavian kitchens employ light, low-contrast textures and simple, sleek cabinet panels and appliances. In contrast, Mediterranean kitchens use ornate appliances, glass panel cabinets, and colorful textures. In total, we have modeled 12 kitchen styles, and we showcase these styles across different floor plans in Figure 1. Each floor plan can be configured to take on any style, resulting in 120 kitchen scenes. Each scene can be customized further by replacing textures from a large selection of high-quality AI-generated textures. We have 100 textures for walls, 100 for the floor, 100 for counters, and 100 for cabinet panels. We use the popular text-to-image tool MidJourney to generate these images. We use these textures as a form of domain randomization to significantly increase the visual diversity of our training datasets.

C. Assets

We create a large repository of intractable 3D assets to accommodate diverse kitchen activities. Our repository includes cabinets, drawers, and various kitchen appliances. We source these assets from 3D model repositories online and convert them to the MuJoCo MJCF model format. Our postprocessing operation involves segmenting appliances into articulated entities, for example, segmenting the door of a microwave and the knobs on a stove. It allows us to represent rich interactions, such as closing a microwave door or turning on a stove. Furthermore, these appliances undergo state changes, *e.g.*, when we turn a stove knob on, the corresponding burner turns on to simulate heat. See Figure 4 for an illustration of our appliances.

Fig. 4: **Examples of Interactable Appliances.** Our simulation framework comes with dozens of appliances. Several types of appliances are articulated. For example, we can open and close doors on microwaves and twist knobs on stoves. Some appliances can undergo state changes, *e.g.*, when we turn the knob on the stove, the corresponding burner turns on.

In addition to appliances, we create a rich library of objects commonly found in kitchens, spanning fruits and vegetables, dairy, poultry, drinks, receptacles, tools, and more. We gather object assets from two sources, the Objaverse [8] dataset and Luma.ai, an online text-to-3D service. We mine a large set of candidate objects and filter out defective or low-quality ones. At the end of this process, we collect 2,509 high-quality assets spanning 153 unique object categories. See Figure 5 for an illustration of our objects.

IV. RoboCasa Activity Dataset

Our simulator supports a wide array of possible kitchen activities, and we represent these activities with a comprehensive suite of 75 tasks. This section outlines these tasks and our large multi-task dataset accompanying them.

A. Atomic Tasks: Building Blocks of Behavior

For a robot to perform complex tasks, it must master the foundational skills needed to solve these tasks. We focus on a set of eight sensorimotor skills that form the basis for the majority of household activities: 1) Pick and place, 2) Opening and closing doors, 3) Opening and closing drawers, 4) Twisting knobs, 5) Turning levers, 6) Pressing buttons, 7) Insertion, and

Fig. 5: **Diverse High-Quality 3D Objects.** RoboCasa offers 2,509 high-quality 3D objects across 153 diverse categories spanning vegetables, poultry, drinks, and more. Here we illustrate a small subset of these objects.

8) Navigation. These skills do not constitute an exhaustive list, and including additional skills centered around behaviors such as deformable manipulation is left for future work. To effectively learn the skills, we propose a set of 25 tasks that each involve one of these eight skills. We will refer to these as *atomic tasks*. The full breakdown of these tasks is outlined in the appendix (Figure 13).

B. Creating Composite Tasks with Large Language Models

Our composite tasks involve sequencing skills to solve semantically meaningful activities such as cooking and cleaning. Our goal in creating these tasks is to capture diverse tasks that reflect the ecological statistics of real-world household activities. We use the guidance of large language models (LLMs) to define our tasks. This approach offers several key benefits. First, LLMs encapsulate diverse sources of human knowledge and can thus effectively communicate diverse ideas grounded in the real world. In addition, these LLMs can be used at scale to define thousands of unique tasks, significantly reducing the human labor involved in task definition. We generate tasks across two steps (see Figure 6). First, we prompt ChatGPT (GPT-4) to list common high-level kitchen activities. We compile a list of 20 activities: *brewing coffee or tea*, *washing dishes*, *restocking kitchen supplies*, *chopping food*, *making toast*, *defrosting food*, *boiling water*, *meat preparation*, *setting the table*, *clearing the table*, *sanitize the surface*, *snack preparation*, *tidying cabinets and drawers*, *washing fruits and vegetables*, *frying*, *reheating food*, *mixing and blending*, *baking*, *serving food*, and *steaming vegetables*. We then prompt GPT-4 and Gemini 1.5 to propose representative tasks for each activity label. The LLMs occasionally exhibit logical flaws, so we must filter or modify some of their outputs. We compile 75 task *blueprints* in total from the LLM and proceed to code implementations for them. Except for a select number of composite tasks designed to work in certain environments, all tasks are simulatable in any of our kitchen scenes. We describe in detail our prompts and tasks in the appendix.

