Manipulate-Anything: Automating Real-World Robots using Vision-Language Models

Jiafei Duan [1*] **Wentao Yuan** [1*] **Wilbert Pumacay** [2] **Yi Ru Wang** [1]
Kiana Ehsani [3] **Dieter Fox** [1,4] **Ranjay Krishna** [1,3]
[1]University of Washington [2]Universidad Católica San Pablo
[3]Allen Institute for Artificial Intelligence [4]NVIDIA

Abstract: Large-scale endeavors like RT-1[1] and widespread community efforts such as Open-X-Embodiment [2] have contributed to growing the scale of robot demonstration data. However, there is still an opportunity to improve the quality, quantity, and diversity of robot demonstration data. Although vision-language models have been shown to automatically generate demonstration data, their utility has been limited to environments with privileged state information, they require hand-designed skills, and are limited to interactions with few object instances. We propose MANIPULATE-ANYTHING, a scalable automated generation method for real-world robotic manipulation. Unlike prior work, our method can operate in real-world environments without any privileged state information, hand-designed skills, and can manipulate any static object. We evaluate our method using two setups. First, MANIPULATE-ANYTHING successfully generates trajectories for all 7 real-world and 14 simulation tasks, significantly outperforming existing methods like VoxPoser. Second, MANIPULATE-ANYTHING's demonstrations can train more robust behavior cloning policies than training with human demonstrations, or from data generated by VoxPoser [3], Scaling-up [4] and Code-As-Policies [5]. We believe MANIPULATE-ANYTHING can be a scalable method for both generating data for robotics and solving novel tasks in a zero-shot setting. Project page: robot-ma.github.io.

Keywords: Zero-shot manipulation, multimodal language models, multiview state verification, robot skill generation, behavior cloning, robotic manipulation

1 Introduction

The success of modern machine learning systems fundamentally relies on the *quantity* [6, 7, 8, 9, 10, 11], *quality* [12, 13, 14, 15, 16], and *diversity* [17, 18, 19, 20, 21] of the data they are trained on. The availability of large-scale internet data made possible significant advances in vision and language [22, 23, 24]. However, the dearth of data has prevented similar advancements in robotics. Human demonstration collection methods do not scale to sufficient *quantity* or *diversity*. Projects like RT-1 [1] demonstrated the utility of high-*quality* human data collected over 17 months. Others have developed low-cost hardware for data collection [25, 26, 27]. However, all these procedures require expensive human data collection.

Automated data collection methods do not scale to sufficient *diversity*. With the advent of vision-language models (VLMs), the robotics community has been abuzz with new systems that leverage VLMs to guide robotic behavior [28, 5, 29, 30, 3, 4, 31]. In these systems, VLMs decompose tasks into language plans [28, 5] or generate code to execute predefined skills [32, 3]. Though successful in simulation, these methods underperform in the real world [32, 3]. Some methods rely on privileged state information only available in simulation [29, 4], require hand-designed skills [30], or are also limited to manipulating a fixed set of object instances with known geometric shape [3, 32].

*Equal contribution

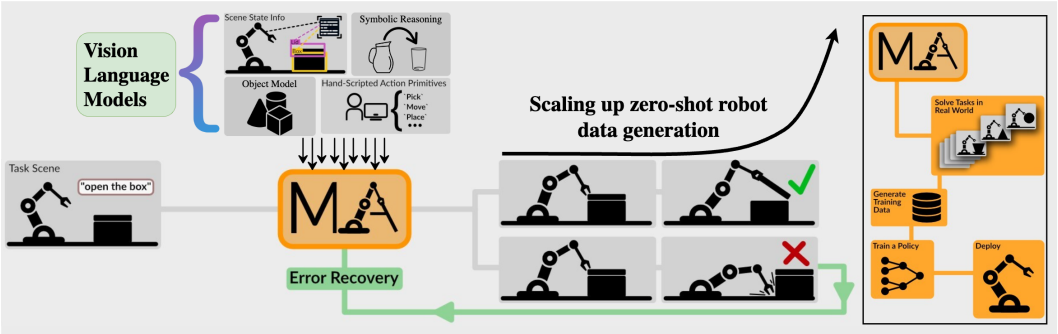

Figure 1: MANIPULATE-ANYTHING is an automated method for robot manipulation in real world environments. Unlike prior methods, it does not require privileged state information, hand-designed skills, or limited to manipulating a fixed number of object instances. It can guide a robot to accomplish a diverse set of unseen tasks, manipulating diverse objects. Furthermore, the generated data enables training behavior cloning policies that outperform training with human demonstrations.

As VLMs improve in performance, and with the vast common-sense knowledge they have shown to possess, could we harvest their capabilities for diverse task completion and scalable data generation? The answer is yes – with careful system design and the right set of input and output formulations, we can not only use VLMs as a means to successfully perform *diverse* tasks in a zero-shot manner, but also generate *quality* data at a high *quantity* to train behaviour cloning policies.

We propose MANIPULATE-ANYTHING a scalable automated demonstration generation method for real-world robotic manipulation. MANIPULATE-ANYTHING produces high *quality* data, at large-*quantities* (if needed), and can manipulate a *diverse* set of objects to perform a *diverse* set of tasks. When placed in a real world environment and given a task (e.g., "open the top drawer" in Figure 2), MANIPULATE-ANYTHING effectively leverages VLMs to guide a robotic arm to complete the task. Unlike prior methods, it doesn't need privileged state information, hand-designed skills, or limited to specific object instances. Not relying on privileged information makes MANIPULATE-ANYTHING environment-agnostic. MANIPULATE-ANYTHING plans a sequence of sub-goals and generates actions to execute the sub-goals. It can verify whether the robot succeeded in the sub-goal using a verifier and re-plan from the current state if needed. This error recovery enables mistake identification, re-planning, and recovering from failure. It also injects recovery behavior into the collected demonstrations. We further enhanced the VLMs' capabilities by incorporating reasoning from multi-viewpoints, significantly improving performance.

We showcase the utility of MANIPULATE-ANYTHING through two evaluation setups. First, we show that it can be prompted with a novel, never-before-seen task and complete it in a zero-shot manner. We quantitatively evaluate across 7 real-world and 14 RLBench [33] simulation tasks and demonstrate capabilities across many real-world everyday tasks (refer to supplementary). Our method significantly outperforms VoxPoser [3] in 10/14 simulation tasks for zero-shot evaluation. It also generalizes to tasks where VoxPoser completely fails because of its limitation to specific object instances. Furthermore, we demonstrated that our approach can solve real-world manipulation tasks in a zero-shot manner, achieving a task-averaged success rate of 38.57%. Second, we show that MANIPULATE-ANYTHING can generate useful training data for a behavior cloning policy. We compare MANIPULATE-ANYTHING generated data against ground truth hand-scripted demonstrations as well as against data from VoxPoser[3], Scaling-up [4] and Code-As-Policies [5]. Surprisingly, policies trained on our data outperforms even human hand-scripted data on 5 out of 12 tasks and performs on par for 4 more when evaluated with RVT-2. Meanwhile, the baselines are unable to generate the training data for some of tasks. MANIPULATE-ANYTHING demonstrates the broad possibility of large-scale deployment of robots across unstructured real-world environments. It also highlights its utility as a training data generator, aiding in the crucial goal of scaling up robot demonstration data.

2 Related work

MANIPULATE-ANYTHING enables scaling of robotic manipulation data using VLMs. As such, we review recent efforts in 1) scaling manipulation data, and 2) applications of VLMs in robotics.

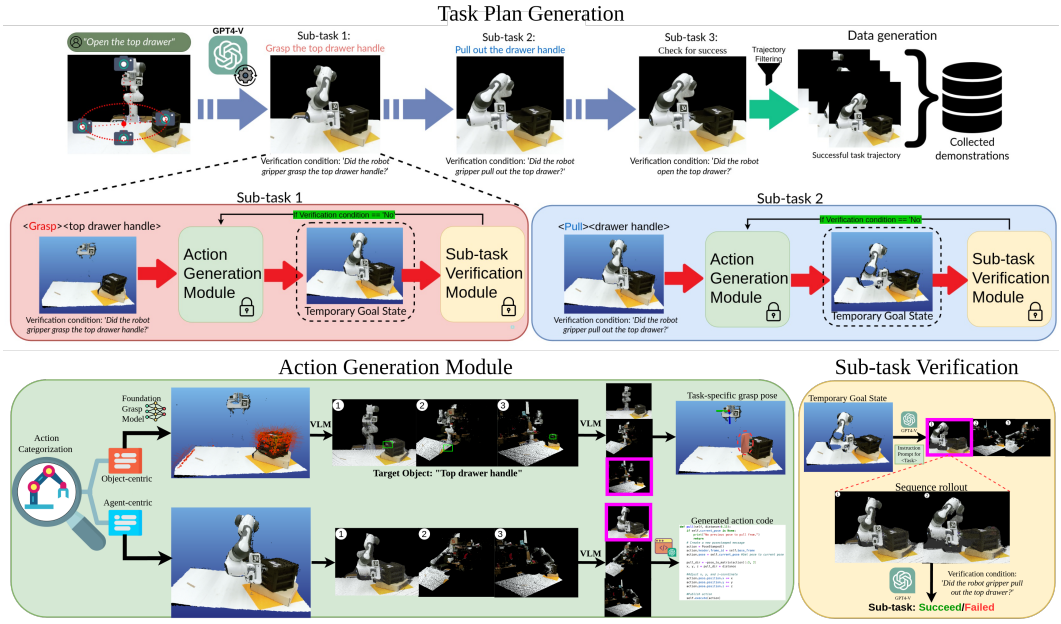

Figure 2: **Manipulate Anything Framework.** The process begins by inputting a scene representation and a natural language task instruction into a VLM, which identifies objects and determines sub-tasks. For each sub-task, we provide multi-view images, verification conditions, and task goals to the action generation module, producing a task-specific grasp pose or action code. This leads to a temporary goal state, assessed by the sub-task verification module for error recovery. Once all sub-tasks are achieved, we filter the trajectories to obtain successful demonstrations for downstream policy training.

Scaling manipulation data. When deploying vision and language-based control policies for real-world applications, a significant challenge revolves around acquiring data. Traditionally, a convenient avenue to collect such trajectories is through human annotations for action (i.e. through teleoperation) and language labeling [34, 35, 36], however, this approach is limited to scale. To address this limitation and achieve autonomous scalability, prior works employ vision-language models or procedural generate language annotations in simulated environments [4, 37, 38]. For action labels, strategies range from random exploration to learned policies [39]. While human egocentric videos are relevant, they lack action labels and require cross-embodiment transfer [40]. Another strategy involves model-based policies, such as task and motion planning (TAMP) [41]. Our approach extends these methods by incorporating common-sense knowledge from large language models (LLMs) and vision language models (VLMs), by providing a framework which combines the strengths of VLMs, object pose prediction, and dynamic retry to synthesize demonstrations in simulated and real environments.