C. RoboCasa Datasets

We outline a comprehensive set of 100 tasks consisting of 25 atomic tasks (Sec. IV-A) and 75 composite tasks created with LLMs (Sec. IV-B). This section explains how we collect our large-scale demonstration dataset across these tasks. We first use human teleoperation to collect a base set of demonstrations and then use automated trajectory generation methods to expand this to a much larger set of demonstrations.

Collecting a base set of demonstrations through human teleoperation. A team of four human operators collect 50 high-quality demonstrations for each atomic task using a 3D SpaceMouse [47, 48]. Each task demonstration is collected in a random kitchen scene (random kitchen floor plan, random kitchen style, and random AI-generated textures). This results in large and diverse simulation datasets through human teleoperation (1,250 demonstrations). However, our experiments show that even this scale of human data is insufficient to solve most of our tasks. This is likely due to the significant scope and diversity of the tasks and scenes. Consequently, we opt to use data generation tools to expand our data quantity.

Leveraging automated trajectory generation methods to synthesize demonstrations. To further scale the dataset's size with minimal human effort, we employ MimicGen [34], a recently developed trajectory generation method. MimicGen can automatically synthesize rich datasets from a seed set of human demonstrations by adapting them to new settings. The core generation mechanism first decomposes each human demonstration into a sequence of object-centric manipulation segments. Then, for a novel scene, it transforms each object-centric segment according to the current pose of the relevant object, stitches the segments together, and has the robot follow the new trajectory to collect a new task demonstration.

MimicGen requires some basic assumptions on simulation. We outline how they are easily satisfied in RoboCasa: MimicGen assumes that tasks consist of a known sequence of object-centric subtasks — this sequence must be specified for each new task. Fortunately, the atomic tasks (Sec. IV-A) consist of eight core skills. All tasks that correspond to a skill have the same or similar sequences of object-centric subtasks, with the main differences coming from the identity of the reference object. For example, for pick-and-place tasks, the first stage is a pick subtask with one reference object, and the second stage is a place subtask with a second reference object. Consequently, specifying subtask sequences takes minimal human effort, *i.e.*, a small fraction of the demonstration collection time. In addition, each human demonstration provided to MimicGen must also be annotated with segments corresponding to each object-centric subtask. This can be done with automated metrics that detect the end of each subtask — these functions only need to be implemented once for each of the eight core skills and can be re-used across the entire set of demonstrations.

MimicGen data generation attempts are not always successful. In practice, it employs a rejection sampling scheme only to keep generation attempts that lead to task success. Leveraging RoboCasa simulation, we can parallelize MimicGen data

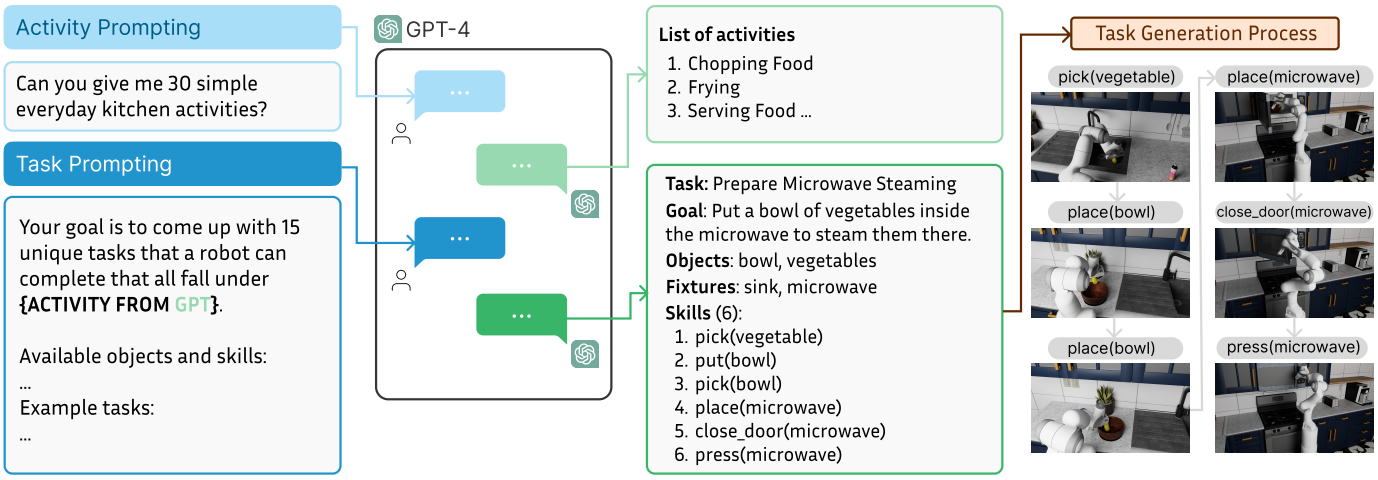

Fig. 6: **Creating Diverse Tasks with Large Language Models.** We employ GPT-4 to generate diverse tasks. First, we prompt GPT-4 to give diverse high-level kitchen activities. Subsequently, for each activity, we prompt GPT-4 (or Gemini 1.5) to suggest a diverse set of representative tasks. We illustrate one such task "Preparing the Microwave For Steaming" for the activity label "Steaming Vegetables".

generation across multiple simulation processes to speed up the data generation process.

V. Experiments

We aim to explore the following research questions in our experiments:

1) How effective are machine-generated trajectories from MimicGen in learning multi-task policies, in comparison to human demonstrations?
2) How will the generalization performances of the imitation learning policy scale with increasing training dataset sizes?
3) Can large-scale simulation datasets facilitate knowledge transfer to downstream tasks within simulation and facilitate policy learning for real-world tasks?

A. Imitation Learning for Atomic Tasks

First, we perform a systematic study with the atomic tasks. One question we are interested in is how the performances of the imitation learning policies compare when trained on human data versus machine-generated data and how the scale of this data plays a role in performance. We compare the following four multi-task data settings:

1) **Human-50**: A dataset of 1250 human demonstrations spanning all 25 atomic tasks, each with 50 human demonstrations.
2) **Generated-3000**: A dataset of 72,000 demonstrations synthesized by MimicGen across 24 atomic tasks. (We exclude the kitchen navigation task, as MimicGen is not currently able to generate mobile manipulation trajectories.) We take the 50 human demonstrations as input for each task and use them to generate 3,000 trajectories autonomously.
3) **Generated-300**: A random subset of 10% of our full generated dataset, where we generate 300 demos per task. This results in a total of 7,200 trajectories.
4) **Generated-100**: A random subset of 3.3% of our full generated dataset, where we generate 100 demos per task. This results in a total of 2,400 trajectories.

Images in the training datasets are rendered with AI-generated textures. We evaluate our policies in kitchen scenes with human-curated textures. In these datasets, our focus is specifically on a Franka Panda robot with an Omron mobile base, resembling the Omni-Frankie robot [13]. We train a visuomotor policy with behavioral cloning on each of these four multi-task datasets. We specifically use the publicly available BC-Transformer implementation in RoboMimic [33]. Refer to Section X for additional details.

After training, we perform a comprehensive evaluation of the model. For each task, we evaluate the model performance across 50 trials across five fixed evaluation scenes, each with a distinct floor plan and style. We highlight that two of these scenes encompass unseen styles that were never encountered in the training data. In order to test generalization capabilities, we only evaluate the policy only on unseen object instances. We report results in Figure 7 where we group results together with tasks belonging to the same skill. The overall performance on human data is 28.8% success rate, and with the fully generated dataset, we observe a significant improvement at 47.6% success rate. Furthermore, we observe a *scaling trend* from using machine-generated data: as we increase the quantity of generated data, the model performance increases steadily. This offers a promising outlook: data generation tools enable us to learn significantly more performant agents at a relatively low cost. We observe several other notable findings in the results. Some skills are significantly easier to learn (*e.g.*, opening and closing doors and drawers), while others are quite challenging (*e.g.*, pick and place). We hypothesize a number of factors explaining these findings. First, tasks exhibiting high diversity are significantly more challenging to learn. One example is the pick and place tasks involving 69 different object categories with a wide range of affordances. In comparison, opening and