Language models for robotics. In the field of robotics, large language models have found diverse applications, including policy learning [42, 43, 44], task and motion planning [45, 46], log summarization [47], policy program synthesis [5], and optimization program generation [28]. Previous research has also explored the physical grounding capabilities of these models [3, 32, 48], while ongoing work investigates their integration with task and motion planners to create expert demonstrations [4]. [36] attempted to collect extensive real-world interaction data, with short-horizon trajectories. [49] proposed a key-point based visual prompting method for real-world manipulation, through predicting affordances and corresponding motions. Our work complements the existing line of works, by leveraging the high-level planning capabilities of language models, scene understanding capabilities of vision language models, and action sampling, to enable synthesis of robot trajectories, which include language, vision, and robot state, given arbitrary tasks and environments.

3 MANIPULATE-ANYTHING

We propose MANIPULATE-ANYTHING, a framework that solves everyday manipulation tasks conditioned on language. Under the hood, MANIPULATE-ANYTHING leverages VLMs to decompose

tasks into sub-tasks, generates code for new skills or task-specific grasp pose, and verifies the success of each sub-task (Figure 3). Note that due to the modularity aspect of our framework, MANIPULATE-ANYTHING will continue to improve as the underlying VLMs continue to improve.

3.1 Task plan generation

MANIPULATE-ANYTHING takes as input any task described by a free-form language instruction, $\mathbf{T}$ (e.g., *'open the top drawer'*). Creating robot trajectories that adheres to $\mathbf{T}$ is challenging due to its potential complexity and ambiguity, requiring a nuanced understanding of the current environment state. Given $\mathbf{T}$, and an image of the scene, we apply a VLM to first identify task-relevant objects in the scene, appending them to a list. Subsequently, we use a LLM along with those information to decompose the main task $\mathbf{T}$ into a series of discrete, smaller sub-tasks, represented as $\mathbf{T}_i$, along with the corresponding verification conditions v_i, where i ranges from 1 to n. For instance, the above task could be decompose into sub-tasks include *'grasp the drawer handle'* or *'pull open the drawer'*, and verification conditions are *'did the robot grasp the handle?'* or *'is the drawer opened?'*. This transforms the instruction $\mathbf{T}$ into a sequence of specific sub-tasks $\{(\mathbf{T}_1, v_1), (\mathbf{T}_2, v_2), \ldots, (\mathbf{T}_n, v_n)\}$. For each sub-task, MANIPULATE-ANYTHING generates desired actions (§ 3.3) and verifies them against the corresponding conditions to ensure successful completion of that sub-task(§ 3.4). This verification step also allows MANIPULATE-ANYTHING to recover from mistakes and attempt again in the case of failure.

3.2 Multi-viewpoint VLM selection

Many prior works that investigated VLM's reasoning capability found that they do not work well given a single viewpoint [50, 51, 52]. In robotic manipulation, we leverage multiple viewpoints from either more than one camera or re-rendering to minimize object occlusion [53, 54]. Leveraging these insights, we proposed a multi-viewpoint selection phase via VLM before using the selected viewpoints for either action generation or sub-task verification for MANIPULATE-ANYTHINGẆe concatenate all available viewpoints from the current observations, either from multiple cameras or re-rendered viewpoints from a single RGB-D, into a single frame. Numbers are annotated on the top left of each concatenated frame to correspond to a specific viewpoint. Using the concatenated multi-viewpoints, we then query the VLM to choose an ideal viewpoint conditioned on the sub-task. Therefore, for both agent-centric and object-centric action generation, we render multiple viewpoints of the scene and query VLMs to choose an ideal viewpoint for either generating actions given the sub-task or verifying if the sub-task verification condition has been met as shown in Figure 3.1

3.3 Action generation module

Given a sub-task, the desired output from the action generation module is a sequence of low-level actions represented as a 6 DoF end-effector pose. The actions can be categorized into two sets: agent-centric or object-centric. Based on the generated sub-task, the LLM planner will classify it as either agent-centric or object-centric actions. For agent-centric actions, it will modify the agent's state; e.g., it can move the robot's end-effector from the current state (e.g., "rotate $90°$"). We first employed our multi-viewpoint selection to sample out the most optimal viewpoints to provide the VLM with along with the in-context learning technique to generate the code for synthesizing the desired motion. Unlike prior methods that use only language models to generate code [5], our approach utilizes VLM to understand and reason about object locations and the scene, which helps to ground the generation in the current state of the scene. This advantage is demonstrated in the ablation studies in the Appendix.

For object-centric actions, it is often used to generate a task-specific grasp pose for grasping a certain object (e.g., "grasp a knife for cutting") or to synthesize a pre-action pose for non-prehensile tasks by allowing the VLM to assign translation offsets to task-specific grasp poses. To obtain an object-centric action for a given sub-task, we first use an **object-agnostic grasp prediction model** [55] to generate all possible 6-DOF grasping poses in the scene. These poses are not conditioned on task specifications and may include invalid grasp poses for the given task. Next, we use the VLM to obtain the bounding box of the target object parts conditioned on the task specification from all available or re-rendered viewpoints (e.g., if the task is "grasp a knife," the VLM will detect the handle

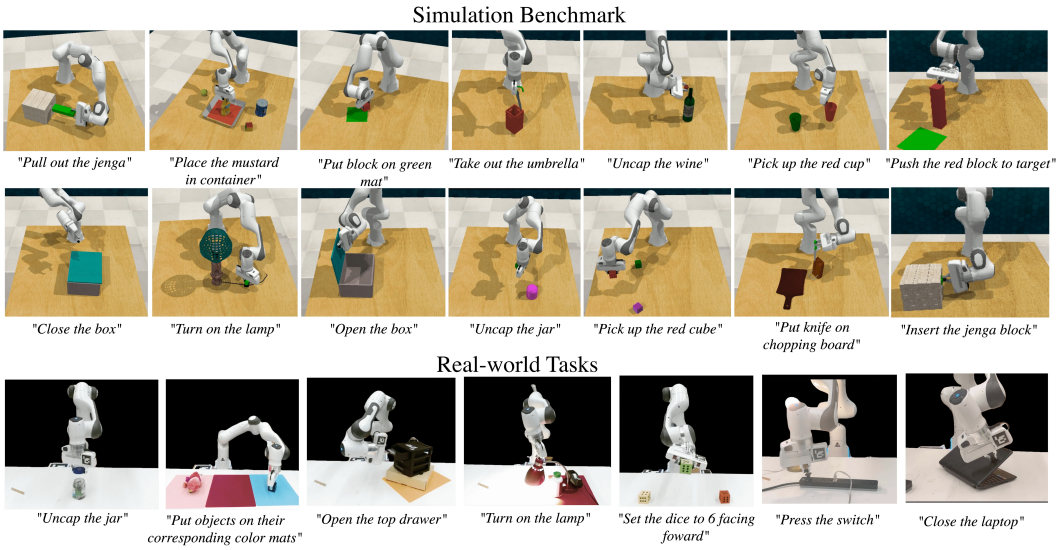

Figure 3: MANIPULATE-ANYTHING is an open-vocabulary autonomous robot demonstration generation system. We show zero-shot demonstrations for 14 tasks in simulation and 7 tasks in the real world.

of the knife and generate a bounding box for the handle). The VLM then performs multi-viewpoint selection to identify the most optimal viewpoint free from occlusion. Finally, using the bounding box detection of the task-specific part of the object from the optimal viewpoint selected by the VLM we filter out a list of proposed candidate grasp poses and select the highest confidence grasp pose. This approach allows us to obtain the most optimal task-specific grasp pose, placement pose, and pre-action pose for non-prehensile tasks, all leveraging the capabilities of the VLM. After the action is generated, a simple motion planner can be used to move the robot to the desired pose as shown in detailed in Fig. 2.

3.4 Sub-task verification

To ensure that each sub-task T_i is executed correctly, we introduce a VLM-based verifier. After every action for each sub-task are executed, we use the VLM to check if the end state matches the verifier condition v_i. Similar to the action generation module, we use **multi-viewpoint VLM selection** to find the optimal view, avoiding errors due to occlusion or ambiguity from a single viewpoint. If the verifier identifies failure, we re-attempt the action generation step for the previous sub-task from the current state. Otherwise, the next sub-task T_{i+1} is attempted. More details of the implementation is in the Appendix.

4 Experiments

Our experiments are designed to address two questions: 1) Can MANIPULATE-ANYTHING accurately solve a diverse set of tasks in a zero-shot manner? 2) Can data generated from MANIPULATE-ANYTHING be used to train a robust policy?

Implementation details. We use both GPT-4V and Qwen-VL [56] as our VLM. We use GPT-4V for task decomposition, action generation, and verification. We use Qwen-VL to detect and extract object information. To ensure zero-shot execution within a reasonable budget, we limit the number of action steps in each trajectory to 50 and the verification module allows a maximum of 30 tries to accomplish a sub-goal. For the task plan generation, we follow the prompting structure adapted from ProgPrompt [28]. All prompts input into the VLM are accompanied by few-shot demonstrations [57]. Additionally, we provide three manually curated primitive action code snippets as examples to prompt the VLM for new action code generation. Full prompts are included in the Appendix. We use four viewpoints $\mathbf{M}_4 = [front, wrist, left\_shoulder, right\_shoulder]$ for the simulation experiments, and re-render three viewpoints for the real-world experiments [53]. For better reasoning by the VLM, we use a resolution of 256×256.

Table 1: **Task-averaged success rate % for zero-shot evaluation.** MANIPULATE-ANYTHING outperformed other baselines in 10 out of 14 simulation tasks from RLBench [33]. Each task was evaluated over 3 seeds to obtain the task-averaged success rate and standard deviations.

Method	Put_block	Play_jenga	Open_jar	Close_box	Open_box	Pickup_cup	Push_block
VoxPoser [3]	70.70±2.31	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	26.70±14.00	**25.33**±8.33
CAP [5]	84.00±16.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	14.67±4.62	8.00±4.00
Scaling-up [4]	77.33±6.11	0.00±0.00	78.67±11.55	0.00±0.00	0.00±0.00	9.33±2.26	5.33±6.11
MA (Ours)	**96.00**±4.00	**77.33**±6.11	**80.00**±4.00	**33.33**±12.86	**29.00**±10.07	**82.67**±14.04	20.00±4.00

Method	Take_umbrella	Sort_mustard	Open_wine	Lamp_on	Put_knife	Pick_&_lift	Insert_block
VoxPoser[3]	33.33±8.33	**96.00**±6.93	8.00±4.00	57.30±12.22	**92.00**±4.00	96.00±0.00	0.00±0.00
CAP[5]	4.00±4.00	0.00±0.00	0.00±0.00	64.00±6.93	14.67±8.33	**100.00**±0.00	0.00±0.00
Scaling-up [4]	6.67±2.31	41.33±12.86	33.33±20.13	60.00±8.00	24.00±0.00	**100.00**±0.00	0.00±0.00
MA (Ours)	**61.33**±20.13	64.00±6.93	**42.00**±4.00	**69.33**±6.11	52.00±10.58	84.00±6.93	**33.33**±4.62

4.1 Zero-shot Performance in Simulation

We empirically study the zero-shot capability of MANIPULATE-ANYTHING in solving 14 diverse tasks in simulation, covering a wide range of task configurations and action primitives for both prehensile and non-prehensile tasks. Our simulation experiments are reported to ensure reproducibility and provide a benchmark for future methods.