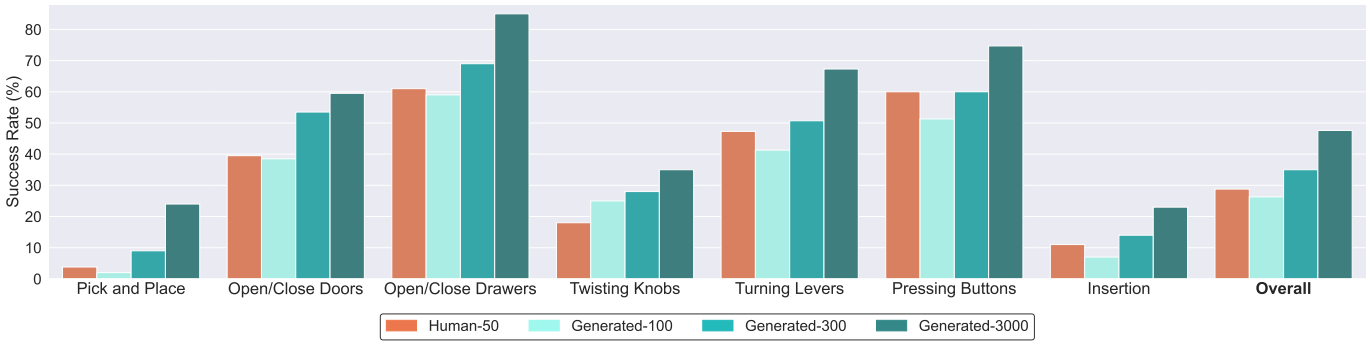

Fig. 7: **Comparison between human demonstrations and machine-generated datasets.** We present learning results across 24 atomic tasks spanning diverse robot skills. We compare training on four different multi-task datasets, including a human dataset with 50 demonstrations per task, a machine generated dataset with 3000 demonstrations per task, and smaller variants with 300 or 100 demonstrations per task. We see a clear scaling trend: increasing the size of the generated dataset can yield consistently higher overall success rates, eventually significantly outperforming performance on human datasets.

closing doors involves six different instances of doors and is thus significantly easier to learn. Another factor is dexterity, as we see that tasks involving high levels of dexterity, such as insertion, are challenging to learn.

B. Imitation Learning for Composite Tasks

Next we study learning on our composite tasks. These tasks are more challenging as they require multiple skills, which increases the horizon of the task and introduces new subtleties. Due to the increased difficulty of these tasks and the challenges of multi-task learning, we opt to learn a single-task policy for each task. For each task, we collected 50 human demonstrations and compared the following settings:

- **Scratch**: learning a policy from scratch on these 50 demonstrations;
- **Fine-tuning**: taking our pre-trained policy learned on atomic tasks with the full MimicGen generated dataset and fine-tuning on these 50 demonstrations.

We independently train models for the following five tasks:

- `ArrangeVegetables`: This task belongs to the "chopping food" activity. The robot must place two vegetables from the sink onto the cutting board on the counter;
- `MicrowaveThawing`: This task belongs to the "defrosting food" activity. The robot must place a frozen food item from the counter to the sink and turn on the sink faucet;
- `RestockPantry`: This task belongs to the "restocking kitchen supplies" activity. The robot must pick and place multiple cans from the counter to the cabinet. There are a number of cans already in the cabinet, among other objects. The robot must locate the existing cans in the cabinet (either on the right or left side) and place the new cans right next to them;
- `PreSoakPan`: This task belongs to the "washing dishes" activity. The robot must pick and place a pan and a sponge into the sink to prepare the pan for washing;
- `PrepareCoffee`: This task belongs to the "brewing coffee of tea" activity. The robot must take a mug out of the cabinet, place it under the coffee machine, and press the coffee machine button to serve it into the mug.

	Scratch	Fine-tuning
`ArrangeVegetables`	2.0%	12.0%
`MicrowaveThawing`	0%	2.0%
`RestockPantry`	0%	6.0%
`PreSoakPan`	0%	4.0%
`PrepareCoffee`	0%	0%

Fig. 8: **Learning Results on Composite Tasks.** We learn single-task policies for five representative composite tasks. We compare learning these tasks from scratch with 50 human demonstrations versus fine-tuning a policy trained on machine-generated atomic task data. The fine-tuning method performs better but still struggles to learn robust behaviors.

See Figure 8 for results. Learning on these composite tasks is very challenging, with the Scratch baseline failing to achieve any non-zero success rate on 4/5 tasks. The fine-tuning method achieves non-zero success rates on 4/5 tasks. Some common failure modes include difficulty with fine-grained manipulation and difficulty effectively transitioning to the next stage of the task. However, we generally observe that the fine-tuned models perform better qualitatively, with more robust picking and placing strategies in particular. We attribute this to the large pretraining dataset of atomic behaviors. Our benchmark leaves room for significant improvement on these tasks. The choice of policy architecture, learning algorithm, and fine-tuning strategy may play a critical role in performance, and these factors warrant investigation in future work.

C. Transfer to Real World Environments

We show how large-scale data generated in simulation can aid in learning tasks in RoboCasa and other domains, including in the real world. We conduct experiments in a real-world kitchen environment with a Franka Emika Panda robot running on the DROID hardware infrastructure [20]. While both the real world and simulated Franka robot are controlled via workspace end effector control, our simulated robot uses Operational Space Control while the DROID-based real robot does not. In addition, our robot controller runs at 20 Hz frequency while the real robot controller runs at 15 Hz.

Fig. 9: **Real-World Experiment Setup.** We conduct experiments in a real-world kitchen environment with a Franka Emika Panda arm on a wheeled mobile platform. The two pictures illustrate two of the three evaluation tasks: (left) pick and place an object from the counter to the cabinet, and (right) pick and place an object from the counter to the sink.

Setting	Task	Real only	Real + Sim (Ours)
Seen Obj	Counter to sink	12.7 ± 2.5	**22.0 ± 2.8**
	Sink to counter	20.0 ± 5.9	**29.3 ± 4.1**
	Counter to cabinet	8.0 ± 1.6	**22.0 ± 5.8**
	Task average	13.6	**24.4**
Unseen Obj.	Counter to sink	3.3 ± 4.7	**8.9 ± 7.9**
	Sink to counter	1.1 ± 1.6	**7.8 ± 4.2**
	Counter to cabinet	3.3 ± 4.7	**11.1 ± 11.0**
	Task average	2.6	**9.3**

Fig. 10: **Real Robot Evaluations.** In a real-world kitchen domain with only a handful of demonstrations, we explore co-training policies with our simulation data. Compared to training policies exclusively on in-domain real-world demonstrations, we see that co-training (sim+real) substantially improves policy performance.

In addition to controller differences, there are differences in camera calibration, lighting conditions, and the placement of the robot base with respect to the scene.

Our experiments include three tasks in the real kitchen environment: 1) pick and place an object from the counter to the sink, 2) pick and place an object from the sink to the counter, and 3) pick and place an object from the counter to the cabinet. These tasks resemble single-stage tasks in RoboCasa. For each task, we collected 50 demonstrations, each over five distinct object categories. We train a policy for each task, and we compare the following two settings:

- **Real only**: training on the real-world demos for the target task only;
- **Real + Sim**: co-training on real-world demos for the target task and all of our simulation MimicGen demonstrations over all single-stage tasks.

In Figure 10, we report policy success rates (mean and standard deviation, in percentage) averaged over 3 seeds. For each seed, we evaluate the model over five seen object categories and 3 unseen object categories (unseen with respect to the real-world demonstrations). On seen objects, we see that co-training with simulated data yields a 24.4% average success rate, compared to 13.6% with using real data only, a relative improvement of 79%. While performance suffers on unseen objects, we still see a significant improvement in incorporating simulation data. We attribute this to the rich diversity and visual and physical realism of our simulator.

VI. Conclusion

We have presented RoboCasa, a large-scale simulation framework for training generalist robots in everyday environments. RoboCasa features 120 realistic scenes, dozens of appliances, 2,500+ high-quality 3D objects spanning 150+ categories, 100 diverse tasks, and a large multi-task dataset of 100K+ trajectories. Generative AI tools play a central role in the creation of our simulation, with object assets from text-to-3D models, environment textures from text-to-image models, and LLMs to generate kitchen activities and corresponding tasks. In our experiments, we show that synthetically generated data in simulation can be useful in scaling robot policy learning.

We now pinpoint limitations and discuss exciting avenues for future future. First, our experiments show that fine-tuning on composite tasks yields relatively low performance, leaving room for improvement. In the future, we will investigate more powerful policy architectures and learning algorithms and improve the quality of our machine-generated datasets. While the generated trajectories are technically considered successful, many exhibited undesirable effects, such as jerky motions and collisions. Many of these behaviors can be automatically detected by checking simulation states and trajectories exhibiting such behaviors can be discarded. Next, while we showed how to use LLMs to create tasks, this process still required human guidance to write the implementations for these tasks. One interesting research direction is to use LLMs to propose thousands of new scenes and tasks and write code to implement these scenes and tasks with minimal human guidance. We anticipate that this will be possible as LLMs become more performant. In addition, we would like to extend the scope beyond kitchen environments and tasks in future releases. Next, our dataset consists of critical coarse manipulation behaviors but does not encapsulate highly dexterous skills, deformable manipulation tasks, or tasks that require bimanual manipulation. Finally, we are interested in exploring training using a combination of data from our simulation, other simulators, and diverse sources such as internet videos and real robot datasets. We envision a future in which these diverse forms of data complement each other to create a powerful foundation model for robotics.

Acknowledgments

We thank Yifeng Zhu for putting forth significant efforts in the cross-embodiment efforts. We would also like to thank Yuqi Xie for rendering support. Finally, we would like to thank all UT Austin Robot Perception and Learning Lab members for their invaluable feedback on RoboCasa. This work has been partially supported by the National Science Foundation (FRR-2145283, EFRI-2318065) and the Office of Naval Research (N00014-22-1-2204).

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

[3] Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

[4] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[5] Open X-Embodiment Collaboration et al. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[6] Murtaza Dalal, Ajay Mandlekar, Caelan Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv:2305.16309*, 2023.

[7] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, 2019.

[8] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[9] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *Robotics: Science and Systems (RSS)*, 2022.

[10] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

[11] Nico Gürtler, Sebastian Blaes, Pavel Kolev, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Bernhard Schölkopf, and Georg Martius. Benchmarking offline reinforcement learning on real-robot hardware. *arXiv preprint arXiv:2307.15690*, 2023.

[12] Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[13] Jesse Haviland, Niko Sünderhauf, and Peter Corke. A holistic approach to reactive mobile manipulation. *IEEE Robotics and Automation Letters*, 7(2):3122–3129, 2022.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[15] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[16] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2021.