Environment and tasks. The simulation is set up in CoppeliaSim and interfaced through PyRep. All simulation experiments use a Franka Panda robot with a parallel gripper. Input observations are captured from four RGB-D cameras positioned around a tabletop setting. We use RLBench [33], a robot learning benchmark with diverse tasks conditioned on language and provided success conditions. We sample 14 tasks from RLBench, covering a diverse range of action primitives, task horizons, and object position perturbations. Each action can be represented as a way-point, and the trajectories are computed and executed via a motion planner using the Open Motion Planning Library [58].

Baselines. We compare against three state-of-the-art zero-shot data generation approaches: Code-as-Policies (CAP) [5], Scaling-up-Distilling-Down (Scaling-up) [4] and VoxPoser [3]. CAP uses language models to generate executable programs that call hand-crafted primitive actions. VoxPoser [3] builds a 3D voxel map of value functions for predicting waypoints. Scaling-up [4] is a Large Language Model (LLM) equipped with a suite of 6 DoF exploration primitives, which it uses within its framework to generate robotic data for distilling a policy. We provided CAP with ground truth simulation state information and object models, and we supplied VoxPoser with ground truth segmented object point clouds from the simulator. This inherently puts MANIPULATE-ANYTHING at a disadvantage in comparison.

Results: MANIPULATE-ANYTHING can generate successful trajectories for all 14 tasks while Scaling-up, VoxPoser and CAP cover only 10, 9 and 7 tasks, respectively (Table 1). Without the privileged state information, the baselines would not succeed on any of the 14 tasks. MANIPULATE-ANYTHING outperforms the baselines in 10 out of the 14 tasks. The three tasks where our method achieves lower performance require fine-grained manipulation of objects, which are the hardest task without the privileged state information used by baselines. VoxPoser fails in the tasks that require moving the arm beyond 4-DoF. MANIPULATE-ANYTHING outperforms the strongest baseline, VoxPoser, by an average task-averaged margin of up to 22%.

4.2 Behavior cloning with demonstrations from MANIPULATE-ANYTHING

Next, we analyze the quality of the generated data by comparing the success rates of behavior cloning models trained with the data. Zero-shot methods like MANIPULATE-ANYTHING are computationally expensive but hold the potential to generate useful training data. To evaluate the quality and effectiveness of the generated training data, we use the methods described in the previous section to generate data for each task. We also compare performance against a model trained on human-generated demonstrations across the 12 tasks. We use the data to train behavior cloning policies.

Table 2: **Behavior Cloning with different generated data.** The behavior cloning policy trained on the data generated by MANIPULATE-ANYTHING provides the best performance on 10 out of 12 tasks compared to the other autonomous data generation baselines. We report the Success Rate % for behaviour cloning policies trained with data generated from VoxPoser [3] and Code as Policies [5] in comparison. Note that the RLBench[33] baseline uses human expert demonstrations and is considered an upper bound for behavior cloning.

Data	Models	Put_block	Play_jenga	Open_jar	Close_box	Open_box	Pickup_cup
VoxPoser[3]	PerAct[34]	2.67±2.31	-	-	-	-	4.00±4.00
CAP[5]	PerAct[34]	6.67±2.31	-	-	-	-	14.67±12.86
Scaling-up [4]	PerAct[34]	22.67±15.14	-	5.33±6.11	-	-	14.67±2.31
MA (Ours)	PerAct[34]	**85.33**±10.07	81.33±2.31	21.33±10.07	42.67±8.33	**30.67**±11.55	54.00±12.49
RLBench[33]	PerAct[34]	20.00±18.33	**81.33**±9.24	**58.67**±45.49	**68.00**±24.98	14.67±6.11	**54.67**±23.09
VoxPoser[3]	RVT-2[59]	73.33±2.31	-	-	-	-	2.67±2.31
CAP[5]	RVT-2[59]	78.66±8.32	-	-	-	-	77.33±19.73
Scaling-up [4]	RVT-2[59]	38.67±2.31	-	33.33±2.31	-	-	92.00±4.00
MA (Ours)	RVT-2[59]	85.33±2.31	82.67±2.31	78.67±10.06	**82.67**±2.31	**24.00**±4.00	97.33±2.31
RLBench[33]	RVT-2[59]	**86.67**±2.31	**85.33**±2.31	**81.33**±6.11	76.00±4.00	4.00±4.00	**97.33**±2.31

Data	Models	Take_umbrella	Sort_mustard	Open_wine	Lamp_on	Put_knife	Pick_&_lift
VoxPoser[3]	PerAct[34]	4.00±4.00	0.00±0.00	1.33±2.31	5.33±4.62	1.33±2.31	5.67±1.64
CAP[5]	PerAct[34]	13.33±10.06	-	-	8.00±16.00	9.33±6.11	46.67±2.31
Scaling-up [4]	PerAct[34]	4.00±4.00	0.00±0.00	81.33±12.86	76.00±4.00	5.33±2.31	53.33±10.06
MA (Ours)	PerAct[34]	**84.00**±6.93	53.33±6.11	86.67±6.11	**89.33**±6.11	8.00±4.00	33.33±2.31
RLBench[33]	PerAct[34]	58.67±50.80	**53.33**±34.02	**86.67**±12.86	84.00±13.86	**30.67**±10.07	**62.67**±9.24
VoxPoser[3]	RVT-2[59]	5.33±6.11	1.33±2.31	1.33±2.31	2.67±2.31	1.33±2.31	17.33±2.31
CAP[5]	RVT-2[59]	89.33±6.11	-	-	85.33±8.32	52.00±10.58	82.66±20.53
Scaling-up [4]	RVT-2[59]	94.67±4.62	24.00±4.00	62.67±2.31	21.33±2.31	53.33±2.31	80.00±6.93
MA (Ours)	RVT-2[59]	94.67±2.31	**73.33**±2.31	**93.33**±6.11	84.00±10.58	69.33±12.85	**82.67**±12.22
RLBench[33]	RVT-2[59]	**97.33**±2.31	69.33±8.33	88.00±8.00	**93.33**±4.62	**72.00**±10.58	64.00±10.58

Data generation details. We generate 10 successful demonstrations per task. We use the system's success condition to filter for successful demonstrations. Each of the demonstrations consist of a language instruction, RGB-D frames for the trajectory, and waypoints represented as 6 DoF gripper poses and states. For the tasks that the baselines were unable to generate any successful demonstrations, we patched the missing training data with RLBench system-generated demonstrations.

Training and evaluation protocol. We train two models using the generated demonstrations: the Perceiver-Actor (PerAct) model [34], a transformer-based robotic manipulation behavior cloning model that expects tokenized voxel grids and language instructions as inputs and predicts discretized voxel grid 6 DoF poses and gripper states, and also RVT-2 model [59], a multi-view transformer-based BC model. The RVT-2 model uses tokenized image patches and CLIP-encoded language instructions as input to predict keypoint actions as translation heatmaps, discretized rotation in Euler angles, and the gripper's binary state. Notably, RVT-2 is currently the highest-performing model on the RLBench benchmark. For all the generated training datasets, we train a multi-task PerAct policy with a batch size of 4 for 30k iterations on a single RTX A100, and RVT-2 with a batch size of 24 for 3.3k iterations on 4 A100s. To ensure consistent evaluation, we generate one set of testing environments with RLBench. We evaluate the last checkpoint from each of the trained policies. Each policy is evaluated for 25 episodes across each task using 3 different seeds. We measure the success rate based on the simulation-defined success condition.

Results: Policies trained using MANIPULATE-ANYTHING data perform similarly to policies trained using hand-scripted demonstrations ($p = 0.973$) for PerAct (Table 2). Training on either MANIPULATE-ANYTHING or on hand-scripted demonstrations results in a performance difference of a mere 0.27% across all tasks. Furthermore, models trained on data from the baselines exhibit a statistically lower performance ($p \leq 0.01$ for VoxPoser, Scaling-up and CAP). One of the main

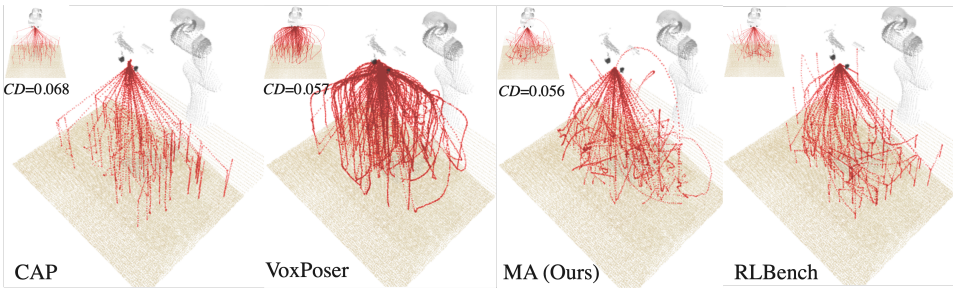

Figure 5: **Action Distribution for Generated Data:** We compare the action distribution of data generated by various methods against human-generated demonstrations via RLBench on the same set of tasks. We observed a high similarity between the distribution of our generated data and the human-generated data. This is further supported by the computed CD between our methods and the RLBench data, which yields the lowest (CD=0.056).

factors potentially contributing to the differences in the performance could be that MANIPULATE-ANYTHING generates diverse expert trajectories that are preferable to humans. This can be seen in Fig. 5, which shows the action distribution of the generated data by different methods for the same given tasks. Additionally, our generated data recorded the lowest Chamfer Distance (CD) of 0.056 with human-generated demonstrations data. We also observed that the policy trained on MA data achieves a lower standard deviation of 3.39 across all tasks, compared to the zero-shot performance standard deviation of 8.48. This suggests the benefits of training over generated data instead of relying solely on zero-shot deployment.

4.3 Real-world experiments

Finally, we evaluate MANIPULATE-ANYTHING in the real world. We also automatically generate real-world demonstrations for training PerAct.

Environment and tasks. We employ a Franka Panda manipulator equipped with a parallel gripper. We use a front-facing Kinect 2 RGB-D camera. To generate multi-view inputs for the MANIPULATE-ANYTHING framework, we re-render virtual viewpoints from the generated point cloud, similar to prior work [53]. We selected 7 representative real-world tasks, both prehensile and non-prehensile: `open_jar`, `sort_objects`, `correct_dices`, `open_drawer`, `on_lamp`, `press_switch`, and `close_laptop`, all conditioned on language instructions. Each task was evaluated over 10 episodes with varying object poses across 3 trials.