[17] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *International Conference on Machine Learning*, 2023.

[18] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[19] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[20] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset, 2024.

[21] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. AI2-THOR: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[22] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021.

[23] Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiko Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *arXiv preprint arXiv:2210.05178*, 2022.

[24] Sergey Levine, Peter Pastor, Alex Krizhevsky, and

Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*, pages 173–184, 2016.

[25] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[26] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[27] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[28] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[29] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[30] Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. *arXiv preprint arXiv:1911.04052*, 2019.

[31] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. In *Robotics: Science and Systems (RSS)*, 2020.

[32] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation, 2020. URL https://arxiv.org/abs/2012.06733.

[33] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

[34] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[35] OpenAI et al. GPT-4 technical report, 2024.

[36] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE Int'l Conference on*. IEEE, 2016.

[37] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[39] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[40] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021.

[41] Gemini Team. Gemini: A family of highly capable multimodal models, 2024.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[43] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[44] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *Int'l Conference on Intelligent Robots and Systems*, 2016.

[45] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, 2020.

[46] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[47] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[48] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

VII. Additional Experiments

A. Task Diversity Experiments

We investigate the impact that task diversity in training data has on policy performance. This is an extension of the atomic task study in the main experiments, in which we consider 20 atomic tasks. Each of these 20 tasks is associated with one or more task variants (refer to Section IX-A for details). In this experiment, we compare two settings: (1) training on 320 task variants versus (2) training on 5% of these task variants, *i.e.*, 16 task variants. For setting (1) we used the same Generated (45k) MimicGen dataset that we present in the main experimental study, which captures 320 task variants. For setting (2), we first compiled a list of all possible task variants and sampled a random subset of 16 of them. Due to the large number of pick-and-place task variants, all of the task variants sampled were from five pick-and-place tasks. For these sampled task variants, we generate an equivalently sized MimicGen dataset with 45k demos. We compare the policy performance on these five pick-and-place tasks (which we call "seen" tasks) and the performance on all 20 atomic tasks. Note that for all pick-and-place tasks, we evaluate unseen object categories, following the same protocol outlined in the main experiments. Thus, even for the seen tasks, we evaluate on task variants that neither method has seen in the training data. We report these results below:

	Evaluation on seen tasks	Evaluation on all tasks
Training on 16 task variants	13.6%	3.8%
Training on 320 task variants	30.4%	40.2%

Fig. 11: **Multi-Task Learning Results on Task Diversity.** These results are obtained by training policies on 16 task variants versus 320 task variations prior to fine-tuning on human demonstrations.

We see that the method trained on 16 task variants performs significantly worse. Even for seen tasks, this method still performs half as well as the method trained on 320 task variants.

Across all tasks, the difference is even more pronounced, as this method completely fails on tasks representing other skills such as twisting knobs and opening and closing doors. This is perhaps unsurprising given that the 16-task variant method was only exposed to pick and place task variants and none of the other skills. To compensate for this limitation we also experiment with fine-tuning each method on the 1,000 human demonstrations for all 20 atomic tasks. Note that the human demonstrations capture a large amount of task diversity; however, they constitute a relatively small amount of data. We report the results below.

We see that even with fine-tuning, pre-trained on 320 task variants outperforms the 16 task variant method by nearly 10% for both seen tasks and over all tasks. These results imply that training on both (1) diverse tasks and (2) large quantities of data for these tasks, are important for policy performance.

	Evaluation on seen tasks	Evaluation on all tasks
Training on 16 task variants	27.2%	31.8%
Training on 320 task variants	36.8%	42.0%

Fig. 12: **Multi-Task Learning Results on Task Diversity.** These results are obtained after fine-tuning the models on human demonstrations for all 20 atomic tasks.

VIII. Simulator

We benchmark the speed of our simulator on the `PickPlaceCounterToCab` task, running for 10 episodes, with each episode spawned in a random scene. We use native MuJoCo rendering on an NVIDIA RTX A5000 GPU which adds overhead to the simulation speed. The simulation physics runs on CPUs and we specifically used an AMD EPYC 7543 32-Core Processor. The average time to reset the scene is 9.50 seconds and the average speed of stepping through the simulator (calling `env.step`) is 25.2 fps. Without rendering, the average reset time is 9.46 seconds and the simulator speed is 31.9 fps. For reference, each timestep in our simulator corresponds to 0.04 seconds in the world or 25 fps. Our observed simulation speed with the rendering of 25.2 fps almost exactly matches this speed, meaning our simulator runs roughly at real-time speed.

IX. Tasks and Datasets

A. Atomic Tasks

We design 25 atomic tasks representing eight foundational robot skills: (1) Pick and place, (2) Opening and closing doors, (3) Opening and closing drawers, (4) Twisting knobs, (5) Turning levers, (6) Pressing buttons, (7) Insertion, and (8) Navigation. We outline details for all tasks in Figure 13. Each task can come in the form of multiple task variants. These task variants can disambiguate the task goal with language. For pick-and-place tasks, there is a task variant for each object category being manipulated to disambiguate objects in clutter. For turning on and off the stove, there is a task variant for each burner being turned on or off, to disambiguate the relevant burner. For navigation, there is a task variant for each appliance the robot is navigating to, to disambiguate the target location.

B. Composite Tasks

We obtain activity labels by asking ChatGPT the following simple prompt: "Can you give me 30 simple everyday high-level kitchen activities? Each activity should be unique." We obtain a set of candidate responses and manually select 20 activities. We use a more elaborate prompt to obtain task suggestions, listing the robot skills, relevant objects, and

constraints of the simulation. We have attached the prompt under the file `chatgpt_task_prompt.txt`. We list all 20 kitchen activities and representative tasks across all activities in Figure 14.

In selecting composite tasks, we filter out LLM task suggestions that have logical flaws. For a list of examples, see below:

Using invalid object (no blender exists):

```
Task: Set Up Blending Station
Goal: Place dairy ingredients next to the
blender for making creamy fillings or batter.
Objects: cheese, milk
Fixtures: counter, blender
Skills (3): Pick_up(dairy), Place(counter),
Push(dairy, blender)
Reasoning: Preparing dairy products near a
blender is common when making mixtures for
baking.
```

Improper use of skills:

```
Task: Wine Selection for Cooking
Goal: Retrieve a bottle of wine from the
cabinet for use in a recipe
Objects: drink (wine)
Fixtures: cabinet Skills (5):
Open(cabinet), Pick_up(drink),
Place(counter), Close(cabinet),
Press(button_on_coffee_machine) (simulating
uncorking)
Reasoning: Wine is occasionally used in baking
recipes and needs to be opened and ready to
use.
```

Picking up objects that should not be grasped (utensils):

```
Task: Retrieve Baking Utensils
Goal: Gather all utensils needed for baking
(spoons, ladle) and place them on the
countertop.
Objects: utensils
Fixtures: drawer, counter
Skills (4): Open(drawer), Pick_up(utensils),
Place(counter), Close(drawer)
Reasoning: Assembling the correct utensils is
essential before starting to bake.
```

C. Datasets

In our atomic task experiments, we use human datasets and machine-generated datasets over 20 tasks. We did not cover all 25 atomic tasks due to time constraints but we will expand these datasets to cover all tasks for the public release. We render images for these tasks using randomly sample AI-generated textures, which replace the native textures for each scene. Due to the high volume of these datasets and time constraints, we opt to use the MuJoCo renderer to render images for these datasets. The MuJoCo renderer is faster to use, however this comes at the expense of lower quality photorealism. For the public release we will provide users the option to render all datasets with the Omniverse renderer.

X. Policy Learning Implementation

Our BC-Transformer policy takes as input the history of the past 10 observations in addition to the language goal for the text and outputs the next ten actions for the robot to execute. The agent replans after executing the first action. We modified the policy to support language conditioning by encoding language goals using a CLIP sentence encoder. For each observation in the input, the policy encodes proprioceptive information (end-effector pose and mobile base pose) and images from three cameras: an eye-in-hand camera, a left workspace camera, and a right workspace camera. It encodes each of these images with a dedicated ResNet-18 encoder stack and fuses the visual representation using FiLM layers. The encoded observations are passed to a 6-layer Transformer with $\sim 20M$ trainable parameters. We train the model for 500k gradient steps at a learning rate of $1e-4$ with a learning rate warmup.

We also experiment with diffusion policy [4]. We implement the diffusion policy in RoboMimic for a fair comparison to our existing BC-Transformer method. The diffusion policy uses the same observation encoder (ResNet, FiLM conditioning) as the BC-Transformer. We use all recommended hyperparameters from the official implementation: 2 timesteps for observation history, a prediction horizon of 16 steps, and an action horizon of 8 steps. We use DDIM [39] with 100 train timesteps and 10 inference timesteps, as recommended by the Diffusion Policy authors. We found the Diffusion Policy to underperform the BC-Transformer implementation significantly. On the single-stage PickPlaceCounterToSink task, BC-Transformer achieves a 56% success rate while Diffusion Policy only achieves 12%. One possible explanation for this is that our BC-Transformer implementation uses a longer history length of 10 observations, while diffusion policy uses a history length of 2 observations (this is the default choice used by Chi et al. which we also opt to use). Incorporating a longer observation history may be critical for our tasks.