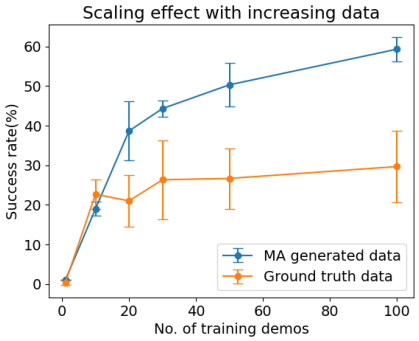

Figure 4: **Scaling experiment.** Scaling effect of model performance with increasing training demonstrations.

Data generation details. We used MANIPULATE-ANYTHING to generate 6 demonstrations for each task and manually perform scene resets when failures occur. We train a similar multi-task PerAct for 120k iterations and evaluate the trained policies in a manner similar to the zero-shot experiments.

Results: MANIPULATE-ANYTHING is able to generate successful demonstrations for each of the 7 real world tasks. Even for the worst-performing task, MANIPULATE-ANYTHING achieves a success rate of more than 25%. Our approach outperforms **CAP** by 38%. Consistent with the simulation results, **training with the data generated by MANIPULATE-ANYTHING produces a more robust policy** compared to performing zero-shot. Additionally, in 4 out of 5 tasks, the trained policies perform better than the zero-shot approach. The policy under-performs on the `sort_object` task, because it requires longer-horizon memory —a known limitation pointed out in PerAct [34].

4.4 Ablations

Table 3: **Real-world Results.** The model trained on the data generated by our model in the real world (no expert in the loop) demonstrates on par results with the model trained on human expert collected data. We present a comparison of success rates for task completion in a zero-shot manner (Code as Policies [5] and MANIPULATE-ANYTHING), and using trained policies from MANIPULATE-ANYTHING data and human expert data.

	Open_drawer	Sort_object	On_lamp	Open_jar	Correct_dice	Press_switch	Close_laptop
CAP (0-shot)	0.00 ± 0.00	13.33 ± 5.77	0.00 ± 0.00	6.67 ± 5.77	6.67 ± 5.77	0.00 ± 0.00	0.00 ± 0.00
MA (0-shot)	**36.67**± 5.77	**60.00**± 10.00	**26.67**± 11.55	**40.00**± 10.00	**53.33**± 5.77	**20.00**± 10.00	**33.33**± 5.77
PerAct (MA data)	50.00 ± 0.00	33.33 ± 5.77	50.00 ± 0.00	56.67 ± 5.77	60.00 ± 0.00	56.67 ± 5.77	33.33 ± 5.77
PerAct (Human data)	**53.33**± 11.55	**36.67**± 5.77	**60.00**± 0.00	**76.67**± 5.77	**80.00**± 10.00	33.33 ± 5.77	**53.33**± 5.77

For effective real-world deployment of MANIPULATE-ANYTHING, it's crucial that the collected data supports scaling of robotics transformers and offers diverse skills and interacted objects. We conducted an ablation study to evaluate the quality of MANIPULATE-ANYTHING-generated data for scaling and its generalization to language instruction changes. For scaling, we generated behavior cloning data, ranging from 1 to 100 training demonstrations from RLBench and MANIPULATE-ANYTHING for a single task (`put_block`), and trained a PerAct policy. For generalization, we varied the `sort_mustard` task with different language instructions and target objects. We compared our approach to VoxPoser to assess robustness to object and language instruction changes. Further implementation details are in the supplementary materials.

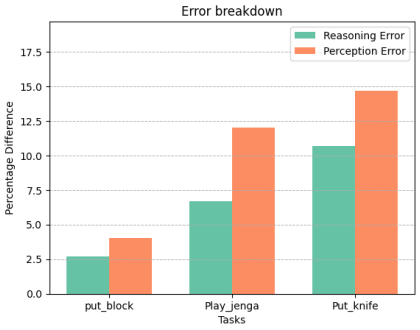

Figure 6: **Error breakdown.** Error breakdown for three tasks from simulation.

Result: Our scaling experiments demonstrate that generating more training data via MANIPULATE-ANYTHING improves PerAct policy performance (Fig. 6). The data from our approach shows a better rate of change with a slope of 0.503 for a linear fit, compared to 0.197 for RLBench-generated data. Additionally, MANIPULATE-ANYTHING data is more generalizable and robust to language instruction changes, outperforming VoxPoser in task success across language and object variations. Detailed results in the appendix.

4.5 Error breakdown

We examine the potential errors introduced by each respective VLM and explore ways to further improve the overall system. We conducted simulation experiments where tasks could be easily reset, focusing on two major components where VLMs make decisions: "Perception error" and "Reasoning error." "Perception error" refers to mistakes made by the VLM in detecting target objects and selecting optimal viewpoints for generating task-specific grasp pose, primarily related to the selection of the Multi-viewpoint VLM. "Reasoning error" involves the Sub-task Verification Module, where the VLM is responsible for deciding if sub-tasks have been successfully completed. Due to resource constraints, we conducted experiments on the `play_jenga` task. We systematically replaced the VLMs in each component with a human, allowing the human to make decisions. By comparing the system's performance with human decision-making to that with VLMs, we were able to quantify the errors caused by the VLMs.

5 Discussion

Limitations. Despite the promising results, MANIPULATE-ANYTHING has several limitations. First, MANIPULATE-ANYTHING depends on the availability of large language models (LLMs), which introduces reliance on foundational models. However, with the ongoing development of open-source LLMs, this issue is likely to diminish over time. Second, MANIPULATE-ANYTHING struggles with dynamic manipulation tasks and non-prehensile tasks, as it cannot generate alternative grasp poses for these scenarios. Third, the system's highly modular nature, integrating multiple VLMs, can lead to compounding errors when generating zero-shot trajectories. Emerging specialized VLMs [51]

and others which may help address this issue. Finally, manual prompt engineering for in-context learning is still required. Nonetheless, recent advancements in alignment and prompting techniques [60, 61, 62] offer potential solutions to reduce the effort involved in prompt engineering.

Conclusion. MANIPULATE-ANYTHING is a scalable environment-agnostic approach for generating zero-shot demonstration for robotic tasks without the use of privileged environment information. MANIPULATE-ANYTHING uses LLMs to do high level planning and scene understanding and is capable of error recovery. This enables high quality data generation for behavior cloning that can achieve better performance that using human data.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[2] O. X.-E. Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[3] 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.

[4] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. *arXiv preprint arXiv:2307.14535*, 2023.

[5] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[6] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[7] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, C. Schuhmann, L. Schmidt, and J. Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.

[8] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[9] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

[10] V. Ramanujan, T. Nguyen, S. Oh, A. Farhadi, and L. Schmidt. On the connection between pre-training data diversity and fine-tuning robustness. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] V. Udandarao, A. Prabhu, A. Ghosh, Y. Sharma, P. H. Torr, A. Bibi, S. Albanie, and M. Bethge. No" zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance. *arXiv preprint arXiv:2404.04125*, 2024.

[12] S. Y. Gadre, G. Ilharco, A. Fang, J. Hayase, G. Smyrnis, T. Nguyen, R. Marten, M. Wortsman, D. Ghosh, J. Zhang, et al. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] T. Nguyen, S. Y. Gadre, G. Ilharco, S. Oh, and L. Schmidt. Improving multimodal datasets with image captioning. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] T. Nguyen, G. Ilharco, M. Wortsman, S. Oh, and L. Schmidt. Quality not quantity: On the interaction between dataset design and robustness of clip. *Advances in Neural Information Processing Systems*, 35:21455–21469, 2022.

[16] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[17] A. Fang, G. Ilharco, M. Wortsman, Y. Wan, V. Shankar, A. Dave, and L. Schmidt. Data determines distributional robustness in contrastive language image pre-training (clip). In *International Conference on Machine Learning*, pages 6216–6234. PMLR, 2022.

[18] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[19] Y. Tian, L. Fan, K. Chen, D. Katabi, D. Krishnan, and P. Isola. Learning vision from models rivals learning vision from data. *arXiv preprint arXiv:2312.17742*, 2023.

[20] H. Xu, S. Xie, X. E. Tan, P.-Y. Huang, R. Howes, V. Sharma, S.-W. Li, G. Ghosh, L. Zettlemoyer, and C. Feichtenhofer. Demystifying clip data. *arXiv preprint arXiv:2309.16671*, 2023.

[21] Z. Chen, A. H. Cano, A. Romanou, A. Bonnet, K. Matoba, F. Salvi, M. Pagliardini, S. Fan, A. Köpf, A. Mohtashami, et al. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.16079*, 2023.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[24] C. Schuhmann, R. Beaumont, R. Vencu, C. W. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. R. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=M3Y74vmsMcY.

[25] 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.

[26] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[27] J. Duan, Y. R. Wang, M. Shridhar, D. Fox, and R. Krishna. Ar2-d2: Training a robot without a robot. *arXiv preprint arXiv:2306.13818*, 2023.

[28] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[29] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang. Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*, 2023.

[30] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Z. Erickson, D. Held, and C. Gan. Genbot: Generative simulation empowers automated robotic skill learning at scale. *arXiv*, 2023.

[31] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, Q. Vuong, T. Zhang, T.-W. E. Lee, K.-H. Lee, P. Xu, S. Kirmani, Y. Zhu, A. Zeng, K. Hausman, N. Heess, C. Finn, S. Levine, and B. Ichter. Pivot: Iterative visual prompting elicits actionable knowledge for vlms, 2024.

[32] 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.

[33] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[34] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[35] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[36] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

[37] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, et al. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[38] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, et al. Autort: Embodied foundation models for large scale orchestration of robotic agents. *arXiv preprint arXiv:2401.12963*, 2024.

[39] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[40] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[41] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[42] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023.

[43] J. Zhang, K. Pertsch, J. Zhang, T. Nam, S. J. Hwang, X. Ren, and J. J. Lim. Sprint: Scalable semantic policy pre-training via language instruction relabeling. 2022.

[44] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*, 2024.

[45] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans, 2023.

[46] 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.

[47] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction, 2023.

[48] 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. *arXiv preprint arXiv:2310.10021*, 2023.

[49] F. Liu, K. Fang, P. Abbeel, and S. Levine. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting. *arXiv preprint arXiv:2403.03174*, 2024.

[50] B. Liu, Y. Dong, Y. Wang, Y. Rao, Y. Tang, W.-C. Ma, and R. Krishna. Coarse correspondence elicit 3d spacetime understanding in multimodal language model. *arXiv preprint arXiv:2408.00754*, 2024.

[51] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[52] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, and C. Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36: 20482–20494, 2023.

[53] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[54] Y. R. Wang, Y. Zhao, H. Xu, S. Eppel, A. Aspuru-Guzik, F. Shkurti, and A. Garg. Mvtrans: Multi-view perception of transparent objects. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3771–3778. IEEE, 2023.

[55] W. Yuan, A. Murali, A. Mousavian, and D. Fox. M2t2: Multi-task masked transformer for object-centric pick and place, 2023.

[56] J. Bai, S. Bai, S. Yang, S. Wang, S. Tan, P. Wang, J. Lin, C. Zhou, and J. Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

[57] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[58] I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[59] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

[60] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[61] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[62] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.