Task	Skill Family	Description	# Task Variants
PickPlaceCounterToCabinet	pick and place	Pick an object from the counter and place it inside the cabinet. The cabinet is already open.	69
PickPlaceCabinetToCounter	pick and place	Pick an object from the cabinet and place it on the counter. The cabinet is already open.	69
PickPlaceCounterToSink	pick and place	Pick an object from the counter and place it in the sink.	58
PickPlaceSinkToCounter	pick and place	Pick an object from the sink and place it on the counter area next to the sink.	30
PickPlaceCounterToMicrowave	pick and place	Pick an object from the counter and place it inside the microwave. The microwave door is already open.	32
PickPlaceMicrowaveToCounter	pick and place	Pick an object from inside the microwave and place it on the counter. The microwave door is already open.	40
PickPlaceCounterToStove	pick and place	Pick an object from the counter and place it in a pan or pot on the stove.	32
PickPlaceStoveToCounter	pick and place	Pick an object from the stove (via a pot or pan) and place it on (the plate on) the counter.	32
OpenSingleDoor	opening and closing doors	Open a microwave door or a cabinet with a single door.	1
CloseSingleDoor	opening and closing doors	Close a microwave door or a cabinet with a single door.	1
OpenDoubleDoor	opening and closing doors	Open a cabinet with two opposite-facing doors.	1
CloseDoubleDoor	opening and closing doors	Close a cabinet with two opposite-facing doors.	1
OpenDrawer	opening and closing drawers	Open a drawer.	1
CloseDrawer	opening and closing drawers	Close a drawer.	1
TurnOnStove	twisting knobs	Turn on a specified stove burner by twisting the respective stove knob.	7
TurnOffStove	twisting knobs	Turn off a specified stove burner by twisting the respective stove knob.	7
TurnOnSinkFacuet	turning levers	Turn on the sink faucet to begin the flow of water.	1
TurnOffSinkFaucet	turning levers	Turn off the sink faucet to begin the flow of water.	1
TurnSinkSpout	turning levers	Turn the sink spout.	1
CoffeePressButton	pressing buttons	Press the button on the coffee machine to pour coffee in to the mug.	1
TurnOnMicrowave	pressing buttons	Turn on the microwave by pressing the start button.	1
TurnOffMicrowave	pressing buttons	Turn off the microwave by pressing the stop button.	1
CoffeeSetupMug	insertion	Pick the mug from the counter and insert it onto the coffee machine mug holder area.	1
CoffeeServeMug	insertion	Remove the mug from the coffee machine mug holder and place it on the counter.	1
NavigateKitchen	navigation	Navigate to a specified appliance in the kitchen.	8

Fig. 13: **Atomic Tasks.**

Task	Activity	Description
PrepareCoffee	Brewing coffee or tea	Place a mug under the coffee machine nozzle and press the start button to make coffee.
DryDishes	Washing dishes	Set bowls and cups on the counter to dry, after they have been washed.
RestockPantry	Restocking kitchen supplies	Group all canned foods together on a shelf in the cabinet, right next to existing canned foods.
ArrangeVegetables	Chopping food	Place vegetables from the sink onto the cutting board on the counter, to prepare for chopping them.
PrepareToast	Making toast	Retrieve bread and jam from cabinets and place them on the counter to prepare for toasting.
ThawInSink	Defrosting food	Pick frozen food and place in sink, then turn on water to thaw the food.
FillKettle	Boiling water	Place an empty kettle from the cabinet and place in the sink to be filled with water.
PrepMarinatingMeat	Meat preparation	Place steak and condiments on cutting board for marination.
DateNite	Setting up the table	Pick up candles and a bottle of wine and place them in the dining area for date night.
CondimentCollection	Clearing the table	Clear condiments from the counter and put them in a cabinet.
FetchCleaningMaterials	Sanitizing	Organize cleaning supplies on the counter for easy access.
MakeFruitBowl	Snack preparation	Gather various fruits into a bowl for serving.
DrawerSort	Tidying cabinets and drawers	Pick and place or push objects into different drawers according to object category.
TransportAndWashFruits	Washing fruits and vegetables	Gather fruits in a bowl, then transport the bowl to the sink and turn on the water.
SetupFrying	Frying	Retrieve a pan and transport it to the stove, then turn on the stove.
MicrowaveMug	Heating food	Place a mug in the microwave and then turn on the microwave.
SetupJuicing	Mixing and blending	Obtains various fruits and set them up for blending.
OrganizeBakingIngredients	Baking	Obtain dairy ingredients and place them together for baking.
PlaceFoodInBowls	Serving food	Arrange bowls on the counter and place an assortment of vegetable in each bowl.
SteamInMicrowave	Steaming vegetables	Put a bowl of vegetables inside the microwave to steam them there.

Fig. 14: **Representative Composite Tasks Across 20 Kitchen Activites.**